COURSE NOTES

Optimal Control

Compiled on Sunday 19th July, 2026
Last repo commit: Git metadata unavailable

Note: Interactive elements may be simplified or omitted in this PDF export.

Advice to Students

Optimal control is learned by moving repeatedly among a physical system, a mathematical formulation, necessary conditions, numerical computation, and independent verification.

For each problem:

1. identify states, controls, parameters, information, and units;
2. state the objective, dynamics, boundary conditions, and path constraints;
3. predict the qualitative structure of a good solution;
4. choose a solution method that matches the problem structure;
5. inspect feasibility, optimality residuals, scaling, and discretization error;
6. re-simulate the control with an independent dynamics solver; and
7. explain what the solution means physically and what assumptions limit it.

A solver's success message is evidence about an algorithm, not by itself evidence that the engineering conclusion is correct.

Chapter 1: Optimization Foundations and Variational Methods

Chapter 1: Optimization Foundations and Variational Methods

From finite-dimensional decisions to variations of entire trajectories

Optimal control begins with a change in what is being chosen: the decision variable grows from a number, to a vector, and finally to an entire trajectory. This chapter follows that progression deliberately. It starts with finite-dimensional optimization, develops functionals and first variations, and then shows how moving or free endpoints generate transversality and natural boundary conditions. These ideas supply the variational language used throughout the rest of the book.

Chapter contents

From Optimization to Optimal Control

Optimal control is concerned with choosing the best possible evolution of a dynamical system. The word *best* must always be interpreted relative to a clearly defined performance measure. Depending on the application, one may wish to minimize travel time, energy consumption, fuel use, tracking error, vibration, or economic cost. In other applications, the objective may be to maximize profit, harvested energy, range, or reliability.

The central ideas of optimal control are easiest to understand by first returning to ordinary scalar optimization. In scalar optimization, the unknown is a number. In optimal control, the unknown is an entire function of time. The mathematical transition from optimizing a number to optimizing a function is the key conceptual step developed in this section.

Optimization as Efficient Decision Making

Optimization appears whenever a task can be accomplished in more than one way and some alternatives are preferred over others. Everyday scheduling, route planning, engineering design, and resource allocation all possess the same basic structure:

1. there is a decision to be made;
2. there is a goal or objective;
3. there are constraints that restrict what is possible;
4. there may be many feasible alternatives;
5. one seeks the alternative that performs best according to a chosen metric.

> **Core idea**
> An optimization problem is not defined by the goal alone. It is defined jointly by the *decision variables*, the *objective function*, and the *constraints*.

For example, suppose a person must complete grocery shopping. The task can be completed by many different routes through the store. One route may minimize time, while another may minimize walking distance or physical effort. The optimal route therefore depends on the chosen objective.

The Elements of an Optimization Problem

A generic finite-dimensional optimization problem can be written as

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} f(\boldsymbol{x})$$

subject to

$$\begin{aligned} g_i(\boldsymbol{x}) &\le 0, \qquad i = 1, \dots, m, \\ h_j(\boldsymbol{x}) &= 0, \qquad j = 1, \dots, p. \end{aligned}$$

Here:

- $\boldsymbol{x} \in \mathbb{R}^n$ is the vector of decision variables;
- $f : \mathbb{R}^n \to \mathbb{R}$ is the objective function;
- g_i are inequality constraints;

- h_j are equality constraints.

The set of all points satisfying the constraints is called the feasible set:

$$\mathcal{F} = \left\{ \boldsymbol{x} \in \mathbb{R}^n : g_i(\boldsymbol{x}) \leq 0,\ h_j(\boldsymbol{x}) = 0 \right\}.$$

Feasible point
A point $\boldsymbol{x} \in \mathbb{R}^n$ is feasible if it satisfies every equality and inequality constraint.

Infeasible problem
An optimization problem is infeasible if its feasible set is empty:

$$\mathcal{F} = \varnothing.$$

The grocery-shopping example illustrates infeasibility. If the complete task must be finished in two minutes, no physically realizable route may satisfy the time constraint. If the allowed time is increased to three hours, the feasible set may become nonempty.

The Objective Function and the Meaning of "Optimal"

The objective function is the numerical quantity used to compare feasible solutions. For a minimization problem, a feasible point $\boldsymbol{x}^*$ is globally optimal if

$$f(\boldsymbol{x}^*) \leq f(\boldsymbol{x}) \qquad \forall \boldsymbol{x} \in \mathcal{F}.$$

For a maximization problem, $\boldsymbol{x}^*$ is globally optimal if

$$f(\boldsymbol{x}^*) \geq f(\boldsymbol{x}) \qquad \forall \boldsymbol{x} \in \mathcal{F}.$$

Different objective functions generally produce different optimal solutions. For the same physical task, minimizing elapsed time may lead to a different decision than minimizing energy expenditure.

Minimum-time versus minimum-energy shopping
Let r denote a route through a store. Suppose

$$T[r] = \text{time required by route } r,$$

and

$$E[r] = \text{energy expended along route } r.$$

Then

$$r_T^* = \arg\min_r T[r]$$

need not equal

$$r_E^* = \arg\min_r E[r].$$

Both solutions may be optimal, but they are optimal relative to different performance measures.

Scalar Functions

The simplest optimization setting is a scalar-valued function of a scalar variable:

$$f : \mathbb{R} \to \mathbb{R}, \qquad y = f(x).$$

The input x is a real number, and the output $f(x)$ is also a real number. For a smooth function, classical calculus provides useful necessary and sufficient conditions for identifying local minima and maxima.

Critical Points and the First-Derivative Condition

Suppose f is differentiable. At an interior local minimum or maximum, the slope must vanish under standard regularity assumptions:

$$\frac{df}{dx}(x^*) = 0.$$

Critical point
A point x_c is called a critical point of f if

$$f'(x_c) = 0$$

or if the derivative does not exist.

For the smooth functions considered here, critical points are obtained by solving

$$f'(x) = 0.$$

Let

$$x_1, x_2, \ldots, x_N$$

denote all roots of this equation. These points are only *candidates* for extrema. A zero derivative does not by itself prove that a point is a minimum or maximum.

Why the first derivative vanishes

If x^* is an interior local minimum and f is differentiable, then for sufficiently small h,

$$f(x^* + h) \geq f(x^*).$$

For $h > 0$,

$$\frac{f(x^* + h) - f(x^*)}{h} \geq 0,$$

while for $h < 0$,

$$\frac{f(x^* + h) - f(x^*)}{h} \leq 0.$$

Taking the limit $h \to 0$ from the right and left implies

$$f'(x^*) = 0.$$

This result is a necessary condition for a smooth interior optimum, but it is not sufficient.

The Second-Derivative Test

The second derivative describes local curvature.

Second-derivative test
Let x_c satisfy $f'(x_c) = 0$. Then:

1. if

$$f''(x_c) > 0,$$

then x_c is a strict local minimum;

2. if

$$f''(x_c) < 0,$$

then x_c is a strict local maximum;

3. if

$$f''(x_c) = 0,$$

then the test is inconclusive.

The interpretation follows from a second-order Taylor expansion:

$$f(x_c + h) = f(x_c) + f'(x_c)h + \frac{1}{2}f''(x_c)h^2 + o(h^2).$$

Because $f'(x_c) = 0$,

$$f(x_c + h) - f(x_c) = \frac{1}{2}f''(x_c)h^2 + o(h^2).$$

Since $h^2 \geq 0$, the sign of $f''(x_c)$ determines whether nearby function values are larger or smaller.

Stationary Points That Are Not Extrema

A stationary point need not be a minimum or maximum.
Consider

$$f(x) = x^3.$$

Then

$$f'(x) = 3x^2,$$

so

$$f'(0) = 0.$$

However, $x = 0$ is neither a local minimum nor a local maximum, because $f(x) < 0$ for $x < 0$ and $f(x) > 0$ for $x > 0$.

Remark
For a scalar function, the point $x = 0$ in $f(x) = x^3$ is more precisely a stationary inflection point. The term *saddle point* is most commonly reserved for multivariable functions.

Local and Global Extrema

Local minimum
A point x^* is a local minimum if there exists $\delta > 0$ such that

$$f(x^*) \leq f(x) \qquad \forall x \in (x^* - \delta, x^* + \delta).$$

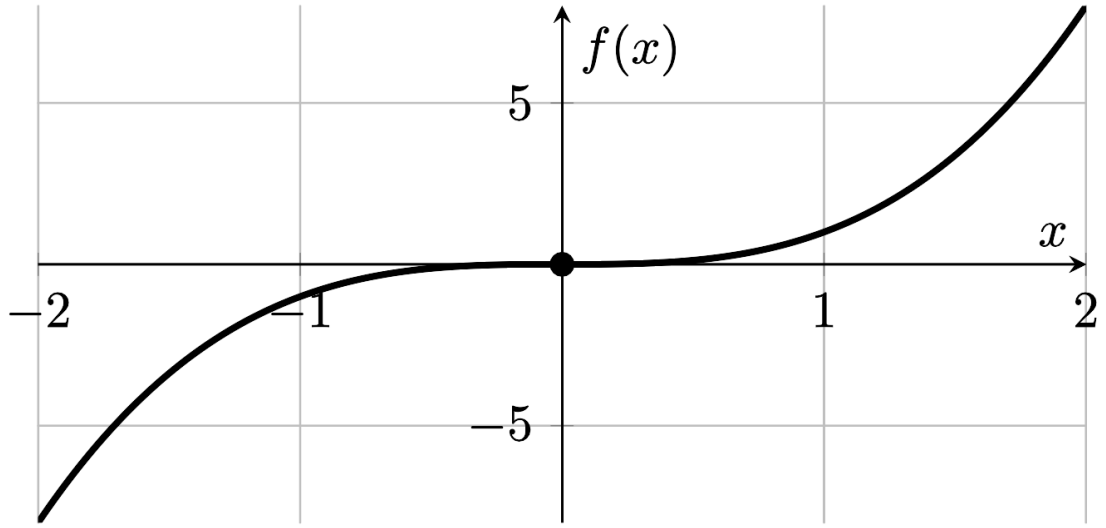

Figure 1: A stationary inflection point: $f'(0) = 0$, but $x = 0$ is neither a local minimum nor a local maximum.

Local maximum
A point x^* is a local maximum if there exists $\delta > 0$ such that

$$f(x^*) \geq f(x) \qquad \forall x \in (x^* - \delta, x^* + \delta).$$

Global minimum
A point x^* is a global minimum on a domain $\mathcal{D}$ if

$$f(x^*) \leq f(x) \qquad \forall x \in \mathcal{D}.$$

A local optimum is best only in a neighborhood. A global optimum is best over the entire admissible domain.

For a smooth function with finitely many local maxima $x_1, \ldots, x_k$, the global maximum among those candidates is found by comparing their objective values:

$$x^* = \underset{x_i,\ i=1,\ldots,k}{\arg\max}\ f(x_i).$$

Constrained Scalar Optimization

Now consider

$$\max_{x \in [a,b]} f(x).$$

The feasible set is the closed interval

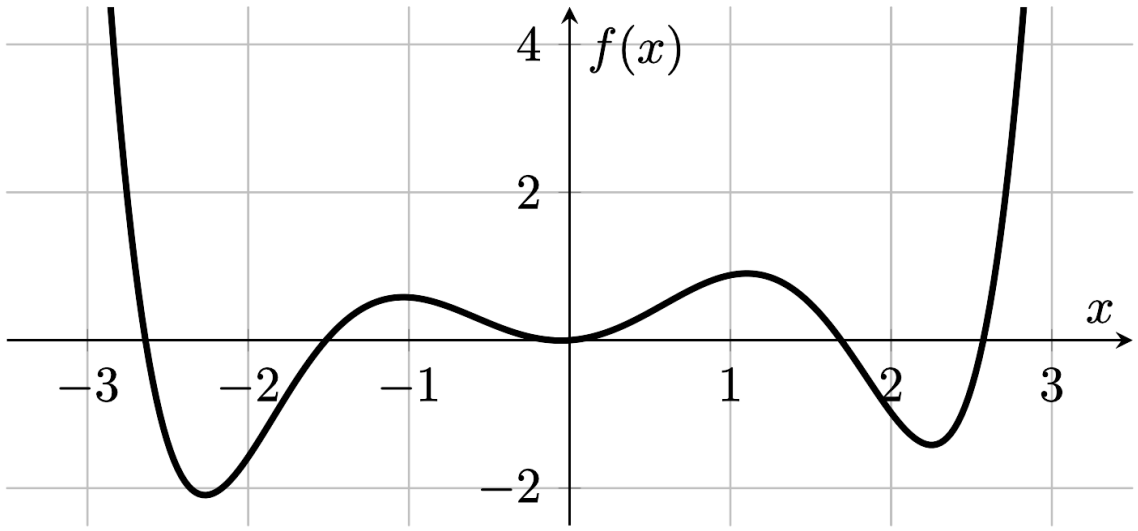

Figure 2: A smooth function may have several local extrema. Only one may be globally optimal.

$$\mathcal{F} = [a, b].$$

A crucial point is that the optimizer may lie at a boundary:

$$x^* = a \qquad \text{or} \qquad x^* = b.$$

At a boundary optimum, the derivative need not be zero.

Why the derivative test can fail

Consider

$$f(x) = x, \qquad x \in [0, 1].$$

Then

$$f'(x) = 1$$

for every $x \in [0, 1]$. There is no solution of $f'(x) = 0$, yet the maximum is clearly

$$x^* = 1, \qquad f(x^*) = 1.$$

Thus, the equation

$$f'(x) = 0$$

detects only smooth interior stationary points. It does not detect active-boundary optima.

A Systematic Procedure on a Closed Interval

If f is continuous on $[a, b]$, the extreme value theorem guarantees that f attains both a global minimum and a global maximum on $[a, b]$.

A systematic procedure is:

1. solve

$$f'(x) = 0$$

for all critical points in (a, b);

2. evaluate f at every interior critical point;
3. evaluate the endpoints:

$$f(a), \qquad f(b);$$

4. compare all candidate values.

Therefore,

$$\max_{x \in [a,b]} f(x) = \max \{f(a), f(b), f(x_1), \ldots, f(x_N)\},$$

where $x_1, \ldots, x_N$ are the critical points in (a, b).

Boundary maximum
Let

$$f(x) = x^2 - 2x, \qquad x \in [0, 3].$$

The derivative is

$$f'(x) = 2x - 2,$$

so the only interior critical point is

$$x = 1.$$

The candidate values are

$$f(0) = 0,$$
$$f(1) = -1,$$
$$f(3) = 3.$$

Hence

$$\arg\max_{x\in[0,3]} f(x) = 3,$$

while

$$\arg\min_{x\in[0,3]} f(x) = 1.$$

The maximum occurs at the boundary, where $f'(3) \neq 0$.

One-Sided Neighborhoods at a Boundary

At an interior local maximum x^*, one compares $f(x^*)$ to function values on both sides. At a right boundary b, only feasible perturbations to the left are allowed.

Thus, b is a local maximum relative to the feasible set $[a, b]$ if there exists $\delta > 0$ such that

$$f(b) \geq f(x), \qquad x \in [b - \delta, b].$$

Similarly, a is a local maximum relative to $[a, b]$ if there exists $\delta > 0$ such that

$$f(a) \geq f(x), \qquad x \in [a, a + \delta].$$

These are one-sided optimality statements because the feasible set restricts allowable perturbations.

Maximum Value Versus Maximizing Argument

There is an important distinction between the value of the objective and the point at which that value occurs.

The maximum value is

$$\max_{x\in[a,b]} f(x).$$

The maximizing argument is

$$x^* = \arg\max_{x\in[a,b]} f(x).$$

The first expression returns a number in the range of f. The second returns a point in the domain of f.

For example, if

$$f(x) = -(x - 2)^2 + 5,$$

then

$$\max_{x\in\mathbb{R}} f(x) = 5,$$

whereas

$$\arg\max_{x\in\mathbb{R}} f(x) = 2.$$

If several points yield the same maximum value, the argmax is a set:

$$\arg\max_{x\in\mathcal{F}} f(x) = \left\{ x \in \mathcal{F} : f(x) = \max_{z\in\mathcal{F}} f(z) \right\}.$$

A Maximum-Principle Viewpoint

A derivative-free statement of optimality is

$$f(x^*) \geq f(x) \qquad \forall x \in \mathcal{F}.$$

Equivalently,

$$x^* \in \arg\max_{x\in\mathcal{F}} f(x).$$

This statement is more general than the condition $f'(x^*) = 0$ because it remains meaningful when:

- the optimum lies on the boundary;
- the function is nonsmooth;
- the decision variable is multidimensional;
- the feasible set is discrete;
- derivatives do not exist or are difficult to compute.

Derivative conditions are powerful computational tools when their assumptions hold, but they are not the definition of optimality.

From Points to Functions

Classical scalar optimization searches for a number:

$$x^* \in \mathbb{R}.$$

Optimal control searches for one or more functions of time:

$$\boldsymbol{u}^*(t), \qquad \boldsymbol{x}^*(t).$$

The unknown is therefore an entire trajectory, not a single point.
A standard optimal control problem has the form

$$\min_{\boldsymbol{u}(\cdot)} J[\boldsymbol{x}(\cdot), \boldsymbol{u}(\cdot)]$$

subject to the differential equation

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}\big(\boldsymbol{x}(t), \boldsymbol{u}(t), t\big),$$

boundary conditions such as

$$\boldsymbol{x}(t_0) = \boldsymbol{x}_0,$$

and possibly path constraints

$$\boldsymbol{g}\big(\boldsymbol{x}(t), \boldsymbol{u}(t), t\big) \leq \boldsymbol{0}.$$

Functions and Functionals

A function maps numbers or vectors to numbers or vectors. For example,

$$f : \mathbb{R} \to \mathbb{R}.$$

A functional maps an entire function to a number:

$$J : \mathcal{U} \to \mathbb{R},$$

where $\mathcal{U}$ is a space of admissible functions.
A common functional is

$$J[u] = \int_{t_0}^{t_f} L\big(u(t), t\big)\, dt.$$

The input to J is the complete function $u(t)$, while the output is a scalar.

Energy of a control signal
Let

$$J[u] = \int_0^T u^2(t)\, dt.$$

Different control histories $u(t)$ produce different scalar values of J. Minimizing J seeks the entire control function that has the smallest accumulated squared magnitude.

A Route-Planning Interpretation

Suppose a traveler wishes to move from an initial location to a destination. A route can be represented by a path

$$\boldsymbol{r}(t) = \begin{bmatrix} x(t) \\ y(t) \end{bmatrix}.$$

Many paths may connect the same endpoints:

$$\boldsymbol{r}(t_0) = \boldsymbol{r}_0, \qquad \boldsymbol{r}(t_f) = \boldsymbol{r}_f.$$

A minimum-time problem seeks

$$\boldsymbol{r}^*(\cdot) \in \arg\min_{\boldsymbol{r}(\cdot)} t_f,$$

subject to kinematic, road, and speed constraints.

A minimum-distance problem instead minimizes

$$J[\boldsymbol{r}] = \int_{t_0}^{t_f} \|\dot{\boldsymbol{r}}(t)\| \, dt.$$

The two formulations may produce different paths. Again, the meaning of optimality depends on the performance index.

The Conceptual Transition

The progression from elementary optimization to optimal control can be summarized as follows:

Concept	Static optimization	Optimal control
Unknown	Point $\boldsymbol{x}$	Function $\boldsymbol{u}(t)$ or trajectory $\boldsymbol{x}(t)$
Objective	$f(\boldsymbol{x})$	Functional $J[\boldsymbol{x}, \boldsymbol{u}]$
Constraints	Algebraic	Differential and algebraic
Solution	Optimal point	Optimal path or control history
Mathematical tools	Calculus, nonlinear programming	Calculus of variations, maximum principle, dynamic programming, direct transcription

Chapter Summary

The main ideas are:

1. Optimization selects the best feasible decision according to a specified performance measure.
2. A problem may be infeasible if its constraints cannot be satisfied.
3. For a smooth scalar function, interior critical points satisfy

$$f'(x) = 0.$$

4. The second derivative classifies nondegenerate critical points:

$$f''(x) > 0 \Rightarrow \text{local minimum},$$

$$f''(x) < 0 \Rightarrow \text{local maximum.}$$

5. A critical point need not be an extremum.
6. Local extrema are not necessarily global extrema.
7. In constrained optimization, the optimizer may occur on the boundary, where the derivative need not vanish.
8. The expression

$$\max_{x \in \mathcal{F}} f(x)$$

gives the optimal value, while

$$\arg\max_{x \in \mathcal{F}} f(x)$$

gives the maximizing argument.

9. Optimal control extends optimization from points to functions.
10. A functional maps a function to a scalar.

Connection. Optimizing trajectories requires a space in which functions themselves can vary, so the scalar ideas above now give way to function spaces and functionals.

Function Spaces, Functionals, and the Calculus of Variations

Optimal control is fundamentally different from ordinary finite-dimensional optimization because the unknown is generally not a point. Instead, the unknown is an entire function of time. A control input

$$\boldsymbol{u}(t)$$

assigns a command at every instant, while the corresponding state trajectory

$$\boldsymbol{x}(t)$$

describes the evolution of the system over an interval of time.

This distinction requires a new mathematical framework. Ordinary calculus studies changes in numbers or vectors. The calculus of variations studies changes in entire functions. The purpose of this section is to establish the foundational distinctions among points, functions, functionals, differences, and variations, and then derive the first-order necessary condition for the classical fixed-endpoint problem in the calculus of variations.

The final result is the Euler–Lagrange equation:

$$\frac{\partial L}{\partial x} - \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial L}{\partial \dot{x}}\right) = 0.$$

Points and Functions

Points in finite-dimensional space

A point in n-dimensional Euclidean space is denoted by

$$\boldsymbol{x} \in \mathbb{R}^n.$$

In component form,

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}.$$

The object $\boldsymbol{x}$ is one point. It contains n numerical components, but it is still a finite-dimensional object.

Vector-valued functions

A vector-valued function of time is written as

$$\boldsymbol{x}(\cdot) \in \Omega,$$

where Ω denotes an appropriate function space. Evaluating the function at a specific time t gives a point:

$$\boldsymbol{x}(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \\ \vdots \\ x_n(t) \end{bmatrix} \in \mathbb{R}^n.$$

Point versus function
The function $\boldsymbol{x}(\cdot)$ is not itself a point in $\mathbb{R}^n$. Rather, it is an element of a function space. For every value of t, the evaluation $\boldsymbol{x}(t)$ is a point in $\mathbb{R}^n$.

A function space may include smoothness, continuity, endpoint, or integrability requirements. Examples include

$$C([t_0, t_f]) = \left\{ x : [t_0, t_f] \to \mathbb{R} \;\middle|\; x \text{ is continuous} \right\},$$
$$C^1([t_0, t_f]) = \left\{ x : [t_0, t_f] \to \mathbb{R} \;\middle|\; x \text{ and } \dot{x} \text{ are continuous} \right\}.$$

Functions and Functionals

Ordinary functions

An ordinary function maps points to points. For example,

$$\boldsymbol{f} : \mathbb{R}^n \to \mathbb{R}^m.$$

Given an input $\boldsymbol{x} \in \mathbb{R}^n$, the function returns

$$\boldsymbol{f}(\boldsymbol{x}) \in \mathbb{R}^m.$$

In the scalar case,

$$f : \mathbb{R}^n \to \mathbb{R}.$$

Functionals

A functional maps an entire function to a scalar:

$$J : \Omega \to \mathbb{R}.$$

Thus,

$$x(\cdot) \in \Omega \quad \Longrightarrow \quad J[x] \in \mathbb{R}.$$

Functional
A functional is a mapping whose input is a function and whose output is a scalar.

A fundamental example is the integral functional

$$J[x] = \int_a^b x(\tau)\,\mathrm{d}\tau.$$

The variable τ is a dummy variable of integration. Once the integration is complete, the result is a real number.

A functional evaluated on two functions
Define

$$J[x] = \int_1^5 x^2(t)\,\mathrm{d}t.$$

If

$$x(t) = t,$$

then

$$J[t] = \int_1^5 t^2\,\mathrm{d}t = \left.\frac{t^3}{3}\right|_1^5 = \frac{124}{3}.$$

If

$$x(t) = \cos t,$$

then

$$J[\cos t] = \int_1^5 \cos^2 t\,\mathrm{d}t.$$

In both cases the input is an entire function, while the output is a real number.

Why the Integration Variable Is Not an Output Variable

Consider

$$J[x] = \int_{t_0}^{t_f} L\big(x(t), \dot{x}(t), t\big)\,\mathrm{d}t.$$

Although the integrand depends explicitly on t, the functional J is not a function of the dummy variable t. The variable is integrated out over the interval $[t_0, t_f]$.

The result depends on the chosen function $x(\cdot)$, on the endpoints, and on the known integrand L, but not on the dummy variable remaining after integration.

Control as a Function

A control input is usually a function of time:

$$\boldsymbol{u} : [t_0, t_f] \to \mathbb{R}^m.$$

At each time t, the control specifies a command

$$\boldsymbol{u}(t) \in \mathbb{R}^m.$$

For a vehicle, controls might include steering angle, throttle, and braking force. The defining feature of a control is that it can be selected, subject to allowable limits such as

$$\boldsymbol{u}_{\min} \leq \boldsymbol{u}(t) \leq \boldsymbol{u}_{\max}.$$

The goal of optimal control is to determine an entire admissible function $\boldsymbol{u}^*(t)$ that guides the dynamical system toward a desired goal while optimizing a performance measure.

Differences and Variations

Difference between points

Let

$$\boldsymbol{x} \in \mathbb{R}^n$$

and consider a second point

$$\boldsymbol{y} = \boldsymbol{x} + \Delta\boldsymbol{x}.$$

Then

$$\Delta\boldsymbol{x} = \boldsymbol{y} - \boldsymbol{x}$$

is the difference or increment between two points.

Variation of a function

For a function $x(\cdot)$, a perturbed function is written

$$\widetilde{x}(t) = x(t) + \delta x(t).$$

Here $\delta x(t)$ is called the variation of $x(t)$. It is itself a function.

Difference versus variation
A difference Δx changes a point. A variation $\delta x(t)$ changes an entire function.

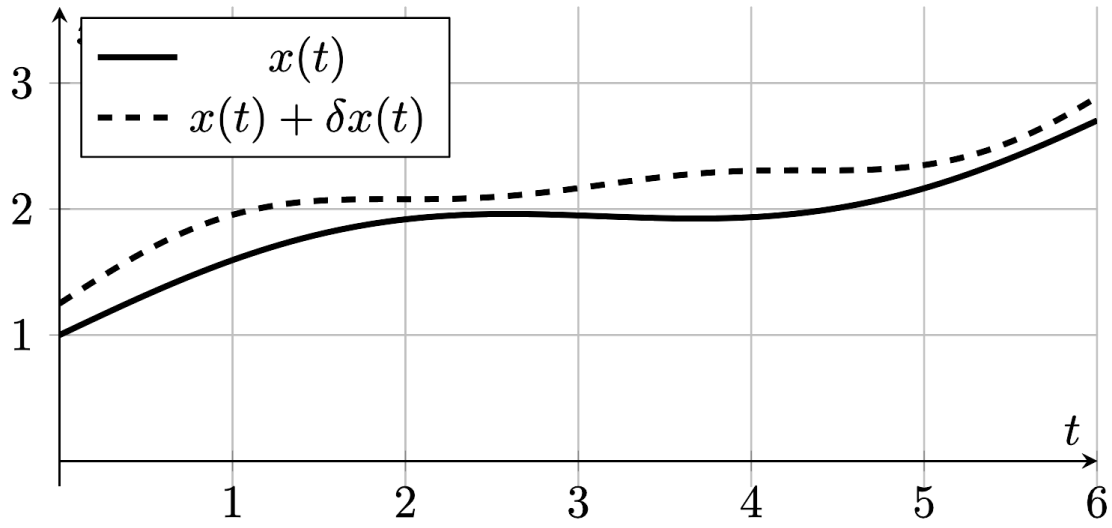

Figure 3: A variation perturbs the entire function, not merely one point.

Increments of Functions and Functionals

Increment of an ordinary function

Let

$$f : \mathbb{R}^n \to \mathbb{R}.$$

The increment in f induced by $\Delta \boldsymbol{q}$ is

$$\Delta f = f(\boldsymbol{q} + \Delta \boldsymbol{q}) - f(\boldsymbol{q}).$$

A first-order Taylor expansion gives

$$\Delta f = \nabla f(\boldsymbol{q})^\top \Delta \boldsymbol{q} + O\left(\|\Delta \boldsymbol{q}\|^2\right).$$

Increment of a functional

Let

$$J : \Omega \to \mathbb{R}.$$

The increment of J produced by the perturbed function

$$q(\cdot) + \delta q(\cdot)$$

is

$$\Delta J = J[q + \delta q] - J[q].$$

The first-order portion of this increment is called the first variation:

$$\delta J.$$

Formally,

$$J[q + \epsilon\eta] = J[q] + \epsilon\, \delta J[q; \eta] + O(\epsilon^2),$$

where $\eta(t)$ is an admissible direction and ϵ is a scalar perturbation parameter.
Equivalently,

$$\delta J[q; \eta] = \left.\frac{\mathrm{d}}{\mathrm{d}\epsilon} J[q + \epsilon\eta]\right|_{\epsilon=0}.$$

Local and Global Minima

Minima of ordinary functions

A point $\boldsymbol{q}^* \in \mathbb{R}^n$ is a local minimum of f if there exists a neighborhood of $\boldsymbol{q}^*$ such that

$$f(\boldsymbol{q}^* + \Delta\boldsymbol{q}) \geq f(\boldsymbol{q}^*)$$

for all sufficiently small $\Delta\boldsymbol{q}$.
It is a global minimum if

$$f(\boldsymbol{q}) \geq f(\boldsymbol{q}^*) \qquad \forall \boldsymbol{q} \in \mathbb{R}^n.$$

Extrema of functionals

A function $x^*(\cdot) \in \Omega$ is a local minimum of J if

$$J[x^* + \delta x] \geq J[x^*]$$

for all sufficiently small admissible variations $\delta x(\cdot)$.
It is a global minimum if

$$J[x] \geq J[x^*] \qquad \forall x(\cdot) \in \Omega.$$

More generally, minima and maxima of functionals are referred to as extrema.

Ordinary Calculus and the Calculus of Variations

Ordinary calculus determines extrema of functions by analyzing derivatives with respect to finite-dimensional variables.

For example, if

$$f : \mathbb{R} \to \mathbb{R},$$

then an interior extremum satisfies

$$\frac{\mathrm{d}f}{\mathrm{d}x} = 0.$$

The calculus of variations performs an analogous task for functionals. An extremizing function $x^*(\cdot)$ must satisfy

$$\delta J = 0$$

for all admissible first-order variations.

Variational analogue of the first derivative
The condition

$$\delta J = 0$$

plays the same conceptual role for functionals that

$$f'(x) = 0$$

plays for ordinary scalar functions.

The Classical Fixed-Endpoint Problem

Consider the functional

$$J[x] = \int_{t_0}^{t_f} L\big(x(t), \dot{x}(t), t\big)\, \mathrm{d}t,$$

where L is known and the function $x(t)$ is unknown.

Assume that

$$x(t_0) = x_0,$$
$$x(t_f) = x_f,$$

and that t_0, t_f, x_0, and x_f are fixed.
The objective is to determine a function

$$x^*(t)$$

that makes $J[x]$ stationary.

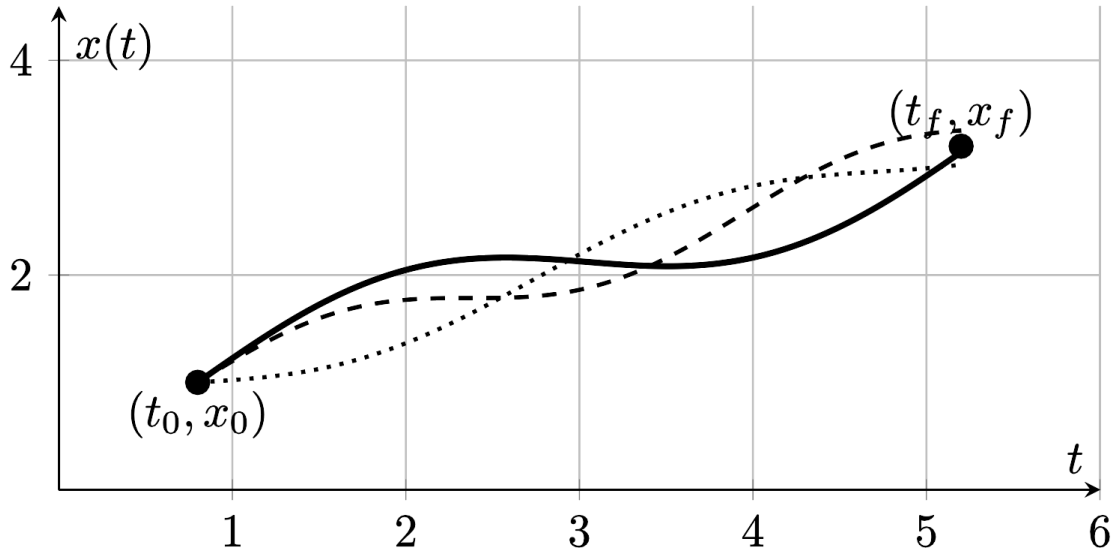

Figure 4: Many admissible functions can connect the same fixed endpoints. The calculus of variations identifies the extremizing path.

Admissible Variations

Let the perturbed path be

$$\widetilde{x}(t) = x(t) + \epsilon\eta(t),$$

where $\eta(t)$ is an arbitrary smooth variation function.
Because the endpoint values are fixed, the perturbed path must satisfy

$$\widetilde{x}(t_0) = x_0,$$
$$\widetilde{x}(t_f) = x_f.$$

Substituting () gives

$$x(t_0) + \epsilon\eta(t_0) = x_0,$$
$$x(t_f) + \epsilon\eta(t_f) = x_f.$$

Since $x(t_0) = x_0$ and $x(t_f) = x_f$, it follows that

$$\eta(t_0) = 0, \qquad \eta(t_f) = 0.$$

Equivalently,

$$\delta x(t_0) = 0, \qquad \delta x(t_f) = 0.$$

Derivation of the First Variation

The first variation of () is

$$\delta J = \delta \int_{t_0}^{t_f} L\big(x, \dot{x}, t\big)\, \mathrm{d}t.$$

Because the integration limits are fixed, the variation can be moved inside the integral:

$$\delta J = \int_{t_0}^{t_f} \delta L\, \mathrm{d}t.$$

To first order,

$$\delta L = \frac{\partial L}{\partial x}\delta x + \frac{\partial L}{\partial \dot{x}}\delta \dot{x}.$$

There is no term involving a variation δt because t is the fixed independent variable in the present problem. Substitution into () gives

$$\delta J = \int_{t_0}^{t_f} \left[\frac{\partial L}{\partial x}\delta x + \frac{\partial L}{\partial \dot{x}}\delta \dot{x}\right] \mathrm{d}t.$$

Dependence of $\delta \dot{x}$ on δx

The variations δx and $\delta \dot{x}$ are not independent because

$$\dot{x}(t) = \frac{\mathrm{d}x}{\mathrm{d}t}.$$

Under standard smoothness assumptions, variation and differentiation commute:

$$\delta \dot{x} = \frac{\mathrm{d}}{\mathrm{d}t}\delta x.$$

Therefore,

$$\delta J = \int_{t_0}^{t_f} \left[\frac{\partial L}{\partial x}\delta x + \frac{\partial L}{\partial \dot{x}}\frac{\mathrm{d}}{\mathrm{d}t}(\delta x)\right] \mathrm{d}t.$$

Integration by Parts

Recall the integration-by-parts identity

$$\int_a^b u(t)\dot{v}(t)\,\mathrm{d}t = u(t)v(t)\big|_a^b - \int_a^b \dot{u}(t)v(t)\,\mathrm{d}t.$$

Choose

$$u(t) = \frac{\partial L}{\partial \dot{x}}, \qquad v(t) = \delta x(t).$$

Then

$$\int_{t_0}^{t_f} \frac{\partial L}{\partial \dot{x}} \frac{\mathrm{d}}{\mathrm{d}t}(\delta x)\,\mathrm{d}t = \frac{\partial L}{\partial \dot{x}}\delta x\Big|_{t_0}^{t_f} - \int_{t_0}^{t_f} \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial L}{\partial \dot{x}}\right)\delta x\,\mathrm{d}t.$$

Substitution into () yields

$$\delta J = \int_{t_0}^{t_f} \left[\frac{\partial L}{\partial x} - \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial L}{\partial \dot{x}}\right)\right]\delta x\,\mathrm{d}t + \frac{\partial L}{\partial \dot{x}}\delta x\Big|_{t_0}^{t_f}.$$

Because the endpoints are fixed,

$$\delta x(t_0) = 0, \qquad \delta x(t_f) = 0.$$

Hence the boundary term vanishes:

$$\frac{\partial L}{\partial \dot{x}}\delta x\Big|_{t_0}^{t_f} = 0.$$

Thus,

$$\delta J = \int_{t_0}^{t_f} \left[\frac{\partial L}{\partial x} - \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial L}{\partial \dot{x}}\right)\right]\delta x\,\mathrm{d}t.$$

The Fundamental Lemma of the Calculus of Variations

Fundamental lemma
Suppose $g(t)$ is continuous on $[t_0, t_f]$ and

$$\int_{t_0}^{t_f} g(t)\eta(t)\,\mathrm{d}t = 0$$

for every sufficiently smooth function $\eta(t)$ satisfying

$$\eta(t_0) = \eta(t_f) = 0.$$

Then

$$g(t) = 0 \qquad \forall t \in [t_0, t_f].$$

Applying the lemma to (), and noting that $\delta x(t)$ is otherwise arbitrary, gives the necessary condition

$$\frac{\partial L}{\partial x} - \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial L}{\partial \dot{x}}\right) = 0.$$

Euler–Lagrange equation
For the fixed-endpoint functional

$$J[x] = \int_{t_0}^{t_f} L(x, \dot{x}, t)\,\mathrm{d}t,$$

a necessary condition for $x^*(t)$ to be an extremal is

$$\boxed{\frac{\partial L}{\partial x} - \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial L}{\partial \dot{x}}\right) = 0}.$$

Interpretation of the Euler–Lagrange Equation

The Euler–Lagrange equation is not the extremizing function itself. It is a differential equation that the extremizing function must satisfy.

The solution procedure is therefore:

1. compute

$$\frac{\partial L}{\partial x};$$

2. compute

$$\frac{\partial L}{\partial \dot{x}};$$

3. differentiate the latter with respect to time:

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial L}{\partial \dot{x}}\right);$$

4. form the Euler–Lagrange differential equation;
5. solve the differential equation subject to the endpoint conditions.

Worked Example: Minimizing the Integral of Squared Slope

Consider

$$J[x] = \int_{t_0}^{t_f} \frac{1}{2}\dot{x}^2(t)\,\mathrm{d}t$$

subject to

$$x(t_0) = x_0, \qquad x(t_f) = x_f.$$

Here,

$$L(x, \dot{x}, t) = \frac{1}{2}\dot{x}^2.$$

Then

$$\frac{\partial L}{\partial x} = 0,$$

and

$$\frac{\partial L}{\partial \dot{x}} = \dot{x}.$$

The Euler–Lagrange equation becomes

$$0 - \frac{\mathrm{d}}{\mathrm{d}t}(\dot{x}) = 0,$$

or

$$\ddot{x} = 0.$$

Integrating twice gives

$$x(t) = c_1 t + c_2.$$

Using the endpoint conditions,

$$x^*(t) = x_0 + \frac{x_f - x_0}{t_f - t_0}(t - t_0).$$

Thus, the extremizing path is the straight line joining the two endpoints.

Worked Example: A Functional with a Trivial Minimizer

Consider

$$J[x] = \int_a^b x^2(t)\,\mathrm{d}t.$$

Because

$$x^2(t) \geq 0$$

for all t, it follows that

$$J[x] \geq 0.$$

The lower bound is achieved by

$$x^*(t) = 0,$$

which yields

$$J[x^*] = 0.$$

The Euler–Lagrange equation confirms this result. Here

$$L = x^2,$$

so

$$\frac{\partial L}{\partial x} = 2x, \qquad \frac{\partial L}{\partial \dot{x}} = 0.$$

Therefore,

$$2x = 0,$$

and hence

$$x^*(t) = 0.$$

Vector Form of the Euler–Lagrange Equation

Let

$$\boldsymbol{x}(t) \in \mathbb{R}^n$$

and consider

$$J[\boldsymbol{x}] = \int_{t_0}^{t_f} L\big(\boldsymbol{x}(t), \dot{\boldsymbol{x}}(t), t\big)\, \mathrm{d}t.$$

The Euler–Lagrange equation becomes

$$\frac{\partial L}{\partial \boldsymbol{x}} - \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial L}{\partial \dot{\boldsymbol{x}}}\right) = \boldsymbol{0}.$$

Componentwise,

$$\frac{\partial L}{\partial x_i} - \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial L}{\partial \dot{x}_i}\right) = 0, \qquad i = 1, \ldots, n.$$

Connection to Lagrangian Mechanics

In analytical mechanics, the Lagrangian is often defined as

$$\mathcal{L}(\boldsymbol{q}, \dot{\boldsymbol{q}}, t) = T(\boldsymbol{q}, \dot{\boldsymbol{q}}, t) - V(\boldsymbol{q}, t),$$

where T is kinetic energy and V is potential energy.
The action functional is

$$\mathcal{S}[\boldsymbol{q}] = \int_{t_0}^{t_f} \mathcal{L}(\boldsymbol{q}, \dot{\boldsymbol{q}}, t)\, \mathrm{d}t.$$

Hamilton's principle states that the physically realized trajectory renders the action stationary:

$$\delta \mathcal{S} = 0.$$

The resulting Euler–Lagrange equations are

$$\frac{\partial \mathcal{L}}{\partial q_i} - \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial \mathcal{L}}{\partial \dot{q}_i}\right) = 0.$$

Thus, the same mathematical structure that determines stationary functionals also generates the equations of motion in classical mechanics.

Principle of Stationary Action

The phrase *principle of least action* is commonly used, but the more precise statement is that the physical path makes the action stationary. The path may correspond to a minimum, maximum, or saddle-type stationary value depending on the problem.

Therefore,

$$\delta \mathcal{S} = 0$$

is the fundamental first-order condition.

Important Assumptions in the Derivation

The derivation of () relied on the following assumptions:

1. the endpoint times t_0 and t_f are fixed;
2. the endpoint values $x(t_0)$ and $x(t_f)$ are fixed;
3. the admissible functions are sufficiently smooth;
4. the integrand $L(x, \dot{x}, t)$ is differentiable in its arguments;
5. the variation and time differentiation commute;
6. only first-order variation terms are retained.

When endpoint values or endpoint times are free, additional boundary conditions arise. These are called natural boundary conditions or transversality conditions.

Chapter Summary

The principal ideas developed in this section are:

1. A point $\boldsymbol{x} \in \mathbb{R}^n$ and a function $\boldsymbol{x}(\cdot) \in \Omega$ are fundamentally different mathematical objects.
2. An ordinary function maps points to values:

$$f : \mathbb{R}^n \to \mathbb{R}^m.$$

3. A functional maps functions to real numbers:

$$J : \Omega \to \mathbb{R}.$$

4. A finite-dimensional increment is denoted by $\Delta \boldsymbol{x}$, while a functional variation is denoted by $\delta x(t)$.
5. The first variation δJ is the first-order change in a functional caused by a variation in its input function.
6. For the fixed-endpoint problem

$$J[x] = \int_{t_0}^{t_f} L(x, \dot{x}, t)\, \mathrm{d}t,$$

the first-order stationarity condition is

$$\delta J = 0.$$

7. Integration by parts converts the dependence on $\delta \dot{x}$ into dependence on δx.
8. Fixed endpoints imply

$$\delta x(t_0) = \delta x(t_f) = 0.$$

9. The resulting Euler–Lagrange equation is

$$\frac{\partial L}{\partial x} - \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial L}{\partial \dot{x}}\right) = 0.$$

10. The Euler–Lagrange equation is a necessary condition, not a guarantee of a minimum.
11. The same equation appears in classical mechanics through Hamilton's principle of stationary action.

Connection. Once interior variations are understood, the next question is what changes when the endpoints are not fixed; endpoint variations provide that extension.

Endpoint Variations and Transversality Conditions

The classical fixed-endpoint problem in the calculus of variations assumes that the initial time, final time, initial state, and final state are all prescribed. Under those assumptions, the first variation leads to the Euler–Lagrange equation, and the endpoint terms vanish because the endpoint variations are zero.

Many practical problems are less restrictive. A system may be required to reach a specified terminal state without prescribing the arrival time. Alternatively, the final time may be fixed while the final state is free. More general problems may allow both endpoint time and endpoint state to vary subject to geometric or algebraic constraints.

These cases produce additional necessary conditions known as *transversality conditions*. This section develops the endpoint variation identities, derives the free-final-time condition, discusses the free-final-state condition, and illustrates the results through a mechanical example.

Review of the Basic Functional

Consider the functional

$$J[x] = \int_{t_0}^{t_f} L\big(x(t), \dot{x}(t), t\big)\, \mathrm{d}t.$$

The variable t is a dummy variable of integration. Therefore, after integration, $J[x]$ is not a function of the dummy variable t. Instead, the value of the functional depends on:

- the entire function $x(\cdot)$;
- the corresponding derivative $\dot{x}(\cdot)$;
- the endpoint parameters t_0 and t_f;
- the integrand L.

If t_f is free, then changing t_f can change the numerical value of the functional. Consequently, the first variation must include a term involving δt_f.

Fixed and Free Endpoint Data

For the most restrictive scalar problem, all endpoint data are fixed:

$$\begin{aligned} t_0 &= \text{fixed}, \\ t_f &= \text{fixed}, \\ x(t_0) &= x_0, \\ x(t_f) &= x_f. \end{aligned}$$

The admissible variations satisfy

$$\delta x(t_0) = 0, \qquad \delta x(t_f) = 0.$$

A first extension is the free-final-time problem:

$$
\begin{aligned}
t_0 &= \text{fixed}, \\
x(t_0) &= x_0, \\
x(t_f) &= x_f, \\
t_f &\text{ is free.}
\end{aligned}
$$

Here, the terminal state value is prescribed, but the time at which that value is reached is unknown.

Geometric Interpretation of a Free Final Time

Suppose an admissible path begins at the fixed point

$$(t_0, x_0)$$

and must reach the prescribed terminal value x_f. If t_f is free, the endpoint may occur anywhere along the horizontal line

$$x = x_f.$$

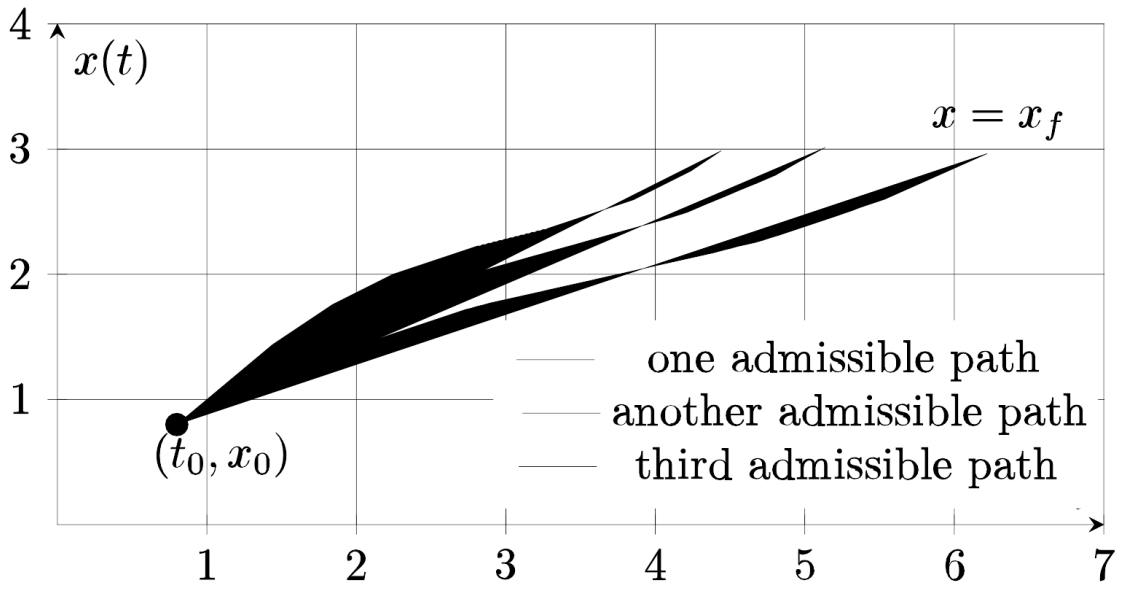

Figure 5: When the terminal state is fixed but the terminal time is free, admissible trajectories may reach the line $x = x_f$ at different times.

Variation of an Endpoint

The endpoint of a trajectory can change in two different ways:

1. the function itself may vary at the original endpoint time;
2. the endpoint time may move.

These effects must be distinguished carefully.
Let the nominal final endpoint be

$$\big(t_f, x_f\big), \qquad x_f = x(t_f).$$

After perturbation, let the endpoint become

$$\big(t_f + \delta t_f,\ x_f + \delta x_f\big).$$

The perturbed function is

$$\widetilde{x}(t) = x(t) + \delta x(t).$$

A first-order Taylor expansion gives

$$\begin{aligned} x_f + \delta x_f &= \widetilde{x}(t_f + \delta t_f) \\ &= x(t_f + \delta t_f) + \delta x(t_f + \delta t_f) \\ &\approx x(t_f) + \dot{x}(t_f)\delta t_f + \delta x(t_f). \end{aligned}$$

Since $x(t_f) = x_f$, we obtain the endpoint variation identity

$$\boxed{\delta x_f = \delta x(t_f) + \dot{x}(t_f)\delta t_f}.$$

Similarly, at the initial endpoint,

$$\boxed{\delta x_0 = \delta x(t_0) + \dot{x}(t_0)\delta t_0}.$$

Do not confuse these quantities
The three quantities

$$\delta x_f, \qquad \delta x(t_f), \qquad \dot{x}(t_f)\delta t_f$$

have different meanings.

- δx_f is the total variation of the endpoint value.
- $\delta x(t_f)$ is the variation of the function evaluated at the original time t_f.
- $\dot{x}(t_f)\delta t_f$ is the first-order change caused by moving the endpoint time along the nominal trajectory.

Endpoint Parameters Versus the Dummy Variable

The symbols t_0 and t_f denote endpoint parameters. They are real numbers defining the integration interval. They are not the same object as the dummy integration variable t.

Thus, while the dummy variable is integrated out, the endpoint parameters may remain as arguments of the resulting functional:

$$J = J[x(\cdot), t_0, t_f].$$

If t_f is free, then

$$\delta t_f$$

is an admissible independent endpoint variation.

Variation of an Integral with a Moving Upper Limit

Consider

$$J[x, t_f] = \int_{t_0}^{t_f} L(x, \dot{x}, t)\,\mathrm{d}t,$$

where t_0 is fixed but t_f may vary.
The variation has two contributions:

$$\delta J = \int_{t_0}^{t_f} \delta L\,\mathrm{d}t + L\big(x(t_f), \dot{x}(t_f), t_f\big)\delta t_f.$$

The second term follows from the fundamental theorem of calculus:

$$\frac{\partial}{\partial t_f}\int_{t_0}^{t_f} L(x, \dot{x}, t)\,\mathrm{d}t = L\big(x(t_f), \dot{x}(t_f), t_f\big).$$

First Variation for a Free Final Time

To first order,

$$\delta L = \frac{\partial L}{\partial x}\delta x + \frac{\partial L}{\partial \dot{x}}\delta \dot{x}.$$

Therefore,

$$\begin{aligned}\delta J = &\int_{t_0}^{t_f}\left[\frac{\partial L}{\partial x}\delta x + \frac{\partial L}{\partial \dot{x}}\delta \dot{x}\right]\mathrm{d}t \\ &+ L(t_f)\delta t_f,\end{aligned}$$

where

$$L(t_f) := L\big(x(t_f), \dot{x}(t_f), t_f\big).$$

Because

$$\delta \dot{x} = \frac{\mathrm{d}}{\mathrm{d}t}\delta x,$$

integration by parts gives

$$\begin{aligned}\delta J = &\int_{t_0}^{t_f} \left[\frac{\partial L}{\partial x} - \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial L}{\partial \dot{x}}\right)\right] \delta x \,\mathrm{d}t \\ &+ \frac{\partial L}{\partial \dot{x}}\delta x\Big|_{t_0}^{t_f} + L(t_f)\delta t_f.\end{aligned}$$

The initial endpoint is fixed, so

$$\delta x(t_0) = 0.$$

Hence,

$$\begin{aligned}\delta J = &\int_{t_0}^{t_f} \left[\frac{\partial L}{\partial x} - \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial L}{\partial \dot{x}}\right)\right] \delta x \,\mathrm{d}t \\ &+ \frac{\partial L}{\partial \dot{x}}\Big|_{t_f} \delta x(t_f) + L(t_f)\delta t_f.\end{aligned}$$

Using the Fixed Terminal State

The terminal state value is fixed:

$$x_f = \text{constant}.$$

Therefore,

$$\delta x_f = 0.$$

Using the endpoint identity (),

$$0 = \delta x(t_f) + \dot{x}(t_f)\delta t_f,$$

so

$$\boxed{\delta x(t_f) = -\dot{x}(t_f)\delta t_f}.$$

Substituting into () gives

$$\begin{aligned}\delta J = &\int_{t_0}^{t_f} \left[\frac{\partial L}{\partial x} - \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial L}{\partial \dot{x}}\right)\right] \delta x \,\mathrm{d}t \\ &+ \left[L - \dot{x}\frac{\partial L}{\partial \dot{x}}\right]_{t_f} \delta t_f.\end{aligned}$$

Now the variation has been separated into two independent parts:

- an arbitrary interior function variation $\delta x(t)$;
- an independent scalar endpoint variation δt_f.

Necessary Conditions for the Free-Final-Time Problem

Because $\delta x(t)$ and δt_f vary independently, both coefficients must vanish.
The interior condition is the Euler–Lagrange equation:

$$\boxed{\frac{\partial L}{\partial x} - \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial L}{\partial \dot{x}}\right) = 0}.$$

The free-final-time transversality condition is

$$\boxed{\left[L - \dot{x}\frac{\partial L}{\partial \dot{x}}\right]_{t_f} = 0}.$$

Free-final-time condition
For

$$J[x, t_f] = \int_{t_0}^{t_f} L(x, \dot{x}, t)\,\mathrm{d}t,$$

with fixed t_0, fixed $x(t_0)$, fixed $x(t_f) = x_f$, and free t_f, the necessary conditions are

$$\frac{\partial L}{\partial x} - \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial L}{\partial \dot{x}}\right) = 0,$$
$$\left[L - \dot{x}\frac{\partial L}{\partial \dot{x}}\right]_{t_f} = 0.$$

Interpretation Through the Hamiltonian-Like Quantity

Define

$$p = \frac{\partial L}{\partial \dot{x}}.$$

Then define the quantity

$$H = p\dot{x} - L.$$

The free-final-time condition () is equivalent to

$$\boxed{H(t_f) = 0}.$$

This relation anticipates the Hamiltonian transversality conditions that appear later in optimal control.

Why Independent Variations Matter

Before integration by parts, the first variation contains both δx and $\delta \dot{x}$. These quantities are not independent because

$$\delta \dot{x} = \frac{\mathrm{d}}{\mathrm{d}t}\delta x.$$

Integration by parts rewrites the expression entirely in terms of δx and endpoint variations.

In the free-final-time problem, the endpoint relation

$$\delta x(t_f) = -\dot{x}(t_f)\delta t_f$$

further removes the dependence between the endpoint state variation and the endpoint time variation.

Only after these dependencies have been eliminated can the coefficients of independent variations be set to zero separately.

Free Final State with Fixed Final Time

Consider a different problem:

$$\begin{aligned} t_0 &= \text{fixed}, \\ t_f &= \text{fixed}, \\ x(t_0) &= x_0, \\ x(t_f) &\text{ is free.} \end{aligned}$$

Since t_f is fixed,

$$\delta t_f = 0.$$

The endpoint identity becomes

$$\delta x_f = \delta x(t_f).$$

The first variation is

$$\begin{aligned} \delta J = &\int_{t_0}^{t_f} \left[\frac{\partial L}{\partial x} - \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial L}{\partial \dot{x}}\right) \right] \delta x \, \mathrm{d}t \\ &+ \left. \frac{\partial L}{\partial \dot{x}} \right|_{t_f} \delta x(t_f). \end{aligned}$$

Because $\delta x(t_f)$ is arbitrary, the natural boundary condition is

$$\boxed{\left.\frac{\partial L}{\partial \dot{x}}\right|_{t_f} = 0}.$$

Thus, a free endpoint value generates an additional boundary condition.

Summary of Common Endpoint Cases

Final endpoint data	Variation relation	Additional condition
x_f fixed, t_f fixed	$\delta x(t_f) = 0$	None beyond prescribed boundary data
x_f fixed, t_f free	$\delta x(t_f) = -\dot{x}(t_f)\delta t_f$	$[L - \dot{x}L_{\dot{x}}]_{t_f} = 0$
x_f free, t_f fixed	$\delta t_f = 0$, $\delta x(t_f)$ arbitrary	$L_{\dot{x}}\|_{t_f} = 0$

Necessary Versus Sufficient Conditions

The Euler–Lagrange and transversality equations are first-order necessary conditions. They identify stationary candidates.

They do not by themselves determine whether a candidate is:

- a local minimum;
- a local maximum;
- a saddle-type stationary path.

A condition is necessary if every optimizer must satisfy it. A condition is sufficient if satisfying it guarantees optimality. The first-order conditions developed here are generally necessary but not sufficient.

Worked Example: Mass–Spring System

Consider a unit-mass, unit-stiffness spring system. Let the Lagrangian be

$$L(x, \dot{x}) = T - V = \frac{1}{2}\dot{x}^2 - \frac{1}{2}x^2.$$

The action functional is

$$J[x] = \int_{t_0}^{t_f} \left(\frac{1}{2}\dot{x}^2 - \frac{1}{2}x^2\right) \mathrm{d}t.$$

The required derivatives are

$$\frac{\partial L}{\partial x} = -x,$$

and

$$\frac{\partial L}{\partial \dot{x}} = \dot{x}.$$

Therefore,

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial L}{\partial \dot{x}}\right) = \ddot{x}.$$

The Euler–Lagrange equation becomes

$$-x - \ddot{x} = 0,$$

or

$$\boxed{\ddot{x} + x = 0}.$$

The general solution is

$$x(t) = A\cos t + B\sin t.$$

For prescribed endpoint values

$$x(t_0) = x_0, \qquad x(t_f) = x_f,$$

the constants A and B are determined from

$$\begin{aligned} A\cos t_0 + B\sin t_0 &= x_0, \\ A\cos t_f + B\sin t_f &= x_f. \end{aligned}$$

In matrix form,

$$\begin{bmatrix} \cos t_0 & \sin t_0 \\ \cos t_f & \sin t_f \end{bmatrix} \begin{bmatrix} A \\ B \end{bmatrix} = \begin{bmatrix} x_0 \\ x_f \end{bmatrix}.$$

Provided the matrix is nonsingular,

$$\sin(t_f - t_0) \neq 0,$$

the constants are uniquely determined.

General Mass–Spring Parameters

For mass m and spring stiffness k,

$$L = \frac{1}{2}m\dot{x}^2 - \frac{1}{2}kx^2.$$

Then

$$\frac{\partial L}{\partial x} = -kx,$$

and

$$\frac{\partial L}{\partial \dot{x}} = m\dot{x}.$$

The Euler–Lagrange equation gives

$$-kx - m\ddot{x} = 0,$$

or

$$\boxed{m\ddot{x} + kx = 0}.$$

This is exactly the familiar equation of motion for an undamped mass–spring oscillator.

Free-Final-Time Condition for the Mass–Spring Lagrangian

For

$$L = \frac{1}{2}m\dot{x}^2 - \frac{1}{2}kx^2,$$

we have

$$\frac{\partial L}{\partial \dot{x}} = m\dot{x}.$$

The free-final-time condition is

$$\begin{aligned} 0 &= \left[L - \dot{x}\frac{\partial L}{\partial \dot{x}}\right]_{t_f} \\ &= \left[\frac{1}{2}m\dot{x}^2 - \frac{1}{2}kx^2 - m\dot{x}^2\right]_{t_f} \\ &= \left[-\frac{1}{2}m\dot{x}^2 - \frac{1}{2}kx^2\right]_{t_f}. \end{aligned}$$

Thus,

$$m\dot{x}^2(t_f) + kx^2(t_f) = 0.$$

For positive m and k, this implies

$$x(t_f) = 0, \qquad \dot{x}(t_f) = 0.$$

This shows that a free-final-time condition can strongly restrict the terminal behavior. It also illustrates why endpoint conditions must be interpreted together with the full problem statement.

Extension to Vector-Valued Functions

Let

$$\boldsymbol{x}(t) \in \mathbb{R}^n$$

and consider

$$J[\boldsymbol{x}] = \int_{t_0}^{t_f} L(\boldsymbol{x}, \dot{\boldsymbol{x}}, t)\, \mathrm{d}t.$$

The endpoint variation identity becomes

$$\delta \boldsymbol{x}_f = \delta \boldsymbol{x}(t_f) + \dot{\boldsymbol{x}}(t_f)\delta t_f.$$

The Euler–Lagrange equations are

$$\frac{\partial L}{\partial \boldsymbol{x}} - \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial L}{\partial \dot{\boldsymbol{x}}}\right) = \boldsymbol{0}.$$

For fixed terminal state and free terminal time, the transversality condition becomes

$$\boxed{\left[L - \dot{\boldsymbol{x}}^{\top}\frac{\partial L}{\partial \dot{\boldsymbol{x}}}\right]_{t_f} = 0}.$$

General Endpoint Curves

A still more general endpoint may be constrained to lie on a curve

$$\psi\left(x_f, t_f\right) = 0.$$

Then

$$\psi_x\, \delta x_f + \psi_t\, \delta t_f = 0.$$

This relation links δx_f and δt_f. Substitution into the general boundary variation produces a transversality condition associated with the endpoint curve.

This geometric viewpoint is fundamental in advanced optimal control, where terminal states may be constrained to manifolds rather than prescribed exactly.

Chapter Summary

The main results are:

1. The functional

$$J[x] = \int_{t_0}^{t_f} L(x, \dot{x}, t)\, \mathrm{d}t$$

depends on the entire path and may also depend on free endpoint parameters.

2. The dummy integration variable t is not the same object as the endpoint parameters t_0 and t_f.
3. The total final-state variation satisfies

$$\delta x_f = \delta x(t_f) + \dot{x}(t_f)\delta t_f.$$

4. If x_f is fixed and t_f is free, then

$$\delta x(t_f) = -\dot{x}(t_f)\delta t_f.$$

5. A moving upper integration limit contributes

$$L(t_f)\delta t_f$$

to the first variation.

6. The interior stationarity condition remains the Euler–Lagrange equation:

$$\frac{\partial L}{\partial x} - \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial L}{\partial \dot{x}}\right) = 0.$$

7. For fixed x_f and free t_f, the additional transversality condition is

$$\left[L - \dot{x}\frac{\partial L}{\partial \dot{x}}\right]_{t_f} = 0.$$

8. For free x_f and fixed t_f, the natural boundary condition is

$$\left.\frac{\partial L}{\partial \dot{x}}\right|_{t_f} = 0.$$

9. These conditions are necessary, not generally sufficient.
10. For a mass–spring Lagrangian, the Euler–Lagrange equation reproduces the physical equation of motion.

Connection. The transversality calculation becomes more useful when organized as natural boundary conditions and embedded in the standard structure of an optimal-control problem.

Natural Boundary Conditions and the Structure of Optimal Control Problems

The calculus of variations produces two distinct kinds of necessary conditions:

1. differential equations governing the trajectory in the interior of the time interval;
2. boundary or transversality conditions governing behavior at the endpoints.

The endpoint conditions depend directly on which endpoint quantities are fixed and which are free. A small change in the problem statement can therefore produce a significantly different boundary condition.

This section first revisits the free-final-time case through a worked example. It then derives the natural boundary condition for a free terminal state at fixed final time. The chapter also develops standard vector-derivative notation and introduces the principal ingredients of a general optimal control problem: dynamics, boundary conditions, static parameters, controls, and performance indices.

Review: Fixed Terminal State and Free Terminal Time

Consider

$$J[x, t_f] = \int_{t_0}^{t_f} L(x, \dot{x}, t)\, \mathrm{d}t,$$

with

$$\begin{aligned} x(t_0) &= x_0, \\ x(t_f) &= x_f, \end{aligned}$$

where t_0, x_0, and x_f are fixed, while t_f is free.

The necessary conditions are

$$\frac{\partial L}{\partial x} - \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial L}{\partial \dot{x}}\right) = 0$$

and

$$\left[L - \dot{x}\frac{\partial L}{\partial \dot{x}}\right]_{t_f} = 0.$$

The first equation governs the interior trajectory. The second supplies the missing boundary condition associated with the free final time.

Worked Example: A Free-Final-Time Variational Problem

Consider

$$J[x, t_f] = \int_1^{t_f} \left[2x(t) + \frac{1}{2}\dot{x}^2(t)\right]\, \mathrm{d}t,$$

subject to

$$x(1) = 4, \qquad x(t_f) = 4, \qquad t_f > 1.$$

The integrand is

$$L(x, \dot{x}, t) = 2x + \frac{1}{2}\dot{x}^2.$$

Euler–Lagrange equation

We have

$$\frac{\partial L}{\partial x} = 2$$

and

$$\frac{\partial L}{\partial \dot{x}} = \dot{x}.$$

Thus,

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial L}{\partial \dot{x}}\right) = \ddot{x}.$$

The Euler–Lagrange equation gives

$$2 - \ddot{x} = 0,$$

or

$$\boxed{\ddot{x} = 2}.$$

Integrating twice yields

$$x(t) = t^2 + c_1 t + c_2.$$

Initial boundary condition

Using $x(1) = 4$,

$$1 + c_1 + c_2 = 4,$$

so

$$c_1 + c_2 = 3.$$

Free-final-time condition

The transversality condition is

$$\left[L - \dot{x}\frac{\partial L}{\partial \dot{x}}\right]_{t_f} = 0.$$

Since

$$\frac{\partial L}{\partial \dot{x}} = \dot{x},$$

we obtain

$$\begin{aligned} 0 &= \left[2x + \frac{1}{2}\dot{x}^2 - \dot{x}^2\right]_{t_f} \\ &= \left[2x - \frac{1}{2}\dot{x}^2\right]_{t_f}. \end{aligned}$$

Using the terminal-state condition $x(t_f) = 4$,

$$8 - \frac{1}{2}\dot{x}^2(t_f) = 0,$$

which implies

$$\dot{x}^2(t_f) = 16.$$

Therefore,

$$\dot{x}(t_f) = \pm 4.$$

From (),

$$\dot{x}(t) = 2t + c_1.$$

Combining this with

$$x(t_f) = t_f^2 + c_1 t_f + c_2 = 4$$

and (), the admissible solution satisfying $t_f > 1$ is

$$\boxed{t_f = 5}$$

with

$$c_1 = -6, \qquad c_2 = 9.$$

Hence,

$$\boxed{x^*(t) = t^2 - 6t + 9}.$$

A direct check gives

$$x^*(1) = 4,$$
$$x^*(5) = 4,$$
$$\dot{x}^*(5) = 4.$$

Free Terminal State with Fixed Final Time

Now consider

$$J[x] = \int_{t_0}^{t_f} L(x, \dot{x}, t)\, \mathrm{d}t,$$

subject to

$$x(t_0) = x_0,$$

where t_0 and t_f are fixed, but $x(t_f)$ is free.
The endpoint configuration is shown conceptually in Figure 6.

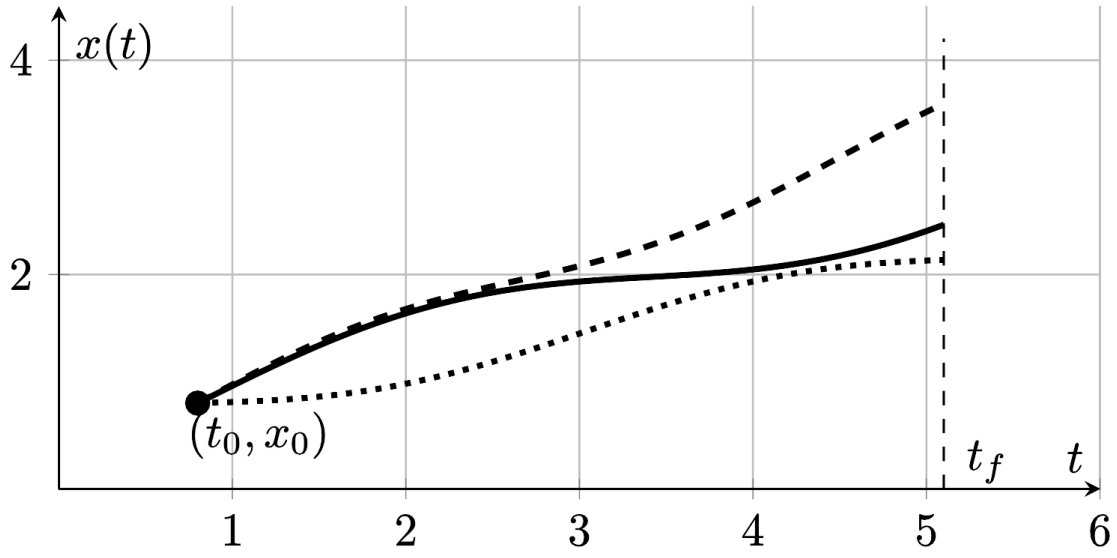

Figure 6: When t_f is fixed but $x(t_f)$ is free, admissible trajectories may terminate at different state values along the vertical line $t = t_f$.

Derivation of the Natural Boundary Condition

Because the endpoint times are fixed, variation of the functional gives

$$\delta J = \int_{t_0}^{t_f} \delta L \, \mathrm{d}t.$$

To first order,

$$\delta L = \frac{\partial L}{\partial x}\delta x + \frac{\partial L}{\partial \dot{x}}\delta \dot{x}.$$

Thus,

$$\delta J = \int_{t_0}^{t_f} \left[\frac{\partial L}{\partial x}\delta x + \frac{\partial L}{\partial \dot{x}}\delta \dot{x} \right] \mathrm{d}t.$$

Using

$$\delta \dot{x} = \frac{\mathrm{d}}{\mathrm{d}t}\delta x$$

and integrating by parts,

$$\delta J = \left. \frac{\partial L}{\partial \dot{x}}\delta x \right|_{t_0}^{t_f} + \int_{t_0}^{t_f} \left[\frac{\partial L}{\partial x} - \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial L}{\partial \dot{x}} \right) \right] \delta x \, \mathrm{d}t.$$

Since the initial state is fixed,

$$\delta x(t_0) = 0.$$

Since t_f is fixed,

$$\delta t_f = 0.$$

The endpoint identity

$$\delta x_f = \delta x(t_f) + \dot{x}(t_f)\delta t_f$$

therefore reduces to

$$\delta x_f = \delta x(t_f).$$

Because the terminal state is free,

$$\delta x(t_f)$$

is arbitrary and generally nonzero.
Equation () becomes

$$\delta J = \left.\frac{\partial L}{\partial \dot{x}}\right|_{t_f} \delta x(t_f) + \int_{t_0}^{t_f} \left[\frac{\partial L}{\partial x} - \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial L}{\partial \dot{x}}\right)\right] \delta x \, \mathrm{d}t.$$

Since the interior variation and terminal variation are independent, the necessary conditions are

$$\boxed{\frac{\partial L}{\partial x} - \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial L}{\partial \dot{x}}\right) = 0}$$

and

$$\boxed{\left.\frac{\partial L}{\partial \dot{x}}\right|_{t_f} = 0}.$$

The second equation is called a *natural boundary condition*.

Boundary Conditions as Products

Boundary terms in variational derivations often appear in the form

$$A\,B = 0.$$

For scalar quantities, this implies

$$A = 0, \qquad B = 0, \qquad \text{or both.}$$

However, one cannot arbitrarily choose which factor is zero. The problem statement determines whether an endpoint variation is zero.

For example,

$$\left.\frac{\partial L}{\partial \dot{x}}\right|_{t_f} \delta x(t_f) = 0.$$

Two cases are possible:

1. If $x(t_f)$ is fixed, then

$$\delta x(t_f) = 0,$$

and no additional natural boundary condition arises.

2. If $x(t_f)$ is free, then $\delta x(t_f)$ is arbitrary, so

$$\left.\frac{\partial L}{\partial \dot{x}}\right|_{t_f} = 0.$$

Do not set factors to zero without justification
The boundary product is zero because of the endpoint structure of the problem. The analyst must first identify which variations are fixed and which are arbitrary.

For vector quantities, a zero inner product

$$\boldsymbol{a}^\top \boldsymbol{b} = 0$$

does not generally imply that either vector is zero. It may instead indicate orthogonality. This distinction becomes important in vector transversality conditions.

Interior Conditions and Endpoint Conditions

The first variation naturally separates into:

- an integral term involving arbitrary path variations;
- boundary terms involving endpoint variations.

The integral term yields the Euler–Lagrange differential equation, which governs the interior path. The boundary terms yield endpoint conditions, whose form depends on which endpoint quantities are free.

This separation is fundamental:

$$\text{path condition} \quad + \quad \text{endpoint conditions.}$$

A First-Principles Workflow

For each variational problem, the safest procedure is:

1. identify the functional;
2. identify all quantities that may vary;
3. identify all quantities that are fixed;
4. compute the first variation;
5. account for moving integration limits if present;
6. use integration by parts to eliminate $\delta\dot{x}$;
7. apply endpoint variation identities;
8. rewrite the result in terms of independent variations;
9. set the coefficients of independent variations to zero.

This approach is more reliable than applying formulas from a table because it reveals the origin of every boundary condition.

Vector Notation

Optimal control problems are usually vector-valued. Let

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \in \mathbb{R}^n.$$

Its transpose is

$$\boldsymbol{x}^\top = \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix}.$$

Throughout this section, vectors are taken to be column vectors unless otherwise stated.

Gradient of a Scalar Function

Let

$$f : \mathbb{R}^n \to \mathbb{R}.$$

Using the row-gradient convention,

$$\nabla f(\boldsymbol{x}) = \begin{bmatrix} \dfrac{\partial f}{\partial x_1} & \dfrac{\partial f}{\partial x_2} & \cdots & \dfrac{\partial f}{\partial x_n} \end{bmatrix}.$$

Some texts define the gradient as a column vector. Either convention is acceptable if used consistently. The present convention is convenient for writing first-order expansions as

$$f(\boldsymbol{x} + \Delta\boldsymbol{x}) = f(\boldsymbol{x}) + \nabla f(\boldsymbol{x})\Delta\boldsymbol{x} + O(\|\Delta\boldsymbol{x}\|^2).$$

The Hessian Matrix

The Hessian of a scalar function is the matrix of second derivatives:

$$\nabla^2 f(\boldsymbol{x}) = \begin{bmatrix} \dfrac{\partial^2 f}{\partial x_1^2} & \cdots & \dfrac{\partial^2 f}{\partial x_1 \partial x_n} \\ \vdots & \ddots & \vdots \\ \dfrac{\partial^2 f}{\partial x_n \partial x_1} & \cdots & \dfrac{\partial^2 f}{\partial x_n^2} \end{bmatrix}.$$

If f is twice continuously differentiable, then

$$\frac{\partial^2 f}{\partial x_i \partial x_j} = \frac{\partial^2 f}{\partial x_j \partial x_i},$$

and the Hessian is symmetric.

Jacobian of a Vector Function

Let

$$\boldsymbol{f} : \mathbb{R}^n \to \mathbb{R}^m$$

with

$$\boldsymbol{f}(\boldsymbol{x}) = \begin{bmatrix} f_1(\boldsymbol{x}) \\ f_2(\boldsymbol{x}) \\ \vdots \\ f_m(\boldsymbol{x}) \end{bmatrix}.$$

The Jacobian is

$$\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \frac{\partial f_m}{\partial x_2} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix} \in \mathbb{R}^{m \times n}.$$

The Jacobian maps a first-order state perturbation to a first-order function perturbation:

$$\Delta \boldsymbol{f} \approx \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}} \Delta \boldsymbol{x}.$$

General Dynamical System

A broad class of physical systems can be written as

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}\left(\boldsymbol{x}(t), \boldsymbol{u}(t), t, \boldsymbol{q}\right),$$

where

- $\boldsymbol{x}(t) \in \mathbb{R}^n$ is the state;
- $\boldsymbol{u}(t) \in \mathbb{R}^m$ is the control or input;
- $\boldsymbol{q} \in \mathbb{R}^p$ is a static parameter vector;
- $\boldsymbol{f}$ defines the dynamics.

The vector $\boldsymbol{q}$ may be optimized, but it is not a function of time.

Why first-order form is general

A second-order mechanical equation such as

$$m\ddot{r} = F$$

can be converted to first-order form by defining

$$x_1 = r, \qquad x_2 = \dot{r}.$$

Then

$$\dot{x}_1 = x_2,$$
$$\dot{x}_2 = \frac{F}{m}.$$

Thus, higher-order systems can generally be rewritten as first-order state-space systems.

General Boundary Conditions

A general equality boundary constraint may be written as

$$\boldsymbol{\phi}\big(\boldsymbol{x}(t_0), t_0, \boldsymbol{x}(t_f), t_f, \boldsymbol{q}\big) = \mathbf{0}.$$

This formulation includes simple fixed endpoint conditions as a special case. For example,

$$\boldsymbol{x}(t_0) = \boldsymbol{x}_0,$$
$$\boldsymbol{x}(t_f) = \boldsymbol{x}_f$$

can be written as

$$\boldsymbol{\phi} = \begin{bmatrix} \boldsymbol{x}(t_0) - \boldsymbol{x}_0 \\ \boldsymbol{x}(t_f) - \boldsymbol{x}_f \end{bmatrix} = \mathbf{0}.$$

Point-to-Point and Locus-to-Locus Problems

A point-to-point problem prescribes exact endpoint states:

$$\boldsymbol{x}(t_0) = \boldsymbol{x}_0, \qquad \boldsymbol{x}(t_f) = \boldsymbol{x}_f.$$

A more general problem may require the initial and final states to lie on specified manifolds:

$$\boldsymbol{\psi}_0\big(\boldsymbol{x}(t_0)\big) = \mathbf{0},$$
$$\boldsymbol{\psi}_f\big(\boldsymbol{x}(t_f)\big) = \mathbf{0}.$$

For example, an orbital-transfer problem may require a spacecraft to begin somewhere on one orbit and end somewhere on another orbit without prescribing the exact position on either orbit.

This broader boundary formulation substantially enlarges the class of problems that can be modeled.

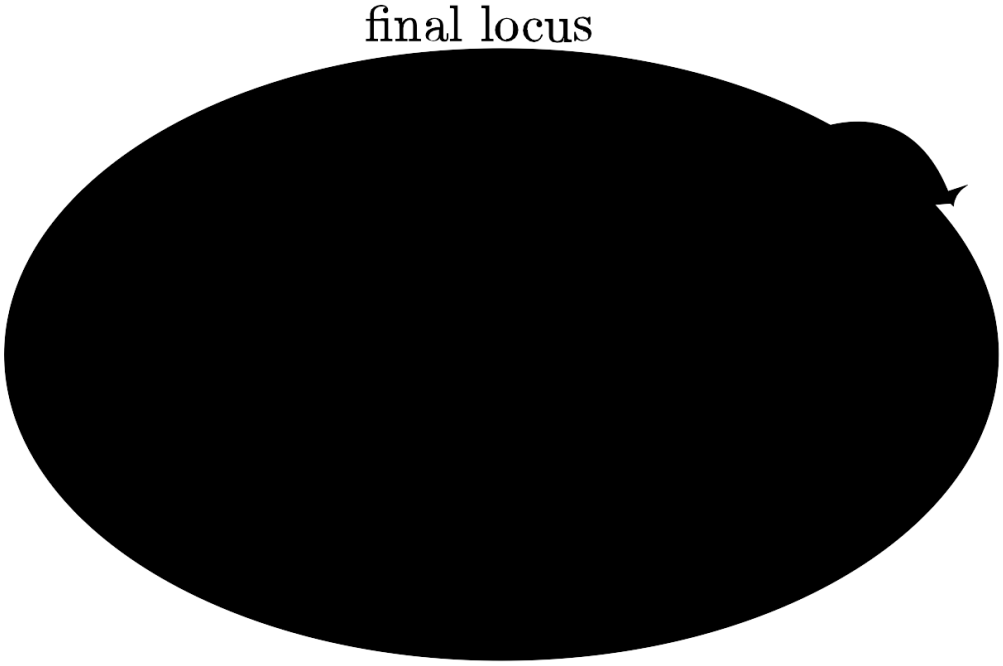

Figure 7: A locus-to-locus transfer constrains endpoints to lie on sets rather than at prescribed points.

Why a Performance Index Is Required

The dynamics and boundary conditions may admit many feasible controls. Therefore, feasibility alone does not identify a unique preferred solution.

A performance index assigns a scalar measure to each admissible process:

$$J = \Phi\big(\boldsymbol{x}(t_0), t_0, \boldsymbol{x}(t_f), t_f, \boldsymbol{q}\big) + \int_{t_0}^{t_f} L\big(\boldsymbol{x}(t), \boldsymbol{u}(t), t, \boldsymbol{q}\big)\,\mathrm{d}t.$$

The endpoint term Φ is often called the *Mayer term*. The integral term is called the *Lagrange term*. Their combination is commonly called the *Bolza form*.

Mayer, Lagrange, and Bolza Forms

Mayer form

A pure endpoint cost has the form

$$J = \Phi\big(\boldsymbol{x}(t_0), t_0, \boldsymbol{x}(t_f), t_f, \boldsymbol{q}\big).$$

Lagrange form

A pure running cost has the form

$$J = \int_{t_0}^{t_f} L(\boldsymbol{x}, \boldsymbol{u}, t, \boldsymbol{q})\, \mathrm{d}t.$$

Bolza form

The general combination is

$$J = \Phi + \int_{t_0}^{t_f} L\, \mathrm{d}t.$$

Fuel-Optimal Interpretation

Suppose spacecraft mass satisfies

$$\dot{m}(t) = -\frac{T(t)}{I_{\mathrm{sp}} g_0},$$

where $T(t)$ is thrust magnitude, I_{sp} is specific impulse, and g_0 is standard gravity.
Integrating,

$$m(t) = m(t_0) - \int_{t_0}^{t} \frac{T(\tau)}{I_{\mathrm{sp}} g_0}\, \mathrm{d}\tau.$$

Minimizing fuel consumption is equivalent to maximizing final mass:

$$\max m(t_f).$$

This can be written as a minimization:

$$\min \left[-m(t_f) \right].$$

More generally,

$$\max F \quad \Longleftrightarrow \quad \min(-F).$$

The General Optimal Control Problem

A standard optimal control problem is:

$$\min_{\boldsymbol{u}(\cdot),\, \boldsymbol{q},\, t_0,\, t_f} J$$

with

$$J = \Phi\big(\boldsymbol{x}(t_0), t_0, \boldsymbol{x}(t_f), t_f, \boldsymbol{q}\big) + \int_{t_0}^{t_f} L\big(\boldsymbol{x}(t), \boldsymbol{u}(t), t, \boldsymbol{q}\big)\,\mathrm{d}t,$$

subject to

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, t, \boldsymbol{q}),$$

and

$$\boldsymbol{\phi}\big(\boldsymbol{x}(t_0), t_0, \boldsymbol{x}(t_f), t_f, \boldsymbol{q}\big) = \boldsymbol{0}.$$

Additional path constraints may also be imposed:

$$\boldsymbol{g}\big(\boldsymbol{x}(t), \boldsymbol{u}(t), t, \boldsymbol{q}\big) \leq \boldsymbol{0}.$$

Why Arbitrary Control Search Is Inadequate

One might attempt to choose candidate controls, integrate the dynamics, evaluate the cost, and retain the best result. This approach is generally ineffective because:

1. the control is an infinite-dimensional function;
2. most arbitrary controls may violate boundary conditions;
3. the number of admissible alternatives is uncountably infinite;
4. random search provides no guarantee of stationarity or optimality.

A systematic variational framework is therefore needed. The next step is to convert the constrained optimal control problem into an augmented calculus-of-variations problem and derive its first-order necessary conditions.

Chapter Summary

The key results are:

1. A free endpoint parameter produces an additional transversality or natural boundary condition.
2. For fixed terminal state and free final time,

$$[L - \dot{x}L_{\dot{x}}]_{t_f} = 0.$$

3. For fixed final time and free terminal state,

$$L_{\dot{x}}(t_f) = 0.$$

4. Boundary terms must be interpreted as products involving endpoint variations. One must determine from the problem statement which factor is constrained to vanish.

5. The Euler–Lagrange equation governs the interior path, while natural and transversality conditions govern the endpoints.
6. Gradients, Hessians, and Jacobians provide the derivative notation required for vector-valued optimal control problems.
7. A general dynamical system can be written as

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, t, \boldsymbol{q}).$$

8. General boundary conditions are expressed as

$$\boldsymbol{\phi}\big(\boldsymbol{x}(t_0), t_0, \boldsymbol{x}(t_f), t_f, \boldsymbol{q}\big) = \mathbf{0}.$$

9. The general Bolza performance index is

$$J = \Phi + \int_{t_0}^{t_f} L \, \mathrm{d}t.$$

10. Dynamics and boundary conditions determine feasibility; the performance index determines preference among feasible processes.

Exercises

1. **Coupled vector Euler–Lagrange equations**

   Consider

   $$J[x,y] = \int_0^T \left[\frac{1}{2}\left(\dot{x}^2 + \dot{y}^2\right) + \alpha x \dot{y} - \frac{\omega^2}{2}\left(x^2 + y^2\right) \right] \mathrm{d}t,$$

   subject to

   $$x(0) = x_0, \qquad y(0) = y_0, \qquad x(T) = x_f, \qquad y(T) = y_f.$$

   (a) Derive the Euler–Lagrange equation for $x(t)$.

   (b) Derive the Euler–Lagrange equation for $y(t)$.

   (c) Write the resulting equations in the matrix form

   $$\ddot{\boldsymbol{q}} + A\dot{\boldsymbol{q}} + B\boldsymbol{q} = \boldsymbol{0}, \qquad \boldsymbol{q} = \begin{bmatrix} x \\ y \end{bmatrix}.$$

   (d) For $\alpha = 0$, solve the complete boundary-value problem.

2. **Weighted minimum-slope trajectory**

   Determine the extremal of

   $$J[x] = \frac{1}{2}\int_0^T e^{\beta t}\dot{x}^2(t)\,\mathrm{d}t$$

   subject to

   $$x(0) = x_0, \qquad x(T) = x_f.$$

   (a) Derive the Euler–Lagrange equation.

   (b) Solve explicitly for $x^*(t)$.

   (c) Evaluate $J[x^*]$.

   (d) Show that the solution approaches the straight-line path as

   $$\beta \to 0.$$

3. **Free terminal state**

   Consider

   $$J[x] = \int_0^T \left(\dot{x}^2 + \omega^2 x^2\right)\,\mathrm{d}t,$$

   subject to

$$x(0) = x_0,$$

where T is fixed and $x(T)$ is free.

(a) Derive the Euler–Lagrange equation.

(b) Derive the natural boundary condition at $t = T$.

(c) Solve the resulting boundary-value problem.

(d) Determine $x(T)$ explicitly in terms of x_0, T, and ω.

4. **Free final time**

Consider

$$J[x, t_f] = \int_0^{t_f} \left[1 + \frac{1}{2}(\dot{x} - at)^2\right] \mathrm{d}t,$$

subject to

$$x(0) = 0, \qquad x(t_f) = x_f,$$

where t_f is free.

(a) Derive the Euler–Lagrange equation.

(b) Determine the general extremal $x(t)$.

(c) Derive the free-final-time condition.

(d) Reduce the unknown final time to a scalar algebraic equation.

(e) State how all positive admissible roots should be checked.

5. **Mass–spring boundary-value problem and singular terminal times**

For

$$L = \frac{1}{2}m\dot{x}^2 - \frac{1}{2}kx^2,$$

consider the endpoint conditions

$$x(0) = x_0, \qquad x(T) = x_f.$$

(a) Derive the Euler–Lagrange equation.

(b) Solve the differential equation for $m > 0$ and $k > 0$.

(c) Determine the constants in the general solution.

(d) Show that a unique solution exists when

$$\sin\left(\sqrt{\frac{k}{m}}\,T\right) \neq 0.$$

(e) Analyze separately the case

$$\sin\left(\sqrt{\frac{k}{m}}\,T\right) = 0$$

and determine when the boundary-value problem has no solution or infinitely many solutions.

6. **First variation and the natural boundary condition**

   Let

$$J[x] = \int_{t_0}^{t_f} L(x, \dot{x}, t)\,\mathrm{d}t,$$

   where t_0, t_f, and $x(t_0)$ are fixed, but $x(t_f)$ is free.

   Starting from

$$x_\epsilon(t) = x(t) + \epsilon\eta(t),$$

   derive the complete first variation. Use integration by parts to show that stationarity requires

$$\frac{\partial L}{\partial x} - \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial L}{\partial \dot{x}}\right) = 0$$

   and

$$\left.\frac{\partial L}{\partial \dot{x}}\right|_{t_f} = 0.$$

7. **Endpoint constrained to a curve**

   Let the initial endpoint be fixed and let the terminal endpoint satisfy

$$\psi(x_f, t_f) = x_f^2 + c^2 t_f^2 - R^2 = 0.$$

   (a) Starting from

$$\delta x_f = \delta x(t_f) + \dot{x}(t_f)\delta t_f,$$

   derive the linearized endpoint constraint

$$\psi_x\,\delta x_f + \psi_t\,\delta t_f = 0.$$

   (b) Compute ψ_x and ψ_t.

   (c) Express $\delta x(t_f)$ in terms of δt_f, assuming $\psi_x \neq 0$.

   (d) Substitute this relation into the terminal contribution

$$\left.\frac{\partial L}{\partial \dot{x}}\right|_{t_f} \delta x(t_f) + L(t_f)\delta t_f$$

   and derive the corresponding scalar transversality relation.

8. **Planar locus-to-locus formulation**

   Let

   $$\boldsymbol{x}(t) = \begin{bmatrix} x(t) \\ y(t) \end{bmatrix}.$$

   The initial state must lie on a circle of radius r_0 centered at (a_0, b_0), and the final state must lie on a circle of radius r_f centered at (a_f, b_f).

   (a) Write the two scalar endpoint constraints.

   (b) Collect them in the vector form

   $$\boldsymbol{\phi}\big(\boldsymbol{x}(t_0), t_0, \boldsymbol{x}(t_f), t_f\big) = \mathbf{0}.$$

   (c) Derive the first-order variations of both endpoint constraints.

   (d) Identify the normal vector to each circle at the corresponding endpoint.

9. **General fixed point-to-point boundary mapping**

   Express

   $$\boldsymbol{x}(t_0) = \boldsymbol{x}_0, \qquad \boldsymbol{x}(t_f) = \boldsymbol{x}_f$$

   in the form

   $$\boldsymbol{\phi}\big(\boldsymbol{x}(t_0), t_0, \boldsymbol{x}(t_f), t_f\big) = \mathbf{0}.$$

   Then:

   (a) determine the dimension of $\boldsymbol{\phi}$ when $\boldsymbol{x} \in \mathbb{R}^n$;

   (b) compute the Jacobians of $\boldsymbol{\phi}$ with respect to $\boldsymbol{x}(t_0)$ and $\boldsymbol{x}(t_f)$;

   (c) write the first-order variation $\delta\boldsymbol{\phi}$;

   (d) show that fixed endpoint states imply $\delta\boldsymbol{x}_0 = \mathbf{0}$ and $\delta\boldsymbol{x}_f = \mathbf{0}$.

10. **Moving upper limit and free-final-time condition**

    Consider

    $$J[x, t_f] = \int_{t_0}^{t_f} L(x, \dot{x}, t)\,\mathrm{d}t,$$

    with fixed t_0, fixed $x(t_0)$, fixed terminal value $x(t_f) = x_f$, and free t_f.

    (a) Use the Leibniz rule to show that variation of the upper limit contributes

    $$L(t_f)\delta t_f.$$

    (b) Derive the complete first variation.

    (c) Use

$$\delta x(t_f) = -\dot{x}(t_f)\delta t_f$$

to eliminate the terminal trajectory variation.

(d) Derive

$$\left[L - \dot{x}\frac{\partial L}{\partial \dot{x}}\right]_{t_f} = 0.$$

(e) Show that this condition is equivalent to

$$H(t_f) = 0, \qquad H = \dot{x}\frac{\partial L}{\partial \dot{x}} - L.$$

Chapter 2: Hamiltonian Formulation and Necessary Conditions

Chapter 2: Hamiltonian Formulation and Necessary Conditions

From dynamic feasibility to Pontryagin's necessary conditions

The variational machinery becomes useful for control only after the system dynamics are built into the optimization problem. This chapter introduces augmentation, costates, endpoint multipliers, and the Hamiltonian, then carries one continuous derivation through endpoint variation, variation of the Hamiltonian integral, and Pontryagin's necessary conditions. The four sections are therefore stages of one argument rather than independent lectures.

Chapter contents

Augmentation, Costates, and the Hamiltonian Formulation

A general optimal control problem contains a performance index, differential equations, endpoint constraints, and possibly path constraints. The objective cannot be minimized independently of these constraints because an unconstrained minimizer of the performance index may violate the dynamics or fail to satisfy the boundary conditions.

The central idea developed in this section is *augmentation*. Each constraint is appended to the performance index using an associated Lagrange multiplier. The resulting augmented functional contains the original constraints explicitly, allowing the entire problem to be treated using the calculus of variations.

This construction introduces two important multiplier variables:

- the time-varying multiplier $\boldsymbol{\lambda}(t)$ associated with the dynamics, called the *costate* or *adjoint*;
- the static multiplier $\boldsymbol{\nu}$ associated with endpoint constraints.

The Hamiltonian

$$H(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\lambda}, t) = L(\boldsymbol{x}, \boldsymbol{u}, t) + \boldsymbol{\lambda}^{\top} \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, t)$$

then emerges naturally as a compact representation of the running cost and dynamic constraint.

General Optimal Control Problem

Consider the Bolza-form problem

$$\min_{\boldsymbol{u}(\cdot),\, t_0,\, t_f} J,$$

where

$$J = \Phi\big(\boldsymbol{x}(t_0), t_0, \boldsymbol{x}(t_f), t_f\big) + \int_{t_0}^{t_f} L\big(\boldsymbol{x}(t), \boldsymbol{u}(t), t\big)\, \mathrm{d}t.$$

The dynamics are

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}\big(\boldsymbol{x}(t), \boldsymbol{u}(t), t\big).$$

The endpoint constraints are

$$\boldsymbol{\phi}\big(\boldsymbol{x}(t_0), t_0, \boldsymbol{x}(t_f), t_f\big) = \boldsymbol{0}.$$

Optional path constraints may also be present:

$$\boldsymbol{c}\big(\boldsymbol{x}(t), \boldsymbol{u}(t), t\big) \le \boldsymbol{0}.$$

For the derivation in this section, path constraints are omitted to avoid introducing complementary-slackness conditions and additional multipliers.

Why the Dynamics Are Constraints

A differential equation is a constraint because it restricts the state derivative at every instant.
Equation () can be written as

$$\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, t) - \dot{\boldsymbol{x}} = \boldsymbol{0}.$$

This form makes the constraint character explicit. For every time t along an admissible trajectory,

$$\dot{\boldsymbol{x}}(t)$$

cannot be chosen arbitrarily. It must equal the right-hand side of the system dynamics.

> **Dynamic feasibility**
> A control may produce a low cost, yet still be inadmissible if the associated state does not satisfy the dynamics or endpoint constraints.

Augmentation with Lagrange Multipliers

For each equality constraint, introduce a Lagrange multiplier.
The dynamic constraint receives a time-varying multiplier

$$\boldsymbol{\lambda}(t) \in \mathbb{R}^n.$$

The endpoint constraint receives a static multiplier

$$\boldsymbol{\nu} \in \mathbb{R}^p.$$

The augmented functional is defined by

$$\begin{aligned} J_a = \Phi\big(\boldsymbol{x}(t_0), t_0, \boldsymbol{x}(t_f), t_f\big) - \boldsymbol{\nu}^\top \boldsymbol{\phi}\big(\boldsymbol{x}(t_0), t_0, \boldsymbol{x}(t_f), t_f\big) \\ + \int_{t_0}^{t_f} \left[L(\boldsymbol{x}, \boldsymbol{u}, t) + \boldsymbol{\lambda}^\top \left(\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, t) - \dot{\boldsymbol{x}}\right)\right] \mathrm{d}t. \end{aligned}$$

On every feasible process,

$$\boldsymbol{f} - \dot{\boldsymbol{x}} = \boldsymbol{0},$$

and

$$\boldsymbol{\phi} = \boldsymbol{0}.$$

Therefore,

$$J_a = J$$

for every feasible solution. In particular,

$$J_a^* = J^*.$$

Why Augmentation Does Not Change the Optimal Cost

The purpose of augmentation is not to modify the physical problem. It is to construct an unconstrained variational representation of the constrained problem.

At a feasible solution,

$$\begin{aligned} \boldsymbol{\nu}^\top \boldsymbol{\phi} &= 0, \\ \boldsymbol{\lambda}^\top (\boldsymbol{f} - \dot{\boldsymbol{x}}) &= 0. \end{aligned}$$

Hence, all augmented terms vanish on the feasible trajectory. The multipliers enlarge the set of unknowns, but they do not alter the original objective value at feasibility.

The Costate and Endpoint Multiplier

Costate
The function $\boldsymbol{\lambda}(t)$ multiplying the dynamic constraint is called the costate or adjoint variable.

The costate is time dependent because the dynamic constraint must hold separately at every time. Each instant has its own local multiplier associated with dynamic feasibility.

Endpoint multiplier
The vector $\boldsymbol{\nu}$ is the Lagrange multiplier associated with the endpoint constraint $\boldsymbol{\phi} = \mathbf{0}$.

Unlike $\boldsymbol{\lambda}(t)$, the multiplier $\boldsymbol{\nu}$ is static because the endpoint constraint is imposed only at the boundary.

Dimensional Bookkeeping

Let

$$\begin{aligned} \boldsymbol{x}(t) &\in \mathbb{R}^n, \\ \boldsymbol{u}(t) &\in \mathbb{R}^m, \\ \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, t) &\in \mathbb{R}^n, \\ \boldsymbol{\phi} &\in \mathbb{R}^p. \end{aligned}$$

Since

$$\boldsymbol{\lambda}^\top (\boldsymbol{f} - \dot{\boldsymbol{x}})$$

must be scalar,

$$\boldsymbol{\lambda}(t) \in \mathbb{R}^n.$$

Similarly, because

$$\boldsymbol{\nu}^\top \boldsymbol{\phi}$$

must be scalar,

$$\boldsymbol{\nu} \in \mathbb{R}^p.$$

The endpoint mapping has the form

$$\boldsymbol{\phi} : \mathbb{R}^n \times \mathbb{R} \times \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^p.$$

Functional Versus Function

The quantity J_a is a functional, not an ordinary function.
It depends on the entire functions

$$\boldsymbol{x}(\cdot), \qquad \boldsymbol{u}(\cdot), \qquad \boldsymbol{\lambda}(\cdot),$$

and returns a scalar.
This distinction determines the correct stationarity operation:

$$\delta J_a = 0.$$

One does not take an ordinary derivative of J_a with respect to a scalar variable unless a specific scalar parameter is being isolated.

Extremum Versus Extremal

An extremum is a point that minimizes or maximizes an ordinary function.
An extremal is a function or trajectory satisfying first-order variational stationarity conditions.
Thus, the augmented optimal control derivation seeks an extremal process:

$$\left(\boldsymbol{x}^*(\cdot), \boldsymbol{u}^*(\cdot), \boldsymbol{\lambda}^*(\cdot)\right).$$

The first-order conditions identify stationary candidates but do not by themselves establish whether the candidate is a minimum or maximum.

Definition of the Hamiltonian

Define the Hamiltonian

$$\boxed{H(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\lambda}, t) = L(\boldsymbol{x}, \boldsymbol{u}, t) + \boldsymbol{\lambda}^\top \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, t)}.$$

Then the augmented functional becomes

$$\begin{aligned} J_a = \Phi - \boldsymbol{\nu}^\top \boldsymbol{\phi} \\ + \int_{t_0}^{t_f} \left[H - \boldsymbol{\lambda}^\top \dot{\boldsymbol{x}} \right] \, \mathrm{d}t. \end{aligned}$$

This is only a change of notation. No new assumption has been introduced.

Quantities That May Vary Independently

Before taking the first variation, identify all independently variable quantities.
For the unrestricted endpoint problem, these are

$$\boldsymbol{x}(\cdot), \quad \boldsymbol{u}(\cdot), \quad \boldsymbol{\lambda}(\cdot), \quad \boldsymbol{\nu}, \quad t_0, \quad t_f, \quad \boldsymbol{x}(t_0), \quad \boldsymbol{x}(t_f).$$

More precisely, the endpoint values and endpoint times are linked by the endpoint variation identities

$$\delta \boldsymbol{x}_0 = \delta \boldsymbol{x}(t_0) + \dot{\boldsymbol{x}}(t_0) \delta t_0,$$

$$\delta \boldsymbol{x}_f = \delta \boldsymbol{x}(t_f) + \dot{\boldsymbol{x}}(t_f) \delta t_f.$$

Thus, the derivation must carefully distinguish:

- path variations;
- endpoint state variations;
- endpoint time variations;
- multiplier variations.

Independent variation does not mean unrelated physics
The endpoint state and endpoint time are physically connected through the trajectory. Their dependence is handled explicitly using endpoint variation identities. After that dependence is removed, the remaining variations can be treated independently.

Variation of the Endpoint Part

Define the endpoint part of the augmented functional as

$$\Psi = \Phi - \boldsymbol{\nu}^\top \boldsymbol{\phi}.$$

Then

$$\delta\Psi = \delta\Phi - \delta\left(\boldsymbol{\nu}^\top \boldsymbol{\phi}\right).$$

We derive these two pieces separately.

Variation of the Mayer Cost

The Mayer cost is

$$\Phi = \Phi\big(\boldsymbol{x}(t_0), t_0, \boldsymbol{x}(t_f), t_f\big).$$

Its first variation is

$$\begin{aligned}\delta\Phi = \frac{\partial\Phi}{\partial\boldsymbol{x}(t_0)}\delta\boldsymbol{x}(t_0) + \frac{\partial\Phi}{\partial t_0}\delta t_0 \\ + \frac{\partial\Phi}{\partial\boldsymbol{x}(t_f)}\delta\boldsymbol{x}(t_f) + \frac{\partial\Phi}{\partial t_f}\delta t_f.\end{aligned}$$

Under the row-gradient convention,

$$\frac{\partial\Phi}{\partial\boldsymbol{x}(t_0)} \in \mathbb{R}^{1\times n},$$

while

$$\delta\boldsymbol{x}(t_0) \in \mathbb{R}^{n\times 1},$$

so their product is scalar.

Variation of the Endpoint Constraint Term

Using the product rule,

$$\delta\left(\boldsymbol{\nu}^\top\boldsymbol{\phi}\right) = \delta\boldsymbol{\nu}^\top\boldsymbol{\phi} + \boldsymbol{\nu}^\top\delta\boldsymbol{\phi}.$$

The first variation of $\boldsymbol{\phi}$ is

$$\begin{aligned}\delta\boldsymbol{\phi} = \frac{\partial\boldsymbol{\phi}}{\partial\boldsymbol{x}(t_0)}\delta\boldsymbol{x}(t_0) + \frac{\partial\boldsymbol{\phi}}{\partial t_0}\delta t_0 \\ + \frac{\partial\boldsymbol{\phi}}{\partial\boldsymbol{x}(t_f)}\delta\boldsymbol{x}(t_f) + \frac{\partial\boldsymbol{\phi}}{\partial t_f}\delta t_f.\end{aligned}$$

The Jacobian dimensions are

$$\frac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{x}(t_0)} \in \mathbb{R}^{p\times n},$$
$$\frac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{x}(t_f)} \in \mathbb{R}^{p\times n}.$$

Therefore,

$$\boldsymbol{\nu}^\top \frac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{x}(t_0)} \delta \boldsymbol{x}(t_0)$$

has dimensions

$$(1\times p)(p\times n)(n\times 1),$$

and is scalar.

Combined Variation of the Endpoint Terms

Substituting (), (), and () into () gives

$$\begin{aligned}\delta\Psi = &\left[\frac{\partial \Phi}{\partial \boldsymbol{x}(t_0)} - \boldsymbol{\nu}^\top \frac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{x}(t_0)}\right] \delta \boldsymbol{x}(t_0)\\ &+ \left[\frac{\partial \Phi}{\partial t_0} - \boldsymbol{\nu}^\top \frac{\partial \boldsymbol{\phi}}{\partial t_0}\right] \delta t_0\\ &+ \left[\frac{\partial \Phi}{\partial \boldsymbol{x}(t_f)} - \boldsymbol{\nu}^\top \frac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{x}(t_f)}\right] \delta \boldsymbol{x}(t_f)\\ &+ \left[\frac{\partial \Phi}{\partial t_f} - \boldsymbol{\nu}^\top \frac{\partial \boldsymbol{\phi}}{\partial t_f}\right] \delta t_f\\ &- \delta \boldsymbol{\nu}^\top \boldsymbol{\phi}.\end{aligned}$$

This completes the variation of all terms outside the integral.

Endpoint variation result
For

$$\Psi = \Phi - \boldsymbol{\nu}^\top \boldsymbol{\phi},$$

the first variation is

$$\delta\Psi = \Psi_{x_0}\delta \boldsymbol{x}(t_0) + \Psi_{t_0}\delta t_0 + \Psi_{x_f}\delta \boldsymbol{x}(t_f) + \Psi_{t_f}\delta t_f - \delta \boldsymbol{\nu}^\top \boldsymbol{\phi},$$

where the shorthand derivatives are understood from ().

Why Endpoint Times Have Two Effects

The endpoint time t_0 appears:

1. directly as an argument of Φ and $\boldsymbol{\phi}$;
2. indirectly through the evaluated state $\boldsymbol{x}(t_0)$.

Therefore, perturbing t_0 can change an endpoint quantity through both channels.
The endpoint relation

$$\delta \boldsymbol{x}_0 = \delta \boldsymbol{x}(t_0) + \dot{\boldsymbol{x}}(t_0)\delta t_0$$

is the mechanism that combines these effects later in the derivation.
The same reasoning applies at t_f.

Perturbation-Parameter Interpretation

A useful way to understand variations is to introduce a scalar perturbation parameter ϵ.
For example,

$$t_0(\epsilon) = t_0 + \epsilon \eta_{t_0}.$$

Then

$$\delta t_0 = \left.\frac{\mathrm{d}t_0(\epsilon)}{\mathrm{d}\epsilon}\right|_{\epsilon=0} = \eta_{t_0}.$$

Similarly,

$$\boldsymbol{x}_\epsilon(t) = \boldsymbol{x}(t) + \epsilon \boldsymbol{\eta}_x(t),$$

so

$$\delta \boldsymbol{x}(t) = \boldsymbol{\eta}_x(t).$$

This construction clarifies that every independently variable quantity must receive its own admissible perturbation.

What Remains for the Full Derivation

The complete first variation of J_a also requires the variation of

$$\int_{t_0}^{t_f} \left[H - \boldsymbol{\lambda}^\top \dot{\boldsymbol{x}}\right] \mathrm{d}t.$$

That derivation must account for:

- moving lower and upper integration limits;
- variations in $\boldsymbol{x}(t)$;
- variations in $\boldsymbol{u}(t)$;

- variations in $\boldsymbol{\lambda}(t)$;
- variation of $\dot{\boldsymbol{x}}(t)$;
- integration by parts;
- endpoint state and time identities.

Once the complete first variation is separated into independent variations, the coefficients yield:

- the state equation;
- the costate equation;
- the stationarity condition;
- the endpoint constraint;
- transversality conditions.

Chapter Summary

The main results are:

1. The state equation is an equality constraint imposed at every time:

$$\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, t) - \dot{\boldsymbol{x}} = \boldsymbol{0}.$$

2. The constrained optimal control problem can be converted into an augmented variational problem using Lagrange multipliers.
3. The costate $\boldsymbol{\lambda}(t)$ is the multiplier associated with the dynamics.
4. The static vector $\boldsymbol{\nu}$ is the multiplier associated with endpoint constraints.
5. The augmented functional is

$$J_a = \Phi - \boldsymbol{\nu}^\top \boldsymbol{\phi} + \int_{t_0}^{t_f} \left[L + \boldsymbol{\lambda}^\top (\boldsymbol{f} - \dot{\boldsymbol{x}}) \right] \, \mathrm{d}t.$$

6. On feasible processes,

$$J_a = J.$$

7. The Hamiltonian is

$$H = L + \boldsymbol{\lambda}^\top \boldsymbol{f}.$$

8. The augmented functional becomes

$$J_a = \Phi - \boldsymbol{\nu}^\top \boldsymbol{\phi} + \int_{t_0}^{t_f} \left[H - \boldsymbol{\lambda}^\top \dot{\boldsymbol{x}} \right] \, \mathrm{d}t.$$

9. The correct stationarity operation is

$$\delta J_a = 0.$$

10. The endpoint variation must include the variations of endpoint states, endpoint times, and the endpoint multiplier.
11. The complete variation of the endpoint terms is given by ().

Connection. After defining the augmented functional and its multipliers, we can begin the central derivation by varying its endpoint contribution.

First Variation of the Augmented Functional

The previous section introduced the augmented optimal-control functional, the costate vector, the endpoint multiplier, and the Hamiltonian. The next step is to compute the first variation of the augmented functional carefully enough that all endpoint and interior terms are retained.

This derivation is delicate for two reasons. First, the initial and final times may vary. Second, the state evaluated at a moving endpoint changes for two distinct reasons: the trajectory may change at a fixed time, and the endpoint time itself may move. Neglecting either contribution produces incorrect transversality conditions.

This section develops the endpoint portion of the derivation and prepares the integral variation that will be completed in the Hamiltonian-integral section.

Learning Objectives
After completing this section, you should be able to:

1. distinguish variations of a trajectory from variations of endpoint values;
2. derive the corrected variation of a Mayer cost with moving initial and final times;
3. derive the variation of vector endpoint constraints;
4. verify all Jacobian and multiplier dimensions;
5. collect endpoint variations in a form suitable for transversality analysis;
6. explain why the endpoint multiplier variation recovers the endpoint constraints.

Problem statement

Consider the optimal-control problem

$$\min_{\boldsymbol{x}(\cdot),\boldsymbol{u}(\cdot),t_0,t_f} \quad J = \Phi\big(\boldsymbol{x}(t_0), t_0, \boldsymbol{x}(t_f), t_f\big) + \int_{t_0}^{t_f} L(\boldsymbol{x}(t), \boldsymbol{u}(t), t)\ \mathrm{d}t,$$

$$\text{subject to} \quad \dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t), t)\,,$$

$$\boldsymbol{\phi}\big(\boldsymbol{x}(t_0), t_0, \boldsymbol{x}(t_f), t_f\big) = \mathbf{0}.$$

The quantities have dimensions

$$\boldsymbol{x}(t) \in \mathbb{R}^n, \quad \boldsymbol{u}(t) \in \mathbb{R}^m, \quad \boldsymbol{\phi} \in \mathbb{R}^q, \quad \boldsymbol{\lambda}(t) \in \mathbb{R}^n, \quad \boldsymbol{\nu} \in \mathbb{R}^q.$$

The endpoint function Φ is scalar-valued, whereas $\boldsymbol{\phi}$ is vector-valued.

Augmented functional

Introduce the costate $\boldsymbol{\lambda}(t)$ and endpoint multiplier $\boldsymbol{\nu}$. The augmented functional is

$$J_a = \Phi\big(\boldsymbol{x}(t_0), t_0, \boldsymbol{x}(t_f), t_f\big) - \boldsymbol{\nu}^\top \boldsymbol{\phi}\big(\boldsymbol{x}(t_0), t_0, \boldsymbol{x}(t_f), t_f\big) + \int_{t_0}^{t_f} \left[L(\boldsymbol{x}, \boldsymbol{u}, t) + \boldsymbol{\lambda}^\top \left(\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, t) - \dot{\boldsymbol{x}}\right)\right] \mathrm{d}t.$$

Define the Hamiltonian

$$H(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\lambda}, t) = L(\boldsymbol{x}, \boldsymbol{u}, t) + \boldsymbol{\lambda}^\top \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, t).$$

Then

$$J_a = \Phi - uvec^\top \boldsymbol{\phi} + \int_{t_0}^{t_f} \left(H - \boldsymbol{\lambda}^\top \dot{\boldsymbol{x}}\right) \mathrm{d}t.$$

Why the augmentation is valid
Along every feasible trajectory, $\dot{\boldsymbol{x}} = \boldsymbol{f}$ and $\boldsymbol{\phi} = \boldsymbol{0}$. Therefore, the multiplier terms vanish, and $J_a = J$. The augmentation changes the analytical representation of the problem but not the objective value along feasible trajectories.

Moving-endpoint variations

Two distinct endpoint state variations

At the initial endpoint, distinguish

- $\delta \boldsymbol{x}(t_0)$: variation of the trajectory evaluated at the fixed nominal time t_0,
- $\delta \boldsymbol{x}_0$: total variation of the actual endpoint state when the endpoint time may also vary.

These quantities satisfy

$$\delta \boldsymbol{x}_0 = \delta \boldsymbol{x}(t_0) + \dot{\boldsymbol{x}}(t_0)\delta t_0.$$

Similarly,

$$\delta \boldsymbol{x}_f = \delta \boldsymbol{x}(t_f) + \dot{\boldsymbol{x}}(t_f)\delta t_f.$$

Equivalently,

$$\delta \boldsymbol{x}(t_0) = \delta \boldsymbol{x}_0 - \dot{\boldsymbol{x}}(t_0)\delta t_0,$$

$$\delta \boldsymbol{x}(t_f) = \delta \boldsymbol{x}_f - \dot{\boldsymbol{x}}(t_f)\delta t_f.$$

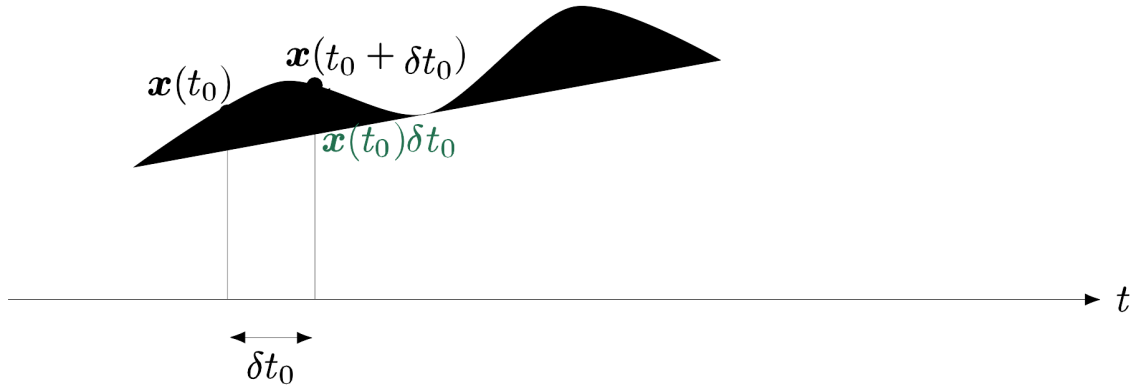

Figure 8: A moving endpoint changes the endpoint state through the time shift as well as through a trajectory variation.

Common omission
Writing only $\delta \boldsymbol{x}_0 = \delta \boldsymbol{x}(t_0)$ is correct only when $\delta t_0 = 0$. When the endpoint time is free, the term $\dot{\boldsymbol{x}}(t_0)\delta t_0$ must be included.

Variation of the Mayer cost

Let

$$\Phi = \Phi\big(\boldsymbol{x}(t_0), t_0, \boldsymbol{x}(t_f), t_f\big)\,.$$

A direct first-order expansion gives

$$\begin{aligned}\delta\Phi = &\frac{\partial \Phi}{\partial \boldsymbol{x}(t_0)}\delta \boldsymbol{x}(t_0) + \frac{\partial \Phi}{\partial \boldsymbol{x}(t_0)}\dot{\boldsymbol{x}}(t_0)\delta t_0 + \frac{\partial \Phi}{\partial t_0}\delta t_0 \\ &+ \frac{\partial \Phi}{\partial \boldsymbol{x}(t_f)}\delta \boldsymbol{x}(t_f) + \frac{\partial \Phi}{\partial \boldsymbol{x}(t_f)}\dot{\boldsymbol{x}}(t_f)\delta t_f + \frac{\partial \Phi}{\partial t_f}\delta t_f.\end{aligned}$$

Using (), and (), the pairs of terms involving the state and endpoint-time shift combine to give

$$\boxed{\delta\Phi = \frac{\partial \Phi}{\partial \boldsymbol{x}(t_0)}\delta \boldsymbol{x}_0 + \frac{\partial \Phi}{\partial t_0}\delta t_0 + \frac{\partial \Phi}{\partial \boldsymbol{x}(t_f)}\delta \boldsymbol{x}_f + \frac{\partial \Phi}{\partial t_f}\delta t_f}\,.$$

Orientation convention

Throughout this text, gradients of scalar functions with respect to vectors are taken as row vectors. Thus,

$$\frac{\partial \Phi}{\partial \boldsymbol{x}(t_0)} \in \mathbb{R}^{1\times n}, \qquad \delta \boldsymbol{x}_0 \in \mathbb{R}^{n\times 1},$$

so their product is scalar.

Interpretation
Equation () has the same form as an ordinary multivariable differential, provided that $\delta \boldsymbol{x}_0$ and $\delta \boldsymbol{x}_f$ are interpreted as total endpoint-state variations rather than fixed-time trajectory variations.

Variation of the endpoint constraints

The endpoint constraints are

$$\boldsymbol{\phi} = \boldsymbol{\phi}\big(\boldsymbol{x}(t_0), t_0, \boldsymbol{x}(t_f), t_f\big) \in \mathbb{R}^q.$$

Following exactly the same moving-endpoint argument used for Φ,

$$\boxed{\delta\boldsymbol{\phi} = \frac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{x}(t_0)}\delta \boldsymbol{x}_0 + \frac{\partial \boldsymbol{\phi}}{\partial t_0}\delta t_0 + \frac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{x}(t_f)}\delta \boldsymbol{x}_f + \frac{\partial \boldsymbol{\phi}}{\partial t_f}\delta t_f}.$$

Because $\boldsymbol{\phi}$ is vector-valued,

$$\frac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{x}(t_0)} \in \mathbb{R}^{q\times n}, \quad \frac{\partial \boldsymbol{\phi}}{\partial t_0} \in \mathbb{R}^{q\times 1},$$
$$\frac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{x}(t_f)} \in \mathbb{R}^{q\times n}, \quad \frac{\partial \boldsymbol{\phi}}{\partial t_f} \in \mathbb{R}^{q\times 1}.$$

Consequently, every term in () belongs to $\mathbb{R}^q$, as required.

Dimension check

For example,

$$\underbrace{\frac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{x}(t_0)}}_{q\times n}\underbrace{\delta \boldsymbol{x}_0}_{n\times 1} \in \mathbb{R}^{q\times 1}.$$

Then

$$\underbrace{\boldsymbol{\nu}^{\mathsf{T}}}_{1\times q}\underbrace{\delta\boldsymbol{\phi}}_{q\times 1} \in \mathbb{R},$$

which is consistent with the fact that $\boldsymbol{\nu}^{\mathsf{T}}\boldsymbol{\phi}$ is scalar.

Vector derivative conventions matter
Using a different Jacobian orientation is possible, but every subsequent expression must be transposed consistently. The convention used here minimizes unnecessary transposes in the first-variation derivation.

Variation of the endpoint multiplier term

Apply the product rule to the scalar product $\boldsymbol{\nu}^{\mathsf{T}}\boldsymbol{\phi}$:

$$\delta\left(\boldsymbol{\nu}^{\mathsf{T}}\boldsymbol{\phi}\right) = \delta\boldsymbol{\nu}^{\mathsf{T}}\boldsymbol{\phi} + \boldsymbol{\nu}^{\mathsf{T}}\delta\boldsymbol{\phi}.$$

Therefore,

$$\delta\left(-\boldsymbol{\nu}^{\mathsf{T}}\boldsymbol{\phi}\right) = -\delta\boldsymbol{\nu}^{\mathsf{T}}\boldsymbol{\phi} - \boldsymbol{\nu}^{\mathsf{T}}\delta\boldsymbol{\phi}.$$

Substituting (),

$$\begin{aligned}\delta\left(-\boldsymbol{\nu}^{\mathsf{T}}\boldsymbol{\phi}\right) = &-\boldsymbol{\nu}^{\mathsf{T}}\frac{\partial\boldsymbol{\phi}}{\partial\boldsymbol{x}(t_0)}\delta\boldsymbol{x}_0 - \boldsymbol{\nu}^{\mathsf{T}}\frac{\partial\boldsymbol{\phi}}{\partial t_0}\delta t_0 \\ &-\boldsymbol{\nu}^{\mathsf{T}}\frac{\partial\boldsymbol{\phi}}{\partial\boldsymbol{x}(t_f)}\delta\boldsymbol{x}_f - \boldsymbol{\nu}^{\mathsf{T}}\frac{\partial\boldsymbol{\phi}}{\partial t_f}\delta t_f - \delta\boldsymbol{\nu}^{\mathsf{T}}\boldsymbol{\phi}.\end{aligned}$$

Because the final term is scalar,

$$\delta\boldsymbol{\nu}^{\mathsf{T}}\boldsymbol{\phi} = \boldsymbol{\phi}^{\mathsf{T}}\delta\boldsymbol{\nu}.$$

Thus the variation with respect to the endpoint multiplier may be written as

$$-\boldsymbol{\phi}^{\mathsf{T}}\delta\boldsymbol{\nu}.$$

Combined endpoint variation

Combine () and (). The variation of the terms outside the integral is

$$\begin{aligned}\delta\left(\Phi - \boldsymbol{\nu}^{\mathsf{T}}\boldsymbol{\phi}\right) = &\left[\frac{\partial\Phi}{\partial\boldsymbol{x}(t_0)} - \boldsymbol{\nu}^{\mathsf{T}}\frac{\partial\boldsymbol{\phi}}{\partial\boldsymbol{x}(t_0)}\right]\delta\boldsymbol{x}_0 \\ &+\left[\frac{\partial\Phi}{\partial t_0} - \boldsymbol{\nu}^{\mathsf{T}}\frac{\partial\boldsymbol{\phi}}{\partial t_0}\right]\delta t_0 \\ &+\left[\frac{\partial\Phi}{\partial\boldsymbol{x}(t_f)} - \boldsymbol{\nu}^{\mathsf{T}}\frac{\partial\boldsymbol{\phi}}{\partial\boldsymbol{x}(t_f)}\right]\delta\boldsymbol{x}_f \\ &+\left[\frac{\partial\Phi}{\partial t_f} - \boldsymbol{\nu}^{\mathsf{T}}\frac{\partial\boldsymbol{\phi}}{\partial t_f}\right]\delta t_f - \boldsymbol{\phi}^{\mathsf{T}}\delta\boldsymbol{\nu}.\end{aligned}$$

This expression should be preserved because it will later be combined with endpoint terms generated by integration by parts in the integral variation.

Endpoint variation coefficients
Define

$$
\begin{aligned}
\boldsymbol{A}_0^{\mathsf{T}} &= \frac{\partial \Phi}{\partial \boldsymbol{x}(t_0)} - \boldsymbol{\nu}^{\mathsf{T}} \frac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{x}(t_0)}, \\
B_0 &= \frac{\partial \Phi}{\partial t_0} - \boldsymbol{\nu}^{\mathsf{T}} \frac{\partial \boldsymbol{\phi}}{\partial t_0}, \\
\boldsymbol{A}_f^{\mathsf{T}} &= \frac{\partial \Phi}{\partial \boldsymbol{x}(t_f)} - \boldsymbol{\nu}^{\mathsf{T}} \frac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{x}(t_f)}, \\
B_f &= \frac{\partial \Phi}{\partial t_f} - \boldsymbol{\nu}^{\mathsf{T}} \frac{\partial \boldsymbol{\phi}}{\partial t_f}.
\end{aligned}
$$

Then

$$
\delta\left(\Phi - \boldsymbol{\nu}^{\mathsf{T}} \boldsymbol{\phi}\right) = \boldsymbol{A}_0^{\mathsf{T}} \delta \boldsymbol{x}_0 + B_0 \delta t_0 + \boldsymbol{A}_f^{\mathsf{T}} \delta \boldsymbol{x}_f + B_f \delta t_f - \boldsymbol{\phi}^{\mathsf{T}} \delta \boldsymbol{\nu}.
$$

Meaning of the multiplier variation

The variation $\delta \boldsymbol{\nu}$ is arbitrary. Therefore, when the total first variation is eventually required to vanish for every admissible variation, the coefficient of $\delta \boldsymbol{\nu}$ must vanish:

$$
\boldsymbol{\phi}\left(\boldsymbol{x}(t_0), t_0, \boldsymbol{x}(t_f), t_f\right) = \boldsymbol{0}.
$$

Thus variation with respect to the multiplier simply recovers the original endpoint constraints.

General Lagrange-multiplier principle
Whenever a constraint is appended to an objective through a multiplier, variation with respect to the multiplier returns the constraint itself. The role of the multiplier is not to change feasibility, but to allow constrained variations to be represented through an unconstrained stationarity calculation.

Preparation for the integral variation

The remaining part of the augmented functional is

$$
I = \int_{t_0}^{t_f} \left(H - \boldsymbol{\lambda}^{\mathsf{T}} \dot{\boldsymbol{x}}\right) \mathrm{d}t.
$$

Its variation must account for

1. motion of the lower integration limit,
2. motion of the upper integration limit,
3. variation of H through $\boldsymbol{x}$, $\boldsymbol{u}$, and $\boldsymbol{\lambda}$,
4. variation of the product $\boldsymbol{\lambda}^{\mathsf{T}} \dot{\boldsymbol{x}}$,
5. integration by parts of the term containing $\delta \dot{\boldsymbol{x}}$.

Using Leibniz's rule,

$$\delta I = \left(H - \boldsymbol{\lambda}^{\mathsf{T}}\dot{\boldsymbol{x}}\right)_{t_f} \delta t_f - \left(H - \boldsymbol{\lambda}^{\mathsf{T}}\dot{\boldsymbol{x}}\right)_{t_0} \delta t_0 + \int_{t_0}^{t_f} \delta\left(H - \boldsymbol{\lambda}^{\mathsf{T}}\dot{\boldsymbol{x}}\right)\,\mathrm{d}t.$$

The interior variation is

$$\delta\left(H - \boldsymbol{\lambda}^{\mathsf{T}}\dot{\boldsymbol{x}}\right) = \delta H - \delta\boldsymbol{\lambda}^{\mathsf{T}}\dot{\boldsymbol{x}} - \boldsymbol{\lambda}^{\mathsf{T}}\delta\dot{\boldsymbol{x}}.$$

Furthermore,

$$\delta H = \frac{\partial H}{\partial \boldsymbol{x}}\delta\boldsymbol{x} + \frac{\partial H}{\partial \boldsymbol{u}}\delta\boldsymbol{u} + \frac{\partial H}{\partial \boldsymbol{\lambda}}\delta\boldsymbol{\lambda}.$$

The final term in () requires integration by parts. the Hamiltonian-integral section will complete this step, combine the resulting endpoint terms with (), and collect the coefficients of all independent variations.

A compact derivation map

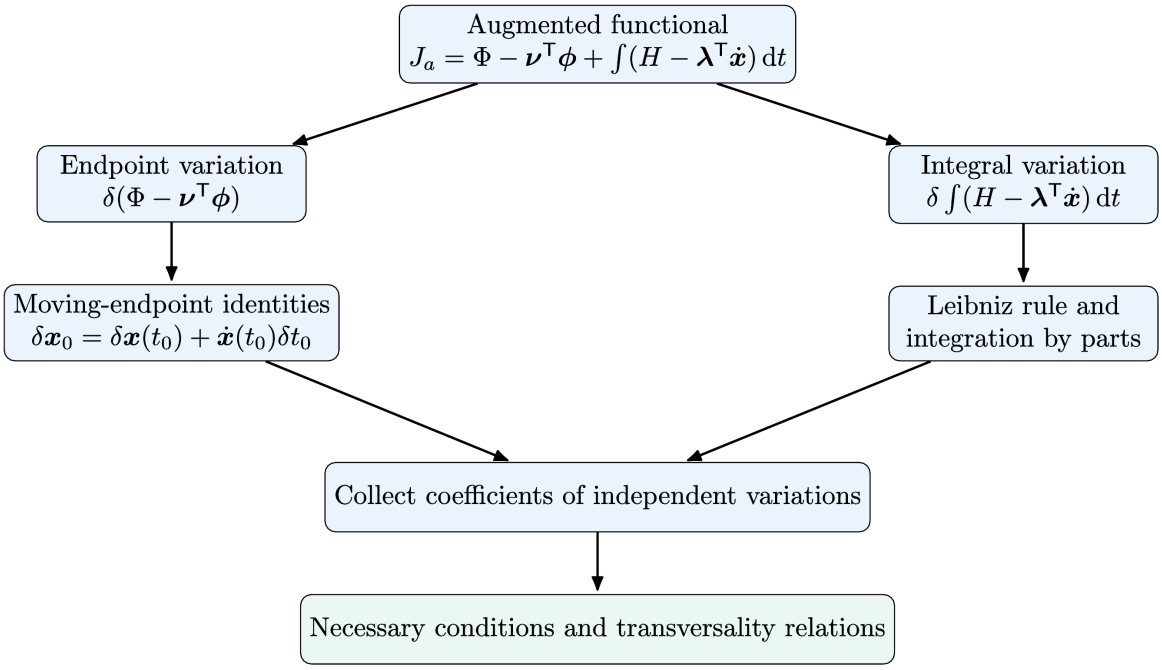

Figure 9: Logical structure of the first-variation derivation. This section completes the left branch and prepares the right branch.

Worked symbolic check

Suppose

$$\Phi = \frac{1}{2}\boldsymbol{x}(t_f)^{\mathsf{T}} Q_f \boldsymbol{x}(t_f) + \alpha t_f,$$

where $Q_f = Q_f^\mathsf{T} \in \mathbb{R}^{n\times n}$, and suppose the endpoint constraint is

$$\boldsymbol{\phi} = C_0\boldsymbol{x}(t_0) + C_f\boldsymbol{x}(t_f) - \boldsymbol{d},$$

with $C_0, C_f \in \mathbb{R}^{q\times n}$.
Then

$$\begin{aligned} \frac{\partial\Phi}{\partial\boldsymbol{x}(t_0)} &= \boldsymbol{0}^\mathsf{T}, & \frac{\partial\Phi}{\partial t_0} &= 0, \\ \frac{\partial\Phi}{\partial\boldsymbol{x}(t_f)} &= \boldsymbol{x}(t_f)^\mathsf{T}Q_f, & \frac{\partial\Phi}{\partial t_f} &= \alpha, \end{aligned}$$

and

$$\begin{aligned} \frac{\partial\boldsymbol{\phi}}{\partial\boldsymbol{x}(t_0)} &= C_0, & \frac{\partial\boldsymbol{\phi}}{\partial t_0} &= \boldsymbol{0}, \\ \frac{\partial\boldsymbol{\phi}}{\partial\boldsymbol{x}(t_f)} &= C_f, & \frac{\partial\boldsymbol{\phi}}{\partial t_f} &= \boldsymbol{0}. \end{aligned}$$

Therefore,

$$\begin{aligned} \delta(\Phi - \boldsymbol{\nu}^\mathsf{T}\boldsymbol{\phi}) = &- \boldsymbol{\nu}^\mathsf{T}C_0\delta\boldsymbol{x}_0 + \left(\boldsymbol{x}(t_f)^\mathsf{T}Q_f - \boldsymbol{\nu}^\mathsf{T}C_f\right)\delta\boldsymbol{x}_f \\ &+ \alpha\delta t_f - \boldsymbol{\phi}^\mathsf{T}\delta\boldsymbol{\nu}. \end{aligned}$$

This result is dimensionally consistent:

$$\boldsymbol{\nu}^\mathsf{T}C_0 \in \mathbb{R}^{1\times n}, \qquad \boldsymbol{x}(t_f)^\mathsf{T}Q_f \in \mathbb{R}^{1\times n}.$$

Connection. The endpoint calculation is only half of the first variation; the Hamiltonian integral must now be varied with equal care.

Variation of the Hamiltonian Integral

the endpoint-variation section derived the first variation of the endpoint portion of the augmented functional. The remaining task is to vary the integral

$$\int_{t_0}^{t_f} \left(H - \boldsymbol{\lambda}^{\mathsf{T}} \dot{\boldsymbol{x}} \right) \, \mathrm{d}t,$$

when the state, control, costate, initial time, and final time are all allowed to vary.

The derivation has four essential steps:

1. apply the Leibniz rule to account for moving integration limits;
2. compute the variation of the Hamiltonian;
3. integrate the term containing $\delta \dot{\boldsymbol{x}}$ by parts;
4. collect the coefficients of the independent variations.

Learning Objectives
After completing this section, you should be able to:

1. apply the Leibniz rule to functionals with variable initial and final times;
2. explain why the independent variable t is not itself varied inside the integral;
3. derive δH in terms of $\delta \boldsymbol{x}$, $\delta \boldsymbol{u}$, and $\delta \boldsymbol{\lambda}$;
4. integrate $\boldsymbol{\lambda}^{\mathsf{T}} \delta \dot{\boldsymbol{x}}$ by parts;
5. convert endpoint trajectory variations into total endpoint variations;
6. obtain the interior coefficients that lead to the state, costate, and stationarity equations.

Starting point

Consider the augmented functional

$$J_a = \Phi - \boldsymbol{\nu}^{\mathsf{T}} \boldsymbol{\phi} + \int_{t_0}^{t_f} \left(H - \boldsymbol{\lambda}^{\mathsf{T}} \dot{\boldsymbol{x}} \right) \, \mathrm{d}t,$$

where

$$H(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\lambda}, t) = L(\boldsymbol{x}, \boldsymbol{u}, t) + \boldsymbol{\lambda}^{\mathsf{T}} \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, t).$$

The endpoint terms $\Phi - \boldsymbol{\nu}^{\mathsf{T}} \boldsymbol{\phi}$ were handled in the endpoint-variation section. Define

$$\mathcal{I} = \int_{t_0}^{t_f} \left(H - \boldsymbol{\lambda}^{\mathsf{T}} \dot{\boldsymbol{x}} \right) \, \mathrm{d}t.$$

This section derives $\delta \mathcal{I}$.

Why t is not varied inside the integral

The symbol t inside the integral is the dummy variable of integration. It is the independent variable with respect to which the trajectories are parameterized. Thus, inside the integral, one varies the functions $\boldsymbol{x}(t)$, $\boldsymbol{u}(t)$, and $\boldsymbol{\lambda}(t)$, but not the dummy variable t itself.

By contrast, t_0 and t_f are scalar parameters that define the lower and upper integration limits. They may be fixed or free and therefore may possess nonzero variations δt_0 and δt_f.

Common conceptual error
The quantities t_0 and t_f are not "values of the variable t that are varied inside the integrand." They are endpoint parameters. Their variations enter through the Leibniz rule, not through a term such as $H_t \delta t$ inside the integral.

Leibniz rule for moving limits

Let

$$F(t) = H(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\lambda}, t) - \boldsymbol{\lambda}^{\mathsf{T}} \dot{\boldsymbol{x}}.$$

Then

$$\mathcal{I} = \int_{t_0}^{t_f} F(t)\, \mathrm{d}t.$$

The first variation is

$$\delta \mathcal{I} = F(t_f)\delta t_f - F(t_0)\delta t_0 + \int_{t_0}^{t_f} \delta F(t)\, \mathrm{d}t.$$

Substituting the definition of F,

$$\begin{aligned} \delta \mathcal{I} = {} & \left[H - \boldsymbol{\lambda}^{\mathsf{T}} \dot{\boldsymbol{x}}\right]_{t_f} \delta t_f - \left[H - \boldsymbol{\lambda}^{\mathsf{T}} \dot{\boldsymbol{x}}\right]_{t_0} \delta t_0 \\ & + \int_{t_0}^{t_f} \delta \left(H - \boldsymbol{\lambda}^{\mathsf{T}} \dot{\boldsymbol{x}}\right) \mathrm{d}t. \end{aligned}$$

Leibniz endpoint contributions
The terms

$$\left[H - \boldsymbol{\lambda}^{\mathsf{T}} \dot{\boldsymbol{x}}\right]_{t_f} \delta t_f, \qquad - \left[H - \boldsymbol{\lambda}^{\mathsf{T}} \dot{\boldsymbol{x}}\right]_{t_0} \delta t_0$$

represent the change in the integral caused solely by moving the upper and lower limits.

Variation of the integrand

The integrand variation is

$$\delta\left(H - \boldsymbol{\lambda}^{\mathsf{T}}\dot{\boldsymbol{x}}\right) = \delta H - \delta\left(\boldsymbol{\lambda}^{\mathsf{T}}\dot{\boldsymbol{x}}\right).$$

Using the product rule,

$$\delta\left(\boldsymbol{\lambda}^{\mathsf{T}}\dot{\boldsymbol{x}}\right) = \delta\boldsymbol{\lambda}^{\mathsf{T}}\dot{\boldsymbol{x}} + \boldsymbol{\lambda}^{\mathsf{T}}\delta\dot{\boldsymbol{x}}.$$

Therefore,

$$\delta\left(H - \boldsymbol{\lambda}^{\mathsf{T}}\dot{\boldsymbol{x}}\right) = \delta H - \delta\boldsymbol{\lambda}^{\mathsf{T}}\dot{\boldsymbol{x}} - \boldsymbol{\lambda}^{\mathsf{T}}\delta\dot{\boldsymbol{x}}.$$

Variation of the Hamiltonian

Because

$$H = H(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\lambda}, t),$$

the variation of H at fixed t is

$$\delta H = H_{\boldsymbol{x}}\delta\boldsymbol{x} + H_{\boldsymbol{u}}\delta\boldsymbol{u} + H_{\boldsymbol{\lambda}}\delta\boldsymbol{\lambda}.$$

Here the derivative convention is

$$H_{\boldsymbol{x}} = \frac{\partial H}{\partial \boldsymbol{x}} \in \mathbb{R}^{1\times n}, \quad H_{\boldsymbol{u}} = \frac{\partial H}{\partial \boldsymbol{u}} \in \mathbb{R}^{1\times m}, \quad H_{\boldsymbol{\lambda}} = \frac{\partial H}{\partial \boldsymbol{\lambda}} \in \mathbb{R}^{1\times n}.$$

Thus each product in () is scalar.
Substitution into () gives

$$\begin{aligned}\delta\left(H - \boldsymbol{\lambda}^{\mathsf{T}}\dot{\boldsymbol{x}}\right) = {} & H_{\boldsymbol{x}}\delta\boldsymbol{x} + H_{\boldsymbol{u}}\delta\boldsymbol{u} + H_{\boldsymbol{\lambda}}\delta\boldsymbol{\lambda} \\ & - \delta\boldsymbol{\lambda}^{\mathsf{T}}\dot{\boldsymbol{x}} - \boldsymbol{\lambda}^{\mathsf{T}}\delta\dot{\boldsymbol{x}}.\end{aligned}$$

Integration by parts

The final term in () contains the derivative of a variation. To express the first variation in terms of $\delta\boldsymbol{x}$ rather than $\delta\dot{\boldsymbol{x}}$, integrate by parts:

$$\int_{t_0}^{t_f} \boldsymbol{\lambda}^{\mathsf{T}}\delta\dot{\boldsymbol{x}}\,\mathrm{d}t = \left[\boldsymbol{\lambda}^{\mathsf{T}}\delta\boldsymbol{x}\right]_{t_0}^{t_f} - \int_{t_0}^{t_f} \dot{\boldsymbol{\lambda}}^{\mathsf{T}}\delta\boldsymbol{x}\,\mathrm{d}t.$$

Therefore,

$$
\begin{aligned}
-\int_{t_0}^{t_f} \boldsymbol{\lambda}^{\mathsf{T}} \delta\dot{\boldsymbol{x}}\,\mathrm{d}t &= -\left[\boldsymbol{\lambda}^{\mathsf{T}} \delta\boldsymbol{x}\right]_{t_0}^{t_f} + \int_{t_0}^{t_f} \dot{\boldsymbol{\lambda}}^{\mathsf{T}} \delta\boldsymbol{x}\,\mathrm{d}t \\
&= \boldsymbol{\lambda}(t_0)^{\mathsf{T}} \delta\boldsymbol{x}(t_0) - \boldsymbol{\lambda}(t_f)^{\mathsf{T}} \delta\boldsymbol{x}(t_f) \\
&\quad + \int_{t_0}^{t_f} \dot{\boldsymbol{\lambda}}^{\mathsf{T}} \delta\boldsymbol{x}\,\mathrm{d}t.
\end{aligned}
$$

Sign discipline
The minus sign multiplies the entire integration-by-parts result. A reliable procedure is to first integrate $\int \boldsymbol{\lambda}^{\mathsf{T}} \delta\dot{\boldsymbol{x}}\,\mathrm{d}t$, and only afterward apply the leading minus sign.

Rearrangement of the interior terms

Substituting () into the integral variation yields

$$
\begin{aligned}
\int_{t_0}^{t_f} \delta\left(H - \boldsymbol{\lambda}^{\mathsf{T}}\dot{\boldsymbol{x}}\right)\,\mathrm{d}t &= \boldsymbol{\lambda}(t_0)^{\mathsf{T}} \delta\boldsymbol{x}(t_0) - \boldsymbol{\lambda}(t_f)^{\mathsf{T}} \delta\boldsymbol{x}(t_f) \\
&\quad + \int_{t_0}^{t_f} \Big[H_{\boldsymbol{x}} \delta\boldsymbol{x} + \dot{\boldsymbol{\lambda}}^{\mathsf{T}} \delta\boldsymbol{x} + H_{\boldsymbol{u}} \delta\boldsymbol{u} \\
&\qquad + H_{\boldsymbol{\lambda}} \delta\boldsymbol{\lambda} - \delta\boldsymbol{\lambda}^{\mathsf{T}} \dot{\boldsymbol{x}} \Big]\,\mathrm{d}t.
\end{aligned}
$$

The state-variation terms combine directly:

$$
H_{\boldsymbol{x}} \delta\boldsymbol{x} + \dot{\boldsymbol{\lambda}}^{\mathsf{T}} \delta\boldsymbol{x} = \left(H_{\boldsymbol{x}} + \dot{\boldsymbol{\lambda}}^{\mathsf{T}}\right) \delta\boldsymbol{x}.
$$

For the costate-variation terms, use the fact that the expression is scalar:

$$
\delta\boldsymbol{\lambda}^{\mathsf{T}} \dot{\boldsymbol{x}} = \dot{\boldsymbol{x}}^{\mathsf{T}} \delta\boldsymbol{\lambda}.
$$

Hence

$$
H_{\boldsymbol{\lambda}} \delta\boldsymbol{\lambda} - \delta\boldsymbol{\lambda}^{\mathsf{T}} \dot{\boldsymbol{x}} = \left(H_{\boldsymbol{\lambda}} - \dot{\boldsymbol{x}}^{\mathsf{T}}\right) \delta\boldsymbol{\lambda}.
$$

Therefore,

$$
\begin{aligned}
\int_{t_0}^{t_f} \delta\left(H - \boldsymbol{\lambda}^{\mathsf{T}}\dot{\boldsymbol{x}}\right)\,\mathrm{d}t &= \boldsymbol{\lambda}(t_0)^{\mathsf{T}} \delta\boldsymbol{x}(t_0) - \boldsymbol{\lambda}(t_f)^{\mathsf{T}} \delta\boldsymbol{x}(t_f) \\
&\quad + \int_{t_0}^{t_f} \Big[\left(H_{\boldsymbol{x}} + \dot{\boldsymbol{\lambda}}^{\mathsf{T}}\right) \delta\boldsymbol{x} + H_{\boldsymbol{u}} \delta\boldsymbol{u} \\
&\qquad + \left(H_{\boldsymbol{\lambda}} - \dot{\boldsymbol{x}}^{\mathsf{T}}\right) \delta\boldsymbol{\lambda} \Big]\,\mathrm{d}t.
\end{aligned}
$$

Complete variation before endpoint conversion

Combining () and () gives

$$\begin{aligned}\delta\mathcal{I} = &\left[H - \boldsymbol{\lambda}^{\mathsf{T}}\dot{\boldsymbol{x}}\right]_{t_f}\delta t_f - \left[H - \boldsymbol{\lambda}^{\mathsf{T}}\dot{\boldsymbol{x}}\right]_{t_0}\delta t_0 \\ &+ \boldsymbol{\lambda}(t_0)^{\mathsf{T}}\delta\boldsymbol{x}(t_0) - \boldsymbol{\lambda}(t_f)^{\mathsf{T}}\delta\boldsymbol{x}(t_f) \\ &+ \int_{t_0}^{t_f}\Big[\left(H_{\boldsymbol{x}} + \dot{\boldsymbol{\lambda}}^{\mathsf{T}}\right)\delta\boldsymbol{x} + H_{\boldsymbol{u}}\delta\boldsymbol{u} \\ &\qquad + \left(H_{\boldsymbol{\lambda}} - \dot{\boldsymbol{x}}^{\mathsf{T}}\right)\delta\boldsymbol{\lambda}\Big]\,\mathrm{d}t.\end{aligned}$$

At this stage, the endpoint state variations are the trajectory variations $\delta\boldsymbol{x}(t_0)$ and $\delta\boldsymbol{x}(t_f)$, not the total endpoint variations $\delta\boldsymbol{x}_0$ and $\delta\boldsymbol{x}_f$.

Conversion to total endpoint variations

The total endpoint variations satisfy

$$\delta\boldsymbol{x}_0 = \delta\boldsymbol{x}(t_0) + \dot{\boldsymbol{x}}(t_0)\delta t_0,$$

$$\delta\boldsymbol{x}_f = \delta\boldsymbol{x}(t_f) + \dot{\boldsymbol{x}}(t_f)\delta t_f.$$

Thus,

$$\delta\boldsymbol{x}(t_0) = \delta\boldsymbol{x}_0 - \dot{\boldsymbol{x}}(t_0)\delta t_0,$$

$$\delta\boldsymbol{x}(t_f) = \delta\boldsymbol{x}_f - \dot{\boldsymbol{x}}(t_f)\delta t_f.$$

Substitute these relations into the boundary terms in ():

$$\begin{aligned}&\boldsymbol{\lambda}(t_0)^{\mathsf{T}}\delta\boldsymbol{x}(t_0) - \boldsymbol{\lambda}(t_f)^{\mathsf{T}}\delta\boldsymbol{x}(t_f) \\ &= \boldsymbol{\lambda}(t_0)^{\mathsf{T}}\delta\boldsymbol{x}_0 - \boldsymbol{\lambda}(t_0)^{\mathsf{T}}\dot{\boldsymbol{x}}(t_0)\delta t_0 \\ &\quad - \boldsymbol{\lambda}(t_f)^{\mathsf{T}}\delta\boldsymbol{x}_f + \boldsymbol{\lambda}(t_f)^{\mathsf{T}}\dot{\boldsymbol{x}}(t_f)\delta t_f.\end{aligned}$$

Now combine these terms with the Leibniz endpoint contributions. At the final time,

$$\begin{aligned}&\left[H - \boldsymbol{\lambda}^{\mathsf{T}}\dot{\boldsymbol{x}}\right]_{t_f}\delta t_f + \boldsymbol{\lambda}(t_f)^{\mathsf{T}}\dot{\boldsymbol{x}}(t_f)\delta t_f \\ &= H(t_f)\delta t_f.\end{aligned}$$

At the initial time,

$$\begin{aligned}&-\left[H - \boldsymbol{\lambda}^{\mathsf{T}}\dot{\boldsymbol{x}}\right]_{t_0}\delta t_0 - \boldsymbol{\lambda}(t_0)^{\mathsf{T}}\dot{\boldsymbol{x}}(t_0)\delta t_0 \\ &= -H(t_0)\delta t_0.\end{aligned}$$

Therefore, all terms involving $\boldsymbol{\lambda}^{\mathsf{T}}\dot{\boldsymbol{x}}\delta t$ cancel exactly.

Final expression for the integral variation

The first variation of the Hamiltonian integral is

$$\boxed{\begin{aligned}\delta\mathcal{I} = {} & \boldsymbol{\lambda}(t_0)^{\mathsf{T}}\delta\boldsymbol{x}_0 - \boldsymbol{\lambda}(t_f)^{\mathsf{T}}\delta\boldsymbol{x}_f - H(t_0)\delta t_0 + H(t_f)\delta t_f \\ & + \int_{t_0}^{t_f} \Big[\left(H_{\boldsymbol{x}} + \dot{\boldsymbol{\lambda}}^{\mathsf{T}}\right)\delta\boldsymbol{x} + H_{\boldsymbol{u}}\delta\boldsymbol{u} \\ & \qquad + \left(H_{\boldsymbol{\lambda}} - \dot{\boldsymbol{x}}^{\mathsf{T}}\right)\delta\boldsymbol{\lambda}\Big]\,\mathrm{d}t.\end{aligned}}$$

This is the principal result of the chapter.

Interpretation
The variation splits naturally into two groups:

1. endpoint contributions multiplying $\delta\boldsymbol{x}_0$, $\delta\boldsymbol{x}_f$, δt_0, and δt_f;
2. interior contributions multiplying the arbitrary function variations $\delta\boldsymbol{x}(t)$, $\delta\boldsymbol{u}(t)$, and $\delta\boldsymbol{\lambda}(t)$.

The vanishing of the coefficients of the interior variations generates the differential necessary conditions. The endpoint terms later combine with the endpoint-cost variation to produce transversality conditions.

Dimension verification

The dimensions of the interior coefficients are

$$\begin{aligned} H_{\boldsymbol{x}} + \dot{\boldsymbol{\lambda}}^{\mathsf{T}} \in \mathbb{R}^{1\times n}, \quad \delta\boldsymbol{x} \in \mathbb{R}^{n\times 1}, \\ H_{\boldsymbol{u}} \in \mathbb{R}^{1\times m}, \quad \delta\boldsymbol{u} \in \mathbb{R}^{m\times 1}, \\ H_{\boldsymbol{\lambda}} - \dot{\boldsymbol{x}}^{\mathsf{T}} \in \mathbb{R}^{1\times n}, \quad \delta\boldsymbol{\lambda} \in \mathbb{R}^{n\times 1}. \end{aligned}$$

Each product is scalar, as required. The endpoint products are also scalar:

$$\boldsymbol{\lambda}(t_0)^{\mathsf{T}}\delta\boldsymbol{x}_0, \quad \boldsymbol{\lambda}(t_f)^{\mathsf{T}}\delta\boldsymbol{x}_f, \quad H(t_0)\delta t_0, \quad H(t_f)\delta t_f.$$

Interior necessary conditions anticipated

Because the interior variations are independent, the coefficients in () lead to

$$H_{\boldsymbol{x}} + \dot{\boldsymbol{\lambda}}^{\mathsf{T}} = \mathbf{0}^{\mathsf{T}},$$

$$H_{\boldsymbol{u}} = \mathbf{0}^{\mathsf{T}},$$

$$H_{\boldsymbol{\lambda}} - \dot{\boldsymbol{x}}^{\mathsf{T}} = \mathbf{0}^{\mathsf{T}}.$$

Equivalently,

$$\dot{\boldsymbol{\lambda}} = -H_{\boldsymbol{x}}^{\mathsf{T}},$$

$$\mathbf{0} = H_{\boldsymbol{u}}^{\mathsf{T}},$$

$$\dot{\boldsymbol{x}} = H_{\boldsymbol{\lambda}}^{\mathsf{T}}.$$

Since

$$H = L + \boldsymbol{\lambda}^{\mathsf{T}} \boldsymbol{f},$$

one has

$$H_{\boldsymbol{\lambda}} = \boldsymbol{f}^{\mathsf{T}},$$

and therefore

$$\dot{\boldsymbol{x}} = H_{\boldsymbol{\lambda}}^{\mathsf{T}} = \boldsymbol{f}.$$

Thus the variation with respect to the costate simply recovers the original state dynamics.

Stationarity is not universally valid at control bounds
The equation $H_{\boldsymbol{u}} = \mathbf{0}^{\mathsf{T}}$ is an interior stationarity condition. It applies when the optimal control lies in the interior of the allowable control set. If the optimum lies on a control boundary, minimization of the Hamiltonian over the admissible control set must be used instead.

Geometric picture of the derivation

Worked symbolic example

Consider the scalar system

$$\dot{x} = ax + bu$$

with running cost

$$L(x, u) = \frac{1}{2} q x^2 + \frac{1}{2} r u^2.$$

The Hamiltonian is

$$H = \frac{1}{2} q x^2 + \frac{1}{2} r u^2 + \lambda (ax + bu).$$

Its derivatives are

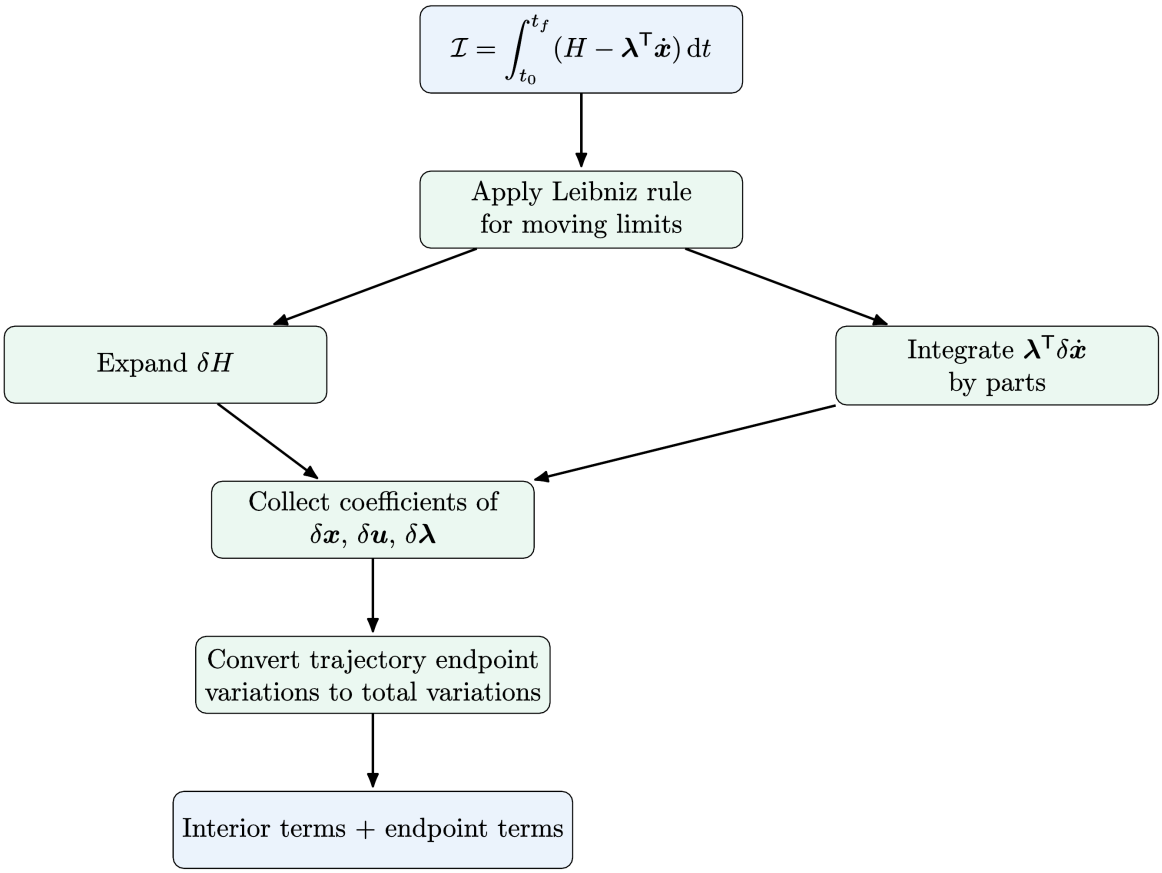

Figure 10: Logical structure of the integral-variation derivation.

$$
\begin{aligned}
H_x &= qx + a\lambda, \\
H_u &= ru + b\lambda, \\
H_\lambda &= ax + bu.
\end{aligned}
$$

The interior part of the first variation is

$$
\int_{t_0}^{t_f} \Big[(qx + a\lambda + \dot{\lambda})\delta x + (ru + b\lambda)\delta u + (ax + bu - \dot{x})\delta\lambda \Big] \, \mathrm{d}t.
$$

The corresponding interior necessary conditions are

$$
\begin{aligned}
\dot{x} &= ax + bu, \\
\dot{\lambda} &= -qx - a\lambda, \\
0 &= ru + b\lambda.
\end{aligned}
$$

If $r > 0$, the stationary control is

$$
u^* = -\frac{b}{r}\lambda.
$$

Connection to the full first variation

the endpoint-variation section obtained

$$
\begin{aligned}
\delta(\Phi - \boldsymbol{\nu}^\mathsf{T}\boldsymbol{\phi}) = &\left(\Phi_{x_0} - \boldsymbol{\nu}^\mathsf{T}\boldsymbol{\phi}_{x_0}\right)\delta\boldsymbol{x}_0 + \left(\Phi_{t_0} - \boldsymbol{\nu}^\mathsf{T}\boldsymbol{\phi}_{t_0}\right)\delta t_0 \\
&+ \left(\Phi_{x_f} - \boldsymbol{\nu}^\mathsf{T}\boldsymbol{\phi}_{x_f}\right)\delta\boldsymbol{x}_f + \left(\Phi_{t_f} - \boldsymbol{\nu}^\mathsf{T}\boldsymbol{\phi}_{t_f}\right)\delta t_f - \boldsymbol{\phi}^\mathsf{T}\delta\boldsymbol{\nu}.
\end{aligned}
$$

Adding () gives the complete first variation

$$
\begin{aligned}
\delta J_a = &\left(\Phi_{x_0} - \boldsymbol{\nu}^\mathsf{T}\boldsymbol{\phi}_{x_0} + \boldsymbol{\lambda}(t_0)^\mathsf{T}\right)\delta\boldsymbol{x}_0 \\
&+ \left(\Phi_{t_0} - \boldsymbol{\nu}^\mathsf{T}\boldsymbol{\phi}_{t_0} - H(t_0)\right)\delta t_0 \\
&+ \left(\Phi_{x_f} - \boldsymbol{\nu}^\mathsf{T}\boldsymbol{\phi}_{x_f} - \boldsymbol{\lambda}(t_f)^\mathsf{T}\right)\delta\boldsymbol{x}_f \\
&+ \left(\Phi_{t_f} - \boldsymbol{\nu}^\mathsf{T}\boldsymbol{\phi}_{t_f} + H(t_f)\right)\delta t_f - \boldsymbol{\phi}^\mathsf{T}\delta\boldsymbol{\nu} \\
&+ \int_{t_0}^{t_f} \Big[\left(H_{\boldsymbol{x}} + \dot{\boldsymbol{\lambda}}^\mathsf{T}\right)\delta\boldsymbol{x} + H_{\boldsymbol{u}}\delta\boldsymbol{u} + \left(H_{\boldsymbol{\lambda}} - \dot{\boldsymbol{x}}^\mathsf{T}\right)\delta\boldsymbol{\lambda} \Big] \, \mathrm{d}t.
\end{aligned}
$$

This expression is the starting point for the necessary-conditions section, where the state equation, costate equation, stationarity condition, endpoint feasibility condition, and transversality conditions are stated systematically.

Common errors

1. **Varying the dummy variable t.** Only t_0 and t_f generate endpoint-time variations.

2. **Omitting Leibniz terms.** Variable limits produce endpoint contributions even before the integrand itself is varied.
3. **Missing the product-rule term.** Both $\delta\boldsymbol{\lambda}^{\mathsf{T}}\dot{\boldsymbol{x}}$ and $\boldsymbol{\lambda}^{\mathsf{T}}\delta\dot{\boldsymbol{x}}$ must appear.
4. **Losing the sign during integration by parts.** The leading minus sign acts on both the endpoint term and the remaining integral.
5. **Confusing $\delta\boldsymbol{x}(t_f)$ with $\delta\boldsymbol{x}_f$.** The latter includes the effect of a moving final time.
6. **Using inconsistent row/column derivative conventions.** Every coefficient–variation product must be scalar.
7. **Declaring $H_{\boldsymbol{u}} = 0$ without checking control bounds.** Stationarity is an interior condition.

Connection. With endpoint and integral terms assembled, the coefficients of the independent variations can finally be interpreted as the necessary conditions.

Pontryagin's Necessary Conditions

the preceding two first-variation sections developed the complete first variation of the augmented optimal-control functional. The endpoint terms were expanded carefully, the Hamiltonian integral was varied using the Leibniz rule, and the term containing $\delta\dot{\boldsymbol{x}}$ was integrated by parts. The remaining task is to interpret the coefficients of the independent variations.

This interpretation produces the equations that every sufficiently regular extremal must satisfy: the state equation, the costate equation, the stationarity or minimum condition, the endpoint constraints, and the transversality conditions. Together, these equations form a coupled two-point boundary-value problem.

Learning Objectives
After completing this section, you should be able to:

1. derive the state, costate, and stationarity conditions from the first variation;
2. distinguish the interior stationarity condition from the full minimum principle;
3. derive endpoint transversality conditions for fixed and free states and times;
4. explain why fixed endpoint variables do not generate additional endpoint equations;
5. formulate the resulting Hamiltonian boundary-value problem;
6. identify when an optimal control must be determined on the boundary of the admissible set;
7. interpret the costate as a sensitivity or shadow-price variable.

Optimal-control problem and augmented functional

Consider the Bolza problem

$$\min_{\boldsymbol{u}(\cdot),\, t_0,\, t_f} \quad J = \Phi\big(\boldsymbol{x}_0, t_0, \boldsymbol{x}_f, t_f\big) + \int_{t_0}^{t_f} L(\boldsymbol{x}, \boldsymbol{u}, t)\,\mathrm{d}t,$$

$$\text{subject to} \quad \dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, t),$$

$$\boldsymbol{\phi}(\boldsymbol{x}_0, t_0, \boldsymbol{x}_f, t_f) = \boldsymbol{0},$$

$$\boldsymbol{u}(t) \in \mathcal{U}(t).$$

Here, $\boldsymbol{x}(t) \in \mathbb{R}^n$, $\boldsymbol{u}(t) \in \mathbb{R}^m$, and $\boldsymbol{\phi} \in \mathbb{R}^q$.
The Hamiltonian is

$$H(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\lambda}, t) = L(\boldsymbol{x}, \boldsymbol{u}, t) + \boldsymbol{\lambda}^{\mathsf{T}} \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, t),$$

and the augmented functional is

$$J_a = \Phi - \boldsymbol{\nu}^\mathsf{T}\boldsymbol{\phi} + \int_{t_0}^{t_f} \left(H - \boldsymbol{\lambda}^\mathsf{T}\dot{\boldsymbol{x}}\right) \mathrm{d}t.$$

The sign convention in () determines the signs in the endpoint conditions developed below.

Complete first variation

After the calculations of the preceding two first-variation sections, the first variation can be written as

$$\begin{aligned}
\delta J_a = \; & \left[\Phi_{\boldsymbol{x}_0} - \boldsymbol{\nu}^\mathsf{T}\boldsymbol{\phi}_{\boldsymbol{x}_0} + \boldsymbol{\lambda}^\mathsf{T}(t_0)\right] \delta\boldsymbol{x}_0 \\
& + \left[\Phi_{t_0} - \boldsymbol{\nu}^\mathsf{T}\boldsymbol{\phi}_{t_0} - H(t_0)\right] \delta t_0 \\
& + \left[\Phi_{\boldsymbol{x}_f} - \boldsymbol{\nu}^\mathsf{T}\boldsymbol{\phi}_{\boldsymbol{x}_f} - \boldsymbol{\lambda}^\mathsf{T}(t_f)\right] \delta\boldsymbol{x}_f \\
& + \left[\Phi_{t_f} - \boldsymbol{\nu}^\mathsf{T}\boldsymbol{\phi}_{t_f} + H(t_f)\right] \delta t_f \\
& - \boldsymbol{\phi}^\mathsf{T}\delta\boldsymbol{\nu} \\
& + \int_{t_0}^{t_f} \left[H_{\boldsymbol{x}} + \dot{\boldsymbol{\lambda}}^\mathsf{T}\right] \delta\boldsymbol{x}\, \mathrm{d}t \\
& + \int_{t_0}^{t_f} H_{\boldsymbol{u}}\, \delta\boldsymbol{u}\, \mathrm{d}t \\
& + \int_{t_0}^{t_f} \left[H_{\boldsymbol{\lambda}} - \dot{\boldsymbol{x}}^\mathsf{T}\right] \delta\boldsymbol{\lambda}\, \mathrm{d}t.
\end{aligned}$$

Subscripts denote row-oriented Jacobians. For example,

$$H_{\boldsymbol{x}} = \frac{\partial H}{\partial \boldsymbol{x}} \in \mathbb{R}^{1\times n}.$$

Why the terms are grouped by variations
The first variation is a linear functional of the perturbations. Each independent variation multiplies a coefficient. For $\delta J_a = 0$ to hold for every admissible perturbation, the relevant coefficients must vanish unless the corresponding variation is constrained to be zero.

Interior necessary conditions

The variations $\delta\boldsymbol{x}(t)$, $\delta\boldsymbol{u}(t)$, and $\delta\boldsymbol{\lambda}(t)$ act within the open interval (t_0, t_f). The fundamental lemma of the calculus of variations implies that their coefficients must satisfy the following conditions almost everywhere.

Costate equation

The coefficient of $\delta\boldsymbol{x}$ is

$$H_{\boldsymbol{x}} + \dot{\boldsymbol{\lambda}}^\mathsf{T}.$$

Therefore,

$$H_x + \dot{\boldsymbol{\lambda}}^\mathsf{T} = \mathbf{0}^\mathsf{T},$$

or, equivalently,

$$\boxed{\dot{\boldsymbol{\lambda}} = -H_x^\mathsf{T} = -\left(\frac{\partial H}{\partial \boldsymbol{x}}\right)^\mathsf{T}.}$$

This is called the *costate equation* or the *adjoint equation*.

Using (),

$$\dot{\boldsymbol{\lambda}} = -L_x^\mathsf{T} - \left(f_x\right)^\mathsf{T} \boldsymbol{\lambda}.$$

Thus, the costate evolves according to both the gradient of the running cost and the local state sensitivity of the dynamics.

Costate
The costate $\boldsymbol{\lambda}(t) \in \mathbb{R}^n$ is the Lagrange-multiplier trajectory associated with the dynamic constraint. It is also interpretable as the first-order sensitivity of the optimal value to a perturbation in the state.

State equation recovered from the Hamiltonian

The coefficient of $\delta\boldsymbol{\lambda}$ yields

$$H_{\boldsymbol{\lambda}} - \dot{\boldsymbol{x}}^\mathsf{T} = \mathbf{0}^\mathsf{T}.$$

Therefore,

$$\boxed{\dot{\boldsymbol{x}} = H_{\boldsymbol{\lambda}}^\mathsf{T}.}$$

Since

$$H = L + \boldsymbol{\lambda}^\mathsf{T} \boldsymbol{f},$$

we have

$$H_{\boldsymbol{\lambda}} = \boldsymbol{f}^\mathsf{T}.$$

Hence,

$$\boxed{\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, t),}$$

which is precisely the original state equation. This condition is not new information; it confirms that the augmented formulation enforces the dynamics.

Stationarity condition

For an unconstrained control, or when the optimal control lies in the interior of the admissible set $\mathcal{U}(t)$, the coefficient of $\delta \boldsymbol{u}$ gives

$$\boxed{H_{\boldsymbol{u}} = \mathbf{0}^{\mathsf{T}}.}$$

Equivalently,

$$\left(\frac{\partial H}{\partial \boldsymbol{u}}\right)^{\mathsf{T}} = \mathbf{0}.$$

If () can be solved uniquely, it defines the control feedback law

$$\boldsymbol{u}^* = \pi(\boldsymbol{x}, \boldsymbol{\lambda}, t).$$

Substitution into the state and costate equations produces a closed Hamiltonian system in $\boldsymbol{x}$ and $\boldsymbol{\lambda}$.

Stationarity is not always valid at a control bound
The equation $H_{\boldsymbol{u}} = 0$ is an interior condition. If $\boldsymbol{u}^*(t)$ lies on the boundary of $\mathcal{U}(t)$, the derivative need not vanish. In that case, the control must be obtained from the Hamiltonian minimization condition.

Hamiltonian minimum condition

For a minimization problem with pointwise admissible control set $\mathcal{U}(t)$, the stronger condition is

$$\boxed{H\big(\boldsymbol{x}^*(t), \boldsymbol{u}^*(t), \boldsymbol{\lambda}^*(t), t\big) = \min_{\boldsymbol{u} \in \mathcal{U}(t)} H\big(\boldsymbol{x}^*(t), \boldsymbol{u}, \boldsymbol{\lambda}^*(t), t\big).}$$

If the minimizing control lies in the interior of $\mathcal{U}(t)$ and the Hamiltonian is differentiable with respect to $\boldsymbol{u}$, then () implies ().

For a scalar bounded control,

$$\mathcal{U} = [u_{\min}, u_{\max}],$$

the minimizer may occur at an endpoint. For example, if

$$H = a(t)u + b(t),$$

then

$$u^*(t) = \begin{cases} u_{\min}, & a(t) > 0, \\ u_{\max}, & a(t) < 0, \end{cases}$$

with additional analysis required when $a(t) = 0$.

Endpoint multiplier condition

The multiplier variation appears as

$$-\boldsymbol{\phi}^{\mathsf{T}}\delta\boldsymbol{\nu}.$$

Because $\delta\boldsymbol{\nu}$ is arbitrary,

$$\boxed{\boldsymbol{\phi}(\boldsymbol{x}_0, t_0, \boldsymbol{x}_f, t_f) = \mathbf{0}.}$$

Thus, variation with respect to the endpoint multiplier recovers the original endpoint constraints.

Transversality conditions

The endpoint variations require more care than the interior variations. An endpoint variable may be fixed, constrained, or free. If it is fixed, its variation is zero and its coefficient need not vanish. If it is free, the variation is arbitrary and the coefficient must vanish.

Initial-state condition

The initial-state contribution is

$$\left[\Phi_{\boldsymbol{x}_0} - \boldsymbol{\nu}^{\mathsf{T}}\boldsymbol{\phi}_{\boldsymbol{x}_0} + \boldsymbol{\lambda}^{\mathsf{T}}(t_0)\right]\delta\boldsymbol{x}_0.$$

Therefore, either

$$\delta\boldsymbol{x}_0 = \mathbf{0},$$

or

$$\boxed{\boldsymbol{\lambda}^{\mathsf{T}}(t_0) = -\Phi_{\boldsymbol{x}_0} + \boldsymbol{\nu}^{\mathsf{T}}\boldsymbol{\phi}_{\boldsymbol{x}_0}.}$$

In column form,

$$\boldsymbol{\lambda}(t_0) = -\Phi_{\boldsymbol{x}_0}^{\mathsf{T}} + \boldsymbol{\phi}_{\boldsymbol{x}_0}^{\mathsf{T}}\boldsymbol{\nu}.$$

Final-state condition

The final-state contribution is

$$\left[\Phi_{\boldsymbol{x}_f} - \boldsymbol{\nu}^{\mathsf{T}}\boldsymbol{\phi}_{\boldsymbol{x}_f} - \boldsymbol{\lambda}^{\mathsf{T}}(t_f)\right]\delta\boldsymbol{x}_f.$$

Therefore, either

$$\delta \boldsymbol{x}_f = \mathbf{0},$$

or

$$\boxed{\boldsymbol{\lambda}^\mathsf{T}(t_f) = \Phi_{\boldsymbol{x}_f} - \boldsymbol{\nu}^\mathsf{T}\boldsymbol{\phi}_{\boldsymbol{x}_f}.}$$

In column form,

$$\boldsymbol{\lambda}(t_f) = \Phi_{\boldsymbol{x}_f}^\mathsf{T} - \boldsymbol{\phi}_{\boldsymbol{x}_f}^\mathsf{T}\boldsymbol{\nu}.$$

Initial-time condition

The initial-time contribution is

$$\left[\Phi_{t_0} - \boldsymbol{\nu}^\mathsf{T}\boldsymbol{\phi}_{t_0} - H(t_0)\right] \delta t_0.$$

Therefore, either

$$\delta t_0 = 0,$$

or

$$\boxed{H(t_0) = \Phi_{t_0} - \boldsymbol{\nu}^\mathsf{T}\boldsymbol{\phi}_{t_0}.}$$

Final-time condition

The final-time contribution is

$$\left[\Phi_{t_f} - \boldsymbol{\nu}^\mathsf{T}\boldsymbol{\phi}_{t_f} + H(t_f)\right] \delta t_f.$$

Therefore, either

$$\delta t_f = 0,$$

or

$$\boxed{H(t_f) = -\Phi_{t_f} + \boldsymbol{\nu}^\mathsf{T}\boldsymbol{\phi}_{t_f}.}$$

Transversality condition
A transversality condition is an endpoint optimality condition obtained from the requirement that the endpoint part of the first variation vanish for all admissible endpoint perturbations.

Fixed and free endpoint cases

Fixed initial state

If $\boldsymbol{x}_0$ is prescribed, then

$$\delta \boldsymbol{x}_0 = \mathbf{0}.$$

No initial costate condition follows from (). The known value of $\boldsymbol{x}(t_0)$ already supplies n boundary conditions.

Free initial state

If $\boldsymbol{x}_0$ is free, then $\delta \boldsymbol{x}_0$ is arbitrary and () must hold.

Fixed final state

If $\boldsymbol{x}_f$ is prescribed, then

$$\delta \boldsymbol{x}_f = \mathbf{0}.$$

No terminal costate equation is generated by this variation. The prescribed terminal state itself supplies the required boundary information.

Free final state with terminal cost

If $\boldsymbol{x}_f$ is free and no endpoint equality constraint is present, then $\boldsymbol{\nu}$ is absent and

$$\boxed{\boldsymbol{\lambda}(t_f) = \Phi_{\boldsymbol{x}_f}^{\mathsf{T}}.}$$

If $\Phi = 0$, this reduces to

$$\boldsymbol{\lambda}(t_f) = \mathbf{0}.$$

Free final time

If t_f is free and no endpoint equality constraint is present, then

$$\boxed{H(t_f) + \Phi_{t_f} = 0.}$$

For an autonomous problem with terminal cost independent of time, this becomes

$$H(t_f) = 0.$$

Endpoint quantity	Variation	Consequence
Fixed $\boldsymbol{x}_0$	$\delta \boldsymbol{x}_0 = 0$	No condition on $\boldsymbol{\lambda}(t_0)$ from this variation
Free $\boldsymbol{x}_0$	Arbitrary $\delta \boldsymbol{x}_0$	Apply ()
Fixed $\boldsymbol{x}_f$	$\delta \boldsymbol{x}_f = 0$	No terminal costate condition from this variation
Free $\boldsymbol{x}_f$	Arbitrary $\delta \boldsymbol{x}_f$	Apply ()
Fixed t_0	$\delta t_0 = 0$	No initial Hamiltonian condition
Free t_0	Arbitrary δt_0	Apply ()
Fixed t_f	$\delta t_f = 0$	No terminal Hamiltonian condition
Free t_f	Arbitrary δt_f	Apply ()

Common endpoint cases and resulting conditions.

Partial freedom in vector endpoints

A vector endpoint may be partly fixed and partly free. Suppose

$$\boldsymbol{x}_f = \begin{bmatrix} \boldsymbol{x}_f^{(F)} \\ \boldsymbol{x}_f^{(R)} \end{bmatrix},$$

where $\boldsymbol{x}_f^{(F)}$ is fixed and $\boldsymbol{x}_f^{(R)}$ is free. Then

$$\delta \boldsymbol{x}_f = \begin{bmatrix} \boldsymbol{0} \\ \delta \boldsymbol{x}_f^{(R)} \end{bmatrix}.$$

Only the coefficient components associated with $\delta \boldsymbol{x}_f^{(R)}$ must vanish. Consequently, transversality conditions are imposed only on the costate components corresponding to free terminal-state components.

Componentwise endpoint logic
Do not label an entire vector endpoint as simply "fixed" or "free" unless all its components share that status. In engineering problems, position may be fixed while velocity, temperature, charge, or orientation remains free.

Hamiltonian system

The state and costate equations form the canonical Hamiltonian system

$$\boxed{\begin{aligned} \dot{\boldsymbol{x}} &= H_{\boldsymbol{\lambda}}^{\mathsf{T}}, \\ \dot{\boldsymbol{\lambda}} &= -H_{\boldsymbol{x}}^{\mathsf{T}}. \end{aligned}}$$

When the stationarity condition can be solved for $\boldsymbol{u}$, substitute $\boldsymbol{u}^* = \boldsymbol{\pi}(\boldsymbol{x}, \boldsymbol{\lambda}, t)$ to obtain

$$\frac{\mathrm{d}}{\mathrm{d}t} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} H_{\boldsymbol{\lambda}}^{\mathsf{T}} \\ -H_{\boldsymbol{x}}^{\mathsf{T}} \end{bmatrix}_{\boldsymbol{u}=\boldsymbol{\pi}(\boldsymbol{x},\boldsymbol{\lambda},t)}.$$

Because both $\boldsymbol{x}$ and $\boldsymbol{\lambda}$ are n-dimensional, this is a $2n$-dimensional differential system.

Two-point boundary-value structure

The boundary data for () are generally divided between t_0 and t_f. For example, a common fixed-initial-state, free-terminal-state problem has

$$\begin{aligned} \boldsymbol{x}(t_0) &= \boldsymbol{x}_0, \\ \boldsymbol{\lambda}(t_f) &= \Phi_{\boldsymbol{x}_f}^{\mathsf{T}}. \end{aligned}$$

The state is specified at the initial time, while the costate is specified at the final time. Therefore, the equations cannot usually be integrated as a conventional initial-value problem.

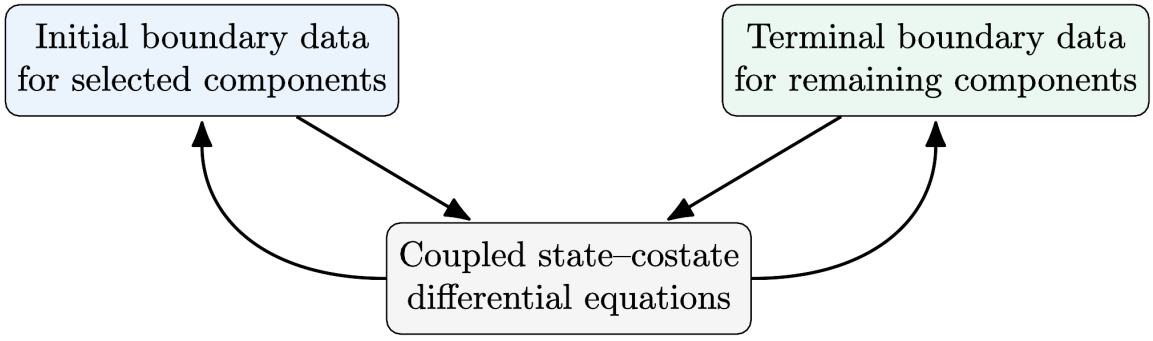

Figure 11: A general optimality system is a two-point boundary-value problem because boundary information is distributed between the initial and terminal times.

Common numerical approaches include:

1. single shooting;
2. multiple shooting;
3. collocation;
4. direct transcription;
5. differential-algebraic boundary-value solvers.

Interpretation of the costate

Let $V(\boldsymbol{x}, t)$ denote the optimal value-to-go from state $\boldsymbol{x}$ at time t. Under suitable smoothness conditions,

$$\boldsymbol{\lambda}(t) = \nabla_{\boldsymbol{x}} V(\boldsymbol{x}^*(t), t).$$

Thus, $\lambda_i(t)$ measures the first-order change in optimal cost resulting from a small perturbation of state component $x_i(t)$.

A large positive λ_i indicates that increasing x_i locally increases the optimal cost. A large negative value indicates that increasing x_i locally decreases it. In economic language, the costate is a shadow price; in engineering design, it is a trajectory-level sensitivity.

Conservation of the Hamiltonian

Along a smooth extremal satisfying the state, costate, and stationarity equations,

$$\frac{\mathrm{d}H}{\mathrm{d}t} = H_t.$$

Therefore, if the problem is autonomous, so that $H_t = 0$, then

$$\boxed{H = \text{constant along the extremal.}}$$

If, in addition, the final time is free and $\Phi_{t_f} = 0$, then $H(t_f) = 0$, implying

$$H(t) = 0 \qquad \text{for all } t \in [t_0, t_f].$$

Worked example: minimum-energy transfer

Consider

$$\min_{u(\cdot)} \quad J = \frac{1}{2}\int_0^T u^2(t)\,\mathrm{d}t,$$

$$\text{subject to} \quad \dot{x} = u,$$

$$x(0) = x_0, \qquad x(T) = x_f,$$

where T is fixed.
The Hamiltonian is

$$H = \frac{1}{2}u^2 + \lambda u.$$

The costate equation is

$$\dot{\lambda} = -H_x = 0,$$

so

$$\lambda(t) = c,$$

where c is constant.
The stationarity condition is

$$H_u = u + \lambda = 0,$$

which gives

$$u^* = -\lambda = -c.$$

Hence,

$$\dot{x} = -c,$$

and

$$x(t) = x_0 - ct.$$

Applying $x(T) = x_f$,

$$c = \frac{x_0 - x_f}{T}.$$

Therefore,

$$\boxed{u^*(t) = \frac{x_f - x_0}{T},}$$

and

$$\boxed{x^*(t) = x_0 + \frac{x_f - x_0}{T}t.}$$

The optimal control is constant, and the state moves linearly from x_0 to x_f.

Why no terminal costate condition was used
The terminal state is fixed, so $\delta x_f = 0$. Consequently, no terminal transversality condition is produced. The unknown constant costate is instead determined indirectly from the prescribed terminal state.

Worked example: free terminal state

Consider

$$\begin{aligned} \min_{u(\cdot)} \quad & J = \frac{1}{2}q\big(x(T) - x_d\big)^2 + \frac{1}{2}\int_0^T ru^2(t)\,\mathrm{d}t, \\ & \dot{x} = ax + bu, \\ & x(0) = x_0, \end{aligned}$$

where T is fixed and $x(T)$ is free.
The Hamiltonian is

$$H = \frac{1}{2}ru^2 + \lambda(ax + bu).$$

The necessary conditions are

$$\begin{aligned}\dot{x} &= ax + bu,\\ \dot{\lambda} &= -a\lambda,\\ 0 &= ru + b\lambda.\end{aligned}$$

Thus,

$$u^* = -\frac{b}{r}\lambda.$$

Because the terminal state is free,

$$\lambda(T) = \Phi_{x_f} = q\big(x(T) - x_d\big).$$

This terminal condition couples the costate to the terminal tracking error.

Compact statement of the necessary conditions

For a smooth extremal of the problem in ()–(), there exist multiplier trajectories $\boldsymbol{\lambda}(t)$ and endpoint multipliers $\boldsymbol{\nu}$ such that

$$\begin{aligned}\dot{\boldsymbol{x}}^* &= H_{\boldsymbol{\lambda}}^{\mathsf{T}},\\ \dot{\boldsymbol{\lambda}}^* &= -H_{\boldsymbol{x}}^{\mathsf{T}},\\ \boldsymbol{u}^*(t) &\in \arg\min_{\boldsymbol{u}\in\mathcal{U}(t)} H(\boldsymbol{x}^*, \boldsymbol{u}, \boldsymbol{\lambda}^*, t),\\ \boldsymbol{\phi}(\boldsymbol{x}_0, t_0, \boldsymbol{x}_f, t_f) &= \mathbf{0},\end{aligned}$$

together with the transversality conditions corresponding to the endpoint variables that are free.

For interior controls, the minimization condition reduces to

$$H_{\boldsymbol{u}} = \mathbf{0}^{\mathsf{T}}.$$

Necessary does not mean sufficient
A trajectory satisfying the state equation, costate equation, stationarity condition, and transversality conditions is an extremal candidate. These conditions alone do not guarantee a minimum. Second-order conditions, convexity, comparison of candidates, or direct evaluation of the objective may still be required.

Common errors

1. **Imposing a costate condition at a fixed state endpoint.** If the endpoint state is fixed, its variation is zero; the corresponding coefficient need not vanish.
2. **Using $H_u = 0$ for a saturated control.** A control on the boundary of its admissible set is determined by Hamiltonian minimization, not necessarily by stationarity.
3. **Losing transposes.** Under row-Jacobian conventions, $H_{\boldsymbol{x}}$ and $H_{\boldsymbol{u}}$ are row vectors, whereas $\dot{\boldsymbol{\lambda}}$ and $\dot{\boldsymbol{x}}$ are columns.
4. **Confusing $\delta\boldsymbol{x}(t_f)$ with $\delta\boldsymbol{x}_f$.** For moving terminal time, these are related by $\delta\boldsymbol{x}_f = \delta\boldsymbol{x}(t_f) + \dot{\boldsymbol{x}}(t_f)\delta t_f$.

5. **Treating all endpoint components identically.** Mixed fixed/free endpoint vectors require component-wise conditions.
6. **Assuming an extremal is automatically optimal.** First-order conditions identify candidates only.

Exercises

1. **Augmented performance index**

   Consider the optimal control problem

   $$\min\ J = \frac{1}{2}\int_0^T \left(x^2 + u^2\right)\,\mathrm{d}t,$$

   subject to

   $$\dot{x} = ax + bu, \qquad x(0) = x_0, \qquad x(T) = x_f.$$

   (a) Construct the augmented functional.

   (b) Define the Hamiltonian.

   (c) Identify the state, control, costate, and endpoint multiplier.

2. **Pontryagin necessary conditions**

   For

   $$\dot{x} = x + u,$$

   with

   $$J = \frac{1}{2}\int_0^T \left(x^2 + u^2\right)\,\mathrm{d}t,$$

   derive

   (a) the Hamiltonian,

   (b) the state equation,

   (c) the costate equation,

   (d) the stationarity condition,

   (e) the complete two-point boundary-value problem.

3. **Linear-quadratic finite-horizon problem**

   Consider

   $$\dot{x} = -2x + u, \qquad x(0) = 1, \qquad x(T) = 0,$$

   with

   $$J = \frac{1}{2}\int_0^T u^2\,dt.$$

   Derive all necessary conditions and eliminate the control to obtain coupled state-costate differential equations.

4. **Double integrator**

   Consider

   $$\dot{x}_1 = x_2, \qquad \dot{x}_2 = u,$$

   with

   $$J = \frac{1}{2}\int_0^T u^2\,dt.$$

   (a) Construct the Hamiltonian.

   (b) Derive the costate equations.

   (c) Derive the optimal control.

   (d) Write the complete boundary-value problem.

5. **Free terminal state**

   Consider

   $$J = \int_0^T \left(\frac{1}{2}u^2 + cx\right)\,dt,$$

   subject to

   $$\dot{x} = u, \qquad x(0) = 0,$$

   where $x(T)$ is free.

   Derive

   (a) the Hamiltonian,

   (b) all necessary conditions,

   (c) the terminal transversality condition,

   (d) the complete boundary-value problem.

6. **Terminal Mayer cost**

   Consider

   $$J = \frac{1}{2}\left(x(T) - x_d\right)^2 + \frac{1}{2}\int_0^T u^2\,dt,$$

   subject to

   $$\dot{x} = u, \qquad x(0) = x_0.$$

   Derive

   (a) the Hamiltonian,

   (b) the costate equation,

(c) the stationarity condition,

(d) the terminal costate condition,

(e) the complete two-point boundary-value problem.

7. **Bounded control**

   Suppose

$$u \in [-2, 2],$$

   and

$$H = \lambda u + \frac{1}{2}u^2.$$

   (a) Determine the unconstrained minimizing control.

   (b) Determine the constrained minimizing control.

   (c) Sketch the optimal control as a function of λ.

8. **Free final time**

   Consider

$$J = \int_0^{t_f} \left(1 + \frac{1}{2}u^2\right) dt,$$

   subject to

$$\dot{x} = u, \qquad x(0) = 0, \qquad x(t_f) = x_f,$$

   where t_f is free.

   Derive

   (a) the Hamiltonian,

   (b) all necessary conditions,

   (c) the free-final-time condition,

   (d) the resulting boundary-value problem.

9. **General endpoint constraints**

   Let

$$\boldsymbol{\phi} = C_0\boldsymbol{x}(t_0) + C_f\boldsymbol{x}(t_f) + \boldsymbol{r}t_f - \boldsymbol{d}.$$

   (a) Compute

$$\delta\boldsymbol{\phi}.$$

   (b) Derive

$$\delta\left(-\boldsymbol{\nu}^{\mathsf{T}}\boldsymbol{\phi}\right).$$

(c) Identify the coefficients multiplying

$$\delta\boldsymbol{x}(t_0), \quad \delta\boldsymbol{x}(t_f), \quad \delta t_f.$$

10. **Complete derivation of Pontryagin's conditions**

Starting from the augmented functional

$$J_a = \Phi - \boldsymbol{\nu}^{\mathsf{T}}\boldsymbol{\phi} + \int_{t_0}^{t_f}\left(H - \boldsymbol{\lambda}^{\mathsf{T}}\dot{\boldsymbol{x}}\right)dt,$$

derive the complete first variation.

Use integration by parts to derive

(a) the state equations,

(b) the costate equations,

(c) the stationarity condition,

(d) the endpoint conditions,

(e) the free-final-time transversality condition.

Chapter 3: Linear–Quadratic, Indirect, and Singular Optimal Control

Chapter 3: Linear–Quadratic, Indirect, and Singular Optimal Control

From Riccati feedback to shooting methods, bang–bang control, and singular arcs

With the necessary conditions in hand, we can study what they reveal and where they become difficult to use. The chapter begins with finite- and infinite-horizon linear–quadratic control, moves to indirect boundary-value methods and Hamiltonian structure, and then examines bang–bang, singular, and path-constrained behavior. The sequence connects an analytically tractable benchmark to the structural and numerical complications of general optimal-control problems.

Chapter contents

Linear–Quadratic Optimal Control and the Riccati Differential Equation

The preceding sections derived the state equation, costate equation, stationarity condition, endpoint constraints, and transversality conditions for a general optimal-control problem. This section applies those necessary conditions to one of the most important analytically tractable problems in control theory: the finite-horizon linear–quadratic regulator (LQR).

The LQR problem is important for two reasons. First, it provides a complete example in which the optimality conditions can be reduced to a computable feedback law. Second, it exposes a central difficulty of indirect optimal control: the state and costate equations form a two-point boundary-value problem, with state information specified at the initial time and costate information specified at the final time.

Learning Objectives
After completing this section, you should be able to:

1. formulate the finite-horizon linear–quadratic optimal-control problem;
2. explain the roles of the matrices Q, R, and S_f;
3. derive the Hamiltonian, costate equation, and stationary control law;
4. assemble the coupled state–costate Hamiltonian system;
5. derive the terminal condition $\boldsymbol{\lambda}(t_f) = S_f \boldsymbol{x}(t_f)$;
6. explain why the necessary conditions form a two-point boundary-value problem;
7. derive the Riccati differential equation using the transformation $\boldsymbol{\lambda} = S\boldsymbol{x} + \boldsymbol{\beta}$;
8. show why the auxiliary function $\boldsymbol{\beta}(t)$ vanishes in the regulator problem;
9. obtain the time-varying feedback law $\boldsymbol{u}^*(t) = -K(t)\boldsymbol{x}^*(t)$;
10. implement the backward–forward solution procedure for finite-horizon LQR.

The finite-horizon LQR problem

Consider the linear time-invariant system

$$\dot{\boldsymbol{x}}(t) = A\boldsymbol{x}(t) + B\boldsymbol{u}(t), \qquad \boldsymbol{x}(0) = \boldsymbol{x}_0,$$

where $\boldsymbol{x}(t) \in \mathbb{R}^n$, $\boldsymbol{u}(t) \in \mathbb{R}^m$, $A \in \mathbb{R}^{n\times n}$, and $B \in \mathbb{R}^{n\times m}$.

The finite-horizon LQR problem is

$$\min_{\boldsymbol{u}(\cdot)} \quad J = \frac{1}{2}\boldsymbol{x}^{\mathsf{T}}(t_f) S_f \boldsymbol{x}(t_f) + \frac{1}{2}\int_0^{t_f} \left[\boldsymbol{x}^{\mathsf{T}}(t) Q \boldsymbol{x}(t) + \boldsymbol{u}^{\mathsf{T}}(t) R \boldsymbol{u}(t)\right] \mathrm{d}t,$$

$$\text{subject to} \quad \dot{\boldsymbol{x}}(t) = A\boldsymbol{x}(t) + B\boldsymbol{u}(t), \qquad \boldsymbol{x}(0) = \boldsymbol{x}_0.$$

The final state is free; it is penalized rather than prescribed.

The standard weighting assumptions are

$$Q = Q^{\mathsf{T}} \succeq 0, \qquad R = R^{\mathsf{T}} \succ 0, \qquad S_f = S_f^{\mathsf{T}} \succeq 0.$$

The matrices may be time varying in the general finite-horizon problem. This section assumes constant A, B, Q, and R to make the derivation transparent; the resulting Riccati equation remains valid in time-varying form after replacing the constant matrices by their time-dependent counterparts.

Interpretation of the quadratic cost

The running cost contains two competing penalties:

$$\frac{1}{2}\boldsymbol{x}^{\mathsf{T}} Q \boldsymbol{x} \qquad \text{and} \qquad \frac{1}{2}\boldsymbol{u}^{\mathsf{T}} R \boldsymbol{u}.$$

The first penalizes undesirable state deviations, while the second penalizes control effort. The terminal term $\frac{1}{2}\boldsymbol{x}^{\mathsf{T}}(t_f) S_f \boldsymbol{x}(t_f)$ penalizes the state remaining at the end of the horizon.

Positive semidefinite state weighting
If $Q \succeq 0$, then

$$\boldsymbol{x}^{\mathsf{T}} Q \boldsymbol{x} \geq 0 \qquad \text{for every } \boldsymbol{x} \in \mathbb{R}^n.$$

Thus, the state contribution to the cost cannot become negative.

Positive definite control weighting
If $R \succ 0$, then

$$\boldsymbol{u}^{\mathsf{T}} R \boldsymbol{u} > 0 \qquad \text{for every nonzero } \boldsymbol{u} \in \mathbb{R}^m.$$

In addition, R is nonsingular, which makes the stationary control law uniquely solvable for $\boldsymbol{u}$.

Controllability and observability assumptions

A common strong set of assumptions is that (A, B) is controllable and $(A, Q^{1/2})$ is observable. For many stabilization results, these may be weakened to stabilizability and detectability, respectively.

Why $Q^{1/2}$ appears
Because $Q \succeq 0$, one may factor it as

$$Q = C^{\mathsf{T}} C,$$

For example, with $C = Q^{1/2}$,

$$\boldsymbol{x}^{\mathsf{T}} Q \boldsymbol{x} = \|C\boldsymbol{x}\|_2^2.$$

Observability or detectability of (A, C) ensures that dynamically relevant state directions are represented in the cost. Writing "(A, Q) observable" is common informal shorthand, but $(A, Q^{1/2})$ is the dimensionally standard statement.

Step 1: Form the Hamiltonian

The Mayer term and running cost are

$$\Phi = \frac{1}{2}\boldsymbol{x}^\mathsf{T}(t_f)S_f\boldsymbol{x}(t_f), \qquad L = \frac{1}{2}\boldsymbol{x}^\mathsf{T}Q\boldsymbol{x} + \frac{1}{2}\boldsymbol{u}^\mathsf{T}R\boldsymbol{u}.$$

The dynamics are

$$\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) = A\boldsymbol{x} + B\boldsymbol{u}.$$

Therefore, the Hamiltonian is

$$H(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\lambda}) = \frac{1}{2}\boldsymbol{x}^\mathsf{T}Q\boldsymbol{x} + \frac{1}{2}\boldsymbol{u}^\mathsf{T}R\boldsymbol{u} + \boldsymbol{\lambda}^\mathsf{T}(A\boldsymbol{x} + B\boldsymbol{u}).$$

Step 2: Derive the costate equation

The costate equation is

$$\dot{\boldsymbol{\lambda}} = -H_{\boldsymbol{x}}^\mathsf{T}.$$

Because $Q = Q^\mathsf{T}$,

$$\frac{\partial}{\partial \boldsymbol{x}}\left(\frac{1}{2}\boldsymbol{x}^\mathsf{T}Q\boldsymbol{x}\right) = \boldsymbol{x}^\mathsf{T}Q.$$

Also,

$$\frac{\partial}{\partial \boldsymbol{x}}\left(\boldsymbol{\lambda}^\mathsf{T}A\boldsymbol{x}\right) = \boldsymbol{\lambda}^\mathsf{T}A.$$

Thus,

$$H_{\boldsymbol{x}} = \boldsymbol{x}^\mathsf{T}Q + \boldsymbol{\lambda}^\mathsf{T}A,$$

and therefore

$$\boxed{\dot{\boldsymbol{\lambda}} = -Q\boldsymbol{x} - A^\mathsf{T}\boldsymbol{\lambda}.}$$

Step 3: Determine the stationary control

The control is unconstrained, so the minimizing control lies in the interior of the admissible control set. The stationarity condition is therefore

$$H_{\boldsymbol{u}} = \mathbf{0}^{\mathsf{T}}.$$

Differentiating () with respect to $\boldsymbol{u}$ gives

$$H_{\boldsymbol{u}} = \boldsymbol{u}^{\mathsf{T}} R + \boldsymbol{\lambda}^{\mathsf{T}} B.$$

Taking the transpose and setting the result equal to zero yields

$$R\boldsymbol{u} + B^{\mathsf{T}}\boldsymbol{\lambda} = \mathbf{0}.$$

Since $R \succ 0$, it is invertible, and the stationary control is

$$\boxed{\boldsymbol{u}^* = -R^{-1}B^{\mathsf{T}}\boldsymbol{\lambda}.}$$

Interior stationarity versus constrained minimization
The equation $H_{\boldsymbol{u}} = 0$ is valid here because the control is unconstrained. If $\boldsymbol{u}$ were restricted to a bounded set, the minimizer could lie on its boundary, in which case $H_{\boldsymbol{u}} = 0$ would not necessarily hold. The correct condition would then be the pointwise Hamiltonian minimum condition.

The coupled state–costate system

Substituting () into the state equation gives

$$\dot{\boldsymbol{x}} = A\boldsymbol{x} - BR^{-1}B^{\mathsf{T}}\boldsymbol{\lambda}.$$

Together with (),

$$\dot{\boldsymbol{\lambda}} = -Q\boldsymbol{x} - A^{\mathsf{T}}\boldsymbol{\lambda}.$$

These equations may be assembled as

$$\boxed{\begin{bmatrix} \dot{\boldsymbol{x}} \\ \dot{\boldsymbol{\lambda}} \end{bmatrix} = \underbrace{\begin{bmatrix} A & -BR^{-1}B^{\mathsf{T}} \\ -Q & -A^{\mathsf{T}} \end{bmatrix}}_{\mathcal{H}} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{\lambda} \end{bmatrix}.}$$

The matrix $\mathcal{H} \in \mathbb{R}^{2n \times 2n}$ is called a Hamiltonian matrix.

Hamiltonian state–costate system
The combined vector

$$\boldsymbol{p}(t) = \begin{bmatrix} \boldsymbol{x}(t) \\ \boldsymbol{\lambda}(t) \end{bmatrix} \in \mathbb{R}^{2n}$$

satisfies

$$\dot{\boldsymbol{p}} = \mathcal{H}\boldsymbol{p}.$$

If $\mathcal{H}$ is constant, then

$$\boldsymbol{p}(t) = e^{\mathcal{H}t}\boldsymbol{p}(0).$$

Although this expression is formally complete, it does not yet solve the optimal-control problem because $\boldsymbol{\lambda}(0)$ is unknown.

Endpoint conditions

Initial endpoint

The initial condition may be represented as the endpoint constraint

$$\boldsymbol{\phi} = \boldsymbol{x}(0) - \boldsymbol{x}_0 = \boldsymbol{0}.$$

Because the initial state is fixed,

$$\delta\boldsymbol{x}(0) = \boldsymbol{0}.$$

Consequently, the initial transversality condition does not provide a useful numerical condition on $\boldsymbol{\lambda}(0)$. Formally, it only relates the initial costate to the endpoint multiplier associated with (); both quantities remain unknown.

Fixed state means no costate boundary condition at that endpoint
A fixed endpoint variation vanishes. Therefore, the coefficient multiplying that variation need not vanish independently. This is why a fully prescribed initial state supplies state data but no additional initial costate data.

Final endpoint

The final state is free, so the terminal transversality condition applies:

$$\boldsymbol{\lambda}(t_f) = \Phi^{\mathsf{T}}_{\boldsymbol{x}(t_f)}.$$

Since

$$\Phi = \frac{1}{2}\boldsymbol{x}^{\mathsf{T}}(t_f)S_f\boldsymbol{x}(t_f)$$

and $S_f = S_f^{\mathsf{T}}$,

$$\boxed{\boldsymbol{\lambda}(t_f) = S_f\boldsymbol{x}(t_f).}$$

The complete boundary data are therefore

$$\boxed{\boldsymbol{x}(0) = \boldsymbol{x}_0, \qquad \boldsymbol{\lambda}(t_f) = S_f\boldsymbol{x}(t_f).}$$

There are n scalar conditions at the initial time and n scalar conditions at the final time, exactly enough for the $2n$ first-order equations in ().

The two-point boundary-value problem

Equations (), and () form a linear two-point boundary-value problem (TPBVP):

$$\dot{\boldsymbol{x}} = A\boldsymbol{x} - BR^{-1}B^{\mathsf{T}}\boldsymbol{\lambda}, \quad \boldsymbol{x}(0) = \boldsymbol{x}_0,$$

$$\dot{\boldsymbol{\lambda}} = -Q\boldsymbol{x} - A^{\mathsf{T}}\boldsymbol{\lambda}, \quad \boldsymbol{\lambda}(t_f) = S_f\boldsymbol{x}(t_f).$$

The difficulty is structural: the full vector $[\boldsymbol{x}^{\mathsf{T}}, \boldsymbol{\lambda}^{\mathsf{T}}]^{\mathsf{T}}$ is not known at either endpoint.

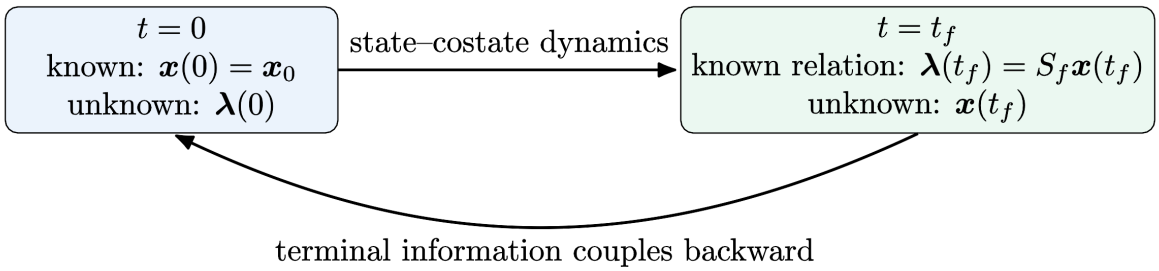

Figure 12: Boundary information for finite-horizon LQR is split between the initial state and terminal costate relation.

A direct matrix-exponential representation is possible. Partition the state-transition matrix as

$$e^{\mathcal{H}t_f} = \begin{bmatrix} \Psi_{11} & \Psi_{12} \\ \Psi_{21} & \Psi_{22} \end{bmatrix}.$$

Then

$$\begin{bmatrix} \boldsymbol{x}(t_f) \\ \boldsymbol{\lambda}(t_f) \end{bmatrix} = \begin{bmatrix} \Psi_{11} & \Psi_{12} \\ \Psi_{21} & \Psi_{22} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_0 \\ \boldsymbol{\lambda}(0) \end{bmatrix}.$$

Using $\boldsymbol{\lambda}(t_f) = S_f \boldsymbol{x}(t_f)$ gives

$$\left(\Psi_{22} - S_f \Psi_{12}\right) \boldsymbol{\lambda}(0) = \left(S_f \Psi_{11} - \Psi_{21}\right) \boldsymbol{x}_0.$$

If the matrix on the left is nonsingular, () determines $\boldsymbol{\lambda}(0)$. The Riccati transformation provides a more useful feedback-oriented representation.

The Riccati transformation

Motivated by the linear structure of the problem, introduce

$$\boxed{\boldsymbol{\lambda}(t) = S(t)\boldsymbol{x}(t) + \boldsymbol{\beta}(t),}$$

where $S(t) \in \mathbb{R}^{n \times n}$ and $\boldsymbol{\beta}(t) \in \mathbb{R}^n$ are to be determined.

This transformation initially loses no generality because $\boldsymbol{\beta}(t)$ is an unrestricted vector function. Its purpose is to select $S(t)$ so that the dynamics of $\boldsymbol{\beta}$ decouple from the state.

Differentiating () gives

$$\dot{\boldsymbol{\lambda}} = \dot{S}\boldsymbol{x} + S\dot{\boldsymbol{x}} + \dot{\boldsymbol{\beta}}.$$

Using

$$\dot{\boldsymbol{x}} = A\boldsymbol{x} - BR^{-1}B^{\mathsf{T}}(S\boldsymbol{x} + \boldsymbol{\beta}),$$

we obtain

$$\begin{aligned}\dot{\boldsymbol{\lambda}} = &\left(\dot{S} + SA - SBR^{-1}B^{\mathsf{T}}S\right)\boldsymbol{x} \\ &+ \dot{\boldsymbol{\beta}} - SBR^{-1}B^{\mathsf{T}}\boldsymbol{\beta}.\end{aligned}$$

On the other hand, the costate equation gives

$$\begin{aligned}\dot{\boldsymbol{\lambda}} &= -Q\boldsymbol{x} - A^{\mathsf{T}}(S\boldsymbol{x} + \boldsymbol{\beta}) \\ &= \left(-Q - A^{\mathsf{T}}S\right)\boldsymbol{x} - A^{\mathsf{T}}\boldsymbol{\beta}.\end{aligned}$$

Equating (), and () yields

$$\begin{aligned}\dot{\boldsymbol{\beta}} = &\left[-\dot{S} - SA - A^{\mathsf{T}}S + SBR^{-1}B^{\mathsf{T}}S - Q\right]\boldsymbol{x} \\ &+ \left(SBR^{-1}B^{\mathsf{T}} - A^{\mathsf{T}}\right)\boldsymbol{\beta}.\end{aligned}$$

Define the matrix residual

$$\Gamma(S) = -\dot{S} - SA - A^{\mathsf{T}}S + SBR^{-1}B^{\mathsf{T}}S - Q.$$

Then

$$\dot{\boldsymbol{\beta}} = \Gamma(S)\boldsymbol{x} + \left(SBR^{-1}B^{\mathsf{T}} - A^{\mathsf{T}}\right)\boldsymbol{\beta}.$$

The Riccati differential equation

Choose $S(t)$ so that

$$\Gamma(S) = 0.$$

This eliminates the state forcing from the $\boldsymbol{\beta}$ equation and gives

$$\boxed{-\dot{S} = A^{\mathsf{T}}S + SA - SBR^{-1}B^{\mathsf{T}}S + Q.}$$

Equivalently,

$$\boxed{\dot{S} = -SA - A^{\mathsf{T}}S + SBR^{-1}B^{\mathsf{T}}S - Q.}$$

Equation () is the finite-horizon Riccati differential equation (RDE).

Riccati differential equation
The RDE is nonlinear in S because of the quadratic matrix term

$$SBR^{-1}B^{\mathsf{T}}S.$$

It is integrated backward from the terminal condition

$$S(t_f) = S_f.$$

Terminal conditions for the Riccati matrix and auxiliary vector

At $t = t_f$, the transformation () gives

$$\boldsymbol{\lambda}(t_f) = S(t_f)\boldsymbol{x}(t_f) + \boldsymbol{\beta}(t_f).$$

The transversality condition requires

$$\boldsymbol{\lambda}(t_f) = S_f\boldsymbol{x}(t_f).$$

A natural choice consistent for every terminal state is therefore

$$\boxed{S(t_f) = S_f, \qquad \boldsymbol{\beta}(t_f) = \mathbf{0}.}$$

Once S satisfies the RDE, () reduces to

$$\dot{\boldsymbol{\beta}} = \left(SBR^{-1}B^{\mathsf{T}} - A^{\mathsf{T}}\right)\boldsymbol{\beta}.$$

This is a homogeneous linear differential equation. Because $\boldsymbol{\beta}(t_f) = 0$, uniqueness implies

$$\boxed{\boldsymbol{\beta}(t) \equiv \mathbf{0} \qquad \text{for all } t \in [0, t_f].}$$

Hence,

$$\boxed{\boldsymbol{\lambda}(t) = S(t)\boldsymbol{x}(t).}$$

When the auxiliary vector does not vanish
The zero result is specific to the regulator problem with no affine dynamics, no reference-tracking terms, and no linear terms in the cost. In tracking and affine optimal-control problems, $\boldsymbol{\beta}(t)$ generally satisfies a nonhomogeneous terminal-value problem and does not vanish.

Optimal feedback law

Substituting () into () gives

$$\boldsymbol{u}^*(t) = -R^{-1}B^{\mathsf{T}}S(t)\boldsymbol{x}^*(t).$$

Define the time-varying gain

$$\boxed{K(t) = R^{-1}B^{\mathsf{T}}S(t).}$$

Then

$$\boxed{\boldsymbol{u}^*(t) = -K(t)\boldsymbol{x}^*(t).}$$

The optimal state satisfies the closed-loop equation

$$\boxed{\dot{\boldsymbol{x}}^*(t) = [A - BK(t)]\,\boldsymbol{x}^*(t), \qquad \boldsymbol{x}^*(0) = \boldsymbol{x}_0.}$$

Backward–forward computational procedure

The Riccati formulation converts the original TPBVP into two sequential initial/terminal-value integrations:

1. Specify A, B, Q, R, S_f, t_f, and $\boldsymbol{x}_0$.
2. Integrate the RDE

$$-\dot{S} = A^{\mathsf{T}}S + SA - SBR^{-1}B^{\mathsf{T}}S + Q$$

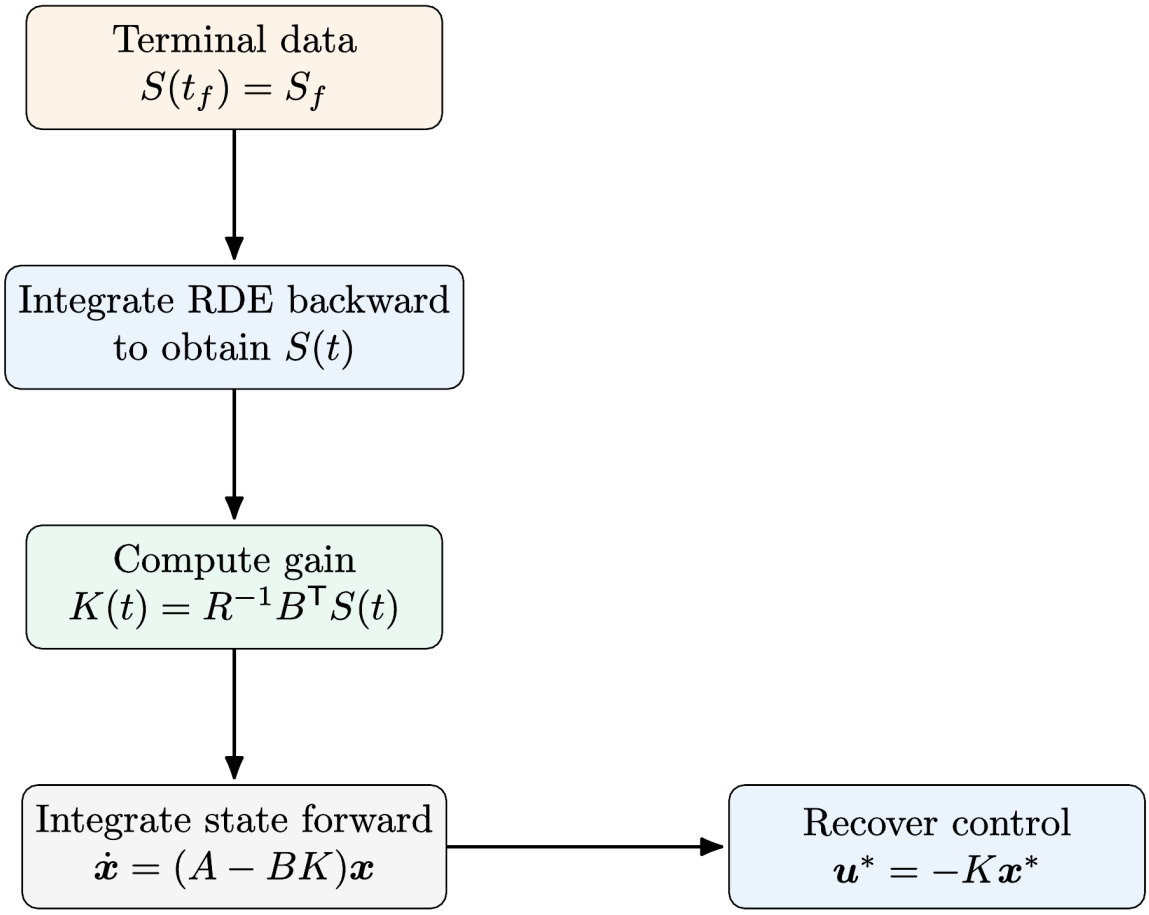

Figure 13: Backward–forward computational workflow for finite-horizon linear–quadratic optimal control.

backward from $S(t_f) = S_f$ to $t = 0$.

3. Compute $K(t) = R^{-1}B^{\mathsf{T}}S(t)$ over the horizon.
4. Integrate

$$\dot{\boldsymbol{x}}^* = [A - BK(t)]\boldsymbol{x}^*, \qquad \boldsymbol{x}^*(0) = \boldsymbol{x}_0,$$

forward from 0 to t_f.

5. Recover

$$\boldsymbol{u}^*(t) = -K(t)\boldsymbol{x}^*(t), \qquad \boldsymbol{\lambda}^*(t) = S(t)\boldsymbol{x}^*(t).$$

6. Verify the state equation, costate equation, stationarity condition, and terminal condition numerically.

Value-function interpretation

For the LQR problem, the optimal cost-to-go has the quadratic form

$$V(\boldsymbol{x}, t) = \frac{1}{2}\boldsymbol{x}^{\mathsf{T}}S(t)\boldsymbol{x}.$$

Its state gradient is

$$V_{\boldsymbol{x}}^{\mathsf{T}} = S(t)\boldsymbol{x}.$$

Thus,

$$\boldsymbol{\lambda}(t) = V_{\boldsymbol{x}}^{\mathsf{T}}(\boldsymbol{x}^*(t), t) = S(t)\boldsymbol{x}^*(t).$$

The Riccati matrix therefore encodes the curvature of the optimal value function and maps the state to its associated costate sensitivity.

Substitution of () into the Hamilton–Jacobi–Bellman equation produces the same RDE as (). This equivalence connects the indirect costate derivation to dynamic programming.

Scalar worked example

Consider

$$\dot{x} = ax + bu, \quad x(0) = x_0,$$

$$J = \frac{1}{2}s_f x^2(t_f) + \frac{1}{2}\int_0^{t_f} \left(qx^2 + ru^2\right)\,\mathrm{d}t,$$

where $q \geq 0$, $r > 0$, and $s_f \geq 0$.

The Hamiltonian is

$$H = \frac{1}{2}qx^2 + \frac{1}{2}ru^2 + \lambda(ax + bu).$$

The stationarity condition gives

$$ru + b\lambda = 0 \quad \Longrightarrow \quad u^* = -\frac{b}{r}\lambda.$$

The costate equation is

$$\dot{\lambda} = -qx - a\lambda, \qquad \lambda(t_f) = s_f x(t_f).$$

Let $\lambda = sx$. Then

$$-\dot{s} = 2as - \frac{b^2}{r}s^2 + q, \qquad s(t_f) = s_f.$$

The optimal feedback law is

$$\boxed{u^*(t) = -\frac{b}{r}s(t)x^*(t).}$$

The closed-loop state equation is

$$\dot{x}^*(t) = \left(a - \frac{b^2}{r}s(t)\right)x^*(t).$$

Typographical distinction
The optimal scalar control is u^*, not the endpoint multiplier v. In handwritten derivations, these symbols can look similar; preserving the distinction prevents confusion between the control and a Lagrange multiplier.

Why this example matters

The LQR derivation illustrates several general lessons:

1. The necessary conditions naturally produce a TPBVP.
2. Fixed endpoint states do not automatically provide costate values at the same endpoint.
3. Free terminal states produce terminal costate conditions through transversality.
4. An appropriate transformation can decouple a state–costate system.
5. The Riccati equation converts an open-loop indirect solution into a feedback representation.
6. Analytical structure reduces computational burden and clarifies what a numerical solver is doing.

The historical importance of such transformations is easy to underestimate. Before modern numerical software, solving a nonlinear or even large linear TPBVP could be prohibitively expensive. Analytical reductions

such as the Riccati formulation made many control problems computationally feasible and remain valuable for interpretation, verification, and real-time implementation.

Common errors

Error 1: Omitting the factor $\frac{1}{2}$ inconsistently
The factor $\frac{1}{2}$ is used so that differentiating a symmetric quadratic form produces $Q\boldsymbol{x}$ or $R\boldsymbol{u}$ rather than $2Q\boldsymbol{x}$ or $2R\boldsymbol{u}$. Omitting it is acceptable only if all subsequent derivatives are changed consistently.

Error 2: Treating $Q \succeq 0$ as invertible
Positive semidefinite matrices may be singular. The derivation requires R^{-1}, not Q^{-1}. This is why R is assumed positive definite.

Error 3: Assigning an arbitrary initial costate
The initial state is known, but $\boldsymbol{\lambda}(0)$ is not. Integrating the Hamiltonian system forward with an arbitrary $\boldsymbol{\lambda}(0)$ generally violates the terminal condition.

Error 4: Integrating the Riccati equation in the wrong direction
For the finite-horizon problem, S is specified at t_f, so the RDE must be integrated backward in time.

Error 5: Dropping S from the state equation
After $\boldsymbol{\lambda} = S\boldsymbol{x}$, the closed-loop state equation is

$$\dot{\boldsymbol{x}} = (A - BR^{-1}B^{\mathsf{T}}S)\boldsymbol{x},$$

not $(A - BR^{-1}B^{\mathsf{T}})\boldsymbol{x}$.

Error 6: Assuming the auxiliary vector is zero too early
The conclusion $\boldsymbol{\beta} \equiv 0$ follows from two facts: its dynamics are homogeneous after choosing S through the RDE, and its terminal condition is zero.

Connection. The finite-horizon Riccati equation points naturally toward steady feedback laws, which leads to infinite-horizon LQR and the algebraic Riccati equation.

Infinite-Horizon Linear Quadratic Regulation

This section continues the study of the linear quadratic optimal control problem. The main goals are:

1. to complete the interpretation of the finite-horizon Riccati solution;
2. to derive the infinite-horizon linear quadratic regulator;
3. to obtain the algebraic Riccati equation;
4. to explain the stabilizing Riccati solution;
5. to introduce the shooting method for two-point boundary-value problems; and
6. to show why shooting can become numerically unstable.

Finite-Horizon Linear Quadratic Optimal Control

Consider the finite-horizon problem

$$\min_{\boldsymbol{u}(\cdot)} J = \frac{1}{2}\boldsymbol{x}^{\mathsf{T}}(t_f)\boldsymbol{S}_f\boldsymbol{x}(t_f) + \frac{1}{2}\int_0^{t_f} \left[\boldsymbol{x}^{\mathsf{T}}\boldsymbol{Q}\boldsymbol{x} + \boldsymbol{u}^{\mathsf{T}}\boldsymbol{R}\boldsymbol{u}\right]\,\mathrm{d}t,$$

subject to

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u}, \qquad \boldsymbol{x}(0) = \boldsymbol{x}_0.$$

The usual assumptions are

$$\boldsymbol{Q} = \boldsymbol{Q}^{\mathsf{T}} \succeq 0, \qquad \boldsymbol{R} = \boldsymbol{R}^{\mathsf{T}} \succ 0,$$

together with controllability of $(\boldsymbol{A}, \boldsymbol{B})$ and observability or detectability of the relevant state-weighted pair. The Hamiltonian is

$$H = \frac{1}{2}\boldsymbol{x}^{\mathsf{T}}\boldsymbol{Q}\boldsymbol{x} + \frac{1}{2}\boldsymbol{u}^{\mathsf{T}}\boldsymbol{R}\boldsymbol{u} + \boldsymbol{\lambda}^{\mathsf{T}}(\boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u}).$$

Stationarity gives

$$\frac{\partial H}{\partial \boldsymbol{u}} = \boldsymbol{R}\boldsymbol{u} + \boldsymbol{B}^{\mathsf{T}}\boldsymbol{\lambda} = \boldsymbol{0},$$

so

$$\boldsymbol{u}^{*} = -\boldsymbol{R}^{-1}\boldsymbol{B}^{\mathsf{T}}\boldsymbol{\lambda}.$$

The costate equation is

$$\dot{\boldsymbol{\lambda}} = -\boldsymbol{Q}\boldsymbol{x} - \boldsymbol{A}^{\mathsf{T}}\boldsymbol{\lambda}.$$

The terminal transversality condition is

$$\boldsymbol{\lambda}(t_f) = \boldsymbol{S}_f \boldsymbol{x}(t_f).$$

The Riccati Transformation

Introduce

$$\boldsymbol{\lambda}(t) = \boldsymbol{S}(t)\boldsymbol{x}(t) + \boldsymbol{\beta}(t).$$

After substitution into the state and costate equations, one obtains

$$\dot{\boldsymbol{x}} = \left(\boldsymbol{A} - \boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^{\mathsf{T}}\boldsymbol{S}\right)\boldsymbol{x} - \boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^{\mathsf{T}}\boldsymbol{\beta},$$
$$\dot{\boldsymbol{\beta}} = \left(\boldsymbol{S}\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^{\mathsf{T}} - \boldsymbol{A}^{\mathsf{T}}\right)\boldsymbol{\beta} + \boldsymbol{\Gamma}(\boldsymbol{S})\boldsymbol{x},$$

where

$$\boldsymbol{\Gamma}(\boldsymbol{S}) = -\dot{\boldsymbol{S}} - \boldsymbol{S}\boldsymbol{A} - \boldsymbol{A}^{\mathsf{T}}\boldsymbol{S} + \boldsymbol{S}\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^{\mathsf{T}}\boldsymbol{S} - \boldsymbol{Q}.$$

Choose $\boldsymbol{S}$ such that

$$\boldsymbol{\Gamma}(\boldsymbol{S}) = \boldsymbol{0}.$$

Then

$$\dot{\boldsymbol{S}} = -\boldsymbol{S}\boldsymbol{A} - \boldsymbol{A}^{\mathsf{T}}\boldsymbol{S} + \boldsymbol{S}\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^{\mathsf{T}}\boldsymbol{S} - \boldsymbol{Q}.$$

Equation () is the **Riccati differential equation**.

Why the auxiliary term vanishes

At $t = t_f$,

$$\boldsymbol{\lambda}(t_f) = \boldsymbol{S}(t_f)\boldsymbol{x}(t_f) + \boldsymbol{\beta}(t_f).$$

But the transversality condition requires

$$\boldsymbol{\lambda}(t_f) = \boldsymbol{S}_f \boldsymbol{x}(t_f).$$

Choosing

$$\boldsymbol{S}(t_f) = \boldsymbol{S}_f$$

gives

$$\boldsymbol{\beta}(t_f) = \mathbf{0}.$$

Since

$$\dot{\boldsymbol{\beta}} = \left(\boldsymbol{S}\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^{\mathsf{T}} - \boldsymbol{A}^{\mathsf{T}}\right)\boldsymbol{\beta}$$

is a homogeneous linear differential equation, its solution can be written as

$$\boldsymbol{\beta}(t) = \boldsymbol{\Phi}_{\beta}(t, t_f)\boldsymbol{\beta}(t_f).$$

Therefore,

$$\boxed{\boldsymbol{\beta}(t) \equiv \mathbf{0}.}$$

Hence

$$\boldsymbol{\lambda}(t) = \boldsymbol{S}(t)\boldsymbol{x}(t).$$

Finite-horizon optimal feedback law

The optimal control becomes

$$\boxed{\boldsymbol{u}^*(t) = -\boldsymbol{R}^{-1}\boldsymbol{B}^{\mathsf{T}}\boldsymbol{S}(t)\boldsymbol{x}^*(t).}$$

The corresponding closed-loop state equation is

$$\dot{\boldsymbol{x}}^* = \left[\boldsymbol{A} - \boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^{\mathsf{T}}\boldsymbol{S}(t)\right]\boldsymbol{x}^*.$$

The finite-horizon solution procedure is therefore:

1. Integrate the Riccati differential equation backward from

$$\boldsymbol{S}(t_f) = \boldsymbol{S}_f.$$

2. Use the resulting $\boldsymbol{S}(t)$ to construct the time-varying feedback gain

$$\boldsymbol{K}(t) = \boldsymbol{R}^{-1}\boldsymbol{B}^{\mathsf{T}}\boldsymbol{S}(t).$$

3. Integrate the state equation forward from $\boldsymbol{x}(0) = \boldsymbol{x}_0$.
4. Recover

$$\boldsymbol{u}^*(t) = -\boldsymbol{K}(t)\boldsymbol{x}^*(t).$$

Key point
The finite-horizon optimal feedback law is time varying because $\boldsymbol{S}(t)$ is obtained from a differential equation with a terminal condition.

Infinite-Horizon Linear Quadratic Regulator

The infinite-horizon problem is

$$\min_{\boldsymbol{u}(\cdot)} J = \frac{1}{2}\int_0^\infty \left[\boldsymbol{x}^\mathsf{T}\boldsymbol{Q}\boldsymbol{x} + \boldsymbol{u}^\mathsf{T}\boldsymbol{R}\boldsymbol{u}\right]\,\mathrm{d}t,$$

subject to

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u}, \qquad \boldsymbol{x}(0) = \boldsymbol{x}_0.$$

Here $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{Q}$, and $\boldsymbol{R}$ are constant matrices.
This problem is the classical **linear quadratic regulator**, or LQR.

Finite horizon versus infinite horizon

Feature	Finite-horizon LQ problem	Infinite-horizon LQR
Time interval	$[0, t_f]$	$[0, \infty)$
Terminal cost	Usually present	Usually absent
Riccati equation	Differential	Algebraic
Feedback gain	Time varying	Constant
Computation	Backward integration	Matrix equation solve

Convergence Requirements on an Infinite Horizon

For the cost

$$J = \frac{1}{2}\int_0^\infty \left[\boldsymbol{x}^\mathsf{T}\boldsymbol{Q}\boldsymbol{x} + \boldsymbol{u}^\mathsf{T}\boldsymbol{R}\boldsymbol{u}\right]\,\mathrm{d}t$$

to remain finite, the integrand must approach zero:

$$\lim_{t\to\infty}\left[\boldsymbol{x}^\mathsf{T}\boldsymbol{Q}\boldsymbol{x} + \boldsymbol{u}^\mathsf{T}\boldsymbol{R}\boldsymbol{u}\right] = 0.$$

Because

$$\boldsymbol{Q} \succeq 0, \qquad \boldsymbol{R} \succ 0,$$

the two nonnegative terms cannot cancel one another. Therefore,

$$\lim_{t\to\infty} \boldsymbol{x}^{\mathsf{T}}\boldsymbol{Q}\boldsymbol{x} = 0,$$
$$\lim_{t\to\infty} \boldsymbol{u}^{\mathsf{T}}\boldsymbol{R}\boldsymbol{u} = 0.$$

Under the standard controllability and observability assumptions, this implies

$$\boldsymbol{x}^*(t) \to \boldsymbol{0}, \qquad \boldsymbol{u}^*(t) \to \boldsymbol{0} \qquad \text{as } t \to \infty.$$

Since

$$\boldsymbol{\lambda} = \boldsymbol{S}\boldsymbol{x},$$

the optimal costate also satisfies

$$\boldsymbol{\lambda}^*(t) \to \boldsymbol{0}.$$

The Algebraic Riccati Equation

For the infinite-horizon, time-invariant problem, the steady-state Riccati matrix is constant. Hence

$$\dot{\boldsymbol{S}} = \boldsymbol{0}.$$

Substituting this into the Riccati differential equation gives

$$-\boldsymbol{S}\boldsymbol{A} - \boldsymbol{A}^{\mathsf{T}}\boldsymbol{S} + \boldsymbol{S}\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^{\mathsf{T}}\boldsymbol{S} - \boldsymbol{Q} = \boldsymbol{0}.$$

Equivalently,

$$\boxed{\boldsymbol{A}^{\mathsf{T}}\boldsymbol{S} + \boldsymbol{S}\boldsymbol{A} - \boldsymbol{S}\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^{\mathsf{T}}\boldsymbol{S} + \boldsymbol{Q} = \boldsymbol{0}.}$$

Equation () is the **algebraic Riccati equation**.
The infinite-horizon feedback law is

$$\boxed{\boldsymbol{u}^* = -\boldsymbol{K}\boldsymbol{x}^*, \qquad \boldsymbol{K} = \boldsymbol{R}^{-1}\boldsymbol{B}^{\mathsf{T}}\boldsymbol{S}.}$$

The closed-loop dynamics are

$$\dot{\boldsymbol{x}}^* = (\boldsymbol{A} - \boldsymbol{B}\boldsymbol{K})\boldsymbol{x}^*.$$

The Stabilizing Riccati Solution

The algebraic Riccati equation is nonlinear and quadratic in $\boldsymbol{S}$. It can possess multiple symmetric solutions. The relevant solution is the one that is:

1. symmetric;
2. positive semidefinite or positive definite under stronger assumptions; and
3. stabilizing.

The stabilizing solution satisfies

$$\operatorname{Re}\left[\lambda_i(\boldsymbol{A}-\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^{\mathsf{T}}\boldsymbol{S})\right]<0$$

for every closed-loop eigenvalue.

Thus,

$$\boldsymbol{x}^*(t)\to\boldsymbol{0}\qquad \text{as } t\to\infty.$$

Key point
Not every algebraic solution of the Riccati equation is useful. The physically relevant LQR solution is the stabilizing symmetric solution.

Computational Remarks

For moderate state dimension, the algebraic Riccati equation is solved efficiently using standard numerical linear algebra routines.

For large-scale systems, however, direct solution becomes expensive because $\boldsymbol{S}\in\mathbb{R}^{n\times n}$ may be dense even when $\boldsymbol{A}$ and $\boldsymbol{B}$ are sparse.

This issue appears frequently after spatial discretization of distributed-parameter systems such as:

- heat equations;
- structural vibration models;
- fluid systems;
- flexible structures; and
- other partial differential equation models.

A spatial discretization can create hundreds or thousands of states, making the Riccati equation itself a large-scale computational problem.

Return to the State–Costate Boundary-Value Problem

Before the Riccati transformation, the finite-horizon optimality system is

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix}\boldsymbol{x}\\ \boldsymbol{\lambda}\end{bmatrix}=\begin{bmatrix}\boldsymbol{A} & -\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^{\mathsf{T}}\\ -\boldsymbol{Q} & -\boldsymbol{A}^{\mathsf{T}}\end{bmatrix}\begin{bmatrix}\boldsymbol{x}\\ \boldsymbol{\lambda}\end{bmatrix}.$$

The boundary conditions are split between the two endpoints:

$$\boldsymbol{x}(0) = \boldsymbol{x}_0, \qquad \boldsymbol{\lambda}(t_f) = \boldsymbol{S}_f \boldsymbol{x}(t_f).$$

This is a two-point boundary-value problem. The initial state is known, but the initial costate is unknown. At the terminal time, the state is unknown and only a relation between state and costate is available.
If

$$\boldsymbol{\Phi}(t, \tau)$$

denotes the state-transition matrix of (), then

$$\begin{bmatrix} \boldsymbol{x}(t) \\ \boldsymbol{\lambda}(t) \end{bmatrix} = \boldsymbol{\Phi}(t, \tau) \begin{bmatrix} \boldsymbol{x}(\tau) \\ \boldsymbol{\lambda}(\tau) \end{bmatrix}.$$

The transition matrix alone does not remove the missing-boundary-condition problem.

The Shooting Method

The shooting method converts a boundary-value problem into an iterative sequence of initial-value problems. Suppose the initial costate is unknown. Define a guess

$$\boldsymbol{\lambda}(0) = \boldsymbol{\eta}.$$

For any chosen $\boldsymbol{\eta}$, integrate the state–costate equations forward to t_f. Define the terminal residual

$$\boldsymbol{r}(\boldsymbol{\eta}) = \boldsymbol{\lambda}(t_f; \boldsymbol{\eta}) - \boldsymbol{S}_f \boldsymbol{x}(t_f; \boldsymbol{\eta}).$$

The correct initial costate satisfies

$$\boxed{\boldsymbol{r}(\boldsymbol{\eta}^*) = \boldsymbol{0}.}$$

The shooting method therefore solves the nonlinear root-finding problem

$$\boldsymbol{\eta}^* = \arg \left\{ \boldsymbol{r}(\boldsymbol{\eta}) = \boldsymbol{0} \right\}.$$

A Newton update takes the form

$$\boldsymbol{\eta}_{k+1} = \boldsymbol{\eta}_k - \left[\frac{\partial \boldsymbol{r}}{\partial \boldsymbol{\eta}} \right]^{-1} \boldsymbol{r}(\boldsymbol{\eta}_k).$$

Backward shooting

One may instead guess the unknown terminal state:

$$\boldsymbol{x}(t_f) = \boldsymbol{\zeta},$$

construct

$$\boldsymbol{\lambda}(t_f) = \boldsymbol{S}_f \boldsymbol{\zeta},$$

and integrate backward. The residual is then

$$\boldsymbol{r}(\boldsymbol{\zeta}) = \boldsymbol{x}(0; \boldsymbol{\zeta}) - \boldsymbol{x}_0.$$

Both approaches seek a boundary guess that causes the integrated trajectory to hit the boundary conditions at the opposite endpoint.

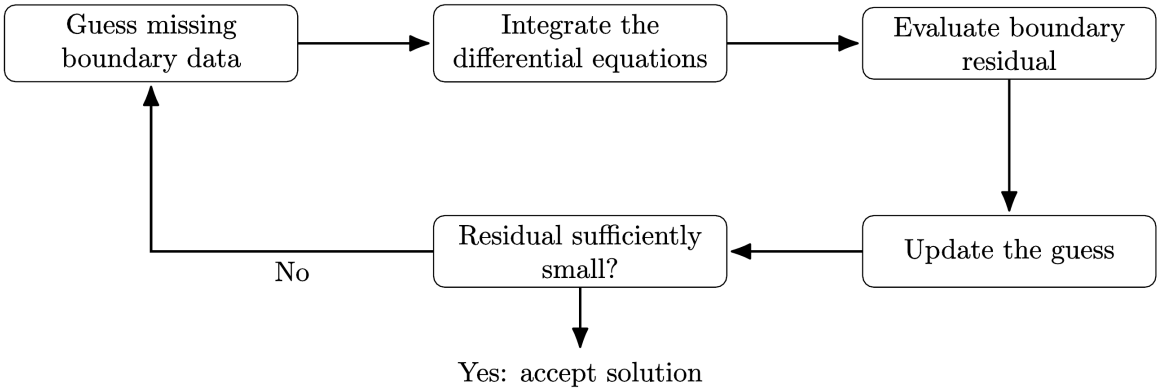

Figure 14: Shooting iterates between a boundary-data guess, trajectory integration, residual evaluation, and guess update.

Why Shooting Can Be Numerically Unstable

Consider the scalar problem

$$J = \frac{1}{2} s_f x^2(t_f) + \frac{1}{2} \int_0^{t_f} \left(q x^2 + r u^2 \right) \, \mathrm{d}t,$$

subject to

$$\dot{x} = a x + b u, \qquad x(0) = x_0,$$

where

$$q > 0, \qquad r > 0, \qquad b \neq 0.$$

The state–costate system is

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix} x \\ \lambda \end{bmatrix} = \underbrace{\begin{bmatrix} a & -b^2/r \\ -q & -a \end{bmatrix}}_{\boldsymbol{F}} \begin{bmatrix} x \\ \lambda \end{bmatrix}.$$

The characteristic equation is

$$\det(\mu \boldsymbol{I} - \boldsymbol{F}) = 0.$$

Thus,

$$\det \begin{bmatrix} \mu - a & b^2/r \\ q & \mu + a \end{bmatrix} = 0,$$

which gives

$$\mu^2 - a^2 - \frac{b^2 q}{r} = 0.$$

Therefore,

$$\boxed{\mu_{1,2} = \pm\sqrt{a^2 + \frac{b^2 q}{r}}.}$$

The eigenvalues are real, nonzero, and symmetric about the origin.
Let

$$\mu = \sqrt{a^2 + \frac{b^2 q}{r}} > 0.$$

Then the general solution contains one growing and one decaying mode:

$$\boldsymbol{p}(t) = c_1 e^{\mu t}\boldsymbol{v}_1 + c_2 e^{-\mu t}\boldsymbol{v}_2.$$

Forward integration

During forward integration, the mode

$$e^{\mu t}$$

grows exponentially. Any small component of the initial-guess error along this mode is amplified.

Backward integration

During backward integration, the mode that decays in forward time becomes unstable. Thus, the term associated with

$$e^{-\mu t}$$

grows when time is traversed backward.

Therefore, the state–costate system is unstable in either integration direction.

Key point
A shooting method may look simple analytically, but small errors in guessed boundary values or numerical integration can grow exponentially. Long horizons and strongly unstable modes make the residual map extremely sensitive.

Consequences for Numerical Optimal Control

Shooting methods may fail or become poorly conditioned when:

- the time horizon is long;
- the system contains strongly unstable modes;
- the initial guess is not close to the solution;
- the costate dynamics are highly sensitive;
- the dynamics are nonlinear; or
- the integrator introduces accumulated numerical error.

These limitations motivate more robust methods such as:

- multiple shooting;
- direct transcription;
- collocation;
- pseudospectral methods; and
- sparse nonlinear programming.

Summary

The major conclusions are:

1. In the finite-horizon LQ problem, the costate can be written as

$$\boldsymbol{\lambda}(t) = \boldsymbol{S}(t)\boldsymbol{x}(t).$$

2. The matrix $\boldsymbol{S}(t)$ satisfies a Riccati differential equation integrated backward from

$$\boldsymbol{S}(t_f) = \boldsymbol{S}_f.$$

3. The finite-horizon feedback gain is time varying.

4. The infinite-horizon LQR produces the algebraic Riccati equation.
5. The relevant algebraic Riccati solution is the stabilizing symmetric solution.
6. The resulting LQR feedback gain is constant.
7. A two-point boundary-value problem can be solved iteratively using shooting.
8. Shooting may become numerically unstable because the Hamiltonian state–costate system contains both growing and decaying modes.

Connection. LQR is unusually tractable; for general nonlinear problems the same necessary conditions form a boundary-value problem that must be solved numerically.

Indirect Numerical Methods, Multiple Shooting, and Pontryagin's Minimum Principle

Most optimal control problems cannot be solved analytically. Nonlinear dynamics, complicated endpoint conditions, control bounds, and long time horizons generally require numerical solution.

This section introduces two broad classes of numerical optimal control methods:

1. **Indirect methods**, which first derive necessary conditions and then solve the resulting boundary-value problem.
2. **Direct methods**, which discretize and optimize the original control problem directly.

The chapter focuses on indirect methods, especially:

- Newton's method for root finding;
- single shooting;
- multiple shooting;
- continuity or checkpoint constraints;
- parallel shooting; and
- Pontryagin's Minimum Principle for constrained controls.

Why Numerical Methods Are Necessary

Consider the general optimal control problem

$$\min_{\boldsymbol{u}(\cdot)} J = \Phi\big(\boldsymbol{x}(t_0), t_0, \boldsymbol{x}(t_f), t_f\big) + \int_{t_0}^{t_f} L(\boldsymbol{x}, \boldsymbol{u}, t)\,\mathrm{d}t,$$

subject to

$$\begin{aligned} \dot{\boldsymbol{x}} &= \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, t), \\ \boldsymbol{\phi}\big(\boldsymbol{x}(t_0), t_0, \boldsymbol{x}(t_f), t_f\big) &= \boldsymbol{0}. \end{aligned}$$

Closed-form solutions are exceptional because:

- the state equations are often nonlinear;
- the costate equations are coupled to the state;
- the endpoint conditions are split across two times;
- the optimal control can switch between interior and boundary values; and
- the resulting boundary-value problem may be unstable or highly sensitive.

Thus, computational methods are central to practical optimal control.

Indirect and Direct Optimization Methods

A finite-dimensional analogy

Suppose one wants to minimize a smooth scalar function

$$f(x).$$

A necessary condition for an unconstrained local optimum is

$$\frac{\mathrm{d}f}{\mathrm{d}x} = 0.$$

Solving this condition is not itself an optimization problem. It is a root-finding problem:

$$g(x) = \frac{\mathrm{d}f}{\mathrm{d}x} = 0.$$

The second derivative then classifies the critical point:

$$\begin{aligned} f''(x^*) > 0 &\quad \Rightarrow \text{local minimum,} \\ f''(x^*) < 0 &\quad \Rightarrow \text{local maximum,} \\ f''(x^*) = 0 &\quad \Rightarrow \text{inconclusive without higher-order analysis.} \end{aligned}$$

This is an **indirect method**: one derives necessary conditions and solves them.

By contrast, a **direct method** operates on $f(x)$ itself and searches for a value that reduces the objective without first solving the stationarity equation exactly.

Key point
An indirect method solves the necessary conditions of optimality. A direct method solves a finite-dimensional approximation of the original optimization problem.

The Hamiltonian Boundary-Value Problem

For an unconstrained-control optimal control problem, the necessary conditions include

$$\begin{aligned} \dot{\boldsymbol{x}} &= \frac{\partial H}{\partial \boldsymbol{\lambda}}, \\ \dot{\boldsymbol{\lambda}} &= -\frac{\partial H}{\partial \boldsymbol{x}}, \\ \mathbf{0} &= \frac{\partial H}{\partial \boldsymbol{u}}, \end{aligned}$$

together with the endpoint constraints and transversality conditions.

Define the combined state–costate vector

$$\boldsymbol{p}(t) = \begin{bmatrix} \boldsymbol{x}(t) \\ \boldsymbol{\lambda}(t) \end{bmatrix}.$$

After eliminating the control through stationarity when possible, the canonical equations can be written as

$$\dot{\boldsymbol{p}} = \boldsymbol{g}(\boldsymbol{p}, t).$$

All endpoint and transversality conditions can be collected into a residual vector

$$\boldsymbol{s}\left(\boldsymbol{p}(t_0), t_0, \boldsymbol{p}(t_f), t_f\right) = \boldsymbol{0}.$$

Equations ()–() form a two-point boundary-value problem.

The numerical task is to find unknown endpoint data such that integration of the differential equations produces zero boundary residual.

Newton's Method for Root Finding

Suppose

$$\boldsymbol{G}(\boldsymbol{z}) = \boldsymbol{0}$$

is a system of nonlinear algebraic equations.

Let $\boldsymbol{z}_k$ be the current iterate and write

$$\boldsymbol{z}_{k+1} = \boldsymbol{z}_k + \Delta \boldsymbol{z}_k.$$

A first-order Taylor expansion gives

$$\boldsymbol{G}(\boldsymbol{z}_k + \Delta \boldsymbol{z}_k) \approx \boldsymbol{G}(\boldsymbol{z}_k) + \frac{\partial \boldsymbol{G}}{\partial \boldsymbol{z}}(\boldsymbol{z}_k)\Delta \boldsymbol{z}_k.$$

Requiring the linearized residual to vanish yields

$$\frac{\partial \boldsymbol{G}}{\partial \boldsymbol{z}}(\boldsymbol{z}_k)\Delta \boldsymbol{z}_k = -\boldsymbol{G}(\boldsymbol{z}_k).$$

Therefore,

$$\boxed{\boldsymbol{z}_{k+1} = \boldsymbol{z}_k - \left[\frac{\partial \boldsymbol{G}}{\partial \boldsymbol{z}}(\boldsymbol{z}_k)\right]^{-1} \boldsymbol{G}(\boldsymbol{z}_k).}$$

In computation, the Jacobian is not explicitly inverted. Instead, one solves the linear system

$$\boldsymbol{J}_k \Delta \boldsymbol{z}_k = -\boldsymbol{G}(\boldsymbol{z}_k).$$

Convergence remarks

Newton's method converges rapidly when:

- the initial guess is sufficiently close to the root;
- the residual is smooth;
- the Jacobian is nonsingular; and
- the Jacobian is computed accurately.

It may fail when the initial guess is poor or the Jacobian is ill-conditioned.

Single Shooting as a Root-Finding Problem

Suppose $\boldsymbol{\eta}$ denotes the unknown initial data, often the initial costate:

$$\boldsymbol{\lambda}(t_0) = \boldsymbol{\eta}.$$

For each guess $\boldsymbol{\eta}$:

1. form the complete initial condition;
2. integrate the canonical equations to t_f;
3. evaluate the terminal and transversality conditions; and
4. construct the shooting residual.

The residual may be written as

$$\boldsymbol{R}(\boldsymbol{\eta}) = \boldsymbol{s}\left(\boldsymbol{p}(t_0;\boldsymbol{\eta}), t_0, \boldsymbol{p}(t_f;\boldsymbol{\eta}), t_f\right).$$

The shooting problem is

$$\boxed{\boldsymbol{R}(\boldsymbol{\eta}^*) = \boldsymbol{0}.}$$

Newton's method gives

$$\boldsymbol{\eta}_{k+1} = \boldsymbol{\eta}_k - \left[\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{\eta}}\right]^{-1} \boldsymbol{R}(\boldsymbol{\eta}_k).$$

Algorithm

1. Choose an initial guess $\boldsymbol{\eta}_0$.
2. Integrate the state–costate equations.
3. Evaluate the boundary residual $\boldsymbol{R}(\boldsymbol{\eta}_k)$.
4. Compute or approximate the shooting Jacobian.
5. Solve for the Newton correction.
6. Update the shooting variables.
7. Repeat until

$$\|\boldsymbol{R}(\boldsymbol{\eta}_k)\| \leq \varepsilon.$$

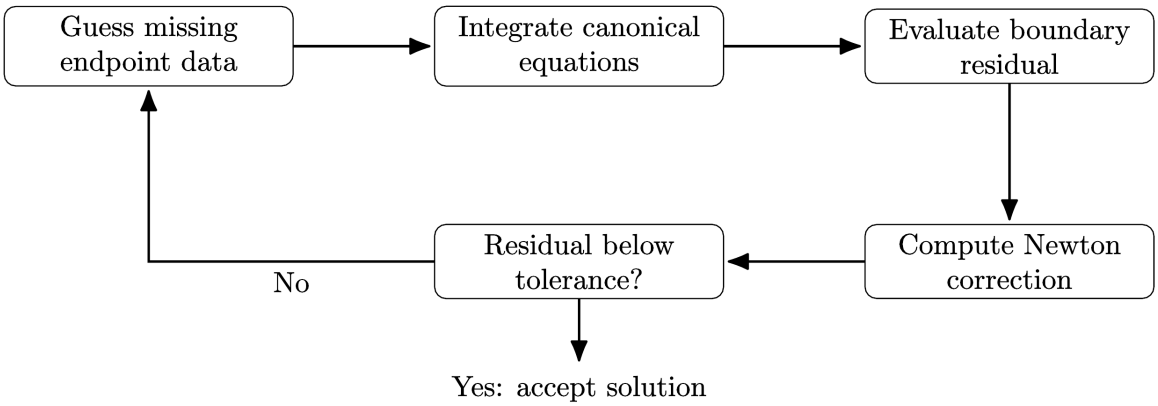

Figure 15: Newton shooting updates missing endpoint data until the boundary residual is below tolerance.

Sensitivity and the Shooting Jacobian

The shooting Jacobian measures the sensitivity of the boundary residual to the guessed endpoint variables:

$$J_{\text{shoot}} = \frac{\partial R}{\partial \eta}.$$

It can be computed using:

- finite differences;
- variational or sensitivity equations;
- automatic differentiation; or
- complex-step differentiation when applicable.

For the parameterized initial-value problem

$$\dot{p} = g(p, t), \qquad p(t_0) = p_0(\eta),$$

define the sensitivity matrix

$$Y(t) = \frac{\partial p(t)}{\partial \eta}.$$

Differentiating the dynamics gives

$$\dot{Y} = \frac{\partial g}{\partial p} Y,$$

with

$$Y(t_0) = \frac{\partial p_0}{\partial \eta}.$$

This matrix can then be used to construct the shooting Jacobian accurately.

Why Single Shooting Becomes Difficult

Single shooting integrates across the entire horizon. If the canonical dynamics contain growing modes, errors in the guessed endpoint data may grow exponentially.
The method becomes difficult when:

- the time horizon is long;
- the state–costate system is unstable;
- the initial guess is poor;
- the system contains multiple time scales;
- the dynamics are strongly nonlinear; or
- the boundary residual is poorly conditioned.

These limitations motivate multiple shooting.

Multiple Shooting

Partition the time interval:

$$t_0 < t_1 < \cdots < t_N = t_f.$$

Introduce an independent parameter at the beginning of each segment:

$$\boldsymbol{p}_i \approx \boldsymbol{p}(t_i), \qquad i = 0, \ldots, N-1.$$

Starting from $\boldsymbol{p}_i$, integrate

$$\dot{\boldsymbol{p}} = \boldsymbol{g}(\boldsymbol{p}, t)$$

over $[t_i, t_{i+1}]$.
Let

$$\widetilde{\boldsymbol{p}}_{i+1} = \varphi_i(\boldsymbol{p}_i)$$

denote the result of this integration at t_{i+1}.
The independently introduced parameter at the next node is $\boldsymbol{p}_{i+1}$. Continuity requires

$$\boxed{\widetilde{\boldsymbol{p}}_{i+1} - \boldsymbol{p}_{i+1} = \boldsymbol{0}, \qquad i = 0, \ldots, N-2.}$$

These are called:

- matching constraints;
- continuity constraints;
- checkpoint constraints; or
- defect constraints in a shooting context.

Why the next segment does not simply start from the previous result

If each segment simply starts from the endpoint produced by the previous segment, the procedure is identical to single shooting.

In multiple shooting, the initial value of each segment is an independent optimization or root-finding variable. The integrated endpoint and the next segment's initial variable are allowed to differ during the iteration. Their equality is imposed only through the matching constraints.

This prevents unstable error growth from accumulating over the full horizon in a single integration.

Multiple-Shooting Residual

Collect all segment initial values into

$$\boldsymbol{Z} = \begin{bmatrix} \boldsymbol{p}_0 \\ \boldsymbol{p}_1 \\ \vdots \\ \boldsymbol{p}_{N-1} \end{bmatrix}.$$

The residual consists of:

1. the original endpoint conditions; and
2. all continuity conditions.

A general multiple-shooting residual is

$$\mathcal{R}(\boldsymbol{Z}) = \begin{bmatrix} s\left(\boldsymbol{p}_0, t_0, \widetilde{\boldsymbol{p}}_N, t_f\right) \\ \widetilde{\boldsymbol{p}}_1 - \boldsymbol{p}_1 \\ \widetilde{\boldsymbol{p}}_2 - \boldsymbol{p}_2 \\ \vdots \\ \widetilde{\boldsymbol{p}}_{N-1} - \boldsymbol{p}_{N-1} \end{bmatrix}.$$

The numerical problem is

$$\boxed{\mathcal{R}(\boldsymbol{Z}) = \boldsymbol{0}.}$$

Newton's method or a related nonlinear root solver can then be applied.

Parallel Shooting Interpretation

Each segment integration depends only on its own initial parameter $\boldsymbol{p}_i$. Therefore, all segment integrations can be performed independently and, in principle, simultaneously.

This is why multiple shooting is also called **parallel shooting**.

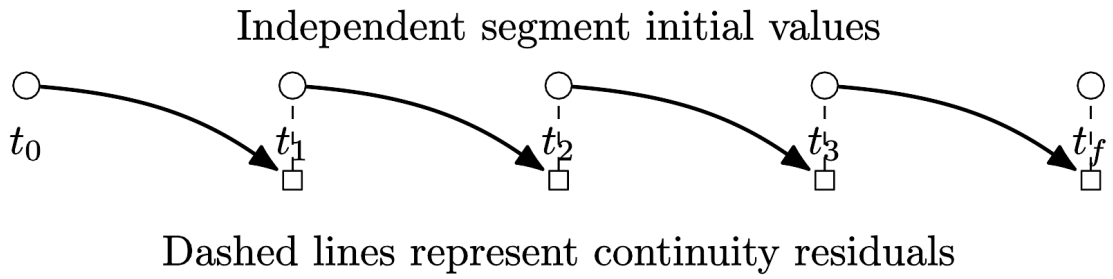

Figure 16: Multiple shooting uses independent segment initial values and enforces continuity through matching residuals.

Single Shooting Versus Multiple Shooting

Feature	Single shooting	Multiple shooting
Number of variables	Small	Larger
Integration horizon	Entire interval	Short segments
Sensitivity to instability	High	Reduced
Memory requirement	Low	Higher
Parallelization	Limited	Natural
Implementation complexity	Lower	Higher
Initial-guess robustness	Often poor	Usually improved

Multiple shooting trades additional variables and continuity constraints for improved numerical robustness.

Historical Computational Perspective

Indirect shooting was attractive in early computational optimal control because it introduced relatively few unknown variables.

If the state dimension is n, then the state–costate vector has dimension $2n$. A basic shooting method may only need to solve for a subset of these endpoint values.

Multiple shooting introduces approximately $2n$ additional variables per segment, increasing memory and algebraic problem size. Historically, this was a serious limitation.

Modern computers reduce this concern, but the tradeoff remains:

$$\text{more variables} \quad \Longleftrightarrow \quad \text{better control of integration sensitivity.}$$

Practical MATLAB Workflow

A simple indirect-shooting implementation can be organized as follows:

1. Write a function for the state–costate dynamics.
2. Write a function that accepts the unknown shooting variables.
3. Construct the complete initial or terminal data.

4. Integrate using an ODE solver.
5. Evaluate the endpoint residual.
6. Pass the residual function to a nonlinear equation solver such as `fsolve`.
7. Verify the endpoint conditions after convergence.

A generic residual function has the conceptual form

```
function residual = shootingResidual(eta)
    p0 = constructInitialData(eta);
    [t,p] = ode45(@canonicalDynamics,[t0 tf],p0);
    pf = p(end,:)';
    residual = endpointConditions(p0,pf);
end
```

The converged numerical result should always be checked independently.

Transition to Constrained Controls

The stationarity condition

$$\frac{\partial H}{\partial \boldsymbol{u}} = \mathbf{0}$$

assumes that the optimal control lies in the interior of the allowable control set.
In practical systems, controls are usually bounded:

$$\boldsymbol{u}(t) \in \mathcal{U}.$$

Examples include:

- actuator-force limits;
- torque limits;
- steering-angle limits;
- thrust limits;
- angle-of-attack constraints;
- voltage and current bounds; and
- rate limits.

When the optimal control reaches a bound, the interior stationarity condition is not generally valid.

Pontryagin's Minimum Principle

Suppose $\boldsymbol{x}^*(t)$ and $\boldsymbol{\lambda}^*(t)$ are the optimal state and costate.
At each time, hold these optimal functions fixed and compare the Hamiltonian evaluated at an arbitrary admissible control $\boldsymbol{u} \in \mathcal{U}$ with the Hamiltonian evaluated at the optimal control $\boldsymbol{u}^*$.
Optimality requires

$$H(\boldsymbol{x}^*(t), \boldsymbol{u}(t), \boldsymbol{\lambda}^*(t), t) \geq H(\boldsymbol{x}^*(t), \boldsymbol{u}^*(t), \boldsymbol{\lambda}^*(t), t)$$

for every admissible $\boldsymbol{u}(t) \in \mathcal{U}$ and almost every t.

Equivalently,

$$\boxed{\boldsymbol{u}^*(t) \in \arg\min_{\boldsymbol{u} \in \mathcal{U}} H(\boldsymbol{x}^*(t), \boldsymbol{u}, \boldsymbol{\lambda}^*(t), t)\,.}$$

Equation () is Pontryagin's Minimum Principle under the minimum-Hamiltonian sign convention used here.

Key point
The Hamiltonian minimization is performed separately at each time. The optimal control minimizes the Hamiltonian over the allowable control set while the optimal state and costate are held fixed.

Minimum Principle Versus Stationarity

If the control set is unconstrained and the Hamiltonian is differentiable, the minimum principle often reduces to

$$\frac{\partial H}{\partial \boldsymbol{u}} = \mathbf{0}.$$

If the control is bounded, the minimizer may occur at the boundary, so

$$\frac{\partial H}{\partial \boldsymbol{u}} \neq \mathbf{0}$$

can hold at the optimum.

Therefore:

$$\boxed{\text{Hamiltonian minimization is more general than stationarity.}}$$

Scalar Bounded-Control Illustration

Suppose

$$u_{\min} \leq u(t) \leq u_{\max}$$

and the Hamiltonian depends linearly on the control:

$$H(x, u, \lambda, t) = H_0(x, \lambda, t) + \sigma(x, \lambda, t)u.$$

The quantity

$$\sigma(t) = \frac{\partial H}{\partial u}$$

is the switching function.

Hamiltonian minimization gives

$$u^*(t) = \begin{cases} u_{\min}, & \sigma(t) > 0, \\ u_{\max}, & \sigma(t) < 0, \\ \text{undetermined by first-order minimization}, & \sigma(t) = 0. \end{cases}$$

This structure motivates bang-bang and singular-control analysis.

Complete Indirect Optimality System

For a bounded-control problem, the indirect necessary conditions are:

$$\begin{aligned} \dot{\boldsymbol{x}}^* &= \frac{\partial H}{\partial \boldsymbol{\lambda}} \left(\boldsymbol{x}^*, \boldsymbol{u}^*, \boldsymbol{\lambda}^*, t\right), \\ \dot{\boldsymbol{\lambda}}^* &= -\frac{\partial H}{\partial \boldsymbol{x}} \left(\boldsymbol{x}^*, \boldsymbol{u}^*, \boldsymbol{\lambda}^*, t\right), \\ \boldsymbol{u}^*(t) &\in \arg\min_{\boldsymbol{u} \in \mathcal{U}} H\left(\boldsymbol{x}^*, \boldsymbol{u}, \boldsymbol{\lambda}^*, t\right), \end{aligned}$$

together with:

- endpoint constraints;
- transversality conditions;
- path constraints, when present; and
- complementary or switching conditions, when needed.

The resulting system is generally a nonlinear two-point boundary-value problem solved by shooting, multiple shooting, or other numerical methods.

Common Errors

1. Treating the shooting residual as the original cost function.
2. Explicitly inverting a Newton Jacobian instead of solving a linear system.
3. Starting every multiple-shooting segment from the previous integrated endpoint.
4. Omitting the continuity constraints in multiple shooting.
5. Assuming $\partial H/\partial u = 0$ remains valid at a control bound.
6. Minimizing the Hamiltonian over all real controls instead of the admissible set.
7. Confusing a necessary condition with a sufficient condition.
8. Accepting solver convergence without checking the endpoint residuals.

Summary

The main conclusions are:

1. Most optimal control problems require numerical solution.
2. Indirect methods solve the necessary conditions of optimality.
3. The optimality system becomes a root-finding problem coupled to differential equations.
4. Single shooting guesses missing endpoint data and iterates until the boundary conditions are satisfied.
5. Newton's method provides a systematic shooting update.
6. Single shooting can be highly sensitive to unstable canonical dynamics.
7. Multiple shooting introduces independent segment initial values and continuity constraints.
8. Multiple shooting can be parallelized and is often more robust.
9. For constrained controls, Hamiltonian stationarity must be replaced by pointwise Hamiltonian minimization.
10. Pontryagin's Minimum Principle is the appropriate general condition for selecting the optimal control.

Connection. The Hamiltonian formulation also yields conservation properties and switching rules, providing a direct route to minimum-time bang–bang control.

Hamiltonian Conservation, Minimum-Time Control, and Bang–Bang Solutions

This section develops several important consequences of the Hamiltonian formulation and applies Pontryagin's Minimum Principle to a classic bounded-control problem.

The main topics are:

1. the evolution of the Hamiltonian along an optimal trajectory;
2. conditions under which the Hamiltonian is constant;
3. conditions under which the optimal Hamiltonian is identically zero;
4. the minimum-time double-integrator problem;
5. bang–bang control;
6. switching functions and switching times;
7. state-space switching curves; and
8. an introduction to singular arcs.

Hamiltonian Evolution Along an Optimal Trajectory

Consider the Hamiltonian

$$H(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\lambda}, t) = L(\boldsymbol{x}, \boldsymbol{u}, t) + \boldsymbol{\lambda}^{\mathsf{T}} \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, t).$$

Suppose the optimal control has been obtained in feedback form:

$$\boldsymbol{u}^* = \boldsymbol{u}^*(\boldsymbol{x}^*, \boldsymbol{\lambda}^*, t).$$

After substitution, the optimal Hamiltonian can be viewed as

$$H^* = H(\boldsymbol{x}^*, \boldsymbol{\lambda}^*, t).$$

The canonical equations are

$$\dot{\boldsymbol{x}}^* = \frac{\partial H}{\partial \boldsymbol{\lambda}},$$
$$\dot{\boldsymbol{\lambda}}^* = -\frac{\partial H}{\partial \boldsymbol{x}}.$$

For clarity, write the scalar components as

$$\dot{x}_i^* = \frac{\partial H}{\partial \lambda_i},$$
$$\dot{\lambda}_i^* = -\frac{\partial H}{\partial x_i}, \qquad i = 1, \ldots, n.$$

The total derivative of the Hamiltonian is

$$\frac{\mathrm{d}H^*}{\mathrm{d}t} = \sum_{i=1}^{n} \left(\frac{\partial H}{\partial x_i} \dot{x}_i^* + \frac{\partial H}{\partial \lambda_i} \dot{\lambda}_i^* \right) + \frac{\partial H}{\partial t}.$$

Substituting the canonical equations gives

$$\begin{aligned} \frac{\mathrm{d}H^*}{\mathrm{d}t} &= \sum_{i=1}^{n} \left(\frac{\partial H}{\partial x_i} \frac{\partial H}{\partial \lambda_i} - \frac{\partial H}{\partial \lambda_i} \frac{\partial H}{\partial x_i} \right) + \frac{\partial H}{\partial t} \\ &= \frac{\partial H}{\partial t}. \end{aligned}$$

Therefore,

$$\boxed{\frac{\mathrm{d}H^*}{\mathrm{d}t} = \frac{\partial H}{\partial t}.}$$

When Is the Hamiltonian Constant?

If the Hamiltonian has no explicit time dependence,

$$\frac{\partial H}{\partial t} = 0,$$

then

$$\frac{\mathrm{d}H^*}{\mathrm{d}t} = 0.$$

Hence,

$$\boxed{H^*(t) = \text{constant}.}$$

This result does not require that the state or costate be constant. Their contributions cancel through the canonical equations.

Key point
If the Hamiltonian is autonomous, meaning that time does not appear explicitly, then the optimal Hamiltonian is constant along the optimal trajectory.

When Is the Optimal Hamiltonian Zero?

The statement

$$H^*(t) \equiv 0$$

requires more than a free final time.

For a problem with endpoint cost

$$\Phi\big(\boldsymbol{x}(t_0), t_0, \boldsymbol{x}(t_f), t_f\big)$$

and endpoint constraints

$$\boldsymbol{\phi}\big(\boldsymbol{x}(t_0), t_0, \boldsymbol{x}(t_f), t_f\big) = \mathbf{0},$$

the final-time transversality condition is

$$H(t_f) = -\frac{\partial \Phi}{\partial t_f} + \boldsymbol{\nu}^\mathsf{T} \frac{\partial \boldsymbol{\phi}}{\partial t_f}.$$

If:

1. the Hamiltonian has no explicit time dependence;
2. the final time is free;
3. the terminal cost has no explicit dependence on t_f; and
4. the terminal constraints have no explicit dependence on t_f,

then

$$H^*(t_f) = 0.$$

Since H^* is constant,

$$\boxed{H^*(t) \equiv 0.}$$

Key point
A free final time alone does not imply that the Hamiltonian is zero. The endpoint cost and endpoint constraints must also have no explicit dependence on the final time.

Minimum-Time Double-Integrator Problem

Consider a point mass subject to a bounded force. After normalization, the dynamics are

$$\ddot{x} = u,$$

with

$$|u(t)| \le 1.$$

Define the state variables

$$x_1 = x, \qquad x_2 = \dot{x}.$$

Then

$$\dot{x}_1 = x_2,$$
$$\dot{x}_2 = u.$$

The boundary conditions are

$$x_1(0) = x_{10},$$
$$x_2(0) = x_{20},$$
$$x_1(t_f) = 0,$$
$$x_2(t_f) = 0.$$

The goal is to minimize transfer time:

$$\boxed{\min_{u(\cdot),\, t_f} J = t_f.}$$

Equivalently,

$$J = \int_0^{t_f} 1 \, \mathrm{d}t.$$

The admissible control set is

$$\mathcal{U} = [-1, 1].$$

Hamiltonian for the Minimum-Time Problem

Using the integral representation of time, the Hamiltonian is

$$H = 1 + \lambda_1 x_2 + \lambda_2 u.$$

Some sign conventions absorb the constant 1 differently, but the control-dependent term is always

$$\lambda_2 u.$$

Because the system is autonomous and the final time is free with no explicit terminal-time dependence,

$$H^*(t) \equiv 0.$$

Why Stationarity Does Not Determine the Control

The stationarity derivative is

$$\frac{\partial H}{\partial u} = \lambda_2.$$

This derivative does not contain u. Therefore, the interior stationarity equation

$$\frac{\partial H}{\partial u} = 0$$

does not provide an explicit formula for the optimal control.

This is expected because the Hamiltonian is linear in u, so its minimum over a closed interval generally occurs at one of the bounds.

Pontryagin's Minimum Principle

The optimal control satisfies

$$u^*(t) \in \arg\min_{u \in [-1,1]} H(x^*(t), u, \lambda^*(t)).$$

Since only the term $\lambda_2 u$ depends on u,

$$u^*(t) \in \arg\min_{u \in [-1,1]} \lambda_2(t) u.$$

Hence,

$$\boxed{u^*(t) = \begin{cases} -1, & \lambda_2(t) > 0, \\ +1, & \lambda_2(t) < 0. \end{cases}}$$

Equivalently,

$$\boxed{u^*(t) = -\operatorname{sgn}(\lambda_2(t))}$$

whenever $\lambda_2(t) \neq 0$.

The switching function is

$$\sigma(t) = \frac{\partial H}{\partial u} = \lambda_2(t).$$

Costate Dynamics

The costate equations are

$$\dot{\lambda}_1 = -\frac{\partial H}{\partial x_1} = 0,$$
$$\dot{\lambda}_2 = -\frac{\partial H}{\partial x_2} = -\lambda_1.$$

Thus,

$$\lambda_1(t) = \bar{\lambda}_1,$$

where $\bar{\lambda}_1$ is constant.
Integrating the second equation gives

$$\lambda_2(t) = -\bar{\lambda}_1 t + \bar{\lambda}_2.$$

Therefore, λ_2 is an affine function of time.

Number of Possible Switches

Because $\lambda_2(t)$ is a straight line, it can cross zero at most once unless it is identically zero.
Therefore, the optimal control can have at most one switching time.
The only possible control structures are:

1. $u^*(t) \equiv +1$;
2. $u^*(t) \equiv -1$;
3. $u^*(t)$ switches from +1 to -1; or
4. $u^*(t)$ switches from -1 to +1.

Key point
The minimum-time double integrator has a bang–bang optimal control with at most one switch.

Excluding an Identically Zero Switching Function

If

$$\lambda_2(t) \equiv 0$$

on an interval, then

$$\dot{\lambda}_2 = -\lambda_1 = 0,$$

so

$$\lambda_1(t) \equiv 0.$$

This would imply

$$\boldsymbol{\lambda}(t) \equiv \mathbf{0}.$$

Such a trivial multiplier violates the nontriviality condition of Pontryagin's Minimum Principle. Therefore, the switching function cannot vanish identically on an interval for this problem.
Hence, no singular arc is present in the minimum-time double integrator.

State Trajectories Along Constant-Control Arcs

Let

$$u = s, \qquad s \in \{-1, +1\}.$$

Then

$$\begin{aligned} \dot{x}_1 &= x_2, \\ \dot{x}_2 &= s. \end{aligned}$$

Integrating from an initial point (x_{10}, x_{20}) gives

$$\begin{aligned} x_2(t) &= x_{20} + st, \\ x_1(t) &= x_{10} + x_{20}t + \frac{1}{2}st^2. \end{aligned}$$

Eliminate time using

$$t = \frac{x_2 - x_{20}}{s}.$$

Since $s^2 = 1$,

$$\boxed{x_1 = x_{10} + \frac{s}{2}\left(x_2^2 - x_{20}^2\right).}$$

Thus, constant-control trajectories are parabolas in the (x_1, x_2) phase plane.

Switching Curves Through the Origin

For a trajectory that reaches the origin under a constant control s, set

$$x_1 = 0, \qquad x_2 = 0$$

in (). The initial point on such a terminal arc must satisfy

$$0 = x_1 - \frac{s}{2}x_2^2.$$

Hence,

$$\boxed{x_1 = \frac{s}{2}x_2^2.}$$

Equivalently, the two switching curves are

$$\begin{aligned} x_1 &= \frac{1}{2}x_2^2, \quad \text{for } u = +1, \\ x_1 &= -\frac{1}{2}x_2^2, \quad \text{for } u = -1. \end{aligned}$$

A common compact representation is

$$\boxed{x_1 = -\frac{1}{2} x_2 |x_2|.}$$

This curve separates regions of the state space associated with different initial control actions.

State-Space Interpretation

If the initial point lies directly on a terminal switching curve, the origin can be reached using a single constant control.

Otherwise:

1. apply one extreme control to move toward the appropriate switching curve;
2. switch control when the trajectory reaches the curve; and
3. apply the opposite extreme control until the state reaches the origin.

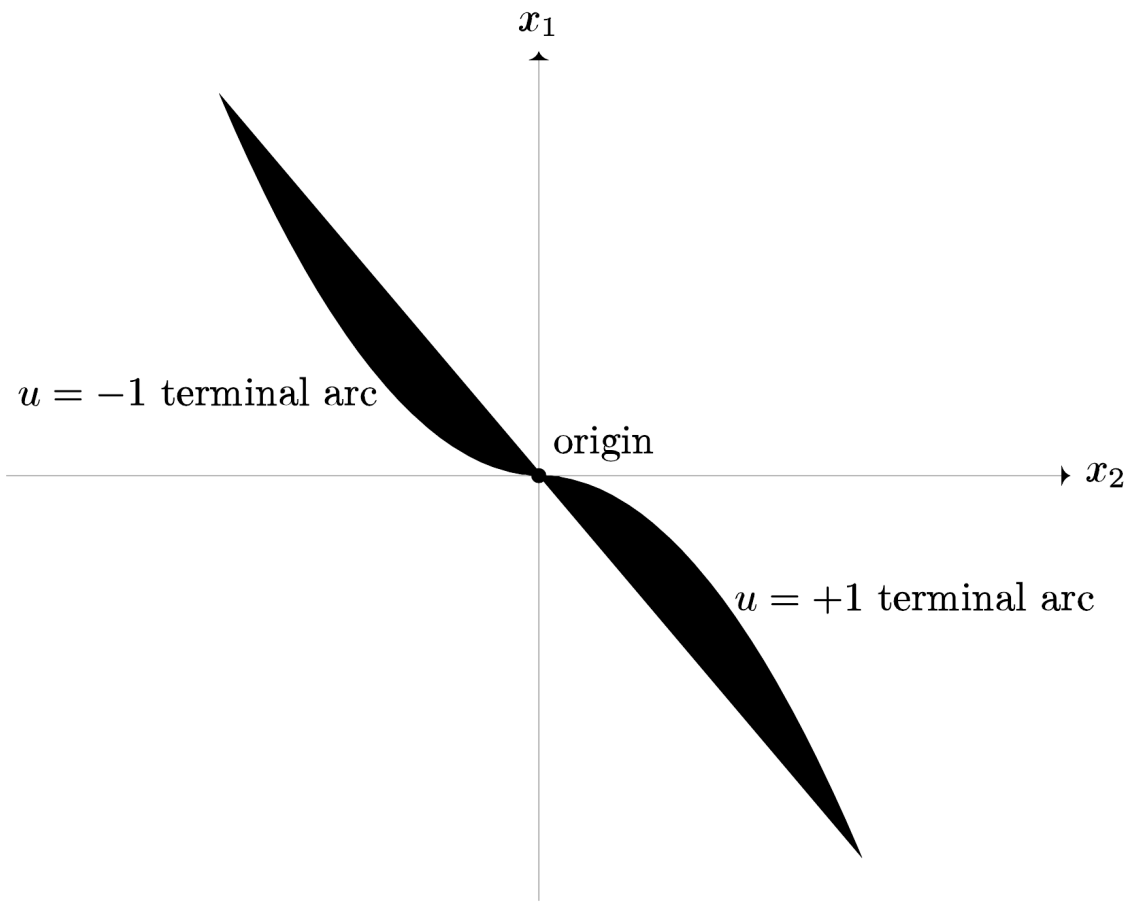

Figure 17: Switching curves for the minimum-time double integrator separate the terminal arcs associated with the two extreme controls.

Piecewise State Solution with One Switch

Suppose the control switches from -1 to +1 at $t = t_s$.
For $0 \le t \le t_s$,

$$x_2(t) = x_{20} - t,$$
$$x_1(t) = x_{10} + x_{20}t - \frac{1}{2}t^2.$$

Define

$$x_{2s} = x_2(t_s),$$
$$x_{1s} = x_1(t_s).$$

For $t_s \le t \le t_f$,

$$x_2(t) = x_{2s} + (t - t_s),$$
$$x_1(t) = x_{1s} + x_{2s}(t - t_s) + \frac{1}{2}(t - t_s)^2.$$

For the reverse switch, from +1 to -1, the signs reverse:

$$x_2(t) = x_{20} + t,$$
$$x_1(t) = x_{10} + x_{20}t + \frac{1}{2}t^2, \qquad 0 \le t \le t_s,$$

followed by

$$x_2(t) = x_{2s} - (t - t_s),$$
$$x_1(t) = x_{1s} + x_{2s}(t - t_s) - \frac{1}{2}(t - t_s)^2.$$

Computing the Switching Time

The switching time is determined by enforcing the terminal conditions:

$$x_1(t_f) = 0, \qquad x_2(t_f) = 0.$$

For a specified initial condition, one selects the switching sequence consistent with the phase-plane region and solves the resulting algebraic equations for t_s and t_f.
An equivalent geometric procedure is:

1. identify the constant-control parabola passing through the initial state;
2. determine its intersection with the appropriate terminal switching curve;
3. use that intersection as the switching point; and
4. compute the travel time along both arcs.

Why Minimum-Time Problems Favor Extreme Controls

If the control enters the dynamics linearly and the cost penalizes only elapsed time, reducing the magnitude of the control generally delays arrival.

Pontryagin's Minimum Principle formalizes this intuition. Since the Hamiltonian is linear in u, minimizing it over a bounded interval selects one of the endpoints.

Thus,

$$u^*(t) \in \{-1, +1\}$$

almost everywhere, except possibly at switching times or on singular arcs.

Higher-Order Integrator Systems

For the rth-order integrator

$$x^{(r)} = u,$$

the costate components become higher-degree polynomials in time.

Therefore, the switching function can possess more zeros, and the optimal control can switch more than once.

The double integrator is especially simple because its switching function is affine in time and therefore has at most one zero.

Singular Arcs

A singular arc occurs when the switching function vanishes over a nonzero interval:

$$\sigma(t) = \frac{\partial H}{\partial u} = 0$$

for all t in that interval.

On a singular arc, direct Hamiltonian minimization does not determine the control. One must differentiate the switching function until the control appears explicitly:

$$\sigma(t) = 0, \quad \dot{\sigma}(t) = 0, \quad \ddot{\sigma}(t) = 0, \quad \dots$$

The resulting conditions may produce a singular control law.

Singular arcs are common in:

- chemical-process control;
- bioreactor optimization;
- aerospace trajectory optimization;
- energy-management problems; and
- systems with one-sided actuation.

Key point
Numerically, a switching function that is merely small can be difficult to distinguish from one that is exactly zero over an interval. This makes singular-arc identification and enforcement challenging.

One-Sided and Two-Sided Control

Mechanical systems often admit two-sided control:

$$-u_{\max} \leq u \leq u_{\max}.$$

Examples include positive and negative torque, force, or acceleration.
Many process systems instead admit one-sided control:

$$0 \leq u \leq u_{\max}.$$

For example, material can be added to a reactor but may not be removable through the same control channel.
One-sided control can create asymmetric switching structures and often increases the likelihood of boundary and singular arcs.

Common Errors

1. Claiming that a free final time always implies $H^* = 0$.
2. Omitting the running-cost term 1 from a minimum-time Hamiltonian without explaining the adopted convention.
3. Applying $\partial H/\partial u = 0$ to a Hamiltonian that is linear in a bounded control.
4. Ignoring the admissible control interval in Hamiltonian minimization.
5. Reversing the sign of the bang–bang law.
6. Assuming that $\lambda_2 = 0$ at an isolated switch implies a singular arc.
7. Treating a switching curve as a state trajectory valid for every control.
8. Forgetting continuity of the state at the switching time.

Summary

The main conclusions are:

1. Along an optimal trajectory,

$$\frac{\mathrm{d}H^*}{\mathrm{d}t} = \frac{\partial H}{\partial t}.$$

2. If the Hamiltonian has no explicit time dependence, it is constant.
3. If the final time is free and the endpoint terms have no explicit final-time dependence, then the constant Hamiltonian is zero.

4. The minimum-time double integrator has a Hamiltonian linear in the bounded control.
5. Stationarity does not determine the optimal control in this problem.
6. Pontryagin's Minimum Principle yields

$$u^* = -\operatorname{sgn}(\lambda_2).$$

7. The switching function is affine in time and can cross zero at most once.
8. The optimal control is bang–bang with at most one switch.
9. Constant-control trajectories are parabolas in the phase plane.
10. Singular arcs require separate higher-order analysis but do not occur in the standard minimum-time double integrator.

Connection. Before treating more delicate switching behavior, it is useful to unify the common Bolza, Mayer, and Lagrange formulations and clarify minimum-time conventions.

Bolza, Mayer, and Lagrange Forms, Minimum-Time Conventions, and Singular Arcs

This section continues the analysis of the minimum-time double-integrator problem and develops several general modeling concepts in optimal control.

The main topics are:

1. equivalent minimum-time formulations;
2. Bolza, Mayer, and Lagrange cost forms;
3. conversion of a Lagrange problem into Mayer form;
4. augmented-state formulations;
5. correct interpretation of Pontryagin's Minimum Principle;
6. exclusion of constant and zero switching functions;
7. definition of singular arcs; and
8. why singular optimal control problems are difficult.

Two Equivalent Minimum-Time Formulations

Consider the minimum-time problem

$$\min t_f.$$

There are two common ways to represent the same cost.

Mayer representation

The cost can be written as a terminal cost:

$$J = t_f.$$

Equivalently,

$$J = \Phi(t_f), \qquad \Phi(t_f) = t_f.$$

If the running cost is zero, the Hamiltonian is

$$H_M = \boldsymbol{\lambda}^\top \boldsymbol{f}.$$

For a free final time, the transversality condition gives

$$H_M(t_f) = -\frac{\partial \Phi}{\partial t_f}.$$

Thus,

$$H_M(t_f) = -1.$$

If the Hamiltonian is autonomous,

$$\boxed{H_M(t) \equiv -1.}$$

Lagrange representation

The same minimum-time objective can be written as

$$J = \int_0^{t_f} 1 \, \mathrm{d}t.$$

Now the running cost is

$$L = 1,$$

and the Hamiltonian becomes

$$H_L = 1 + \boldsymbol{\lambda}^\mathsf{T} \boldsymbol{f}.$$

There is no terminal cost:

$$\Phi = 0.$$

For a free final time,

$$H_L(t_f) = 0.$$

If the Hamiltonian is autonomous,

$$\boxed{H_L(t) \equiv 0.}$$

Consistency of the two formulations

The two Hamiltonians differ by a constant:

$$H_L = 1 + H_M.$$

Since

$$H_M = -1,$$

we obtain

$$H_L = 0.$$

Thus, both formulations produce the same state, costate, control, and switching structure.

Key point
The numerical value of the Hamiltonian depends on the adopted cost representation and sign convention. The optimal control solution does not.

Bolza, Mayer, and Lagrange Forms

The most general standard cost functional is the Bolza form:

$$\boxed{J = \Phi\big(\boldsymbol{x}(t_0), t_0, \boldsymbol{x}(t_f), t_f\big) + \int_{t_0}^{t_f} L(\boldsymbol{x}, \boldsymbol{u}, t)\,\mathrm{d}t.}$$

It contains both an endpoint cost and a running cost.

Mayer form

The Mayer form contains only an endpoint cost:

$$\boxed{J = \Phi\big(\boldsymbol{x}(t_0), t_0, \boldsymbol{x}(t_f), t_f\big)\,.}$$

Lagrange form

The Lagrange form contains only a running cost:

$$\boxed{J = \int_{t_0}^{t_f} L(\boldsymbol{x}, \boldsymbol{u}, t)\,\mathrm{d}t.}$$

Relationship among the forms

Form	Endpoint term	Integral term
Mayer	Present	Absent
Lagrange	Absent	Present
Bolza	Present	Present

Converting Lagrange Form to Mayer Form

Consider a Lagrange-form problem:

$$J = \int_{t_0}^{t_f} L(\boldsymbol{x}, \boldsymbol{u}, t)\,\mathrm{d}t,$$

subject to

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, t).$$

Introduce an additional state:

$$x_{n+1}(t) = \int_{t_0}^{t} L(\boldsymbol{x}(\tau), \boldsymbol{u}(\tau), \tau)\,\mathrm{d}\tau.$$

Then

$$\dot{x}_{n+1} = L(\boldsymbol{x}, \boldsymbol{u}, t),$$

with

$$x_{n+1}(t_0) = 0.$$

At the final time,

$$x_{n+1}(t_f) = \int_{t_0}^{t_f} L(\boldsymbol{x}, \boldsymbol{u}, t)\,\mathrm{d}t = J.$$

Therefore, the original Lagrange problem is equivalent to the Mayer problem

$$\boxed{\min x_{n+1}(t_f).}$$

Augmented-State Dynamics

Define the augmented state

$$\boldsymbol{y} = \begin{bmatrix} \boldsymbol{x} \\ x_{n+1} \end{bmatrix}.$$

Then

$$\dot{\boldsymbol{y}} = \begin{bmatrix} \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, t) \\ L(\boldsymbol{x}, \boldsymbol{u}, t) \end{bmatrix} = \boldsymbol{g}(\boldsymbol{y}, \boldsymbol{u}, t).$$

The augmented Mayer problem is

$$\min x_{n+1}(t_f),$$

subject to

$$\dot{\boldsymbol{y}} = \boldsymbol{g}(\boldsymbol{y}, \boldsymbol{u}, t).$$

Key point
Every Lagrange-form optimal control problem can be converted into Mayer form by adding one state that accumulates the running cost.

Can Mayer Form Always Be Converted to Lagrange Form?

The reverse conversion is not always as direct.

A terminal cost

$$\Phi(\boldsymbol{x}(t_f), t_f)$$

may be nonlinear or may depend on endpoint variables in a way that is not naturally represented as a standard running cost.

In some cases, one can use identities such as

$$\Phi(\boldsymbol{x}(t_f)) - \Phi(\boldsymbol{x}(t_0)) = \int_{t_0}^{t_f} \frac{\mathrm{d}}{\mathrm{d}t}\Phi(\boldsymbol{x}(t))\,\mathrm{d}t,$$

but this generally introduces state derivatives and additional assumptions.

Therefore, the Lagrange-to-Mayer conversion is universal, whereas the reverse direction may be inconvenient or problem dependent.

Return to the Minimum-Time Double Integrator

The normalized dynamics are

$$\dot{x}_1 = x_2,$$
$$\dot{x}_2 = u,$$

with

$$-1 \le u \le 1.$$

Using the Mayer cost

$$J = t_f,$$

the Hamiltonian is

$$H = \lambda_1 x_2 + \lambda_2 u.$$

Because the problem is autonomous,

$$H^*(t) = \text{constant}.$$

The free-final-time transversality condition yields

$$\boxed{H^*(t) \equiv -1.}$$

Costate Equations

The costate equations are

$$\dot{\lambda}_1 = -\frac{\partial H}{\partial x_1} = 0,$$
$$\dot{\lambda}_2 = -\frac{\partial H}{\partial x_2} = -\lambda_1.$$

Hence,

$$\lambda_1(t) = C_1,$$

and

$$\lambda_2(t) = -C_1 t + C_2.$$

Thus, λ_2 is affine in time.

Careful Interpretation of the Minimum Principle

Pontryagin's Minimum Principle states that

$$H(\boldsymbol{x}^*(t), \boldsymbol{u}^*(t), \boldsymbol{\lambda}^*(t), t) \leq H(\boldsymbol{x}^*(t), \boldsymbol{u}(t), \boldsymbol{\lambda}^*(t), t)$$

for every admissible control $\boldsymbol{u}(t) \in \mathcal{U}$.
The state and costate are held fixed at their optimal values. Only the control is varied.
Therefore,

$$u^*(t) \in \underset{u \in [-1,1]}{\arg\min} \left[\lambda_1^*(t) x_2^*(t) + \lambda_2^*(t) u\right].$$

Since the first term is independent of u,

$$u^*(t) \in \underset{u \in [-1,1]}{\arg\min}\, \lambda_2^*(t) u.$$

Thus,

$$\boxed{u^*(t) = -\operatorname{sgn}\left(\lambda_2^*(t)\right)}$$

when $\lambda_2^*(t) \neq 0$.

Key point
The minimum principle does not permit arbitrary changes in the state or costate while minimizing the Hamiltonian. These quantities are frozen at their optimal values.

Possible Shapes of the Switching Function

The switching function is

$$\sigma(t) = \lambda_2(t) = -C_1 t + C_2.$$

The possibilities are:

1. positive for the entire interval;
2. negative for the entire interval;
3. positive initially and negative later;
4. negative initially and positive later; or
5. constant.

The first four cases correspond to zero or one switch.
The constant case requires

$$C_1 = 0.$$

Then

$$\lambda_1(t) \equiv 0,$$

and

$$\lambda_2(t) \equiv C_2.$$

Can the Switching Function Be Constant?

The autonomous Mayer-form Hamiltonian must satisfy

$$H^*(t) = \lambda_1^* x_2^* + \lambda_2^* u^* = -1.$$

If $C_1 = 0$, then

$$\lambda_1^*(t) = 0,$$

so

$$\lambda_2^*(t)u^*(t) = -1.$$

A nonzero constant λ_2 can satisfy this identity only for a compatible constant bang control and a specific multiplier normalization. Such cases do not introduce a singular arc because the sign of λ_2 still determines the control.

The important problematic case is

$$\lambda_2(t) \equiv 0.$$

Then

$$H^*(t) = 0,$$

which contradicts

$$H^*(t) \equiv -1.$$

Therefore,

$$\boxed{\lambda_2(t) \text{ cannot be identically zero on a finite interval.}}$$

Definition of a Singular Arc

Let the switching function be

$$\sigma(t) = \frac{\partial H}{\partial u}.$$

A **singular arc** or **singular interval** is a nonzero-duration interval

$$, \qquad t_2 > t_1,$$

on which

$$\sigma(t) \equiv 0.$$

On such an interval, the usual Hamiltonian minimization condition does not determine the control directly.

For the minimum-time double integrator,

$$\sigma(t) = \lambda_2(t).$$

Since λ_2 cannot vanish identically on a finite interval, the standard minimum-time double integrator has no singular arc.

Isolated Switching Time Versus Singular Arc

At a normal switching time t_s,

$$\sigma(t_s) = 0,$$

but

$$\sigma(t) \neq 0$$

immediately before and after the switch.

A singular arc instead satisfies

$$\sigma(t) = 0$$

throughout an interval.

Therefore,

An isolated zero of the switching function is not a singular arc.

General Mechanism Producing Singular Arcs

Consider

$$H = L(\boldsymbol{x}, t) + \boldsymbol{\lambda}^{\mathsf{T}} \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, t),$$

where L does not depend explicitly on the control.

If the control-dependent contribution becomes insensitive to u along an interval, then Hamiltonian minimization cannot determine the optimal control there.

For a control-affine system,

$$\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, t) = \boldsymbol{f}_0(\boldsymbol{x}, t) + \boldsymbol{B}(\boldsymbol{x}, t)\boldsymbol{u},$$

the Hamiltonian is

$$H = L(\boldsymbol{x}, t) + \boldsymbol{\lambda}^{\mathsf{T}} \boldsymbol{f}_0 + \boldsymbol{\lambda}^{\mathsf{T}} \boldsymbol{B}\boldsymbol{u}.$$

The switching function is

$$\sigma(t) = \frac{\partial H}{\partial \boldsymbol{u}} = \boldsymbol{B}^{\top}\boldsymbol{\lambda}.$$

A singular arc occurs when

$$\sigma(t) \equiv \mathbf{0}$$

on a finite interval.

Determining a Singular Control

On a singular interval, one differentiates the switching function until the control appears explicitly:

$$\begin{aligned}
\sigma(t) &= \mathbf{0}, \\
\dot{\sigma}(t) &= \mathbf{0}, \\
\ddot{\sigma}(t) &= \mathbf{0}, \\
&\vdots
\end{aligned}$$

Suppose the control first appears in the rth derivative:

$$\sigma^{(r)} = \boldsymbol{a}(\boldsymbol{x}, \boldsymbol{\lambda}, t) + \boldsymbol{B}_s(\boldsymbol{x}, \boldsymbol{\lambda}, t)\boldsymbol{u}.$$

Then the candidate singular control satisfies

$$\boxed{\boldsymbol{u}_s = -\boldsymbol{B}_s^{-1}\boldsymbol{a},}$$

provided the required inverse exists.
This candidate must also satisfy:

- the control bounds;
- the state and costate equations;
- endpoint and continuity conditions;
- generalized Legendre–Clebsch conditions; and
- global consistency with neighboring arcs.

Why Singular Arcs Are Numerically Difficult

Singular arcs are challenging because:

- the switching function may be very small without being identically zero;
- bang and singular arcs may meet at unknown times;
- high-order derivatives may be required;
- numerical differentiation amplifies error;
- control bounds must still be enforced; and

- the resulting boundary-value problem becomes highly sensitive.

A numerical method may incorrectly approximate a singular arc as:

- rapid switching;
- chattering;
- a smoothed bang-bang transition; or
- an interior control with no correct theoretical basis.

Examples Where Singular Arcs Arise

Singular arcs appear in many applications, including:

- Goddard rocket ascent;
- chemical reactors;
- bioreactor feeding;
- aerospace trajectory optimization;
- energy management;
- thermal processes; and
- systems with one-sided actuation.

In the Goddard rocket problem, the optimal thrust structure can contain:

$$\text{maximum thrust} \longrightarrow \text{singular thrust} \longrightarrow \text{minimum thrust}.$$

The middle segment cannot be determined from the basic switching-function sign alone.

One-Sided Control and Singular Behavior

Many mechanical systems permit two-sided actuation:

$$-u_{\max} \leq u \leq u_{\max}.$$

Many chemical and process systems permit only one-sided actuation:

$$0 \leq u \leq u_{\max}.$$

For example, material may be added to a reactor but not removed through the same input channel.
One-sided control often creates:

- asymmetric bang arcs;
- boundary arcs;
- long zero-control intervals; and
- singular arcs.

Common Errors

1. Treating the Mayer and Lagrange minimum-time Hamiltonians as numerically identical.
2. Forgetting that the two Hamiltonians differ by a constant.
3. Calling a general endpoint-plus-integral cost "Mayer form" instead of Bolza form.
4. Adding an accumulated-cost state but forgetting its initial condition.
5. Minimizing the Hamiltonian while allowing the state and costate to vary.
6. Calling every zero of the switching function a singular arc.
7. Assuming that $\partial H/\partial u = 0$ determines the control on a singular interval.
8. Ignoring Hamiltonian transversality when testing whether a singular arc is possible.

Summary

The main conclusions are:

1. Minimum time can be represented in Mayer or Lagrange form.
2. The Mayer Hamiltonian is constant at -1 under the convention used here.
3. The equivalent Lagrange Hamiltonian is constant at zero.
4. Bolza form contains both endpoint and running costs.
5. Every Lagrange problem can be converted to Mayer form by adding one accumulated-cost state.
6. Pontryagin's Minimum Principle varies only the control while holding the optimal state and costate fixed.
7. The minimum-time double-integrator switching function is affine in time.
8. An identically zero switching function is incompatible with the Hamiltonian condition in this problem.
9. A singular arc is a finite interval on which the switching function vanishes identically.
10. Singular controls require differentiated switching conditions and additional optimality tests.

Connection. Bang–bang reasoning ceases to determine the control when the switching function vanishes over an interval; this is the origin of singular control.

Singular Optimal Control Theory

A singular arc is a nonzero-duration interval on which the standard first-order optimality conditions fail to determine the optimal control. Singular arcs are important because an optimal control may lie strictly inside the admissible set while still being unobtainable from the usual stationarity equation.

This section develops:

- strong and weak Hamiltonian optimality conditions;
- the definition of singular intervals;
- the switching function;
- differentiation-based recovery of singular controls;
- order of a singular arc;
- generalized Legendre–Clebsch conditions;
- junction conditions between bang and singular arcs; and
- numerical difficulties associated with singular solutions.

Strong and Weak Optimality Conditions

Consider

$$\min_{\boldsymbol{u}(\cdot)} J = \Phi(\boldsymbol{x}(t_f), t_f) + \int_{t_0}^{t_f} L(\boldsymbol{x}, \boldsymbol{u}, t)\,\mathrm{d}t$$

subject to

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, t), \qquad \boldsymbol{u}(t) \in \mathcal{U}.$$

The Hamiltonian is

$$H(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\lambda}, t) = L(\boldsymbol{x}, \boldsymbol{u}, t) + \boldsymbol{\lambda}^{\mathsf{T}} \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, t).$$

If the optimal control lies in the interior of $\mathcal{U}$ and the Hamiltonian is differentiable in $\boldsymbol{u}$, then the stationarity condition is

$$\boxed{\frac{\partial H}{\partial \boldsymbol{u}} = \boldsymbol{0}.}$$

However, this implication is one-way:

$$\frac{\partial H}{\partial \boldsymbol{u}} = \boldsymbol{0} \quad \Rightarrow \quad \boldsymbol{u}^* \text{ may be interior},$$

but an interior control does not guarantee that stationarity determines it uniquely.

The general pointwise minimum condition is

$$\boxed{H(\boldsymbol{x}^*, \boldsymbol{u}^*, \boldsymbol{\lambda}^*, t) \le H(\boldsymbol{x}^*, \boldsymbol{u}, \boldsymbol{\lambda}^*, t) \quad \forall \boldsymbol{u} \in \mathcal{U}.}$$

Key point
During Hamiltonian minimization, the state and costate are fixed at their optimal values. Only the control is varied.

Definition of a Singular Arc

Let

$$\boldsymbol{\sigma}(t) = \frac{\partial H}{\partial \boldsymbol{u}}$$

be the switching function.

A singular interval is a finite interval

$$, \qquad t_2 > t_1,$$

on which

$$\boxed{\boldsymbol{\sigma}(t) \equiv \mathbf{0}.}$$

On this interval, neither the sign of the switching function nor the stationarity condition directly determines the control.

An isolated zero,

$$\boldsymbol{\sigma}(t_s) = \mathbf{0},$$

is generally a switching time, not a singular arc.

Autonomous Hamiltonians

Along an optimal solution,

$$\frac{\mathrm{d}H^*}{\mathrm{d}t} = \frac{\partial H}{\partial t}.$$

If H is not an explicit function of time,

$$\frac{\partial H}{\partial t} = 0,$$

and therefore

$$\boxed{H^*(t) = \text{constant}.}$$

This property is useful because repeated time differentiation of a constant Hamiltonian produces zero:

$$\frac{\mathrm{d}^q H^*}{\mathrm{d}t^q} = 0, \qquad q \geq 1.$$

In practice, singular-control derivations are more often formulated by differentiating the switching function rather than the Hamiltonian itself.

Control-Affine Systems

Consider

$$\dot{\boldsymbol{x}} = \boldsymbol{f}_0(\boldsymbol{x}, t) + \boldsymbol{B}(\boldsymbol{x}, t)\boldsymbol{u}$$

with running cost independent of $\boldsymbol{u}$.

Then

$$H = L(\boldsymbol{x}, t) + \boldsymbol{\lambda}^\top \boldsymbol{f}_0 + \boldsymbol{\lambda}^\top \boldsymbol{B}\boldsymbol{u}.$$

The switching function is

$$\boxed{\boldsymbol{\sigma} = \boldsymbol{B}^\top \boldsymbol{\lambda}.}$$

For a scalar bounded control,

$$u_{\min} \leq u \leq u_{\max},$$

Hamiltonian minimization gives

$$u^* = \begin{cases} u_{\min}, & \sigma > 0, \\ u_{\max}, & \sigma < 0, \\ \text{undetermined}, & \sigma = 0. \end{cases}$$

The third case is the singular candidate.

Recovering the Singular Control

On a singular interval,

$$\sigma = 0.$$

Differentiate until the control appears explicitly:

$$\begin{aligned}\sigma &= 0,\\ \dot{\sigma} &= 0,\\ \ddot{\sigma} &= 0,\\ &\vdots\\ \sigma^{(q)} &= 0.\end{aligned}$$

Suppose the first derivative containing u is

$$\sigma^{(q)} = a(\boldsymbol{x}, \boldsymbol{\lambda}, t) + b(\boldsymbol{x}, \boldsymbol{\lambda}, t)u.$$

Then the candidate singular control is

$$\boxed{u_s = -\frac{a}{b},}$$

provided $b \neq 0$.
The control must satisfy

$$u_s \in [u_{\min}, u_{\max}].$$

If it violates the bounds, the candidate singular arc is inadmissible.

Order of a Singular Arc

For many control-affine problems, the control first appears in an even derivative:

$$\frac{\mathrm{d}^{2r}\sigma}{\mathrm{d}t^{2r}}.$$

Then the singular arc is said to have order r.
A common structure is

$$\frac{\mathrm{d}^{2r}\sigma}{\mathrm{d}t^{2r}} = A(\boldsymbol{x}, \boldsymbol{\lambda}, t) + B(\boldsymbol{x}, \boldsymbol{\lambda}, t)u.$$

The singular control becomes

$$u_s = -B^{-1}A.$$

Generalized Legendre–Clebsch Condition

A singular control satisfying the differentiated switching equations is only a candidate.
For a minimization problem, a necessary higher-order condition is

$$\boxed{(-1)^r \frac{\partial}{\partial u}\left(\frac{\mathrm{d}^{2r}}{\mathrm{d}t^{2r}}\frac{\partial H}{\partial u}\right) \geq 0.}$$

This is the generalized Legendre–Clebsch condition.
Failure of this condition rules out the candidate singular arc as a local minimizer.

Bang, Singular, and Boundary Arcs

A constrained optimal trajectory can contain:

- a lower-bound bang arc, $u = u_{\min}$;
- an upper-bound bang arc, $u = u_{\max}$;
- a singular arc, $u = u_s$;
- a state-constraint boundary arc; or
- combinations of these.

A typical structure is

$$u_{\max} \longrightarrow u_s \longrightarrow u_{\min}.$$

The entry and exit times are unknown event times that must be solved as part of the optimal-control problem.

Junction Conditions

At a junction between bang and singular arcs, the state is normally continuous:

$$\boldsymbol{x}(t_s^-) = \boldsymbol{x}(t_s^+).$$

The costate is also continuous when no interior point constraint or impulse is present:

$$\boldsymbol{\lambda}(t_s^-) = \boldsymbol{\lambda}(t_s^+).$$

The switching conditions satisfy

$$\sigma(t_s) = 0,$$

and, depending on the singular order,

$$\dot{\sigma}(t_s) = 0, \quad \ldots, \quad \sigma^{(2r-1)}(t_s) = 0.$$

These equations help determine the unknown switching times and interface values.

Why Singular Arcs Are Difficult Numerically

Singular arcs are difficult because:

1. the switching interval is unknown;
2. σ may be numerically small without being identically zero;
3. differentiation magnifies numerical noise;
4. bang and singular arcs may connect through sensitive junctions;
5. a discretized solution may approximate a singular arc by rapid switching;
6. the singular control may violate its bounds; and
7. multiple singular arcs may occur.

Key point
A smooth interior control produced by a numerical optimizer is not automatically a singular control. The switching-function conditions must be checked.

A Practical Singular-Arc Analysis Procedure

1. Form the Hamiltonian.
2. Derive state and costate equations.
3. Derive the switching function.
4. Determine the bang controls from the sign of the switching function.
5. Test whether $\sigma \equiv 0$ is theoretically possible.
6. Differentiate σ until the control appears.
7. Solve for the singular control.
8. check admissibility and higher-order conditions.
9. derive junction conditions.
10. solve for switching times and trajectory parameters.
11. verify all endpoint and Hamiltonian conditions.

Summary

1. Singular arcs occur when first-order conditions do not determine the control.
2. An isolated zero of the switching function is not a singular interval.
3. The control is recovered by differentiating the switching function until u appears.
4. The resulting singular control is only a candidate.
5. Admissibility and generalized Legendre–Clebsch conditions must be checked.
6. Bang–singular–bang trajectories require unknown junction times.
7. Numerical singular-arc detection is intrinsically difficult.

Connection. The Goddard rocket problem turns the abstract singular-arc conditions into a physical thrust profile with bang, singular, and coast phases.

The Goddard Rocket Problem and Singular Thrust

The Goddard rocket problem is a classical optimal-control problem in which a vertically ascending rocket uses a finite amount of fuel to maximize altitude. Despite its simple physical statement, the optimal thrust can contain a singular interval.

This section develops:

- the one-dimensional rocket model;
- the Mayer objective;
- the Hamiltonian and costate equations;
- the thrust switching function;
- bang and singular thrust conditions;
- differentiation of the singular condition;
- bang–singular–coast structure; and
- computational implications.

Problem Statement

Let:

- $h(t)$ be altitude;
- $v(t)$ be vertical velocity;
- $m(t)$ be mass;
- $T(t)$ be thrust;
- $D(h, v)$ be aerodynamic drag;
- g be gravitational acceleration; and
- c be effective exhaust velocity.

The objective is to maximize final altitude:

$$\max h(t_f).$$

Using minimization notation,

$$\boxed{\min J = -h(t_f).}$$

The final time is free.

Dynamics

The one-dimensional ascent dynamics are

$$\begin{aligned}\dot{h} &= v,\\ \dot{v} &= \frac{T - D(h,v)}{m} - g,\\ \dot{m} &= -\frac{T}{c}.\end{aligned}$$

The thrust is bounded:

$$\boxed{0 \le T(t) \le T_{\max}.}$$

Typical initial conditions are

$$\begin{aligned}h(0) &= 0,\\ v(0) &= 0,\\ m(0) &= m_0.\end{aligned}$$

A fixed terminal mass is imposed:

$$m(t_f) = m_f.$$

The final altitude and final velocity are free.

Hamiltonian

Because the cost is in Mayer form and there is no running cost,

$$H = \lambda_h \dot{h} + \lambda_v \dot{v} + \lambda_m \dot{m}.$$

Therefore,

$$\boxed{H = \lambda_h v + \lambda_v \left(\frac{T - D}{m} - g \right) - \lambda_m \frac{T}{c}.}$$

Group the thrust-dependent terms:

$$H = \lambda_h v - \lambda_v \left(\frac{D}{m} + g \right) + \left(\frac{\lambda_v}{m} - \frac{\lambda_m}{c} \right) T.$$

Hamiltonian Constancy

The Hamiltonian has no explicit time dependence:

$$\frac{\partial H}{\partial t} = 0.$$

Thus,

$$H^*(t) = \text{constant}.$$

Since the final time is free and the terminal cost has no explicit dependence on t_f,

$$H^*(t_f) = 0.$$

Therefore,

$$\boxed{H^*(t) \equiv 0.}$$

Costate Equations

The costate equations are

$$\dot{\lambda}_i = -\frac{\partial H}{\partial x_i}.$$

For altitude:

$$\dot{\lambda}_h = \frac{\lambda_v}{m}\frac{\partial D}{\partial h}.$$

For velocity:

$$\boxed{\dot{\lambda}_v = -\lambda_h + \frac{\lambda_v}{m}\frac{\partial D}{\partial v}.}$$

For mass:

$$\boxed{\dot{\lambda}_m = \lambda_v \frac{T - D}{m^2}.}$$

Terminal Transversality Conditions

The terminal cost is

$$\Phi = -h(t_f).$$

Since $h(t_f)$ and $v(t_f)$ are free,

$$\lambda_h(t_f) = \frac{\partial \Phi}{\partial h(t_f)} = -1,$$
$$\lambda_v(t_f) = \frac{\partial \Phi}{\partial v(t_f)} = 0.$$

The terminal mass is fixed, so $\lambda_m(t_f)$ is not prescribed directly.

Switching Function

The Hamiltonian is linear in thrust. Define

$$\boxed{\psi(t) = \frac{\partial H}{\partial T} = \frac{\lambda_v}{m} - \frac{\lambda_m}{c}.}$$

Pontryagin's Minimum Principle gives

$$T^* = \begin{cases} T_{\max}, & \psi < 0, \\ 0, & \psi > 0, \\ \text{undetermined}, & \psi = 0. \end{cases}$$

Thus:

- $\psi < 0$ gives maximum thrust;
- $\psi > 0$ gives zero thrust; and
- $\psi = 0$ is the singular candidate.

Equivalent Singular Condition

Multiply the switching function by cm:

$$cm\psi = c\lambda_v - m\lambda_m.$$

Define

$$\boxed{I = c\lambda_v - m\lambda_m.}$$

On a singular arc,

$$I = 0.$$

Since $c > 0$ and $m > 0$, this condition is equivalent to $\psi = 0$.

First Derivative of the Singular Condition

Differentiate

$$I = c\lambda_v - m\lambda_m.$$

Then

$$\dot{I} = c\dot{\lambda}_v - \dot{m}\lambda_m - m\dot{\lambda}_m.$$

Substitute

$$\dot{\lambda}_v = -\lambda_h + \frac{\lambda_v}{m} D_v,$$
$$\dot{m} = -\frac{T}{c},$$
$$\dot{\lambda}_m = \lambda_v \frac{T - D}{m^2},$$

where

$$D_h = \frac{\partial D}{\partial h}, \qquad D_v = \frac{\partial D}{\partial v}.$$

This gives

$$\dot{I} = -c\lambda_h + \frac{c\lambda_v}{m} D_v + \frac{T}{c}\lambda_m - \frac{\lambda_v}{m}(T - D).$$

Rearranging:

$$\dot{I} = -c\lambda_h + \frac{c\lambda_v}{m} D_v + \frac{\lambda_v D}{m} + T\left(\frac{\lambda_m}{c} - \frac{\lambda_v}{m}\right).$$

On the singular arc,

$$\frac{\lambda_v}{m} - \frac{\lambda_m}{c} = 0.$$

Hence the thrust-dependent term vanishes and

$$\boxed{\dot{I} = -c\lambda_h + \frac{c\lambda_v}{m} D_v + \frac{\lambda_v D}{m} = 0.}$$

The first derivative still does not determine T.

Second Derivative and Singular Thrust

Differentiate $\dot{I} = 0$ once more:

$$\ddot{I} = 0.$$

After substituting the state and costate dynamics, the second derivative has the general form

$$\boxed{\ddot{I} = A(h, v, m, \lambda_h, \lambda_v, \lambda_m) + B(h, v, m, \lambda_h, \lambda_v, \lambda_m)T.}$$

Therefore, the candidate singular thrust is

$$\boxed{T_s = -\frac{A}{B},}$$

provided $B \neq 0$.

The explicit expressions depend on the selected drag law $D(h, v)$.
The candidate is admissible only if

$$0 \le T_s \le T_{\max}.$$

A Common Drag Model

A frequently used model is

$$D(h, v) = \frac{1}{2}\rho(h) C_D A v^2,$$

where

$$\rho(h) = \rho_0 e^{-h/H_s}.$$

Then

$$D_h = -\frac{D}{H_s},$$
$$D_v = \frac{2D}{v} \qquad (v \neq 0).$$

These expressions simplify the costate and singular-thrust equations.

Expected Optimal Thrust Structure

The rocket must initially generate enough thrust to lift off, so the first arc is generally

$$T^* = T_{\max}.$$

A classical optimal structure is

$$\boxed{T_{\max} \longrightarrow T_s \longrightarrow 0.}$$

This is a bang–singular–bang structure, where the final "bang" is the lower thrust bound.

Entry and Exit Conditions

The unknown singular entry time t_1 and exit time t_2 satisfy:

$$I(t_1) = 0,$$
$$\dot{I}(t_1) = 0,$$
$$I(t_2) = 0,$$
$$\dot{I}(t_2) = 0.$$

The state and costate must also be continuous:

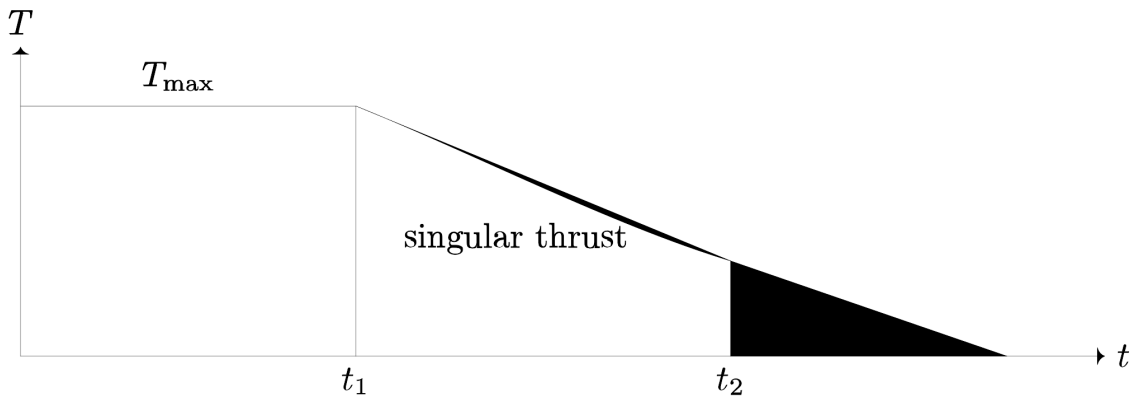

Figure 18: Expected bang–singular–bang thrust structure for the Goddard rocket problem.

$$\begin{aligned} \boldsymbol{x}(t_i^-) &= \boldsymbol{x}(t_i^+), \\ \boldsymbol{\lambda}(t_i^-) &= \boldsymbol{\lambda}(t_i^+), \qquad i = 1, 2. \end{aligned}$$

These conditions, together with the endpoint conditions and integrated dynamics, determine the full multiphase solution.

Why Maximum Thrust Followed by Zero Is Not Always Optimal

A naive strategy is

$$T_{\max} \longrightarrow 0.$$

This ignores the interaction among:

- fuel consumption;
- atmospheric drag;
- changing mass;
- velocity-dependent drag losses; and
- the value of thrust at different altitudes.

The singular arc balances these effects. It can reduce drag losses while still using the available fuel effectively.

Numerical Formulation

A practical numerical solution can be posed as a multiphase problem:

1. Phase 1: $T = T_{\max}$.
2. Phase 2: $T = T_s$ and $I = \dot{I} = 0$.
3. Phase 3: $T = 0$.

Unknowns include:

- initial costates;
- t_1;
- t_2;
- t_f;
- singular-arc state and costate values; and
- possibly parameters of the drag model.

Continuity and endpoint conditions form the nonlinear residual system.

Verification Conditions

A computed solution should satisfy:

1. state dynamics;
2. costate dynamics;
3. thrust bounds;
4. Hamiltonian minimization;
5. $H^*(t) = 0$;
6. terminal transversality;
7. $I = \dot{I} = 0$ on the singular arc;
8. continuity at t_1 and t_2; and
9. a generalized Legendre–Clebsch condition.

Common Errors

1. Using $\partial H/\partial T = 0$ and assuming it directly gives thrust.
2. Ignoring the possibility $\psi = 0$ over an interval.
3. Treating an isolated switching point as a singular arc.
4. Forgetting that final time is free.
5. Omitting the terminal Hamiltonian condition.
6. Using the wrong sign in the thrust switching law.
7. Assuming a candidate singular thrust is automatically admissible.
8. Failing to enforce continuity at the singular entry and exit points.

Summary

1. The Goddard rocket problem maximizes terminal altitude using bounded thrust.
2. The Hamiltonian is linear in thrust.
3. The switching function is

$$\psi = \frac{\lambda_v}{m} - \frac{\lambda_m}{c}.$$

4. Negative ψ gives maximum thrust and positive ψ gives zero thrust.
5. $\psi \equiv 0$ defines a singular candidate.
6. The first derivative of the singular condition does not contain thrust.
7. The second derivative introduces thrust and yields a candidate singular law.
8. The optimal trajectory may have a maximum-thrust, singular-thrust, zero-thrust structure.
9. Entry and exit times must be solved as unknown events.
10. The problem illustrates how simple dynamics can produce highly nontrivial optimal controls.

Connection. Even an exact set of state–costate equations may be numerically ill conditioned, so the coupled Hamiltonian dynamics and shooting sensitivity deserve separate study.

Coupled State–Costate Dynamics, Hamiltonian Matrices, and Shooting Instability

The first-order necessary conditions of optimal control form a coupled dynamical system. The state and costate cannot generally be solved independently because the optimal control depends on both.

This section develops that structure through a scalar quadratic optimal-control example and uses the result to explain why simple shooting can fail, even when:

- the original dynamics are linear;
- the optimality equations are solved analytically;
- there is no numerical integration error; and
- the initial costate guess is extremely accurate.

The main topics are:

1. coupled state–costate equations;
2. the costate as a sensitivity or dual variable;
3. a fixed-endpoint quadratic example;
4. eigenvalue and eigenvector analysis;
5. Hamiltonian matrix structure;
6. spectral symmetry of Hamiltonian systems;
7. propagation of shooting errors;
8. long-horizon instability; and
9. motivation for robust numerical optimal-control methods.

Canonical State–Costate Equations

Consider the optimal-control problem

$$\min_{\boldsymbol{u}(\cdot)} \left[\Phi(\boldsymbol{x}(t_f), t_f) + \int_{t_0}^{t_f} L(\boldsymbol{x}, \boldsymbol{u}, t)\, \mathrm{d}t \right],$$

subject to

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, t).$$

The Hamiltonian is

$$H(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\lambda}, t) = L(\boldsymbol{x}, \boldsymbol{u}, t) + \boldsymbol{\lambda}^{\mathsf{T}} \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, t).$$

The canonical equations are

$$\dot{\boldsymbol{x}} = \left(\frac{\partial H}{\partial \boldsymbol{\lambda}} \right)^{\mathsf{T}},$$

$$\dot{\boldsymbol{\lambda}} = -\left(\frac{\partial H}{\partial \boldsymbol{x}} \right)^{\mathsf{T}}.$$

Expanding the costate equation gives

$$\dot{\boldsymbol{\lambda}} = -\left[\frac{\partial L}{\partial \boldsymbol{x}} + \boldsymbol{\lambda}^{\mathsf{T}} \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\right]^{\mathsf{T}}.$$

Thus, the costate generally depends on:

- the state;
- the control;
- the costate itself; and
- possibly time.

Why the State and Costate Are Coupled

Suppose the pointwise optimality condition gives

$$\boldsymbol{u}^* = \boldsymbol{u}^*(\boldsymbol{x}^*, \boldsymbol{\lambda}^*, t).$$

Substitution into the state equation gives

$$\dot{\boldsymbol{x}}^* = \boldsymbol{f}(\boldsymbol{x}^*, \boldsymbol{u}^*(\boldsymbol{x}^*, \boldsymbol{\lambda}^*, t), t)\,.$$

Therefore,

$$\dot{\boldsymbol{x}}^* = \boldsymbol{g}_x(\boldsymbol{x}^*, \boldsymbol{\lambda}^*, t).$$

Similarly,

$$\dot{\boldsymbol{\lambda}}^* = \boldsymbol{g}_\lambda(\boldsymbol{x}^*, \boldsymbol{\lambda}^*, t).$$

The combined optimality system is

$$\boxed{\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix}\boldsymbol{x}^* \\ \boldsymbol{\lambda}^*\end{bmatrix} = \begin{bmatrix}\boldsymbol{g}_x(\boldsymbol{x}^*, \boldsymbol{\lambda}^*, t) \\ \boldsymbol{g}_\lambda(\boldsymbol{x}^*, \boldsymbol{\lambda}^*, t)\end{bmatrix}.}$$

The two systems must be solved simultaneously.

> **Key point**
> The costate is not an auxiliary differential equation that can generally be solved first and substituted afterward. The state and costate form one coupled boundary-value problem.

Interpretation of the Costate

The costate is a dual or adjoint variable associated with the state dynamics.
Conceptually, it measures the sensitivity of the optimal cost to a perturbation in the state.

A common interpretation is

$$\lambda_i(t) \sim \frac{\partial J^*}{\partial x_i(t)},$$

where J^* denotes the optimal value function, subject to sign and formulation conventions.

If perturbing the state had no influence on the cost, then there would be no meaningful optimization pressure associated with that state direction.

The costate transmits this sensitivity through the optimality system and directly affects the optimal control.

A Scalar Quadratic Example

Consider

$$\boxed{\min_{u(\cdot)} J = \frac{1}{2}\int_0^{t_f} \left(x^2 + u^2\right)\, \mathrm{d}t}$$

subject to

$$\dot{x} = x + u.$$

The time horizon t_f is fixed, and the state is fixed at both endpoints:

$$x(0) = x_0,$$
$$x(t_f) = x_f.$$

There is no terminal cost.

Hamiltonian and Optimal Control

The Hamiltonian is

$$H = \frac{1}{2}\left(x^2 + u^2\right) + \lambda(x + u).$$

The Hamiltonian is quadratic in u, and the control is unconstrained. Therefore, the stationarity condition applies:

$$\frac{\partial H}{\partial u} = u + \lambda = 0.$$

Hence,

$$\boxed{u^* = -\lambda^*.}$$

This formula explicitly demonstrates that the optimal control depends on the costate.

Coupled Optimality Dynamics

The state equation becomes

$$\dot{x}^* = x^* - \lambda^*.$$

The costate equation is

$$\begin{aligned}\dot{\lambda}^* &= -\frac{\partial H}{\partial x} \\ &= -\left(x^* + \lambda^*\right).\end{aligned}$$

Therefore,

$$\boxed{\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix} x \\ \lambda \end{bmatrix} = \begin{bmatrix} 1 & -1 \\ -1 & -1 \end{bmatrix}\begin{bmatrix} x \\ \lambda \end{bmatrix}.}$$

Define

$$\boldsymbol{p} = \begin{bmatrix} x \\ \lambda \end{bmatrix}, \qquad \boldsymbol{A} = \begin{bmatrix} 1 & -1 \\ -1 & -1 \end{bmatrix}.$$

Then

$$\dot{\boldsymbol{p}} = \boldsymbol{A}\boldsymbol{p}.$$

Boundary Conditions

The state is fixed at both endpoints:

$$\begin{aligned}x(0) &= x_0, \\ x(t_f) &= x_f.\end{aligned}$$

Because both endpoint values of the state are fixed, the costate has no directly prescribed initial or final value. Thus, the problem is a two-point boundary-value problem with:

- one known state condition at $t = 0$;
- one known state condition at $t = t_f$; and
- an unknown initial costate that must be selected correctly.

Eigenvalue Analysis

For

$$\dot{\boldsymbol{p}} = \boldsymbol{A}\boldsymbol{p},$$

the eigenvalues satisfy

$$\det(\mu \boldsymbol{I} - \boldsymbol{A}) = 0.$$

Here,

$$\mu \boldsymbol{I} - \boldsymbol{A} = \begin{bmatrix} \mu - 1 & 1 \\ 1 & \mu + 1 \end{bmatrix}.$$

Therefore,

$$\begin{aligned} 0 &= (\mu - 1)(\mu + 1) - 1 \\ &= \mu^2 - 2. \end{aligned}$$

Hence,

$$\boxed{\mu_{1,2} = \pm\sqrt{2}.}$$

One eigenvalue is negative and the other is positive.
Thus, the state–costate dynamics contain:

- one exponentially decaying mode in forward time; and
- one exponentially growing mode in forward time.

Eigenvectors

For

$$\mu_1 = -\sqrt{2},$$

an eigenvector satisfies

$$(\mu_1 \boldsymbol{I} - \boldsymbol{A})\boldsymbol{v}_1 = \boldsymbol{0}.$$

Choosing the second component equal to one gives

$$\boldsymbol{v}_1 = \begin{bmatrix} \dfrac{1}{1 + \sqrt{2}} \\ 1 \end{bmatrix}.$$

For

$$\mu_2 = +\sqrt{2},$$

one may choose

$$\boldsymbol{v}_2 = \begin{bmatrix} 1 \\ \overline{1-\sqrt{2}} \\ 1 \end{bmatrix}.$$

The general solution is

$$\boxed{\boldsymbol{p}(t) = C_1 \boldsymbol{v}_1 e^{-\sqrt{2}t} + C_2 \boldsymbol{v}_2 e^{+\sqrt{2}t}.}$$

Stable and Unstable Modes

In forward time:

$$e^{-\sqrt{2}t} \longrightarrow 0,$$

whereas

$$e^{+\sqrt{2}t} \longrightarrow \infty.$$

In backward time, the roles reverse.
Therefore, integrating the full Hamiltonian system in either direction generally exposes one unstable mode.

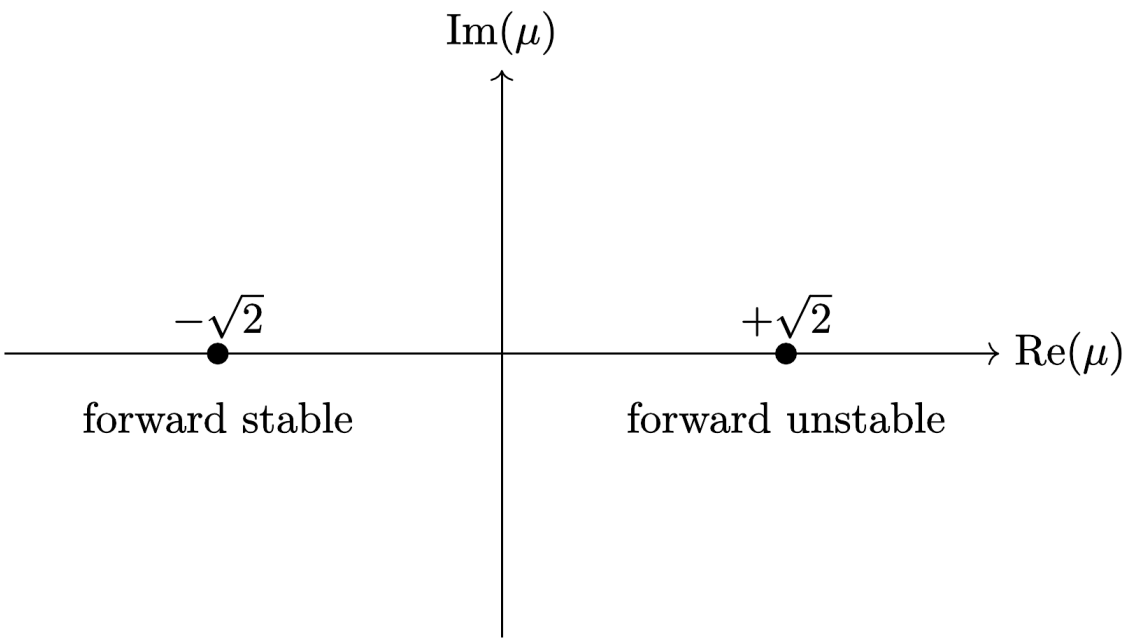

Figure 19: A Hamiltonian system exposes paired stable and unstable eigenvalues during forward integration.

Hamiltonian Matrix Structure

For a general canonical system,

$$\dot{\boldsymbol{p}} = \boldsymbol{g}(\boldsymbol{p}), \qquad \boldsymbol{p} = \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{\lambda} \end{bmatrix},$$

the Jacobian of the Hamiltonian vector field has the block form

$$\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{p}} = \begin{bmatrix} H_{\lambda x} & H_{\lambda\lambda} \\ -H_{xx} & -H_{x\lambda} \end{bmatrix}.$$

This is a Hamiltonian matrix under the standard smoothness and symmetry assumptions on second derivatives.
A real matrix $\boldsymbol{A}$ is Hamiltonian if

$$\boldsymbol{A}^{\top}\boldsymbol{J} + \boldsymbol{J}\boldsymbol{A} = \boldsymbol{0},$$

where

$$\boldsymbol{J} = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{I} \\ -\boldsymbol{I} & \boldsymbol{0} \end{bmatrix}.$$

Equivalently,

$$\boldsymbol{J}\boldsymbol{A}$$

is symmetric.

Spectral Symmetry

Hamiltonian matrices have a characteristic eigenvalue symmetry.
If

$$\mu$$

is an eigenvalue, then the spectrum also contains

$$-\mu.$$

For real matrices, complex conjugacy adds

$$\overline{\mu} \quad \text{and} \quad -\overline{\mu}.$$

Thus, a complex eigenvalue generally appears in a quartet:

$$\boxed{\mu,\ -\mu,\ \overline{\mu},\ -\overline{\mu}.}$$

For a two-dimensional Hamiltonian system, only two eigenvalues exist. Therefore, a nonreal eigenvalue with nonzero real part would require four eigenvalues and cannot occur. The pair must instead lie:

- symmetrically on the real axis; or
- symmetrically on the imaginary axis.

In the present example, the pair is

$$\pm\sqrt{2}.$$

Simple Shooting Formulation

Suppose the unknown initial costate is guessed as

$$\lambda(0) = \bar{\lambda}_0.$$

Then

$$\begin{bmatrix} x(t) \\ \lambda(t) \end{bmatrix} = e^{At} \begin{bmatrix} x_0 \\ \bar{\lambda}_0 \end{bmatrix}.$$

The shooting method adjusts $\bar{\lambda}_0$ until

$$x(t_f) = x_f.$$

Define the residual

$$R(\bar{\lambda}_0) = x(t_f; \bar{\lambda}_0) - x_f.$$

The desired initial costate satisfies

$$R(\lambda_0^*) = 0.$$

Propagation of Initial-Costate Error

Let the exact initial costate be

$$\lambda_0^*.$$

Define the initial error

$$\delta\lambda_0 = \bar{\lambda}_0 - \lambda_0^*.$$

The trajectory error is

$$\boldsymbol{e}(t) = \begin{bmatrix} x(t) - x^*(t) \\ \lambda(t) - \lambda^*(t) \end{bmatrix}.$$

Because the initial state is exact,

$$\boldsymbol{e}(0) = \begin{bmatrix} 0 \\ \delta\lambda_0 \end{bmatrix}.$$

Therefore,

$$\boxed{\boldsymbol{e}(t) = e^{\boldsymbol{A}t} \begin{bmatrix} 0 \\ \delta\lambda_0 \end{bmatrix}.}$$

Using the eigendecomposition

$$\boldsymbol{A} = \boldsymbol{V}\boldsymbol{D}\boldsymbol{V}^{-1},$$

we have

$$e^{\boldsymbol{A}t} = \boldsymbol{V} \begin{bmatrix} e^{-\sqrt{2}t} & 0 \\ 0 & e^{+\sqrt{2}t} \end{bmatrix} \boldsymbol{V}^{-1}.$$

Unless the initial error lies exactly in the stable eigenspace, it has a component that grows like

$$e^{+\sqrt{2}t}.$$

Long-Horizon Error Amplification

Suppose the initial costate error has magnitude

$$|\delta\lambda_0| = \varepsilon.$$

The unstable contribution at the final time behaves approximately as

$$\varepsilon e^{\sqrt{2}t_f}.$$

Thus, even a tiny initial error becomes large when t_f is sufficiently long.
To maintain a fixed final error tolerance τ, the initial error must satisfy roughly

$$\varepsilon \lesssim \tau e^{-\sqrt{2}t_f}.$$

The required initial-costate accuracy therefore becomes exponentially severe with the horizon length.

Key point
Long-horizon simple shooting can require more initial-condition accuracy than floating-point arithmetic can provide.

Overflow Can Occur Without Integration Error

The instability is not caused only by numerical integration.

Even if

$$e^{\boldsymbol{A}t}$$

is evaluated analytically or exactly to machine precision, the unstable mode can exceed the largest representable floating-point number.

Thus, overflow may occur because of:

$$\text{dynamical instability},$$

not because of a low-order integration method.

A numerical integrator adds truncation and roundoff errors, which provide additional disturbances that can excite the unstable mode.

Why Backward Integration Does Not Solve the Problem

Forward integration contains the growing mode

$$e^{+\sqrt{2}t}.$$

One might attempt backward integration from t_f. However, backward time reverses stability:

$$e^{-\sqrt{2}t}$$

becomes the growing mode under backward propagation.

Therefore, the Hamiltonian boundary-value problem is generally not converted into a well-conditioned initial-value problem merely by changing the direction of integration.

Shooting Jacobian Conditioning

For scalar shooting,

$$R(\lambda_0) = x(t_f; \lambda_0) - x_f.$$

Its derivative is

$$\frac{\mathrm{d}R}{\mathrm{d}\lambda_0} = \frac{\partial x(t_f)}{\partial \lambda_0}.$$

This sensitivity is a component of the state-transition matrix:

$$\frac{\partial \boldsymbol{p}(t_f)}{\partial \boldsymbol{p}(0)} = e^{\boldsymbol{A}t_f}.$$

Because this matrix contains exponentially large and small singular directions, the shooting Jacobian can become poorly scaled or ill-conditioned.

This makes Newton corrections sensitive to:

- residual errors;
- Jacobian approximation errors;
- finite precision; and
- the initial guess.

Stable and Unstable Manifold Interpretation

The exact optimal trajectory lies at the intersection of conditions imposed at opposite ends of the interval.

The initial endpoint conditions define a set of allowable initial points. The terminal conditions define a set of trajectories that arrive at the required terminal set.

The correct solution must lie on a special manifold that balances stable and unstable Hamiltonian modes.

A small error moves the trajectory away from this manifold, after which the unstable mode dominates.

Why Multiple Shooting Helps

Multiple shooting divides

$$[0, t_f]$$

into shorter intervals.

Instead of propagating one initial error over the full horizon, each segment propagates errors only over a shorter duration.

If the maximum segment length is Δt, the local amplification behaves like

$$e^{\sqrt{2}\Delta t}$$

rather than

$$e^{\sqrt{2}t_f}.$$

Continuity constraints connect the segment solutions.

The method introduces more variables but improves conditioning.

Why Direct Methods Are Attractive

Direct methods avoid solving the continuous Hamiltonian boundary-value problem by shooting from one endpoint.

Instead, they discretize:

- states;
- controls;

- dynamics;
- path constraints; and
- endpoint conditions.

The resulting nonlinear programming problem can exploit:

- sparse linear algebra;
- defect constraints;
- scaling;
- robust globalization strategies; and
- simultaneous adjustment of the complete trajectory.

This does not eliminate all numerical difficulties, but it often avoids catastrophic initial-value sensitivity.

Practical Diagnostic Workflow

Before implementing simple shooting:

1. derive the coupled state–costate dynamics;
2. substitute the optimal control;
3. linearize around a representative trajectory;
4. inspect eigenvalues or finite-time growth;
5. estimate the amplification across the horizon;
6. inspect scaling and endpoint sensitivity;
7. test a short horizon first; and
8. use multiple shooting or direct methods if the system is strongly unstable.

Common Errors

1. Treating state and costate dynamics as decoupled.
2. Solving the costate first and substituting it afterward without justification.
3. Forgetting that the optimal control depends on the costate.
4. Assuming a linear original plant produces a stable Hamiltonian system.
5. Attributing every shooting failure to the ODE solver.
6. Assuming backward integration removes instability.
7. Ignoring exponential error amplification over long horizons.
8. Using simple shooting without inspecting the spectrum or sensitivities.
9. Confusing machine overflow with an algebraic error in the equations.
10. Simplifying the physical problem instead of selecting a suitable numerical method.

Summary

The main conclusions are:

1. State and costate equations form a coupled optimality system.
2. The costate represents the sensitivity of the cost to state perturbations.
3. In the quadratic example, the optimal control is $u^* = -\lambda^*$.
4. The resulting Hamiltonian dynamics contain eigenvalues $\pm\sqrt{2}$.
5. Hamiltonian spectra are symmetric about the real and imaginary axes.
6. One mode grows in forward time and another grows in backward time.
7. Initial-costate errors can grow exponentially across the horizon.
8. This failure can occur even with analytic propagation and no integration error.
9. Long-horizon simple shooting can exceed finite-precision capability.
10. Multiple shooting and direct methods provide more practical alternatives.

Connection. The final complication for the indirect framework is a constraint that acts continuously along the trajectory, creating active sets, multipliers, and constrained arcs.

Path Constraints, Active Sets, and Augmented Hamiltonians

Real optimal-control problems almost always include restrictions that must hold throughout the trajectory. These are called path constraints.

This section develops:

- general path-constraint formulations;
- conversion of two-sided bounds into standard inequality form;
- active and inactive arcs;
- multiplier functions;
- complementary slackness;
- the augmented Hamiltonian;
- boundary arcs;
- switching structure; and
- why path constraints make indirect methods difficult.

General Path Constraints

A general path constraint may be written as

$$\boldsymbol{c}(\boldsymbol{x}(t), \boldsymbol{u}(t), t) \leq \boldsymbol{0}.$$

The constraint may involve:

- only the state;
- only the control;
- both state and control;
- time explicitly; or
- nonlinear combinations of all of them.

Examples include:

$$\begin{aligned} v(t) &\leq v_{\max}, \\ |u(t)| &\leq u_{\max}, \\ q(\boldsymbol{x}, \boldsymbol{u}, t) &\leq q_{\max}, \\ T(\boldsymbol{x}, \boldsymbol{u}, t) &\leq T_{\max}, \\ g_{\text{load}}(\boldsymbol{x}, \boldsymbol{u}, t) &\leq g_{\max}. \end{aligned}$$

Two-Sided Bounds

Suppose

$$\boldsymbol{c}_{\min} \leq \boldsymbol{c}(\boldsymbol{x}, \boldsymbol{u}, t) \leq \boldsymbol{c}_{\max}.$$

This can be written as two one-sided inequalities:

$$\begin{aligned} \boldsymbol{c}(\boldsymbol{x}, \boldsymbol{u}, t) - \boldsymbol{c}_{\max} &\leq \boldsymbol{0}, \\ -\boldsymbol{c}(\boldsymbol{x}, \boldsymbol{u}, t) + \boldsymbol{c}_{\min} &\leq \boldsymbol{0}. \end{aligned}$$

Stacking them gives

$$\boxed{\boldsymbol{k}(\boldsymbol{x}, \boldsymbol{u}, t) = \begin{bmatrix} \boldsymbol{c} - \boldsymbol{c}_{\max} \\ -\boldsymbol{c} + \boldsymbol{c}_{\min} \end{bmatrix} \leq \boldsymbol{0}.}$$

Thus, one-sided inequality form is sufficient for the theory.

Augmented Cost Functional

Consider a Bolza problem:

$$J = \Phi + \int_{t_0}^{t_f} L(\boldsymbol{x}, \boldsymbol{u}, t)\, \mathrm{d}t$$

subject to

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, t),$$

and

$$\boldsymbol{c}(\boldsymbol{x}, \boldsymbol{u}, t) \leq \boldsymbol{0}.$$

Introduce the multiplier function $\boldsymbol{\mu}(t)$ for the path constraint.
The augmented integrand becomes

$$L + \boldsymbol{\lambda}^{\mathsf{T}}(\boldsymbol{f} - \dot{\boldsymbol{x}}) - \boldsymbol{\mu}^{\mathsf{T}}\boldsymbol{c}.$$

The sign convention used here corresponds to the inequality

$$\boldsymbol{c} \leq \boldsymbol{0}.$$

Augmented Hamiltonian

The ordinary Hamiltonian is

$$H = L + \boldsymbol{\lambda}^{\mathsf{T}}\boldsymbol{f}.$$

The augmented Hamiltonian is

$$\boxed{H_a = H - \boldsymbol{\mu}^{\mathsf{T}}\boldsymbol{c}.}$$

Equivalently,

$$H_a = L + \boldsymbol{\lambda}^\top \boldsymbol{f} - \boldsymbol{\mu}^\top \boldsymbol{c}.$$

All first-order conditions must now be derived from H_a rather than H.

Active and Inactive Constraints

For a scalar path constraint

$$c(\boldsymbol{x}, \boldsymbol{u}, t) \leq 0,$$

there are two possibilities.

Inactive arc

If

$$c(\boldsymbol{x}^*, \boldsymbol{u}^*, t) < 0,$$

then the constraint does not affect the optimal solution locally.
Therefore,

$$\mu^*(t) = 0.$$

Active arc

If

$$c(\boldsymbol{x}^*, \boldsymbol{u}^*, t) = 0,$$

then the trajectory lies on the constraint boundary.
The multiplier may be nonzero:

$$\mu^*(t) \leq 0$$

under the sign convention adopted here.

Complementary Slackness

The active/inactive structure is summarized by

$$c(\boldsymbol{x}^*, \boldsymbol{u}^*, t) \le 0,$$
$$\mu^*(t) \le 0,$$
$$\mu^*(t) c(\boldsymbol{x}^*, \boldsymbol{u}^*, t) = 0.$$

In vector form:

$$\boxed{\mu_i^*(t) c_i(\boldsymbol{x}^*, \boldsymbol{u}^*, t) = 0 \quad \forall i.}$$

This is complementary slackness.

Key point
At each time, either the constraint has slack and its multiplier is zero, or the constraint is active and its multiplier may influence the optimality system.

Typical Active-Set Structure

A path constraint may be inactive, become active, remain active over an interval, and later become inactive.
A typical sequence is

$$\text{interior arc} \longrightarrow \text{boundary arc} \longrightarrow \text{interior arc}.$$

The transition times are unknown and must be solved as part of the problem.

Boundary Arcs

On an active state path constraint,

$$c(\boldsymbol{x}, t) = 0.$$

To remain on the boundary, one must also satisfy consistency conditions such as

$$\frac{\mathrm{d}}{\mathrm{d}t} c(\boldsymbol{x}, t) = 0.$$

If the control does not appear, differentiate again:

$$\frac{\mathrm{d}^2}{\mathrm{d}t^2} c(\boldsymbol{x}, t) = 0.$$

Continue until the control appears. The resulting condition can determine the boundary control.
This resembles singular-control analysis, but the origin is different:

- a singular arc arises from loss of control information in Hamiltonian minimization;
- a boundary arc arises because a state or mixed constraint is active.

Modified Costate Dynamics

The costate equation becomes

$$\dot{\boldsymbol{\lambda}} = -\left(\frac{\partial H_a}{\partial \boldsymbol{x}}\right)^{\mathsf{T}}.$$

Therefore,

$$\boxed{\dot{\boldsymbol{\lambda}} = -\left(\frac{\partial H}{\partial \boldsymbol{x}} - \boldsymbol{\mu}^{\mathsf{T}}\frac{\partial \boldsymbol{c}}{\partial \boldsymbol{x}}\right)^{\mathsf{T}}.}$$

On inactive arcs,

$$\boldsymbol{\mu} = \mathbf{0},$$

so the ordinary costate equation is recovered.
On active arcs, the multiplier modifies the costate dynamics.

Modified Control Optimality Condition

The control must minimize the augmented Hamiltonian:

$$\boldsymbol{u}^* \in \arg\min_{\boldsymbol{u}\in\mathcal{U}} H_a(\boldsymbol{x}^*, \boldsymbol{u}, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*, t).$$

If an interior stationarity condition applies,

$$\boxed{\frac{\partial H_a}{\partial \boldsymbol{u}} = \mathbf{0}.}$$

Thus,

$$\frac{\partial H}{\partial \boldsymbol{u}} - \left(\frac{\partial \boldsymbol{c}}{\partial \boldsymbol{u}}\right)^{\mathsf{T}}\boldsymbol{\mu} = \mathbf{0}.$$

Why Path Constraints Complicate Indirect Methods

Indirect methods require solving:

- state equations;
- costate equations;
- control optimality conditions;
- endpoint conditions;
- multiplier conditions;
- complementary slackness;

- switching times;
- boundary-arc equations; and
- continuity or jump conditions.

The active set is not known beforehand.
Therefore, the analyst may need to guess:

- how many active intervals exist;
- where each one begins;
- where each one ends; and
- which constraints are active on each interval.

A wrong structural guess produces the wrong boundary-value problem.

Constraint Violation During Shooting

In a shooting method, an initial costate guess generates a trajectory.
If the trajectory violates

$$\boldsymbol{c}(\boldsymbol{x}, \boldsymbol{u}, t) \leq \boldsymbol{0},$$

then it is inadmissible.
However, the violation may occur only after integrating for some time. At that point, the method must determine whether to:

- reject the trajectory;
- introduce a boundary arc;
- modify the control;
- add an event time; or
- restart with a different structural assumption.

This makes indirect path-constrained shooting fragile.

Physical Examples

Vehicle motion

$$\begin{aligned} v(t) &\leq v_{\max}, \\ a_{\min} \leq a(t) &\leq a_{\max}, \\ y_{\min}(x) \leq y(t) &\leq y_{\max}(x). \end{aligned}$$

Aerospace systems

$$
\begin{aligned}
q(t) &\le q_{\max}, \\
\dot{Q}(t) &\le \dot{Q}_{\max}, \\
n(t) &\le n_{\max}, \\
\alpha_{\min} \le \alpha(t) &\le \alpha_{\max}.
\end{aligned}
$$

Chemical processes

$$
\begin{aligned}
T(t) &\le T_{\max}, \\
p(t) &\le p_{\max}, \\
0 \le u(t) &\le u_{\max}, \\
c_i(t) &\ge 0.
\end{aligned}
$$

Common Errors

1. Treating every path constraint as active everywhere.
2. Forgetting that inactive constraints have zero multipliers.
3. Using the ordinary Hamiltonian instead of the augmented Hamiltonian.
4. Ignoring complementary slackness.
5. Guessing an active interval without checking admissibility.
6. Confusing singular arcs with state-constraint boundary arcs.
7. Assuming a numerical trajectory is valid even when it briefly violates a path constraint.
8. Ignoring the unknown entry and exit times of active arcs.

Summary

1. Path constraints restrict state, control, or mixed quantities throughout time.
2. Two-sided bounds can be converted to stacked one-sided inequalities.
3. Path constraints introduce multiplier functions.
4. The augmented Hamiltonian is $H_a = H - \boldsymbol{\mu}^{\mathsf{T}} \boldsymbol{c}$.
5. Complementary slackness determines active and inactive behavior.
6. Active constraints create boundary arcs.
7. Unknown active-set structure makes indirect methods difficult.
8. These difficulties motivate direct numerical methods.

Exercises

1. **Finite-horizon Riccati equation**

   Consider

   $$A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \qquad B = \begin{bmatrix} 0 \\ 1 \end{bmatrix},$$

   with

   $$Q = \begin{bmatrix} q_1 & 0 \\ 0 & q_2 \end{bmatrix}, \qquad R = r.$$

   Assume

   $$S(t) = \begin{bmatrix} s_{11} & s_{12} \\ s_{12} & s_{22} \end{bmatrix}.$$

   (a) Derive the matrix Riccati differential equation.

   (b) Derive the three independent scalar Riccati equations.

   (c) State the terminal conditions.

2. **Scalar finite-horizon LQR**

   Consider

   $$\dot{x} = ax + bu,$$

   with

   $$J = \frac{1}{2}x^2(T)S_f + \frac{1}{2}\int_0^T (qx^2 + ru^2)\, dt.$$

   Derive

   (a) the Hamiltonian,

   (b) the optimal control,

   (c) the Riccati equation,

   (d) the terminal condition,

   (e) the optimal feedback law.

3. **Costate decomposition**

   Assume

   $$\lambda = Sx + \beta.$$

   Starting from the state and costate equations,

   (a) derive the Riccati equation,

   (b) derive the differential equation for β,

(c) show that if

$$\beta(T) = 0,$$

then

$$\beta(t) \equiv 0.$$

4. **Infinite-horizon scalar LQR**
   For

$$\dot{x} = ax + bu,$$

   derive the algebraic Riccati equation.
   (a) Solve analytically for both roots.
   (b) Determine which solution gives a stabilizing controller.
   (c) Compute the optimal feedback gain.
5. **Hamiltonian matrix**
   Consider

$$F = \begin{bmatrix} a & -b^2/r \\ -q & -a \end{bmatrix}.$$

   (a) Derive the characteristic polynomial.
   (b) Compute the eigenvalues.
   (c) Determine when they are real or imaginary.
   (d) Explain the implication for closed-loop stability.
6. **Time-varying LQR**
   Assume

$$A(t), \quad B(t), \quad Q(t), \quad R(t)$$

   are time dependent.
   Derive
   (a) the optimal control law,
   (b) the Riccati differential equation,
   (c) the terminal condition,
   (d) the closed-loop state equation.
7. **Hamiltonian conservation**
   Starting from the canonical equations,
   derive

$$\frac{dH^*}{dt} = \frac{\partial H}{\partial t}.$$

Then show that the optimal Hamiltonian is constant whenever the Hamiltonian has no explicit time dependence.

8. **Symmetry of the Riccati equation**

   Assume

$$S_f = S_f^{\mathsf{T}}, \qquad Q = Q^{\mathsf{T}}, \qquad R = R^{\mathsf{T}}.$$

   Show that the Riccati differential equation preserves symmetry by proving that

$$S(t) = S^{\mathsf{T}}(t)$$

   for all t.

9. **Optimal feedback law**

   For the double integrator,

$$\dot{x}_1 = x_2, \qquad \dot{x}_2 = u,$$

   with quadratic performance index,

   (a) derive the optimal feedback law,

   (b) write the closed-loop dynamics,

   (c) express the gain matrix in terms of the Riccati solution,

   (d) explain why the resulting controller is time varying over a finite horizon.

10. **Backward–forward verification**

    Implement the finite-horizon LQR algorithm for any controllable second-order system.

    (a) Integrate the Riccati equation backward.

    (b) Simulate the closed-loop system forward.

    (c) Verify numerically that

$$\dot{\lambda} + Qx + A^{\mathsf{T}}\lambda \approx 0,$$

$$Ru + B^{\mathsf{T}}\lambda \approx 0,$$

    and

$$\lambda(T) - S_f x(T) \approx 0.$$

    Report the maximum residual in each equation.

Chapter 4: Direct Numerical Methods and Collocation

Chapter 4: Direct Numerical Methods and Collocation

From trajectory parameterization to sparse nonlinear-program transcriptions

Indirect methods optimize through the necessary conditions; direct methods instead discretize the original problem. This chapter develops that alternative from the ground up. Direct shooting motivates finite-dimensional parameterization, numerical integration explains how continuous dynamics become discrete equations, and Euler collocation turns those equations into a sparse nonlinear program on a normalized time domain.

Chapter contents

Direct Numerical Methods and Direct Shooting

Indirect methods derive and solve the continuous first-order optimality conditions. Direct methods instead approximate the state and/or control with finite-dimensional representations and transcribe the optimal-control problem into a nonlinear programming problem.

This section introduces:

- indirect versus direct methods;
- state and control parameterization;
- trial or basis functions;
- finite-dimensional transcription;
- direct shooting;
- numerical integration of dynamics and cost;
- the nonlinear programming formulation;
- parameterization error;
- discontinuous controls; and
- method-selection considerations.

Indirect and Direct Methods

Indirect methods

An indirect method:

1. derives first-order necessary conditions;
2. constructs the state–costate boundary-value problem;
3. determines switching, singular, and boundary structures; and
4. solves the resulting system.

Direct methods

A direct method:

1. approximates the state and/or control;
2. discretizes or integrates the dynamics;
3. approximates the objective;
4. enforces constraints numerically; and
5. solves the resulting finite-dimensional nonlinear program.

From Infinite to Finite Dimensions

The original unknowns are functions:

$$\boldsymbol{x}(\cdot), \qquad \boldsymbol{u}(\cdot).$$

This is an infinite-dimensional optimization problem.
A finite-dimensional approximation may be written as

$$\boxed{\boldsymbol{x}(t) \approx \sum_{i=1}^{N_x} \boldsymbol{c}_{x,i}\phi_i(t),}$$

and

$$\boxed{\boldsymbol{u}(t) \approx \sum_{i=1}^{N_u} \boldsymbol{c}_{u,i}\alpha_i(t).}$$

The functions ϕ_i and α_i are known basis or trial functions.
The coefficients are unknown decision variables.

Common Basis Functions

Possible choices include:

- monomials;
- piecewise constants;
- piecewise linear functions;
- splines;
- Lagrange polynomials;
- Legendre polynomials;
- Chebyshev polynomials; and
- local finite-element basis functions.

A global polynomial example is

$$u(t) \approx c_0 + c_1 t + c_2 t^2 + \cdots + c_N t^N.$$

Direct Shooting

Direct shooting parameterizes only the control.
Let

$$\boxed{U(t;\boldsymbol{c}) = \sum_{i=1}^{N} c_i \alpha_i(t).}$$

The approximate control is substituted into

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, U(t;\boldsymbol{c}), t).$$

Given:

- the parameter vector $\boldsymbol{c}$;
- an initial state;
- an initial time; and
- a final time,

the state trajectory is obtained by forward integration.

Unknowns in Direct Shooting

Typical decision variables are

$$z = \begin{bmatrix} \boldsymbol{c} \\ \boldsymbol{x}(t_0) \\ t_0 \\ t_f \end{bmatrix},$$

depending on which quantities are fixed.
The terminal state is not independently parameterized:

$$\boldsymbol{x}(t_f) = \boldsymbol{x}(t_f; z),$$

because it is produced by integration.

Cost Approximation

The original cost is

$$J = \Phi(\boldsymbol{x}(t_0), t_0, \boldsymbol{x}(t_f), t_f) + \int_{t_0}^{t_f} L(\boldsymbol{x}, U, t)\,\mathrm{d}t.$$

After control parameterization and numerical integration,

$$J \approx J_a(z).$$

Thus, the functional becomes an ordinary function of finitely many variables.

Integrating the Cost as an Additional State

Define

$$\dot{x}_{n+1} = L(\boldsymbol{x}, U, t),$$

with

$$x_{n+1}(t_0) = 0.$$

Then

$$x_{n+1}(t_f) = \int_{t_0}^{t_f} L(\boldsymbol{x}, U, t)\,\mathrm{d}t.$$

The augmented integration is

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix}\boldsymbol{x}\\ x_{n+1}\end{bmatrix} = \begin{bmatrix}\boldsymbol{f}(\boldsymbol{x}, U, t)\\ L(\boldsymbol{x}, U, t)\end{bmatrix}.$$

Using the same integration method for dynamics and cost provides consistency.

Endpoint Constraints

Suppose the endpoint constraints are

$$\boldsymbol{\Psi}\left(\boldsymbol{x}(t_0), t_0, \boldsymbol{x}(t_f), t_f\right) = \mathbf{0}.$$

After integration,

$$\boldsymbol{\Psi} = \boldsymbol{g}_e(\boldsymbol{z}).$$

The direct-shooting problem becomes

$$\boxed{\begin{array}{ll}\min\limits_{\boldsymbol{z}} & J_a(\boldsymbol{z})\\ \text{subject to} & \boldsymbol{g}_e(\boldsymbol{z}) = \mathbf{0}.\end{array}}$$

Additional inequalities may be included:

$$\boldsymbol{g}_i(\boldsymbol{z}) \le \mathbf{0}.$$

This is a nonlinear programming problem.

The NLP View

The abstract nonlinear program is

$$\begin{array}{rl}\min\limits_{\boldsymbol{z}} & f(\boldsymbol{z})\\ \text{subject to} & \boldsymbol{g}_e(\boldsymbol{z}) = \mathbf{0},\\ & \boldsymbol{g}_i(\boldsymbol{z}) \le \mathbf{0}.\end{array}$$

For direct shooting:

$$\begin{aligned} f(\boldsymbol{z}) &= J_a(\boldsymbol{z}),\\ \boldsymbol{g}_e(\boldsymbol{z}) &= \text{endpoint residuals},\\ \boldsymbol{g}_i(\boldsymbol{z}) &= \text{parameter and sampled path constraints}.\end{aligned}$$

A Basic Algorithm

1. Choose a control parameterization.
2. Choose an initial guess for the parameters.
3. Construct $U(t; \boldsymbol{c})$.
4. Integrate the dynamics.
5. Integrate the running cost.
6. Evaluate endpoint constraints.
7. Evaluate sampled path constraints.
8. Return objective and constraints to an NLP solver.
9. Update the decision variables.
10. Repeat until convergence.

The Main Strength of Direct Shooting

Direct shooting is attractive because:

- the state equations are satisfied through integration;
- the number of decision variables may be small;
- implementation is straightforward;
- standard ODE solvers can be used;
- standard NLP solvers can be used; and
- it can work well when the control has a simple known structure.

The Main Weakness: Parameterization Error

Direct shooting does not optimize over all admissible controls. It optimizes over the selected approximation space:

$$\mathcal{U}_N = \operatorname{span}\{\alpha_1, \ldots, \alpha_N\}.$$

Therefore, it finds

$$u_N^* = \arg\min_{u \in \mathcal{U}_N} J[u],$$

not necessarily

$$u^* = \arg\min_{u \in \mathcal{U}} J[u].$$

The result can be optimal for the parameterization but poor for the original problem.

Bang–Bang Controls and Smooth Basis Functions

Suppose the true control is

$$u^*(t) = \begin{cases} u_{\min}, & t < t_s, \\ u_{\max}, & t > t_s. \end{cases}$$

This control is discontinuous.
A global polynomial approximation is smooth. Therefore, it cannot represent the switch exactly.
Increasing polynomial degree may introduce:

- oscillation;
- overshoot;
- poor local resolution;
- numerical ill-conditioning; and
- inaccurate switching times.

Structure-Aware Parameterization

If the bang–bang structure is known, a better approximation is

$$U(t) = \begin{cases} c_1, & t < t_s, \\ c_2, & t \geq t_s, \end{cases}$$

with decision variables

$$c_1, \quad c_2, \quad t_s.$$

For a bounded minimum-time problem, one may even impose

$$c_1 = u_{\min}, \qquad c_2 = u_{\max},$$

and optimize only the switching time.
This can recover the exact solution when the assumed structure is correct.

The Structural Knowledge Dilemma

A structure-aware parameterization is powerful only when the solution structure is known.
However, in a difficult problem, the analyst may not know:

- how many switches occur;
- whether singular arcs exist;
- whether path constraints become active;
- whether the control is smooth;
- whether the control is discontinuous; or

- where local features occur.

Thus, direct shooting can be highly effective or highly misleading, depending on the chosen parameterization.

Path Constraints in Direct Shooting

Direct shooting does not automatically enforce path constraints continuously.
One common approximation is to enforce

$$\boldsymbol{c}(\boldsymbol{x}(t_k), U(t_k), t_k) \leq \mathbf{0}$$

at selected sample times t_k.
However, a constraint may be satisfied at the samples and violated between them.
Therefore, verification requires:

- dense post-processing;
- adaptive sampling;
- event detection;
- mesh refinement; or
- a more suitable transcription method.

Sensitivity and Conditioning

The NLP solver needs derivatives such as

$$\frac{\partial J_a}{\partial \boldsymbol{z}}$$

and

$$\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{z}}.$$

These may be computed by:

- finite differences;
- sensitivity equations;
- complex-step differentiation;
- automatic differentiation; or
- adjoint methods.

Long-horizon instability can still make direct shooting poorly conditioned because terminal quantities may be extremely sensitive to the control parameters.

When Direct Shooting Works Well

Direct shooting can work well when:

- the control is low-dimensional;
- the control is smooth or piecewise simple;
- the time horizon is moderate;
- the dynamics are not excessively unstable;
- path constraints are mild;
- a good initial guess is available; and
- the solution structure is known approximately.

Examples may include some launch-vehicle guidance problems with piecewise low-order steering laws.

When Direct Shooting Performs Poorly

It can perform poorly when:

- the control contains many local features;
- switching structure is unknown;
- singular arcs occur;
- strong path constraints are present;
- the horizon is long;
- the state is highly sensitive to parameters;
- the chosen basis is globally smooth but the solution is not; or
- many coefficients are required.

Why More General Direct Methods Are Needed

More advanced methods parameterize both states and controls and enforce dynamics at many points. These include:

- direct multiple shooting;
- direct transcription;
- collocation;
- orthogonal collocation;
- pseudospectral methods; and
- finite-element methods in time.

These methods usually introduce more decision variables but provide:

- better local resolution;
- improved handling of path constraints;

- sparse NLP structure;
- reduced sensitivity to one-way propagation; and
- systematic mesh refinement.

Practical Engineering Perspective

A computational method must do more than return a feasible-looking trajectory.
It should provide evidence that:

- the solution is close to the true optimum;
- constraints are satisfied;
- the result is insensitive to mesh or parameterization changes;
- the objective has converged;
- the control structure is resolved; and
- the numerical result is physically meaningful.

Small improvements can have large economic value in:

- aerospace propellant usage;
- chemical processing;
- refinery energy consumption;
- thermal systems;
- power generation; and
- long-duration industrial operation.

Common Errors

1. Calling the approximate control the exact optimal control.
2. Choosing basis functions without considering solution regularity.
3. Using a global polynomial for a discontinuous control.
4. Ignoring violations between path-constraint sample points.
5. Treating terminal state values as independent of the control parameters.
6. Using an inconsistent integration method for dynamics and cost.
7. Failing to test higher-order or refined parameterizations.
8. Reporting an NLP optimum without verifying the original continuous problem.

Verification Checklist

After convergence:

1. reintegrate with tighter tolerances;
2. sample constraints densely;

3. increase parameterization order;
4. change the basis;
5. perturb the initial guess;
6. compare objective values;
7. inspect control smoothness and switching;
8. check endpoint residuals;
9. compare against necessary conditions when possible; and
10. document convergence behavior.

Summary

1. Direct methods convert optimal control into finite-dimensional optimization.
2. Direct shooting parameterizes only the control.
3. States and cost are obtained through numerical integration.
4. The resulting problem is an NLP.
5. Direct shooting is simple and useful for low-dimensional control structures.
6. Its accuracy is limited by the approximation space.
7. Smooth global functions may poorly represent discontinuous controls.
8. Structure-aware parameterizations can be excellent when the structure is known.
9. Path constraints and long-horizon sensitivities remain challenging.
10. More general direct methods address many of these limitations.

Connection. Direct shooting depends on repeated propagation of the dynamics, making numerical integration the essential bridge to more general direct transcriptions.

Numerical Integration of Ordinary Differential Equations and Collocation

Numerical optimal-control methods depend fundamentally on numerical methods for ordinary differential equations. Before introducing direct transcription and collocation, it is essential to understand how differential equations are approximated, how time-marching schemes work, and why discretization changes the mathematical problem.

This section develops:

- time grids and step sizes;
- one-step time-marching methods;
- forward and backward Euler methods;
- the generalized θ-method;
- explicit and implicit schemes;
- truncation and roundoff errors;
- stability and accuracy;
- classical fourth-order Runge–Kutta;
- the distinction between continuous and discrete solutions;
- simultaneous enforcement of discrete dynamics; and
- the transition from time marching to collocation.

Initial-Value Problems

Consider the ordinary differential equation

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(t, \boldsymbol{x}(t)),$$

with initial condition

$$\boldsymbol{x}(t_0) = \boldsymbol{x}_0.$$

The exact solution is a continuous function

$$\boldsymbol{x} : [t_0, t_f] \to \mathbb{R}^n.$$

A numerical method replaces this continuous solution by approximations at a finite set of points.

Time Discretization

Introduce a sequence of grid points:

$$t_0 < t_1 < t_2 < \cdots < t_N = t_f.$$

Define the step size

$$h_k = t_{k+1} - t_k.$$

The grid need not be uniform. If all steps are equal,

$$h_k = h,$$

the mesh is uniform.
Let

$$\boldsymbol{x}_k \approx \boldsymbol{x}(t_k).$$

A time-marching method computes

$$\boldsymbol{x}_0 \to \boldsymbol{x}_1 \to \boldsymbol{x}_2 \to \cdots \to \boldsymbol{x}_N.$$

Finite-Difference Approximation of the Derivative

A forward finite-difference approximation is

$$\dot{\boldsymbol{x}}(t_k) \approx \frac{\boldsymbol{x}_{k+1} - \boldsymbol{x}_k}{h_k}.$$

Taylor expansion gives

$$\boldsymbol{x}(t_{k+1}) = \boldsymbol{x}(t_k) + h_k \dot{\boldsymbol{x}}(t_k) + \frac{h_k^2}{2} \ddot{\boldsymbol{x}}(\xi_k),$$

for some

$$\xi_k \in (t_k, t_{k+1}).$$

Therefore,

$$\frac{\boldsymbol{x}(t_{k+1}) - \boldsymbol{x}(t_k)}{h_k} = \dot{\boldsymbol{x}}(t_k) + O(h_k).$$

The derivative approximation is first-order accurate.

Forward Euler Method

Substitute the forward difference into the differential equation and evaluate the vector field at the beginning of the interval:

$$\frac{\boldsymbol{x}_{k+1} - \boldsymbol{x}_k}{h_k} = \boldsymbol{f}(t_k, \boldsymbol{x}_k).$$

Rearranging gives

$$\boxed{x_{k+1} = x_k + h_k f(t_k, x_k).}$$

This is the forward Euler method.

Geometric interpretation

The method uses the tangent direction at the beginning of the interval and extrapolates along a straight line.

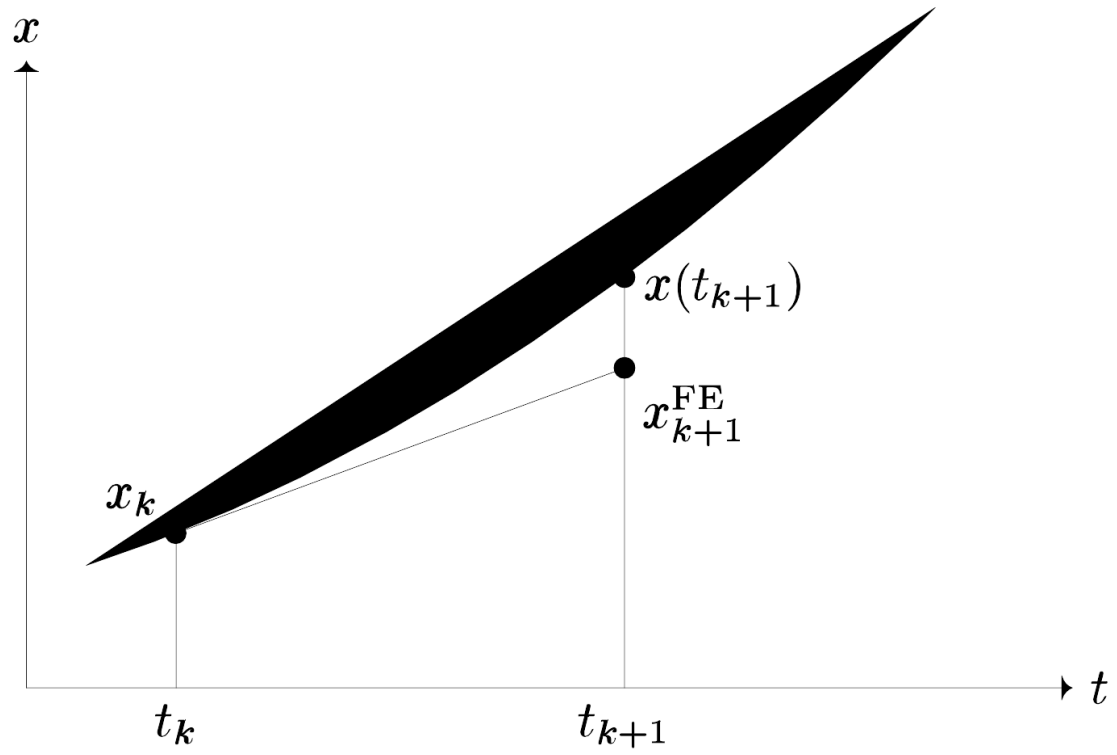

Figure 20: Forward Euler extrapolates from the current state along the initial tangent, producing a local discretization error relative to the exact trajectory.

Advantages

- very simple;
- fully explicit;
- one vector-field evaluation per step;
- easy to implement;
- useful when computational resources are severely limited.

Disadvantages

- only first-order accurate;
- poor stability for many systems;

- requires small step sizes;
- accumulates significant global error;
- performs poorly for stiff systems.

Local and Global Error

The local truncation error is the error made in one step assuming the starting value is exact.
For forward Euler,

$$\tau_{k+1} = O(h_k^2).$$

The global error after many steps is

$$\boldsymbol{e}_k = \boldsymbol{x}(t_k) - \boldsymbol{x}_k.$$

For a sufficiently smooth and stable problem,

$$\boldsymbol{e}_k = O(h).$$

Thus, forward Euler is globally first-order accurate.

Truncation Error and Roundoff Error

Reducing the step size decreases truncation error, but it increases the number of arithmetic operations.
Two competing effects occur:

1. **Large step size:** truncation error dominates.
2. **Extremely small step size:** roundoff and cancellation errors may dominate.

If

$$h_k \boldsymbol{f}(t_k, \boldsymbol{x}_k)$$

is smaller than machine resolution relative to $\boldsymbol{x}_k$, the update may numerically satisfy

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k,$$

even when the true solution is changing.

Key point
A smaller step size is not always better. The step must balance truncation error, roundoff error, stability, and computational cost.

Backward Euler Method

Evaluate the vector field at the end of the interval:

$$\frac{\boldsymbol{x}_{k+1} - \boldsymbol{x}_k}{h_k} = \boldsymbol{f}(t_{k+1}, \boldsymbol{x}_{k+1}).$$

Therefore,

$$\boxed{\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + h_k \boldsymbol{f}(t_{k+1}, \boldsymbol{x}_{k+1}).}$$

This is the backward Euler method.
The method is implicit because $\boldsymbol{x}_{k+1}$ appears on both sides.

Solving the Implicit Equation

Define the residual

$$\boldsymbol{R}(\boldsymbol{x}_{k+1}) = \boldsymbol{x}_{k+1} - \boldsymbol{x}_k - h_k \boldsymbol{f}(t_{k+1}, \boldsymbol{x}_{k+1}).$$

At every step, solve

$$\boldsymbol{R}(\boldsymbol{x}_{k+1}) = \boldsymbol{0}.$$

A fixed-point iteration is

$$\boldsymbol{x}_{k+1}^{(j)} = \boldsymbol{x}_k + h_k \boldsymbol{f}\left(t_{k+1}, \boldsymbol{x}_{k+1}^{(j-1)}\right).$$

A simple initial guess is

$$\boldsymbol{x}_{k+1}^{(0)} = \boldsymbol{x}_k,$$

or a forward Euler prediction.
Newton's method may instead be used:

$$\boldsymbol{x}_{k+1}^{(j+1)} = \boldsymbol{x}_{k+1}^{(j)} - \left[\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{x}_{k+1}}\right]^{-1} \boldsymbol{R}\left(\boldsymbol{x}_{k+1}^{(j)}\right).$$

The Jacobian is

$$\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{x}_{k+1}} = \boldsymbol{I} - h_k \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\left(t_{k+1}, \boldsymbol{x}_{k+1}\right).$$

Explicit and Implicit Schemes

A method is explicit if

$$\boldsymbol{x}_{k+1}$$

can be computed directly from known quantities.
A method is implicit if the new state satisfies a nonlinear algebraic equation.
Explicit methods are typically cheaper per step.
Implicit methods are typically more stable and are often preferred for stiff problems.

The Generalized θ-Method

A unified one-step family is

$$\frac{\boldsymbol{x}_{k+1} - \boldsymbol{x}_k}{h_k} = \theta \boldsymbol{f}(t_k, \boldsymbol{x}_k) + (1-\theta)\boldsymbol{f}(t_{k+1}, \boldsymbol{x}_{k+1}).$$

Equivalently,

$$\boxed{\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + h_k \left[\theta \boldsymbol{f}_k + (1-\theta)\boldsymbol{f}_{k+1}\right].}$$

Special cases are:

$$\begin{aligned} \theta = 1 \quad &\Rightarrow \quad \text{forward Euler,} \\ \theta = 0 \quad &\Rightarrow \quad \text{backward Euler,} \\ \theta = \frac{1}{2} \quad &\Rightarrow \quad \text{trapezoidal or Crank–Nicolson method.} \end{aligned}$$

The trapezoidal scheme is

$$\boxed{\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \frac{h_k}{2}\left[\boldsymbol{f}(t_k, \boldsymbol{x}_k) + \boldsymbol{f}(t_{k+1}, \boldsymbol{x}_{k+1})\right].}$$

Unlike Euler methods, it is second-order accurate.

Stability Analysis Using the Test Equation

Consider

$$\dot{x} = \lambda x,$$

with

$$\mathrm{Re}(\lambda) < 0.$$

The exact solution decays:

$$x(t) = x_0 e^{\lambda t}.$$

A numerical method produces

$$x_{k+1} = R(z)x_k, \qquad z = h\lambda,$$

where $R(z)$ is the stability function.
The method is stable for a given z if

$$|R(z)| \leq 1.$$

Forward Euler

$$R_{\mathrm{FE}}(z) = 1 + z.$$

Stability requires

$$|1 + z| \leq 1.$$

For real negative λ,

$$-2 \leq h\lambda \leq 0.$$

Therefore, the step size is restricted.

Backward Euler

$$R_{\mathrm{BE}}(z) = \frac{1}{1 - z}.$$

Backward Euler is stable for the entire left half-plane:

$$\mathrm{Re}(z) \leq 0.$$

It is therefore A-stable.

Trapezoidal rule

$$R_{\mathrm{TR}}(z) = \frac{1 + z/2}{1 - z/2}.$$

It is also A-stable, but unlike backward Euler, it does not strongly damp very fast modes.

Accuracy Versus Stability

Accuracy and stability are distinct concepts.
A method may be:

- accurate but unstable for a selected step;
- stable but inaccurate;
- both stable and accurate; or
- neither.

Backward Euler and forward Euler are both first-order accurate, but backward Euler has much stronger stability properties.

Multiple-Stage Methods

Instead of estimating the slope at only one point, multiple-stage methods estimate slopes at several locations within a step.

Runge–Kutta methods are one-step, multiple-stage methods.

They do not require values from earlier time steps beyond $\boldsymbol{x}_k$, but they use several vector-field evaluations during the interval

.

Classical Fourth-Order Runge–Kutta Method

Let

$$h = h_k.$$

Define:

$$\begin{aligned}
\boldsymbol{k}_1 &= h\boldsymbol{f}(t_k, \boldsymbol{x}_k), \\
\boldsymbol{k}_2 &= h\boldsymbol{f}\left(t_k + \frac{h}{2}, \boldsymbol{x}_k + \frac{\boldsymbol{k}_1}{2}\right), \\
\boldsymbol{k}_3 &= h\boldsymbol{f}\left(t_k + \frac{h}{2}, \boldsymbol{x}_k + \frac{\boldsymbol{k}_2}{2}\right), \\
\boldsymbol{k}_4 &= h\boldsymbol{f}\left(t_k + h, \boldsymbol{x}_k + \boldsymbol{k}_3\right).
\end{aligned}$$

The update is

$$\boxed{\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \frac{1}{6}\left(\boldsymbol{k}_1 + 2\boldsymbol{k}_2 + 2\boldsymbol{k}_3 + \boldsymbol{k}_4\right).}$$

The method requires four vector-field evaluations per step.

Its global error is

$$\mathcal{O}(h^4).$$

Interpretation of RK4

RK4 computes:

- one slope at the beginning;
- two slopes near the midpoint; and
- one slope at the end.

The weighted average

$$\frac{1}{6}\left(\boldsymbol{k}_1 + 2\boldsymbol{k}_2 + 2\boldsymbol{k}_3 + \boldsymbol{k}_4\right)$$

approximates the integrated effect of the vector field across the interval.

Computational Cost

A higher-order method typically reduces error for a given step size but requires more work per step.
The relevant cost is not simply the number of stages.
One should compare:

$$\text{total computational cost} = \text{cost per step} \times \text{number of steps required}.$$

A fourth-order method may be cheaper overall because it permits much larger steps for the same accuracy.

The Discrete Problem Is Not the Continuous Problem

The continuous problem is

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(t, \boldsymbol{x}).$$

After discretization, the numerical method defines a different problem:

$$\boldsymbol{x}_{k+1} = \boldsymbol{\Psi}\left(t_k, \boldsymbol{x}_k, h_k\right).$$

The step size h_k is a parameter that does not appear in the original differential equation.
For each selected mesh, the discrete problem generally has a different solution.
Therefore, one should say:

> The computation provides a numerical solution of a discrete approximation to the differential equation.

It does not provide the exact continuous solution unless a special exactness property happens to hold.

Consistency, Stability, and Convergence

Three concepts are fundamental.

Consistency

A method is consistent if the local truncation error vanishes as

$$h \to 0.$$

Stability

A method is stable if perturbations do not grow uncontrollably under the discrete propagation.

Convergence

A method is convergent if

$$\boldsymbol{x}_k \to \boldsymbol{x}(t_k)$$

as the mesh is refined.

For standard linear multistep settings, consistency plus zero-stability implies convergence. For one-step methods, analogous stability and consistency results apply.

Sequential Form of Forward Euler

The forward Euler equations are:

$$\begin{aligned} \boldsymbol{x}_1 &= \boldsymbol{x}_0 + h_0 \boldsymbol{f}_0, \\ \boldsymbol{x}_2 &= \boldsymbol{x}_1 + h_1 \boldsymbol{f}_1, \\ &\vdots \\ \boldsymbol{x}_{k+1} &= \boldsymbol{x}_k + h_k \boldsymbol{f}_k. \end{aligned}$$

In time marching, each equation is used after the preceding state has been computed.

Residual Form

Each equation can instead be written as a residual:

$$\boldsymbol{r}_k = \boldsymbol{x}_{k+1} - \boldsymbol{x}_k - h_k \boldsymbol{f}(t_k, \boldsymbol{x}_k) = \boldsymbol{0}.$$

Stack the unknown states:

$$\boldsymbol{y} = \begin{bmatrix} \boldsymbol{x}_0 \\ \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_N \end{bmatrix}.$$

Stack all residuals:

$$\boxed{\boldsymbol{G}(\boldsymbol{y}, \boldsymbol{h}) = \begin{bmatrix} \boldsymbol{r}_0 \\ \boldsymbol{r}_1 \\ \vdots \\ \boldsymbol{r}_{N-1} \end{bmatrix} = \boldsymbol{0}.}$$

This is a system of nonlinear algebraic equations.

Simultaneous Solution

Instead of computing the states sequentially, solve all state values simultaneously:

$$\boldsymbol{G}(\boldsymbol{y}, \boldsymbol{h}) = \boldsymbol{0}.$$

Possible root-finding methods include:

- Newton's method;
- quasi-Newton methods;
- trust-region methods;
- nonlinear least-squares methods; and
- sparse nonlinear equation solvers.

For forward Euler, sequential and simultaneous solutions are mathematically equivalent when solved exactly.
However, the simultaneous viewpoint is the conceptual bridge to direct transcription.

Backward Euler in Residual Form

Backward Euler gives

$$\boldsymbol{r}_k = \boldsymbol{x}_{k+1} - \boldsymbol{x}_k - h_k \boldsymbol{f}(t_{k+1}, \boldsymbol{x}_{k+1}) = \boldsymbol{0}.$$

Stacking all intervals yields

$$\boldsymbol{G}_{\mathrm{BE}}(\boldsymbol{y}, \boldsymbol{h}) = \boldsymbol{0}.$$

The entire trajectory can be solved as one algebraic system.
This avoids performing a separate nonlinear iteration at each time step, although the global nonlinear solver still performs iterations.

Introduction to Collocation

Collocation enforces equality between:

- an approximation of the derivative; and
- the differential-equation vector field

at selected points.

For forward Euler:

$$\boxed{\frac{\boldsymbol{x}_{k+1} - \boldsymbol{x}_k}{h_k} - \boldsymbol{f}(t_k, \boldsymbol{x}_k) = \boldsymbol{0}.}$$

For backward Euler:

$$\boxed{\frac{\boldsymbol{x}_{k+1} - \boldsymbol{x}_k}{h_k} - \boldsymbol{f}(t_{k+1}, \boldsymbol{x}_{k+1}) = \boldsymbol{0}.}$$

These equations force the derivative approximation and vector field to coincide at the selected collocation point.

Defect Constraints

The residual

$$\boldsymbol{\Delta}_k = \boldsymbol{x}_{k+1} - \boldsymbol{x}_k - h_k \boldsymbol{f}(t_k, \boldsymbol{x}_k)$$

is often called a defect.

The discrete dynamics are enforced by

$$\boxed{\boldsymbol{\Delta}_k = \boldsymbol{0}.}$$

In direct optimal control, state values are optimization variables and defects become equality constraints in a nonlinear program.

Time Marching Versus Collocation

Time marching

- computes states sequentially;
- naturally solves initial-value problems;
- propagates errors forward;
- may be sensitive to unstable dynamics.

Collocation

- treats many or all state values as unknowns;
- enforces dynamics simultaneously;
- naturally supports boundary-value problems;
- integrates well with nonlinear programming;
- produces sparse algebraic structures.

Connection to Direct Optimal Control

In a direct transcription method, one may optimize over:

$$\{\boldsymbol{x}_0, \ldots, \boldsymbol{x}_N, \boldsymbol{u}_0, \ldots, \boldsymbol{u}_N\}.$$

The differential equations become algebraic defect constraints:

$$\boldsymbol{\Delta}_k(\boldsymbol{x}_k, \boldsymbol{x}_{k+1}, \boldsymbol{u}_k, \boldsymbol{u}_{k+1}) = \mathbf{0}.$$

The objective is discretized:

$$J \approx J_N.$$

Path constraints are enforced at nodes or collocation points:

$$\boldsymbol{c}(\boldsymbol{x}_k, \boldsymbol{u}_k, t_k) \leq \mathbf{0}.$$

Thus, the continuous optimal-control problem becomes a sparse nonlinear programming problem.

Common Errors

1. Calling forward Euler accurate merely because the step is small.
2. Ignoring roundoff when reducing the step size.
3. Confusing local and global truncation error.
4. Assuming implicit methods are automatically higher order.
5. Assuming stability and accuracy are the same.
6. Comparing methods only by the number of function evaluations.
7. Saying a numerical result is the exact solution of the differential equation.
8. Forgetting that the step size changes the discrete problem.
9. Treating collocation as unrelated to finite differences.
10. Ignoring defect convergence under mesh refinement.

Verification Guidelines

A numerical ODE solution should be checked by:

1. reducing the step size;
2. comparing multiple integration methods;
3. monitoring local error estimates;
4. checking invariants or conserved quantities;
5. comparing with analytic solutions when available;

6. evaluating residuals;
7. inspecting stability restrictions;
8. testing variable-step methods;
9. checking sensitivity to tolerances; and
10. documenting convergence rates.

Summary

The main conclusions are:

1. Numerical integration approximates a continuous ODE on a discrete mesh.
2. Forward Euler is explicit, simple, and first-order accurate.
3. Backward Euler is implicit, first-order accurate, and much more stable.
4. The θ-method unifies forward Euler, backward Euler, and trapezoidal integration.
5. RK4 uses four stages and provides fourth-order global accuracy.
6. Truncation, roundoff, stability, and cost must all be balanced.
7. A discretized equation is a different mathematical problem from the original ODE.
8. Time-marching equations can be rewritten as simultaneous algebraic residuals.
9. Enforcing derivative approximations and vector fields at common points leads to collocation.
10. Collocation is a central foundation of direct numerical optimal-control methods.

Connection. A time-marching formula can be imposed simultaneously rather than sequentially; Euler collocation makes this algebraic viewpoint explicit.

Direct Collocation with Euler Discretization

The purpose of this section is to move from an abstract description of direct methods to a concrete nonlinear programming formulation. The focus is not on software syntax, but on the mathematical structure that must be created before an NLP solver can be used.

The main topics are:

- Euler forward in collocation form;
- simultaneous solution of discrete dynamics;
- root finding for differential-equation residuals;
- transcription of a nonlinear optimal-control problem;
- discretization of time, state, control, and cost;
- treatment of boundary conditions;
- counting variables, constraints, and degrees of freedom; and
- interpretation of the resulting nonlinear program.

Euler Forward: Time-Marching and Collocation Forms

For the ordinary differential equation

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(t, \boldsymbol{x}(t)),$$

Euler forward gives

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + h_k \boldsymbol{f}(t_k, \boldsymbol{x}_k).$$

This is the time-marching form.

The same equation can be written as a residual:

$$\boxed{\boldsymbol{x}_{k+1} - \boldsymbol{x}_k - h_k \boldsymbol{f}(t_k, \boldsymbol{x}_k) = \boldsymbol{0}.}$$

This is the collocation or defect form.

Mesh and Discrete State Vector

Let

$$t_0 < t_1 < \cdots < t_K = t_f,$$

with

$$h_k = t_{k+1} - t_k.$$

Define

$$\boldsymbol{x}_k \approx \boldsymbol{x}(t_k).$$

Collect all state approximations into

$$\boldsymbol{y} = \begin{bmatrix} \boldsymbol{x}_0 \\ \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_K \end{bmatrix}.$$

The Euler residuals form a nonlinear system:

$$\boldsymbol{G}(\boldsymbol{y}) = \boldsymbol{0}.$$

Initial Conditions as Algebraic Constraints

Suppose

$$\boldsymbol{x}(t_0) = \boldsymbol{a},$$

where $\boldsymbol{a}$ is known.
One may either substitute

$$\boldsymbol{x}_0 = \boldsymbol{a}$$

directly or add the residual equation

$$\boxed{\boldsymbol{x}_0 - \boldsymbol{a} = \boldsymbol{0}.}$$

The second form is common in collocation because it keeps all conditions in one residual system.

Newton's Method for the Global Residual System

To solve

$$\boldsymbol{G}(\boldsymbol{y}) = \boldsymbol{0},$$

consider

$$\boldsymbol{G}(\boldsymbol{y} + \Delta\boldsymbol{y}) \approx \boldsymbol{G}(\boldsymbol{y}) + \frac{\partial \boldsymbol{G}}{\partial \boldsymbol{y}} \Delta\boldsymbol{y}.$$

Setting this approximation equal to zero gives

$$\frac{\partial \boldsymbol{G}}{\partial \boldsymbol{y}} \Delta\boldsymbol{y} = -\boldsymbol{G}(\boldsymbol{y}).$$

Hence,

$$\Delta \boldsymbol{y} = -\left(\frac{\partial \boldsymbol{G}}{\partial \boldsymbol{y}}\right)^{-1} \boldsymbol{G}(\boldsymbol{y}).$$

The Newton iteration is

$$\boxed{\boldsymbol{y}^{(j+1)} = \boldsymbol{y}^{(j)} - \left[\frac{\partial \boldsymbol{G}}{\partial \boldsymbol{y}}\left(\boldsymbol{y}^{(j)}\right)\right]^{-1} \boldsymbol{G}\left(\boldsymbol{y}^{(j)}\right).}$$

In implementation, one solves the linear system rather than explicitly forming the inverse.

A Nonlinear Optimal-Control Example

Consider

$$\boxed{\min_{x(\cdot),u(\cdot)} J = \frac{1}{2}\int_0^{t_f} \left(x^2(t) + u^2(t)\right)\, \mathrm{d}t}$$

subject to

$$\dot{x}(t) = -x^3(t) + u(t),$$

with

$$x(0) = a, \qquad x(t_f) = b,$$

and fixed t_f.
This problem is nonlinear because of the term $-x^3$.

Uniform Time Mesh

Choose K intervals:

$$h = \frac{t_f - t_0}{K}.$$

The mesh points are

$$t_k = t_0 + kh, \qquad k = 0, \ldots, K.$$

There are:

- K intervals;
- $K + 1$ mesh points.

State and Control Discretization

For Euler forward, the state is defined at all mesh points:

$$x_0, x_1, \ldots, x_K.$$

The control is required only at the beginning of each interval:

$$u_0, u_1, \ldots, u_{K-1}.$$

There is no u_K in this transcription because the final node does not begin another interval.

> **Key point**
> Euler forward uses the state and control at the left endpoint of each interval. Therefore, there are $K+1$ state values but only K control values.

Discrete Dynamics

Euler forward gives

$$x_{k+1} = x_k + h\left(-x_k^3 + u_k\right).$$

The defect equation is

$$\boxed{\Delta_k = x_{k+1} - x_k - h\left(-x_k^3 + u_k\right) = 0,}$$

for

$$k = 0, \ldots, K-1.$$

Equivalently,

$$\Delta_k = x_{k+1} - x_k + hx_k^3 - hu_k = 0.$$

Discretization of the Cost

The running cost is

$$L(x, u) = \frac{1}{2}(x^2 + u^2).$$

Using a left-rectangle rule consistent with Euler forward,

$$\int_0^{t_f} (x^2 + u^2)\,\mathrm{d}t \approx h \sum_{k=0}^{K-1} (x_k^2 + u_k^2).$$

Therefore,

$$\boxed{J_a = \frac{h}{2} \sum_{k=0}^{K-1} \left(x_k^2 + u_k^2\right).}$$

The original functional has become an ordinary function of finitely many variables.

Boundary Conditions

The continuous boundary conditions map to

$$\begin{aligned} x_0 - a &= 0, \\ x_K - b &= 0. \end{aligned}$$

These are equality constraints in the NLP.

Decision Vector

Define the decision vector

$$\boxed{z = \begin{bmatrix} x_0 & x_1 & \cdots & x_K & u_0 & u_1 & \cdots & u_{K-1} \end{bmatrix}^\mathsf{T}.}$$

The total number of variables is

$$(K + 1) + K = 2K + 1.$$

Complete NLP Formulation

The Euler direct-collocation problem is

$$\boxed{\begin{aligned} \min_{z} \quad & \frac{h}{2} \sum_{k=0}^{K-1} \left(x_k^2 + u_k^2\right) \\ \text{subject to} \quad & x_{k+1} - x_k - h(-x_k^3 + u_k) = 0, \\ & k = 0, \ldots, K-1, \\ & x_0 - a = 0, \\ & x_K - b = 0. \end{aligned}}$$

This is a nonlinear programming problem.

Variables, Constraints, and Degrees of Freedom

The number of variables is

$$n_z = 2K + 1.$$

The number of equality constraints is

$$n_c = K + 2.$$

Therefore,

$$\boxed{n_{\text{dof}} = n_z - n_c = K - 1.}$$

The problem must have degrees of freedom because the optimizer must choose among many feasible trajectories.

Why the System Is Underdetermined Before Optimization

The defect equations and boundary conditions alone do not uniquely determine all state and control values.
There are many controls that may approximately transfer the state from a to b.
The objective selects the one that minimizes

$$\frac{1}{2}\int_0^{t_f} (x^2 + u^2)\,\mathrm{d}t.$$

Thus:

- the constraints define feasibility;
- the objective selects optimality.

Generalization to Multiple States and Controls

For

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(t, \boldsymbol{x}, \boldsymbol{u}),$$

with

$$\boldsymbol{x} \in \mathbb{R}^{n_x}, \qquad \boldsymbol{u} \in \mathbb{R}^{n_u},$$

Euler forward gives

$$\boldsymbol{x}_{k+1} - \boldsymbol{x}_k - h_k \boldsymbol{f}(t_k, \boldsymbol{x}_k, \boldsymbol{u}_k) = \boldsymbol{0}.$$

The number of variables becomes

$$(K+1)n_x + Kn_u.$$

The number of dynamic constraints becomes

$$Kn_x.$$

Endpoint and path constraints are added separately.

Forward Versus Backward Euler Indexing

Euler forward uses

$$\boldsymbol{f}(t_k, \boldsymbol{x}_k, \boldsymbol{u}_k).$$

Backward Euler uses

$$\boldsymbol{f}(t_{k+1}, \boldsymbol{x}_{k+1}, \boldsymbol{u}_{k+1}).$$

Hence:

- forward Euler naturally uses control values $u_0, \ldots, u_{K-1}$;
- backward Euler naturally uses control values $u_1, \ldots, u_K$.

The quadrature indexing should be consistent with the dynamic discretization.

Accuracy of the Transcription

Euler forward is first-order accurate.

Therefore, the discrete optimal-control solution is expected to converge at first order under appropriate regularity and stability assumptions.

A valid numerical study should compare solutions for:

$$K, \quad 2K, \quad 4K, \ldots$$

and monitor:

- objective convergence;
- state convergence;
- control convergence;
- defect residuals;
- endpoint residuals.

Implementation Workflow

1. Select K.

2. Compute h.
3. Create an initial guess for all states and controls.
4. Build the objective function.
5. Build the Euler defect constraints.
6. Add endpoint constraints.
7. Supply bounds if needed.
8. Call an NLP solver.
9. Reconstruct the trajectories.
10. Refine the mesh and repeat.

Common Errors

1. Using K state points instead of $K + 1$.
2. Including u_K in an Euler-forward transcription without a reason.
3. Summing the quadrature from 0 to K instead of 0 to $K - 1$.
4. Forgetting the endpoint constraints.
5. Counting variables or constraints incorrectly.
6. Solving the defect equations without including the objective.
7. Calling the discrete solution exact.
8. Failing to refine the mesh.

Summary

1. Euler forward can be written as collocation defects.
2. The state and control values become NLP decision variables.
3. The running cost becomes a finite quadrature sum.
4. Endpoint conditions become algebraic constraints.
5. The resulting finite-dimensional problem is an NLP.
6. The example contains $2K + 1$ variables and $K + 2$ equality constraints.
7. The remaining $K - 1$ degrees of freedom are optimized through the cost.
8. Mesh refinement is required to assess accuracy.

Connection. Once collocation has produced an NLP, its usefulness depends on exploiting local dependence, sparse derivatives, and a fixed normalized time interval.

Sparse NLP Structure and Time Normalization

Direct collocation produces nonlinear programs that are often large but sparse. This structure is one of the main reasons collocation methods are computationally practical.

This section develops:

- local dependence of collocation constraints;
- constraint-Jacobian sparsity;
- banded structure;
- nonzero counting;
- sparsity scaling with mesh refinement;
- implications for NLP solvers;
- time normalization to a fixed interval;
- transformation of dynamics and cost;
- free initial and final times; and
- extensions to large and multi-state systems.

Review of the Euler NLP

Consider the decision vector

$$z = \begin{bmatrix} x_0 & x_1 & \cdots & x_K & u_0 & u_1 & \cdots & u_{K-1} \end{bmatrix}^{\mathsf{T}}.$$

The defect equations are

$$c_k = x_{k+1} - x_k - h(-x_k^3 + u_k) = 0,$$

for

$$k = 0, \ldots, K-1.$$

The endpoint constraints are

$$b_1 = x_0 - a = 0,$$
$$b_2 = x_K - b = 0.$$

Local Dependence

Each defect c_k depends only on:

$$x_k, \qquad x_{k+1}, \qquad u_k.$$

It does not depend on distant variables such as:

$$x_{k+4}, \quad u_{k+6}, \quad \text{etc.}$$

This local dependence creates sparsity.

Constraint Jacobian

Define

$$g(z) = \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_{K-1} \\ b_1 \\ b_2 \end{bmatrix}.$$

The constraint Jacobian is

$$\boxed{J_c = \frac{\partial g}{\partial z}.}$$

Its dimensions are

$$(K+2) \times (2K+1).$$

Derivatives of One Defect

For

$$c_k = x_{k+1} - x_k + h x_k^3 - h u_k,$$

the nonzero derivatives are

$$\begin{aligned} \frac{\partial c_k}{\partial x_k} &= -1 + 3h x_k^2, \\ \frac{\partial c_k}{\partial x_{k+1}} &= 1, \\ \frac{\partial c_k}{\partial u_k} &= -h. \end{aligned}$$

All other partial derivatives of c_k are zero.

Sparsity Pattern

The structure is schematically:

	x_0	x_1	x_2	$\cdots$	x_{K-1}	x_K	u_0	u_1	u_2	$\cdots$	u_{K-1}
c_0	×	×					×				
c_1		×	×					×			
c_2			×	×					×		
$\vdots$				$\ddots$	$\ddots$				$\ddots$		
c_{K-1}					×	×				×	
b_1	×										
b_2						×					

Only a narrow band of entries is nonzero.

Counting Nonzero Entries

Each defect contributes three nonzeros.
There are K defects, so:

$$3K$$

nonzeros arise from the dynamics.
The two endpoint constraints contribute two more.
Therefore,

$$\boxed{\text{nnz}(\boldsymbol{J}_c) = 3K + 2.}$$

Jacobian Density

The total number of entries is

$$(K + 2)(2K + 1).$$

The density is

$$\rho_K = \frac{3K + 2}{(K + 2)(2K + 1)}.$$

As

$$K \to \infty,$$

$$\rho_K \sim \frac{3}{2K} \to 0.$$

Thus, the problem becomes increasingly sparse as the mesh is refined.

Key point
The NLP grows larger with mesh refinement, but its derivative matrices become proportionally sparser.

Why Sparse Structure Matters

A dense treatment stores and processes all entries.
A sparse treatment stores and processes only nonzero entries.
This affects:

- memory usage;
- Jacobian assembly;
- linear-system solution;
- factorization cost;
- scalability;
- parallelization.

For large trajectory-optimization problems, sparse linear algebra is essential.

Sparse NLP Solvers

Modern sparse NLP solvers exploit derivative structure.
Examples include:

- IPOPT;
- SNOPT;
- KNITRO;
- WORHP;
- filterSQP-type solvers;
- sparse interior-point and SQP methods.

MATLAB's `fmincon` may be useful for small instructional problems, but large collocation problems benefit from solvers designed explicitly for sparse NLPs.

Why Collocation Naturally Produces Sparsity

A collocation equation on interval k depends only on variables in or near that interval.
Typical local dependencies include:

$$(\boldsymbol{x}_k, \boldsymbol{x}_{k+1}, \boldsymbol{u}_k, \boldsymbol{u}_{k+1}).$$

Therefore, distant nodes do not interact directly in the Jacobian.
This remains true for many methods:

- Euler forward;

- Euler backward;
- trapezoidal collocation;
- Hermite–Simpson;
- Runge–Kutta collocation;
- finite-element transcription;
- multiple shooting.

Multiple States and Controls

For

$$\boldsymbol{x} \in \mathbb{R}^{n_x}, \qquad \boldsymbol{u} \in \mathbb{R}^{n_u},$$

each defect block has dimension n_x.
A defect typically depends on:

- n_x components of $\boldsymbol{x}_k$;
- n_x components of $\boldsymbol{x}_{k+1}$;
- n_u components of $\boldsymbol{u}_k$.

The Jacobian becomes block sparse and block banded.

Time Normalization

Instead of discretizing directly on

$$t \in [t_0, t_f],$$

it is convenient to map the domain to

$$\tau \in [-1, 1].$$

Define

$$\boxed{\tau = 2\frac{t - t_0}{t_f - t_0} - 1.}$$

The inverse transformation is

$$\boxed{t(\tau) = \frac{t_f - t_0}{2}\tau + \frac{t_f + t_0}{2}.}$$

Derivation of the Mapping

Starting from

$$t_0 \le t \le t_f,$$

subtract t_0:

$$0 \le t - t_0 \le t_f - t_0.$$

Divide by $t_f - t_0$:

$$0 \le \frac{t - t_0}{t_f - t_0} \le 1.$$

Multiply by 2 and subtract 1:

$$-1 \le 2\frac{t - t_0}{t_f - t_0} - 1 \le 1.$$

Transformation of the Dynamics

The original dynamics are

$$\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t} = \boldsymbol{f}(t, \boldsymbol{x}, \boldsymbol{u}).$$

By the chain rule,

$$\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}\tau} = \frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t}\frac{\mathrm{d}t}{\mathrm{d}\tau}.$$

From the inverse map,

$$\frac{\mathrm{d}t}{\mathrm{d}\tau} = \frac{t_f - t_0}{2}.$$

Therefore,

$$\boxed{\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}\tau} = \frac{t_f - t_0}{2}\boldsymbol{f}\left(t(\tau), \boldsymbol{x}(\tau), \boldsymbol{u}(\tau)\right).}$$

Transformation of the Cost

For

$$J = \Phi + \int_{t_0}^{t_f} L(t, \boldsymbol{x}, \boldsymbol{u})\,\mathrm{d}t,$$

use

$$\mathrm{d}t = \frac{t_f - t_0}{2}\,\mathrm{d}\tau.$$

Then

$$\boxed{J = \Phi + \frac{t_f - t_0}{2}\int_{-1}^{1} L\left(t(\tau), \boldsymbol{x}(\tau), \boldsymbol{u}(\tau)\right)\,\mathrm{d}\tau.}$$

Free Initial and Final Times

If t_0 or t_f is free, they become NLP decision variables.
The normalized computational domain remains fixed:

$$\tau \in [-1, 1].$$

However, the duration

$$t_f - t_0$$

appears explicitly in:

- the transformed dynamics;
- the transformed cost;
- time-dependent path constraints;
- endpoint conditions.

Why Normalize Time

Time normalization provides:

- a fixed computational domain;
- simpler mesh generation;
- easier handling of free final time;
- direct compatibility with orthogonal polynomials;
- a natural foundation for pseudospectral methods;
- improved software modularity.

Normalized Euler Defects

Let

$$-1 = \tau_0 < \tau_1 < \cdots < \tau_K = 1.$$

Define

$$\Delta\tau_k = \tau_{k+1} - \tau_k.$$

Euler forward in normalized time gives

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \Delta\tau_k \frac{t_f - t_0}{2} \boldsymbol{f}(t_k, \boldsymbol{x}_k, \boldsymbol{u}_k).$$

Thus,

$$\boxed{\boldsymbol{x}_{k+1} - \boldsymbol{x}_k - \Delta\tau_k \frac{t_f - t_0}{2} \boldsymbol{f}(t_k, \boldsymbol{x}_k, \boldsymbol{u}_k) = \boldsymbol{0}.}$$

Normalized Quadrature

A left-rectangle approximation becomes

$$J_a = \Phi + \frac{t_f - t_0}{2} \sum_{k=0}^{K-1} \Delta\tau_k L(t_k, \boldsymbol{x}_k, \boldsymbol{u}_k).$$

For a uniform normalized mesh,

$$\Delta\tau = \frac{2}{K}.$$

Sparsity with Free Final Time

When t_f is a decision variable, every defect depends on t_f through the factor

$$\frac{t_f - t_0}{2}.$$

This introduces one additional dense column in the Jacobian.
However, the rest of the Jacobian remains sparse.
Thus, free-time problems retain strong sparse structure.

Finite Versus Infinite Horizons

The affine map above applies to finite intervals.
An infinite horizon,

$$t \in [t_0, \infty),$$

requires a nonlinear transformation, for example:

$$t = t_0 + \frac{1+\tau}{1-\tau}, \qquad \tau \in [-1, 1).$$

Such mappings require special treatment because:

- the derivative scaling becomes singular near the mapped endpoint;
- quadrature weights may become extreme;
- asymptotic behavior must be represented accurately;
- convergence analysis is more delicate.

Solver-Oriented Implementation Workflow

1. Normalize time.
2. Select the mesh.
3. Define the decision vector.
4. Define the discretized objective.
5. Define defect constraints.
6. Define endpoint and path constraints.
7. Provide sparse Jacobian information.
8. Provide variable and constraint bounds.
9. Solve the NLP.
10. Verify convergence under mesh refinement.

Derivative Information

NLP solvers may use:

- finite differences;
- analytic derivatives;
- automatic differentiation;
- complex-step derivatives.

For large collocation problems, exact or automatic sparse derivatives are usually preferable.

Common Errors

1. Treating the Jacobian as dense.
2. Counting $2K - 1$ variables instead of $2K + 1$ in the scalar Euler example.
3. Forgetting endpoint-constraint nonzeros.
4. Assuming sparsity disappears for nonlinear dynamics.
5. Omitting the factor $(t_f - t_0)/2$ after time normalization.
6. Transforming the dynamics but not the cost.

7. Forgetting that $t(\tau)$ depends on both endpoint times.
8. Using a finite-interval affine map for an infinite-horizon problem.
9. Supplying inconsistent derivative sparsity to the solver.
10. Failing to verify mesh convergence.

Summary

1. Collocation constraints depend locally on neighboring variables.
2. This produces sparse, banded Jacobians.
3. The scalar Euler example has $3K + 2$ Jacobian nonzeros.
4. Jacobian density approaches zero as the mesh is refined.
5. Sparse NLP solvers exploit this structure.
6. Time normalization maps arbitrary finite intervals to $[-1, 1]$.
7. The dynamics scale by $(t_f - t_0)/2$.
8. The cost scales by the same factor.
9. Free endpoint times become NLP decision variables.
10. Normalization is foundational for advanced collocation and pseudospectral methods.

Exercises

1. **Direct shooting with a polynomial control parameterization**
   Consider

$$\dot{x}(t) = -x(t) + u(t), \qquad x(0) = x_0,$$

   with fixed final time T, terminal constraint

$$x(T) = x_f,$$

   and performance index

$$J = \frac{1}{2}\int_0^T \left(x^2(t) + \rho u^2(t)\right)\,\mathrm{d}t.$$

   Approximate the control by

$$u(t) = c_0 + c_1 t + c_2 t^2.$$

   (a) Identify the finite-dimensional decision vector.
   (b) Write the state obtained by integrating the dynamics for a given coefficient vector.
   (c) Express the terminal constraint as a nonlinear algebraic constraint on c_0, c_1, c_2.
   (d) Introduce an accumulated-cost state and write the complete direct-shooting nonlinear program.
2. **Forward and backward Euler stability**
   Apply forward Euler and backward Euler to

$$\dot{x} = -5x, \qquad x(0) = 1.$$

   (a) Derive the update equation and amplification factor for each method.
   (b) Derive the forward-Euler stability restriction on h.
   (c) Show that backward Euler is stable for every $h > 0$.
   (d) For a uniform mesh, derive explicit expressions for x_k^{FE} and x_k^{BE}.
   (e) Compare both discrete solutions with

$$x(t_k) = e^{-5t_k}.$$

3. **Generalized θ-method**
   Consider the convention

$$x_{k+1} = x_k + h\left[\theta f(t_k, x_k) + (1-\theta) f(t_{k+1}, x_{k+1})\right].$$

   (a) Identify the values of θ corresponding to forward Euler, backward Euler, and the trapezoidal method.

(b) Determine which value of θ produces an explicit method for a general nonlinear vector field.

(c) For the test equation

$$\dot{x} = \lambda x, \qquad z = h\lambda,$$

derive the stability function

$$R_\theta(z).$$

(d) Recover the stability functions of the three special cases.

(e) Compare the limits of the backward-Euler and trapezoidal stability functions as $z \to -\infty$.

4. **Runge–Kutta integration of controlled dynamics**

Consider one mesh interval for

$$\dot{x} = -x^3 + u,$$

where the control is constant on the interval:

$$u(t) = u_k, \qquad t \in [t_k, t_{k+1}].$$

(a) Write the four classical RK4 stage equations.

(b) Derive the RK4 update for x_{k+1}.

(c) State the number of vector-field evaluations required per interval.

(d) Explain why RK4 is a one-step method despite using four stages.

(e) State its expected global order of accuracy.

5. **Euler direct-collocation transcription**

Transcribe

$$\min_{x(\cdot),u(\cdot)} J = \frac{1}{2}\int_0^T \left(x^2 + u^2\right)\,\mathrm{d}t$$

subject to

$$\dot{x} = -x^3 + u, \qquad x(0) = a, \qquad x(T) = b,$$

using forward-Euler collocation on K uniform intervals.

(a) Define the mesh and step size.

(b) Define the complete NLP decision vector.

(c) Derive all Euler defect constraints.

(d) Derive the left-rectangle discrete objective.

(e) Write the two endpoint constraints.

(f) Determine the number of decision variables, equality constraints, and remaining degrees of freedom.

6. **Global residual equations and their Jacobian**

   For the scalar nonlinear initial-value problem

   $$\dot{x} = f(t, x), \qquad x(t_0) = x_0,$$

   on a mesh with K intervals:

   (a) Write all forward-Euler equations as one simultaneous algebraic residual system.

   (b) Derive the nonzero Jacobian entries of a generic forward-Euler residual.

   (c) Repeat parts (a) and (b) for backward Euler.

   (d) Explain mathematically why the backward-Euler residual is implicit.

   (e) State how Newton's method would be applied to the global backward-Euler residual system.

7. **Sparse Jacobian structure**

   For the scalar Euler-collocation problem in Exercise 5, use

   $$z = \begin{bmatrix} x_0 & \cdots & x_K & u_0 & \cdots & u_{K-1} \end{bmatrix}^{\mathsf{T}}.$$

   (a) Derive the three nonzero partial derivatives in one defect row.

   (b) Show that the complete equality-constraint Jacobian has dimensions

   $$(K + 2) \times (2K + 1).$$

   (c) Show that it has $3K + 2$ nonzero entries.

   (d) Derive its exact density.

   (e) Compute the density for $K = 10$.

   (f) Prove that the density is asymptotically

   $$O\left(\frac{1}{K}\right).$$

   (g) Extend the variable count, constraint count, and maximum local nonzero count to n_x states and n_u controls.

8. **Time normalization and free final time**

   Map

   $$t \in [t_0, t_f]$$

   to

   $$\tau \in [-1, 1].$$

   (a) Derive the forward and inverse affine transformations.

   (b) Derive $\mathrm{d}t/\mathrm{d}\tau$.

(c) Transform

$$\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t} = \boldsymbol{f}(t, \boldsymbol{x}, \boldsymbol{u}).$$

(d) Transform the Bolza objective

$$J = \Phi + \int_{t_0}^{t_f} L(t, \boldsymbol{x}, \boldsymbol{u})\, \mathrm{d}t.$$

(e) On a normalized mesh, derive the forward-Euler defect constraints.

(f) When t_f is free, identify every part of the NLP in which it appears.

(g) Explain why the t_f column is globally coupled while the remaining defect Jacobian is still sparse.

9. **Piecewise-constant shooting with an unknown switching time**

Consider

$$\dot{x} = u, \qquad x(0) = 0, \qquad x(T) = x_f,$$

and parameterize the control as

$$u(t) = \begin{cases} u_1, & 0 \le t < t_s, \\ u_2, & t_s \le t \le T, \end{cases}$$

where u_1, u_2, and t_s are decision variables.

(a) Integrate the dynamics analytically on both intervals.

(b) Derive the continuity condition at t_s.

(c) Express the terminal constraint as an algebraic equation in u_1, u_2, t_s.

(d) For

$$J = \frac{1}{2} \int_0^T u^2(t)\, \mathrm{d}t,$$

derive the finite-dimensional objective.

(e) Write the resulting direct-shooting NLP, including bounds

$$u_{\min} \le u_1, u_2 \le u_{\max}, \qquad 0 \le t_s \le T.$$

10. **Analytical mesh-refinement study**

Apply forward Euler to

$$\dot{x} = -x, \qquad x(0) = 1,$$

on $[0, T]$ using K uniform intervals.

(a) Show that

$$x_K = \left(1 - \frac{T}{K}\right)^K.$$

(b) Show that

$$\lim_{K\to\infty} x_K = e^{-T}.$$

(c) Use a logarithmic expansion to obtain the leading-order terminal error.

(d) Show that the terminal error is first order in $h = T/K$.

(e) Derive the error ratio expected when the mesh size is doubled.

(f) State how this result can be used to verify a direct-collocation implementation.

Chapter 5: Orthogonal Collocation and Pseudospectral Foundations

Chapter 5: Orthogonal Collocation and Pseudospectral Foundations

From Gaussian quadrature to Legendre pseudospectral transcription

Low-order collocation is local and transparent, but smooth trajectories can be represented far more efficiently with global polynomials. This chapter builds the required approximation theory in layers: orthogonal collocation, Legendre polynomials, Lagrange interpolation, Gaussian quadrature, and differentiation matrices. Those ingredients then lead naturally to the LGL, LG, and LGR pseudospectral schemes.

Chapter contents

Orthogonal Collocation and Gaussian Quadrature

This section introduces orthogonal collocation, polynomial state approximations, Gaussian quadrature, and Legendre–Gauss (LG), Legendre–Gauss–Radau (LGR), and Legendre–Gauss–Lobatto (LGL) collocation points.

From Euler Collocation to General Collocation

Euler collocation enforces

$$\frac{x_{k+1} - x_k}{h_k} = f(x_k, u_k, t_k),$$

or equivalently

$$x_{k+1} - x_k - h_k f(x_k, u_k, t_k) = 0.$$

The left-hand side approximates the derivative while the right-hand side evaluates the dynamics at the same point. This is the fundamental idea of collocation.

Polynomial Approximation

Over each mesh interval,

$$x^{(k)}(t) \approx \sum_{j=0}^{m} c_j^{(k)} s_j^{(k)}(t),$$

where $s_j^{(k)}(t)$ are basis polynomials.

Differentiating,

$$\dot{x}^{(k)}(t) \approx \sum_{j=0}^{m} c_j^{(k)} \dot{s}_j^{(k)}(t).$$

The derivative polynomial has degree $m - 1$.

Collocation Conditions

Choose m collocation points inside each mesh interval,

$$\tau_1, \tau_2, \ldots, \tau_m.$$

At every collocation point,

$$\dot{x}^{(k)}(\tau_i) = f\left(x^{(k)}(\tau_i), u^{(k)}(\tau_i), \tau_i\right), \qquad i = 1, \ldots, m.$$

Because the derivative polynomial has degree $m - 1$, these provide m equations. One additional condition is required.

Continuity Condition

The missing equation is supplied by continuity:

$$x^{(k)}(t_k) = x^{(k+1)}(t_k).$$

For the first interval,

$$x(t_0) = x_0.$$

Thus every interval has exactly the correct number of equations.

Euler as a Special Case

Euler forward corresponds to:

- a linear state approximation,
- a constant derivative approximation,
- one collocation point per interval,
- one continuity condition.

Quadrature

The cost integral

$$J = \int_{-1}^{1} f(\tau)\, d\tau$$

is approximated by

$$J \approx \sum_{i=1}^{N} w_i f(\tau_i),$$

where w_i are quadrature weights satisfying

$$\sum_i w_i = 2.$$

Choosing Optimal Points

Rather than equally spaced points, Gaussian quadrature chooses both the locations τ_i and weights w_i to minimize integration error.

LGL Points

Endpoints are included:

$$\tau_1 = -1, \qquad \tau_N = 1.$$

Interior points are roots of

$$\frac{d}{d\tau} P_{N-1}(\tau).$$

LGL exactly integrates polynomials of degree

$$2N - 3.$$

LGR Points

Only the left endpoint is fixed. The points are roots of

$$P_N(\tau) + P_{N-1}(\tau).$$

LGR exactly integrates polynomials of degree

$$2N - 2.$$

LG Points

Neither endpoint is fixed. The points are roots of

$$P_N(\tau).$$

LG exactly integrates polynomials of degree

$$2N - 1.$$

Accuracy Comparison

Method	Endpoints Included	Exact Polynomial Degree
LGL	Both	$2N - 3$
LGR	One	$2N - 2$
LG	None	$2N - 1$

Relaxing endpoint constraints increases quadrature accuracy.

Orthogonal Collocation

Legendre polynomials satisfy

$$\int_{-1}^{1} P_i(\tau) P_j(\tau)\, d\tau = 0, \qquad i \neq j.$$

Because of this orthogonality, collocation at Gaussian points yields highly accurate numerical integration and transcription schemes.

Connection to Optimal Control

Within each mesh interval:

1. Approximate the state by a polynomial.
2. Differentiate the polynomial.
3. Enforce the dynamics at Gaussian collocation points.
4. Use Gaussian quadrature to approximate the integral cost.
5. Enforce continuity between neighboring mesh intervals.

This produces a sparse nonlinear programming problem.

Summary

- Euler collocation is the simplest orthogonal collocation method.
- Polynomial approximations replace piecewise linear approximations.
- Continuity supplies the additional equation needed in each interval.
- LG, LGR, and LGL differ only in endpoint constraints.
- Gaussian quadrature dramatically improves integration accuracy.
- These ideas form the basis of pseudospectral optimal-control methods.

Connection. The polynomial constructions used in orthogonal collocation rest on Legendre orthogonality, Lagrange interpolation, and exactness properties of Gaussian quadrature.

Legendre Polynomials, Lagrange Interpolation, and Gaussian Quadrature

This section develops the mathematical foundation behind Gaussian quadrature and orthogonal collocation. The key ingredients are Legendre polynomials, Lagrange interpolating polynomials, and Gaussian integration rules.

Legendre Polynomials

Legendre polynomials satisfy

$$\frac{d}{d\tau}\left((1-\tau^2)\frac{dP_n}{d\tau}\right) + n(n+1)P_n(\tau) = 0,$$

for $n = 0, 1, \ldots$ on $\tau \in [-1, 1]$.
The first two are

$$P_0(\tau) = 1, \qquad P_1(\tau) = \tau.$$

The recurrence relation is

$$(n+1)P_{n+1} = (2n+1)\tau P_n - nP_{n-1}.$$

Orthogonality

Legendre polynomials satisfy

$$\int_{-1}^{1} P_i(\tau)P_j(\tau)\,d\tau = \frac{2}{2j+1}\delta_{ij}.$$

Therefore every polynomial of degree at most $n-1$ is orthogonal to P_n.

Lagrange Interpolation

Given distinct support points

$$\tau_1, \ldots, \tau_N,$$

define

$$L_i(\tau) = \prod_{j \neq i} \frac{\tau - \tau_j}{\tau_i - \tau_j}.$$

These satisfy

$$L_i(\tau_j) = \delta_{ij}.$$

Hence an interpolant is

$$g(\tau) \approx \sum_{i=1}^{N} g(\tau_i) L_i(\tau).$$

Gaussian Quadrature

Approximate

$$\int_{-1}^{1} g(\tau)\, d\tau \approx \sum_{i=1}^{N} w_i g(\tau_i),$$

where

$$w_i = \int_{-1}^{1} L_i(\tau)\, d\tau.$$

Legendre–Gauss (LG)

LG points are the roots of

$$P_N(\tau).$$

Using N points exactly integrates every polynomial of degree

$$2N - 1.$$

Legendre–Gauss–Radau (LGR)

One endpoint is fixed, typically

$$\tau_1 = -1.$$

The remaining points are roots of

$$P_N(\tau) + P_{N-1}(\tau).$$

Exactness:

$$2N - 2.$$

Legendre–Gauss–Lobatto (LGL)

Both endpoints are fixed,

$$\tau_1 = -1, \qquad \tau_N = 1.$$

Interior points are roots of

$$P'_{N-1}(\tau).$$

Exactness:

$$2N - 3.$$

Comparison

Method	Endpoints	Exact degree
LG	None	$2N - 1$
LGR	One	$2N - 2$
LGL	Both	$2N - 3$

Why Gaussian Quadrature Works

A polynomial of degree at most $2N - 1$ can be decomposed into

$$g(\tau) = P_N(\tau) f(\tau) + h(\tau),$$

where f and h have degree at most $N - 1$. Orthogonality eliminates the first term after integration, while the remainder is represented exactly by the Lagrange interpolant.

Accuracy Growth

For LG quadrature:

Points	Exact degree
2	3
3	5
4	7
5	9
6	11
7	13

Each additional collocation point increases the exact polynomial degree by two.

Application to Optimal Control

Gaussian quadrature provides highly accurate approximations for:

- integral cost functions,
- state integration,
- orthogonal collocation methods,
- pseudospectral optimal control.

Summary

- Legendre polynomials are orthogonal on $[-1, 1]$.
- Lagrange polynomials interpolate exactly at support points.
- Gaussian quadrature combines both ideas.
- LG, LGR and LGL differ by endpoint constraints.
- Gaussian quadrature achieves much higher accuracy than low-order rules using the same number of points.

Connection. Those approximation tools determine both quadrature weights and the endpoint structure of the LGL transcription.

Gaussian Quadrature Weights and LGL Pseudospectral Methods

This section develops Gaussian quadrature weights and introduces the Legendre–Gauss–Lobatto (LGL) pseudospectral transcription for optimal control. It concludes by explaining why the differentiation matrix of the LGL method is singular and therefore does not constitute a true quadrature (integration) method.

Gaussian Quadrature Weights

For N Legendre–Gauss (LG) points,

$$w_i = \frac{2}{(1-\tau_i^2)\left(P_N'(\tau_i)\right)^2}, \qquad i = 1, \ldots, N.$$

For Legendre–Gauss–Radau (LGR),

$$w_1 = \frac{2}{N^2},$$

and

$$w_i = \frac{1}{(1-\tau_i)\left(P_{N-1}'(\tau_i)\right)^2}, \qquad i = 2, \ldots, N.$$

For Legendre–Gauss–Lobatto (LGL),

$$w_1 = w_N = \frac{2}{N(N-1)},$$

and

$$w_i = \frac{2}{N(N-1)\left(P_{N-1}(\tau_i)\right)^2}, \qquad i = 2, \ldots, N-1.$$

Model Optimal Control Problem

Consider

$$\min \Phi = x(1)$$

subject to

$$\dot{x} = f(x, u, t),$$

with

$$x(-1) = x_0.$$

The interval has already been normalized to

$$\tau \in [-1, 1].$$

LGL State Approximation

Using all LGL nodes,

$$x(\tau) \approx X(\tau) = \sum_{i=1}^{N} X_i L_i(\tau),$$

where $L_i(\tau)$ are Lagrange basis polynomials.
Because

$$L_i(\tau_j) = \delta_{ij},$$

the coefficients satisfy

$$X(\tau_j) = X_j.$$

Differentiation Matrix

Differentiating,

$$\dot{X}(\tau) = \sum_{i=1}^{N} X_i L_i'(\tau).$$

Evaluating at each collocation point,

$$\sum_{i=1}^{N} D_{ji} X_i = f(X_j, U_j),$$

where

$$D_{ji} = L_i'(\tau_j).$$

Stacking every node gives

$$DX = F.$$

Boundary Conditions

The discretized initial condition becomes

$$X_1 = x_0,$$

and the Mayer objective becomes

$$\Phi = X_N.$$

The resulting NLP is

$$\min X_N$$

subject to

$$DX - F = 0,$$

and

$$X_1 = x_0.$$

Integral Form

A true integration method would satisfy

$$X_j = X_1 + \sum_{i=1}^{N} A_{ji} F_i,$$

which discretizes

$$x(\tau_j) = x(-1) + \int_{-1}^{\tau_j} f(\tau)\, d\tau.$$

The matrix A is obtained from Gaussian quadrature weights.

Why LGL Fails as a True Quadrature Method

The interpolation polynomial has degree

$$N - 1.$$

Its derivative therefore has degree

$$N - 2.$$

However, collocation is enforced at all N LGL points.

Thus the differentiation system contains one more equation than is required to uniquely determine the derivative polynomial.

Consequently,

$$D$$

is singular.

Therefore,

$$D^{-1}$$

does not exist.

The differential system

$$DX = F$$

cannot be transformed into

$$X = X_1\mathbf{1} + AF.$$

Hence the LGL differentiation equations are not equivalent to a Gaussian quadrature rule.

Interpretation

Differentiation destroys constants since

$$\frac{d}{dt}(c) = 0.$$

Therefore constant functions lie in the null space of the differentiation operator.

The singularity of the differentiation matrix is the finite-dimensional manifestation of this null space.

Comparison with Euler

Euler forward,

$$\frac{x_{k+1} - x_k}{h} = f_k,$$

may be rewritten as

$$x_{k+1} = x_k + hf_k,$$

which is an explicit integration rule.

Therefore Euler possesses an equivalent integral form.

The LGL differentiation equations do not.

Consequences

The lecture emphasizes that although LGL nodes appear attractive because they include both endpoints, their differentiation matrix is singular. This motivates the development of Legendre–Gauss and Legendre–Gauss–Radau pseudospectral methods, which preserve Gaussian quadrature while still enforcing boundary conditions correctly.

Summary

- Closed-form Gaussian quadrature weights were introduced.
- The LGL pseudospectral transcription was derived.
- The differentiation matrix was constructed.
- The LGL differentiation matrix is singular.
- Constants form the null space of differentiation.
- Unlike Euler integration, the LGL differentiation equations cannot be converted into an equivalent quadrature rule.
- This motivates LG and LGR pseudospectral methods.

Connection. Removing both endpoints from the collocation grid produces the LG method and reveals the relationship between rectangular differentiation and exact integration matrices.

The Legendre–Gauss Pseudospectral Method

This section develops the Legendre–Gauss (LG) pseudospectral method for direct optimal control. The method is particularly important because it shows how Gaussian quadrature can be preserved even though the LG points do not include either endpoint of the interval.

The key ideas are:

- the LG points lie strictly inside the interval $[-1, 1]$;
- the initial point is added to the state approximation as a non-collocated point;
- the terminal point is recovered using Gaussian quadrature;
- the resulting differentiation matrix is rectangular rather than square;
- a square submatrix of the differentiation matrix is invertible;
- its inverse is exactly the LG integration matrix; and
- the differential and integral forms of the LG method are equivalent.

Model Optimal-Control Problem

Consider the normalized optimal-control problem

$$\min_{\boldsymbol{x}(\cdot),\boldsymbol{u}(\cdot)} \Phi\big(\boldsymbol{x}(1)\big),$$

subject to

$$\dot{\boldsymbol{x}}(\tau) = \boldsymbol{f}\big(\boldsymbol{x}(\tau), \boldsymbol{u}(\tau), \tau\big), \qquad \tau \in [-1, 1],$$

and the initial condition

$$\boldsymbol{x}(-1) = \boldsymbol{x}_0.$$

The interval is already normalized. A general interval $[t_0, t_f]$ may be mapped to $[-1, 1]$ through an affine transformation.

Location of the LG Points

Let

$$\tau_1, \tau_2, \ldots, \tau_N$$

be the roots of the Legendre polynomial $P_N(\tau)$.

These points satisfy

$$-1 < \tau_1 < \tau_2 < \cdots < \tau_N < 1.$$

Therefore, neither endpoint belongs to the LG quadrature set.

Key point
The boundary conditions are imposed at $\tau = -1$ and $\tau = 1$, but the LG collocation points lie strictly inside the interval.

Why the Naive LG Construction Fails

Suppose the state were approximated only at the N LG points:

$$\boldsymbol{X}(\tau) = \sum_{i=1}^{N} \boldsymbol{X}_i L_i(\tau).$$

This polynomial has degree $N - 1$.

Its derivative has degree $N - 2$.

Collocating the derivative at all N LG points would again produce an over-collocated system, just as in the LGL formulation. In addition, the initial and final conditions would be enforced at the wrong points.

Thus, the LG method must be constructed differently.

Introduction of Non-Collocated Endpoints

Introduce

$$\tau_0 = -1$$

as a non-collocated initial point.

Also define

$$\tau_{N+1} = 1$$

as a non-collocated terminal point.

The initial point is included in the state approximation.

The terminal point is not included in the collocation polynomial, but it will later be recovered by Gaussian quadrature.

LG State Approximation

Approximate the state using the initial point and all N LG points:

$$\boxed{\boldsymbol{X}(\tau) = \sum_{i=0}^{N} \boldsymbol{X}_i L_i(\tau),}$$

where

$$L_i(\tau) = \prod_{\substack{j=0 \\ j \neq i}}^{N} \frac{\tau - \tau_j}{\tau_i - \tau_j}.$$

The approximation contains $N+1$ support points and therefore has degree N.
At the support points,

$$\boldsymbol{X}(\tau_j) = \boldsymbol{X}_j, \qquad j = 0, \ldots, N.$$

Derivative Approximation

Differentiating,

$$\dot{\boldsymbol{X}}(\tau) = \sum_{i=0}^{N} \boldsymbol{X}_i L_i'(\tau).$$

Since $\boldsymbol{X}(\tau)$ has degree N, its derivative has degree $N-1$.
A polynomial of degree $N-1$ is uniquely determined by N conditions.
The LG rule provides exactly N collocation points.
Thus, unlike the naive construction, the collocation count is now correct.

LG Collocation Equations

Collocate the derivative approximation at the N LG points:

$$\dot{\boldsymbol{X}}(\tau_j) = \boldsymbol{f}(\boldsymbol{X}_j, \boldsymbol{U}_j, \tau_j), \qquad j = 1, \ldots, N.$$

Therefore,

$$\sum_{i=0}^{N} \boldsymbol{X}_i L_i'(\tau_j) = \boldsymbol{f}(\boldsymbol{X}_j, \boldsymbol{U}_j, \tau_j).$$

Define

$$D_{ji} = L_i'(\tau_j), \qquad j = 1, \ldots, N, \qquad i = 0, \ldots, N.$$

The LG differentiation matrix is

$$\boxed{\boldsymbol{D} \in \mathbb{R}^{N \times (N+1)}.}$$

It is rectangular.

Matrix Form of the Differential Equations

Partition the matrix as

$$\boldsymbol{D} = \begin{bmatrix} \boldsymbol{D}_0 & \boldsymbol{D}_{1:N} \end{bmatrix},$$

where:

- $\boldsymbol{D}_0 \in \mathbb{R}^{N\times 1}$ is the first column;
- $\boldsymbol{D}_{1:N} \in \mathbb{R}^{N\times N}$ contains the remaining columns.

Define

$$\boldsymbol{X}_{1:N} = \begin{bmatrix} \boldsymbol{X}_1 \\ \vdots \\ \boldsymbol{X}_N \end{bmatrix},$$

and

$$\boldsymbol{F}_{1:N} = \begin{bmatrix} f(\boldsymbol{X}_1, \boldsymbol{U}_1, \tau_1) \\ \vdots \\ f(\boldsymbol{X}_N, \boldsymbol{U}_N, \tau_N) \end{bmatrix}.$$

The collocation equations become

$$\boxed{\boldsymbol{D}_0 \boldsymbol{X}_0 + \boldsymbol{D}_{1:N} \boldsymbol{X}_{1:N} = \boldsymbol{F}_{1:N}.}$$

The initial condition is

$$\boldsymbol{X}_0 = \boldsymbol{x}_0.$$

Integral Form of the Dynamics

The exact dynamics satisfy

$$\boldsymbol{x}(\tau) = \boldsymbol{x}(-1) + \int_{-1}^{\tau} \boldsymbol{f}\big(\boldsymbol{x}(s), \boldsymbol{u}(s), s\big)\, \mathrm{d}s.$$

At an LG point τ_j,

$$\boldsymbol{x}(\tau_j) = \boldsymbol{x}(-1) + \int_{-1}^{\tau_j} \boldsymbol{f}\big(\boldsymbol{x}(s), \boldsymbol{u}(s), s\big)\, \mathrm{d}s.$$

Approximate each integral using a scaled LG quadrature:

$$\boxed{\boldsymbol{X}_j = \boldsymbol{X}_0 + \sum_{i=1}^{N} A_{ji} \boldsymbol{F}_i, \qquad j = 1, \ldots, N.}$$

The coefficients A_{ji} are the integration weights from -1 to τ_j.
In matrix form,

$$\boxed{\boldsymbol{X}_{1:N} = \mathbf{1}\,\boldsymbol{X}_0 + \boldsymbol{A}_{1:N}\boldsymbol{F}_{1:N}.}$$

Here,

$$\boldsymbol{A}_{1:N} \in \mathbb{R}^{N\times N}$$

is the LG integration matrix.

Recovery of the Terminal State

The final point $\tau_{N+1} = 1$ is not a collocation point.
However,

$$\boldsymbol{x}(1) = \boldsymbol{x}(-1) + \int_{-1}^{1} \boldsymbol{f}\left(\boldsymbol{x}(\tau), \boldsymbol{u}(\tau), \tau\right) \mathrm{d}\tau.$$

Using the standard LG quadrature weights w_i,

$$\boxed{\boldsymbol{X}_{N+1} = \boldsymbol{X}_0 + \sum_{i=1}^{N} w_i \boldsymbol{F}_i.}$$

Equivalently,

$$\boxed{\boldsymbol{X}_{N+1} = \boldsymbol{X}_0 + \boldsymbol{w}^{\mathsf{T}}\boldsymbol{F}_{1:N}.}$$

The Mayer objective becomes

$$\Phi(\boldsymbol{X}_{N+1}).$$

Thus the terminal state is recovered without making $\tau = 1$ a collocation point.

Equivalence of Differential and Integral Forms

Starting with

$$\boldsymbol{D}_0\boldsymbol{X}_0 + \boldsymbol{D}_{1:N}\boldsymbol{X}_{1:N} = \boldsymbol{F}_{1:N},$$

assume that

$$\boldsymbol{D}_{1:N}$$

is nonsingular.

Then

$$\boldsymbol{X}_{1:N} = -\boldsymbol{D}_{1:N}^{-1}\boldsymbol{D}_0\boldsymbol{X}_0 + \boldsymbol{D}_{1:N}^{-1}\boldsymbol{F}_{1:N}.$$

The critical identities are

$$\boxed{-\boldsymbol{D}_{1:N}^{-1}\boldsymbol{D}_0 = \mathbf{1},}$$

and

$$\boxed{\boldsymbol{D}_{1:N}^{-1} = \boldsymbol{A}_{1:N}.}$$

Therefore,

$$\boldsymbol{X}_{1:N} = \mathbf{1}\,\boldsymbol{X}_0 + \boldsymbol{A}_{1:N}\boldsymbol{F}_{1:N},$$

which is exactly the integral form.

Key point
The LG differential transcription is mathematically equivalent to an LG Gaussian quadrature integration scheme.

Why the Initial Point Must Be Isolated

The initial condition acts as the integration constant.
When solving

$$\dot{\boldsymbol{x}} = \boldsymbol{f},$$

integration gives

$$\boldsymbol{x}(\tau) = \boldsymbol{x}(-1) + \int_{-1}^{\tau} \boldsymbol{f}(s)\,\mathrm{d}s.$$

The initial value is not part of the quadrature itself.
It must remain separate from the accumulated integral.
This is why the LG method includes $\tau_0 = -1$ in the state approximation but does not collocate the dynamics there.

Comparison with the LGL Method

Feature	LGL	LG
Quadrature endpoints	Both included	Neither included
State approximation	LGL nodes	Initial point plus LG nodes
Differentiation matrix	Square	Rectangular
Key submatrix	Singular	Nonsingular
Equivalent integral form	Not directly available	Exactly available
Terminal state	Direct node	Recovered by quadrature

Complete LG NLP Formulation

A simplified LG transcription is

$$\begin{aligned} \min_{\boldsymbol{X},\boldsymbol{U}} \quad & \Phi(\boldsymbol{X}_{N+1}) \\ \text{subject to} \quad & \boldsymbol{D}_0\boldsymbol{X}_0 + \boldsymbol{D}_{1:N}\boldsymbol{X}_{1:N} - \boldsymbol{F}_{1:N} = \boldsymbol{0}, \\ & \boldsymbol{X}_0 - \boldsymbol{x}_0 = \boldsymbol{0}, \\ & \boldsymbol{X}_{N+1} - \boldsymbol{X}_0 - \boldsymbol{w}^{\mathsf{T}}\boldsymbol{F}_{1:N} = \boldsymbol{0}. \end{aligned}$$

Additional endpoint, path, and control constraints may be appended.

Computational Interpretation

The differential form is often computationally preferable because:

- derivative matrices are easy to precompute;
- the collocation constraints retain sparse structure;
- the form is convenient for automatic differentiation;
- the final-state equation is a single quadrature constraint.

However, the integral form is essential for understanding why the method is a valid Gaussian quadrature method.

Remaining Limitation of the LG Method

The control is approximated only at the LG points:

$$\boldsymbol{U}_1, \ldots, \boldsymbol{U}_N.$$

There is no control variable at

$$\tau = -1.$$

Thus the first available control value occurs at

$$\tau_1 > -1.$$

This is inconvenient in:

- real-time control;
- receding-horizon control;
- model predictive control;
- applications requiring the control at the initial time.

This limitation motivates the Legendre–Gauss–Radau method, which includes one endpoint in the quadrature set.

Generalization to Multiple Mesh Intervals

For a multiple-interval transcription:

1. divide the normalized horizon into mesh intervals;
2. map each interval locally to $[-1, 1]$;
3. apply LG collocation inside each interval;
4. include the left endpoint of each interval as a non-collocated state point;
5. recover the right endpoint through quadrature;
6. impose continuity between neighboring intervals.

This produces an hp-type pseudospectral method when both interval sizes and polynomial degrees may vary.

Summary

1. LG points exclude both endpoints.
2. A naive LG transcription is invalid.
3. The initial point is added as a non-collocated support point.
4. The state polynomial has degree N.
5. Its derivative has degree $N - 1$.
6. The derivative is collocated at exactly N LG points.
7. The LG differentiation matrix is rectangular.
8. Its square submatrix $\boldsymbol{D}_{1:N}$ is invertible.
9. The inverse of this submatrix is the LG integration matrix.
10. The differential and integral forms are equivalent.
11. The final state is recovered using standard LG quadrature.
12. The method is a true Gaussian quadrature integration scheme.
13. The absence of an initial control value motivates LGR collocation.

Connection. Including one endpoint instead yields LGR, a closely related formulation with practical advantages for initial control and multiple-interval assembly.

The Legendre–Gauss–Radau Pseudospectral Method

This section develops the Legendre–Gauss–Radau (LGR) pseudospectral method, derives both its differential and integral forms, compares it with the LG method, and explains why LGR is often preferred for practical optimal control because it provides a control variable at the initial time.

Review of the LG Method

For LG collocation,

$$\tau_1 > -1, \qquad \tau_N < 1.$$

The state approximation is

$$X(\tau) = \sum_{i=0}^{N} X_i L_i(\tau),$$

leading to

$$D_0 X_0 + D_{1:N} X_{1:N} = F_{1:N},$$

and

$$X_{N+1} = X_0 + \mathbf{w}^T F_{1:N}.$$

LGR Collocation Points

Unlike LG, the LGR points include the left endpoint,

$$\tau_1 = -1,$$

while

$$\tau_N < 1.$$

Introduce one additional non-collocated point

$$\tau_{N+1} = 1.$$

State Approximation

Approximate the state by

$$X(\tau) = \sum_{i=1}^{N+1} X_i L_i(\tau),$$

with

$$L_i(\tau) = \prod_{\substack{j=1 \\ j \neq i}}^{N+1} \frac{\tau - \tau_j}{\tau_i - \tau_j}.$$

The approximation uses $N + 1$ support points and therefore has degree N.

Derivative Approximation

Differentiating,

$$\dot{X}(\tau) = \sum_{i=1}^{N+1} X_i L_i'(\tau).$$

The derivative has degree $N - 1$ and is uniquely determined by N conditions.

Collocation

Enforce the dynamics at the N LGR points,

$$\dot{X}(\tau_j) = f(X_j, U_j), \qquad j = 1, \ldots, N.$$

Define

$$D_{ji} = L_i'(\tau_j),$$

giving

$$\sum_{i=1}^{N+1} D_{ji} X_i = F_j.$$

Partition

$$D = \begin{bmatrix} D_1 & D_{2:N+1} \end{bmatrix},$$

to obtain

$$\boxed{D_1 X_1 + D_{2:N+1} X_{2:N+1} = F_{1:N}.}$$

Integral Form

The continuous dynamics satisfy

$$X(\tau) = X(-1) + \int_{-1}^{\tau} f(s)\, ds.$$

Evaluating at

$$\tau_j, \qquad j = 2, \ldots, N+1,$$

and approximating each integral by LGR quadrature gives

$$X_j = X_1 + \sum_{i=1}^{N} \alpha_{ji} F_i.$$

Collecting all equations,

$$\boxed{X_{2:N+1} = \mathbf{1} X_1 + A_{2:N+1} F_{1:N}.}$$

Equivalence

If

$$D_{2:N+1}$$

is nonsingular,

$$X_{2:N+1} = -D_{2:N+1}^{-1} D_1 X_1 + D_{2:N+1}^{-1} F_{1:N}.$$

The key identities are

$$-D_{2:N+1}^{-1} D_1 = \mathbf{1},$$

and

$$D_{2:N+1}^{-1} = A_{2:N+1}.$$

Hence the differential and integral formulations are mathematically identical.

Terminal State

The final row yields

$$X_{N+1} = X_1 + \sum_{i=1}^{N} w_i F_i,$$

which is exactly the LGR Gaussian quadrature formula.

Comparison with LG

Property	LG	LGR
Left endpoint	No	Yes
Right endpoint	No	No
Extra state point	Initial	Final
Control at initial time	No	Yes
Quadrature exactness	$2N - 1$	$2N - 2$
Integration method	Yes	Yes

Control Approximation

LG provides controls only at interior Gauss points,

$$U_1, \ldots, U_N,$$

so there is no control at t_0.
LGR provides

$$U_1, \ldots, U_N,$$

where U_1 corresponds to

$$\tau = -1,$$

allowing the optimal control to be applied immediately.

General Time Interval

For

$$t \in [t_0, t_f],$$

the mapping

$$t = \frac{t_f - t_0}{2}\tau + \frac{t_f + t_0}{2}$$

gives

$$\frac{dx}{d\tau} = \frac{t_f - t_0}{2} f(x, u, t),$$

and

$$J = \Phi + \frac{t_f - t_0}{2} \int_{-1}^{1} L(x, u, \tau)\, d\tau.$$

Only the scaling factor changes; the transcription remains unchanged.

Advantages

- True Gaussian quadrature method.
- Differential and integral forms are equivalent.
- Initial control value is available.
- Terminal state is recovered naturally.
- Sparse NLP structure.

Summary

1. LGR includes the left endpoint.
2. The state polynomial has degree N.
3. Dynamics are collocated at N Radau points.
4. The differentiation matrix has a full-rank square submatrix.
5. Its inverse is the Radau integration matrix.
6. The terminal state follows directly from Radau quadrature.
7. LGR sacrifices one degree of quadrature exactness relative to LG but gains an initial control value.

Exercises

1. **Legendre polynomials and orthogonality**
   The Legendre polynomials satisfy

$$(n+1)P_{n+1}(\tau) = (2n+1)\tau P_n(\tau) - nP_{n-1}(\tau),$$

   with

$$P_0(\tau) = 1, \qquad P_1(\tau) = \tau.$$

   (a) Use the recurrence relation to derive $P_2(\tau)$, $P_3(\tau)$, and $P_4(\tau)$.
   (b) Verify directly that

$$\int_{-1}^{1} P_3(\tau)P_j(\tau)\,\mathrm{d}\tau = 0, \qquad j = 0, 1, 2.$$

   (c) Verify that

$$\int_{-1}^{1} P_3^2(\tau)\,\mathrm{d}\tau = \frac{2}{7}.$$

   (d) Explain why every polynomial of degree at most 2 is orthogonal to P_3.
2. **Lagrange interpolation and differentiation**
   Consider the support points

$$\tau_1 = -1, \qquad \tau_2 = 0, \qquad \tau_3 = 1.$$

   (a) Construct the three Lagrange basis polynomials $L_i(\tau)$.
   (b) Verify the interpolation property

$$L_i(\tau_j) = \delta_{ij}.$$

   (c) Construct the interpolation polynomial for

$$g(\tau) = \tau^2 + 2\tau - 1$$

   using only the values $g(\tau_i)$, and show that the polynomial is recovered exactly.
   (d) Construct the differentiation matrix

$$D_{ji} = L_i'(\tau_j).$$

   (e) Verify that

$$\boldsymbol{D}\mathbf{1} = \mathbf{0}.$$

(f) Use $\boldsymbol{D}$ to recover $g'(\tau_i)$ exactly.

3. **Three-point Legendre–Gauss quadrature**

For $N = 3$, the LG points are the roots of $P_3(\tau)$.

(a) Determine the three LG points.

(b) Using symmetry and exact integration of 1, τ^2, and τ^4, derive the three quadrature weights.

(c) Verify that the resulting rule integrates every monomial τ^k, $k = 0, \ldots, 5$, exactly.

(d) Evaluate the quadrature error for

$$g(\tau) = \tau^6.$$

(e) Explain why this result is consistent with the LG exactness degree $2N - 1$.

4. **Proof of Legendre–Gauss exactness**

Let $g(\tau)$ be a polynomial of degree at most $2N - 1$.

(a) Show that it can be written as

$$g(\tau) = P_N(\tau)q(\tau) + r(\tau),$$

where q and r have degree at most $N - 1$.

(b) Use Legendre orthogonality to show that

$$\int_{-1}^{1} P_N(\tau)q(\tau)\,\mathrm{d}\tau = 0.$$

(c) Show that the LG quadrature contribution from $P_N(\tau)q(\tau)$ is also zero.

(d) Prove that the quadrature integrates $r(\tau)$ exactly.

(e) Conclude that N-point LG quadrature is exact for every polynomial of degree at most $2N - 1$.

5. **Low-order LGL construction and singularity**

Consider the $N = 4$ LGL grid.

(a) Starting from the roots of $P_3'(\tau)$, derive the four LGL points.

(b) Compute the four LGL quadrature weights.

(c) Construct the Lagrange basis polynomials associated with the four points.

(d) Construct the 4×4 differentiation matrix

$$D_{ji} = L_i'(\tau_j).$$

(e) Verify that

$$\boldsymbol{D}\mathbf{1} = \mathbf{0}.$$

(f) Conclude that $\boldsymbol{D}$ is singular.

(g) Relate the singularity to the fact that the state polynomial has degree $N-1$, whereas its derivative has degree $N-2$.

6. **Legendre–Gauss differential and integral forms**

Let $\tau_1, \dots, \tau_N$ be the LG points and introduce the non-collocated initial point

$$\tau_0 = -1.$$

Approximate the state by

$$\boldsymbol{X}(\tau) = \sum_{i=0}^{N} \boldsymbol{X}_i L_i(\tau).$$

(a) Show that the state polynomial has degree at most N.

(b) Show that its derivative has degree at most $N-1$.

(c) Derive the rectangular differentiation equations

$$\boldsymbol{D}_0 \boldsymbol{X}_0 + \boldsymbol{D}_{1:N} \boldsymbol{X}_{1:N} = \boldsymbol{F}_{1:N}.$$

(d) Assuming $\boldsymbol{D}_{1:N}$ is invertible, derive

$$\boldsymbol{X}_{1:N} = -\boldsymbol{D}_{1:N}^{-1} \boldsymbol{D}_0 \boldsymbol{X}_0 + \boldsymbol{D}_{1:N}^{-1} \boldsymbol{F}_{1:N}.$$

(e) Explain why $\boldsymbol{D}_{1:N}^{-1}$ acts as an integration matrix.

(f) Derive the terminal-state recovery equation

$$\boldsymbol{X}_{N+1} = \boldsymbol{X}_0 + \boldsymbol{w}^{\mathsf{T}} \boldsymbol{F}_{1:N}.$$

7. **Explicit two-point LG differentiation matrix**

Use $N = 2$ LG points,

$$\tau_1 = -\frac{1}{\sqrt{3}}, \qquad \tau_2 = \frac{1}{\sqrt{3}},$$

together with the initial support point $\tau_0 = -1$.

(a) Construct the three Lagrange basis polynomials.

(b) Construct the 2×3 LG differentiation matrix evaluated at τ_1 and τ_2.

(c) Partition it as

$$\boldsymbol{D} = \begin{bmatrix} \boldsymbol{D}_0 & \boldsymbol{D}_{1:2} \end{bmatrix}.$$

(d) Verify that $\boldsymbol{D}_{1:2}$ is nonsingular.

(e) Compute $\boldsymbol{D}_{1:2}^{-1}$.

(f) For

$$F(\tau) = 1 + \tau, \qquad X(-1) = X_0,$$

recover $X(\tau_1)$, $X(\tau_2)$, and $X(1)$, and compare them with the exact integral of F.

8. **Legendre–Gauss–Radau differential and integral forms**

Let $\tau_1, \ldots, \tau_N$ be left-LGR collocation points, where

$$\tau_1 = -1, \qquad \tau_N < 1,$$

and introduce the non-collocated terminal support point

$$\tau_{N+1} = 1.$$

Approximate the state by

$$\boldsymbol{X}(\tau) = \sum_{i=1}^{N+1} \boldsymbol{X}_i L_i(\tau).$$

(a) Determine the degree of the state polynomial.

(b) Determine the degree of its derivative.

(c) Show that collocation at the N LGR points gives

$$\boldsymbol{D}_1 \boldsymbol{X}_1 + \boldsymbol{D}_{2:N+1} \boldsymbol{X}_{2:N+1} = \boldsymbol{F}_{1:N}.$$

(d) Assuming $\boldsymbol{D}_{2:N+1}$ is invertible, derive the corresponding integral form.

(e) Explain why the terminal state is directly included in the state polynomial but is not a collocation value.

(f) Explain why the LGR scheme provides a control value at the initial time, whereas the LG scheme does not.

9. **Comparison of LG, LGR, and LGL**

For each of the LG, LGR, and LGL methods using N quadrature points, determine:

(a) which endpoints are included among the collocation points;

(b) the polynomial defining the collocation points;

(c) the exact polynomial degree of the quadrature rule;

(d) the degree of the state interpolant used in the corresponding pseudospectral construction;

(e) the degree of the derivative polynomial;

(f) whether the state differentiation matrix is square or rectangular;

(g) whether an ordinary inverse or an invertible square submatrix exists;

(h) how the terminal state is represented.

Then, for $N = 6$, compute the exactness degree of all three quadrature rules and determine which methods are guaranteed to integrate τ^{10} exactly.

10. **Complete LGR pseudospectral transcription**

On the normalized interval $[-1, 1]$, consider

$$\min_{x(\cdot),u(\cdot)} J = \frac{1}{2}x^2(1) + \frac{1}{2}\int_{-1}^{1} \left(x^2(\tau) + u^2(\tau)\right)\, \mathrm{d}\tau,$$

subject to

$$\dot{x}(\tau) = -x(\tau) + u(\tau), \qquad x(-1) = x_0.$$

Use N left-LGR collocation points and the non-collocated terminal point $\tau_{N+1} = 1$.

(a) Define all state and control decision variables.

(b) Write the LGR state interpolation polynomial.

(c) Construct the collocation equations in matrix form.

(d) Write the Gaussian-quadrature approximation of the integral cost.

(e) Write the complete finite-dimensional objective.

(f) Write the initial boundary constraint.

(g) Assemble the complete nonlinear program.

(h) Determine the number of scalar decision variables and equality constraints.

(i) Explain where the terminal state enters the NLP even though $\tau = 1$ is not a collocation point.

Chapter 6: Costate Recovery and Nonlinear Programming Implementation

Chapter 6: Costate Recovery and Nonlinear Programming Implementation

From KKT multipliers to continuous costates and solver-ready LGR implementations

A direct transcription produces primal variables and KKT multipliers, whereas continuous optimal-control theory describes states, controls, and costates. This chapter connects the two descriptions. It derives costate mappings for LG, LGR, and LGL grids, generalizes the transcription to practical Bolza problems, and closes by organizing the resulting arrays, constraints, objective, and initial guess for an NLP solver.

Chapter contents

Costate Estimation Using the Legendre–Gauss Method

This section begins the derivation of costate estimates from the dual variables of a Legendre–Gauss (LG) pseudospectral nonlinear programming problem. The central task is to connect the continuous-time first-order optimality conditions with the Karush–Kuhn–Tucker conditions of the discretized nonlinear program.

The chapter develops:

- the continuous Hamiltonian optimality conditions;
- the LG pseudospectral nonlinear program;
- the NLP Lagrangian;
- the KKT stationarity conditions;
- a separate polynomial approximation for the costate;
- the adjoint differentiation matrix $\boldsymbol{D}^{\dagger}$; and
- the matrix identities needed for the costate mapping theorem.

Simplified Mayer Optimal-Control Problem

Consider

$$\min_{\boldsymbol{x}(\cdot),\boldsymbol{u}(\cdot)} \Phi\big(\boldsymbol{x}(1)\big),$$

subject to

$$\dot{\boldsymbol{x}}(\tau) = \boldsymbol{f}\big(\boldsymbol{x}(\tau), \boldsymbol{u}(\tau)\big), \qquad \tau \in [-1, 1],$$

and

$$\boldsymbol{x}(-1) = \boldsymbol{x}_0.$$

No Lagrange running cost is included, so the Hamiltonian is

$$H(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\lambda}) = \boldsymbol{\lambda}^{\mathsf{T}} \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}).$$

Continuous-Time Optimality Conditions

The first-order necessary conditions are:

Terminal Transversality

$$\boxed{\boldsymbol{\lambda}(1) = \nabla_{\boldsymbol{x}} \Phi\big(\boldsymbol{x}(1)\big).}$$

Costate Equation

$$\boxed{\dot{\boldsymbol{\lambda}}(\tau) = -\nabla_{\boldsymbol{x}} H = -\nabla_{\boldsymbol{x}} \left[\boldsymbol{\lambda}(\tau)^{\mathsf{T}} \boldsymbol{f}\left(\boldsymbol{x}(\tau), \boldsymbol{u}(\tau)\right)\right].}$$

Stationarity with Respect to Control

$$\boxed{\nabla_{\boldsymbol{u}} H = \nabla_{\boldsymbol{u}} \left[\boldsymbol{\lambda}(\tau)^{\mathsf{T}} \boldsymbol{f}\left(\boldsymbol{x}(\tau), \boldsymbol{u}(\tau)\right)\right] = \mathbf{0}.}$$

The objective is to recover approximations of $\boldsymbol{\lambda}(\tau)$ from the KKT multipliers of the LG transcription.

LG Grid and State Approximation

Let

$$\tau_1, \ldots, \tau_N$$

be the LG points, with

$$-1 < \tau_1 < \cdots < \tau_N < 1.$$

Define

$$\tau_0 = -1, \qquad \tau_{N+1} = 1.$$

Approximate the state as

$$\boldsymbol{X}(\tau) = \sum_{i=0}^{N} \boldsymbol{X}_i L_i(\tau),$$

where

$$L_i(\tau) = \prod_{\substack{j=0 \\ j \neq i}}^{N} \frac{\tau - \tau_j}{\tau_i - \tau_j}.$$

Differentiation gives

$$\dot{\boldsymbol{X}}(\tau) = \sum_{i=0}^{N} \boldsymbol{X}_i L_i'(\tau).$$

LG Differentiation Matrix

Define

$$D_{ji} = L_i'(\tau_j), \qquad j = 1, \ldots, N, \qquad i = 0, \ldots, N.$$

Thus,

$$\boldsymbol{D} \in \mathbb{R}^{N \times (N+1)}.$$

The collocation equations are

$$\boxed{\boldsymbol{D}\boldsymbol{X}_{0:N} = \boldsymbol{F}_{1:N},}$$

where

$$\boldsymbol{F}_j = \boldsymbol{f}(\boldsymbol{X}_j, \boldsymbol{U}_j).$$

The terminal state is estimated by LG quadrature:

$$\boxed{\boldsymbol{X}_{N+1} = \boldsymbol{X}_0 + \boldsymbol{w}^\mathsf{T}\boldsymbol{F}_{1:N}.}$$

The LG Nonlinear Program

The discretized NLP is

$$\boxed{\begin{array}{rl} \min\limits_{\boldsymbol{X},\boldsymbol{U}} & \Phi(\boldsymbol{X}_{N+1}) \\ \text{subject to} & \boldsymbol{F}_{1:N} - \boldsymbol{D}\boldsymbol{X}_{0:N} = \boldsymbol{0}, \\ & \boldsymbol{w}^\mathsf{T}\boldsymbol{F}_{1:N} + \boldsymbol{X}_0 - \boldsymbol{X}_{N+1} = \boldsymbol{0}, \\ & \boldsymbol{X}_0 - \boldsymbol{x}_0 = \boldsymbol{0}. \end{array}}$$

Because $\boldsymbol{X}_0$ is fixed, it may be removed from the optimization vector after substitution.

KKT Multipliers

Let

$$\boldsymbol{\Lambda}_{1:N} = \begin{bmatrix} \boldsymbol{\Lambda}_1 \\ \vdots \\ \boldsymbol{\Lambda}_N \end{bmatrix}$$

denote the multipliers associated with the collocation equations.
Let

$$\boldsymbol{\Lambda}_{N+1}$$

denote the multiplier associated with the terminal quadrature equation.

These are NLP multipliers, not yet costate estimates.

NLP Lagrangian

Define

$$\begin{aligned}\mathcal{L} = \Phi(\boldsymbol{X}_{N+1}) &+ \langle \boldsymbol{\Lambda}_{1:N}, \boldsymbol{F}_{1:N} - \boldsymbol{D}\boldsymbol{X}_{0:N} \rangle \\ &+ \left\langle \boldsymbol{\Lambda}_{N+1}, \boldsymbol{w}^{\mathsf{T}} \boldsymbol{F}_{1:N} + \boldsymbol{X}_0 - \boldsymbol{X}_{N+1} \right\rangle .\end{aligned}$$

The angle brackets denote the appropriate Euclidean or Frobenius inner product.

Stationarity Conditions of the NLP

Differentiation with respect to the discrete state variables at the LG points yields

$$\boxed{\boldsymbol{D}_{1:N}^{\mathsf{T}} \boldsymbol{\Lambda}_{1:N} = \nabla_{\boldsymbol{X}} \left\langle \boldsymbol{\Lambda}_{1:N} + \boldsymbol{W}\mathbf{1}\, \boldsymbol{\Lambda}_{N+1}, \boldsymbol{F}_{1:N} \right\rangle .}$$

Here,

$$\boldsymbol{W} = \operatorname{diag}(w_1, \ldots, w_N).$$

Differentiation with respect to the terminal state gives

$$\boxed{\boldsymbol{\Lambda}_{N+1} = \nabla_{\boldsymbol{X}} \Phi(\boldsymbol{X}_{N+1}).}$$

Differentiation with respect to the controls gives

$$\boxed{\mathbf{0} = \nabla_{\boldsymbol{U}} \left\langle \boldsymbol{W}^{-1} \boldsymbol{\Lambda}_{1:N} + \mathbf{1}\, \boldsymbol{\Lambda}_{N+1}, \boldsymbol{F}_{1:N} \right\rangle .}$$

These equations already resemble the continuous costate, transversality, and stationarity conditions.

Why a Second Polynomial Space Is Needed

The state approximation uses

$$\tau_0, \tau_1, \ldots, \tau_N,$$

that is, the initial point plus the LG points.
The costate approximation must instead use

$$\tau_1, \ldots, \tau_N, \tau_{N+1},$$

that is, the LG points plus the final point.
This shift is essential because the terminal costate is specified by the transversality condition.

Costate Approximation

Approximate the costate by

$$\boxed{\boldsymbol{\Lambda}(\tau) = \sum_{i=1}^{N+1} \boldsymbol{\lambda}_i L_i^\dagger(\tau),}$$

where

$$L_i^\dagger(\tau) = \prod_{\substack{j=1 \\ j\neq i}}^{N+1} \frac{\tau - \tau_j}{\tau_i - \tau_j}.$$

This is a polynomial of degree N.
Its derivative is

$$\dot{\boldsymbol{\Lambda}}(\tau) = \sum_{i=1}^{N+1} \boldsymbol{\lambda}_i \left(L_i^\dagger\right)'(\tau).$$

Adjoint Differentiation Matrix

Define

$$D_{ji}^\dagger = \left(L_i^\dagger\right)'(\tau_j), \qquad j = 1, \ldots, N, \qquad i = 1, \ldots, N+1.$$

Then

$$\boxed{\boldsymbol{D}^\dagger \in \mathbb{R}^{N\times(N+1)}.}$$

Partition it as

$$\boldsymbol{D}^\dagger = \begin{bmatrix} \boldsymbol{D}_{1:N}^\dagger & \boldsymbol{d}_{N+1}^\dagger \end{bmatrix}.$$

Relationship Between the State and Costate Differentiation Matrices

The crucial weighted transpose relation is

$$\boxed{\boldsymbol{D}_{1:N}^{\mathsf{T}} = -\boldsymbol{W}\boldsymbol{D}_{1:N}^\dagger \boldsymbol{W}^{-1}.}$$

Equivalently,

$$\boxed{\boldsymbol{D}_{1:N}^\dagger = -\boldsymbol{W}^{-1}\boldsymbol{D}_{1:N}^{\mathsf{T}} \boldsymbol{W}.}$$

This identity provides the bridge between the KKT system and the discrete costate equation.

Derivative of a Constant

Because a differentiation matrix must annihilate a constant polynomial,

$$\boldsymbol{D}^{\dagger}\mathbf{1}_{N+1} = \mathbf{0}.$$

Therefore,

$$\boldsymbol{D}_{1:N}^{\dagger}\mathbf{1} + \boldsymbol{d}_{N+1}^{\dagger} = \mathbf{0},$$

and

$$\boxed{\boldsymbol{d}_{N+1}^{\dagger} = -\boldsymbol{D}_{1:N}^{\dagger}\mathbf{1}.}$$

This relation is needed to incorporate the terminal costate into the discrete adjoint equation.

Preview of the Costate Mapping

Define the interior costate estimate by

$$\boxed{\boldsymbol{\lambda}_{1:N} = \boldsymbol{W}^{-1}\boldsymbol{\Lambda}_{1:N} + \mathbf{1}\,\boldsymbol{\Lambda}_{N+1}.}$$

Define the terminal costate by

$$\boxed{\boldsymbol{\lambda}_{N+1} = \boldsymbol{\Lambda}_{N+1}.}$$

Substitution into the KKT equations produces a discrete analogue of

$$\dot{\boldsymbol{\lambda}} = -\nabla_{\boldsymbol{x}}\left(\boldsymbol{\lambda}^{\mathsf{T}}\boldsymbol{f}\right).$$

The detailed transformation is developed in Session 22B.

Summary

1. The continuous costate equation follows from the Hamiltonian.
2. The LG transcription produces collocation and terminal quadrature constraints.
3. Each constraint has an associated KKT multiplier.
4. The NLP Lagrangian yields discrete stationarity conditions.
5. The state and costate use different Lagrange polynomial bases.
6. The costate basis uses the LG points plus the final point.
7. The weighted transpose relation between $\boldsymbol{D}$ and $\boldsymbol{D}^{\dagger}$ is the key algebraic identity.
8. These results prepare the costate mapping theorem.

Connection. The LG KKT equations provide the raw multiplier relations; the next step completes their transformation into endpoint and interior costate estimates.

Costate Mapping and Recovery of Continuous Costates

This section completes the LG costate-estimation derivation. It transforms the KKT multipliers of the pseudospectral nonlinear program into estimates of the continuous costate at the LG points and both endpoints.

The principal results are:

$$\boxed{\boldsymbol{\lambda}_{N+1} = \boldsymbol{\Lambda}_{N+1},}$$

$$\boxed{\boldsymbol{\lambda}_{1:N} = \boldsymbol{W}^{-1}\boldsymbol{\Lambda}_{1:N} + \mathbf{1}\,\boldsymbol{\Lambda}_{N+1},}$$

and

$$\boxed{\boldsymbol{\lambda}_0 = \boldsymbol{\lambda}_{N+1} - \boldsymbol{d}_0^{\mathsf{T}}\boldsymbol{\Lambda}_{1:N}.}$$

Starting KKT System

The LG KKT conditions are

$$\boldsymbol{D}_{1:N}^{\mathsf{T}}\boldsymbol{\Lambda}_{1:N} = \nabla_{\boldsymbol{X}}\left\langle \boldsymbol{\Lambda}_{1:N} + \boldsymbol{W}\mathbf{1}\boldsymbol{\Lambda}_{N+1}, \boldsymbol{F}_{1:N}\right\rangle,$$

$$\boldsymbol{\Lambda}_{N+1} = \nabla_{\boldsymbol{X}}\Phi(\boldsymbol{X}_{N+1}),$$

and

$$\mathbf{0} = \nabla_{\boldsymbol{U}}\left\langle \boldsymbol{W}^{-1}\boldsymbol{\Lambda}_{1:N} + \mathbf{1}\boldsymbol{\Lambda}_{N+1}, \boldsymbol{F}_{1:N}\right\rangle.$$

Weighted Transpose Identity

Use

$$\boldsymbol{D}_{1:N}^{\mathsf{T}} = -\boldsymbol{W}\boldsymbol{D}_{1:N}^{\dagger}\boldsymbol{W}^{-1}.$$

Substitution into the first KKT condition gives

$$-\boldsymbol{W}\boldsymbol{D}_{1:N}^{\dagger}\boldsymbol{W}^{-1}\boldsymbol{\Lambda}_{1:N} = \nabla_{\boldsymbol{X}}\left\langle \boldsymbol{\Lambda}_{1:N} + \boldsymbol{W}\mathbf{1}\boldsymbol{\Lambda}_{N+1}, \boldsymbol{F}_{1:N}\right\rangle.$$

Premultiplying by $\boldsymbol{W}^{-1}$ yields

$$-\boldsymbol{D}_{1:N}^{\dagger}\boldsymbol{W}^{-1}\boldsymbol{\Lambda}_{1:N} = \nabla_{\boldsymbol{X}}\left\langle \boldsymbol{W}^{-1}\boldsymbol{\Lambda}_{1:N} + \mathbf{1}\boldsymbol{\Lambda}_{N+1}, \boldsymbol{F}_{1:N}\right\rangle.$$

Interior Costate Mapping

Define

$$\boxed{\boldsymbol{\lambda}_{1:N} = \boldsymbol{W}^{-1}\boldsymbol{\Lambda}_{1:N} + \mathbf{1}\boldsymbol{\Lambda}_{N+1}.}$$

Therefore,

$$\boldsymbol{W}^{-1}\boldsymbol{\Lambda}_{1:N} = \boldsymbol{\lambda}_{1:N} - \mathbf{1}\boldsymbol{\Lambda}_{N+1}.$$

Substituting,

$$-\boldsymbol{D}_{1:N}^{\dagger}\left(\boldsymbol{\lambda}_{1:N} - \mathbf{1}\boldsymbol{\Lambda}_{N+1}\right) = \nabla_{\boldsymbol{X}}\left\langle \boldsymbol{\lambda}_{1:N}, \boldsymbol{F}_{1:N}\right\rangle.$$

Expanding,

$$-\boldsymbol{D}_{1:N}^{\dagger}\boldsymbol{\lambda}_{1:N} + \boldsymbol{D}_{1:N}^{\dagger}\mathbf{1}\boldsymbol{\Lambda}_{N+1} = \nabla_{\boldsymbol{X}}\left\langle \boldsymbol{\lambda}_{1:N}, \boldsymbol{F}_{1:N}\right\rangle.$$

Terminal Costate Mapping

Define

$$\boxed{\boldsymbol{\lambda}_{N+1} = \boldsymbol{\Lambda}_{N+1}.}$$

This agrees immediately with the continuous terminal condition:

$$\boldsymbol{\lambda}_{N+1} = \nabla_{\boldsymbol{X}}\Phi(\boldsymbol{X}_{N+1}).$$

Using

$$\boldsymbol{d}_{N+1}^{\dagger} = -\boldsymbol{D}_{1:N}^{\dagger}\mathbf{1},$$

the adjoint equation becomes

$$\boxed{\boldsymbol{D}_{1:N}^{\dagger}\boldsymbol{\lambda}_{1:N} + \boldsymbol{d}_{N+1}^{\dagger}\boldsymbol{\lambda}_{N+1} = -\nabla_{\boldsymbol{X}}\left\langle \boldsymbol{\lambda}_{1:N}, \boldsymbol{F}_{1:N}\right\rangle.}$$

Equivalently,

$$\boxed{\boldsymbol{D}^{\dagger}\boldsymbol{\lambda}_{1:N+1} = -\nabla_{\boldsymbol{X}}\left\langle \boldsymbol{\lambda}_{1:N}, \boldsymbol{F}_{1:N}\right\rangle.}$$

This is the discrete costate equation.

Comparison with the Continuous Costate Equation

The continuous equation is

$$\dot{\boldsymbol{\lambda}}(\tau) = -\nabla_{\boldsymbol{x}} \left[\boldsymbol{\lambda}(\tau)^{\mathsf{T}} \boldsymbol{f}(\boldsymbol{x}(\tau), \boldsymbol{u}(\tau)) \right].$$

The discrete LG equation is

$$\boldsymbol{D}^{\dagger} \boldsymbol{\lambda}_{1:N+1} = -\nabla_{\boldsymbol{X}} \left\langle \boldsymbol{\lambda}_{1:N}, \boldsymbol{F}_{1:N} \right\rangle.$$

Thus, $\boldsymbol{D}^{\dagger}\boldsymbol{\lambda}$ approximates the derivative of the costate polynomial at the LG points.

Discrete Control Stationarity

The control KKT condition becomes

$$\boxed{\mathbf{0} = \nabla_{\boldsymbol{U}} \left\langle \boldsymbol{\lambda}_{1:N}, \boldsymbol{F}_{1:N} \right\rangle.}$$

This is the discrete equivalent of

$$\nabla_{\boldsymbol{u}} H(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\lambda}) = \mathbf{0}.$$

Recovery of the Initial Costate

The costate dynamics satisfy

$$\boldsymbol{\lambda}(1) - \boldsymbol{\lambda}(-1) = \int_{-1}^{1} \dot{\boldsymbol{\lambda}}(\tau) \, \mathrm{d}\tau.$$

Therefore,

$$\boldsymbol{\lambda}(-1) = \boldsymbol{\lambda}(1) - \int_{-1}^{1} \dot{\boldsymbol{\lambda}}(\tau) \, \mathrm{d}\tau.$$

Using LG quadrature,

$$\boldsymbol{\lambda}_0 \approx \boldsymbol{\lambda}_{N+1} - \boldsymbol{w}^{\mathsf{T}} \boldsymbol{D}^{\dagger} \boldsymbol{\lambda}_{1:N+1}.$$

Using the relation between $\boldsymbol{D}^{\dagger}$ and the original state differentiation matrix, this reduces to

$$\boxed{\boldsymbol{\lambda}_0 = \boldsymbol{\lambda}_{N+1} - \boldsymbol{d}_0^{\mathsf{T}} \boldsymbol{\Lambda}_{1:N},}$$

where $\boldsymbol{d}_0$ is the first column of the original LG differentiation matrix.

Complete LG Costate Mapping Theorem

Key point
Let $\boldsymbol{\Lambda}_{1:N}$ be the KKT multipliers associated with the LG collocation constraints and let $\boldsymbol{\Lambda}_{N+1}$ be the multiplier associated with the terminal quadrature equation. Then the costate estimates are

$$\boldsymbol{\lambda}_{N+1} = \boldsymbol{\Lambda}_{N+1},$$

$$\boldsymbol{\lambda}_{1:N} = \boldsymbol{W}^{-1}\boldsymbol{\Lambda}_{1:N} + \mathbf{1}\boldsymbol{\Lambda}_{N+1},$$

and

$$\boldsymbol{\lambda}_0 = \boldsymbol{\lambda}_{N+1} - \boldsymbol{d}_0^{\mathsf{T}}\boldsymbol{\Lambda}_{1:N}.$$

Interpretation of the Mapping

The collocation multipliers alone are not the costates.

Two corrections are required:

1. scaling by the inverse quadrature weights;
2. addition of the terminal quadrature multiplier.

The terminal multiplier appears because the LG state approximation does not contain the terminal point, and the final state is imposed through an integral constraint.

Implementation Algorithm

1. Solve the LG pseudospectral NLP.
2. Extract the multipliers $\boldsymbol{\Lambda}_{1:N}$ associated with the collocation equations.
3. Extract $\boldsymbol{\Lambda}_{N+1}$ associated with the terminal quadrature equation.
4. Form

$$\boldsymbol{\lambda}_{N+1} = \boldsymbol{\Lambda}_{N+1}.$$

5. Compute

$$\boldsymbol{\lambda}_{1:N} = \boldsymbol{W}^{-1}\boldsymbol{\Lambda}_{1:N} + \mathbf{1}\boldsymbol{\lambda}_{N+1}.$$

6. Compute

$$\boldsymbol{\lambda}_0 = \boldsymbol{\lambda}_{N+1} - \boldsymbol{d}_0^{\mathsf{T}}\boldsymbol{\Lambda}_{1:N}.$$

7. Compare the resulting costate with analytical or indirect-solution results when available.

Practical Issues

Constraint Ordering

The NLP solver returns multipliers in the same ordering as the constraints. The software implementation must preserve a clear map from each multiplier block to its corresponding collocation or quadrature constraint.

Sign Conventions

Different solvers use different conventions for equality-constraint multipliers. A sign reversal may be required if the NLP is written as

$$\boldsymbol{DX} - \boldsymbol{F} = \mathbf{0}$$

instead of

$$\boldsymbol{F} - \boldsymbol{DX} = \mathbf{0}.$$

Scaling

Poor state or constraint scaling can degrade multiplier accuracy even when the primal solution is accurate.

Mesh Refinement

Costate estimates should be checked under mesh or polynomial-order refinement. Smooth convergence is an important validation test.

Why Costate Estimation Matters

Costates provide:

- sensitivity information;
- interpretation of active design tradeoffs;
- necessary-condition verification;
- information for switching-function analysis;
- insight into state importance;
- a bridge between direct and indirect methods.

Summary

1. KKT multipliers from the LG NLP can be mapped to continuous costate estimates.
2. The terminal costate equals the terminal quadrature multiplier.
3. Interior costates require quadrature-weight scaling and a terminal correction.

4. The resulting discrete adjoint equation matches the continuous costate equation.
5. The initial costate follows from quadrature of the discrete adjoint dynamics.
6. The mapping is computationally inexpensive once the NLP has been solved.

Connection. The same continuous–discrete comparison can be repeated on the Radau grid, where the included initial endpoint changes the mapping structure.

Costate Estimation Using the Legendre–Gauss–Radau Method

This section develops costate estimation for an optimal-control problem transcribed with the Legendre–Gauss–Radau (LGR) pseudospectral method. The main objective is to derive a direct relationship between:

1. the Karush–Kuhn–Tucker (KKT) multipliers of the discretized nonlinear program; and
2. approximations of the continuous-time costate.

The LGR method is especially convenient for costate estimation because one endpoint is itself a collocation point. For the left-Radau grid considered here, the initial point is included among the LGR points, whereas the final point is a noncollocated interpolation point.

The final mapping is

$$\boxed{\boldsymbol{\lambda}_{1:N} = \boldsymbol{W}^{-1}\boldsymbol{\Lambda}_{1:N},}$$

and

$$\boxed{\boldsymbol{\lambda}_{N+1} = \boldsymbol{d}_{N+1}^{\mathsf{T}}\boldsymbol{\Lambda}_{1:N},}$$

where $\boldsymbol{\Lambda}_{1:N}$ are the multipliers of the LGR collocation equations, $\boldsymbol{W}$ is the diagonal matrix of Radau quadrature weights, and $\boldsymbol{d}_{N+1}$ is the final column of the LGR differentiation matrix.

Continuous Mayer Optimal-Control Problem

Consider the simplified Mayer problem

$$\min_{\boldsymbol{x}(\cdot),\boldsymbol{u}(\cdot)} \Phi\big(\boldsymbol{x}(1)\big),$$

subject to

$$\dot{\boldsymbol{x}}(\tau) = \boldsymbol{f}\big(\boldsymbol{x}(\tau), \boldsymbol{u}(\tau)\big), \qquad \tau \in [-1, 1],$$

and

$$\boldsymbol{x}(-1) = \boldsymbol{x}_0.$$

Because the objective contains no running cost, the Hamiltonian is

$$H(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\lambda}) = \boldsymbol{\lambda}^{\mathsf{T}} \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}).$$

Continuous First-Order Necessary Conditions

The continuous first-order conditions are:

Terminal Transversality

$$\boxed{\boldsymbol{\lambda}(1) = \nabla_{\boldsymbol{x}} \Phi\big(\boldsymbol{x}(1)\big).}$$

Initial Boundary Multiplier

If $\boldsymbol{\mu}$ is the multiplier associated with the fixed initial condition, then

$$\boxed{\boldsymbol{\lambda}(-1) = \boldsymbol{\mu}.}$$

Costate Equation

$$\boxed{\dot{\boldsymbol{\lambda}}(\tau) = -\nabla_{\boldsymbol{x}} \left[\boldsymbol{\lambda}(\tau)^{\mathsf{T}} \boldsymbol{f}\big(\boldsymbol{x}(\tau), \boldsymbol{u}(\tau)\big)\right].}$$

Stationarity with Respect to the Control

$$\boxed{\nabla_{\boldsymbol{u}} \left[\boldsymbol{\lambda}(\tau)^{\mathsf{T}} \boldsymbol{f}\big(\boldsymbol{x}(\tau), \boldsymbol{u}(\tau)\big)\right] = \boldsymbol{0}.}$$

The discrete LGR KKT conditions will be transformed into approximations of these equations.

LGR Grid

Let

$$\tau_1, \ldots, \tau_N$$

denote the N LGR points. For a left-Radau scheme,

$$\tau_1 = -1,$$

and

$$-1 = \tau_1 < \tau_2 < \cdots < \tau_N < 1.$$

The noncollocated terminal interpolation point is

$$\tau_{N+1} = 1.$$

Thus:

- the initial point is a collocation point;
- the final point is included in the state interpolation;
- the final point is not a collocation point.

State Approximation

Approximate the state by a polynomial of degree at most N:

$$\boxed{\boldsymbol{X}(\tau) = \sum_{i=1}^{N+1} \boldsymbol{X}_i L_i(\tau),}$$

where

$$L_i(\tau) = \prod_{\substack{j=1 \\ j \neq i}}^{N+1} \frac{\tau - \tau_j}{\tau_i - \tau_j}.$$

The state values are

$$\boldsymbol{X}_i \approx \boldsymbol{x}(\tau_i), \qquad i = 1, \ldots, N+1.$$

The derivative of the approximation is

$$\dot{\boldsymbol{X}}(\tau) = \sum_{i=1}^{N+1} \boldsymbol{X}_i L_i'(\tau).$$

LGR Differentiation Matrix

Define the LGR differentiation matrix by

$$D_{ji} = L_i'(\tau_j), \qquad j = 1, \ldots, N, \qquad i = 1, \ldots, N+1.$$

Therefore,

$$\boxed{\boldsymbol{D} \in \mathbb{R}^{N \times (N+1)}.}$$

Collocating at the N LGR points gives

$$\sum_{i=1}^{N+1} D_{ji} \boldsymbol{X}_i = \boldsymbol{f}(\boldsymbol{X}_j, \boldsymbol{U}_j), \qquad j = 1, \ldots, N.$$

In matrix form,

$$\boxed{\boldsymbol{D}\boldsymbol{X}_{1:N+1} = \boldsymbol{F}_{1:N},}$$

where

$$\boldsymbol{F}_j = \boldsymbol{f}(\boldsymbol{X}_j, \boldsymbol{U}_j).$$

LGR Nonlinear Program

The LGR transcription is

$$\boxed{\begin{aligned} \min_{\boldsymbol{X},\boldsymbol{U}} \quad & \Phi(\boldsymbol{X}_{N+1}) \\ \text{subject to} \quad & \boldsymbol{F}_{1:N} - \boldsymbol{D}\boldsymbol{X}_{1:N+1} = \boldsymbol{0}, \\ & \boldsymbol{x}_0 - \boldsymbol{X}_1 = \boldsymbol{0}. \end{aligned}}$$

Unlike the LG method, no separate quadrature equation is needed to recover the terminal state, because $\boldsymbol{X}_{N+1}$ is already an NLP variable.

Key point
The LG transcription requires a terminal quadrature constraint. The LGR transcription does not. This difference is why the LGR costate mapping is algebraically simpler.

LGR NLP Lagrangian

Let

$$\boldsymbol{\Lambda}_{1:N} = \begin{bmatrix} \boldsymbol{\Lambda}_1 \\ \vdots \\ \boldsymbol{\Lambda}_N \end{bmatrix}$$

be the multipliers associated with the N collocation equations.

Let

$$\boldsymbol{\mu}$$

be the multiplier associated with the initial condition.

The NLP Lagrangian is

$$\begin{aligned} \mathcal{L} = \Phi(\boldsymbol{X}_{N+1}) &+ \langle \boldsymbol{\Lambda}_{1:N}, \boldsymbol{F}_{1:N} - \boldsymbol{D}\boldsymbol{X}_{1:N+1} \rangle \\ &+ \left\langle \boldsymbol{\mu}, \boldsymbol{x}_0 - \boldsymbol{X}_1 \right\rangle . \end{aligned}$$

KKT Conditions for the LGR NLP

The NLP variables are

$$\boldsymbol{X}_1, \ldots, \boldsymbol{X}_{N+1}, \qquad \boldsymbol{U}_1, \ldots, \boldsymbol{U}_N.$$

State Stationarity for Interior LGR Points

For

$$j = 2, \ldots, N,$$

stationarity with respect to $\boldsymbol{X}_j$ gives

$$\boxed{\sum_{i=1}^{N} D_{ij} \boldsymbol{\Lambda}_i = \left(\nabla_{\boldsymbol{X}} \boldsymbol{f}(\boldsymbol{X}_j, \boldsymbol{U}_j)\right)^{\mathsf{T}} \boldsymbol{\Lambda}_j.}$$

In scalar shorthand,

$$\sum_{i=1}^{N} D_{ij} \Lambda_i = \Lambda_j \nabla_{\boldsymbol{X}} f(\boldsymbol{X}_j, \boldsymbol{U}_j).$$

State Stationarity at the Initial LGR Point

Because $\boldsymbol{X}_1$ also appears in the initial-condition constraint,

$$\boxed{\sum_{i=1}^{N} D_{i1} \boldsymbol{\Lambda}_i = \left(\nabla_{\boldsymbol{X}} \boldsymbol{f}(\boldsymbol{X}_1, \boldsymbol{U}_1)\right)^{\mathsf{T}} \boldsymbol{\Lambda}_1 - \boldsymbol{\mu}.}$$

Equivalently,

$$\sum_{i=1}^{N} D_{i1} \boldsymbol{\Lambda}_i + \boldsymbol{\mu} = \left(\nabla_{\boldsymbol{X}} \boldsymbol{f}(\boldsymbol{X}_1, \boldsymbol{U}_1)\right)^{\mathsf{T}} \boldsymbol{\Lambda}_1.$$

Control Stationarity

For

$$j = 1, \ldots, N,$$

stationarity with respect to $\boldsymbol{U}_j$ gives

$$\boxed{\left(\nabla_{\boldsymbol{U}} \boldsymbol{f}(\boldsymbol{X}_j, \boldsymbol{U}_j)\right)^{\mathsf{T}} \boldsymbol{\Lambda}_j = \boldsymbol{0}.}$$

Terminal-State Stationarity

Differentiation with respect to $\boldsymbol{X}_{N+1}$ gives

$$\boxed{\nabla_{\boldsymbol{X}} \Phi(\boldsymbol{X}_{N+1}) = \boldsymbol{d}_{N+1}^{\mathsf{T}} \boldsymbol{\Lambda}_{1:N},}$$

where

$$\boldsymbol{d}_{N+1}$$

is the final column of $\boldsymbol{D}$.
This equation provides the terminal costate estimate.

Definition of the LGR Adjoint Differentiation Matrix

Let

$$\boldsymbol{W} = \mathrm{diag}(w_1, \ldots, w_N),$$

where w_i are the LGR quadrature weights.
Define

$$\boldsymbol{D}^{\dagger} \in \mathbb{R}^{N \times N}.$$

A commonly used representation is

$$\boxed{D_{11}^{\dagger} = -D_{11} - \frac{1}{w_1},}$$

and, for the remaining entries,

$$\boxed{D_{ij}^{\dagger} = -\frac{w_j}{w_i} D_{ji}.}$$

This matrix plays the role of a differentiation matrix for the discrete costate approximation at the LGR points.

Interior Costate Transformation

Define the costate estimate at the LGR points by

$$\boxed{\boldsymbol{\lambda}_{1:N} = \boldsymbol{W}^{-1} \boldsymbol{\Lambda}_{1:N}.}$$

Componentwise,

$$\boxed{\boldsymbol{\lambda}_i = \frac{\boldsymbol{\Lambda}_i}{w_i}, \qquad i = 1, \ldots, N.}$$

This mapping is simpler than the LG mapping because there is no additional terminal quadrature multiplier.

Transformation of the Discrete Adjoint Equation

Substitute

$$\boldsymbol{\Lambda}_i = w_i \boldsymbol{\lambda}_i$$

into the KKT state equations.
For the interior points, the transformed equation becomes

$$\boxed{\sum_{j=1}^{N} D_{ij}^{\dagger} \boldsymbol{\lambda}_j = -\nabla_X \left[\boldsymbol{\lambda}_i^{\mathsf{T}} \boldsymbol{f}(\boldsymbol{X}_i, \boldsymbol{U}_i) \right], \qquad i = 2, \ldots, N.}$$

This is the discrete LGR approximation of

$$\dot{\boldsymbol{\lambda}} = -\nabla_x \left(\boldsymbol{\lambda}^{\mathsf{T}} \boldsymbol{f} \right).$$

At the first LGR point,

$$\boxed{\sum_{j=1}^{N} D_{1j}^{\dagger} \boldsymbol{\lambda}_j = -\nabla_X \left[\boldsymbol{\lambda}_1^{\mathsf{T}} \boldsymbol{f}(\boldsymbol{X}_1, \boldsymbol{U}_1) \right] + \frac{1}{w_1} \left(\boldsymbol{\mu} - \boldsymbol{\lambda}_1 \right).}$$

The additional term reflects the initial-condition multiplier.

Asymptotic Endpoint Consistency

The continuous boundary condition is

$$\boldsymbol{\lambda}(-1) = \boldsymbol{\mu}.$$

Because

$$\tau_1 = -1,$$

the first LGR costate estimate should approximate the initial costate:

$$\boldsymbol{\lambda}_1 \approx \boldsymbol{\mu}.$$

As the polynomial degree increases,

$$\boxed{\boldsymbol{\mu} - \boldsymbol{\lambda}_1 \to \mathbf{0}.}$$

Consequently, the correction term

$$\frac{1}{w_1} \left(\boldsymbol{\mu} - \boldsymbol{\lambda}_1 \right)$$

vanishes asymptotically.

Terminal Costate Recovery

The terminal-state KKT condition is

$$\nabla_{\boldsymbol{X}}\Phi(\boldsymbol{X}_{N+1}) = \boldsymbol{d}_{N+1}^{\mathsf{T}}\boldsymbol{\Lambda}_{1:N}.$$

From the continuous transversality condition,

$$\boldsymbol{\lambda}(1) = \nabla_{\boldsymbol{x}}\Phi(\boldsymbol{x}(1)).$$

Therefore, define

$$\boxed{\boldsymbol{\lambda}_{N+1} = \boldsymbol{d}_{N+1}^{\mathsf{T}}\boldsymbol{\Lambda}_{1:N}.}$$

This gives the costate estimate at the noncollocated final point.

Alternative Interpretation Through Quadrature

Since

$$\dot{\boldsymbol{\lambda}} = -\nabla_{\boldsymbol{x}}\left(\boldsymbol{\lambda}^{\mathsf{T}}\boldsymbol{f}\right),$$

integration gives

$$\boldsymbol{\lambda}(1) = \boldsymbol{\lambda}(-1) + \int_{-1}^{1}\dot{\boldsymbol{\lambda}}(\tau)\,\mathrm{d}\tau.$$

Using LGR quadrature,

$$\boldsymbol{\lambda}_{N+1} \approx \boldsymbol{\lambda}_1 + \sum_{i=1}^{N} w_i\left[-\nabla_{\boldsymbol{X}}\left(\boldsymbol{\lambda}_i^{\mathsf{T}}\boldsymbol{F}_i\right)\right].$$

When

$$\boldsymbol{\lambda}_1 \approx \boldsymbol{\mu},$$

this quadrature relation is consistent with the terminal formula obtained directly from the KKT conditions.

Discrete Control Stationarity

Using

$$\boldsymbol{\Lambda}_i = w_i\boldsymbol{\lambda}_i,$$

the control stationarity conditions become

$$(\nabla_{\boldsymbol{U}} \boldsymbol{f}(\boldsymbol{X}_i, \boldsymbol{U}_i))^{\mathsf{T}} w_i \boldsymbol{\lambda}_i = \boldsymbol{0}.$$

Since $w_i > 0$,

$$\boxed{\nabla_{\boldsymbol{U}} \left[\boldsymbol{\lambda}_i^{\mathsf{T}} \boldsymbol{f}(\boldsymbol{X}_i, \boldsymbol{U}_i)\right] = \boldsymbol{0}, \qquad i = 1, \ldots, N.}$$

This is the discrete equivalent of

$$\nabla_{\boldsymbol{u}} H = \boldsymbol{0}.$$

Complete LGR Costate Mapping Theorem

Key point
Let $\boldsymbol{\Lambda}_{1:N}$ denote the KKT multipliers associated with the LGR collocation equations. Let $\boldsymbol{W}$ be the diagonal matrix of LGR quadrature weights, and let $\boldsymbol{d}_{N+1}$ be the final column of the LGR differentiation matrix. Then

$$\boldsymbol{\lambda}_{1:N} = \boldsymbol{W}^{-1} \boldsymbol{\Lambda}_{1:N},$$

and

$$\boldsymbol{\lambda}_{N+1} = \boldsymbol{d}_{N+1}^{\mathsf{T}} \boldsymbol{\Lambda}_{1:N}.$$

The first point satisfies

$$\boldsymbol{\lambda}_1 \approx \boldsymbol{\mu},$$

with the discrepancy tending to zero as the approximation is refined.

Computational Algorithm

After solving the LGR NLP:

1. Extract the multipliers associated with the collocation constraints:

$$\boldsymbol{\Lambda}_{1:N}.$$

2. Form the diagonal weight matrix:

$$\boldsymbol{W} = \operatorname{diag}(w_1, \ldots, w_N).$$

3. Compute the costates at the LGR points:

$$\boldsymbol{\lambda}_{1:N} = \boldsymbol{W}^{-1}\boldsymbol{\Lambda}_{1:N}.$$

4. Extract the final differentiation-matrix column:

$$\boldsymbol{d}_{N+1} = \boldsymbol{D}(:, N+1).$$

5. Compute the terminal costate:

$$\boldsymbol{\lambda}_{N+1} = \boldsymbol{d}_{N+1}^{\mathsf{T}}\boldsymbol{\Lambda}_{1:N}.$$

6. Compare

$$\boldsymbol{\lambda}_1 \quad \text{and} \quad \boldsymbol{\mu}$$

as an endpoint-consistency check.

7. Check the discrete adjoint residual:

$$\boldsymbol{D}^{\dagger}\boldsymbol{\lambda}_{1:N} + \nabla_{\boldsymbol{X}} \left\langle \boldsymbol{\lambda}_{1:N}, \boldsymbol{F}_{1:N} \right\rangle.$$

8. Verify convergence under polynomial-order or mesh refinement.

Comparison with the LG Costate Mapping

Feature	LG method	LGR method
Collocation points	Interior Gauss points	Includes one endpoint
Initial point	Noncollocated interpolation point	Collocation point
Final point	Noncollocated and recovered by quadrature	Noncollocated state variable
Terminal quadrature constraint	Required	Not required
Interior costate mapping	$\boldsymbol{W}^{-1}\boldsymbol{\Lambda} + \mathbf{1}\boldsymbol{\Lambda}_{N+1}$	$\boldsymbol{W}^{-1}\boldsymbol{\Lambda}$
Terminal costate	Terminal quadrature multiplier	Final column of $\boldsymbol{D}$ times collocation multipliers
Algebraic complexity	Higher	Lower

Comparison of LG and LGR costate estimation.

Sign Conventions

The formulas above assume the collocation constraints are written as

$$\boldsymbol{F} - \boldsymbol{DX} = \boldsymbol{0}.$$

If they are instead written as

$$\boldsymbol{DX} - \boldsymbol{F} = \boldsymbol{0},$$

the reported KKT multipliers may have the opposite sign. Therefore, the implementation must be consistent with the exact NLP constraint convention.

Scaling and Multiplier Accuracy

Accurate primal states do not automatically guarantee accurate dual variables.

Potential sources of poor multiplier accuracy include:

- badly scaled state variables;
- badly scaled dynamic defects;
- loose NLP tolerances;
- insufficient polynomial degree;
- poor mesh placement;
- inconsistent derivative information;
- active path constraints near switching points.

A reliable implementation should check both primal and dual convergence.

Interpretation of the Costate

The costate can be interpreted as a sensitivity of the optimal value to perturbations of the state. In the LGR transcription, the mapped KKT multipliers provide a discrete approximation of this sensitivity along the trajectory.

Costate estimates are useful for:

- verifying first-order necessary conditions;
- interpreting state importance;
- identifying switching structure;
- comparing direct and indirect methods;
- constructing neighboring optimal-control laws;
- sensitivity analysis;
- mesh refinement diagnostics.

Summary

1. The LGR state polynomial uses the N LGR points and the final interpolation point.
2. The LGR NLP contains collocation constraints and an initial-condition constraint.

3. No terminal quadrature constraint is required.
4. The KKT multipliers of the collocation equations are scaled by the inverse Radau weights.
5. The interior costate mapping is

$$\boldsymbol{\lambda}_{1:N} = \boldsymbol{W}^{-1}\boldsymbol{\Lambda}_{1:N}.$$

6. The terminal costate is recovered from

$$\boldsymbol{\lambda}_{N+1} = \boldsymbol{d}_{N+1}^{\mathsf{T}}\boldsymbol{\Lambda}_{1:N}.$$

7. The initial costate estimate satisfies

$$\boldsymbol{\lambda}_1 \approx \boldsymbol{\mu}.$$

8. The transformed KKT equations reproduce the continuous adjoint and control-stationarity conditions in discrete form.

Connection. Completing the LGL case makes it possible to compare all three Legendre schemes and then restore the full Bolza, endpoint, and path-constraint terms.

LGL Costate Estimation and General Pseudospectral Transcription

This section completes the treatment of costate estimation for the three principal Legendre pseudospectral schemes:

1. Legendre–Gauss (LG),
2. Legendre–Gauss–Radau (LGR), and
3. Legendre–Gauss–Lobatto (LGL).

The first part derives the LGL costate equations from the KKT conditions of the LGL nonlinear program. The second part compares the LG, LGR, and LGL costate mappings. The final part extends the pseudospectral transcription from the simplified Mayer problem on $[-1, 1]$ to a general Bolza problem defined on $[t_0, t_f]$ with endpoint constraints and path constraints.

Legendre–Gauss–Lobatto Grid

Let

$$\tau_1, \ldots, \tau_N$$

be the LGL points. Unlike LG and LGR grids, the LGL grid includes both endpoints:

$$\boxed{\tau_1 = -1, \qquad \tau_N = 1.}$$

The interior points are roots of the derivative of an appropriate Legendre polynomial.

LGL State Approximation

Approximate the state by

$$\boxed{\boldsymbol{X}(\tau) = \sum_{i=1}^{N} \boldsymbol{X}_i L_i(\tau),}$$

where

$$L_i(\tau) = \prod_{\substack{j=1 \\ j \neq i}}^{N} \frac{\tau - \tau_j}{\tau_i - \tau_j}.$$

This interpolation polynomial has degree at most $N - 1$.
Its derivative is

$$\dot{\boldsymbol{X}}(\tau) = \sum_{i=1}^{N} \boldsymbol{X}_i L_i'(\tau),$$

which has degree at most $N - 2$.

LGL Differentiation Matrix

Define

$$D_{ji} = L_i'(\tau_j), \qquad i, j = 1, \ldots, N.$$

Thus,

$$\boxed{\boldsymbol{D} \in \mathbb{R}^{N \times N}.}$$

Collocating the dynamics at all LGL points gives

$$\sum_{i=1}^{N} D_{ji} \boldsymbol{X}_i = \boldsymbol{f}(\boldsymbol{X}_j, \boldsymbol{U}_j), \qquad j = 1, \ldots, N.$$

In matrix form,

$$\boxed{\boldsymbol{D}\boldsymbol{X}_{1:N} = \boldsymbol{F}_{1:N}.}$$

Over-Collocation and Singularity

The derivative polynomial has degree at most $N - 2$ and is therefore determined by $N - 1$ independent coefficients. However, it is collocated at N points. This creates a redundancy.

As a result,

$$\boxed{\boldsymbol{D} \text{ is singular.}}$$

This fact has important consequences for costate estimation and for the relationship between differential and integral pseudospectral formulations.

Simplified Mayer Problem

Consider

$$\min_{\boldsymbol{x}(\cdot), \boldsymbol{u}(\cdot)} \Phi\big(\boldsymbol{x}(1)\big),$$

subject to

$$\dot{\boldsymbol{x}}(\tau) = \boldsymbol{f}\big(\boldsymbol{x}(\tau), \boldsymbol{u}(\tau)\big), \qquad \tau \in [-1, 1],$$

and

$$\boldsymbol{x}(-1) = \boldsymbol{x}_0.$$

The Hamiltonian is

$$H(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\lambda}) = \boldsymbol{\lambda}^{\mathsf{T}} \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}).$$

The continuous optimality conditions are

$$\boldsymbol{\lambda}(1) = \nabla_{\boldsymbol{x}} \Phi\big(\boldsymbol{x}(1)\big),$$

$$\dot{\boldsymbol{\lambda}} = -\nabla_{\boldsymbol{x}} \left(\boldsymbol{\lambda}^{\mathsf{T}} \boldsymbol{f}\right),$$

and

$$\nabla_{\boldsymbol{u}} \left(\boldsymbol{\lambda}^{\mathsf{T}} \boldsymbol{f}\right) = \boldsymbol{0}.$$

LGL Nonlinear Program

The LGL transcription is

$$\boxed{\begin{array}{ll} \min\limits_{\boldsymbol{X}, \boldsymbol{U}} & \Phi(\boldsymbol{X}_N) \\ \text{subject to} & \boldsymbol{F}_{1:N} - \boldsymbol{D}\boldsymbol{X}_{1:N} = \boldsymbol{0}, \\ & \boldsymbol{x}_0 - \boldsymbol{X}_1 = \boldsymbol{0}. \end{array}}$$

The NLP variables are

$$\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N, \qquad \boldsymbol{U}_1, \ldots, \boldsymbol{U}_N.$$

LGL NLP Lagrangian

Let

$$\boldsymbol{\Lambda}_{1:N} = \begin{bmatrix} \boldsymbol{\Lambda}_1 \\ \vdots \\ \boldsymbol{\Lambda}_N \end{bmatrix}$$

denote the multipliers associated with the collocation equations.
Let

$$\boldsymbol{\mu}$$

denote the multiplier associated with the initial condition.
The NLP Lagrangian is

$$\begin{aligned} \mathcal{L} = \Phi(\boldsymbol{X}_N) &+ \langle \boldsymbol{\Lambda}_{1:N}, \boldsymbol{F}_{1:N} - \boldsymbol{D}\boldsymbol{X}_{1:N} \rangle \\ &+ \left\langle \boldsymbol{\mu}, \boldsymbol{x}_0 - \boldsymbol{X}_1 \right\rangle. \end{aligned}$$

KKT Conditions

Because both endpoints are collocation points, the initial and terminal state equations must be treated separately from the interior equations.

Initial Point

Stationarity with respect to $\boldsymbol{X}_1$ gives

$$\boxed{\nabla_X \left[\boldsymbol{\Lambda}_1^\mathsf{T} \boldsymbol{f}(\boldsymbol{X}_1, \boldsymbol{U}_1)\right] - \boldsymbol{d}_1^\mathsf{T} \boldsymbol{\Lambda}_{1:N} = \boldsymbol{\mu},}$$

where $\boldsymbol{d}_1$ is the first column of $\boldsymbol{D}$.

Terminal Point

Stationarity with respect to $\boldsymbol{X}_N$ gives

$$\boxed{\boldsymbol{d}_N^\mathsf{T} \boldsymbol{\Lambda}_{1:N} - \nabla_X \left[\boldsymbol{\Lambda}_N^\mathsf{T} \boldsymbol{f}(\boldsymbol{X}_N, \boldsymbol{U}_N)\right] = \nabla_X \Phi(\boldsymbol{X}_N).}$$

Interior Points

For

$$j = 2, \ldots, N-1,$$

stationarity gives

$$\boxed{\nabla_X \left[\boldsymbol{\Lambda}_j^\mathsf{T} \boldsymbol{f}(\boldsymbol{X}_j, \boldsymbol{U}_j)\right] = \boldsymbol{d}_j^\mathsf{T} \boldsymbol{\Lambda}_{1:N},}$$

where $\boldsymbol{d}_j$ is the jth column of $\boldsymbol{D}$.

Control Stationarity

For all LGL points,

$$\boxed{\nabla_U \langle \boldsymbol{\Lambda}_{1:N}, \boldsymbol{F}_{1:N} \rangle = \boldsymbol{0}.}$$

LGL Costate Mapping

Let

$$\boldsymbol{W} = \mathrm{diag}(w_1, \ldots, w_N),$$

where w_i are the LGL quadrature weights.

Define the costate estimate by

$$\boxed{\boldsymbol{\lambda}_{1:N} = \boldsymbol{W}^{-1}\boldsymbol{\Lambda}_{1:N}.}$$

Componentwise,

$$\boldsymbol{\lambda}_i = \frac{\boldsymbol{\Lambda}_i}{w_i}.$$

This is algebraically the simplest of the three mappings.

LGL Adjoint Differentiation Matrix

Define

$$\boldsymbol{D}^\dagger \in \mathbb{R}^{N\times N}$$

such that

$$D^\dagger_{ii} = D_{ii}$$

for interior diagonal entries, while the endpoint diagonal entries are modified:

$$\boxed{D^\dagger_{11} = -D_{11} - \frac{1}{w_1},}$$

and

$$\boxed{D^\dagger_{NN} = -D_{NN} + \frac{1}{w_N}.}$$

For

$$i \neq j,$$

define

$$\boxed{D^\dagger_{ij} = -\frac{w_j}{w_i} D_{ji}.}$$

Compact Discrete Adjoint Equation

Let

$$\boldsymbol{e}_1 = \begin{bmatrix} 1 & 0 & \cdots & 0 \end{bmatrix}^\mathsf{T}, \qquad \boldsymbol{e}_N = \begin{bmatrix} 0 & \cdots & 0 & 1 \end{bmatrix}^\mathsf{T}.$$

Then the transformed KKT system can be written as

$$\boxed{\begin{aligned} \boldsymbol{D}^{\dagger}\boldsymbol{\lambda}_{1:N} = & -\nabla_{\boldsymbol{X}} \left\langle \boldsymbol{\lambda}_{1:N}, \boldsymbol{F}_{1:N} \right\rangle \\ & + \frac{\boldsymbol{e}_1}{w_1} \left(\boldsymbol{\mu} - \boldsymbol{\lambda}_1\right) + \frac{\boldsymbol{e}_N}{w_N} \left[\boldsymbol{\lambda}_N - \nabla_{\boldsymbol{X}} \Phi(\boldsymbol{X}_N)\right]. \end{aligned}}$$

The control stationarity equation becomes

$$\boxed{\nabla_{\boldsymbol{U}} \left\langle \boldsymbol{\lambda}_{1:N}, \boldsymbol{F}_{1:N} \right\rangle = \boldsymbol{0}.}$$

Coupling of Boundary Conditions and Costate Dynamics

For a pure discrete adjoint equation, one would prefer

$$\boldsymbol{D}^{\dagger}\boldsymbol{\lambda}_{1:N} = -\nabla_{\boldsymbol{X}} \left\langle \boldsymbol{\lambda}_{1:N}, \boldsymbol{F}_{1:N} \right\rangle .$$

Instead, the LGL equation contains two additional endpoint terms:

$$\frac{\boldsymbol{e}_1}{w_1} \left(\boldsymbol{\mu} - \boldsymbol{\lambda}_1\right)$$

and

$$\frac{\boldsymbol{e}_N}{w_N} \left[\boldsymbol{\lambda}_N - \nabla_{\boldsymbol{X}} \Phi(\boldsymbol{X}_N)\right].$$

Therefore,

> **Key point**
> In LGL collocation, the endpoint transversality conditions remain coupled to the discrete approximation of the costate dynamics.

This differs from LG, where both endpoint conditions are decoupled, and LGR, where only one endpoint requires special treatment.

Why LGL Costate Estimates Can Be Less Accurate

The LGL differentiation matrix is square and singular. Consequently:

- there is no ordinary inverse of $\boldsymbol{D}$;
- the differential form has no uniquely equivalent integration matrix;
- a null-space component can enter the discrete adjoint representation;
- endpoint residuals remain embedded in the costate dynamics;
- the resulting costate estimate may be less accurate than LG or LGR estimates.

The primal state and control solution may still be accurate. The limitation is specifically associated with recovering the continuous adjoint from the NLP multipliers.

Summary of Costate Mappings

LG Collocation

Let $\boldsymbol{\Lambda}_{1:N}$ be the collocation multipliers and let $\boldsymbol{\Lambda}_{N+1}$ be the multiplier of the terminal quadrature equation. The costate at the Gauss points is

$$\boxed{\boldsymbol{\lambda}_{1:N} = \boldsymbol{W}^{-1}\boldsymbol{\Lambda}_{1:N} + \mathbf{1}\boldsymbol{\Lambda}_{N+1}.}$$

The final costate is

$$\boxed{\boldsymbol{\lambda}_{N+1} = \boldsymbol{\Lambda}_{N+1}.}$$

The initial costate is recovered from

$$\boxed{\boldsymbol{\lambda}_0 = \boldsymbol{\lambda}_{N+1} - \boldsymbol{d}_0^{\mathsf{T}}\boldsymbol{\Lambda}_{1:N},}$$

with the precise column notation chosen consistently with the LG differentiation matrix.

LGR Collocation

The costate at the LGR points is

$$\boxed{\boldsymbol{\lambda}_{1:N} = \boldsymbol{W}^{-1}\boldsymbol{\Lambda}_{1:N}.}$$

The terminal costate is

$$\boxed{\boldsymbol{\lambda}_{N+1} = \boldsymbol{d}_{N+1}^{\mathsf{T}}\boldsymbol{\Lambda}_{1:N}.}$$

For a left-Radau grid,

$$\tau_1 = -1,$$

so $\boldsymbol{\lambda}_1$ directly approximates the initial costate.

LGL Collocation

The costate at all LGL points, including both endpoints, is

$$\boxed{\boldsymbol{\lambda}_{1:N} = \boldsymbol{W}^{-1}\boldsymbol{\Lambda}_{1:N}.}$$

However, the endpoint transversality residuals remain coupled to the discrete adjoint dynamics.

Comparison of LG, LGR, and LGL

Feature	LG	LGR	LGL
Included endpoints	Neither	One	Both
Differentiation matrix	Rectangular	Rectangular	Square
Matrix singularity	No relevant square singularity	No relevant square singularity	Singular
Equivalent integral form	Yes	Yes	Not uniquely invertible
Endpoint coupling	Decoupled	One endpoint partly coupled	Both endpoints coupled
Interior mapping	$\boldsymbol{W}^{-1}\boldsymbol{\Lambda} + \mathbf{1}\boldsymbol{\Lambda}_{N+1}$	$\boldsymbol{W}^{-1}\boldsymbol{\Lambda}$	$\boldsymbol{W}^{-1}\boldsymbol{\Lambda}$
Typical costate quality	High	High	Can be less accurate

Comparison of Legendre pseudospectral costate estimation methods.

General Bolza Optimal-Control Problem

Consider

$$\min_{\boldsymbol{x}(\cdot),\boldsymbol{u}(\cdot),t_0,t_f} J = \Phi\left(\boldsymbol{x}(t_0), t_0, \boldsymbol{x}(t_f), t_f\right) + \int_{t_0}^{t_f} L\left(\boldsymbol{x}(t), \boldsymbol{u}(t), t\right)\, \mathrm{d}t.$$

The dynamics are

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}\left(\boldsymbol{x}(t), \boldsymbol{u}(t), t\right).$$

Endpoint constraints may be written as

$$\boldsymbol{\phi}\left(\boldsymbol{x}(t_0), t_0, \boldsymbol{x}(t_f), t_f\right) = \mathbf{0}.$$

Path constraints may be written as

$$\boldsymbol{c}_{\min} \leq \boldsymbol{c}\left(\boldsymbol{x}(t), \boldsymbol{u}(t), t\right) \leq \boldsymbol{c}_{\max}.$$

Mapping the Time Interval to $[-1, 1]$

Define

$$\boxed{\tau = \frac{2(t - t_0)}{t_f - t_0} - 1.}$$

The inverse transformation is

$$\boxed{t = \frac{t_f - t_0}{2}\tau + \frac{t_f + t_0}{2}.}$$

Differentiation gives

$$\frac{\mathrm{d}\tau}{\mathrm{d}t} = \frac{2}{t_f - t_0},$$

and

$$\boxed{\frac{\mathrm{d}t}{\mathrm{d}\tau} = \frac{t_f - t_0}{2}.}$$

Transformed Bolza Problem

The endpoint states map as

$$\boldsymbol{x}(t_0) = \boldsymbol{x}(-1), \qquad \boldsymbol{x}(t_f) = \boldsymbol{x}(1).$$

The cost becomes

$$\begin{aligned} J = &\Phi\left(\boldsymbol{x}(-1), t_0, \boldsymbol{x}(1), t_f\right) \\ &+ \frac{t_f - t_0}{2}\int_{-1}^{1} L\left(\boldsymbol{x}(\tau), \boldsymbol{u}(\tau), t(\tau; t_0, t_f)\right)\,\mathrm{d}\tau. \end{aligned}$$

The transformed dynamics are

$$\boxed{\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}\tau} = \frac{t_f - t_0}{2}\boldsymbol{f}\left(\boldsymbol{x}(\tau), \boldsymbol{u}(\tau), t(\tau; t_0, t_f)\right).}$$

The endpoint constraints become

$$\boldsymbol{\phi}\left(\boldsymbol{x}(-1), t_0, \boldsymbol{x}(1), t_f\right) = \boldsymbol{0}.$$

The path constraints become

$$\boldsymbol{c}_{\min} \leq \boldsymbol{c}\left(\boldsymbol{x}(\tau), \boldsymbol{u}(\tau), t(\tau; t_0, t_f)\right) \leq \boldsymbol{c}_{\max}.$$

LGR Discretization of the General Bolza Problem

For a left-LGR grid, the initial point is $\tau_1 = -1$ and the final interpolation point is $\tau_{N+1} = 1$.

Discrete Objective

The objective is approximated by

$$J_N = \Phi\left(\boldsymbol{X}_1, t_0, \boldsymbol{X}_{N+1}, t_f\right) + \frac{t_f - t_0}{2} \sum_{i=1}^{N} w_i L\left(\boldsymbol{X}_i, \boldsymbol{U}_i, t_i\right),$$

where

$$t_i = \frac{t_f - t_0}{2}\tau_i + \frac{t_f + t_0}{2}.$$

Discrete Dynamics

The collocation equations are

$$\boxed{\boldsymbol{D}\boldsymbol{X}_{1:N+1} = \frac{t_f - t_0}{2}\boldsymbol{F}_{1:N}.}$$

Equivalently,

$$\boldsymbol{D}\boldsymbol{X}_{1:N+1} - \frac{t_f - t_0}{2}\boldsymbol{F}_{1:N} = \boldsymbol{0}.$$

Endpoint Constraints

The endpoint constraints become

$$\boxed{\boldsymbol{\phi}\left(\boldsymbol{X}_1, t_0, \boldsymbol{X}_{N+1}, t_f\right) = \boldsymbol{0}.}$$

Path Constraints

At each collocation point,

$$\boxed{\boldsymbol{c}_{\min} \le \boldsymbol{c}\left(\boldsymbol{X}_i, \boldsymbol{U}_i, t_i\right) \le \boldsymbol{c}_{\max}, \qquad i = 1, \ldots, N.}$$

Interpretation of the Time-Scaling Factor

The only structural change in the dynamics and integral cost is the factor

$$\boxed{\frac{t_f - t_0}{2}.}$$

When

$$t_0 = -1, \qquad t_f = 1,$$

this factor equals one. Thus, the simplified problem used in the derivations is a special case of the general transcription.

The initial and final times may themselves be NLP variables. In that case, they enter:

- the endpoint cost;
- the quadrature scaling;
- the transformed dynamics;
- the physical time values t_i;
- the endpoint constraints;
- the path constraints.

Extension to LG and LGL

The same time transformation applies to LG and LGL schemes.

LG

The LG transcription additionally requires a quadrature equation for the terminal state because neither endpoint is a collocation point.

LGR

The LGR transcription requires one noncollocated endpoint state and has a rectangular differentiation matrix.

LGL

The LGL transcription includes both endpoint states directly and has a square singular differentiation matrix.

Differential and Integral Interpretations

For LG and LGR, a suitable square portion of the rectangular differentiation matrix is invertible. Its inverse acts as an integration matrix. Consequently, their differential collocation forms have equivalent integral forms.

For LGL, the differentiation matrix is singular. Therefore, no unique ordinary inverse exists, and the differential form does not possess the same uniquely equivalent integration representation.

This distinction explains much of the difference in costate recovery quality.

Implementation Considerations

A practical implementation should carefully manage:

1. the ordering of state, control, time, and parameter variables;
2. the ordering of defect, endpoint, and path constraints;
3. the sign convention used for equality constraints;
4. the association between solver multipliers and constraint blocks;
5. state, control, defect, and objective scaling;

6. exact or algorithmic derivatives;
7. mesh refinement;
8. endpoint consistency;
9. primal and dual residuals.

Summary

1. LGL collocation includes both endpoints.
2. Its state differentiation matrix is square and singular.
3. The LGL costate estimate is obtained by

$$\boldsymbol{\lambda}_{1:N} = \boldsymbol{W}^{-1}\boldsymbol{\Lambda}_{1:N}.$$

4. The transformed adjoint equation contains both initial and terminal boundary residuals.
5. These endpoint terms couple the transversality conditions to the costate dynamics.
6. LG and LGR generally provide cleaner costate mappings because their rectangular differentiation matrices admit equivalent integration forms.
7. A general Bolza problem on $[t_0, t_f]$ is mapped to $[-1, 1]$ by an affine time transformation.
8. The transformed cost and dynamics acquire the factor

$$(t_f - t_0)/2.$$

9. Endpoint and path constraints are imposed directly in the resulting NLP.

Connection. With the mathematical transcription complete, attention shifts to the data layout and function interfaces required by a general NLP solver.

Practical NLP Implementation of LGR Optimal Control

The previous sessions developed the mathematical foundations of Legendre pseudospectral transcription and costate recovery. This section explains how to implement an LGR transcription inside a general nonlinear programming solver.

The central practical issue is that an optimal-control transcription is naturally expressed using matrices of state and control values, whereas an NLP solver expects a single decision vector. Therefore, the implementation must provide a systematic mapping between:

1. the continuous optimal-control problem;
2. the LGR-discretized state and control arrays;
3. the NLP decision vector;
4. the scalar objective function;
5. the stacked equality and inequality constraints; and
6. the initial guess supplied to the solver.

Standard Nonlinear Programming Form

A general nonlinear program may be written as

$$\begin{array}{ll} \min\limits_{z\in\mathbb{R}^{n_z}} & f(z) \\ \text{subject to} & g(z) = 0, \\ & h_{\min} \le h(z) \le h_{\max}, \\ & z_{\min} \le z \le z_{\max}. \end{array}$$

Here:

- z is the NLP decision vector;
- $f(z)$ is the scalar objective;
- $g(z)$ contains equality constraints;
- $h(z)$ contains inequality constraints; and
- $z_{\min}$ and $z_{\max}$ are variable bounds.

For implementation, the equality and inequality constraints may be stacked into one vector,

$$k(z) = \begin{bmatrix} g(z) \\ h(z) \end{bmatrix},$$

with corresponding lower and upper bounds.

LGR-Discretized Bolza Problem

Consider a Bolza problem on $[t_0, t_f]$:

$$J = \Phi\left(\boldsymbol{x}(t_0), t_0, \boldsymbol{x}(t_f), t_f\right) + \int_{t_0}^{t_f} L\left(\boldsymbol{x}(t), \boldsymbol{u}(t), t\right)\, \mathrm{d}t.$$

After mapping the interval to $[-1, 1]$ and applying LGR collocation, the objective is approximated by

$$J_N = \Phi\left(\boldsymbol{X}_1, t_0, \boldsymbol{X}_{N+1}, t_f\right) + \frac{t_f - t_0}{2} \sum_{i=1}^{N} w_i L\left(\boldsymbol{X}_i, \boldsymbol{U}_i, t_i\right).$$

The defect equations are

$$\boxed{\boldsymbol{D}\boldsymbol{X}_{1:N+1} - \frac{t_f - t_0}{2}\boldsymbol{F}_{1:N} = \boldsymbol{0}.}$$

Endpoint constraints are

$$\boldsymbol{\phi}\left(\boldsymbol{X}_1, t_0, \boldsymbol{X}_{N+1}, t_f\right) = \boldsymbol{0}.$$

Path constraints are imposed at the N LGR points:

$$\boldsymbol{c}_{\min} \le \boldsymbol{c}\left(\boldsymbol{X}_i, \boldsymbol{U}_i, t_i\right) \le \boldsymbol{c}_{\max}, \qquad i = 1, \ldots, N.$$

Dimensions of the Discrete Variables

Let:

- n_x be the number of states;
- n_u be the number of controls;
- N be the number of LGR collocation points.

Then

$$\boldsymbol{X} \in \mathbb{R}^{(N+1)\times n_x},$$

because the state is stored at the N LGR points plus the noncollocated terminal point.
Similarly,

$$\boldsymbol{U} \in \mathbb{R}^{N\times n_u},$$

because the control is represented only at the collocation points.
If both t_0 and t_f are optimization variables, the total number of NLP variables is

$$\boxed{n_z = (N+1)n_x + Nn_u + 2.}$$

Packing the NLP Decision Vector

The NLP solver expects a single column vector. A standard column-major packing is

$$\boxed{z = \begin{bmatrix} \mathrm{vec}(\boldsymbol{X}) \\ \mathrm{vec}(\boldsymbol{U}) \\ t_0 \\ t_f \end{bmatrix}.}$$

The state block is

$$z_x = \mathrm{vec}(\boldsymbol{X}) \in \mathbb{R}^{(N+1)n_x},$$

and the control block is

$$z_u = \mathrm{vec}(\boldsymbol{U}) \in \mathbb{R}^{Nn_u}.$$

The time variables occupy the final two entries.

Index Ranges

The state portion is

$$z_x = z\,(1 : (N+1)n_x)\,.$$

The control portion is

$$z_u = z\,((N+1)n_x + 1 : (N+1)n_x + Nn_u)\,.$$

The time variables are

$$t_0 = z_{(N+1)n_x + Nn_u + 1},$$

and

$$t_f = z_{(N+1)n_x + Nn_u + 2}.$$

Unpacking and Reshaping

Inside the objective and constraint functions, reshape the vector blocks back into matrices:

$$\boxed{\boldsymbol{X} = \mathrm{reshape}\,(z_x, N+1, n_x)\,,}$$

and

$$\boxed{\boldsymbol{U} = \text{reshape}\,(z_u, N, n_u)\,.}$$

In MATLAB:

```
X = reshape(zx, N+1, nx);
U = reshape(zu, N,   nu);
```

The state values at the collocation points are

$$\boldsymbol{X}_{\text{LGR}} = \boldsymbol{X}(1:N,:).$$

The endpoint state values are

$$\boldsymbol{X}_{\text{init}} = \boldsymbol{X}(1,:), \qquad \boldsymbol{X}_{\text{final}} = \boldsymbol{X}(N+1,:).$$

Physical Time at the LGR Points

The affine time transformation gives

$$t_i = \frac{t_f - t_0}{2}\tau_i + \frac{t_f + t_0}{2}.$$

In vector form,

$$\boxed{\mathbf{t} = \frac{t_f - t_0}{2}\boldsymbol{\tau} + \frac{t_f + t_0}{2}\mathbf{1}.}$$

These time values are used to evaluate the dynamics, Lagrange integrand, and path constraints.

Evaluating the Dynamics

Evaluate the dynamics at the LGR points:

$$\boldsymbol{F}_i = \boldsymbol{f}(\boldsymbol{X}_i, \boldsymbol{U}_i, t_i), \qquad i = 1, \ldots, N.$$

Stacking these rows produces

$$\boldsymbol{F}_{1:N} \in \mathbb{R}^{N \times n_x}.$$

The dynamics evaluation is problem dependent.
A MATLAB-style implementation is:

```
Xlgr = X(1:N,:);
F = dynamics(Xlgr, U, t);
```

Defect Constraints

The LGR defect matrix is

$$\boxed{\boldsymbol{\Delta} = \boldsymbol{D}\boldsymbol{X} - \frac{t_f - t_0}{2}\boldsymbol{F}.}$$

Its dimension is

$$\boldsymbol{\Delta} \in \mathbb{R}^{N \times n_x}.$$

An NLP solver requires a vector, so vectorize the matrix:

$$\boxed{\boldsymbol{\delta} = \mathrm{vec}(\boldsymbol{\Delta}) \in \mathbb{R}^{N n_x}.}$$

In MATLAB:

```
defect = D*X - 0.5*(tf -t0)*F;
defect_col = defect(:);
```

The expression

```
A(:)
```

stacks the columns of a matrix into one column vector.

Evaluating the Objective

Separate the objective into Mayer and Lagrange terms.

Mayer Term

Evaluate

$$J_M = \Phi\left(\boldsymbol{X}_{\text{init}}, t_0, \boldsymbol{X}_{\text{final}}, t_f\right).$$

Lagrange Term

Evaluate the integrand at all LGR points:

$$\ell_i = L(\boldsymbol{X}_i, \boldsymbol{U}_i, t_i).$$

Let

$$\boldsymbol{\ell} = \begin{bmatrix} \ell_1 & \cdots & \ell_N \end{bmatrix}^{\mathsf{T}}.$$

Then

$$J_L = \frac{t_f - t_0}{2} \boldsymbol{w}^\top \boldsymbol{\ell}.$$

The total cost is

$$\boxed{J_N = J_M + J_L.}$$

A MATLAB-style implementation is:

```
mayer = endpointCost(X(1,:), t0, X(end,:), tf);
ell   = runningCost(X(1:N,:), U, t);
cost  = mayer + 0.5*(tf -t0)*(w.'*ell);
```

Boundary Constraints

Suppose the endpoint constraints are described by

$$\boldsymbol{\phi}\left(\boldsymbol{X}_{\text{init}}, t_0, \boldsymbol{X}_{\text{final}}, t_f\right).$$

Evaluate

$$\boxed{\mathbf{b} = \boldsymbol{\phi}\left(\boldsymbol{X}(1,:), t_0, \boldsymbol{X}(N+1,:), t_f\right).}$$

These constraints are usually already returned as a vector and do not require additional reshaping.

Path Constraints

Suppose there are n_c path-constraint functions. Evaluating them at all N LGR points produces

$$\boldsymbol{C} \in \mathbb{R}^{N \times n_c}.$$

The matrix is vectorized as

$$\boxed{\boldsymbol{c}_{\text{col}} = \text{vec}(\boldsymbol{C}) \in \mathbb{R}^{N n_c}.}$$

In MATLAB:

```
C = pathConstraints(X(1:N,:), U, t);
Ccol = C(:);
```

Stacking the Complete Constraint Vector

The full NLP constraint vector can be assembled as

$$\boxed{k(z) = \begin{bmatrix} \delta \\ c_{\mathrm{col}} \\ \mathbf{b} \end{bmatrix}.}$$

The corresponding lower and upper bounds are stacked in the same order:

$$k_{\min} = \begin{bmatrix} \mathbf{0} \\ c_{\min,\mathrm{col}} \\ \mathbf{b}_{\min} \end{bmatrix},$$

and

$$k_{\max} = \begin{bmatrix} \mathbf{0} \\ c_{\max,\mathrm{col}} \\ \mathbf{b}_{\max} \end{bmatrix}.$$

This arrangement is convenient for solvers such as SNOPT that permit general lower and upper bounds on constraint functions.

Objective Function Template

A practical MATLAB-style objective function is:

```
function J = objective(z, data)

N  = data.N;
nx = data.nx;
nu = data.nu;
w  = data.w;
tau = data.tau;

nX = (N+1)*nx;
nU = N*nu;

zx = z(1:nX);
zu = z(nX+1:nX+nU);
t0 = z(nX+nU+1);
tf = z(nX+nU+2);

X = reshape(zx, N+1, nx);
U = reshape(zu, N, nu);

t = 0.5*(tf -t0)*tau + 0.5*(tf+t0);

JM = endpointCost(X(1,:), t0, X(end,:), tf);
L  = runningCost(X(1:N,:), U, t);

J = JM + 0.5*(tf -t0)*(w.'*L);
end
```

Constraint Function Template

A practical constraint function is:

```
function K = constraints(z, data)

N  = data.N;
nx = data.nx;
nu = data.nu;
D  = data.D;
tau = data.tau;

nX = (N+1)*nx;
nU = N*nu;

zx = z(1:nX);
zu = z(nX+1:nX+nU);
t0 = z(nX+nU+1);
tf = z(nX+nU+2);

X = reshape(zx, N+1, nx);
U = reshape(zu, N, nu);

Xlgr = X(1:N,:);
t = 0.5*(tf -t0)*tau + 0.5*(tf+t0);

F = dynamics(Xlgr, U, t);

defect = D*X - 0.5*(tf -t0)*F;
defect_col = defect(:);

BC = endpointConstraints(X(1,:), t0, X(end,:), tf);

C = pathConstraints(Xlgr, U, t);
Ccol = C(:);

K = [defect_col; Ccol; BC];
end
```

Solver-Specific Treatment of Inequalities

Different NLP solvers accept constraints in different formats.

Bounded Constraint Form

Some solvers allow

$$k_{\min} \le k(z) \le k_{\max}.$$

This is natural for path constraints of the form

$$\boldsymbol{c}_{\min} \le \boldsymbol{c}(\boldsymbol{z}) \le \boldsymbol{c}_{\max}.$$

MATLAB `fmincon` Form

The standard `fmincon` interface uses

$$\mathbf{c}(\boldsymbol{z}) \le \mathbf{0}, \qquad \mathbf{c}_{\mathrm{eq}}(\boldsymbol{z}) = \mathbf{0}.$$

A two-sided bound

$$\boldsymbol{c}_{\min} \le \boldsymbol{c}(\boldsymbol{z}) \le \boldsymbol{c}_{\max}$$

must be converted to

$$\boldsymbol{c}(\boldsymbol{z}) - \boldsymbol{c}_{\max} \le \mathbf{0},$$

and

$$\boldsymbol{c}_{\min} - \boldsymbol{c}(\boldsymbol{z}) \le \mathbf{0}.$$

Thus:

```
c = [Ccol - Cmax_col;
     Cmin_col - Ccol];

ceq = [defect_col;
       BC];
```

Constructing the Initial Guess

An NLP solver requires an initial guess

$$\boldsymbol{z}_{\mathrm{guess}}.$$

A simple and widely used procedure is:

1. guess the initial and final state;
2. linearly interpolate the state over the LGR grid plus the terminal point;
3. guess the control at the first and last collocation points;
4. linearly interpolate the control over the LGR points;
5. guess t_0 and t_f;
6. vectorize and stack all quantities.

State Guess

Given guesses

$$\boldsymbol{X}_1^{(0)} \quad \text{and} \quad \boldsymbol{X}_{N+1}^{(0)},$$

construct

$$\boldsymbol{X}^{(0)}(\tau)$$

by straight-line interpolation at

$$\tau_1, \ldots, \tau_N, \tau_{N+1} = 1.$$

Control Guess

Given endpoint or representative control guesses, interpolate over the LGR points.

Packing the Guess

Finally,

$$\boxed{\boldsymbol{z}_{\text{guess}} = \begin{bmatrix} \text{vec}(\boldsymbol{X}^{(0)}) \\ \text{vec}(\boldsymbol{U}^{(0)}) \\ t_0^{(0)} \\ t_f^{(0)} \end{bmatrix}}.$$

A MATLAB-style implementation is:

```
Xguess = interp1([ -1; 1], [X0guess; Xfguess], ...
                 [tau; 1], 'linear');

Uguess = interp1([ -1; 1], [U0guess; Ufguess], ...
                 tau, 'linear');

zguess = [Xguess(:);
          Uguess(:);
          t0guess;
          tfguess];
```

Variable and Constraint Bounds

Bounds must be packed in exactly the same order as the decision vector:

$$z_{\min} = \begin{bmatrix} \mathrm{vec}(\boldsymbol{X}_{\min}) \\ \mathrm{vec}(\boldsymbol{U}_{\min}) \\ t_{0,\min} \\ t_{f,\min} \end{bmatrix},$$

and similarly for $z_{\max}$.
The defect constraints have zero lower and upper bounds:

$$\boldsymbol{\delta}_{\min} = \boldsymbol{\delta}_{\max} = \mathbf{0}.$$

Endpoint equality constraints also receive identical lower and upper bounds.

Recovering the Solution

After the NLP solver returns

$$z^{\star},$$

unpack it using the same indexing convention:

$$\boldsymbol{X}^{\star} = \mathrm{reshape}\left(z_x^{\star}, N+1, n_x\right),$$

$$\boldsymbol{U}^{\star} = \mathrm{reshape}\left(z_u^{\star}, N, n_u\right).$$

Then reconstruct the physical time vector and inspect:

- state trajectories;
- control trajectories;
- endpoint residuals;
- defect residuals;
- path-constraint residuals;
- solver optimality measures;
- KKT multipliers and costate estimates.

Common Implementation Errors

1. Using inconsistent row-major and column-major packing.
2. Reshaping the state to $N \times n_x$ instead of $(N+1) \times n_x$.
3. Evaluating the control or dynamics at the noncollocated terminal point.
4. Omitting the time-scaling factor $(t_f - t_0)/2$.
5. Using $\boldsymbol{X}_N$ instead of $\boldsymbol{X}_{N+1}$ as the terminal state.
6. Stacking constraint values in an order inconsistent with their bounds.

7. Using the wrong sign convention for the defects.
8. Providing a differentiation matrix of the wrong size.
9. Forgetting to remove the row associated with a noncollocated point when a general differentiation routine returns a square matrix.
10. Starting directly with a complex research problem before validating the implementation on a known problem.

Computing LGR Points, Weights, and Differentiation Matrices

The implementation requires:

- the LGR nodes τ_i;
- the quadrature weights w_i;
- the rectangular differentiation matrix $\boldsymbol{D}$.

A general differentiation routine may return a square matrix on the grid

$$\tau_1, \ldots, \tau_N, \tau_{N+1}.$$

For left-LGR collocation, the final point is noncollocated. Therefore, the row corresponding to $\tau_{N+1} = 1$ must be removed so that

$$\boldsymbol{D} \in \mathbb{R}^{N \times (N+1)}.$$

The columns must not be removed because all $N + 1$ state values participate in the interpolation.

Recommended Validation Workflow

A robust development sequence is:

1. Implement a scalar problem with a known analytical solution.
2. Verify variable packing and unpacking.
3. Verify dimensions of every matrix and vector.
4. Check the objective independently.
5. Evaluate the constraints at a known feasible trajectory.
6. Solve the problem at low polynomial order.
7. Increase N and verify convergence.
8. Compare the state and control against the known solution.
9. Recover and compare the costate when available.
10. Only then adapt the implementation to the research problem.

Key point
Do not begin by solving a complex problem whose solution is unknown. First reproduce two or three simpler benchmark problems. This separates implementation errors from genuine features of the new optimal-control problem.

Minimal End-to-End Workflow

The complete computational process is:

1. Choose N.
2. Compute LGR points, weights, and $\boldsymbol{D}$.
3. Define state, control, and time bounds.
4. Construct $\boldsymbol{z}_{\text{guess}}$.
5. Call the NLP solver.
6. Inside the objective function:
   (a) unpack $\boldsymbol{z}$;
   (b) reshape $\boldsymbol{X}$ and $\boldsymbol{U}$;
   (c) compute physical time;
   (d) evaluate the Mayer and Lagrange costs.
7. Inside the constraint function:
   (a) unpack and reshape;
   (b) evaluate the dynamics;
   (c) form the defect matrix;
   (d) vectorize defects;
   (e) evaluate and vectorize path constraints;
   (f) evaluate endpoint constraints;
   (g) stack all constraints.
8. Recover the optimized trajectories.
9. Check residuals and convergence.
10. Extract multipliers and estimate costates if needed.

Summary

1. The NLP solver operates on one decision vector, not state and control matrices.
2. The discrete state and control arrays must be vectorized and stacked.
3. Inside each NLP function, the vector is unpacked and reshaped.
4. The LGR defect matrix is

$$\boldsymbol{DX} - \frac{t_f - t_0}{2}\boldsymbol{F}.$$

5. Defect and path-constraint matrices must be converted into column vectors.
6. The objective consists of a Mayer term plus an LGR quadrature approximation.
7. Endpoint and path constraints are stacked with the defects.
8. A straight-line state and control profile is often sufficient as an initial guess for simple problems.
9. Solver interfaces differ mainly in how they accept equality and inequality constraints.
10. Implementation should first be validated on simple benchmark problems.

Exercises

1. **LG nonlinear-programming Lagrangian and KKT equations**

   Consider the simplified Mayer problem on $[-1, 1]$,

   $$\min_{\boldsymbol{x}(\cdot),\boldsymbol{u}(\cdot)} \ \Phi(\boldsymbol{x}(1)) \,,$$

   subject to

   $$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}), \qquad \boldsymbol{x}(-1) = \boldsymbol{x}_0.$$

   Use N Legendre–Gauss points, the noncollocated initial state $\boldsymbol{X}_0$, the collocated states $\boldsymbol{X}_{1:N}$, and the recovered terminal state $\boldsymbol{X}_{N+1}$.

   (a) Write the LG defect equations.

   (b) Write the terminal quadrature equation.

   (c) Introduce KKT multipliers for the defect equations, terminal quadrature equation, and initial condition.

   (d) Write the complete NLP Lagrangian using explicit summations.

   (e) Differentiate with respect to $\boldsymbol{U}_i$, $\boldsymbol{X}_j$, $\boldsymbol{X}_{N+1}$, and the multiplier of the initial condition.

   (f) Identify the discrete equations corresponding to continuous control stationarity, costate dynamics, terminal transversality, and initial feasibility.

2. **LG costate mapping**

   Let $\boldsymbol{\Lambda}_{1:N}$ denote the multipliers of the LG defect equations and let $\boldsymbol{\Lambda}_{N+1}$ denote the multiplier of the terminal quadrature equation.

   Starting from the LG KKT equations, derive

   $$\boldsymbol{\lambda}_{N+1} = \boldsymbol{\Lambda}_{N+1},$$

   $$\boldsymbol{\lambda}_{1:N} = \boldsymbol{W}^{-1}\boldsymbol{\Lambda}_{1:N} + \mathbf{1}\boldsymbol{\Lambda}_{N+1},$$

   and

   $$\boldsymbol{\lambda}_0 = \boldsymbol{\lambda}_{N+1} - \boldsymbol{d}_0^{\mathsf{T}}\boldsymbol{\Lambda}_{1:N}.$$

   (a) State the dimensions of every matrix and vector.

   (b) Explain why the terminal quadrature multiplier appears in every interior costate value.

   (c) Show that the mapped multipliers satisfy the discrete terminal transversality condition.

   (d) Explain how the initial costate estimate can be checked against the multiplier of the initial condition.

3. **LGR KKT equations and interior costate mapping**

   Consider a left-LGR transcription with collocation points $\tau_1 = -1, \tau_2, \ldots, \tau_N$ and the noncollocated terminal state $\boldsymbol{X}_{N+1}$.

   The defect equations are written using the convention

$$\boldsymbol{F}_{1:N} - \boldsymbol{D}\boldsymbol{X}_{1:N+1} = \boldsymbol{0}.$$

(a) Write the NLP Lagrangian for the simplified Mayer problem.

(b) Differentiate with respect to each control vector $\boldsymbol{U}_i$.

(c) Differentiate with respect to the interior state vectors $\boldsymbol{X}_j$, $j = 2, \ldots, N$.

(d) Show that the control KKT condition can be written as

$$\nabla_{\boldsymbol{U}_i} \left(\boldsymbol{\lambda}_i^{\mathsf{T}} \boldsymbol{F}_i \right) = \boldsymbol{0}.$$

(e) Derive the interior costate mapping

$$\boldsymbol{\lambda}_{1:N} = \boldsymbol{W}^{-1} \boldsymbol{\Lambda}_{1:N}.$$

(f) Explain why this mapping is algebraically simpler than the LG mapping.

4. **LGR endpoint costates**

For the same LGR transcription, let

$$\boldsymbol{d}_{N+1} = \boldsymbol{D}(:, N+1).$$

(a) Differentiate the NLP Lagrangian with respect to $\boldsymbol{X}_{N+1}$.

(b) Derive

$$\nabla_{\boldsymbol{X}} \Phi(\boldsymbol{X}_{N+1}) = \boldsymbol{d}_{N+1}^{\mathsf{T}} \boldsymbol{\Lambda}_{1:N}.$$

(c) Conclude that the terminal costate estimate is

$$\boldsymbol{\lambda}_{N+1} = \boldsymbol{d}_{N+1}^{\mathsf{T}} \boldsymbol{\Lambda}_{1:N}.$$

(d) Derive the KKT equation associated with $\boldsymbol{X}_1$.

(e) Show how the initial-condition multiplier provides the consistency check

$$\boldsymbol{\lambda}_1 \approx \boldsymbol{\mu}.$$

5. **LGL discrete adjoint system**

Let $\boldsymbol{D} \in \mathbb{R}^{N \times N}$ be the LGL differentiation matrix and

$$\boldsymbol{W} = \operatorname{diag}(w_1, \ldots, w_N).$$

Define

$$\boldsymbol{\lambda}_{1:N} = \boldsymbol{W}^{-1} \boldsymbol{\Lambda}_{1:N}.$$

(a) Starting from the LGL KKT equations, derive the off-diagonal entries of the adjoint differentiation matrix

$$D^{\dagger}_{ij} = -\frac{w_j}{w_i} D_{ji}, \qquad i \neq j.$$

(b) Derive the modified endpoint diagonal entries.

(c) Show that the compact discrete adjoint equation is

$$\boldsymbol{D}^{\dagger}\boldsymbol{\lambda}_{1:N} = -\nabla_{\boldsymbol{X}} \left\langle \boldsymbol{\lambda}_{1:N}, \boldsymbol{F}_{1:N} \right\rangle + \frac{\boldsymbol{e}_1}{w_1} \left(\boldsymbol{\mu} - \boldsymbol{\lambda}_1 \right) + \frac{\boldsymbol{e}_N}{w_N} \left[\boldsymbol{\lambda}_N - \nabla_{\boldsymbol{X}} \Phi(\boldsymbol{X}_N) \right].$$

(d) Explain why the two endpoint residuals remain coupled to the LGL costate dynamics.

(e) Relate this coupling to the singularity of the LGL differentiation matrix.

6. **Comparison of LG, LGR, and LGL costate recovery**

Construct a comparison for the three Legendre pseudospectral schemes.

For each method, determine:

(a) which endpoints are collocation points;

(b) whether a terminal quadrature equation is required;

(c) the mapping from defect multipliers to interior costates;

(d) the formula for the initial costate;

(e) the formula for the terminal costate;

(f) whether endpoint transversality residuals are decoupled from or coupled to the discrete costate dynamics;

(g) whether a full-rank square submatrix of the state differentiation matrix exists;

(h) which method has the simplest multiplier-to-costate mapping.

Then explain why accurate state and control trajectories do not automatically imply accurate recovered costates.

7. **General LGR Bolza transcription**

Consider

$$J = \Phi\big(\boldsymbol{x}(t_0), t_0, \boldsymbol{x}(t_f), t_f\big) + \int_{t_0}^{t_f} L\big(\boldsymbol{x}(t), \boldsymbol{u}(t), t\big)\, \mathrm{d}t,$$

subject to

$$\dot{\boldsymbol{x}} = \boldsymbol{f}\big(\boldsymbol{x}, \boldsymbol{u}, t\big),$$

endpoint constraints

$$\boldsymbol{\phi}\big(\boldsymbol{x}(t_0), t_0, \boldsymbol{x}(t_f), t_f\big) = \boldsymbol{0},$$

and path bounds

$$\boldsymbol{c}_{\min} \le \boldsymbol{c}\big(\boldsymbol{x}, \boldsymbol{u}, t\big) \le \boldsymbol{c}_{\max}.$$

(a) Derive the affine map from $[t_0, t_f]$ to $[-1, 1]$.

(b) Derive

$$\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}\tau} = \frac{t_f - t_0}{2}\boldsymbol{f}.$$

(c) Derive the LGR quadrature approximation of the running cost.

(d) Write the LGR defect matrix.

(e) Write the discrete endpoint constraints.

(f) State precisely where the path constraints are imposed.

(g) Identify every expression in which a free t_f appears.

(h) Assemble the complete finite-dimensional NLP.

8. **Solver-ready decision vector and constraint assembly**

Consider an LGR transcription with N collocation points, n_x states, n_u controls, and free final time.

(a) Define one decision vector containing all state values, control values, and t_f.

(b) Derive its total dimension.

(c) Give explicit reshape operations that recover the state and control arrays.

(d) Write the defect matrix for the linear dynamics

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u}.$$

(e) Show how to vectorize the defect matrix into one column vector.

(f) Add endpoint constraints and two-sided path constraints.

(g) Write the complete equality- and inequality-constraint vectors passed to an NLP solver.

(h) Explain how the ordering of constraint blocks determines the interpretation of the returned KKT multipliers.

9. **Sign conventions and multiplier recovery**

Suppose one implementation uses

$$\boldsymbol{F} - \boldsymbol{D}\boldsymbol{X} = \boldsymbol{0},$$

while another uses

$$\boldsymbol{D}\boldsymbol{X} - \boldsymbol{F} = \boldsymbol{0}.$$

(a) Write the NLP Lagrangian under each convention.

(b) Show how the defect multipliers are related.

(c) Determine how the recovered costates are affected if the same mapping formula is used without correcting the sign.

(d) Derive a practical sign check using the terminal transversality condition.

(e) Derive a second sign check using the control-stationarity residual.

(f) Explain why the exact solver convention must be documented before interpreting multiplier values physically.

10. **Primal and dual verification study**

Use an LGR transcription for

$$\min\ J = \frac{1}{2}x^2(t_f) + \frac{1}{2}\int_0^{t_f} \left(x^2 + u^2\right)\,\mathrm{d}t,$$

subject to

$$\dot{x} = -x + u, \qquad x(0) = x_0,$$

with fixed t_f.

(a) Write the complete LGR NLP.

(b) Recover the interior and terminal costates from the KKT multipliers.

(c) Compute the maximum dynamic-defect residual.

(d) Compute the maximum control-stationarity residual.

(e) Compute the maximum discrete-adjoint residual.

(f) Check the terminal transversality residual.

(g) Check the initial-costate consistency residual.

(h) Repeat the solution for increasing polynomial order.

(i) Compare convergence of the state, control, objective, and costate.

(j) Explain how primal convergence can occur before dual convergence.

Chapter 7: Path Constraints, Dynamic Programming, and Feedback

Chapter 7: Path Constraints, Dynamic Programming, and Feedback

Constraints, value functions, feedback synthesis, and neighboring-optimal corrections

Practical control design must handle restrictions, changing initial conditions, and deviations from a nominal trajectory. The chapter first develops equality and inequality path constraints, including constrained arcs and their numerical enforcement. It then changes viewpoint from trajectory optimization to the value function through dynamic programming and HJB theory, recovers LQR feedback, places optimal control within a guidance–navigation–control architecture, and concludes with analytical simplification and neighboring-optimal corrections.

Chapter contents

Equality Path Constraints in Optimal Control

This section introduces path constraints in optimal control and develops the necessary conditions for equality path constraints. It begins by interpreting the system dynamics themselves as path constraints, then distinguishes differential, algebraic, control, and state constraints. The augmented cost functional and augmented Hamiltonian are introduced, followed by the necessary conditions for control-only, state-only, and mixed state–control equality path constraints.

What Is a Path Constraint?

A path constraint is a condition that must hold pointwise along the trajectory. The familiar dynamics

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t), t)$$

can be rewritten as

$$\boxed{\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, t) - \dot{\boldsymbol{x}} = \boldsymbol{0}.}$$

Thus, the dynamics themselves are equality path constraints because they must hold at every time.

Differential Forms and Differential Inclusions

A more general differential constraint may be written as

$$\boxed{\boldsymbol{\psi}(\boldsymbol{x}, \dot{\boldsymbol{x}}, \boldsymbol{u}, t) = \boldsymbol{0}.}$$

This is called a differential form because the derivative need not be explicitly isolated.

The inequality generalization is

$$\boxed{\boldsymbol{\psi}_{\min} \leq \boldsymbol{\psi}(\boldsymbol{x}, \dot{\boldsymbol{x}}, \boldsymbol{u}, t) \leq \boldsymbol{\psi}_{\max},}$$

which is a differential inclusion.

In classical optimal-control theory, the state rates are usually handled through explicit first-order dynamics, while additional constraints are treated as algebraic path constraints.

Algebraic Path Constraints

An algebraic path constraint does not explicitly involve $\dot{\boldsymbol{x}}$. Typical forms are:

$$\begin{aligned} \boldsymbol{c}(\boldsymbol{u}, t) &= \boldsymbol{0}, \\ \boldsymbol{c}(\boldsymbol{x}, t) &= \boldsymbol{0}, \\ \boldsymbol{c}(\boldsymbol{x}, \boldsymbol{u}, t) &= \boldsymbol{0}, \end{aligned}$$

for equality constraints, or corresponding inequalities.

Original Augmented Cost Functional

Consider a Bolza problem with endpoint constraints. The standard augmented functional is

$$J_A = \Phi(\boldsymbol{x}(t_0), t_0, \boldsymbol{x}(t_f), t_f) - \boldsymbol{\nu}^\mathsf{T} \boldsymbol{\phi}(\boldsymbol{x}(t_0), t_0, \boldsymbol{x}(t_f), t_f) + \int_{t_0}^{t_f} \left[L(\boldsymbol{x}, \boldsymbol{u}, t) + \boldsymbol{\lambda}^\mathsf{T} \left(\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, t) - \dot{\boldsymbol{x}} \right) \right] \, \mathrm{d}t.$$

At any feasible solution, the added constraint terms vanish. Therefore, the original and augmented costs agree.

Feasibility Principle
At any feasible trajectory, every equality constraint is zero. Hence, adding a Lagrange-multiplier term does not change the objective value on the feasible set.

Control Equality Path Constraints

Consider

$$\boxed{\boldsymbol{c}(\boldsymbol{u}, t) = \boldsymbol{0},}$$

with

$$\boldsymbol{c} : \mathbb{R}^{n_u} \times \mathbb{R} \to \mathbb{R}^p.$$

Introduce a time-varying multiplier

$$\boldsymbol{\mu}(t) \in \mathbb{R}^p.$$

The augmented functional becomes

$$J_A = \Phi - \boldsymbol{\nu}^\mathsf{T} \boldsymbol{\phi} + \int_{t_0}^{t_f} \left[L + \boldsymbol{\lambda}^\mathsf{T} (\boldsymbol{f} - \dot{\boldsymbol{x}}) - \boldsymbol{\mu}^\mathsf{T} \boldsymbol{c} \right] \, \mathrm{d}t.$$

Augmented Hamiltonian

Define the ordinary Hamiltonian

$$\mathcal{H}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\lambda}, t) = L(\boldsymbol{x}, \boldsymbol{u}, t) + \boldsymbol{\lambda}^\mathsf{T} \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, t).$$

The augmented Hamiltonian is

$$\boxed{\mathcal{H}_A = \mathcal{H} - \boldsymbol{\mu}^\mathsf{T} \boldsymbol{c}.}$$

The necessary conditions have the same structure as before, but with $\mathcal{H}_A$ replacing $\mathcal{H}$.

Necessary Conditions for Control Equality Constraints

The state equation remains

$$\dot{\boldsymbol{x}} = \frac{\partial \mathcal{H}_A}{\partial \boldsymbol{\lambda}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, t).$$

Because $\boldsymbol{c}$ does not depend on $\boldsymbol{x}$,

$$\frac{\partial \mathcal{H}_A}{\partial \boldsymbol{x}} = \frac{\partial \mathcal{H}}{\partial \boldsymbol{x}}.$$

Therefore,

$$\boxed{\dot{\boldsymbol{\lambda}} = -\frac{\partial \mathcal{H}}{\partial \boldsymbol{x}}.}$$

The stationarity condition changes:

$$\boxed{\frac{\partial \mathcal{H}_A}{\partial \boldsymbol{u}} = \frac{\partial \mathcal{H}}{\partial \boldsymbol{u}} - \left(\frac{\partial \boldsymbol{c}}{\partial \boldsymbol{u}}\right)^{\mathsf{T}} \boldsymbol{\mu} = \boldsymbol{0}.}$$

The path constraint itself must also hold:

$$\boxed{\boldsymbol{c}(\boldsymbol{u}, t) = \boldsymbol{0}.}$$

Thus, the unknown multiplier $\boldsymbol{\mu}(t)$ is determined together with the control through a coupled algebraic system.

Differential-Algebraic Structure

For a control equality constraint, the optimality system includes:

1. state differential equations;
2. costate differential equations;
3. stationarity equations;
4. algebraic path constraints;
5. endpoint constraints and transversality conditions.

The result is generally a differential-algebraic boundary-value problem.

State Equality Path Constraints

Now consider

$$\boxed{\boldsymbol{c}(\boldsymbol{x}, t) = \boldsymbol{0}.}$$

The augmented Hamiltonian remains

$$\mathcal{H}_A = \mathcal{H} - \boldsymbol{\mu}^\mathsf{T} \boldsymbol{c}.$$

The costate equation becomes

$$\boxed{\dot{\boldsymbol{\lambda}} = -\frac{\partial \mathcal{H}_A}{\partial \boldsymbol{x}} = -\frac{\partial \mathcal{H}}{\partial \boldsymbol{x}} + \left(\frac{\partial \boldsymbol{c}}{\partial \boldsymbol{x}}\right)^\mathsf{T} \boldsymbol{\mu}.}$$

However, because $\boldsymbol{c}$ does not explicitly depend on $\boldsymbol{u}$,

$$\frac{\partial \mathcal{H}_A}{\partial \boldsymbol{u}} = \frac{\partial \mathcal{H}}{\partial \boldsymbol{u}}.$$

Therefore, the original stationarity condition contains no explicit information about the effect of the state constraint on the control.

Constraint Order

To expose the control, differentiate the state constraint with respect to time.
Starting from

$$\boldsymbol{c}(\boldsymbol{x}, t) = \boldsymbol{0},$$

differentiate:

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{c}(\boldsymbol{x}, t) = \frac{\partial \boldsymbol{c}}{\partial \boldsymbol{x}}\dot{\boldsymbol{x}} + \frac{\partial \boldsymbol{c}}{\partial t} = \boldsymbol{0}.$$

Substitute the dynamics:

$$\boldsymbol{c}^{(1)} = \frac{\partial \boldsymbol{c}}{\partial \boldsymbol{x}}\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, t) + \frac{\partial \boldsymbol{c}}{\partial t} = \boldsymbol{0}.$$

If the control still does not appear explicitly, differentiate again. Continue until the control appears.

Order of a State Path Constraint
The order q of a state path constraint is the smallest number of total time differentiations required before the control appears explicitly.

Symbolically,

$$\begin{aligned}
\boldsymbol{c}^{(0)}(\boldsymbol{x}, t) &= \boldsymbol{0}, \\
\boldsymbol{c}^{(1)}(\boldsymbol{x}, t) &= \boldsymbol{0}, \\
&\vdots \\
\boldsymbol{c}^{(q-1)}(\boldsymbol{x}, t) &= \boldsymbol{0}, \\
\boldsymbol{c}^{(q)}(\boldsymbol{x}, \boldsymbol{u}, t) &= \boldsymbol{0}.
\end{aligned}$$

The final relation is control dependent and can be used in the stationarity system.

Why All Lower-Order Conditions Matter

It is not sufficient to enforce only the highest derivative condition. If

$$\boldsymbol{c}^{(q)} = \mathbf{0},$$

then integration introduces constants. Therefore, the original condition and all lower derivatives must also be satisfied:

$$\boxed{\boldsymbol{c}^{(j)} = \mathbf{0}, \qquad j = 0, \ldots, q.}$$

Mixed State–Control Equality Constraints

For

$$\boxed{\boldsymbol{c}(\boldsymbol{x}, \boldsymbol{u}, t) = \mathbf{0},}$$

the augmented Hamiltonian is again

$$\mathcal{H}_A = \mathcal{H} - \boldsymbol{\mu}^{\mathsf{T}} \boldsymbol{c}.$$

The costate equation is

$$\boxed{\dot{\boldsymbol{\lambda}} = -\frac{\partial \mathcal{H}}{\partial \boldsymbol{x}} + \left(\frac{\partial \boldsymbol{c}}{\partial \boldsymbol{x}}\right)^{\mathsf{T}} \boldsymbol{\mu}.}$$

The stationarity condition is

$$\boxed{\frac{\partial \mathcal{H}}{\partial \boldsymbol{u}} - \left(\frac{\partial \boldsymbol{c}}{\partial \boldsymbol{u}}\right)^{\mathsf{T}} \boldsymbol{\mu} = \mathbf{0}.}$$

The algebraic constraint must also hold:

$$\boxed{\boldsymbol{c}(\boldsymbol{x}, \boldsymbol{u}, t) = \mathbf{0}.}$$

Summary

1. The dynamics are equality path constraints.
2. Differential forms generalize explicit state equations.
3. Differential inclusions generalize differential equalities to inequalities.
4. Algebraic path constraints are imposed pointwise along the trajectory.

5. Equality path constraints introduce time-varying Lagrange multipliers.
6. The augmented Hamiltonian is

$$\mathcal{H}_A = \mathcal{H} - \boldsymbol{\mu}^{\mathsf{T}} \boldsymbol{c}.$$

7. A control equality constraint modifies stationarity but not the costate equation.
8. A state equality constraint modifies the costate equation but initially provides no direct control relation.
9. State constraints must be differentiated until the control appears.
10. Equality path constraints lead naturally to differential-algebraic optimality systems.

Connection. Equality constraints establish the multiplier framework; inequalities add activity decisions, complementary slackness, and junction behavior.

Inequality Path Constraints, Complementarity, and Constrained Arcs

This section develops the necessary conditions for inequality path constraints. It introduces active and inactive constraints, complementary slackness, control inequality constraints, state inequality constraints, constrained arcs, and entry conditions. The emphasis is on the theoretical structure that must hold along an optimal trajectory.

Control Inequality Path Constraints

Consider

$$\boxed{\boldsymbol{c}(\boldsymbol{u}, t) \leq \mathbf{0}.}$$

Introduce the augmented Hamiltonian

$$\mathcal{H}_A = \mathcal{H} - \boldsymbol{\mu}^{\mathsf{T}} \boldsymbol{c}.$$

The multiplier is time varying. The stationarity condition is

$$\boxed{\frac{\partial \mathcal{H}}{\partial \boldsymbol{u}} - \left(\frac{\partial \boldsymbol{c}}{\partial \boldsymbol{u}}\right)^{\mathsf{T}} \boldsymbol{\mu} = \mathbf{0}.}$$

Because the constraint is independent of the state,

$$\dot{\boldsymbol{\lambda}} = -\frac{\partial \mathcal{H}}{\partial \boldsymbol{x}}.$$

Active and Inactive Constraints

For a scalar constraint $c(\boldsymbol{u}, t) \leq 0$:

- The constraint is *inactive* if $c < 0$.
- The constraint is *active* if $c = 0$.

If the unconstrained optimal control already satisfies the bound strictly, then the constraint has no effect on the solution.

Inactive Constraint
An inactive inequality constraint behaves as though it were absent from the problem.

Complementary Slackness

The multiplier and constraint must satisfy

$$\boxed{\boldsymbol{\mu}^{\mathsf{T}} \boldsymbol{c} = \mathbf{0}.}$$

With the sign convention

$$\mathcal{H}_A = \mathcal{H} - \boldsymbol{\mu}^\mathsf{T} \boldsymbol{c}, \qquad \boldsymbol{c} \le \mathbf{0},$$

the multiplier satisfies

$$\boxed{\boldsymbol{\mu} \le \mathbf{0}.}$$

Componentwise,

$$\mu_i c_i = 0, \qquad \mu_i \le 0, \qquad c_i \le 0.$$

Thus:

$$c_i < 0 \Longrightarrow \mu_i = 0,$$
$$c_i = 0 \Longrightarrow \mu_i \le 0.$$

Interpretation of the Multiplier

The path multiplier represents the local sensitivity of the optimal cost to relaxation of the constraint. An inactive constraint has zero sensitivity. An active constraint may have a nonzero multiplier.

The sign depends on the convention used in the augmented Hamiltonian. A different sign convention changes the sign restriction but not the underlying complementarity structure.

Complete Conditions for a Control Inequality

For

$$\boldsymbol{c}(\boldsymbol{u}, t) \le \mathbf{0},$$

the necessary conditions include

$$\begin{aligned}
\dot{\boldsymbol{x}} &= \frac{\partial \mathcal{H}_A}{\partial \boldsymbol{\lambda}}, \\
\dot{\boldsymbol{\lambda}} &= -\frac{\partial \mathcal{H}_A}{\partial \boldsymbol{x}}, \\
\frac{\partial \mathcal{H}_A}{\partial \boldsymbol{u}} &= \mathbf{0}, \\
\boldsymbol{c} &\le \mathbf{0}, \\
\boldsymbol{\mu} &\le \mathbf{0}, \\
\boldsymbol{\mu}^\mathsf{T} \boldsymbol{c} &= 0.
\end{aligned}$$

These are supplemented by endpoint conditions and transversality conditions.

State Inequality Path Constraints

Now consider

$$\boxed{\boldsymbol{c}(\boldsymbol{x}, t) \leq \boldsymbol{0}.}$$

This case is more difficult for two reasons:

1. the constraint does not directly appear in the control stationarity equation;
2. the times at which the constraint becomes active are generally unknown.

The augmented Hamiltonian is

$$\mathcal{H}_A = \mathcal{H} - \boldsymbol{\mu}^{\mathsf{T}} \boldsymbol{c}.$$

The costate equation is

$$\boxed{\dot{\boldsymbol{\lambda}} = -\frac{\partial \mathcal{H}}{\partial \boldsymbol{x}} + \left(\frac{\partial \boldsymbol{c}}{\partial \boldsymbol{x}}\right)^{\mathsf{T}} \boldsymbol{\mu}.}$$

The ordinary stationarity equation remains

$$\frac{\partial \mathcal{H}}{\partial \boldsymbol{u}} = \boldsymbol{0}$$

until the state constraint is differentiated enough times for the control to appear.

Unconstrained and Constrained Arcs

An optimal trajectory may consist of:

- an unconstrained arc, where $\boldsymbol{c} < \boldsymbol{0}$;
- a constrained arc, where $\boldsymbol{c} = \boldsymbol{0}$;
- a second unconstrained arc after leaving the boundary.

A qualitative scalar picture is shown below.

Unknown Junction Times

The entry time t_{in} and exit time t_{out} are generally unknown. They become additional unknowns in an indirect formulation.

This is one reason state inequality constraints are significantly harder than control bounds.

Constraint Order on a Constrained Arc

Suppose the state constraint has order q. Define

$$\boldsymbol{c}^{(0)} = \boldsymbol{c}.$$

Differentiate until the control appears:

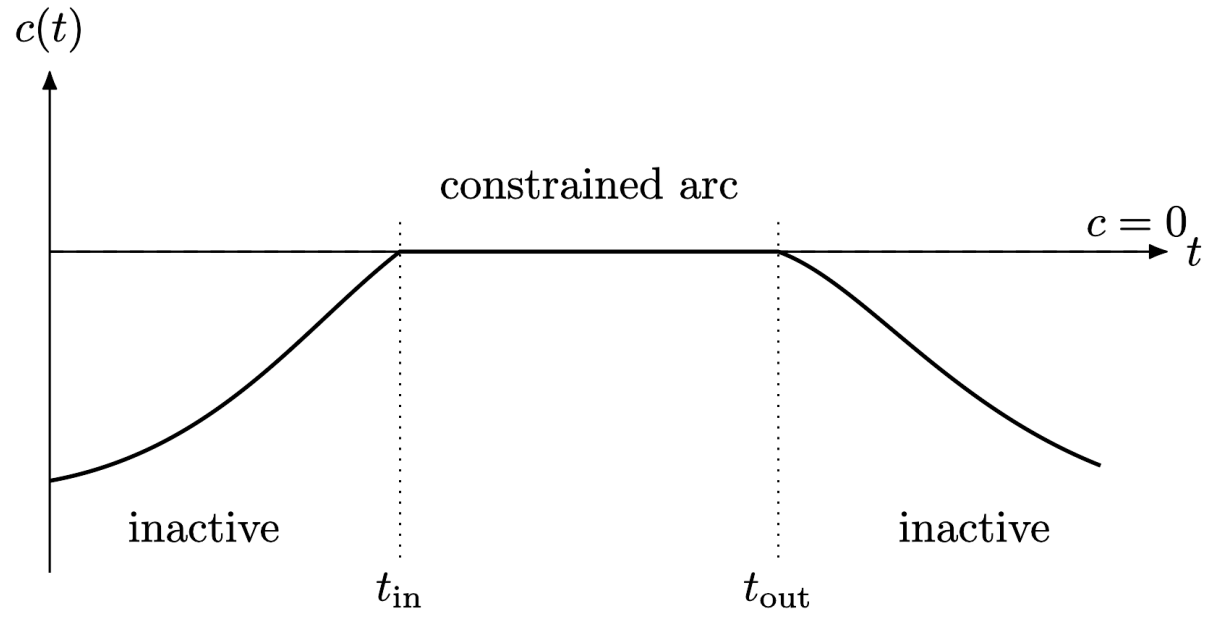

Figure 21: An inequality constraint with inactive and constrained arcs.

$$
\begin{aligned}
\boldsymbol{c}^{(0)}(\boldsymbol{x}, t) &= \boldsymbol{0}, \\
\boldsymbol{c}^{(1)}(\boldsymbol{x}, t) &= \boldsymbol{0}, \\
&\vdots \\
\boldsymbol{c}^{(q-1)}(\boldsymbol{x}, t) &= \boldsymbol{0}, \\
\boldsymbol{c}^{(q)}(\boldsymbol{x}, \boldsymbol{u}, t) &= \boldsymbol{0}.
\end{aligned}
$$

These equalities hold on the active constrained arc.

Entry Conditions

At the entry point, continuity and tangency require

$$\boxed{\boldsymbol{c}^{(j)}(t_{\text{in}}) = \boldsymbol{0}, \qquad j = 0, \ldots, q-1.}$$

For a first-order scalar state constraint,

$$c(t_{\text{in}}) = 0.$$

For a second-order state constraint,

$$c(t_{\text{in}}) = 0, \qquad \dot{c}(t_{\text{in}}) = 0.$$

These conditions ensure tangential contact with the constraint boundary rather than crossing into the infeasible region.

Tangency Interpretation

At the moment a smooth trajectory reaches an active state boundary, it must become tangent to that boundary. For a scalar state constraint:

$$c < 0$$

on the unconstrained side and

$$c = 0$$

on the constrained arc.

If $\dot{c}$ were positive at entry for a constraint $c \leq 0$, the trajectory would immediately enter the infeasible region. Therefore, the appropriate tangency conditions must hold.

Control on the Constrained Arc

Once the state constraint has been differentiated until the control appears, the highest-order relation

$$\boldsymbol{c}^{(q)}(\boldsymbol{x}, \boldsymbol{u}, t) = \boldsymbol{0}$$

can often be solved for the boundary control:

$$\boxed{\boldsymbol{u} = \boldsymbol{u}_{\text{boundary}}(\boldsymbol{x}, t).}$$

This control keeps the state exactly on the active boundary.

Multiplier Behavior

On inactive arcs:

$$\boldsymbol{\mu} = \boldsymbol{0}.$$

On active arcs:

$$\boldsymbol{c} = \boldsymbol{0}, \qquad \boldsymbol{\mu} \leq \boldsymbol{0}.$$

The multiplier may change at entry and exit points. Depending on the order and regularity of the state constraint, the costate may exhibit junction conditions or jumps. A full derivation of those jump conditions belongs to the continuation of the state-constraint theory.

Why Inequality State Constraints Are Difficult

They require simultaneous determination of:

1. the unconstrained solution;

2. the constrained solution;
3. the entry time;
4. the exit time;
5. the boundary control;
6. the path multiplier;
7. the junction conditions.

Therefore, the resulting indirect problem is a multiphase boundary-value problem with unknown switching times.

Direct Transcription Interpretation

In direct collocation, inequality path constraints are imposed at the discretization points:

$$\boldsymbol{c}_{\min} \le \boldsymbol{c}(\boldsymbol{X}_i, \boldsymbol{U}_i, t_i) \le \boldsymbol{c}_{\max}.$$

The NLP solver determines which constraints are active. The corresponding NLP multipliers approximate the continuous path multipliers.
This is one reason direct methods are often preferred for complicated path-constrained problems.

Control Bounds as a Special Case

A box constraint

$$\boldsymbol{u}_{\min} \le \boldsymbol{u}(t) \le \boldsymbol{u}_{\max}$$

can be represented by

$$\boldsymbol{u} - \boldsymbol{u}_{\max} \le \boldsymbol{0},$$
$$\boldsymbol{u}_{\min} - \boldsymbol{u} \le \boldsymbol{0}.$$

Each side has its own multiplier and complementary-slackness condition.

State Bounds as a Special Case

Similarly,

$$\boldsymbol{x}_{\min} \le \boldsymbol{x}(t) \le \boldsymbol{x}_{\max}$$

is represented by two sets of state inequality constraints.
Unlike control bounds, state bounds may have nonzero order and can generate constrained arcs.

Summary

1. Inequality path constraints may be active or inactive.

2. Inactive constraints have zero multiplier.
3. Active constraints satisfy the boundary equality and may have nonzero multiplier.
4. Complementary slackness is

$$\boldsymbol{\mu}^{\mathsf{T}}\boldsymbol{c} = 0.$$

5. Under the convention $\mathcal{H}_A = \mathcal{H} - \boldsymbol{\mu}^{\mathsf{T}}\boldsymbol{c}$ with $\boldsymbol{c} \leq 0$, one has $\boldsymbol{\mu} \leq 0$.
6. Control inequalities directly modify stationarity.
7. State inequalities must be differentiated until the control appears.
8. Active state constraints generate constrained arcs.
9. Entry conditions enforce contact and tangency.
10. Unknown entry and exit times make indirect formulations difficult.
11. Direct collocation handles active-set identification through the NLP solver.

Connection. State inequalities are especially subtle because the control affects them through the dynamics and their active intervals are not known in advance.

State Inequality Path Constraints and Direct Collocation

This section continues the treatment of path constraints by focusing on state inequality path constraints and their numerical implementation. These constraints are challenging because their activity is not generally known in advance. A state inequality constraint may be inactive over part of the trajectory, active over one or more constrained arcs, and inactive again later.

The continuous-time constraint is written as

$$\boxed{c(\boldsymbol{x}(t), t) \leq 0.}$$

For clarity, most of the derivations in this section use a scalar constraint and a scalar control. The same ideas extend to vector-valued constraints and multiple controls.

Why State Inequality Constraints Are Difficult

A priori, one generally does not know whether

$$c(\boldsymbol{x}(t), t) \leq 0$$

will be active at the optimal solution.

If

$$c(\boldsymbol{x}^*(t), t) < 0$$

for all t, then the constraint is inactive and has no effect on the optimal trajectory. If it becomes active, then there exists an interval over which

$$c(\boldsymbol{x}^*(t), t) = 0.$$

The times at which the constraint becomes active and inactive are generally unknown. These unknown junction times create much of the difficulty in indirect formulations.

Assumed Constrained Arc

Suppose that there exists a single interval

$$\left[t_1, t_2\right] \subset \left[t_0, t_f\right]$$

such that

$$\boxed{c(\boldsymbol{x}(t), t) = 0, \qquad t \in [t_1, t_2].}$$

The interval $[t_1, t_2]$ is called a constrained arc. The times t_1 and t_2 are the entry and exit times.

A qualitative representation is shown below.

This structure is an assumption. The true solution may have no constrained arc, one constrained arc, or several constrained arcs.

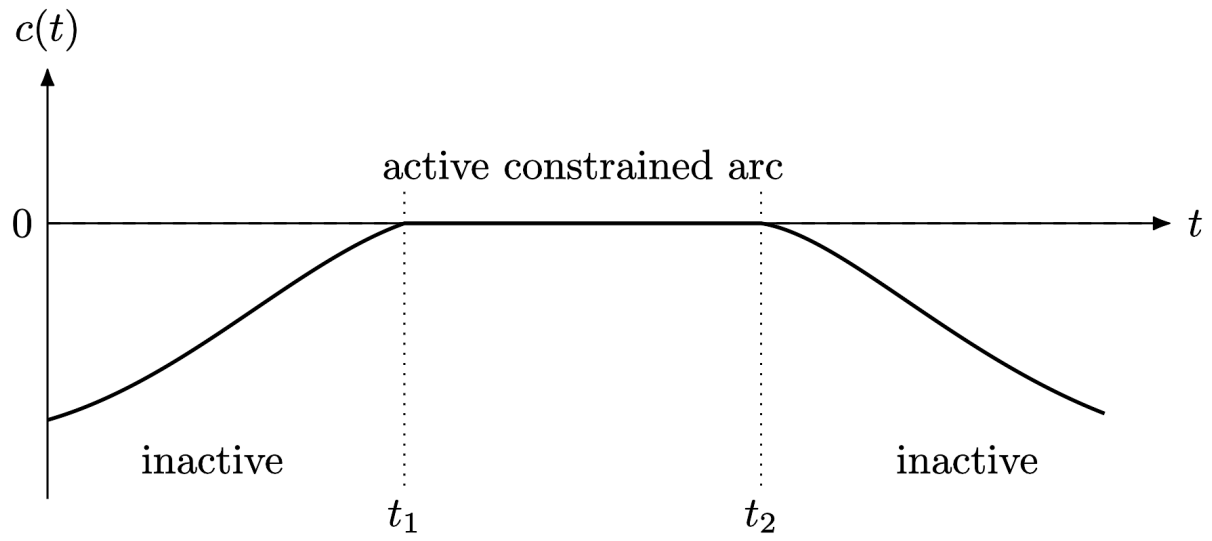

Figure 22: A state inequality constraint that is active on one interval.

Constraint Derivatives

On the constrained arc,

$$c(\boldsymbol{x}(t), t) = 0.$$

Therefore, all total time derivatives must also vanish:

$$\begin{aligned} c(\boldsymbol{x}(t), t) &= 0, \\ \frac{\mathrm{d}c}{\mathrm{d}t} &= 0, \\ \frac{\mathrm{d}^2 c}{\mathrm{d}t^2} &= 0, \\ &\vdots \\ \frac{\mathrm{d}^q c}{\mathrm{d}t^q} &= 0. \end{aligned}$$

The total derivative is

$$\frac{\mathrm{d}c}{\mathrm{d}t} = \frac{\partial c}{\partial \boldsymbol{x}}\dot{\boldsymbol{x}} + \frac{\partial c}{\partial t}.$$

Substituting the dynamics

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, t)$$

gives

$$\frac{\mathrm{d}c}{\mathrm{d}t} = \frac{\partial c}{\partial \boldsymbol{x}} \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, t) + \frac{\partial c}{\partial t}.$$

Order of a State Constraint

Assume that the control does not appear explicitly in

$$c, \dot{c}, \ldots, c^{(q-1)},$$

but appears explicitly in

$$c^{(q)}.$$

Then the state constraint is said to have order q.

Constraint Order
The order q is the smallest number of total time differentiations required before the control appears explicitly.

Thus,

$$\begin{aligned} c^{(0)}(\boldsymbol{x}, t) &= 0, \\ c^{(1)}(\boldsymbol{x}, t) &= 0, \\ &\vdots \\ c^{(q-1)}(\boldsymbol{x}, t) &= 0, \\ c^{(q)}(\boldsymbol{x}, \boldsymbol{u}, t) &= 0. \end{aligned}$$

The highest derivative provides a control-dependent equality that can be used to determine the boundary control.

Tangency Conditions at Entry

At the entry time t_1, the following conditions must hold:

$$\boxed{c^{(j)}(\boldsymbol{x}(t_1), t_1) = 0, \qquad j = 0, \ldots, q-1.}$$

These are the tangency conditions.
For a first-order constraint,

$$c(\boldsymbol{x}(t_1), t_1) = 0.$$

For a second-order constraint,

$$c(\boldsymbol{x}(t_1), t_1) = 0,$$
$$\dot{c}(\boldsymbol{x}(t_1), t_1) = 0.$$

For a third-order constraint,

$$c(\boldsymbol{x}(t_1), t_1) = 0,$$
$$\dot{c}(\boldsymbol{x}(t_1), t_1) = 0,$$
$$\ddot{c}(\boldsymbol{x}(t_1), t_1) = 0.$$

These conditions ensure that the trajectory joins the constraint boundary smoothly rather than crossing into the infeasible region.

Interior-Point Interpretation

The tangency conditions act as interior-point constraints:

$$\boldsymbol{\psi}(\boldsymbol{x}(t_1), t_1) = \mathbf{0},$$

where

$$\boldsymbol{\psi} = \begin{bmatrix} c \\ \dot{c} \\ \vdots \\ c^{(q-1)} \end{bmatrix}.$$

Therefore, the constrained problem can be interpreted as a multiphase optimal control problem with an interior junction at t_1.

The trajectory must satisfy:

1. unconstrained dynamics on $[t_0, t_1)$;
2. tangency conditions at t_1;
3. constrained dynamics on $[t_1, t_2]$;
4. exit or junction conditions at t_2;
5. unconstrained dynamics on $(t_2, t_f]$.

Replacing the Inequality on the Active Arc

On the active arc, the original inequality

$$c(\boldsymbol{x}(t), t) \leq 0$$

is replaced by the equality

$$\boxed{c^{(q)}(\boldsymbol{x}(t), \boldsymbol{u}(t), t) = 0, \qquad t \in [t_1, t_2].}$$

The lower-order conditions remain satisfied as a consequence of the tangency conditions and the constrained dynamics.

In favorable cases, one can solve

$$c^{(q)}(\boldsymbol{x}, \boldsymbol{u}, t) = 0$$

for the boundary control:

$$\boxed{\boldsymbol{u} = \boldsymbol{u}_{\text{boundary}}(\boldsymbol{x}, t).}$$

Single and Multiple Constrained Arcs

The assumption of one active interval may be incorrect. The unconstrained solution may suggest several separate violations.

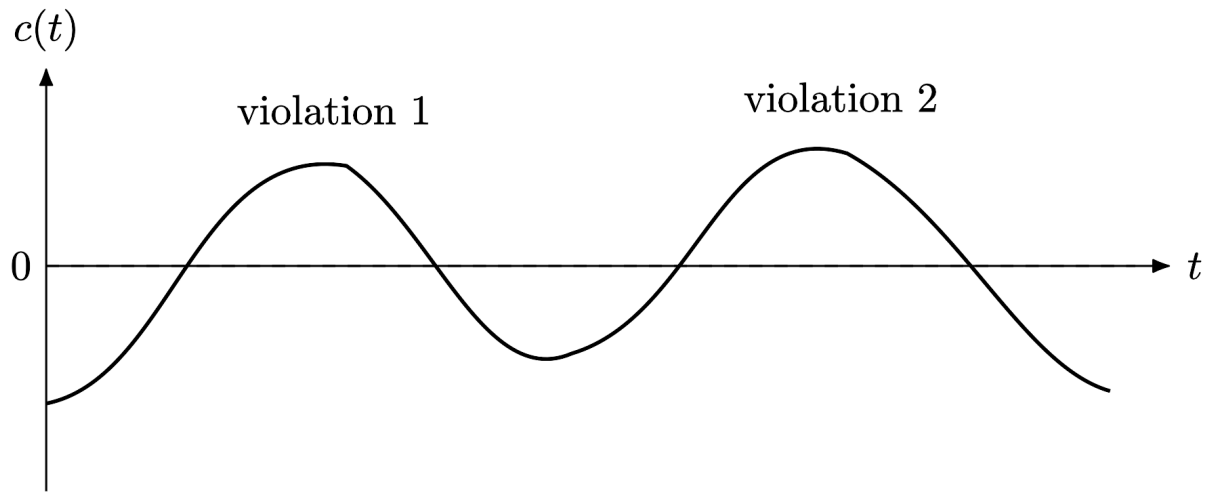

Figure 23: An unconstrained solution may violate a state inequality constraint over multiple intervals.

However, two unconstrained violations do not guarantee that the constrained optimal solution has two constrained arcs. Enforcing one arc may alter the trajectory enough to eliminate another violation.

Therefore, the active-set structure must be tested rather than blindly assumed.

Solve the Unconstrained Problem First

A practical indirect-method strategy is:

1. Solve the problem without the state inequality constraint.
2. Evaluate $c(\boldsymbol{x}(t), t)$ along the unconstrained trajectory.
3. If $c < 0$ everywhere, the inequality is inactive.
4. If $c > 0$ somewhere, identify approximate violation intervals.
5. Use these intervals to initialize candidate entry and exit times.

6. Solve constrained formulations with one or more candidate arcs.
7. Verify complementarity, tangency, and feasibility.

This strategy provides structural information before the difficult constrained boundary-value problem is attempted.

Risk of Assuming an Active Constraint Incorrectly

Suppose one assumes an active arc and imposes tangency and boundary-control conditions, even though the true optimum never activates the constraint.

The resulting solution is generally feasible for a more restrictive problem but is not the optimum of the original inequality-constrained problem. It is therefore suboptimal.

This illustrates a key principle:

> **Structural Verification**
> A mathematically valid solution to an incorrectly assumed arc structure is not necessarily the solution of the original optimal control problem.

Difficulty of Indirect Shooting

Without path constraints, an indirect method typically solves a two-point boundary-value problem consisting of:

- state equations;
- costate equations;
- stationarity conditions;
- endpoint boundary conditions;
- transversality conditions.

A state inequality path constraint introduces:

- unknown active intervals;
- unknown entry and exit times;
- interior tangency conditions;
- boundary-control equations;
- algebraic constraints on constrained arcs;
- possible costate junction conditions.

The problem becomes a multiphase differential-algebraic boundary-value problem.

Transition to Direct Collocation

Direct collocation avoids explicit derivation of the constrained-arc structure in the problem formulation.

Consider the continuous optimal control problem

$$\begin{aligned}\min_{\boldsymbol{x}(\cdot),\boldsymbol{u}(\cdot),t_0,t_f} \quad & J = \Phi(\boldsymbol{x}(t_0), t_0, \boldsymbol{x}(t_f), t_f) + \int_{t_0}^{t_f} L(\boldsymbol{x}, \boldsymbol{u}, t)\,\mathrm{d}t, \\ & \dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, t), \\ & \boldsymbol{\phi}(\boldsymbol{x}(t_0), t_0, \boldsymbol{x}(t_f), t_f) = \boldsymbol{0}, \\ & \boldsymbol{c}_{\min} \le \boldsymbol{c}(\boldsymbol{x}, \boldsymbol{u}, t) \le \boldsymbol{c}_{\max}.\end{aligned}$$

LGR/Radau Discretization

Using an N-point Legendre–Gauss–Radau discretization, the cost is approximated by

$$\begin{aligned}J_N = {} & \Phi(\boldsymbol{X}_1, t_0, \boldsymbol{X}_{N+1}, t_f) \\ & + \frac{t_f - t_0}{2} \sum_{i=1}^{N} w_i L(\boldsymbol{X}_i, \boldsymbol{U}_i, t_i).\end{aligned}$$

The defect constraints are

$$\boxed{\boldsymbol{D}\boldsymbol{X} - \frac{t_f - t_0}{2}\boldsymbol{F} = \boldsymbol{0}.}$$

More explicitly,

$$\sum_{j=1}^{N+1} D_{ij}\boldsymbol{X}_j - \frac{t_f - t_0}{2}\boldsymbol{f}(\boldsymbol{X}_i, \boldsymbol{U}_i, t_i) = \boldsymbol{0}, \qquad i = 1, \ldots, N.$$

Endpoint constraints become

$$\boldsymbol{\phi}(\boldsymbol{X}_1, t_0, \boldsymbol{X}_{N+1}, t_f) = \boldsymbol{0}.$$

Discrete Path Constraints

The continuous path constraint

$$\boldsymbol{c}_{\min} \le \boldsymbol{c}(\boldsymbol{x}(t), \boldsymbol{u}(t), t) \le \boldsymbol{c}_{\max}$$

is enforced at the Radau points:

$$\boxed{\boldsymbol{c}_{\min} \le \boldsymbol{c}(\boldsymbol{X}_i, \boldsymbol{U}_i, t_i) \le \boldsymbol{c}_{\max}, \qquad i = 1, \ldots, N.}$$

Equivalently,

$$\begin{bmatrix} \boldsymbol{c}(\boldsymbol{X}_1, \boldsymbol{U}_1, t_1) \\ \boldsymbol{c}(\boldsymbol{X}_2, \boldsymbol{U}_2, t_2) \\ \vdots \\ \boldsymbol{c}(\boldsymbol{X}_N, \boldsymbol{U}_N, t_N) \end{bmatrix} \in [\boldsymbol{c}_{\min}, \boldsymbol{c}_{\max}].$$

No continuous-time path multiplier is explicitly introduced into the NLP formulation.

Why No Explicit Multiplier Is Added

The nonlinear program is written directly in terms of decision variables and constraints. The NLP solver introduces and estimates its own Lagrange multipliers internally as part of the Karush–Kuhn–Tucker conditions.

Therefore, the user provides:

- the objective;
- equality constraints;
- inequality constraints;
- variable bounds.

The solver determines which inequality constraints are active at the discrete optimum.

Major Advantage of Direct Collocation

Direct collocation does not require the user to specify:

- whether a path constraint is active;
- the number of constrained arcs;
- the entry times;
- the exit times;
- tangency conditions in the NLP formulation.

The NLP solver identifies the discrete active set.

Direct-Method Advantage
Path constraints are incorporated as ordinary finite-dimensional inequalities at the collocation points. Active-set identification is delegated to the NLP solver.

What the NLP Solver Actually Returns

The NLP solver returns a finite-dimensional decision vector,

$$z = \begin{bmatrix} \mathrm{vec}(\boldsymbol{X}) \\ \mathrm{vec}(\boldsymbol{U}) \\ t_0 \\ t_f \end{bmatrix},$$

not a continuous trajectory.

The quantities $\boldsymbol{X}_i$ and $\boldsymbol{U}_i$ are discrete samples. A reconstructed state and control must be produced by interpolation.

Therefore, satisfaction of a path constraint at the collocation points does not automatically imply satisfaction between those points.

Discrete Activity Does Not Prove a Continuous Constrained Arc

Suppose the NLP solution satisfies

$$c(\boldsymbol{X}_i, \boldsymbol{U}_i, t_i) = 0$$

at several neighboring Radau points.
This means only that those discrete constraints are active. It does not prove that

$$c(\boldsymbol{x}(t), \boldsymbol{u}(t), t) = 0$$

for every t between them.
The reconstructed trajectory must be checked independently.

State Interpolation

For LGR collocation, the state approximation is uniquely defined by a Lagrange polynomial:

$$\boldsymbol{x}^{(N)}(\tau) = \sum_{j=1}^{N+1} \boldsymbol{X}_j \ell_j(\tau).$$

Thus, the state can be evaluated at dense points between the collocation nodes.
The corresponding physical time is

$$t(\tau) = \frac{t_f - t_0}{2}\tau + \frac{t_f + t_0}{2}.$$

Control Interpolation

The control is represented only by its values at the collocation points:

$$\boldsymbol{U}_1, \ldots, \boldsymbol{U}_N.$$

Unlike the state, there is no unique control interpolation implied by the collocation derivation.
Possible choices include:

- piecewise linear interpolation;
- piecewise constant interpolation;
- cubic splines;
- local polynomial interpolation;
- barycentric interpolation.

Different interpolation choices may produce different between-node behavior.

Hidden Constraint Violations

Even if

$$c(\boldsymbol{X}_i, \boldsymbol{U}_i, t_i) \leq 0$$

at every collocation point, an interpolated path may violate the constraint between nodes.

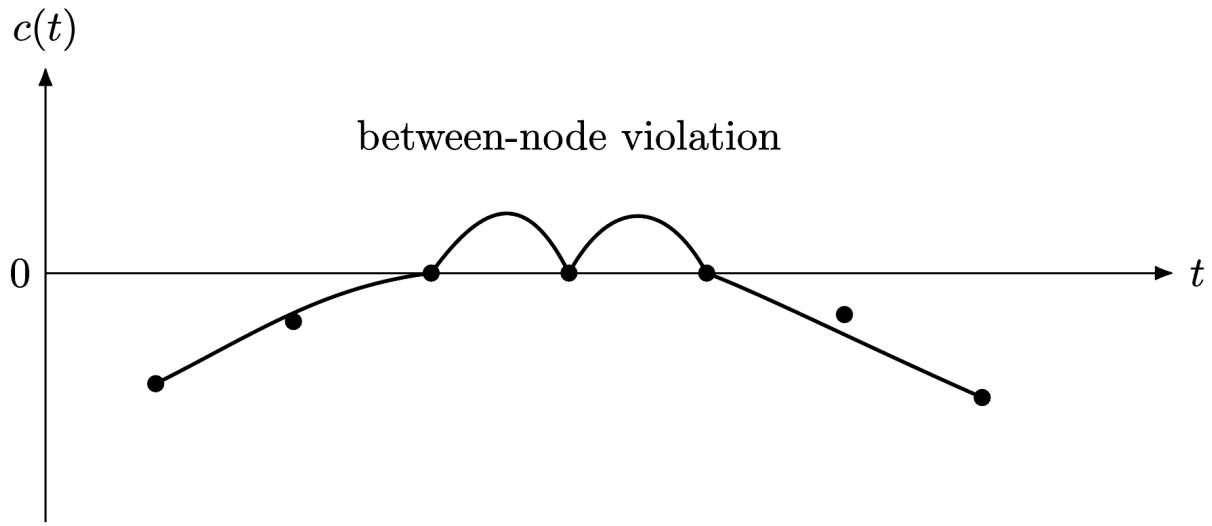

Figure 24: Discrete feasibility does not guarantee continuous feasibility.

This is a discretization error, not necessarily an NLP error.

Dense-Grid Verification

After solving the NLP:

1. reconstruct the state on a dense grid;
2. choose and document a control interpolation;
3. evaluate the path constraint on the dense grid;
4. compute the maximum violation;
5. compare it with a prescribed feasibility tolerance.

Define

$$v(t) = \max\left\{\boldsymbol{c}_{\min} - \boldsymbol{c}(\boldsymbol{x}(t), \boldsymbol{u}(t), t),\ \boldsymbol{c}(\boldsymbol{x}(t), \boldsymbol{u}(t), t) - \boldsymbol{c}_{\max},\ \boldsymbol{0}\right\}.$$

A useful scalar metric is

$$\boxed{v_{\max} = \max_{t \in [t_0, t_f]} \|v(t)\|_\infty.}$$

Mesh Refinement

If the reconstructed solution violates the path constraint beyond tolerance, increase the mesh resolution.
Refinement may involve:

- increasing the number of collocation points;
- splitting the time interval into multiple mesh segments;
- adding points near suspected entry and exit regions;
- using adaptive h-, p-, or hp-refinement.

The process is repeated until the continuous residuals and violations are acceptably small.

Theory-Based Verification

Direct collocation simplifies the formulation, but theory remains essential for verifying the result.
For an apparently active state constraint, one should check:

1. whether the constraint is approximately zero on an interval;
2. whether the lower-order tangency conditions hold near entry;
3. whether the boundary control relation is approximately satisfied;
4. whether the constraint multiplier has the correct activity pattern;
5. whether the solution converges under mesh refinement.

NLP Multipliers and Continuous Path Multipliers

The NLP solver returns multipliers associated with the discrete path constraints. These can be interpreted as approximations of the continuous path multiplier after appropriate scaling.
On inactive points, the discrete multiplier should be near zero. On active points, it may be nonzero and should satisfy discrete complementarity.
For an upper-bound constraint

$$c_i(\boldsymbol{z}) \leq 0,$$

the discrete KKT conditions include

$$\begin{aligned} c_i(\boldsymbol{z}^*) &\leq 0, \\ \mu_i^* &\geq 0, \\ \mu_i^* c_i(\boldsymbol{z}^*) &= 0. \end{aligned}$$

The sign depends on the NLP convention.

Equality Versus Inequality Constraints in an NLP

Defect constraints are equalities:

$$\boldsymbol{D}\boldsymbol{X} - \frac{t_f - t_0}{2}\boldsymbol{F} = \boldsymbol{0}.$$

The NLP solver knows these must be active.

Path inequalities may be active or inactive:

$$\boldsymbol{c}_{\min} \le \boldsymbol{c}_i \le \boldsymbol{c}_{\max}.$$

The solver must determine which bounds bind at the solution.

This active-set determination may be performed through active-set methods, sequential quadratic programming, or interior-point methods.

Implementation Pattern

A typical MATLAB nonlinear-constraint function may be written as:

```
function [c, ceq] = constraints(z, data)

    [X, U, t0, tf] = unpackDecisionVector(z, data);

    t = ((tf - t0)/2) * data.tau + (tf + t0)/2;

    F = dynamics(X(:,1:data.N), U, t, data);

    defects = data.D * X.' ...
            - ((tf - t0)/2) * F.';

    boundary = endpointConstraints( ...
        X(:,1), t0, X(:,end), tf, data);

    path = pathFunctions( ...
        X(:,1:data.N), U, t, data);

    % Example upper -bound form: path <= pathMax
    cUpper = path - data.pathMax;

    % Example lower -bound form: path >= pathMin
    cLower = data.pathMin - path;

    c = [cUpper(:); cLower(:)];
    ceq = [defects(:); boundary(:)];
end
```

The path constraints are evaluated only at the collocation points inside the NLP. Dense-grid checking is performed after optimization.

Recommended Numerical Workflow

1. Formulate the continuous optimal control problem.

2. Identify all path constraints and their physical meaning.
3. Solve an unconstrained or relaxed problem first when useful.
4. Discretize the states and controls using collocation.
5. Add the path inequalities at all collocation points.
6. Solve the NLP using a scaled initial guess.
7. Inspect the solver status and KKT information.
8. Reconstruct state and control trajectories.
9. Evaluate dynamics residuals on a dense grid.
10. Evaluate path constraints on a dense grid.
11. Inspect active intervals and approximate tangency.
12. Refine the mesh and resolve.
13. Compare successive solutions and objective values.

Theory and Computation Must Work Together

A numerical solver does not understand optimal-control theory. It solves the finite-dimensional equations supplied by the user.
Therefore, obtaining a solver message such as "optimal solution found" is not enough. One must verify:

- that the implemented equations represent the intended problem;
- that the discretization is sufficiently accurate;
- that path constraints hold between nodes;
- that the result is consistent with necessary conditions;
- that the solution is stable under mesh refinement.

Common Failure Modes

Incorrect Path Function

A sign error in the path constraint may reverse the feasible region.

Constraint Enforced at the Wrong Nodes

The control and dynamics may be defined only at the N Radau points, while the state includes an additional terminal point. Indexing errors are common.

Assuming Discrete Feasibility Implies Continuous Feasibility

Interpolation can violate the constraint between nodes.

Inadequate Mesh Resolution

Entry and exit regions may be poorly resolved.

Poor Control Interpolation

A high-order global interpolation may oscillate and create artificial violations.

Blind Trust in Solver Output

An NLP solver can converge to a local optimum, a poorly scaled point, or the solution of incorrectly coded equations.

Summary

1. State inequality constraints may be inactive or active.
2. A constrained arc is an interval on which the state constraint is exactly active.
3. The order of the constraint is the number of derivatives required for the control to appear.
4. Entry tangency conditions require

$$c^{(j)}(t_1) = 0, \qquad j = 0, \ldots, q-1.$$

5. On the active arc, the highest derivative condition determines the boundary control.
6. Indirect methods require assumptions about constrained-arc structure.
7. An incorrect active-set assumption may produce a suboptimal solution.
8. Direct collocation enforces path inequalities at discrete points.
9. The NLP solver determines the discrete active set.
10. Discrete feasibility does not guarantee continuous feasibility.
11. State and control interpolations must be checked on a dense grid.
12. Mesh refinement is required until constraint violations and residuals are acceptably small.
13. Continuous optimal-control theory is essential for validating direct numerical solutions.

Connection. Path-constrained trajectory optimization remains tied to a nominal initial condition; dynamic programming instead asks for optimal cost from every state and time.

The Principle of Optimality and the Hamilton–Jacobi–Bellman Equation

Earlier sections developed optimality conditions through the calculus of variations, Pontryagin's minimum principle, and direct numerical methods. Those approaches typically produce an open-loop optimal trajectory for a specified initial condition and model.

Dynamic programming takes a different viewpoint. Instead of solving only for one optimal trajectory from one initial condition, it asks:

> What is the minimum remaining cost from every possible state and every possible time?

The answer is encoded in the value function, also called the optimal cost-to-go. The partial differential equation satisfied by this function is the Hamilton–Jacobi–Bellman equation.

Open-Loop and Closed-Loop Optimal Control

An open-loop control is determined as a function of time:

$$\boldsymbol{u}^*(t).$$

It is computed for a particular initial state and a particular model. Once computed, the control does not explicitly change in response to deviations of the measured state.

A closed-loop control is a feedback law:

$$\boxed{\boldsymbol{u}^*(t) = \boldsymbol{\pi}^*(\boldsymbol{x}(t), t).}$$

The current state is used to determine the control. Dynamic programming naturally generates this feedback form because it computes the optimal action for every state-time pair.

The Principle of Optimality

Suppose an optimal trajectory connects point A to point B. Let D and E be any two points lying on that trajectory.

Then the segment of the trajectory connecting D to E must itself be optimal for the subproblem that begins at D and ends at E.

Bellman's Principle of Optimality
An optimal policy has the property that, regardless of the initial state and initial decision, the remaining decisions must constitute an optimal policy with respect to the state resulting from the first decision.

A shorter geometric statement is:

Every subarc of an optimal trajectory is optimal for its own endpoints.

What the Principle Does Not Mean

Suppose C is not on the optimal trajectory from A to B. Even if the path from A to C is optimal and the path from C to B is optimal, the concatenated path

$$A \to C \to B$$

need not be optimal from A to B.
The principle of optimality applies only to subarcs of the original optimal path.

Geometric Illustration

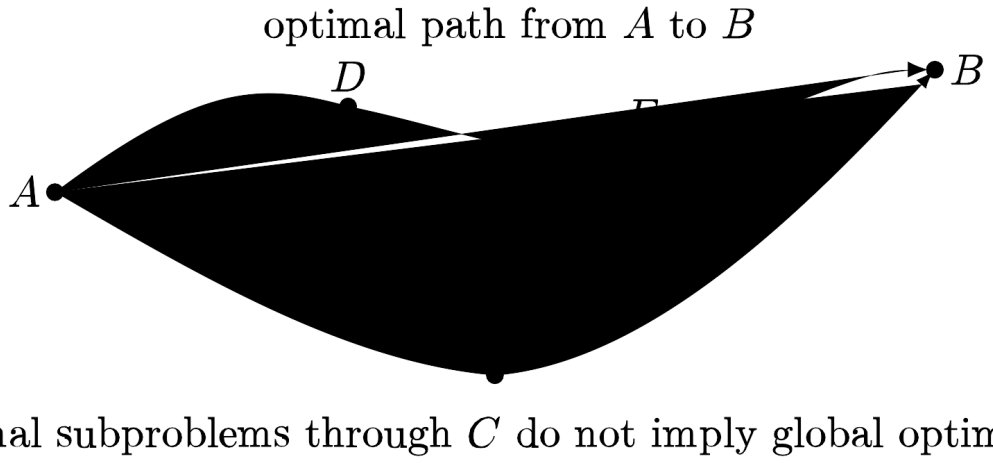

Figure 25: The principle of optimality applies to subarcs of the optimal trajectory.

Planning Backward and Executing Forward

Dynamic programming is based on a backward-planning idea:

1. Start from the terminal objective.
2. Determine the minimum remaining cost from states near the terminal time.
3. Continue backward in time.
4. Once the full policy is known, execute it forward in time.

This is often summarized by the statement:

> Life is lived forward, but it is planned backward.

In control, one determines the optimal cost remaining from the current state to the terminal time. If the system deviates from the nominal path, the same value function provides the new optimal action from the new state.

Continuous-Time Optimal Control Problem

Consider

$$\begin{aligned} \dot{\boldsymbol{x}}(t) &= \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t), t), \\ \boldsymbol{x}(t_0) &= \boldsymbol{x}_0, \end{aligned}$$

with Bolza cost

$$J = S(\boldsymbol{x}(t_f), t_f) + \int_{t_0}^{t_f} V(\boldsymbol{x}(t), \boldsymbol{u}(t), t)\,\mathrm{d}t.$$

The objective is to minimize J over admissible controls.

The Cost-to-Go Function

Instead of starting only at $(\boldsymbol{x}_0, t_0)$, define a problem beginning at an arbitrary state $\boldsymbol{x}$ and time t:

$$J(\boldsymbol{x}, t; \boldsymbol{u}) = S(\boldsymbol{x}(t_f), t_f) + \int_{t}^{t_f} V(\boldsymbol{x}(\tau), \boldsymbol{u}(\tau), \tau)\,\mathrm{d}\tau.$$

The optimal cost-to-go, or value function, is

$$\boxed{J^*(\boldsymbol{x}, t) = \min_{\boldsymbol{u}(\cdot)} \left[S(\boldsymbol{x}(t_f), t_f) + \int_{t}^{t_f} V(\boldsymbol{x}(\tau), \boldsymbol{u}(\tau), \tau)\,\mathrm{d}\tau \right].}$$

The terminal boundary condition is

$$\boxed{J^*(\boldsymbol{x}, t_f) = S(\boldsymbol{x}, t_f).}$$

Interpretation of the Value Function

The value function answers the question:

> If the system is currently at state $\boldsymbol{x}$ at time t, what is the smallest possible cost that remains from now until the terminal time?

At the initial condition,

$$J^*(\boldsymbol{x}_0, t_0)$$

is the optimal value of the original problem.
Along an optimal trajectory,

$$J^*(\boldsymbol{x}^*(t), t)$$

decreases as cost is accumulated and eventually reaches the terminal cost.

Total Derivative Along the Optimal Trajectory

Along an optimal trajectory,

$$J^* = J^*(\boldsymbol{x}^*(t), t).$$

Its total derivative is

$$\frac{\mathrm{d}J^*}{\mathrm{d}t} = \frac{\partial J^*}{\partial t} + \left(\frac{\partial J^*}{\partial \boldsymbol{x}}\right)^{\mathsf{T}} \dot{\boldsymbol{x}}^*(t).$$

Because

$$J^*(\boldsymbol{x}^*(t), t) = S(\boldsymbol{x}^*(t_f), t_f) + \int_t^{t_f} V(\boldsymbol{x}^*(\tau), \boldsymbol{u}^*(\tau), \tau)\,\mathrm{d}\tau,$$

differentiating with respect to the lower integration limit gives

$$\frac{\mathrm{d}J^*}{\mathrm{d}t} = -V(\boldsymbol{x}^*(t), \boldsymbol{u}^*(t), t).$$

Therefore,

$$-V(\boldsymbol{x}^*, \boldsymbol{u}^*, t) = \frac{\partial J^*}{\partial t} + \left(\frac{\partial J^*}{\partial \boldsymbol{x}}\right)^{\mathsf{T}} \dot{\boldsymbol{x}}^*.$$

Substituting

$$\dot{\boldsymbol{x}}^* = \boldsymbol{f}(\boldsymbol{x}^*, \boldsymbol{u}^*, t)$$

yields

$$\boxed{\frac{\partial J^*}{\partial t} + V(\boldsymbol{x}^*, \boldsymbol{u}^*, t) + \left(\frac{\partial J^*}{\partial \boldsymbol{x}}\right)^{\mathsf{T}} \boldsymbol{f}(\boldsymbol{x}^*, \boldsymbol{u}^*, t) = 0.}$$

Hamiltonian Form

Define the Hamiltonian

$$\mathcal{H}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{p}, t) = V(\boldsymbol{x}, \boldsymbol{u}, t) + \boldsymbol{p}^{\mathsf{T}} \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, t),$$

where

$$\boldsymbol{p} = \nabla_{\boldsymbol{x}} J^*(\boldsymbol{x}, t).$$

Then, along the optimal trajectory,

$$\frac{\partial J^*}{\partial t} + \mathcal{H}\left(\boldsymbol{x}^*, \boldsymbol{u}^*, \nabla_{\boldsymbol{x}} J^*, t\right) = 0.$$

Because the control must minimize the Hamiltonian, the general HJB equation is

$$\boxed{\frac{\partial J^*}{\partial t} + \min_{\boldsymbol{u}} \left[V(\boldsymbol{x}, \boldsymbol{u}, t) + (\nabla_{\boldsymbol{x}} J^*)^{\mathsf{T}} \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, t) \right] = 0.}$$

The terminal condition is

$$\boxed{J^*(\boldsymbol{x}, t_f) = S(\boldsymbol{x}, t_f).}$$

The Hamilton–Jacobi–Bellman Equation

The compact Hamiltonian form is

$$\boxed{J_t^*(\boldsymbol{x}, t) + \min_{\boldsymbol{u}} \mathcal{H}\left(\boldsymbol{x}, \boldsymbol{u}, \nabla_{\boldsymbol{x}} J^*, t\right) = 0.}$$

This is a nonlinear first-order partial differential equation with a terminal boundary condition.

Why the HJB Equation Is a Backward-Time Equation

The boundary condition is given at t_f, not at t_0:

$$J^*(\boldsymbol{x}, t_f) = S(\boldsymbol{x}, t_f).$$

Therefore, the HJB equation is integrated backward from the terminal time toward the initial time.

This matches the cost-to-go interpretation: one begins with the cost remaining at the end and works backward to determine the cost remaining at all earlier times.

Feedback Control from the HJB Equation

Once $J^*(\boldsymbol{x}, t)$ is known, the optimal control is obtained from

$$\boxed{\boldsymbol{u}^*(\boldsymbol{x}, t) = \arg\min_{\boldsymbol{u}} \left[V(\boldsymbol{x}, \boldsymbol{u}, t) + (\nabla_{\boldsymbol{x}} J^*)^{\mathsf{T}} \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, t) \right].}$$

This is a state-feedback law.

The central advantage is that the solution is defined for all states in the domain, not merely along one nominal trajectory.

Relationship to the Costate

In Pontryagin's minimum principle,

$$\boldsymbol{\lambda}^*(t)$$

is the costate.

In dynamic programming,

$$\nabla_x J^*(x, t)$$

is the gradient of the value function.
Along an optimal trajectory,

$$\boxed{\lambda^*(t) = \nabla_x J^*(x^*(t), t).}$$

Thus, the costate is the gradient of the optimal cost-to-go evaluated along the optimal path.

Connection to the Costate Equation

Differentiate

$$\lambda^*(t) = \nabla_x J^*(x^*(t), t)$$

with respect to time:

$$\dot{\lambda}^*(t) = \frac{\mathrm{d}}{\mathrm{d}t} \nabla_x J^*(x^*(t), t).$$

The HJB equation implies the same costate dynamics as Pontryagin's minimum principle:

$$\boxed{\dot{\lambda}^* = -\frac{\partial \mathcal{H}}{\partial x}.}$$

Therefore, the HJB equation and the minimum principle are consistent descriptions of optimality.

Why HJB Is More General Than a Single Open-Loop Solution

Pontryagin's principle typically produces an extremal trajectory for a specified initial state.
The HJB equation produces the full value function

$$J^*(x, t),$$

which contains the optimal cost from every state and time.
Consequently, the HJB equation can generate feedback control even after disturbances or modeling errors move the system away from the nominal trajectory.

Replanning Interpretation

Suppose the nominal trajectory is interrupted by an unexpected change. The current state becomes

$$x_{\text{new}}.$$

One does not need to return to the original initial condition. Instead, the optimal policy evaluates

$$\boldsymbol{u}^*(\boldsymbol{x}_{\text{new}}, t)$$

and continues optimally from the current state.

This is the mathematical version of replanning a route after encountering a road closure.

Computational Challenge

The HJB equation is a PDE in state and time. If the state dimension is n_x, then the value function is defined over an $(n_x + 1)$-dimensional domain.

The computational burden grows rapidly with n_x. This is known as the curse of dimensionality.

Curse of Dimensionality
A grid with M points per state dimension requires approximately M^{n_x} state-grid points. The computational cost therefore grows exponentially with the number of states.

This makes direct solution of the HJB equation impractical for many high-dimensional systems.

Introductory Scalar Linear–Quadratic Example

Consider

$$\boxed{J = \frac{1}{2}x^2(t_f) + \frac{1}{2}\int_0^{t_f} \left[x^2(t) + u^2(t)\right]\, \mathrm{d}t}$$

subject to

$$\boxed{\dot{x}(t) = -2x(t) + u(t), \qquad x(0) = x_0.}$$

The running cost is

$$V(x, u, t) = \frac{1}{2}x^2 + \frac{1}{2}u^2.$$

The terminal cost is

$$S(x(t_f), t_f) = \frac{1}{2}x^2(t_f).$$

The dynamics are

$$f(x, u, t) = -2x + u.$$

Hamiltonian for the Example

The HJB Hamiltonian is

$$\mathcal{H}(x, u, J_x, t) = \frac{1}{2}x^2 + \frac{1}{2}u^2 + J_x(-2x + u).$$

The minimizing control satisfies

$$\frac{\partial \mathcal{H}}{\partial u} = u + J_x = 0.$$

Therefore,

$$\boxed{u^*(x, t) = -J_x^*(x, t).}$$

Substituting into the Hamiltonian gives

$$\begin{aligned}\mathcal{H}^* &= \frac{1}{2}x^2 + \frac{1}{2}(J_x^*)^2 + J_x^*\,(-2x - J_x^*) \\ &= \frac{1}{2}x^2 - \frac{1}{2}(J_x^*)^2 - 2xJ_x^*.\end{aligned}$$

Hence, the HJB equation is

$$\boxed{J_t^* + \frac{1}{2}x^2 - \frac{1}{2}(J_x^*)^2 - 2xJ_x^* = 0.}$$

The terminal condition is

$$\boxed{J^*(x, t_f) = \frac{1}{2}x^2.}$$

Quadratic Value-Function Ansatz

Because the dynamics are linear and the cost is quadratic, assume

$$\boxed{J^*(x, t) = \frac{1}{2}P(t)x^2.}$$

The terminal condition gives

$$\frac{1}{2}P(t_f)x^2 = \frac{1}{2}x^2,$$

so

$$\boxed{P(t_f) = 1.}$$

The derivatives are

$$J_x^* = P(t)x,$$
$$J_t^* = \frac{1}{2}\dot{P}(t)x^2.$$

Optimal Feedback Law

Using

$$u^* = -J_x^*,$$

the optimal control becomes

$$\boxed{u^*(t) = -P(t)x^*(t).}$$

This is a time-varying linear state-feedback law.
The feedback gain is

$$K(t) = P(t).$$

Riccati Differential Equation

Substitute the quadratic ansatz into the HJB equation:

$$\frac{1}{2}\dot{P}x^2 + \frac{1}{2}x^2 - \frac{1}{2}P^2x^2 - 2Px^2 = 0.$$

For nonzero x, divide by $\frac{1}{2}x^2$:

$$\dot{P} + 1 - P^2 - 4P = 0.$$

Therefore,

$$\boxed{\dot{P} = P^2 + 4P - 1, \qquad P(t_f) = 1.}$$

This differential equation is integrated backward in time.

Closed-Loop State Dynamics

Substituting

$$u^* = -P(t)x$$

into the system dynamics gives

$$\boxed{\dot{x}^*(t) = -\left[2 + P(t)\right] x^*(t).}$$

Once $P(t)$ is found backward in time, the optimal state can be propagated forward from

$$x(0) = x_0.$$

Backward Planning and Forward Execution

The solution procedure has two stages:

1. Integrate the Riccati equation backward from

$$P(t_f) = 1.$$

2. Integrate the closed-loop state equation forward from

$$x(0) = x_0.$$

This is the precise mathematical realization of planning backward and executing forward.

Numerical MATLAB Template

```
tf = 5;

% Backward Riccati equation
riccati = @(t,P) P.^2 + 4*P - 1;

% Integrate backward from P(tf) = 1
[tP, P] = ode45(riccati, [tf 0], 1);

% Reverse arrays for interpolation in increasing time
tP = flipud(tP);
P  = flipud(P);

Pfun = @(t) interp1(tP, P, t, 'pchip');

% Initial state
x0 = 1;

% Closed -loop dynamics
closedLoop = @(t,x) -(2 + Pfun(t))*x;

% Integrate state forward
[tx, x] = ode45(closedLoop, [0 tf], x0);
```

```
% Optimal control
u = -arrayfun(Pfun, tx).*x;
```

Verification Checks

For the scalar example, verify:

1. $P(t_f) = 1$;
2. the Riccati residual is small;
3. $u^*(t) = -P(t)x^*(t)$;
4. the state satisfies

$$\dot{x}^* = -2x^* + u^*;$$

5. the HJB residual is small;
6. the value function decreases consistently with the accumulated running cost.

HJB Residual

For an approximate value function $\widehat{J}(x,t)$ and control $\widehat{u}(x,t)$, define

$$R_{\mathrm{HJB}} = \widehat{J}_t + V(x,\widehat{u},t) + \widehat{J}_x f(x,\widehat{u},t).$$

A high-quality approximation should satisfy

$$\boxed{R_{\mathrm{HJB}} \approx 0.}$$

Comparison with Pontryagin's Minimum Principle

For this example, the Hamiltonian in the minimum principle is

$$\mathcal{H} = \frac{1}{2}x^2 + \frac{1}{2}u^2 + \lambda(-2x + u).$$

Stationarity gives

$$u^* = -\lambda.$$

Dynamic programming gives

$$u^* = -J_x^*.$$

Therefore,

$$\boxed{\lambda(t) = J_x^*(x^*(t), t) = P(t)x^*(t).}$$

The two methods produce the same optimal control.

Summary

1. The principle of optimality states that every subarc of an optimal trajectory is itself optimal.
2. The value function is the minimum remaining cost from a state-time pair.
3. The terminal condition is specified at t_f.
4. The HJB equation is solved backward in time.
5. The general continuous-time HJB equation is

$$J_t^* + \min_{\boldsymbol{u}} \left[V + (\nabla_{\boldsymbol{x}} J^*)^\mathsf{T} \boldsymbol{f} \right] = 0.$$

6. The optimal feedback law is obtained by minimizing the HJB Hamiltonian.
7. Along the optimal trajectory,

$$\boldsymbol{\lambda}^* = \nabla_x J^*.$$

8. Dynamic programming produces a feedback law rather than only one open-loop trajectory.
9. The HJB equation suffers from the curse of dimensionality.
10. For linear systems with quadratic costs, a quadratic value function leads to a Riccati differential equation.
11. The scalar example produces

$$u^*(t) = -P(t)x^*(t).$$

Connection. Applying the HJB equation to finite-horizon LQR shows how a quadratic value function turns a nonlinear PDE into a matrix differential equation.

HJB Formulation of the Finite-Horizon Linear Quadratic Regulator

This session applies the Hamilton–Jacobi–Bellman equation to the finite-horizon linear quadratic regulator (LQR). The objective is to recover, using dynamic programming, the same matrix Riccati differential equation and state-feedback law obtained previously from the calculus of variations and Pontryagin's minimum principle.

The importance of this derivation is not merely algebraic. It shows that the Riccati equation is a direct consequence of the HJB partial differential equation when the system is linear and the cost is quadratic.

Finite-Horizon LQR Problem

Consider the time-varying linear system

$$\boxed{\dot{\boldsymbol{x}}(t) = \boldsymbol{A}(t)\boldsymbol{x}(t) + \boldsymbol{B}(t)\boldsymbol{u}(t), \qquad \boldsymbol{x}(t_0) = \boldsymbol{x}_0.}$$

The finite-horizon quadratic cost is

$$\boxed{J = \frac{1}{2}\boldsymbol{x}^{\mathsf{T}}(t_f)\boldsymbol{S}_f\boldsymbol{x}(t_f) + \frac{1}{2}\int_{t_0}^{t_f} \left[\boldsymbol{x}^{\mathsf{T}}(t)\boldsymbol{Q}(t)\boldsymbol{x}(t) + \boldsymbol{u}^{\mathsf{T}}(t)\boldsymbol{R}(t)\boldsymbol{u}(t)\right] \mathrm{d}t.}$$

Assume

$$\begin{aligned} \boldsymbol{Q}(t) &= \boldsymbol{Q}^{\mathsf{T}}(t) \succeq 0, \\ \boldsymbol{R}(t) &= \boldsymbol{R}^{\mathsf{T}}(t) \succ 0, \\ \boldsymbol{S}_f &= \boldsymbol{S}_f^{\mathsf{T}} \succeq 0. \end{aligned}$$

The positive definiteness of $\boldsymbol{R}$ guarantees that the Hamiltonian is strictly convex in the control.

Structural Assumptions

For the classical finite-horizon LQR problem, one often assumes that

- $(\boldsymbol{A}(t), \boldsymbol{B}(t))$ is controllable or stabilizable;
- $(\boldsymbol{A}(t), \boldsymbol{Q}^{1/2}(t))$ is observable or detectable;
- all matrices are sufficiently smooth functions of time.

These assumptions ensure that the problem is well posed and that the resulting feedback law has the expected properties.

Cost-to-Go Function

At an arbitrary state $\boldsymbol{x}$ and time t, define the optimal cost-to-go

$$J^*(\boldsymbol{x}, t) = \min_{\boldsymbol{u}(\cdot)} \left[\frac{1}{2}\boldsymbol{x}^{\mathsf{T}}(t_f)\boldsymbol{S}_f\boldsymbol{x}(t_f) + \frac{1}{2}\int_t^{t_f} \left(\boldsymbol{x}^{\mathsf{T}}\boldsymbol{Q}\boldsymbol{x} + \boldsymbol{u}^{\mathsf{T}}\boldsymbol{R}\boldsymbol{u}\right) \mathrm{d}\tau\right].$$

The terminal condition is

$$\boxed{J^*(\boldsymbol{x}, t_f) = \frac{1}{2}\boldsymbol{x}^\top \boldsymbol{S}_f \boldsymbol{x}.}$$

Hamiltonian in the HJB Formulation

The running cost is

$$L(\boldsymbol{x}, \boldsymbol{u}, t) = \frac{1}{2}\boldsymbol{x}^\top \boldsymbol{Q}\boldsymbol{x} + \frac{1}{2}\boldsymbol{u}^\top \boldsymbol{R}\boldsymbol{u}.$$

The dynamics are

$$\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, t) = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u}.$$

The HJB Hamiltonian is therefore

$$\boxed{\mathcal{H} = \frac{1}{2}\boldsymbol{x}^\top \boldsymbol{Q}\boldsymbol{x} + \frac{1}{2}\boldsymbol{u}^\top \boldsymbol{R}\boldsymbol{u} + J_{\boldsymbol{x}}^{*\top}(\boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u}).}$$

Here,

$$J_{\boldsymbol{x}}^* = \nabla_{\boldsymbol{x}} J^*(\boldsymbol{x}, t).$$

Optimal Control from Hamiltonian Minimization

The optimal control minimizes the Hamiltonian:

$$\boldsymbol{u}^* = \arg\min_{\boldsymbol{u}} \mathcal{H}.$$

Differentiate with respect to $\boldsymbol{u}$:

$$\frac{\partial \mathcal{H}}{\partial \boldsymbol{u}} = \boldsymbol{R}\boldsymbol{u} + \boldsymbol{B}^\top J_{\boldsymbol{x}}^*.$$

The first-order condition is

$$\boldsymbol{R}\boldsymbol{u}^* + \boldsymbol{B}^\top J_{\boldsymbol{x}}^* = \boldsymbol{0}.$$

Because $\boldsymbol{R} \succ 0$,

$$\boxed{\boldsymbol{u}^* = -\boldsymbol{R}^{-1}\boldsymbol{B}^\top J_{\boldsymbol{x}}^*.}$$

This is already a feedback relationship, although the value-function gradient is not yet known.

Reduced Optimal Hamiltonian

Substitute

$$\boldsymbol{u}^* = -\boldsymbol{R}^{-1}\boldsymbol{B}^\mathsf{T} J_x^*$$

into the Hamiltonian.
First,

$$\frac{1}{2}\boldsymbol{u}^{*\mathsf{T}}\boldsymbol{R}\boldsymbol{u}^* = \frac{1}{2}J_x^{*\mathsf{T}}\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^\mathsf{T} J_x^*.$$

Second,

$$J_x^{*\mathsf{T}}\boldsymbol{B}\boldsymbol{u}^* = -J_x^{*\mathsf{T}}\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^\mathsf{T} J_x^*.$$

Therefore,

$$\boxed{\mathcal{H}^* = \frac{1}{2}\boldsymbol{x}^\mathsf{T}\boldsymbol{Q}\boldsymbol{x} - \frac{1}{2}J_x^{*\mathsf{T}}\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^\mathsf{T} J_x^* + J_x^{*\mathsf{T}}\boldsymbol{A}\boldsymbol{x}.}$$

Hamilton–Jacobi–Bellman PDE

The HJB equation is

$$J_t^* + \mathcal{H}^* = 0.$$

Thus,

$$\boxed{J_t^* + \frac{1}{2}\boldsymbol{x}^\mathsf{T}\boldsymbol{Q}\boldsymbol{x} - \frac{1}{2}J_x^{*\mathsf{T}}\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^\mathsf{T} J_x^* + J_x^{*\mathsf{T}}\boldsymbol{A}\boldsymbol{x} = 0.}$$

The terminal boundary condition is

$$\boxed{J^*(\boldsymbol{x}, t_f) = \frac{1}{2}\boldsymbol{x}^\mathsf{T}\boldsymbol{S}_f\boldsymbol{x}.}$$

Classification of the PDE

The HJB equation is a first-order nonlinear partial differential equation:

- first order in time because it contains J_t^*;
- first order in the state because it contains J_x^*;
- nonlinear because J_x^* appears quadratically.

The terminal condition is specified at t_f, so the problem is solved backward in time.

Why an Additional Structural Assumption Is Needed

The PDE contains derivatives with respect to both time and state. For a general nonlinear problem, one must solve the PDE over the full state-time domain.

The LQR problem is special because the terminal cost is quadratic and the system and running cost preserve quadratic structure. This motivates a quadratic value function.

Quadratic Value-Function Ansatz

Assume

$$\boxed{J^*(\boldsymbol{x}, t) = \frac{1}{2}\boldsymbol{x}^\mathsf{T} \boldsymbol{S}(t)\boldsymbol{x},}$$

where

$$\boldsymbol{S}(t) = \boldsymbol{S}^\mathsf{T}(t).$$

This form is consistent with the terminal boundary condition.
At $t = t_f$,

$$\frac{1}{2}\boldsymbol{x}^\mathsf{T} \boldsymbol{S}(t_f)\boldsymbol{x} = \frac{1}{2}\boldsymbol{x}^\mathsf{T} \boldsymbol{S}_f \boldsymbol{x}.$$

Hence,

$$\boldsymbol{x}^\mathsf{T} \left[\boldsymbol{S}(t_f) - \boldsymbol{S}_f\right] \boldsymbol{x} = 0 \quad \text{for all } \boldsymbol{x}.$$

Therefore,

$$\boxed{\boldsymbol{S}(t_f) = \boldsymbol{S}_f.}$$

State Gradient of the Value Function

Since $\boldsymbol{S}$ is symmetric,

$$\boxed{J_{\boldsymbol{x}}^* = \boldsymbol{S}(t)\boldsymbol{x}.}$$

To see this,

$$\frac{\partial}{\partial \boldsymbol{x}} \left(\frac{1}{2}\boldsymbol{x}^\mathsf{T} \boldsymbol{S}\boldsymbol{x}\right) = \frac{1}{2}(\boldsymbol{S} + \boldsymbol{S}^\mathsf{T})\boldsymbol{x} = \boldsymbol{S}\boldsymbol{x}.$$

Time Derivative of the Value Function

The partial derivative with respect to time holds $\boldsymbol{x}$ fixed:

$$\boxed{J_t^* = \frac{1}{2}\boldsymbol{x}^\top \dot{\boldsymbol{S}}(t)\boldsymbol{x}.}$$

Only the explicit time dependence of $\boldsymbol{S}(t)$ is differentiated.

Substitution into the HJB Equation

Substituting

$$J_x^* = \boldsymbol{S}\boldsymbol{x},$$
$$J_t^* = \frac{1}{2}\boldsymbol{x}^\top \dot{\boldsymbol{S}}\boldsymbol{x}$$

into the HJB equation gives

$$0 = \frac{1}{2}\boldsymbol{x}^\top \dot{\boldsymbol{S}}\boldsymbol{x} + \frac{1}{2}\boldsymbol{x}^\top \boldsymbol{Q}\boldsymbol{x} - \frac{1}{2}\boldsymbol{x}^\top \boldsymbol{S}\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^\top \boldsymbol{S}\boldsymbol{x} + \boldsymbol{x}^\top \boldsymbol{S}\boldsymbol{A}\boldsymbol{x}.$$

The final term is scalar. Therefore,

$$\boldsymbol{x}^\top \boldsymbol{S}\boldsymbol{A}\boldsymbol{x} = \frac{1}{2}\boldsymbol{x}^\top \left(\boldsymbol{S}\boldsymbol{A} + \boldsymbol{A}^\top \boldsymbol{S}\right)\boldsymbol{x}.$$

Symmetrization of the Cross Term

For any square matrix $\boldsymbol{M}$,

$$\boldsymbol{x}^\top \boldsymbol{M}\boldsymbol{x} = \boldsymbol{x}^\top \left[\frac{1}{2}(\boldsymbol{M} + \boldsymbol{M}^\top)\right]\boldsymbol{x}.$$

The skew-symmetric part contributes nothing because

$$\boldsymbol{x}^\top \left[\frac{1}{2}(\boldsymbol{M} - \boldsymbol{M}^\top)\right]\boldsymbol{x} = 0.$$

Applying this to $\boldsymbol{M} = \boldsymbol{S}\boldsymbol{A}$ gives

$$\boldsymbol{x}^\top \boldsymbol{S}\boldsymbol{A}\boldsymbol{x} = \frac{1}{2}\boldsymbol{x}^\top (\boldsymbol{S}\boldsymbol{A} + \boldsymbol{A}^\top \boldsymbol{S})\boldsymbol{x}.$$

Quadratic-Form Identity

The HJB equation becomes

$$\frac{1}{2}\boldsymbol{x}^\top \left[\dot{\boldsymbol{S}} + \boldsymbol{Q} - \boldsymbol{S}\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^\top \boldsymbol{S} + \boldsymbol{S}\boldsymbol{A} + \boldsymbol{A}^\top \boldsymbol{S}\right]\boldsymbol{x} = 0.$$

Because this must hold for every $\boldsymbol{x}$,

$$\boxed{\dot{S} + Q - SBR^{-1}B^{\mathsf{T}}S + SA + A^{\mathsf{T}}S = 0.}$$

Part II develops the Riccati equation, feedback law, implementation, interpretation, and verification.

Summary

1. The finite-horizon LQR problem has linear dynamics and quadratic cost.
2. The HJB Hamiltonian is minimized with respect to the control.
3. The minimizing control is

$$u^* = -R^{-1}B^{\mathsf{T}}J_x^*.$$

4. The HJB PDE is nonlinear in the value-function gradient.
5. A quadratic value-function ansatz is consistent with the problem structure.
6. The terminal condition gives

$$S(t_f) = S_f.$$

7. Substitution reduces the HJB PDE to a matrix differential equation.

Connection. The quadratic ansatz now yields the Riccati equation, the optimal feedback law, and a practical backward-in-time numerical procedure.

Riccati Equation, Optimal Feedback Law, and Numerical Implementation

Matrix Riccati Differential Equation

From the quadratic-form identity derived in Part I,

$$\dot{\boldsymbol{S}} + \boldsymbol{Q} - \boldsymbol{S}\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^{\mathsf{T}}\boldsymbol{S} + \boldsymbol{S}\boldsymbol{A} + \boldsymbol{A}^{\mathsf{T}}\boldsymbol{S} = \boldsymbol{0}.$$

Solving for $\dot{\boldsymbol{S}}$ gives the differential Riccati equation:

$$\boxed{\dot{\boldsymbol{S}} = -\boldsymbol{A}^{\mathsf{T}}\boldsymbol{S} - \boldsymbol{S}\boldsymbol{A} + \boldsymbol{S}\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^{\mathsf{T}}\boldsymbol{S} - \boldsymbol{Q}.}$$

The terminal condition is

$$\boxed{\boldsymbol{S}(t_f) = \boldsymbol{S}_f.}$$

This equation is integrated backward in time.

Equivalent Backward-Time Form

Define the backward-time variable

$$\tau = t_f - t.$$

Then

$$\frac{\mathrm{d}\boldsymbol{S}}{\mathrm{d}\tau} = -\frac{\mathrm{d}\boldsymbol{S}}{\mathrm{d}t}.$$

Therefore,

$$\boxed{\frac{\mathrm{d}\boldsymbol{S}}{\mathrm{d}\tau} = \boldsymbol{A}^{\mathsf{T}}\boldsymbol{S} + \boldsymbol{S}\boldsymbol{A} - \boldsymbol{S}\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^{\mathsf{T}}\boldsymbol{S} + \boldsymbol{Q},}$$

with

$$\boldsymbol{S}(0) = \boldsymbol{S}_f.$$

This form can be convenient for numerical integration because the independent variable increases forward.

Optimal Feedback Law

From Part I,

$$\boldsymbol{u}^* = -\boldsymbol{R}^{-1}\boldsymbol{B}^{\mathsf{T}}\boldsymbol{J}_{\boldsymbol{x}}^*.$$

Using

$$J_x^* = \boldsymbol{S}(t)\boldsymbol{x},$$

we obtain

$$\boxed{\boldsymbol{u}^*(t) = -\boldsymbol{R}^{-1}(t)\boldsymbol{B}^{\mathsf{T}}(t)\boldsymbol{S}(t)\boldsymbol{x}^*(t).}$$

Define the time-varying feedback gain

$$\boxed{\boldsymbol{K}(t) = \boldsymbol{R}^{-1}(t)\boldsymbol{B}^{\mathsf{T}}(t)\boldsymbol{S}(t).}$$

Then

$$\boxed{\boldsymbol{u}^*(t) = -\boldsymbol{K}(t)\boldsymbol{x}^*(t).}$$

Closed-Loop Dynamics

Substituting the feedback law into the state equation gives

$$\begin{aligned}\dot{\boldsymbol{x}}^* &= \boldsymbol{A}\boldsymbol{x}^* + \boldsymbol{B}\boldsymbol{u}^* \\ &= \left[\boldsymbol{A} - \boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^{\mathsf{T}}\boldsymbol{S}\right]\boldsymbol{x}^*.\end{aligned}$$

Therefore,

$$\boxed{\dot{\boldsymbol{x}}^*(t) = \left[\boldsymbol{A}(t) - \boldsymbol{B}(t)\boldsymbol{K}(t)\right]\boldsymbol{x}^*(t).}$$

Once $\boldsymbol{S}(t)$ has been computed backward in time, the closed-loop state is propagated forward from $\boldsymbol{x}(t_0) = \boldsymbol{x}_0$.

Backward Planning and Forward Execution

The LQR solution consists of two distinct integrations:

1. Integrate the Riccati equation backward:

$$\boldsymbol{S}(t_f) = \boldsymbol{S}_f.$$

2. Compute $\boldsymbol{K}(t)$ from $\boldsymbol{S}(t)$.
3. Integrate the closed-loop state equation forward:

$$\boldsymbol{x}(t_0) = \boldsymbol{x}_0.$$

4. Recover the optimal control:

$$u^*(t) = -K(t)x^*(t).$$

This is the exact mathematical realization of Bellman's idea: plan backward and execute forward.

Interpretation of the Matrix $S(t)$

The value function is

$$J^*(x,t) = \frac{1}{2}x^\top S(t)x.$$

Thus, $S(t)$ describes the local curvature of the optimal cost-to-go with respect to the state.

Large eigenvalues of $S(t)$ indicate directions in state space for which deviations are expensive. Small eigenvalues indicate directions that are relatively inexpensive.

Relationship to the Costate

Along the optimal trajectory,

$$\lambda^*(t) = \nabla_x J^*(x^*(t),t).$$

Hence,

$$\boxed{\lambda^*(t) = S(t)x^*(t).}$$

Substituting this into the Pontryagin control law

$$u^* = -R^{-1}B^\top\lambda^*$$

recovers the HJB feedback law.

Consistency with Pontryagin's Minimum Principle

For the LQR problem, the PMP Hamiltonian is

$$\mathcal{H} = \frac{1}{2}x^\top Qx + \frac{1}{2}u^\top Ru + \lambda^\top(Ax + Bu).$$

Stationarity gives

$$u^* = -R^{-1}B^\top\lambda.$$

The costate equation is

$$\dot{\lambda} = -Qx - A^\top\lambda.$$

Assume

$$\boldsymbol{\lambda} = \boldsymbol{S}\boldsymbol{x}.$$

Then

$$\dot{\boldsymbol{\lambda}} = \dot{\boldsymbol{S}}\boldsymbol{x} + \boldsymbol{S}\dot{\boldsymbol{x}}.$$

Substituting the closed-loop dynamics and equating terms yields the same differential Riccati equation.
Therefore, HJB and PMP are fully consistent for the LQR problem.

Why the LQR Problem Is Special

The LQR problem is unusually tractable because:

- the dynamics are linear;
- the cost is quadratic;
- the optimal value function remains quadratic;
- the HJB PDE reduces to a matrix ODE;
- the optimal control becomes a linear feedback law.

A small nonlinear perturbation may destroy this structure and return the problem to a high-dimensional nonlinear PDE.

Curse of Dimensionality

For a general nonlinear system with n_x states, the HJB equation is defined over an (n_x+1)-dimensional state-time domain.
If each state dimension is discretized with M grid points, the number of state-grid points is approximately

$$M^{n_x}.$$

This exponential growth is the curse of dimensionality.
The LQR problem avoids this difficulty because the unknown value function is represented by the matrix $\boldsymbol{S}(t)$ rather than by values on a full state-space grid.

Time-Varying Versus Time-Invariant LQR

For time-varying matrices, the Riccati equation is

$$\dot{\boldsymbol{S}} = -\boldsymbol{A}^{\mathsf{T}}\boldsymbol{S} - \boldsymbol{S}\boldsymbol{A} + \boldsymbol{S}\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^{\mathsf{T}}\boldsymbol{S} - \boldsymbol{Q}.$$

For an infinite-horizon time-invariant problem, one seeks a constant solution $\boldsymbol{S}_\infty$ satisfying

$$\boxed{\boldsymbol{A}^{\mathsf{T}}\boldsymbol{S}_\infty + \boldsymbol{S}_\infty\boldsymbol{A} - \boldsymbol{S}_\infty\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^{\mathsf{T}}\boldsymbol{S}_\infty + \boldsymbol{Q} = \boldsymbol{0}.}$$

This is the continuous algebraic Riccati equation.

Finite-Horizon MATLAB Implementation

The following template integrates the Riccati equation backward and the state equation forward.

```
function demo_finite_horizon_lqr

    t0 = 0;
    tf = 10;

    A = [0 1; -2 -0.5];
    B = [0; 1];
    Q = eye(2);
    R = 1;
    Sf = eye(2);

    n = size(A,1);

    % Backward Riccati equation
    riccati = @(t,svec) riccati_rhs( ...
        t, svec, A, B, Q, R, n);

    [tS, Svec] = ode45( ...
        riccati, [tf t0], Sf(:));

    % Reverse for increasing -time interpolation
    tS = flipud(tS);
    Svec = flipud(Svec);

    Sfun = @(t) reshape( ...
        interp1(tS, Svec, t, 'pchip'), n, n);

    x0 = [1; 0];

    % Closed -loop state dynamics
    closed_loop = @(t,x) ...
        (A - B*(R\(B.'*Sfun(t))))*x;

    [tx, x] = ode45(closed_loop, [t0 tf], x0);

    u = zeros(length(tx),1);
    for k = 1:length(tx)
        S = Sfun(tx(k));
        K = R\(B.'*S);
        u(k) = -K*x(k,:).';
    end

    figure;
    plot(tx,x,'LineWidth',1.5);
```

```
    xlabel('Time');
    ylabel('State');
    legend('x_1','x_2');
    grid on;

    figure;
    plot(tx,u,'LineWidth',1.5);
    xlabel('Time');
    ylabel('Control');
    grid on;
end

function ds = riccati_rhs(~,svec,A,B,Q,R,n)

    S = reshape(svec,n,n);

    dS = -A.'*S - S*A ...
         + S*B*(R\B.')*S - Q;

    ds = dS(:);
end
```

Numerical Symmetry Preservation

In exact arithmetic, $S(t)$ remains symmetric if S_f is symmetric.
Numerical integration may introduce small asymmetry. A common correction is

$$S \leftarrow \frac{1}{2}(S + S^{\mathsf{T}}).$$

This should be used only to remove numerical roundoff, not to hide a significant implementation error.

Verification of the Riccati Solution

A computed solution should be checked using the residual

$$R_{\mathrm{DRE}}(t) = \dot{S} + A^{\mathsf{T}}S + SA - SBR^{-1}B^{\mathsf{T}}S + Q.$$

A valid numerical solution should satisfy

$$\|R_{\mathrm{DRE}}(t)\| \ll 1.$$

Also verify

$$\|S(t_f) - S_f\| \ll 1.$$

Verification of the Value Function

For the optimal state and control,

$$J^*(\boldsymbol{x}_0, t_0) = \frac{1}{2}\boldsymbol{x}_0^\mathsf{T}\boldsymbol{S}(t_0)\boldsymbol{x}_0.$$

The same quantity should be recovered from numerical integration:

$$J_{\text{sim}} = \frac{1}{2}\boldsymbol{x}^\mathsf{T}(t_f)\boldsymbol{S}_f\boldsymbol{x}(t_f) + \frac{1}{2}\int_{t_0}^{t_f}\left(\boldsymbol{x}^\mathsf{T}\boldsymbol{Q}\boldsymbol{x} + \boldsymbol{u}^\mathsf{T}\boldsymbol{R}\boldsymbol{u}\right)\,\mathrm{d}t.$$

The difference

$$|J_{\text{sim}} - J^*(\boldsymbol{x}_0, t_0)|$$

should be small.

Feedback Interpretation

The HJB derivation generates

$$\boldsymbol{u}^*(t) = -\boldsymbol{K}(t)\boldsymbol{x}(t).$$

Thus, the control depends on the current state, not only on the nominal state trajectory.
If a disturbance changes the state, the same gain matrix maps the new state to a corrective control.

Open-Loop and Closed-Loop Views

The LQR solution can be interpreted in two ways:

1. An open-loop optimal trajectory is obtained by propagating the optimal closed-loop system from a specified initial state.
2. The feedback law itself is valid for every state in the domain.

The HJB solution therefore contains more information than a single open-loop trajectory.

Common Derivation Errors

Dropping the Factor of One-Half

The factors of one-half simplify derivatives. Removing them inconsistently produces factor-of-two errors.

Incorrect Gradient Orientation

Use a consistent column-gradient convention:

$$J_{\boldsymbol{x}} = \nabla_{\boldsymbol{x}} J.$$

Forgetting Matrix Symmetry

The derivative of

$$\frac{1}{2}\boldsymbol{x}^{\top}\boldsymbol{S}\boldsymbol{x}$$

is $\boldsymbol{S}\boldsymbol{x}$ only when $\boldsymbol{S}$ is symmetric.

Incorrect Riccati Sign Convention

The finite-horizon Riccati equation is commonly written either in forward-time derivative form with a terminal condition or in backward-time variable form. Mixing the two conventions causes sign errors.

Integrating from the Wrong Boundary

The Riccati equation must satisfy

$$\boldsymbol{S}(t_f) = \boldsymbol{S}_f.$$

Assuming Numerical Optimality Without Verification

The Riccati residual, terminal condition, closed-loop dynamics, and cost identity must all be checked.

Recommended Computational Workflow

1. Define $\boldsymbol{A}(t)$, $\boldsymbol{B}(t)$, $\boldsymbol{Q}(t)$, $\boldsymbol{R}(t)$, and $\boldsymbol{S}_f$.
2. Confirm symmetry and definiteness assumptions.
3. Integrate the Riccati equation backward.
4. Verify the terminal condition.
5. Verify symmetry of $\boldsymbol{S}(t)$.
6. Compute $\boldsymbol{K}(t)$.
7. Integrate the closed-loop state equation forward.
8. Compute $\boldsymbol{u}^*(t)$.
9. Evaluate the cost numerically.
10. Compare it with the value-function prediction.
11. Check Riccati and state-equation residuals.

Summary

1. The HJB equation reduces to the differential Riccati equation for the LQR problem.
2. The matrix $\boldsymbol{S}(t)$ is integrated backward from $\boldsymbol{S}_f$.
3. The optimal feedback gain is

$$\boldsymbol{K}(t) = \boldsymbol{R}^{-1}\boldsymbol{B}^{\mathsf{T}}\boldsymbol{S}(t).$$

4. The optimal control is

$$\boldsymbol{u}^*(t) = -\boldsymbol{K}(t)\boldsymbol{x}^*(t).$$

5. The costate is

$$\boldsymbol{\lambda}^*(t) = \boldsymbol{S}(t)\boldsymbol{x}^*(t).$$

6. HJB and PMP produce the same Riccati equation.
7. The LQR problem avoids the curse of dimensionality because the value function is represented by a matrix.
8. Numerical solutions require residual, symmetry, boundary-condition, and cost checks.

Connection. A feedback law is one component of a larger system, so the discussion now places optimization alongside guidance, navigation, estimation, and inner-loop control.

Optimal Control within Guidance, Navigation, Estimation, and Control

Optimal control can appear disconnected from classical feedback control. Classical control courses often emphasize transfer functions, compensators, stability margins, and feedback loops. Optimal control, by contrast, introduces calculus of variations, necessary conditions, boundary-value problems, direct transcription, and numerical optimization.

The purpose of this session is to place optimal control within the larger architecture of a complete control system. The central conclusion is:

> **Central Interpretation**
> Optimal control is most naturally interpreted as a method for generating high-quality reference state and control trajectories. In aerospace terminology, this is optimal guidance.

The resulting reference signals are then tracked by an inner-loop feedback controller using estimated states obtained from navigation and estimation.

Major Components of a Complete Control System

A practical closed-loop system commonly contains five major components:

1. **Plant or dynamics**: the physical system being controlled.
2. **Measurement system**: sensors that produce measurable outputs.
3. **State estimation**: an algorithm that reconstructs the system state.
4. **Control**: a feedback law that computes corrective control action.
5. **Reference generation or guidance**: a method for generating desired state and control trajectories.

These components are related but not interchangeable.

Plant Dynamics

The plant is described by a state-space model

$$\boxed{\dot{\boldsymbol{x}}(t) = \boldsymbol{f}\big(\boldsymbol{x}(t), \boldsymbol{u}(t), t\big).}$$

The control input is $\boldsymbol{u}(t)$. The derivative $\dot{\boldsymbol{x}}(t)$ is integrated to obtain the state:

$$\boldsymbol{x}(t) = \boldsymbol{x}(t_0) + \int_{t_0}^{t} \dot{\boldsymbol{x}}(\tau)\,\mathrm{d}\tau.$$

Thus, the plant block maps the state and control to the state derivative.

Measurement and Navigation

In general, the full state cannot be measured directly. Instead, sensors produce

$$\boxed{\boldsymbol{y}(t) = \boldsymbol{h}\big(\boldsymbol{x}(t), t\big) + \boldsymbol{v}(t),}$$

where $v(t)$ represents measurement noise.

In aerospace applications, the process of determining position, velocity, attitude, and related states is called navigation.

Navigation
Navigation answers the question:

Where am I, and how am I moving?

Navigation is fundamentally a measurement and state-determination problem.

Guidance

Guidance determines the desired path or trajectory.

Guidance
Guidance answers the question:

Where should I go, and what path should I follow?

A navigation system may determine that a vehicle is at a given latitude and longitude. A guidance system determines the desired route from that location to the destination.

This distinction is often blurred in everyday language. Turn-by-turn route instructions are guidance, even if they are marketed as navigation.

Control

Control generates the actuator commands required to track the reference trajectory.

A feedback controller typically maps tracking error into a corrective control:

$$\boxed{\Delta \boldsymbol{u}(t) = \boldsymbol{\kappa}\big(\Delta \boldsymbol{x}(t), t\big).}$$

The tracking error may be defined as

$$\boxed{\Delta \boldsymbol{x}(t) = \boldsymbol{x}_{\mathrm{ref}}(t) - \hat{\boldsymbol{x}}(t).}$$

The full control is

$$\boxed{\boldsymbol{u}(t) = \boldsymbol{u}_{\mathrm{ref}}(t) + \Delta \boldsymbol{u}(t).}$$

Thus, guidance supplies the nominal control and state, while feedback control supplies corrections.

State Estimation

Because the state is not generally measured directly, an estimator combines the measured output and known control:

$$\boxed{\hat{\boldsymbol{x}}(t) = \mathcal{E}\big(\boldsymbol{y}(\cdot), \boldsymbol{u}(\cdot)\big).}$$

Examples include:

- Luenberger observers,
- Kalman filters,
- extended Kalman filters,
- unscented Kalman filters,
- particle filters.

The estimator output $\hat{\boldsymbol{x}}$ is an estimate, not the exact state.

Complete Guidance–Navigation–Estimation–Control Architecture

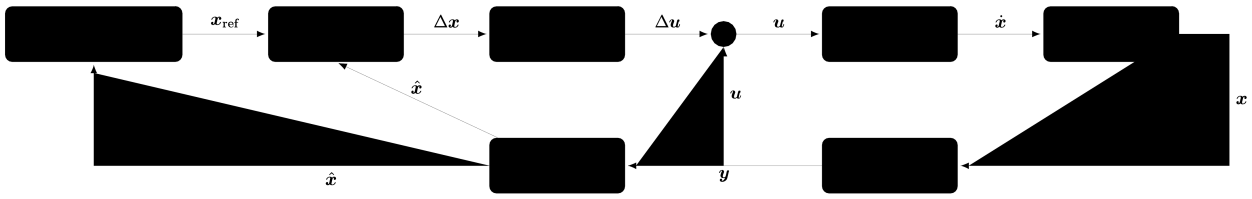

Figure 26: A complete guidance, navigation, estimation, and control architecture.

Reference-State and Reference-Control Generation

Guidance typically produces two signals:

$$\boldsymbol{x}_{\text{ref}}(t), \qquad \boldsymbol{u}_{\text{ref}}(t).$$

The reference state describes the desired trajectory. The reference control is the nominal input required to follow it.
If the system is exactly on the reference trajectory,

$$\hat{\boldsymbol{x}}(t) = \boldsymbol{x}_{\text{ref}}(t),$$

then

$$\Delta \boldsymbol{x}(t) = \boldsymbol{0}.$$

Ideally,

$$\Delta \boldsymbol{u}(t) = \mathbf{0},$$

and the plant is driven only by the nominal control $\boldsymbol{u}_{\text{ref}}$.

Inner Loop and Outer Loop

The architecture naturally separates into two loops.

Inner Loop

The inner loop is the feedback controller:

$$\Delta \boldsymbol{x} \longmapsto \Delta \boldsymbol{u}.$$

It operates at a relatively high update rate and rejects disturbances, compensates for model error, and stabilizes the plant.

Outer Loop

The outer loop is guidance:

$$\hat{\boldsymbol{x}} \longmapsto (\boldsymbol{x}_{\text{ref}}, \boldsymbol{u}_{\text{ref}}).$$

It is responsible for replanning and reference generation. It may operate less frequently because solving a guidance problem is often computationally expensive.

Optimal Control as Optimal Guidance

An optimal control problem generates a state and control trajectory that optimize a performance index:

$$\min_{\boldsymbol{u}(\cdot)} J$$

subject to

$$\begin{aligned} \dot{\boldsymbol{x}} &= \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, t), \\ \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{u}, t) &\leq \mathbf{0}, \\ \boldsymbol{\psi}\big(\boldsymbol{x}(t_0), \boldsymbol{x}(t_f), t_0, t_f\big) &= \mathbf{0}. \end{aligned}$$

The solution is

$$\boxed{(\boldsymbol{x}^*(t), \boldsymbol{u}^*(t))\,.}$$

Within the full control architecture, these are interpreted as

$$\boldsymbol{x}_{\text{ref}}(t) = \boldsymbol{x}^*(t),$$
$$\boldsymbol{u}_{\text{ref}}(t) = \boldsymbol{u}^*(t).$$

Therefore,

Optimal control is optimal reference-trajectory generation.

In aerospace terminology,

Optimal control is optimal guidance.

Open-Loop Guidance

Suppose the optimal control problem is solved once before execution:

$$(\boldsymbol{x}^*(t), \boldsymbol{u}^*(t)), \qquad t \in [t_0, t_f].$$

If this reference is never recomputed, the guidance solution is open loop.
The inner-loop controller may still be closed loop, but the guidance trajectory itself is fixed.
This distinction is important:

- open-loop optimal control does not imply that the entire vehicle is operated without feedback;
- it only means that the reference trajectory is not regenerated using new state information.

Receding-Horizon or Replanning Guidance

Suppose guidance is recomputed every T_g seconds. At update time t_k,

$$t_k = t_0 + kT_g.$$

The current estimated state becomes the new initial condition:

$$\boldsymbol{x}(t_k) = \hat{\boldsymbol{x}}(t_k).$$

A new optimal control problem is solved from t_k to the terminal time.
This produces updated reference signals:

$$\boldsymbol{x}_{\text{ref}}^{(k)}(t), \qquad \boldsymbol{u}_{\text{ref}}^{(k)}(t).$$

This procedure is closely related to:

- model predictive control,
- receding-horizon control,
- trajectory replanning,

- midcourse correction.

Guidance Update Switch

Guidance can be viewed as operating through a switch that closes every T_g seconds.
When the switch is open, the system continues to use the most recently generated reference.
When the switch closes:

1. the latest state estimate is passed to guidance;
2. the reference-generation problem is resolved;
3. new state and control references are issued.

The update period must be compatible with the computational time required to solve the guidance problem.

Computational Feasibility

A reference-generation method is useful only if it can provide a solution before the next guidance update.
If the guidance update period is T_g and the solution time is T_s, practical implementation requires

$$\boxed{T_s < T_g.}$$

For fast systems, T_g may be very small. A highly accurate but slow optimization method may therefore be unusable in real time.
This motivates:

- simplified dynamics,
- reduced-order models,
- precomputed policies,
- lookup tables,
- explicit guidance laws,
- warm-started optimization,
- surrogate models,
- approximate optimal control.

Efficient Guidance versus Exact Optimal Guidance

A practical guidance law need not be exactly optimal.
A useful design may satisfy

$$J_{\text{guidance}} \approx J^*,$$

while requiring substantially less computational effort.
Thus, practical engineering often seeks efficient guidance rather than mathematically exact optimal guidance.

Engineering Tradeoff
A slightly suboptimal guidance law that can be evaluated reliably in real time may be more valuable than an exact optimal solution that arrives too late.

Launch-Vehicle Ascent

Launch-vehicle ascent is a classic example of reference generation.

A typical ascent contains several phases:

1. vertical rise,
2. pitch-over maneuver,
3. gravity turn,
4. stage separation,
5. upper-stage burns,
6. orbital insertion.

Within each phase, the steering profile may be approximated using simple parameterizations, such as:

- constant pitch rate,
- piecewise-linear pitch history,
- low-order polynomial steering,
- linear-tangent steering.

These simple guidance laws can be extremely close to optimal because the physical structure of the problem is well understood.

Direct Shooting in Ascent Guidance

Suppose the control is represented by N parameters:

$$\boldsymbol{u}(t) = \boldsymbol{u}(t; \boldsymbol{p}), \qquad \boldsymbol{p} \in \mathbb{R}^N.$$

The ascent optimization becomes

$$\min_{\boldsymbol{p}} J(\boldsymbol{p})$$

subject to the integrated vehicle dynamics and terminal orbital constraints.

This is direct shooting.

Direct shooting is especially attractive when the steering law can be accurately represented with only a few parameters.

Powered Explicit Guidance

Powered explicit guidance uses the current state and desired terminal state to construct a required velocity change.

Define

$$\Delta \boldsymbol{v} = \boldsymbol{v}_{\text{target}} - \boldsymbol{v}_{\text{current}}.$$

A simple guidance principle is to orient thrust approximately in the direction of $\Delta \boldsymbol{v}$:

$$\hat{\boldsymbol{T}} \propto \Delta \boldsymbol{v}.$$

The algorithm is repeatedly updated so that the required velocity error decreases.

Although physically intuitive and computationally inexpensive, such a law may also be interpreted as an approximation to an optimal steering law.

Linear-Tangent Steering

In several simplified ascent and intercept problems, optimality conditions yield a steering law for which the tangent of the steering angle is linear in time:

$$\boxed{\tan \theta(t) = a + bt.}$$

This is linear-tangent steering.

The significance is that a simple explicit law can emerge from optimal control theory and can then be implemented in real time without solving the full optimal control problem online.

Spacecraft Reorientation

Spacecraft attitude reorientation may require much more precise guidance than launch ascent.

Typical objectives include:

- minimum maneuver time,
- minimum fuel,
- minimum reaction-wheel momentum buildup,
- minimum structural excitation,
- terminal pointing accuracy.

The optimal control structure may involve:

- bang-bang control,
- singular arcs,
- actuator saturation,
- nonlinear rotational dynamics,
- state constraints.

Because pointing accuracy is critical, approximations must be designed and verified carefully.

Minimum-Time and Minimum-Fuel Guidance

For minimum time,

$$J = t_f.$$

For minimum fuel,

$$J = \int_{t_0}^{t_f} \|\boldsymbol{u}(t)\| \, \mathrm{d}t$$

or a related propellant-consumption model.
These objectives lead to different reference controls.
Thus, "best" has no meaning unless the performance criterion is specified.

What Does "Better" Mean?

Whenever one trajectory is judged better than another, an implicit performance measure is being used.
A formal optimization problem makes that measure explicit:

$$J = J\left[\boldsymbol{x}(\cdot), \boldsymbol{u}(\cdot)\right].$$

Even when engineers judge solutions qualitatively, they are implicitly considering criteria such as:

- time,
- energy,
- fuel,
- safety,
- smoothness,
- comfort,
- robustness,
- tracking accuracy,
- computational effort.

Role of Necessary Conditions

The calculus of variations and Pontryagin's minimum principle produce conditions that an optimal solution must satisfy.
These conditions do not generally provide a simple algorithm for obtaining the solution.
They are:

- necessary rather than automatically sufficient;
- often analytically solvable only for special problems;
- frequently converted into difficult two-point boundary-value problems.

Their value is theoretical insight, structural characterization, and verification.

Role of Numerical Methods

Direct and indirect numerical methods generate approximations to the optimal reference trajectory.
Examples include:

- single shooting,
- multiple shooting,
- direct transcription,
- collocation,
- pseudospectral methods,
- sequential quadratic programming,
- interior-point methods.

The objective is ultimately to produce

$$\boldsymbol{x}_{\mathrm{ref}}(t), \qquad \boldsymbol{u}_{\mathrm{ref}}(t),$$

that can be passed to the feedback controller.

Why Numerical Optimal Control May Be Too Slow

A high-fidelity optimal control problem may contain:

- nonlinear dynamics,
- hundreds or thousands of decision variables,
- path constraints,
- phase changes,
- discrete events,
- uncertainty,
- expensive simulation models.

A general nonlinear programming solver may require seconds, minutes, hours, or longer.
If the physical system evolves faster than the optimizer, the method cannot be used directly for online guidance.

Strategies for Real-Time Implementation

Several approaches can make optimal guidance practical.

Offline Optimization

Solve the problem before operation and store the resulting trajectory.

Trajectory Libraries

Precompute many trajectories and select or interpolate among them online.

Reduced-Order Models

Use simplified dynamics during online optimization.

Explicit Guidance Laws

Fit a low-order analytical law to high-fidelity optimal solutions.

Warm Starting

Initialize the current optimization using the previous solution.

Model Predictive Control

Solve a shorter-horizon problem repeatedly.

Surrogate Modeling

Replace expensive simulation models with fast approximations.

Policy Approximation

Approximate the optimal feedback law using regression, neural networks, or basis functions.

Navigation Errors and Estimation Errors

Measurements are always imperfect. Common error sources include:

- sensor noise,
- bias,
- scale-factor errors,
- misalignment,
- drift,
- delays,
- environmental disturbances,
- model mismatch.

State estimation filters these measurements and combines them with the model and control history.

If navigation errors become large, guidance may generate the wrong reference because it is solving the problem from an incorrect initial state estimate.

Interaction among Guidance, Estimation, and Control

The three subsystems are coupled:

- guidance depends on the estimated state;
- control depends on the reference and estimated state;
- estimation depends on the measured output and applied control;
- the plant responds to the total control;
- new measurements close the loop.

Poor performance in one subsystem can degrade the entire architecture.

Hierarchical View of the System

The system may be viewed hierarchically:

1. **Mission level**: select objectives and terminal goals.
2. **Guidance level**: generate feasible reference trajectories.
3. **Control level**: track the references.
4. **Actuation level**: produce physical forces and moments.
5. **Navigation and estimation level**: determine the current state.

Optimal control primarily operates at the guidance level, although optimal design methods can also be used at the control and estimation levels.

Open-Loop and Closed-Loop Terminology

Care is required when using the term open loop.

An optimal trajectory may be open loop at the guidance level while the vehicle remains closed loop at the control level.

For example:

- the reference trajectory is computed once;
- the feedback controller tracks it continuously;
- the guidance problem is never resolved.

The overall system contains feedback, but the optimal reference generation remains open loop.

Midcourse Correction

A midcourse correction occurs when the reference is recomputed after a deviation.

At correction time t_c:

1. navigation determines the current estimated state $\hat{\boldsymbol{x}}(t_c)$;
2. guidance solves a new problem from $\hat{\boldsymbol{x}}(t_c)$ to the target;
3. a new reference trajectory is generated;

4. the controller tracks the updated reference.

This is a practical implementation of Bellman's principle of optimality.

Design Questions

When embedding optimal control into a complete system, ask:

1. What variables can be measured?
2. What states must be estimated?
3. How often is guidance updated?
4. How fast can the optimization be solved?
5. What model fidelity is required?
6. How large are the expected disturbances?
7. What constraints must always be respected?
8. Is exact optimality necessary?
9. What feedback controller will track the reference?
10. How will the full architecture be verified?

Verification of the Complete Architecture

Verification should include:

- reference feasibility,
- tracking error,
- state-estimation error,
- actuator saturation,
- robustness to disturbances,
- sensitivity to model mismatch,
- guidance update timing,
- computational latency,
- terminal-condition accuracy,
- safety-constraint satisfaction.

Summary

1. A complete control system includes plant dynamics, measurement, navigation, estimation, control, and guidance.
2. Navigation determines where the system is.
3. Guidance determines where the system should go.
4. Control generates corrective actuator commands.

5. Estimation reconstructs the state from measurements and control history.
6. Optimal control naturally generates reference state and control trajectories.
7. Therefore, optimal control is best interpreted as optimal guidance.
8. Guidance is an outer-loop function; tracking control is an inner-loop function.
9. An optimal reference may be generated once or repeatedly.
10. Repeated replanning produces receding-horizon or midcourse-correction behavior.
11. Exact optimality may be sacrificed for speed and implementability.
12. Practical guidance often uses simplified laws derived from or validated against optimal control theory.
13. Launch-vehicle ascent and spacecraft reorientation illustrate different levels of guidance complexity.
14. Numerical optimal control is valuable only when it can deliver a reference in time for the physical system.

Connection. Before invoking a large numerical transcription, analytical reduction can expose a low-dimensional shooting problem and improve both insight and robustness.

Analytical Direct Shooting and Problem Simplification

This section revisits a minimum-time optimal control problem to emphasize an important modeling principle: not every problem requires heavy numerical machinery. Before writing code, the governing equations and optimality conditions should be simplified as far as possible analytically.

The example considered here is

$$\min_{u(\cdot),\, t_f} J = t_f$$

subject to

$$\begin{aligned} \dot{x} &= v \sin u, \\ \dot{y} &= -v \cos u, \\ \dot{v} &= g \cos u, \end{aligned}$$

with boundary conditions

$$\begin{aligned} x(0) &= 0, \quad y(0) = 0, \quad v(0) = 0, \\ x(t_f) &= 2, \quad y(t_f) = -2, \end{aligned}$$

and free terminal velocity.

Hamiltonian and Costate Equations

The Hamiltonian is

$$\mathcal{H} = 1 + \lambda_x v \sin u - \lambda_y v \cos u + \lambda_v g \cos u.$$

The costate equations are

$$\begin{aligned} \dot{\lambda}_x &= -\frac{\partial \mathcal{H}}{\partial x} = 0, \\ \dot{\lambda}_y &= -\frac{\partial \mathcal{H}}{\partial y} = 0, \\ \dot{\lambda}_v &= -\frac{\partial \mathcal{H}}{\partial v} = -\lambda_x \sin u + \lambda_y \cos u. \end{aligned}$$

Thus,

$$\lambda_x = \text{constant}, \qquad \lambda_y = \text{constant}.$$

Stationarity Condition

For an interior optimal control,

$$\frac{\partial \mathcal{H}}{\partial u} = 0.$$

Hence,

$$\lambda_x v \cos u + \lambda_y v \sin u - \lambda_v g \sin u = 0.$$

Rearranging,

$$\lambda_x v \cos u = \left(\lambda_v g - \lambda_y v\right) \sin u.$$

Therefore,

$$\boxed{\tan u = \frac{\lambda_x v}{\lambda_v g - \lambda_y v}.}$$

By differentiating this relation and using the state and costate equations, one obtains

$$\boxed{\ddot{u} = 0.}$$

Therefore, the optimal control is affine in time:

$$\boxed{u(t) = c_0 + c_1 t.}$$

This structural result is the key simplification.

Direct-Shooting Parameterization

A conventional direct-shooting implementation would parameterize $u(t)$, integrate the dynamics numerically, and optimize over the control parameters and final time.

Here, however, the affine control allows the dynamics to be integrated analytically.

The unknown parameter vector is

$$\boxed{\boldsymbol{p} = \begin{bmatrix} c_0 & c_1 & t_f \end{bmatrix}^{\mathsf{T}}.}$$

Analytical Integration of the Velocity

Using

$$u(t) = c_0 + c_1 t,$$

the velocity equation becomes

$$\dot{v} = g \cos(c_0 + c_1 t).$$

Integrating from 0 to t,

$$v(t) = v(0) + \int_0^t g\cos(c_0 + c_1\tau)\,\mathrm{d}\tau$$
$$= \frac{g}{c_1}\left[\sin(c_0 + c_1 t) - \sin c_0\right].$$

Since $v(0) = 0$,

$$\boxed{v(t) = \frac{g}{c_1}\left[\sin(c_0 + c_1 t) - \sin c_0\right].}$$

The case $c_1 = 0$ must be treated separately because the above form contains division by c_1.

Analytical Integration of the Horizontal Position

The horizontal dynamics are

$$\dot{x} = v\sin u.$$

Substituting $v(t)$ and $u(t)$,

$$\dot{x} = \frac{g}{c_1}\left[\sin(c_0 + c_1 t) - \sin c_0\right]\sin(c_0 + c_1 t).$$

Expanding,

$$\dot{x} = \frac{g}{c_1}\sin^2(c_0 + c_1 t) - \frac{g\sin c_0}{c_1}\sin(c_0 + c_1 t).$$

Using

$$\sin^2\alpha = \frac{1 - \cos 2\alpha}{2},$$

we obtain

$$\dot{x} = \frac{g}{2c_1} - \frac{g}{2c_1}\cos(2c_0 + 2c_1 t) - \frac{g\sin c_0}{c_1}\sin(c_0 + c_1 t).$$

Integrating and enforcing $x(0) = 0$ gives

$$x(t) = \frac{gt}{2c_1} - \frac{g}{4c_1^2}\left[\sin(2c_0 + 2c_1 t) - \sin(2c_0)\right]$$
$$+ \frac{g\sin c_0}{c_1^2}\left[\cos(c_0 + c_1 t) - \cos c_0\right].$$

Therefore,

$$\boxed{x(t) = \frac{gt}{2c_1} - \frac{g}{4c_1^2}\left[\sin(2c_0 + 2c_1 t) - \sin 2c_0\right] + \frac{g\sin c_0}{c_1^2}\left[\cos(c_0 + c_1 t) - \cos c_0\right].}$$

Analytical Integration of the Vertical Position

The vertical dynamics are

$$\dot{y} = -v\cos u.$$

Substituting,

$$\dot{y} = -\frac{g}{c_1}\left[\sin(c_0 + c_1 t) - \sin c_0\right]\cos(c_0 + c_1 t).$$

Thus,

$$\begin{aligned}\dot{y} = & -\frac{g}{c_1}\sin(c_0 + c_1 t)\cos(c_0 + c_1 t) \\ & + \frac{g\sin c_0}{c_1}\cos(c_0 + c_1 t).\end{aligned}$$

Integrating and enforcing $y(0) = 0$,

$$\begin{aligned}y(t) = & -\frac{g}{2c_1^2}\left[\sin^2(c_0 + c_1 t) - \sin^2 c_0\right] \\ & + \frac{g\sin c_0}{c_1^2}\left[\sin(c_0 + c_1 t) - \sin c_0\right].\end{aligned}$$

Therefore,

$$\boxed{y(t) = -\frac{g}{2c_1^2}\left[\sin^2(c_0 + c_1 t) - \sin^2 c_0\right] + \frac{g\sin c_0}{c_1^2}\left[\sin(c_0 + c_1 t) - \sin c_0\right].}$$

Terminal Constraints

The terminal conditions require

$$\begin{aligned}x(t_f; c_0, c_1) &= 2, \\ y(t_f; c_0, c_1) &= -2.\end{aligned}$$

Define

$$\begin{aligned}F_1(c_0, c_1, t_f) &= x(t_f; c_0, c_1) - 2, \\ F_2(c_0, c_1, t_f) &= y(t_f; c_0, c_1) + 2.\end{aligned}$$

The optimization problem becomes

$$\boxed{\min_{c_0, c_1, t_f} t_f}$$

subject to

$$\boxed{F_1(c_0, c_1, t_f) = 0, \qquad F_2(c_0, c_1, t_f) = 0.}$$

No numerical integration of the state equations is required.

Degrees-of-Freedom Analysis

There are three unknowns:

$$c_0, \quad c_1, \quad t_f.$$

There are two terminal equality constraints:

$$F_1 = 0, \qquad F_2 = 0.$$

Hence, the number of free degrees of freedom is

$$\boxed{3 - 2 = 1.}$$

That remaining degree of freedom allows minimization of t_f.
If t_f were fixed, there would be two unknowns and two equations, leaving no optimization degree of freedom.

Why This Formulation Is Superior

The analytical reduction provides several advantages:

- no numerical ODE integration;
- no integration tolerances;
- no accumulated state-integration error;
- fewer sources of implementation error;
- faster optimization;
- clearer interpretation of the unknowns;
- easier verification.

The main lesson is not that numerical integration is wrong. The lesson is that it should not be used when the problem can be simplified exactly.

Treatment of the Case $c_1 = 0$

If $c_1 = 0$, then

$$u(t) = c_0.$$

The velocity becomes

$$v(t) = gt\cos c_0.$$

Then

$$x(t) = \frac{1}{2}gt^2 \sin c_0 \cos c_0,$$
$$y(t) = -\frac{1}{2}gt^2 \cos^2 c_0.$$

This special case can be checked separately against the terminal constraints.

MATLAB Formulation

```
function solve_session31_part1

    g = 9.81;

    z0 = [0.5; 0.5; 1.0]; % [c0, c1, tf]

    obj = @(z) z(3);

    nonlcon = @(z) terminal_constraints(z,g);

    options = optimoptions('fmincon', ...
        'Display','iter', ...
        'Algorithm','sqp', ...
        'ConstraintTolerance',1e -10, ...
        'OptimalityTolerance',1e -10);

    lb = [ -pi; -10; 1e -3];
    ub = [ pi;  10; 20];

    z = fmincon(obj,z0,[],[],[],[],lb,ub,nonlcon,options);

    c0 = z(1);
    c1 = z(2);
    tf = z(3);

    fprintf('c0 = %.12f\n',c0);
    fprintf('c1 = %.12f\n',c1);
    fprintf('tf = %.12f\n',tf);
end

function [c,ceq] = terminal_constraints(z,g)

    c0 = z(1);
    c1 = z(2);
    tf = z(3);

    if abs(c1) < 1e -8
```

```
        x = 0.5*g*tf^2*sin(c0)*cos(c0);
        y = -0.5*g*tf^2*cos(c0)^2;
    else
        x = g*tf/(2*c1) ...
          - g/(4*c1^2)*(sin(2*c0+2*c1*tf) -sin(2*c0)) ...
          + g*sin(c0)/c1^2*(cos(c0+c1*tf) -cos(c0));

        y = -g/(2*c1^2)*(sin(c0+c1*tf)^2 -sin(c0)^2) ...
          + g*sin(c0)/c1^2*(sin(c0+c1*tf) -sin(c0));
    end

    c = [];
    ceq = [x -2; y+2];
end
```

Verification

After obtaining (c_0, c_1, t_f):

1. verify the terminal constraints;
2. reconstruct $u(t)$ analytically;
3. reconstruct $v(t)$, $x(t)$, and $y(t)$;
4. optionally integrate the original dynamics numerically;
5. compare the numerical and analytical trajectories.

General Lessons for Direct Shooting

1. Derive the structure of the optimal control first.
2. Use analytical integration whenever possible.
3. Count unknowns, equations, and degrees of freedom.
4. Do not use a numerical solver as a substitute for analysis.
5. Formulate the smallest nonlinear program that represents the problem.

Connection. When the reduced Hamiltonian is linear in a bounded control, switching functions distinguish bang arcs from possible singular intervals.

Bang–Bang Control, Switching Functions, and Singular Arcs

This section revisits a bounded-control optimal control problem whose Hamiltonian is linear in the control. Such problems commonly produce bang-bang control, singular control, or combinations of both.

Consider

$$\min_{u(\cdot)} J = \int_0^1 x_2(t)u(t)\,\mathrm{d}t$$

subject to

$$\begin{aligned}\dot{x}_1 &= x_2,\\ \dot{x}_2 &= -x_2 + u,\end{aligned}$$

with

$$0 \le u(t) \le 2,$$

and boundary conditions

$$\begin{aligned}x_1(0) &= 0, \quad x_2(0) = 1,\\ x_1(1) &= 1, \quad x_2(1) = 1.\end{aligned}$$

Hamiltonian

The Hamiltonian is

$$\begin{aligned}\mathcal{H} &= x_2 u + \lambda_1 x_2 + \lambda_2(-x_2 + u)\\ &= (x_2 + \lambda_2)u + (\lambda_1 - \lambda_2)x_2.\end{aligned}$$

Thus,

$$\boxed{\mathcal{H} = \phi(t)u + (\lambda_1 - \lambda_2)x_2,}$$

where

$$\boxed{\phi(t) = x_2(t) + \lambda_2(t)}$$

is the switching function.

Bang-Bang Minimization Rule

Because the Hamiltonian is affine in u,

$$u^* = \arg\min_{0 \le u \le 2} \mathcal{H}.$$

Therefore,

$$\boxed{u^*(t) = \begin{cases} 0, & \phi(t) > 0, \\ 2, & \phi(t) < 0, \\ \text{undetermined by first-order minimization}, & \phi(t) = 0. \end{cases}}$$

The case $\phi = 0$ is the candidate singular case.

Costate Equations

The costate equations are

$$\dot{\lambda}_1 = -\frac{\partial \mathcal{H}}{\partial x_1} = 0,$$
$$\dot{\lambda}_2 = -\frac{\partial \mathcal{H}}{\partial x_2}.$$

Since

$$\frac{\partial \mathcal{H}}{\partial x_2} = u + \lambda_1 - \lambda_2,$$

we obtain

$$\boxed{\dot{\lambda}_2 = -u - \lambda_1 + \lambda_2.}$$

Also,

$$\boxed{\lambda_1 = \text{constant}.}$$

Testing Constant Bang Controls

Before pursuing a singular arc, one should test whether $u = 0$ or $u = 2$ over the entire interval can satisfy the boundary conditions.

Case $u = 0$

Then

$$\dot{x}_2 = -x_2,$$

so

$$x_2(t) = e^{-t}.$$

Thus,

$$x_2(1) = e^{-1} \neq 1.$$

Therefore, $u = 0$ over the full interval is infeasible.

Case $u = 2$

Then

$$\dot{x}_2 = -x_2 + 2.$$

With $x_2(0) = 1$,

$$x_2(t) = 2 - e^{-t}.$$

Hence,

$$x_2(1) = 2 - e^{-1} \neq 1.$$

Therefore, $u = 2$ over the full interval is also infeasible.

Singular-Arc Condition

On a singular arc,

$$\boxed{\phi(t) = x_2(t) + \lambda_2(t) = 0.}$$

Therefore,

$$\lambda_2 = -x_2.$$

Differentiating,

$$\dot{\phi} = \dot{x}_2 + \dot{\lambda}_2.$$

Using

$$\begin{aligned} \dot{x}_2 &= -x_2 + u, \\ \dot{\lambda}_2 &= -u - \lambda_1 + \lambda_2, \end{aligned}$$

we obtain

$$\begin{aligned} \dot{\phi} &= (-x_2 + u) + (-u - \lambda_1 + \lambda_2) \\ &= -x_2 - \lambda_1 + \lambda_2. \end{aligned}$$

On the singular arc, $\lambda_2 = -x_2$, so

$$\dot{\phi} = -2x_2 - \lambda_1.$$

Hence,

$$\boxed{2x_2 + \lambda_1 = 0.}$$

Therefore,

$$x_2 = -\frac{\lambda_1}{2}.$$

Because λ_1 is constant, x_2 is constant on the singular arc.

Determination of the Singular Control

Since x_2 is constant,

$$\dot{x}_2 = 0.$$

Using the state equation,

$$0 = -x_2 + u.$$

Thus,

$$\boxed{u_{\text{sing}} = x_2.}$$

The boundary conditions require

$$x_2(0) = 1, \qquad x_2(1) = 1.$$

Therefore, the constant singular state is

$$x_2(t) = 1.$$

Hence,

$$\boxed{u^*(t) = 1 \qquad \text{for } 0 \le t \le 1.}$$

Verification of the State Trajectory

With $u = 1$,

$$\dot{x}_2 = -x_2 + 1.$$

Since $x_2(0) = 1$,

$$x_2(t) = 1.$$

Then

$$\dot{x}_1 = x_2 = 1.$$

With $x_1(0) = 0$,

$$x_1(t) = t.$$

Therefore,

$$x_1(1) = 1,$$
$$x_2(1) = 1.$$

All boundary conditions are satisfied.

Costate Values

From

$$2x_2 + \lambda_1 = 0$$

and $x_2 = 1$,

$$\boxed{\lambda_1 = -2.}$$

Since

$$\lambda_2 = -x_2,$$

we obtain

$$\boxed{\lambda_2 = -1.}$$

These values satisfy

$$\dot{\lambda}_2 = -u - \lambda_1 + \lambda_2 = -1 - (-2) - 1 = 0.$$

Hamiltonian along the Singular Arc

The Hamiltonian is

$$\mathcal{H} = x_2 u + \lambda_1 x_2 + \lambda_2(-x_2 + u).$$

Using

$$x_2 = 1, \quad u = 1, \quad \lambda_1 = -2, \quad \lambda_2 = -1,$$

we obtain

$$\mathcal{H} = 1 - 2 + 0 = -1.$$

Thus, the Hamiltonian is constant, as expected for an autonomous fixed-final-time problem.

Order of the Singular Arc

The control does not appear explicitly in

$$\phi = x_2 + \lambda_2.$$

It also cancels from

$$\dot{\phi}.$$

The control appears when enforcing constancy of x_2, or equivalently in the next derivative of the switching function.
Differentiate

$$\dot{\phi} = -2x_2 - \lambda_1.$$

Since λ_1 is constant,

$$\ddot{\phi} = -2\dot{x}_2 = -2(-x_2 + u).$$

On the singular arc,

$$\ddot{\phi} = 0,$$

so

$$u = x_2.$$

The control appears in the second derivative of the switching function.

Generalized Legendre–Clebsch Check

For a minimization problem, a singular arc must satisfy an appropriate generalized Legendre–Clebsch condition.

Here,

$$\ddot{\phi} = 2x_2 - 2u.$$

Therefore,

$$\frac{\partial \ddot{\phi}}{\partial u} = -2.$$

Using the common sign convention for a first-order singular arc,

$$(-1)^1 \frac{\partial \ddot{\phi}}{\partial u} = 2 > 0.$$

Thus, the singular control is consistent with local minimization.

Why Pure Bang-Bang Reasoning Is Insufficient

A common error is to conclude that a Hamiltonian linear in the control always implies bang-bang control.

The correct statement is:

> **Key Principle**
> An affine Hamiltonian produces bang-bang control where the switching function is nonzero. Where the switching function and enough of its derivatives vanish, a singular arc may occur.

In this problem, the entire feasible optimal solution lies on the singular arc.

Alternative Direct Verification

Because the dynamics imply

$$u = \dot{x}_2 + x_2,$$

the cost becomes

$$\begin{aligned} J &= \int_0^1 x_2(\dot{x}_2 + x_2)\,\mathrm{d}t \\ &= \int_0^1 x_2\dot{x}_2\,\mathrm{d}t + \int_0^1 x_2^2\,\mathrm{d}t. \end{aligned}$$

The first term is

$$\int_0^1 x_2 \dot{x}_2 \, \mathrm{d}t = \frac{1}{2} \left[x_2^2(1) - x_2^2(0) \right] = 0.$$

Hence,

$$J = \int_0^1 x_2^2(t) \, \mathrm{d}t.$$

Also,

$$x_1(1) - x_1(0) = \int_0^1 x_2(t) \, \mathrm{d}t = 1.$$

By Cauchy–Schwarz,

$$\left(\int_0^1 x_2(t) \, \mathrm{d}t \right)^2 \leq \left(\int_0^1 1^2 \, \mathrm{d}t \right) \left(\int_0^1 x_2^2(t) \, \mathrm{d}t \right).$$

Therefore,

$$1 \leq J.$$

Equality occurs only when $x_2(t)$ is constant. Since its average is one,

$$x_2(t) = 1.$$

Therefore,

$$u(t) = 1.$$

This provides an independent global-optimality argument.

Optimal Cost

Using

$$x_2 = 1, \qquad u = 1,$$

the optimal cost is

$$\boxed{J^* = \int_0^1 1 \, \mathrm{d}t = 1.}$$

Numerical Verification in MATLAB

```
function verify_session31_part2
```

```
    tspan = [0 1];
    x0 = [0;1];

    u = @(t) 1;

    dyn = @(t,x) [x(2); -x(2)+u(t)];

    [t,x] = ode45(dyn,tspan,x0);

    J = trapz(t,x(:,2).*arrayfun(u,t));

    fprintf('x1(tf) = %.12f\n',x(end,1));
    fprintf('x2(tf) = %.12f\n',x(end,2));
    fprintf('J       = %.12f\n',J);

    figure;
    plot(t,x,'LineWidth',1.5);
    xlabel('Time');
    ylabel('State');
    legend('x_1','x_2');
    grid on;
end
```

Recommended Singular-Control Workflow

1. Form the Hamiltonian.
2. Identify the switching function.
3. Determine the bang controls from its sign.
4. Test whether constant bang arcs satisfy the boundary conditions.
5. Set the switching function equal to zero for a singular arc.
6. Differentiate until the control appears explicitly.
7. Solve for the singular control.
8. Verify the control bounds.
9. Verify state and costate equations.
10. Check Hamiltonian constancy.
11. Verify second-order singular optimality conditions.

Common Errors

- assuming every affine-Hamiltonian problem is purely bang-bang;
- forgetting to define the switching function;
- stopping after $\phi = 0$ without differentiating;
- failing to test feasibility of constant bang arcs;

- using the wrong sign in the costate equations;
- failing to verify the control bounds;
- failing to check terminal boundary conditions.

Summary

1. The Hamiltonian is affine in the control.
2. The switching function is $\phi = x_2 + \lambda_2$.
3. The bang controls are $u = 0$ and $u = 2$.
4. Neither constant bang control satisfies the terminal conditions.
5. A singular arc satisfies $\phi = 0$.
6. Differentiation yields $x_2 = -\lambda_1/2$.
7. Since λ_1 is constant, x_2 is constant.
8. The state equation gives $u = x_2$.
9. Boundary conditions imply $x_2 = 1$.
10. Therefore, $u^* = 1$ over the entire interval.
11. The optimal state is $x_1 = t$, $x_2 = 1$.
12. The optimal cost is $J^* = 1$.

Connection. Finally, neighboring optimal control uses local sensitivity information to correct a nominal solution without resolving the full nonlinear problem after every disturbance.

Neighboring Optimal Control

Suppose an optimal state, control, and costate trajectory have already been computed:

$$\boldsymbol{x}^*(t), \qquad \boldsymbol{u}^*(t), \qquad \boldsymbol{\lambda}^*(t).$$

During implementation, disturbances, model mismatch, sensor error, and imperfect actuation may move the system away from this nominal optimal trajectory. One response is to solve the original nonlinear optimal control problem again. However, repeated nonlinear optimization may be too computationally expensive for real-time implementation.

Neighboring optimal control provides a local alternative. It constructs an approximately optimal correction near the nominal solution by:

1. linearizing the dynamics about the nominal optimal trajectory;
2. approximating the cost to second order;
3. solving the resulting time-varying linear-quadratic problem;
4. applying a feedback correction to the nominal control.

Central Idea
Neighboring optimal control generates a locally optimal feedback correction around a previously computed nonlinear optimal trajectory.

Nominal Optimal Control Problem

Consider the fixed-time problem

$$\min_{\boldsymbol{u}(\cdot)} J = \Phi\big(\boldsymbol{x}(t_f)\big) + \int_{t_0}^{t_f} L\big(\boldsymbol{x}(t), \boldsymbol{u}(t)\big)\, \mathrm{d}t$$

subject to

$$\begin{aligned} \dot{\boldsymbol{x}}(t) &= \boldsymbol{f}\big(\boldsymbol{x}(t), \boldsymbol{u}(t)\big), \\ \boldsymbol{x}(t_0) &= \boldsymbol{x}_0. \end{aligned}$$

For clarity, explicit time dependence, path constraints, and free-final-time terms are omitted. The neighboring-optimal-control idea extends to more general formulations.

Assume

$$(\boldsymbol{x}^*(t), \boldsymbol{u}^*(t), \boldsymbol{\lambda}^*(t))$$

satisfies the first-order necessary conditions.

Neighboring Trajectories

Define perturbed trajectories:

$$\boldsymbol{x}(t) = \boldsymbol{x}^*(t) + \delta\boldsymbol{x}(t),$$
$$\boldsymbol{u}(t) = \boldsymbol{u}^*(t) + \delta\boldsymbol{u}(t),$$
$$\boldsymbol{\lambda}(t) = \boldsymbol{\lambda}^*(t) + \delta\boldsymbol{\lambda}(t).$$

The quantities

$$\delta\boldsymbol{x}(t), \qquad \delta\boldsymbol{u}(t), \qquad \delta\boldsymbol{\lambda}(t)$$

are neighboring variations.
If the actual initial condition differs from the nominal one,

$$\boldsymbol{x}(t_0) = \boldsymbol{x}_0 + \delta\boldsymbol{x}_0,$$

then

$$\boxed{\delta\boldsymbol{x}(t_0) = \delta\boldsymbol{x}_0.}$$

Why a Second-Order Cost Approximation Is Required

At an optimal solution, the first variation of the cost vanishes:

$$\delta J = 0.$$

Therefore, a first-order cost approximation cannot distinguish neighboring directions. The leading nonzero local change in cost is generally the second variation:

$$J \approx J^* + \frac{1}{2}\delta^2 J.$$

This is analogous to a static minimization problem.

Static Second-Order Expansion

For a scalar function $g(\boldsymbol{z})$,

$$g(\boldsymbol{z}^* + \Delta\boldsymbol{z}) \approx g(\boldsymbol{z}^*) + g_{\boldsymbol{z}}(\boldsymbol{z}^*)\Delta\boldsymbol{z} + \frac{1}{2}\Delta\boldsymbol{z}^{\mathsf{T}} g_{\boldsymbol{z}\boldsymbol{z}}(\boldsymbol{z}^*)\Delta\boldsymbol{z}.$$

At a local minimum,

$$g_{\boldsymbol{z}}(\boldsymbol{z}^*) = \boldsymbol{0},$$

so

$$g(z^* + \Delta z) \approx g(z^*) + \frac{1}{2}\Delta z^\top g_{zz}(z^*)\Delta z.$$

The neighboring optimal control derivation follows the same principle for functions of time.

Second Variation of the Cost Functional

The terminal contribution is

$$\frac{1}{2}\delta \boldsymbol{x}^\top(t_f)\Phi^*_{\boldsymbol{xx}}\delta \boldsymbol{x}(t_f),$$

where

$$\Phi^*_{\boldsymbol{xx}} = \left.\frac{\partial^2 \Phi}{\partial \boldsymbol{x}^2}\right|_{\boldsymbol{x}^*(t_f)}.$$

The running-cost contribution is

$$\frac{1}{2}\int_{t_0}^{t_f} \begin{bmatrix}\delta \boldsymbol{x} \\ \delta \boldsymbol{u}\end{bmatrix}^\top \begin{bmatrix}L^*_{\boldsymbol{xx}} & L^*_{\boldsymbol{xu}} \\ L^*_{\boldsymbol{ux}} & L^*_{\boldsymbol{uu}}\end{bmatrix} \begin{bmatrix}\delta \boldsymbol{x} \\ \delta \boldsymbol{u}\end{bmatrix} \mathrm{d}t.$$

Define

$$\begin{aligned}\boldsymbol{P}_f &= \Phi^*_{\boldsymbol{xx}}, \\ \boldsymbol{Q}(t) &= L^*_{\boldsymbol{xx}}, \\ \boldsymbol{M}(t) &= L^*_{\boldsymbol{xu}}, \\ \boldsymbol{R}(t) &= L^*_{\boldsymbol{uu}}.\end{aligned}$$

Then

$$\begin{aligned}\delta^2 J = &\frac{1}{2}\delta \boldsymbol{x}^\top(t_f)\boldsymbol{P}_f\delta \boldsymbol{x}(t_f) \\ &+ \frac{1}{2}\int_{t_0}^{t_f}\left[\delta \boldsymbol{x}^\top \boldsymbol{Q}\delta \boldsymbol{x} + 2\delta \boldsymbol{x}^\top \boldsymbol{M}\delta \boldsymbol{u} + \delta \boldsymbol{u}^\top \boldsymbol{R}\delta \boldsymbol{u}\right]\mathrm{d}t.\end{aligned}$$

Important Difference from the Standard LQR
The neighboring problem generally includes the state–control cross term

$$2\delta \boldsymbol{x}^\top \boldsymbol{M}\delta \boldsymbol{u}.$$

This term arises from mixed second derivatives of the original running cost.

Linearized Neighboring Dynamics

Expand the nonlinear dynamics:

$$\dot{\boldsymbol{x}}^* + \delta\dot{\boldsymbol{x}} \approx \boldsymbol{f}(\boldsymbol{x}^*, \boldsymbol{u}^*) + \boldsymbol{f}_{\boldsymbol{x}}^* \delta\boldsymbol{x} + \boldsymbol{f}_{\boldsymbol{u}}^* \delta\boldsymbol{u}.$$

Because

$$\dot{\boldsymbol{x}}^* = \boldsymbol{f}(\boldsymbol{x}^*, \boldsymbol{u}^*),$$

the nominal terms cancel.
Define

$$\boldsymbol{F}(t) = \left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\right|_{(\boldsymbol{x}^*, \boldsymbol{u}^*)},$$
$$\boldsymbol{G}(t) = \left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}\right|_{(\boldsymbol{x}^*, \boldsymbol{u}^*)}.$$

The neighboring dynamics are

$$\boxed{\delta\dot{\boldsymbol{x}} = \boldsymbol{F}(t)\delta\boldsymbol{x} + \boldsymbol{G}(t)\delta\boldsymbol{u}.}$$

Neighboring Optimal Control Problem

The local correction problem is therefore

$$\min_{\delta\boldsymbol{u}(\cdot)} \delta^2 J$$

subject to

$$\delta\dot{\boldsymbol{x}} = \boldsymbol{F}\delta\boldsymbol{x} + \boldsymbol{G}\delta\boldsymbol{u},$$
$$\delta\boldsymbol{x}(t_0) = \delta\boldsymbol{x}_0.$$

This is a finite-horizon, time-varying linear-quadratic optimal control problem with state–control coupling.

Neighboring Hamiltonian

The Hamiltonian for the neighboring problem is

$$\mathcal{H}_N = \frac{1}{2}\delta\boldsymbol{x}^\top \boldsymbol{Q}\delta\boldsymbol{x} + \delta\boldsymbol{x}^\top \boldsymbol{M}\delta\boldsymbol{u} + \frac{1}{2}\delta\boldsymbol{u}^\top \boldsymbol{R}\delta\boldsymbol{u} + \delta\boldsymbol{\lambda}^\top \left(\boldsymbol{F}\delta\boldsymbol{x} + \boldsymbol{G}\delta\boldsymbol{u}\right).$$

Neighboring Costate Equation

The costate dynamics satisfy

$$\delta\dot{\boldsymbol{\lambda}} = -\frac{\partial \mathcal{H}_N}{\partial \delta\boldsymbol{x}}.$$

Hence,

$$\boxed{\delta\dot{\boldsymbol{\lambda}} = -\boldsymbol{Q}\delta\boldsymbol{x} - \boldsymbol{M}\delta\boldsymbol{u} - \boldsymbol{F}^{\mathsf{T}}\delta\boldsymbol{\lambda}.}$$

The terminal boundary condition is

$$\boxed{\delta\boldsymbol{\lambda}(t_f) = \boldsymbol{P}_f\delta\boldsymbol{x}(t_f).}$$

Stationarity and Neighboring Optimal Control

The stationarity condition is

$$\frac{\partial\mathcal{H}_N}{\partial\delta\boldsymbol{u}} = \boldsymbol{0}.$$

Therefore,

$$\boldsymbol{M}^{\mathsf{T}}\delta\boldsymbol{x} + \boldsymbol{R}\delta\boldsymbol{u} + \boldsymbol{G}^{\mathsf{T}}\delta\boldsymbol{\lambda} = \boldsymbol{0}.$$

Assuming $\boldsymbol{R}$ is invertible,

$$\boxed{\delta\boldsymbol{u}^* = -\boldsymbol{R}^{-1}\left(\boldsymbol{M}^{\mathsf{T}}\delta\boldsymbol{x} + \boldsymbol{G}^{\mathsf{T}}\delta\boldsymbol{\lambda}\right).}$$

Coupled State–Costate System

Substituting the stationarity condition into the state dynamics gives

$$\boxed{\delta\dot{\boldsymbol{x}} = \left(\boldsymbol{F} - \boldsymbol{G}\boldsymbol{R}^{-1}\boldsymbol{M}^{\mathsf{T}}\right)\delta\boldsymbol{x} - \boldsymbol{G}\boldsymbol{R}^{-1}\boldsymbol{G}^{\mathsf{T}}\delta\boldsymbol{\lambda}.}$$

Substitution into the costate equation gives

$$\boxed{\delta\dot{\boldsymbol{\lambda}} = -\left(\boldsymbol{Q} - \boldsymbol{M}\boldsymbol{R}^{-1}\boldsymbol{M}^{\mathsf{T}}\right)\delta\boldsymbol{x} - \left(\boldsymbol{F}^{\mathsf{T}} - \boldsymbol{M}\boldsymbol{R}^{-1}\boldsymbol{G}^{\mathsf{T}}\right)\delta\boldsymbol{\lambda}.}$$

The boundary conditions are split:

$$\begin{aligned}\delta\boldsymbol{x}(t_0) &= \delta\boldsymbol{x}_0,\\ \delta\boldsymbol{\lambda}(t_f) &= \boldsymbol{P}_f\delta\boldsymbol{x}(t_f).\end{aligned}$$

Thus, the neighboring necessary conditions form a linear two-point boundary-value problem.

Riccati Transformation

Assume a linear relationship between neighboring costate and neighboring state:

$$\boxed{\delta\boldsymbol{\lambda}(t) = \boldsymbol{P}(t)\delta\boldsymbol{x}(t).}$$

Differentiating,

$$\delta\dot{\boldsymbol{\lambda}} = \dot{\boldsymbol{P}}\delta\boldsymbol{x} + \boldsymbol{P}\delta\dot{\boldsymbol{x}}.$$

Substituting the state and costate equations and requiring the result to hold for all $\delta\boldsymbol{x}$ yields a matrix Riccati differential equation.

Generalized Riccati Differential Equation

The resulting equation is

$$\begin{aligned}\dot{\boldsymbol{P}} = & -\boldsymbol{Q} + \boldsymbol{M}\boldsymbol{R}^{-1}\boldsymbol{M}^{\mathsf{T}} \\ & - \left(\boldsymbol{F}^{\mathsf{T}} - \boldsymbol{M}\boldsymbol{R}^{-1}\boldsymbol{G}^{\mathsf{T}}\right)\boldsymbol{P} \\ & - \boldsymbol{P}\left(\boldsymbol{F} - \boldsymbol{G}\boldsymbol{R}^{-1}\boldsymbol{M}^{\mathsf{T}}\right) \\ & + \boldsymbol{P}\boldsymbol{G}\boldsymbol{R}^{-1}\boldsymbol{G}^{\mathsf{T}}\boldsymbol{P}.\end{aligned}$$

Equivalently,

$$\begin{aligned}-\dot{\boldsymbol{P}} = & \boldsymbol{Q} + \boldsymbol{F}^{\mathsf{T}}\boldsymbol{P} + \boldsymbol{P}\boldsymbol{F} \\ & - \left(\boldsymbol{P}\boldsymbol{G} + \boldsymbol{M}\right)\boldsymbol{R}^{-1}\left(\boldsymbol{G}^{\mathsf{T}}\boldsymbol{P} + \boldsymbol{M}^{\mathsf{T}}\right).\end{aligned}$$

The terminal condition is

$$\boxed{\boldsymbol{P}(t_f) = \boldsymbol{P}_f.}$$

This equation is integrated backward from t_f to t_0.

Neighboring Optimal Feedback Law

Using

$$\delta\boldsymbol{\lambda} = \boldsymbol{P}\delta\boldsymbol{x}$$

in the stationarity condition,

$$\delta\boldsymbol{u}^* = -\boldsymbol{R}^{-1}\left(\boldsymbol{M}^{\mathsf{T}} + \boldsymbol{G}^{\mathsf{T}}\boldsymbol{P}\right)\delta\boldsymbol{x}.$$

Define the time-varying gain

$$\boxed{\boldsymbol{K}(t) = \boldsymbol{R}^{-1}(t)\left[\boldsymbol{M}^{\mathsf{T}}(t) + \boldsymbol{G}^{\mathsf{T}}(t)\boldsymbol{P}(t)\right].}$$

Then

$$\boxed{\delta \boldsymbol{u}^*(t) = -\boldsymbol{K}(t)\delta \boldsymbol{x}(t).}$$

The implemented control becomes

$$\boxed{\boldsymbol{u}(t) = \boldsymbol{u}^*(t) - \boldsymbol{K}(t)\left[\boldsymbol{x}(t) - \boldsymbol{x}^*(t)\right].}$$

This is a trajectory-tracking feedback law centered on the nonlinear optimal solution.

Closed-Loop Neighboring Dynamics

Substituting the feedback law into the linearized dynamics gives

$$\delta \dot{\boldsymbol{x}} = [\boldsymbol{F} - \boldsymbol{G}\boldsymbol{K}]\,\delta \boldsymbol{x}.$$

Hence,

$$\boxed{\delta \dot{\boldsymbol{x}} = \left[\boldsymbol{F} - \boldsymbol{G}\boldsymbol{R}^{-1}\left(\boldsymbol{M}^{\mathsf{T}} + \boldsymbol{G}^{\mathsf{T}}\boldsymbol{P}\right)\right]\delta \boldsymbol{x}.}$$

Block-Diagram Interpretation

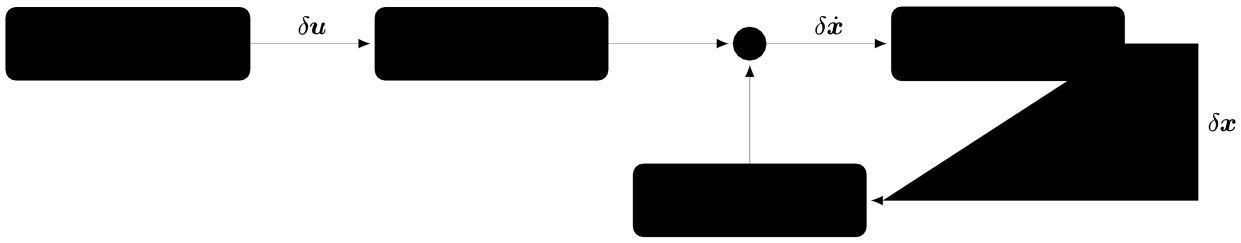

Figure 27: Neighboring optimal feedback around the nominal trajectory.

Interpretation of the Cross Matrix M

The matrix

$$\boldsymbol{M} = L^*_{\boldsymbol{x}\boldsymbol{u}}$$

captures local coupling between state and control variations in the running cost.

When $\boldsymbol{M} = \boldsymbol{0}$, the generalized Riccati equation reduces to the standard finite-horizon time-varying LQR equation:

$$-\dot{\boldsymbol{P}} = \boldsymbol{Q} + \boldsymbol{F}^{\mathsf{T}}\boldsymbol{P} + \boldsymbol{P}\boldsymbol{F} - \boldsymbol{P}\boldsymbol{G}\boldsymbol{R}^{-1}\boldsymbol{G}^{\mathsf{T}}\boldsymbol{P}.$$

The feedback gain becomes

$$K = R^{-1} G^{\top} P.$$

Completing the Square

The running quadratic form can be written as

$$\begin{aligned} & \delta x^{\top} Q \delta x + 2 \delta x^{\top} M \delta u + \delta u^{\top} R \delta u \\ = & \left(\delta u + R^{-1} M^{\top} \delta x\right)^{\top} R \left(\delta u + R^{-1} M^{\top} \delta x\right) \\ & + \delta x^{\top} \left(Q - M R^{-1} M^{\top}\right) \delta x. \end{aligned}$$

This shows why the effective state penalty is

$$\boxed{Q_{\text{eff}} = Q - M R^{-1} M^{\top}.}$$

Conditions for a Well-Defined Neighboring Controller

Several assumptions are important.

Positive Definiteness of R

Because the feedback law requires R^{-1},

$$\boxed{R(t) \succ 0}$$

is commonly imposed.
Positive semidefiniteness is insufficient when R becomes singular.

State Penalty

Typically,

$$Q(t) \succeq 0, \qquad P_f \succeq 0.$$

With a cross term, a stronger convexity condition may be expressed through the block Hessian:

$$\begin{bmatrix} Q & M \\ M^{\top} & R \end{bmatrix} \succeq 0.$$

Controllability or Stabilizability

The pair

$$(\boldsymbol{F}(t), \boldsymbol{G}(t))$$

should be controllable or at least stabilizable over the interval.

Otherwise, some deviations cannot be corrected.

Observability or Detectability

The state penalty and system dynamics should provide adequate detectability. In standard LQR language, an appropriate pair related to

$$(\boldsymbol{F}, \boldsymbol{Q}^{1/2})$$

should be detectable.

Small Perturbations

The method relies on local approximations:

$$\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) \approx \boldsymbol{f}^* + \boldsymbol{F}\delta\boldsymbol{x} + \boldsymbol{G}\delta\boldsymbol{u},$$
$$J \approx J^* + \frac{1}{2}\delta^2 J.$$

Therefore,

$$\|\delta\boldsymbol{x}\|, \qquad \|\delta\boldsymbol{u}\|$$

must remain sufficiently small.

Neighboring Control versus Reoptimization

Two correction strategies are possible after a disturbance.

Neighboring-Control Correction

Use

$$\delta\boldsymbol{u} = -\boldsymbol{K}\delta\boldsymbol{x}.$$

Advantages include:

- rapid online evaluation;
- no repeated nonlinear optimization;
- locally optimal correction;
- simple feedback implementation.

The limitation is local validity.

Full Reoptimization

Resolve the nonlinear optimal control problem from the new state.
Advantages include:

- handles large deviations;
- incorporates nonlinearities directly;
- can update active constraints and trajectory structure.

The limitation is computational cost.

> **Practical Decision**
> Use neighboring optimal control for small deviations. Use full replanning or reoptimization when deviations are too large for the local approximation to remain accurate.

Offline and Online Computation

A practical implementation can separate computation into two stages.

Offline Stage

1. solve the nonlinear optimal control problem;
2. store $\boldsymbol{x}^*(t)$ and $\boldsymbol{u}^*(t)$;
3. compute $\boldsymbol{F}$, $\boldsymbol{G}$, $\boldsymbol{Q}$, $\boldsymbol{M}$, and $\boldsymbol{R}$;
4. integrate the Riccati equation backward;
5. store $\boldsymbol{P}(t)$ or $\boldsymbol{K}(t)$.

Online Stage

At each control update:

1. estimate the current state $\hat{\boldsymbol{x}}(t)$;
2. compute

$$\delta \boldsymbol{x}(t) = \hat{\boldsymbol{x}}(t) - \boldsymbol{x}^*(t);$$

3. evaluate the stored gain $\boldsymbol{K}(t)$;
4. compute

$$\delta \boldsymbol{u} = -\boldsymbol{K}(t)\delta \boldsymbol{x};$$

5. apply

$$\boldsymbol{u} = \boldsymbol{u}^* + \delta\boldsymbol{u}.$$

The Riccati equation need not be integrated online if the nominal trajectory and gain schedule are known in advance.

Parallel-Processor Interpretation

In some architectures, one processor may compute or update the Riccati solution while another applies the real-time feedback law. However, for a fixed nominal trajectory, it is often preferable to compute the gain schedule offline and store it.

Parallel online computation may be useful when:

- the nominal trajectory changes;
- the final time changes;
- model parameters are updated;
- gains must be recomputed adaptively.

Relation to Time-Varying LQR

Neighboring optimal control is closely related to trajectory linearization followed by finite-horizon time-varying LQR.

The distinction is that its weighting matrices are not chosen arbitrarily. They arise from the second derivatives of the original nonlinear optimal control problem:

$$\begin{aligned}\boldsymbol{Q} &= L^*_{\boldsymbol{xx}},\\ \boldsymbol{M} &= L^*_{\boldsymbol{xu}},\\ \boldsymbol{R} &= L^*_{\boldsymbol{uu}},\\ \boldsymbol{P}_f &= \Phi^*_{\boldsymbol{xx}}.\end{aligned}$$

Thus, the local controller is tied directly to the original nonlinear objective.

Connection to Differential Dynamic Programming

Neighboring optimal control and differential dynamic programming share the use of:

- local linearization of dynamics;
- second-order cost information;
- backward Riccati-like recursions;
- local feedback policies around a nominal trajectory.

Differential dynamic programming goes further by iteratively updating the nominal trajectory and repeating the local approximation.

Implementation Workflow

A complete workflow is:

1. solve the nominal nonlinear optimal control problem;
2. verify the nominal state, control, and constraints;
3. compute first derivatives of the dynamics;
4. compute second derivatives of terminal and running costs;
5. construct the neighboring LQ problem;
6. check symmetry and definiteness;
7. solve the generalized Riccati equation backward;
8. construct the gain schedule;
9. simulate the perturbed nonlinear system;
10. compare neighboring feedback with open-loop execution;
11. determine the perturbation range over which the approximation remains accurate.

MATLAB Skeleton

```
t = linspace(t0,tf,N);

% F(:,:,k), G(:,:,k), Q(:,:,k), M(:,:,k), and R(:,:,k)
% are evaluated along x_star(t), u_star(t).

% Integrate P backward from P(tf) = Pf.
P = zeros(nx,nx,N);
P(:,:,N) = Pf;

for k = N: -1:2
    dt = t(k) -t(k -1);

    Fk = F(:,:,k);
    Gk = G(:,:,k);
    Qk = Q(:,:,k);
    Mk = M(:,:,k);
    Rk = R(:,:,k);
    Pk = P(:,:,k);

    Pdot = -Qk ...
        + Mk*(Rk\Mk') ...
        - (Fk' - Mk*(Rk\Gk'))*Pk ...
        - Pk*(Fk - Gk*(Rk\Mk')) ...
        + Pk*Gk*(Rk\Gk')*Pk;

    % Backward Euler illustration.
    P(:,:,k -1) = Pk - dt*Pdot;
end

% Gain schedule.
K = zeros(nu,nx,N);
for k = 1:N
```

```
    K(:,:,k) = R(:,:,k) \ ...
        (M(:,:,k)' + G(:,:,k)'*P(:,:,k));
end

% Online correction:
% delta_x = x_measured - x_star;
% delta_u = -K(:,:,k)*delta_x;
% u = u_star + delta_u;
```

For high-quality computation, use a suitable ODE solver rather than the illustrative backward-Euler update.

Verification Tests

A neighboring optimal controller should be tested for:

- zero correction on the nominal trajectory;
- correct Riccati terminal condition;
- symmetry of $\boldsymbol{P}(t)$;
- positive definiteness of $\boldsymbol{R}(t)$;
- bounded feedback gains;
- closed-loop decay of small perturbations;
- performance under nonlinear dynamics;
- sensitivity to state-estimation error;
- actuator limits;
- failure under large perturbations.

Common Errors

- retaining only first-order cost terms even though $\delta J = 0$;
- omitting the mixed Hessian $\boldsymbol{M}$;
- using the wrong Riccati terminal condition;
- integrating the Riccati equation forward instead of backward;
- confusing $\delta \boldsymbol{u}$ with the full control $\boldsymbol{u}$;
- applying the local controller too far from the nominal trajectory;
- assuming $\boldsymbol{R}^{-1}$ exists without checking definiteness;
- linearizing about a nonoptimal or dynamically inconsistent trajectory.

Summary

1. Neighboring optimal control begins with a nominal nonlinear optimal trajectory.
2. The state, control, and costate are perturbed around that trajectory.
3. The first variation vanishes on the optimal solution.

4. Therefore, the cost must be retained to second order.
5. The dynamics are linearized to first order.
6. The result is a time-varying LQ problem with a state–control cross term.
7. The neighboring Hamiltonian produces linear state and costate equations.
8. A Riccati transformation converts the two-point boundary-value problem into a backward matrix differential equation.
9. The optimal correction is a time-varying feedback law.
10. The total control is the nominal optimal control plus the neighboring correction.
11. The method is appropriate for small perturbations.
12. Large deviations generally require nonlinear reoptimization.

Exercises

1. **Equality path constraints and the augmented Hamiltonian**

   Consider the Bolza problem

   $$\min_{\boldsymbol{u}(\cdot)} \; J = \Phi\big(\boldsymbol{x}(t_f)\big) + \int_{t_0}^{t_f} L(\boldsymbol{x}, \boldsymbol{u}, t)\, \mathrm{d}t,$$

   subject to

   $$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, t)$$

   and the mixed equality path constraint

   $$\boldsymbol{c}(\boldsymbol{x}, \boldsymbol{u}, t) = \boldsymbol{0}.$$

   (a) Introduce a time-varying multiplier $\boldsymbol{\mu}(t)$ for the path constraint and construct the augmented Hamiltonian

   $$\overline{H} = L + \boldsymbol{\lambda}^{\mathsf{T}} \boldsymbol{f} + \boldsymbol{\mu}^{\mathsf{T}} \boldsymbol{c}.$$

   (b) Derive the state equation, costate equation, and control-stationarity condition in terms of $\overline{H}$.

   (c) Show that variation with respect to $\boldsymbol{\mu}(t)$ recovers the path constraint.

   (d) Specialize the result to a control-only constraint

   $$\boldsymbol{c}(\boldsymbol{u}, t) = \boldsymbol{0}.$$

   (e) Specialize the result to a state-only constraint

   $$\boldsymbol{c}(\boldsymbol{x}, t) = \boldsymbol{0}$$

   and identify the additional term in the costate equation.

2. **Inequality path constraints and complementarity**

   Suppose a minimization problem contains the scalar inequality path constraint

   $$c(\boldsymbol{x}, \boldsymbol{u}, t) \le 0.$$

   Using

   $$\overline{H} = H + \mu c,$$

   derive the pointwise conditions

   $$c \le 0, \qquad \mu \ge 0, \qquad \mu c = 0.$$

Then:

(a) derive the necessary conditions on an inactive arc;

(b) derive the necessary conditions on an active arc;

(c) explain why the active-set structure is generally unknown in advance;

(d) for the control bound

$$u(t) - u_{\max} \le 0,$$

determine the multiplier behavior when $u(t) < u_{\max}$ and when $u(t) = u_{\max}$;

(e) state how the inequality would be imposed in a direct-collocation nonlinear program.

3. **Order of a state inequality constraint**

Consider the double integrator

$$\dot{x}_1 = x_2, \qquad \dot{x}_2 = u,$$

with the state path constraint

$$c(\boldsymbol{x}) = x_1 - x_{\max} \le 0.$$

(a) Differentiate c repeatedly along the dynamics until the control appears explicitly.

(b) Show that the constraint has order two.

(c) On a constrained arc, derive the tangency conditions

$$x_1 = x_{\max}, \qquad x_2 = 0.$$

(d) Determine the control required to remain on the constrained arc.

(e) State the consistency conditions that must hold at entry into the constrained arc.

(f) Explain why enforcing $x_1 \le x_{\max}$ only at collocation nodes does not guarantee continuous-time feasibility.

(g) Formulate a dense-grid verification test for the reconstructed state trajectory.

4. **Principle of optimality and the HJB equation**

Define the optimal cost-to-go

$$J^*(\boldsymbol{x}, t) = \min_{\boldsymbol{u}(\cdot)} \left[\Phi\big(\boldsymbol{x}(t_f)\big) + \int_t^{t_f} L(\boldsymbol{x}(\tau), \boldsymbol{u}(\tau), \tau)\, \mathrm{d}\tau \right].$$

Starting from Bellman's principle of optimality over the short interval $[t, t + \Delta t]$:

(a) write the dynamic-programming recursion;

(b) expand $J^*(\boldsymbol{x}(t + \Delta t), t + \Delta t)$ to first order;

(c) derive the Hamilton–Jacobi–Bellman equation

$$0 = J_t^* + \min_{\boldsymbol{u}} \left[L + \nabla_{\boldsymbol{x}} J^{*\mathsf{T}} \boldsymbol{f} \right];$$

(d) derive the terminal condition for J^*;

(e) show that, along the optimal trajectory,

$$\frac{\mathrm{d}J^*}{\mathrm{d}t} = -L(\boldsymbol{x}^*, \boldsymbol{u}^*, t);$$

(f) identify the relationship between the value-function gradient and the costate.

5. **Scalar HJB equation and Riccati feedback**
   Consider

$$\dot{x} = -2x + u$$

with

$$J = \frac{1}{2}x^2(t_f) + \frac{1}{2}\int_t^{t_f} \left(x^2 + u^2\right)\, \mathrm{d}\tau.$$

Assume

$$J^*(x,t) = \frac{1}{2}P(t)x^2.$$

(a) Derive J_t^* and J_x^*.

(b) Substitute the assumed value function into the HJB equation.

(c) Minimize the HJB expression with respect to u.

(d) Derive the optimal feedback law.

(e) Show that $P(t)$ satisfies

$$\dot{P} = P^2 + 4P - 1.$$

(f) Show that the terminal condition is

$$P(t_f) = 1.$$

(g) Derive the optimal closed-loop state equation.

(h) Show that

$$\lambda(t) = P(t)x^*(t).$$

6. **Matrix HJB formulation of finite-horizon LQR**

Consider

$$\dot{\boldsymbol{x}} = A(t)\boldsymbol{x} + B(t)\boldsymbol{u},$$

with performance index

$$J = \frac{1}{2}\boldsymbol{x}^{\mathsf{T}}(t_f)S_f\boldsymbol{x}(t_f) + \frac{1}{2}\int_t^{t_f}\left[\boldsymbol{x}^{\mathsf{T}}Q\boldsymbol{x} + \boldsymbol{u}^{\mathsf{T}}R\boldsymbol{u}\right]\,\mathrm{d}\tau,$$

where $Q(t) = Q^{\mathsf{T}}(t) \succeq 0$ and $R(t) = R^{\mathsf{T}}(t) \succ 0$.

Assume

$$J^*(\boldsymbol{x}, t) = \frac{1}{2}\boldsymbol{x}^{\mathsf{T}}S(t)\boldsymbol{x}.$$

(a) Derive

$$J_t^* = \frac{1}{2}\boldsymbol{x}^{\mathsf{T}}\dot{S}\boldsymbol{x}.$$

(b) Derive the minimizing control.

(c) Derive the Riccati differential equation

$$-\dot{S} = A^{\mathsf{T}}S + SA - SBR^{-1}B^{\mathsf{T}}S + Q.$$

(d) Derive the terminal condition

$$S(t_f) = S_f.$$

(e) Derive the feedback gain and the closed-loop state equation.

(f) Show that

$$\boldsymbol{\lambda} = S\boldsymbol{x}^*.$$

(g) Explain why the Riccati equation is integrated backward while the state equation is integrated forward.

7. **Bang–bang and singular control**

Consider

$$\min_{u(\cdot)} J = \int_0^1 x_2(t)u(t)\,\mathrm{d}t$$

subject to

$$\dot{x}_1 = x_2, \qquad \dot{x}_2 = -x_2 + u,$$

$$0 \le u(t) \le 2,$$

and

$$x_1(0) = 0, \qquad x_2(0) = 1, \qquad x_1(1) = 1, \qquad x_2(1) = 1.$$

(a) Construct the Hamiltonian.

(b) Derive the costate equations.

(c) Define the switching function

$$\phi(t) = x_2(t) + \lambda_2(t).$$

(d) Derive the bang–bang minimization rule for $\phi > 0$ and $\phi < 0$.

(e) Impose

$$\phi = \dot{\phi} = 0$$

on a candidate singular arc.

(f) Derive the singular control.

(g) Verify that the singular state and control satisfy all endpoint conditions.

(h) Determine the order of the singular arc and state the corresponding generalized Legendre–Clebsch check.

8. **Analytical direct shooting for the minimum-energy double integrator**

Consider

$$\min_{u(\cdot)} \; J = \frac{1}{2} \int_0^T u^2(t)\, \mathrm{d}t$$

subject to

$$\dot{x}_1 = x_2, \qquad \dot{x}_2 = u,$$

with fixed endpoint data

$$x_1(0) = a_0, \qquad x_2(0) = v_0, \qquad x_1(T) = a_f, \qquad x_2(T) = v_f.$$

(a) Derive the Hamiltonian and costate equations.

(b) Show that λ_1 is constant and λ_2 is affine in time.

(c) Derive the optimal control as an affine function of time.

(d) Integrate the state equations analytically.

(e) Reduce the terminal conditions to a 2×2 linear system for the two unknown initial costates.

(f) Solve this system explicitly.

(g) Explain why the original infinite-dimensional problem has been reduced to a finite-dimensional shooting problem.

9. **General neighboring-optimal-control derivation**

Let $(\boldsymbol{x}^*, \boldsymbol{u}^*, \boldsymbol{\lambda}^*)$ be a nominal nonlinear optimal trajectory. Define

$$\delta\boldsymbol{x} = \boldsymbol{x} - \boldsymbol{x}^*, \qquad \delta\boldsymbol{u} = \boldsymbol{u} - \boldsymbol{u}^*, \qquad \delta\boldsymbol{\lambda} = \boldsymbol{\lambda} - \boldsymbol{\lambda}^*.$$

Assume the neighboring dynamics and second-order cost are

$$\delta\dot{\boldsymbol{x}} = F(t)\delta\boldsymbol{x} + G(t)\delta\boldsymbol{u},$$

$$\delta^2 J = \frac{1}{2}\delta\boldsymbol{x}^{\mathsf{T}}(t_f)P_f\delta\boldsymbol{x}(t_f) + \frac{1}{2}\int_{t_0}^{t_f}\left[\delta\boldsymbol{x}^{\mathsf{T}}Q\delta\boldsymbol{x} + 2\delta\boldsymbol{x}^{\mathsf{T}}M\delta\boldsymbol{u} + \delta\boldsymbol{u}^{\mathsf{T}}R\delta\boldsymbol{u}\right]\,\mathrm{d}t.$$

(a) Construct the neighboring Hamiltonian.

(b) Derive the neighboring costate equation.

(c) Derive the stationarity condition.

(d) Show that

$$\delta\boldsymbol{u} = -R^{-1}\left(M^{\mathsf{T}}\delta\boldsymbol{x} + G^{\mathsf{T}}\delta\boldsymbol{\lambda}\right).$$

(e) Assume

$$\delta\boldsymbol{\lambda} = P(t)\delta\boldsymbol{x}$$

and derive the generalized Riccati equation.

(f) Show that

$$P(t_f) = P_f.$$

(g) Derive the neighboring feedback gain

$$K(t) = R^{-1}\left(M^{\mathsf{T}} + G^{\mathsf{T}}P\right).$$

(h) Show that the standard time-varying LQR Riccati equation is recovered when $M = \mathbf{0}$.

10. **Neighboring feedback for a nonlinear scalar system**

Consider the nonlinear dynamics

$$\dot{x} = x + x^2 + u$$

and the nominal trajectory

$$x^*(t) = 0, \qquad u^*(t) = 0.$$

The neighboring cost is

$$\delta^2 J = \frac{1}{2} p_f \delta x^2(t_f) + \frac{1}{2} \int_0^{t_f} \left(q\delta x^2 + 2m\delta x\, \delta u + r\delta u^2 \right) \mathrm{d}t,$$

with $r > 0$.

(a) Linearize the dynamics about the nominal trajectory and identify F and G.

(b) Derive the scalar neighboring Riccati equation.

(c) Derive the feedback gain

$$K(t) = \frac{m + GP(t)}{r}.$$

(d) Write the neighboring correction

$$\delta u(t) = -K(t)\delta x(t).$$

(e) Write the complete implemented control $u = u^* + \delta u$.

(f) Show that the correction is zero on the nominal trajectory.

(g) Describe a numerical test using several initial perturbation magnitudes.

(h) State the residuals and qualitative behaviors that should be checked to determine when the neighboring approximation is no longer reliable.

Chapter 8: Verification and Engineering Applications

Chapter 8: Verification and Engineering Applications

From optimality conditions to defensible engineering conclusions

Theory and algorithms are valuable only when they lead to defensible engineering conclusions. This chapter begins with a verification-oriented workflow and then applies the book's recurring ideas to three contrasting problems: fixed-time ascent steering, fuel-efficient lunar landing, and minimum-time aircraft climb through an energy-state approximation. Together they illustrate analytical structure, constrained control, reduced-order modeling, and the need to check every claimed optimum.

Chapter contents

Capstone Projects, Solution Verification, and Practical Methodology

This section synthesizes the complete optimal control workflow developed throughout the course. Rather than introducing new theory, it explains how calculus of variations, Pontryagin's Minimum Principle, indirect methods, direct methods, dynamic programming, and numerical optimization fit together in solving practical engineering problems.

Course Roadmap

Topics include:

1. Calculus of Variations
2. Pontryagin's Minimum Principle
3. Indirect Numerical Methods
4. Direct Shooting
5. Multiple Shooting
6. Direct Collocation
7. Pseudospectral Methods
8. Costate Estimation
9. Dynamic Programming and the HJB Equation

Indirect Solution Workflow

The indirect approach consists of:

1. Derive first-order optimality conditions.
2. Form state, costate, and stationarity equations.
3. Solve the resulting TPBVP.
4. Recover the optimal state, control, and costate trajectories.
5. Verify second-order conditions.

Extremals versus Optimal Solutions

The first-order necessary conditions generate an *extremal*, not necessarily an optimal solution. A candidate trajectory satisfies the necessary conditions but must still satisfy second-order sufficient conditions for local optimality.

Legendre–Clebsch Condition

For regular problems,

$$\frac{\partial^2 H}{\partial u^2} \succ 0$$

is a local minimum test. For LQR,

$$H = \tfrac{1}{2}x^T Q x + \tfrac{1}{2}u^T R u + \lambda^T (Ax + Bu),$$

so

$$\frac{\partial^2 H}{\partial u^2} = R.$$

Thus positive definiteness of R guarantees satisfaction of the Legendre–Clebsch condition.

Direct Methods

Direct methods parameterize the control (or state and control), transcribe the problem into an NLP, solve the NLP, then reconstruct trajectories.

Comparing Direct and Indirect Solutions

Recommended comparisons:

- state trajectories;
- control histories;
- cost values;
- costates (when available through collocation multipliers);
- convergence under mesh or polynomial refinement.

Project Methodology

A good capstone project should:

1. choose a tractable nonlinear system;
2. solve it by one indirect method;
3. solve it by one direct method;
4. compare both solutions;
5. explain any discrepancies;
6. discuss numerical robustness and computational cost.

Common Pitfalls

- overly complicated dynamics;
- bang–bang controls approximated by low-order polynomials;
- singular arcs ignored;
- insufficient discretization;
- comparing only controls while ignoring costates.

Research Perspective

Engineering research requires interpreting imperfect numerical results, diagnosing inconsistencies, and validating conclusions using theory rather than relying solely on software output.

Suggested Capstone Workflow

1. Select a manageable problem.
2. Derive optimality conditions.
3. Implement an indirect solver.
4. Implement a direct solver.
5. Verify convergence.
6. Compare trajectories and costs.
7. Interpret discrepancies.
8. Draw engineering conclusions.

Connection. The workflow becomes concrete in a fixed-time ascent problem whose costates yield an analytical bilinear tangent steering law.

Bilinear Tangent Steering Law for Fixed-Time Ascent

This section develops an analytical optimal control solution for a fixed-time planar steering problem. The objective is to steer an accelerating object from the origin to a prescribed terminal altitude while maximizing its terminal horizontal velocity and forcing the terminal vertical velocity to zero.

The example illustrates several central ideas:

- conversion of a maximization problem into minimization form;
- Hamiltonian construction;
- costate integration;
- transversality conditions;
- analytical recovery of the optimal steering law;
- interpretation of a bilinear tangent law;
- explicit integration of the state equations.

Problem Statement

Consider the planar dynamics

$$\begin{aligned} \dot{x} &= u, \\ \dot{y} &= v, \\ \dot{u} &= a\cos\beta, \\ \dot{v} &= a\sin\beta, \end{aligned}$$

where:

- $x(t)$ is horizontal position;
- $y(t)$ is vertical position;
- $u(t)$ is horizontal velocity;
- $v(t)$ is vertical velocity;
- a is the acceleration magnitude;
- $\beta(t)$ is the steering angle and control input.

The final time T is fixed.
The initial conditions are

$$x(0) = 0, \qquad y(0) = 0, \qquad u(0) = 0, \qquad v(0) = 0.$$

The terminal conditions are

$$y(T) = H, \qquad v(T) = 0.$$

The terminal horizontal position $x(T)$ and terminal horizontal velocity $u(T)$ are free.
The objective is

$$\max_{\beta(\cdot)} u(T).$$

Equivalently,

$$\boxed{\min_{\beta(\cdot)} \Phi = -u(T).}$$

Physical Interpretation

The acceleration vector has constant magnitude a and direction β:

$$\boldsymbol{a} = a \begin{bmatrix} \cos\beta \\ \sin\beta \end{bmatrix}.$$

Thus, the control does not independently choose horizontal and vertical acceleration. It chooses the direction of a vector whose magnitude is fixed.

The terminal constraint

$$v(T) = 0$$

requires the final velocity to be horizontal.

Hamiltonian

Since there is no running cost,

$$L = 0.$$

The Hamiltonian is

$$\mathcal{H} = \lambda_x u + \lambda_y v + \lambda_u a\cos\beta + \lambda_v a\sin\beta.$$

Therefore,

$$\boxed{\mathcal{H} = \lambda_x u + \lambda_y v + a\lambda_u \cos\beta + a\lambda_v \sin\beta.}$$

Costate Equations

The costate equations are

$$\dot{\lambda}_x = -\frac{\partial \mathcal{H}}{\partial x} = 0,$$
$$\dot{\lambda}_y = -\frac{\partial \mathcal{H}}{\partial y} = 0,$$
$$\dot{\lambda}_u = -\frac{\partial \mathcal{H}}{\partial u} = -\lambda_x,$$
$$\dot{\lambda}_v = -\frac{\partial \mathcal{H}}{\partial v} = -\lambda_y.$$

Hence,

$$\lambda_x(t) = c_1,$$
$$\lambda_y(t) = c_2,$$
$$\lambda_u(t) = -c_1 t + c_3,$$
$$\lambda_v(t) = -c_2 t + c_4.$$

Optimal Control Condition

For an interior solution,

$$\frac{\partial \mathcal{H}}{\partial \beta} = 0.$$

Therefore,

$$-a\lambda_u \sin\beta + a\lambda_v \cos\beta = 0.$$

Assuming $\lambda_u \neq 0$,

$$\lambda_v \cos\beta = \lambda_u \sin\beta.$$

Thus,

$$\boxed{\tan\beta = \frac{\lambda_v}{\lambda_u}.}$$

Using the costate expressions,

$$\boxed{\tan\beta(t) = \frac{-c_2 t + c_4}{-c_1 t + c_3}.}$$

The right-hand side is a linear function divided by another linear function.

Bilinear Tangent Law
The optimal steering law is called a bilinear tangent law because $\tan\beta$ is a bilinear, or linear-fractional, function of time.

Transversality Conditions

The terminal cost is

$$\Phi = -u(T).$$

The terminal constraints are

$$\begin{aligned}\psi_1 &= y(T) - H = 0,\\ \psi_2 &= v(T) = 0.\end{aligned}$$

Introduce terminal multipliers ν_y and ν_v. The transversality condition is

$$\boldsymbol{\lambda}(T) = \frac{\partial \Phi}{\partial \boldsymbol{x}(T)} + \left(\frac{\partial \boldsymbol{\psi}}{\partial \boldsymbol{x}(T)}\right)^{\mathsf{T}} \boldsymbol{\nu}.$$

Therefore,

$$\begin{aligned}\lambda_x(T) &= 0,\\ \lambda_y(T) &= \nu_y,\\ \lambda_u(T) &= -1,\\ \lambda_v(T) &= \nu_v.\end{aligned}$$

Since λ_x is constant,

$$\boxed{\lambda_x(t) \equiv 0.}$$

Therefore,

$$c_1 = 0.$$

Then

$$\lambda_u(t) = c_3.$$

Since

$$\lambda_u(T) = -1,$$

we obtain

$$\boxed{\lambda_u(t) \equiv -1.}$$

The control law becomes

$$\tan\beta = -\lambda_v.$$

Since

$$\lambda_v = -c_2 t + c_4,$$

we have

$$\tan\beta = c_2 t - c_4.$$

Define constants

$$c = -c_2, \qquad \tan\beta_0 = -c_4.$$

Then

$$\boxed{\tan\beta(t) = \tan\beta_0 - ct.}$$

This is the linear tangent steering law.

Control Reconstruction

Let

$$q(t) = \tan\beta(t) = q_0 - ct,$$

where

$$q_0 = \tan\beta_0.$$

Then

$$\cos\beta = \frac{1}{\sqrt{1+q^2}},$$
$$\sin\beta = \frac{q}{\sqrt{1+q^2}}.$$

Also,

$$\dot{q} = -c.$$

Hence,

$$\mathrm{d}t = -\frac{1}{c}\,\mathrm{d}q.$$

Analytical Horizontal Velocity

The horizontal velocity satisfies

$$\dot{u} = a\cos\beta = \frac{a}{\sqrt{1+q^2}}.$$

Using $\mathrm{d}t = -\mathrm{d}q/c$,

$$u(t) = -\frac{a}{c}\int_{q_0}^{q(t)} \frac{\mathrm{d}q}{\sqrt{1+q^2}}.$$

Since

$$\int \frac{\mathrm{d}q}{\sqrt{1+q^2}} = \operatorname{arsinh}(q) = \ln\left(q + \sqrt{1+q^2}\right),$$

we obtain

$$u(t) = \frac{a}{c}\ln\left[\frac{q_0 + \sqrt{1+q_0^2}}{q(t) + \sqrt{1+q^2(t)}}\right].$$

Using $q = \tan\beta$ and $\sqrt{1+q^2} = \sec\beta$,

$$\boxed{u(t) = \frac{a}{c}\ln\left[\frac{\tan\beta_0 + \sec\beta_0}{\tan\beta(t) + \sec\beta(t)}\right].}$$

Analytical Vertical Velocity

The vertical velocity satisfies

$$\dot{v} = a\sin\beta = a\frac{q}{\sqrt{1+q^2}}.$$

Therefore,

$$v(t) = -\frac{a}{c}\int_{q_0}^{q(t)} \frac{q}{\sqrt{1+q^2}}\,\mathrm{d}q.$$

Since

$$\int \frac{q}{\sqrt{1+q^2}}\,\mathrm{d}q = \sqrt{1+q^2},$$

we obtain

$$\boxed{v(t) = \frac{a}{c}\left[\sec\beta_0 - \sec\beta(t)\right].}$$

Analytical Horizontal Position

The horizontal position is

$$x(t) = \int_0^t u(\tau)\,\mathrm{d}\tau.$$

After integration,

$$x(t) = \frac{a}{c^2}\Bigg[\sec\beta_0 - \sec\beta - \tan\beta\ln\left(\frac{\tan\beta_0 + \sec\beta_0}{\tan\beta + \sec\beta}\right)\Bigg].$$

Therefore,

$$\boxed{x(t) = \frac{a}{c^2}\left[\sec\beta_0 - \sec\beta - \tan\beta\ln\left(\frac{\tan\beta_0 + \sec\beta_0}{\tan\beta + \sec\beta}\right)\right].}$$

Analytical Vertical Position

Similarly,

$$y(t) = \frac{a}{2c^2}\Bigg[(\tan\beta_0 - \tan\beta)(\sec\beta_0 + \sec\beta) - \ln\left(\frac{\tan\beta_0 + \sec\beta_0}{\tan\beta + \sec\beta}\right)\Bigg].$$

Therefore,

$$\boxed{y(t) = \frac{a}{2c^2}\left[(\tan\beta_0 - \tan\beta)(\sec\beta_0 + \sec\beta) - \ln\left(\frac{\tan\beta_0 + \sec\beta_0}{\tan\beta + \sec\beta}\right)\right].}$$

Terminal Vertical-Velocity Condition

The terminal condition

$$v(T) = 0$$

gives

$$\sec\beta_0 = \sec\beta(T).$$

For the nontrivial symmetric solution,

$$\boxed{\beta(T) = -\beta_0.}$$

Since

$$\tan\beta(T) = \tan\beta_0 - cT,$$

we have

$$-\tan\beta_0 = \tan\beta_0 - cT.$$

Thus,

$$\boxed{c = \frac{2\tan\beta_0}{T}.}$$

The steering law becomes

$$\boxed{\tan\beta(t) = \tan\beta_0\left(1 - \frac{2t}{T}\right).}$$

This law is antisymmetric in tangent space about $t = T/2$.

Altitude Equation

Substituting the terminal conditions into the expression for $y(T)$ gives an implicit equation for β_0:

$$\boxed{\frac{4H}{aT^2} = \frac{2\tan\beta_0\sec\beta_0 - \ln\left(\frac{\sec\beta_0+\tan\beta_0}{\sec\beta_0-\tan\beta_0}\right)}{2\tan^2\beta_0}.}$$

This nonlinear scalar equation determines the initial steering angle.
Once β_0 is found, the constant c follows from

$$c = \frac{2\tan\beta_0}{T}.$$

Maximum Terminal Horizontal Velocity

The terminal horizontal velocity is

$$u(T) = \frac{a}{c}\ln\left[\frac{\tan\beta_0 + \sec\beta_0}{-\tan\beta_0 + \sec\beta_0}\right].$$

Using

$$c = \frac{2\tan\beta_0}{T},$$

we obtain

$$\boxed{u_{\max} = \frac{aT}{2\tan\beta_0} \ln\left[\frac{\sec\beta_0 + \tan\beta_0}{\sec\beta_0 - \tan\beta_0}\right].}$$

Terminal Horizontal Position

Using the symmetry condition,

$$x(T) = \frac{T}{2}u(T).$$

Therefore,

$$\boxed{u_{\max} = \frac{2x(T)}{T}.}$$

This relation follows from the symmetry of the steering law and the resulting horizontal-velocity profile.

Qualitative Behavior of the Solution

The tangent of the steering angle varies linearly:

$$\tan\beta(t) = \tan\beta_0\left(1 - \frac{2t}{T}\right).$$

Therefore:

- $\beta(0) = \beta_0$;
- $\beta(T/2) = 0$;
- $\beta(T) = -\beta_0$;
- the initial acceleration has an upward component;
- the midpoint acceleration is horizontal;
- the final acceleration has a downward component;
- the downward component removes vertical velocity at T.

Trajectory Sketch

Numerical Solution Procedure

For specified a, T, and H:

1. solve the scalar altitude equation for β_0;
2. compute

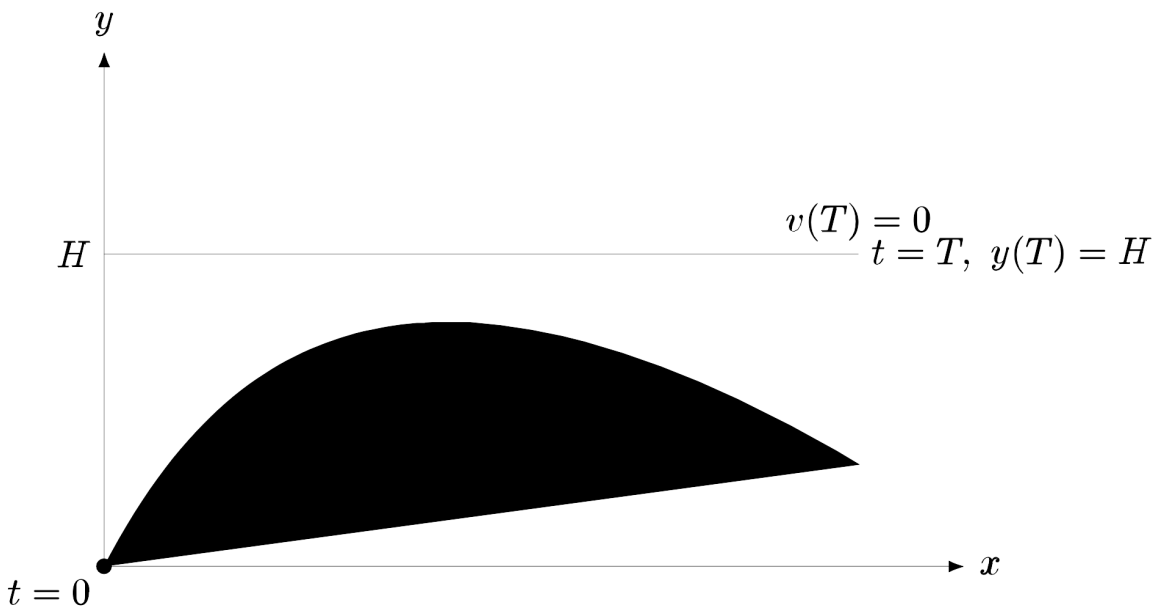

Figure 28: Conceptual terminal-altitude steering problem.

$$c = \frac{2\tan\beta_0}{T};$$

3. evaluate

$$\tan\beta(t) = \tan\beta_0\left(1 - \frac{2t}{T}\right);$$

4. reconstruct $u(t)$, $v(t)$, $x(t)$, and $y(t)$;
5. verify $y(T) = H$ and $v(T) = 0$;
6. compute $u_{\max} = u(T)$.

MATLAB Implementation

```
function session34_solution(a,T,H)

    target = 4*H/(a*T^2);

    altitude_equation = @(beta0) ...
        (2*tan(beta0)*sec(beta0) ...
        - log((sec(beta0)+tan(beta0)) ...
        /(sec(beta0) -tan(beta0)))) ...
        /(2*tan(beta0)^2) ...
        - target;

    beta0 = fzero(altitude_equation,[1e -4,1.4]);
```

```
c = 2*tan(beta0)/T;

t = linspace(0,T,500);
tan_beta = tan(beta0).*(1 -2*t/T);
beta = atan(tan_beta);
sec_beta = sqrt(1+tan_beta.^2);

ratio = (tan(beta0)+sec(beta0)) ...
    ./ (tan_beta+sec_beta);

u = a/c .* log(ratio);

v = a/c .* (sec(beta0) -sec_beta);

x = a/c^2 .* (sec(beta0) -sec_beta ...
    - tan_beta.*log(ratio));

y = a/(2*c^2) .* ( ...
    (tan(beta0) -tan_beta) ...
    .* (sec(beta0)+sec_beta) ...
    - log(ratio));

umax = u(end);

fprintf('beta0 = %.12f rad\n',beta0);
fprintf('c     = %.12f\n',c);
fprintf('x(T)  = %.12f\n',x(end));
fprintf('y(T)  = %.12f\n',y(end));
fprintf('u(T)  = %.12f\n',u(end));
fprintf('v(T)  = %.12f\n',v(end));

figure;
plot(x,y,'LineWidth',1.5);
xlabel('x');
ylabel('y');
grid on;

figure;
plot(t,beta,'LineWidth',1.5);
xlabel('Time');
ylabel('\beta(t)');
grid on;

figure;
plot(t,u,'LineWidth',1.5);
hold on;
plot(t,v,'LineWidth',1.5);
xlabel('Time');
ylabel('Velocity');
```

```
    legend('u','v');
    grid on;
end
```

Verification of the Minimum Principle

The steering-dependent Hamiltonian term is

$$a\lambda_u \cos\beta + a\lambda_v \sin\beta.$$

It can be written as

$$a\sqrt{\lambda_u^2 + \lambda_v^2}\cos(\beta - \theta_\lambda),$$

where

$$\theta_\lambda = \operatorname{atan2}(\lambda_v, \lambda_u).$$

Depending on the sign convention used for the Hamiltonian and whether the problem is posed as minimization or maximization, the minimizing or maximizing steering direction must be selected consistently.

The stationary relation

$$\tan\beta = \frac{\lambda_v}{\lambda_u}$$

must therefore be checked against the appropriate branch of the minimum principle.

Why the Solution Is Historically Important

The control law has the same basic form as steering laws used in powered-flight guidance. Its practical importance comes from the fact that:

- the law has very low computational complexity;
- only a small number of constants must be determined;
- the control can be evaluated rapidly;
- the law is suitable for limited onboard computation;
- the underlying structure arises directly from optimal control theory.

Lessons from Analytical Optimal Control

1. The structure of the Hamiltonian reveals the structure of the steering law.
2. Transversality conditions can eliminate unknown costate constants.
3. Terminal constraints may generate symmetry conditions.
4. Analytical integration can reduce a trajectory optimization problem to a scalar nonlinear equation.
5. A physically meaningful control law may emerge from a highly simplified model.

6. Mathematical structure is often more valuable than brute-force computation.

Common Errors

- forgetting to convert maximization to minimization consistently;
- applying incorrect signs in the Hamiltonian;
- confusing state variables u and v with the control;
- omitting terminal multipliers for fixed terminal components;
- treating all terminal costates as known;
- failing to distinguish $\tan\beta$ from β;
- missing the correct branch of the inverse tangent;
- using $\sec\beta = \sqrt{1+\tan^2\beta}$ without checking angle range;
- skipping verification of the minimum principle;
- failing to verify terminal constraints after solving for β_0.

Summary

1. The system has four states and one steering control.
2. The objective is to maximize terminal horizontal velocity.
3. The Hamiltonian contains no running-cost term.
4. Two costates are constant and two are affine in time.
5. Stationarity gives a bilinear tangent law.
6. Transversality forces $\lambda_x = 0$ and $\lambda_u = -1$.
7. The control reduces to a linear tangent law.
8. For constant acceleration, all state trajectories can be integrated analytically.
9. The terminal vertical-velocity constraint implies $\beta(T) = -\beta_0$.
10. The initial angle is determined by a scalar altitude equation.
11. The maximum terminal horizontal velocity follows explicitly once β_0 is known.

Connection. Soft lunar landing adds a bounded thrust and fuel objective, illustrating how physical limits shape the necessary conditions.

Soft Lunar Landing on a Flat Moon

This section studies the classical 1964 George Meditch soft-landing problem. The objective is to land a lunar module with zero altitude and zero velocity while minimizing fuel consumption. The lecture emphasizes problem formulation and optimality conditions rather than carrying the algebra to completion.

Dynamics

$$\dot{h} = v,$$
$$\dot{v} = -g + u,$$

where h is altitude, v vertical velocity, u the thrust acceleration, and

$$0 \le u \le 3.$$

Initial conditions:

$$h(0) = h_0, \qquad v(0) = v_0.$$

Terminal conditions:

$$h(t_f) = 0, \qquad v(t_f) = 0.$$

The terminal time is free.

Performance Index

Fuel consumption is approximated by

$$J = \int_0^{t_f} u(t)\, dt,$$

which is proportional to total thrust because $u \ge 0$.

Hamiltonian

$$H = 1 + \lambda_h v + \lambda_v(-g + u) = (1 + \lambda_v)u + \lambda_h v - \lambda_v g.$$

Costate Equations

$$\dot{\lambda}_h = 0,$$
$$\dot{\lambda}_v = -\lambda_h.$$

Hence

$$\lambda_h = C_1, \qquad \lambda_v = -C_1 t + C_2.$$

Pontryagin Minimum Principle

Because the Hamiltonian is linear in u, the minimum principle is used directly:

$$u^* = \arg\min_{0 \le u \le 3} H.$$

Therefore

$$u^* = \begin{cases} 0, & 1 + \lambda_v > 0, \\ 3, & 1 + \lambda_v < 0. \end{cases}$$

A singular arc would require

$$1 + \lambda_v \equiv 0,$$

which cannot occur except in the degenerate case, so no singular arc exists.

Bang–Bang Structure

Since λ_v is linear in time, $1 + \lambda_v$ is also linear. Consequently, the optimal control has at most one switching time:

$$0 \to 3 \quad \text{or} \quad 3 \to 0,$$

or remains constant over the entire trajectory.

Free Final Time

Because t_f is free and the Hamiltonian has no explicit time dependence,

$$H(t_f) = 0,$$

giving

$$1 + (1 + \lambda_v(t_f))u(t_f) - g\lambda_v(t_f) = 0,$$

since $v(t_f) = 0$.

Transversality

Both initial and terminal states are completely specified, so the transversality conditions provide no additional costate information.

Solution Strategy

The solution proceeds by:

1. Assume one of the admissible bang–bang structures.

2. Integrate dynamics on each segment.
3. Enforce continuity at the switching time.
4. Apply terminal conditions.
5. Verify the Hamiltonian condition.
6. Compare admissible solutions.

The state trajectories are piecewise:

- velocity: linear,
- altitude: quadratic.

Historical Significance

George Meditch's 1964 paper showed how optimal control could be used to derive fuel-optimal lunar landing trajectories. This problem became one of the classical examples in aerospace optimal control.

Connection. The aircraft climb problem shows a different engineering strategy: reduce the model through a physically meaningful energy coordinate before optimizing.

Energy-State Approximation and Minimum-Time Climb

This section develops the mathematical foundations of the energy-state approximation for aircraft performance optimization. The central application is the minimum-time climb of a supersonic aircraft. The method became historically important because it transformed a difficult high-dimensional trajectory optimization problem into a lower-dimensional problem with a strong physical interpretation.

The key idea is to replace speed and altitude by a total specific energy variable and then use the rate of change of this energy as the primary performance quantity.

The chapter develops:

- the longitudinal aircraft equations of motion;
- the total specific energy equation;
- the reduced-order energy-state approximation;
- the minimum-time climb problem;
- the Hamiltonian and optimality conditions;
- the interpretation of maximizing excess power;
- the non-intuitive dive–climb trajectory;
- numerical solution strategies;
- limitations of the approximation.

Historical Context

The energy-state approximation was used in classic aerospace optimization studies during the 1960s. At that time, computational resources were extremely limited, so simplified models were essential.

The central engineering problem was:

> How should a high-performance aircraft vary its speed, altitude, and flight path angle to reach a specified terminal energy state in minimum time?

A direct climb is not necessarily optimal. In many supersonic aircraft, aerodynamic drag rises sharply near Mach 1. An optimal trajectory may therefore climb, dive to accelerate through the transonic region, and then climb again.

Longitudinal Aircraft Model

Consider planar longitudinal motion. Let

- V be the airspeed;
- γ be the flight-path angle;
- h be altitude;
- x be horizontal range;
- m be mass;
- T be thrust;
- D be drag;

- L be lift;
- α be angle of attack;
- g be gravitational acceleration.

A standard point-mass model is

$$\begin{aligned} m\dot{V} &= T\cos\alpha - D - mg\sin\gamma, \\ mV\dot{\gamma} &= T\sin\alpha + L - mg\cos\gamma, \\ \dot{h} &= V\sin\gamma, \\ \dot{x} &= V\cos\gamma, \\ \dot{m} &= -\dot{m}_f. \end{aligned}$$

The first two equations describe kinetics. The last two describe kinematics.

Aerodynamic Model

Dynamic pressure is

$$q = \frac{1}{2}\rho(h)V^2.$$

Lift and drag are

$$\begin{aligned} L &= qSC_L(M,\alpha), \\ D &= qSC_D(M,\alpha), \end{aligned}$$

where S is reference area and M is Mach number.
A common aerodynamic approximation is

$$C_D = C_{D0}(M) + K(M)C_L^2.$$

Then

$$D = qS\left[C_{D0}(M) + K(M)C_L^2\right].$$

The thrust model may be represented as

$$T = T(V,h,\eta),$$

where η is throttle.

Total Specific Energy

Define total specific mechanical energy

$$E = \frac{V^2}{2} + gh.$$

This quantity combines kinetic and potential energy per unit mass.
Differentiate:

$$\dot{E} = V\dot{V} + g\dot{h}.$$

Using

$$\dot{h} = V \sin\gamma$$

and

$$\dot{V} = \frac{T\cos\alpha - D}{m} - g\sin\gamma,$$

we obtain

$$\dot{E} = V\left[\frac{T\cos\alpha - D}{m} - g\sin\gamma\right] + gV\sin\gamma.$$

The gravity terms cancel:

$$\boxed{\dot{E} = \frac{V}{m}\left(T\cos\alpha - D\right).}$$

For small angle of attack,

$$\cos\alpha \approx 1,$$

so

$$\boxed{\dot{E} \approx \frac{V}{m}(T - D).}$$

The quantity

$$P_s = \dot{E}$$

is called the specific excess power.

Physical Interpretation of Specific Excess Power

Specific excess power measures the rate at which the aircraft can gain total energy.

- If $T > D$, then $\dot{E} > 0$ and the aircraft gains energy.
- If $T = D$, then $\dot{E} = 0$ and total energy remains constant.
- If $T < D$, then $\dot{E} < 0$ and the aircraft loses energy.

The aircraft may exchange kinetic and potential energy without changing total energy. For example, during a constant-energy dive,

$$\frac{V^2}{2} + gh = \text{constant}.$$

Differentiating gives

$$V\dot{V} + g\dot{h} = 0.$$

Thus a decrease in altitude causes an increase in speed.

Energy-State Approximation

The energy-state approximation is based on the following assumptions:

1. Motion is nearly horizontal.
2. Flight-path angle changes slowly.
3. Normal acceleration is small:

$$V\dot{\gamma} \approx 0.$$

4. Thrust normal to the flight path is negligible:

$$T \sin \alpha \approx 0.$$

5. The aircraft is approximately in normal-force equilibrium:

$$L \approx mg.$$

6. Speed and altitude can be parameterized through total energy.

Under these assumptions, the flight-path angle dynamics are neglected.
The reduced equations become

$$\begin{aligned} \dot{E} &= \frac{V}{m}(T - D), \\ \dot{h} &= V \sin \gamma, \\ \dot{x} &= V \cos \gamma. \end{aligned}$$

For a pure performance optimization in the (V, h) plane, the energy equation may be used as the dominant dynamic equation.

Energy Curves in the Speed–Altitude Plane

For constant energy,

$$E = \frac{V^2}{2} + gh.$$

Solving for altitude,

$$\boxed{h = \frac{E}{g} - \frac{V^2}{2g}.}$$

Thus constant-energy curves are parabolic in the (V, h) plane.
The slope is

$$\frac{\mathrm{d}h}{\mathrm{d}V} = -\frac{V}{g}.$$

Moving to higher speed along a constant-energy curve requires a decrease in altitude.

Minimum-Time Climb Problem

Suppose the aircraft starts at

$$V(0) = V_0, \qquad h(0) = h_0,$$

and must reach

$$V(t_f) = V_f, \qquad h(t_f) = h_f.$$

The final energy is

$$E_f = \frac{V_f^2}{2} + gh_f.$$

The objective is

$$\boxed{\min t_f = \min \int_0^{t_f} 1 \, \mathrm{d}t.}$$

In the reduced energy-state problem,

$$\dot{E} = P_s(E, V) = \frac{V}{m} \left[T(E, V) - D(E, V) \right].$$

The control is effectively the speed V selected at each energy level, subject to feasible flight-envelope constraints:

$$V_{\min}(E) \le V \le V_{\max}(E).$$

Time as an Integral over Energy

Since

$$\dot{E} = \frac{\mathrm{d}E}{\mathrm{d}t},$$

we have

$$\mathrm{d}t = \frac{\mathrm{d}E}{\dot{E}}.$$

Therefore,

$$t_f = \int_{E_0}^{E_f} \frac{1}{\dot{E}(E,V)}\,\mathrm{d}E.$$

Thus,

$$\boxed{t_f = \int_{E_0}^{E_f} \frac{m}{V[T(E,V) - D(E,V)]}\,\mathrm{d}E.}$$

At each energy level, minimizing the integrand requires maximizing $\dot{E}$.
Therefore,

$$\boxed{V^*(E) = \arg\max_V \dot{E}(E,V).}$$

Equivalently,

$$\boxed{V^*(E) = \arg\max_V \frac{V}{m}[T(E,V) - D(E,V)].}$$

This is the mathematical basis for the statement that minimum climb time is obtained by maximizing specific excess power at each energy level.

Hamiltonian Derivation

Consider the reduced one-state problem

$$\dot{E} = P_s(E,V),$$

with objective

$$J = \int_0^{t_f} 1\,\mathrm{d}t.$$

The Hamiltonian is

$$\mathcal{H} = 1 + \lambda_E P_s(E, V).$$

For an interior optimum,

$$\frac{\partial \mathcal{H}}{\partial V} = 0.$$

Therefore,

$$\lambda_E \frac{\partial P_s}{\partial V} = 0.$$

For a nontrivial extremal,

$$\lambda_E \neq 0,$$

so

$$\boxed{\frac{\partial P_s}{\partial V} = 0.}$$

This confirms that the optimal speed at each energy level is a stationary point of specific excess power. To ensure it is a maximum,

$$\boxed{\frac{\partial^2 P_s}{\partial V^2} < 0.}$$

Costate Equation

The costate equation is

$$\dot{\lambda}_E = -\frac{\partial \mathcal{H}}{\partial E}.$$

Hence,

$$\boxed{\dot{\lambda}_E = -\lambda_E \frac{\partial P_s}{\partial E}.}$$

This linear differential equation has solution

$$\lambda_E(t) = \lambda_E(0) \exp\left[-\int_0^t \frac{\partial P_s}{\partial E}\, \mathrm{d}\tau\right].$$

Therefore, if $\lambda_E(0) \neq 0$, the costate cannot cross zero.

Free-Final-Time Condition

If t_f is free and there is no terminal cost, then

$$\mathcal{H}(t_f) = 0.$$

Thus,

$$1 + \lambda_E(t_f)P_s(E_f, V_f) = 0.$$

Therefore,

$$\boxed{\lambda_E(t_f) = -\frac{1}{P_s(E_f, V_f)}.}$$

For a feasible climb with positive excess power,

$$P_s > 0,$$

so

$$\lambda_E(t_f) < 0.$$

Since λ_E does not change sign,

$$\lambda_E(t) < 0$$

along the trajectory.
Then minimizing

$$\mathcal{H} = 1 + \lambda_E P_s$$

requires maximizing P_s.

Optimal Speed Schedule

The optimal energy-state speed schedule is determined by

$$\boxed{\frac{\partial}{\partial V}\left[V(T - D)\right] = 0.}$$

If mass is approximately constant,

$$0 = T - D + V\left(\frac{\partial T}{\partial V} - \frac{\partial D}{\partial V}\right).$$

Hence,

$$\boxed{T - D + V(T_V - D_V) = 0.}$$

This equation determines the speed giving maximum specific excess power at a prescribed energy.

Why the Optimal Trajectory May Dive

Near Mach 1, wave drag may increase sharply. A direct climb through this region can be slow because:

- thrust margin is low;
- drag is high;
- excess power is small;
- climbing consumes energy as potential energy.

A dive allows the aircraft to convert potential energy into kinetic energy:

$$\Delta\left(\frac{V^2}{2}\right) = -g\Delta h.$$

If $\Delta h < 0$, then

$$\Delta(V^2/2) > 0.$$

Thus the aircraft can accelerate through the high-drag transonic region more quickly by sacrificing altitude. After passing through the transonic region, the aircraft may regain altitude where excess power is more favorable.

Approximate Constant-Energy Dive

During an approximately constant-energy dive,

$$\dot{E} \approx 0.$$

Therefore,

$$T \approx D.$$

Also,

$$V\dot{V} + g\dot{h} \approx 0.$$

Hence,

$$\boxed{\dot{h} \approx -\frac{V}{g}\dot{V}.}$$

Integrating,

$$\boxed{h_2 - h_1 = -\frac{V_2^2 - V_1^2}{2g}.}$$

This relation quantifies the altitude sacrificed to gain speed.

Full Model versus Energy-State Approximation

The full longitudinal model includes:

$$\begin{aligned}
\dot{V} &= \frac{T\cos\alpha - D}{m} - g\sin\gamma, \\
\dot{\gamma} &= \frac{T\sin\alpha + L}{mV} - \frac{g\cos\gamma}{V}, \\
\dot{h} &= V\sin\gamma.
\end{aligned}$$

The energy-state approximation reduces the problem to

$$\dot{E} = \frac{V}{m}(T - D).$$

The approximate solution is useful as:

- an initial guess for a full indirect method;
- an initial guess for direct collocation;
- a means of explaining the qualitative trajectory;
- a reduced-order onboard guidance law;
- a method for generating performance envelopes.

Initial Guess Construction

A practical initial guess may be built in stages:

1. accelerate on the runway;
2. climb toward a transonic energy level;
3. dive approximately along a constant-energy contour;
4. perform a shallow energy-building climb;
5. complete a steep terminal climb.

The approximate path is then refined using a full optimal control solver.

Direct Collocation Formulation

Let the state be

$$\boldsymbol{x} = \begin{bmatrix} V & \gamma & h & x & m \end{bmatrix}^{\mathsf{T}},$$

and the control be

$$\boldsymbol{u} = \begin{bmatrix} \alpha & \eta \end{bmatrix}^{\mathsf{T}}.$$

The minimum-time problem is

$$\min_{\boldsymbol{x}(\cdot),\boldsymbol{u}(\cdot),t_f} t_f$$

subject to the full dynamics, endpoint constraints, and path constraints.
Using normalized time

$$\tau \in [0,1], \qquad t = t_0 + \tau(t_f - t_0),$$

the dynamics become

$$\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}\tau} = (t_f - t_0)\boldsymbol{f}(\boldsymbol{x},\boldsymbol{u}).$$

A collocation method enforces defects

$$\boldsymbol{\Delta}_k = 0$$

at all mesh intervals.

Indirect Formulation

For the full model, define

$$\mathcal{H} = 1 + \lambda_V \dot{V} + \lambda_\gamma \dot{\gamma} + \lambda_h \dot{h} + \lambda_x \dot{x} + \lambda_m \dot{m}.$$

The costates satisfy

$$\dot{\boldsymbol{\lambda}} = -\frac{\partial \mathcal{H}}{\partial \boldsymbol{x}}.$$

The interior controls satisfy

$$\frac{\partial \mathcal{H}}{\partial \alpha} = 0,$$
$$\frac{\partial \mathcal{H}}{\partial \eta} = 0.$$

The resulting two-point boundary-value problem is highly nonlinear. The energy-state solution is valuable because it supplies a physically meaningful initial guess.

Flight Envelope Constraints

The optimization must respect limits such as:

$$
\begin{aligned}
\alpha_{\min} \le \alpha \le \alpha_{\max},\\
0 \le \eta \le 1,\\
q \le q_{\max},\\
n \le n_{\max},\\
M \le M_{\max},\\
h \ge 0.
\end{aligned}
$$

Dynamic pressure is

$$
q = \frac{1}{2}\rho V^2.
$$

Load factor is approximately

$$
n = \frac{L}{mg}.
$$

These constraints may force the solution away from the unconstrained maximum-excess-power curve.

Numerical Procedure

A robust workflow is:

1. construct aerodynamic and propulsion lookup tables;
2. interpolate $T(V,h)$ and $D(V,h,\alpha)$;
3. compute energy contours;
4. compute the maximum-excess-power curve;
5. build an energy-state initial trajectory;
6. transcribe the full model using direct collocation;
7. solve the nonlinear program;
8. refine the mesh;
9. verify endpoint and path constraints;
10. compare with the reduced-order solution.

MATLAB Skeleton

```
function esa_demo()

    g = 9.81;
    m = 12000;

    Egrid = linspace(0.5e5,3.0e6,250);
    Vgrid = linspace(80,700,500);

    Vstar = nan(size(Egrid));
```

```
    Psmax = nan(size(Egrid));

    for i = 1:length(Egrid)

        E = Egrid(i);
        h = (E - 0.5*Vgrid.^2)/g;

        feasible = h >= 0;

        T = thrust_model(Vgrid,h);
        D = drag_model(Vgrid,h);

        Ps = Vgrid.*(T -D)/m;
        Ps(~feasible) = -Inf;

        [Psmax(i),idx] = max(Ps);
        Vstar(i) = Vgrid(idx);
    end

    figure;
    plot(Egrid,Vstar,'LineWidth',1.5);
    xlabel('Specific energy E');
    ylabel('Optimal speed V^*(E)');
    grid on;

    figure;
    plot(Egrid,Psmax,'LineWidth',1.5);
    xlabel('Specific energy E');
    ylabel('Maximum specific excess power');
    grid on;
end

function T = thrust_model(V,h)
    T0 = 90000;
    T = T0 .* exp( -h/18000) .* max(0.4,1 -2e -4*V);
end

function D = drag_model(V,h)
    rho = 1.225 .* exp( -h/8500);
    S = 50;
    CD0 = 0.02 + 0.04*exp( -((V -340)/45).^2);
    D = 0.5 .* rho .* V.^2 .* S .* CD0;
end
```

Interpretation of the Famous Dive–Climb Trajectory

The non-intuitive trajectory can be understood as follows:

1. At low speed, the aircraft accelerates and climbs.
2. Near transonic speed, drag increases rapidly.

3. A direct climb allocates energy to altitude when speed is urgently needed.
4. A dive converts potential energy into kinetic energy.
5. The aircraft crosses the transonic region faster.
6. At higher supersonic speed, the excess-power characteristics improve.
7. The aircraft then climbs to the final altitude and speed.

The optimal trajectory therefore minimizes time through energy management rather than altitude monotonicity.

Approximation Error

The energy-state approximation introduces error because:

- γ dynamics are neglected;
- lift equilibrium may not hold exactly;
- thrust may not align with velocity;
- mass may change;
- path constraints may become active;
- engine and aerodynamic models may be nonsmooth;
- rapid maneuvers violate quasi-steady assumptions.

The approximation should therefore be validated against the full model.

Verification Tests

A credible solution should satisfy:

1. terminal speed and altitude constraints;
2. all path constraints;
3. monotonic mesh convergence;
4. agreement between numerical and finite-difference derivatives;
5. Hamiltonian consistency for indirect solutions;
6. agreement between approximate and refined trajectory structure;
7. physical feasibility across the entire flight envelope.

Common Errors

- confusing total energy with kinetic energy;
- omitting the cancellation of gravity terms in $\dot{E}$;
- treating speed as an unconstrained algebraic variable;
- maximizing thrust rather than excess power;
- ignoring drag variation near Mach 1;
- assuming altitude must increase monotonically;

- using incomplete lookup tables without checking extrapolation;
- interpreting the reduced model as exact;
- failing to verify full-model dynamics;
- using an arbitrary initial guess for an indirect method.

Summary

1. Total specific energy is

$$E = \frac{V^2}{2} + gh.$$

2. Its rate of change is

$$\dot{E} = \frac{V}{m}(T - D).$$

3. Minimum climb time can be written as

$$t_f = \int_{E_0}^{E_f} \frac{1}{\dot{E}} \, \mathrm{d}E.$$

4. Therefore, the energy-state approximation chooses

$$V^*(E) = \arg\max_V \dot{E}(E, V).$$

5. Constant-energy curves describe exchange between speed and altitude.
6. A dive may be optimal because it accelerates the aircraft through a high-drag region.
7. The reduced solution provides an excellent initial guess for the full optimal control problem.
8. The method remains one of the clearest examples of physics-informed trajectory optimization.

Exercises

1. **Verification of a computed optimal trajectory**

   Suppose a direct-collocation solver returns the discrete solution

   $$\{\boldsymbol{X}_k, \boldsymbol{U}_k\}_{k=0}^{K}$$

   for

   $$\min_{\boldsymbol{u}(\cdot)} \; J = \Phi\big(\boldsymbol{x}(t_f)\big) + \int_{t_0}^{t_f} L(\boldsymbol{x}, \boldsymbol{u}, t)\, \mathrm{d}t$$

   subject to

   $$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, t).$$

   (a) Define a dynamic-defect residual on each mesh interval.

   (b) Define the maximum endpoint-constraint residual.

   (c) Define the maximum path-constraint violation.

   (d) Explain how to re-simulate the reconstructed control using an independent differential-equation solver.

   (e) Define a mesh-refinement table containing the objective, terminal residual, maximum defect, and maximum path violation.

   (f) State what convergence behavior would support the claim that the numerical result is credible.

   (g) Explain why a solver-success message alone is not evidence that the engineering conclusion is correct.

2. **Complete derivation of the bilinear tangent steering law**

   Consider

   $$\dot{x} = u, \qquad \dot{y} = v, \qquad \dot{u} = a\cos\beta, \qquad \dot{v} = a\sin\beta,$$

   with

   $$x(0) = y(0) = u(0) = v(0) = 0,$$

   $$y(T) = H, \qquad v(T) = 0,$$

   where T is fixed. The terminal values $x(T)$ and $u(T)$ are free, and the objective is

   $$\max_{\beta(\cdot)} u(T).$$

   (a) Convert the problem to minimization form.

   (b) Construct the Hamiltonian.

   (c) Derive and integrate all four costate equations.

   (d) Derive the interior stationarity condition

$$\tan\beta = \frac{\lambda_v}{\lambda_u}.$$

(e) Apply terminal transversality to show that

$$\lambda_x(t) \equiv 0, \qquad \lambda_u(t) \equiv -1.$$

(f) Show that the general bilinear tangent law reduces to

$$\tan\beta(t) = \tan\beta_0 - ct.$$

(g) Explain why the correct branch of β must still be checked against the minimum principle.

3. **Analytical state reconstruction for fixed-time ascent**

Let

$$q(t) = \tan\beta(t) = q_0 - ct.$$

(a) Show that

$$\cos\beta = \frac{1}{\sqrt{1+q^2}}, \qquad \sin\beta = \frac{q}{\sqrt{1+q^2}}, \qquad \mathrm{d}t = -\frac{1}{c}\,\mathrm{d}q.$$

(b) Starting from

$$\dot{u} = a\cos\beta,$$

derive

$$u(t) = \frac{a}{c}\ln\left[\frac{q_0 + \sqrt{1+q_0^2}}{q(t) + \sqrt{1+q^2(t)}}\right].$$

(c) Starting from

$$\dot{v} = a\sin\beta,$$

derive

$$v(t) = \frac{a}{c}\left[\sqrt{1+q_0^2} - \sqrt{1+q^2(t)}\right].$$

(d) Use $v(T) = 0$ to show that the nontrivial solution satisfies

$$\beta(T) = -\beta_0.$$

(e) Derive

$$c = \frac{2\tan\beta_0}{T}.$$

(f) Show that

$$\tan\beta(t) = \tan\beta_0\left(1 - \frac{2t}{T}\right).$$

(g) Explain the symmetry of the steering law about $t = T/2$.

4. **Altitude equation and terminal horizontal velocity**

For the fixed-time ascent problem, the analytical vertical position is

$$y(t) = \frac{a}{2c^2}\left[\left(\tan\beta_0 - \tan\beta\right)\left(\sec\beta_0 + \sec\beta\right) - \ln\left(\frac{\tan\beta_0 + \sec\beta_0}{\tan\beta + \sec\beta}\right)\right].$$

(a) Substitute

$$\beta(T) = -\beta_0, \qquad c = \frac{2\tan\beta_0}{T}$$

and derive the scalar altitude equation

$$\frac{4H}{aT^2} = \frac{2\tan\beta_0\sec\beta_0 - \ln\left(\frac{\sec\beta_0+\tan\beta_0}{\sec\beta_0-\tan\beta_0}\right)}{2\tan^2\beta_0}.$$

(b) Explain why the original trajectory-optimization problem has now been reduced to one scalar nonlinear equation.

(c) Derive the expression for the terminal horizontal velocity $u(T)$.

(d) Describe a numerical procedure for solving for β_0.

(e) State the endpoint and minimum-principle checks that must be performed after a root is obtained.

5. **Necessary conditions for fuel-optimal lunar landing**

Consider

$$\dot{h} = v, \qquad \dot{v} = -g + u,$$

with

$$0 \le u \le 3,$$

$$h(0) = h_0, \qquad v(0) = v_0,$$

$$h(t_f) = 0, \qquad v(t_f) = 0,$$

and free t_f. Minimize

$$J = \int_0^{t_f} u(t)\,\mathrm{d}t.$$

Use the chapter's normal Hamiltonian convention

$$H = 1 + \lambda_h v + \lambda_v(-g + u).$$

(a) Derive the costate equations and their general solutions.

(b) Define the switching function

$$\sigma(t) = 1 + \lambda_v(t).$$

(c) Derive the bang–bang control law.

(d) Prove that a nondegenerate singular arc cannot exist.

(e) Show that the optimal control has at most one switching time.

(f) Derive the free-final-time condition

$$H(t_f) = 0.$$

6. **Piecewise analytical lunar-landing trajectory**

   For the lunar-landing model, first derive the state solution on an interval $[t_a, t]$ when the initial values at t_a are h_a and v_a.

   (a) For $u = 0$, derive $v(t)$ and $h(t)$.

   (b) For $u = 3$, derive $v(t)$ and $h(t)$.

   (c) Assume a coast-then-burn structure,

$$u(t) = \begin{cases} 0, & 0 \le t < t_s, \\ 3, & t_s \le t \le t_f. \end{cases}$$

   Derive $h(t_s)$ and $v(t_s)$.

   (d) Enforce continuity and derive the two algebraic equations produced by

$$h(t_f) = 0, \qquad v(t_f) = 0.$$

   (e) Express the fuel cost in terms of t_s and t_f.

   (f) Repeat the formulation for a burn-then-coast structure.

   (g) State how the admissible structures should be compared and verified.

7. **Direct and indirect verification of lunar landing**

   Formulate a numerical verification study for the lunar-landing problem.

   (a) Write a direct-shooting formulation using t_s and t_f as decision variables for a coast-then-burn control.

   (b) Write a direct-collocation formulation with state variables h and v, control u, and free final time.

(c) State the boundary and control constraints.

(d) Define the terminal residuals.

(e) Define the Hamiltonian residual at the final time.

(f) Explain how to compare the shooting and collocation solutions.

(g) Explain why a smooth numerical approximation of the switch does not by itself disprove the bang–bang analytical structure.

8. **Total specific energy and constant-energy contours**

Consider the longitudinal aircraft equations

$$\dot{V} = \frac{T\cos\alpha - D}{m} - g\sin\gamma, \qquad \dot{h} = V\sin\gamma.$$

Define

$$E = \frac{V^2}{2} + gh.$$

(a) Differentiate E with respect to time.

(b) Show explicitly that the gravitational terms cancel.

(c) Derive

$$\dot{E} = \frac{V}{m}\left(T\cos\alpha - D\right).$$

(d) Under the small-angle approximation, derive

$$\dot{E} \approx \frac{V}{m}(T - D).$$

(e) For constant E, derive

$$h = \frac{E}{g} - \frac{V^2}{2g}.$$

(f) Derive

$$\frac{\mathrm{d}h}{\mathrm{d}V} = -\frac{V}{g}.$$

(g) Derive the altitude loss required to accelerate from V_1 to V_2 at constant energy.

9. **Energy-state minimum-time optimality conditions**

Consider the reduced model

$$\dot{E} = P_s(E,V) = \frac{V}{m}\left[T(E,V) - D(E,V)\right],$$

with

$$\min t_f = \min \int_0^{t_f} 1 \, \mathrm{d}t.$$

(a) Rewrite the final time as

$$t_f = \int_{E_0}^{E_f} \frac{1}{P_s(E,V)} \, \mathrm{d}E.$$

(b) Explain directly from this integral why minimum time requires maximum specific excess power at each energy level.

(c) Construct the reduced Hamiltonian

$$H = 1 + \lambda_E P_s(E,V).$$

(d) Derive the stationarity condition

$$\frac{\partial P_s}{\partial V} = 0.$$

(e) Derive the costate equation and show that λ_E cannot change sign along a normal extremal.

(f) Use the free-final-time condition to show that $\lambda_E < 0$ for a feasible climb with $P_s > 0$.

(g) Show that the speed schedule satisfies

$$T - D + V\,(T_V - D_V) = 0.$$

(h) State the second-derivative condition required for a local maximum of specific excess power.

10. **Full-model refinement of the energy-state solution**

Let the full aircraft state and control be

$$\boldsymbol{x} = \begin{bmatrix} V & \gamma & h & x & m \end{bmatrix}^{\mathsf{T}}, \qquad \boldsymbol{u} = \begin{bmatrix} \alpha & \eta \end{bmatrix}^{\mathsf{T}}.$$

The objective is to minimize t_f.

(a) Write the full longitudinal dynamics used in the chapter.

(b) Write the minimum-time Hamiltonian for an indirect formulation.

(c) State the interior stationarity equations with respect to α and η.

(d) Write the normalized-time dynamics on $\tau \in [0,1]$.

(e) List the flight-envelope constraints discussed in the chapter.

(f) Explain how an energy-state speed schedule can be converted into an initial guess for the full problem.

(g) Define verification checks for terminal feasibility, path feasibility, mesh convergence, Hamiltonian consistency, and independent re-simulation.

(h) Explain why agreement in qualitative dive–climb structure is useful but does not prove quantitative accuracy of the reduced model.

Chapter 9: Multiple-Interval Pseudospectral Methods and Adaptive Meshes

Chapter 9: Multiple-Interval Pseudospectral Methods and Adaptive Meshes

Local spectral accuracy, sparse global assembly, and adaptive hp refinement

Global pseudospectral approximations are exceptionally accurate for smooth solutions but can struggle near switches, boundary layers, or localized high curvature. The final chapter resolves that limitation by dividing the horizon into polynomial mesh intervals. It develops the local method, assembles the global sparse transcription, and then uses error estimates to decide where intervals should be split or polynomial degree increased.

Chapter contents

Multiple-Interval LGR Pseudospectral Methods

Single-interval pseudospectral methods approximate the state and control over the entire time horizon using a global polynomial. These methods are highly accurate when the solution is smooth. However, many realistic optimal control problems contain:

- bang–bang controls;
- state or control discontinuities;
- boundary layers;
- sharp changes in curvature;
- active-set transitions;
- nonsmooth dynamics;
- path-constraint junctions.

A single global polynomial is often inadequate for such problems. The practical remedy is to divide the horizon into multiple mesh intervals and perform local collocation in each interval.

This section develops the multiple-interval Legendre–Gauss–Radau (LGR) pseudospectral method in detail. The main goals are to:

1. review LG, LGR, and LGL collocation structures;
2. explain why global polynomial approximations may fail;
3. derive the multiple-interval LGR discretization;
4. obtain the quadrature approximation of the objective;
5. derive the local collocation equations;
6. impose state continuity between intervals;
7. describe the block-sparse nonlinear programming structure;
8. explain mesh refinement and practical implementation.

Review of Gaussian Collocation Points

Three common sets of orthogonal collocation points are:

1. Legendre–Gauss (LG);
2. Legendre–Gauss–Radau (LGR);
3. Legendre–Gauss–Lobatto (LGL).

All are commonly defined on the reference domain

$$\tau \in [-1, 1].$$

Legendre–Gauss Points

LG points lie strictly inside the interval:

$$-1 < \tau_1 < \tau_2 < \cdots < \tau_N < 1.$$

Neither endpoint is included.

For an LG transcription, the state approximation typically includes the initial point, the interior LG nodes, and a separately reconstructed terminal state. The controls are defined only at the LG collocation points.

Legendre–Gauss–Radau Points

Forward LGR points include the left endpoint but exclude the right endpoint:

$$\tau_1 = -1, \qquad \tau_N < 1.$$

A terminal state value at

$$\tau_{N+1} = 1$$

is added to the state approximation, but it is not a collocation point.

Thus, an LGR discretization uses:

- N collocation points;
- $N + 1$ state approximation points;
- N control approximation points.

Legendre–Gauss–Lobatto Points

LGL points include both endpoints:

$$\tau_1 = -1, \qquad \tau_N = 1.$$

The state and control are typically represented at all LGL nodes.

Why Multiple Intervals Are Necessary

Global Polynomial Approximation

Suppose the entire time interval

$$t \in [t_0, t_f]$$

is mapped to

$$\tau \in [-1, 1].$$

A global polynomial approximation is then used:

$$x(\tau) \approx X(\tau) = \sum_{j=1}^{N+1} X_j \ell_j(\tau),$$

where ℓ_j are Lagrange interpolation polynomials.

This approach is spectrally accurate for analytic solutions. The error can decrease extremely rapidly with increasing polynomial degree.

Failure Near Discontinuities

If the exact control has a jump, such as

$$u(t) = \begin{cases} u_{\min}, & t < t_s, \\ u_{\max}, & t > t_s, \end{cases}$$

then a global polynomial remains smooth and cannot represent the discontinuity exactly.

Increasing the polynomial degree produces oscillations near the jump. This is the Gibbs phenomenon.

> **Gibbs Phenomenon**
> Near a discontinuity, increasing polynomial degree narrows the oscillatory region but does not eliminate the characteristic overshoot. The error remains concentrated near the jump.

This behavior is analogous to approximating a square wave using a truncated Fourier series.

Local Polynomial Approximation

The remedy is to divide the domain into intervals:

$$[t_0, t_f] = [t_0, t_1] \cup [t_1, t_2] \cup \cdots \cup [t_{K-1}, t_K],$$

where

$$t_K = t_f.$$

A separate polynomial is used in each interval. Then:

- discontinuities can be aligned with interval boundaries;
- smooth portions can use high-order polynomials;
- difficult regions can use smaller intervals;
- sparsity is preserved.

Comparison of LG, LGR, and LGL on Multiple Intervals

Consider two adjacent intervals:

$$[t_0, t_1], \qquad [t_1, t_f].$$

The state must satisfy continuity:

$$\boldsymbol{x}(t_1^-) = \boldsymbol{x}(t_1^+).$$

The control need not be continuous unless the physics or problem formulation requires it.

LG Structure

LG points exclude both interval endpoints.
Consequences:

- neither the end of interval 1 nor the beginning of interval 2 is a control collocation point;
- the interface state is reconstructed;
- no control is directly available at the interface.

LGR Structure

Forward LGR includes the left endpoint of each interval but excludes the right endpoint.
At the interface:

- the terminal state of interval k equals the initial state of interval $k + 1$;
- the control is defined at the start of interval $k + 1$;
- only one control value is associated with the interface;
- the control is not artificially double-valued.

This is one reason LGR is attractive for multiple-interval transcription.

LGL Structure

LGL includes both endpoints of every interval.
At a shared interface, there are two control values:

$$u(t_k^-) \qquad \text{and} \qquad u(t_k^+).$$

These values may differ if the optimal control is discontinuous. If the user incorrectly enforces equality, a physically valid control jump may be suppressed.

Original Optimal Control Problem

Consider the Bolza problem

$$\min_{\boldsymbol{x}(\cdot),\boldsymbol{u}(\cdot),t_0,t_f,\boldsymbol{p}} J = \Phi\left(\boldsymbol{x}(t_0), t_0, \boldsymbol{x}(t_f), t_f, \boldsymbol{p}\right) + \int_{t_0}^{t_f} L\left(\boldsymbol{x}(t), \boldsymbol{u}(t), t, \boldsymbol{p}\right)\, \mathrm{d}t.$$

The dynamics are

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}\left(\boldsymbol{x}(t), \boldsymbol{u}(t), t, \boldsymbol{p}\right).$$

Endpoint constraints are

$$\boldsymbol{\phi}\left(\boldsymbol{x}(t_0), t_0, \boldsymbol{x}(t_f), t_f, \boldsymbol{p}\right) = \boldsymbol{0}.$$

Path constraints are

$$\boldsymbol{c}_{\min} \leq \boldsymbol{c}\left(\boldsymbol{x}(t), \boldsymbol{u}(t), t, \boldsymbol{p}\right) \leq \boldsymbol{c}_{\max}.$$

Global Time Transformation

Map physical time $t \in [t_0, t_f]$ to

$$s \in [-1, 1]$$

using

$$\boxed{s = \frac{2(t - t_0)}{t_f - t_0} - 1.}$$

The inverse mapping is

$$\boxed{t = \frac{t_f - t_0}{2} s + \frac{t_f + t_0}{2}.}$$

The derivative relationships are

$$\frac{\mathrm{d}s}{\mathrm{d}t} = \frac{2}{t_f - t_0},$$
$$\frac{\mathrm{d}t}{\mathrm{d}s} = \frac{t_f - t_0}{2}.$$

Thus,

$$\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}s} = \frac{t_f - t_0}{2} \boldsymbol{f}\left(\boldsymbol{x}, \boldsymbol{u}, t, \boldsymbol{p}\right).$$

The cost becomes

$$J = \Phi\left(\boldsymbol{x}(-1), t_0, \boldsymbol{x}(1), t_f, \boldsymbol{p}\right) + \frac{t_f - t_0}{2} \int_{-1}^{1} L\left(\boldsymbol{x}(s), \boldsymbol{u}(s), t(s), \boldsymbol{p}\right) \mathrm{d}s.$$

Mesh Definition

Divide the transformed domain into K mesh intervals:

$$-1 = s_0 < s_1 < \cdots < s_{K-1} < s_K = 1.$$

The kth interval is

$$\mathcal{S}_k = [s_{k-1}, s_k].$$

The quantities

$$s_0, s_1, \ldots, s_K$$

are called mesh points.
For a fixed mesh, they are constants during the nonlinear programming solve.
The mesh width is

$$\Delta s_k = s_k - s_{k-1}.$$

A uniform mesh satisfies

$$\Delta s_k = \frac{2}{K}.$$

A nonuniform mesh allows smaller intervals in difficult regions.

Local Coordinate Mapping

Standard LGR points are defined on

$$\tau \in [-1, 1].$$

For interval $\mathcal{S}_k = [s_{k-1}, s_k]$, map τ to s using

$$\boxed{s = \frac{s_k - s_{k-1}}{2}\tau + \frac{s_k + s_{k-1}}{2}.}$$

Equivalently,

$$\boxed{\tau = \frac{2s - (s_k + s_{k-1})}{s_k - s_{k-1}}.}$$

The derivative scaling is

$$\frac{\mathrm{d}s}{\mathrm{d}\tau} = \frac{\Delta s_k}{2}, \qquad \frac{\mathrm{d}\tau}{\mathrm{d}s} = \frac{2}{\Delta s_k}.$$

If the standard LGR nodes are

$$\tau_1^{(k)}, \ldots, \tau_{N_k}^{(k)},$$

then the nodes in interval k are

$$\boxed{s_i^{(k)} = \frac{\Delta s_k}{2}\tau_i^{(k)} + \frac{s_k + s_{k-1}}{2}, \qquad i = 1, \ldots, N_k.}$$

The noncollocated terminal state point is

$$s_{N_k+1}^{(k)} = s_k.$$

Local State Approximation

In interval k, approximate the state using

$$\boldsymbol{x}^{(k)}(s) \approx \boldsymbol{X}^{(k)}(s) = \sum_{j=1}^{N_k+1} \boldsymbol{X}_j^{(k)} \ell_j^{(k)}(s),$$

where

$$\boldsymbol{X}_j^{(k)} \approx \boldsymbol{x}\left(s_j^{(k)}\right).$$

The Lagrange basis polynomial is

$$\ell_j^{(k)}(s) = \prod_{\substack{m=1 \\ m \neq j}}^{N_k+1} \frac{s - s_m^{(k)}}{s_j^{(k)} - s_m^{(k)}}.$$

It satisfies the cardinal property

$$\ell_j^{(k)}\left(s_i^{(k)}\right) = \delta_{ij}.$$

Control Approximation

The control is represented only at the N_k LGR collocation points:

$$\boldsymbol{U}_i^{(k)} \approx \boldsymbol{u}\left(s_i^{(k)}\right), \qquad i = 1, \ldots, N_k.$$

A polynomial control interpolant may be formed if desired:

$$\boldsymbol{u}^{(k)}(s) \approx \sum_{j=1}^{N_k} \boldsymbol{U}_j^{(k)} \hat{\ell}_j^{(k)}(s).$$

However, in the NLP, only the nodal control values are usually required.

Local Differentiation Matrix

Differentiate the state approximation:

$$\frac{\mathrm{d}\boldsymbol{X}^{(k)}}{\mathrm{d}s} = \sum_{j=1}^{N_k+1} \boldsymbol{X}_j^{(k)} \frac{\mathrm{d}\ell_j^{(k)}}{\mathrm{d}s}.$$

Evaluate at the collocation points:

$$\left.\frac{\mathrm{d}\boldsymbol{X}^{(k)}}{\mathrm{d}s}\right|_{s=s_i^{(k)}} = \sum_{j=1}^{N_k+1} D_{ij}^{(k)} \boldsymbol{X}_j^{(k)},$$

where

$$\boxed{D_{ij}^{(k)} = \left.\frac{\mathrm{d}\ell_j^{(k)}}{\mathrm{d}s}\right|_{s=s_i^{(k)}}}.$$

The local differentiation matrix has dimensions

$$\boldsymbol{D}^{(k)} \in \mathbb{R}^{N_k \times (N_k+1)}.$$

Scaling from the Standard LGR Matrix

Let

$$\hat{\boldsymbol{D}}^{(k)}$$

be the LGR differentiation matrix on $\tau \in [-1, 1]$.
Because

$$\frac{\mathrm{d}}{\mathrm{d}s} = \frac{2}{\Delta s_k} \frac{\mathrm{d}}{\mathrm{d}\tau},$$

the interval differentiation matrix is

$$\boxed{\boldsymbol{D}^{(k)} = \frac{2}{\Delta s_k} \hat{\boldsymbol{D}}^{(k)}}.$$

Collocation Equations

The transformed dynamics are

$$\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}s} = \frac{t_f - t_0}{2}\boldsymbol{f}\left(\boldsymbol{x}, \boldsymbol{u}, t, \boldsymbol{p}\right).$$

At node i of interval k:

$$\sum_{j=1}^{N_k+1} D_{ij}^{(k)} \boldsymbol{X}_j^{(k)} = \frac{t_f - t_0}{2}\boldsymbol{f}\left(\boldsymbol{X}_i^{(k)}, \boldsymbol{U}_i^{(k)}, t_i^{(k)}, \boldsymbol{p}\right).$$

Thus,

$$\boxed{\sum_{j=1}^{N_k+1} D_{ij}^{(k)} \boldsymbol{X}_j^{(k)} - \frac{t_f - t_0}{2}\boldsymbol{F}_i^{(k)} = \boldsymbol{0}.}$$

These equations hold for

$$i = 1, \ldots, N_k, \qquad k = 1, \ldots, K.$$

Physical Time at Each Node

The global transformed coordinate $s_i^{(k)}$ maps to physical time through

$$\boxed{t_i^{(k)} = \frac{t_f - t_0}{2} s_i^{(k)} + \frac{t_f + t_0}{2}.}$$

If t_0 or t_f is free, these nodal times depend on the NLP decision variables.

Quadrature over Multiple Intervals

For an arbitrary integrand $G(s)$:

$$\int_{-1}^{1} G(s)\,\mathrm{d}s = \sum_{k=1}^{K} \int_{s_{k-1}}^{s_k} G(s)\,\mathrm{d}s.$$

In interval k, LGR quadrature gives

$$\int_{s_{k-1}}^{s_k} G(s)\,\mathrm{d}s \approx \sum_{i=1}^{N_k} w_i^{(k)} G\left(s_i^{(k)}\right).$$

The interval weights scale from the standard LGR weights:

$$\boxed{w_i^{(k)} = \frac{\Delta s_k}{2}\hat{w}_i^{(k)}.}$$

Therefore,

$$\boxed{\int_{-1}^{1} G(s)\,\mathrm{d}s \approx \sum_{k=1}^{K}\sum_{i=1}^{N_k} w_i^{(k)} G_i^{(k)}.}$$

Discrete Cost Functional

The objective becomes

$$\begin{aligned} J \approx J_h = \Phi\left(\boldsymbol{X}_1^{(1)}, t_0, \boldsymbol{X}_{N_K+1}^{(K)}, t_f, \boldsymbol{p}\right) \\ + \frac{t_f - t_0}{2} \sum_{k=1}^{K}\sum_{i=1}^{N_k} w_i^{(k)} L\left(\boldsymbol{X}_i^{(k)}, \boldsymbol{U}_i^{(k)}, t_i^{(k)}, \boldsymbol{p}\right). \end{aligned}$$

Hence,

$$\boxed{J_h = \Phi_h + \frac{t_f - t_0}{2} \sum_{k=1}^{K}\sum_{i=1}^{N_k} w_i^{(k)} L_i^{(k)}.}$$

State Continuity between Mesh Intervals

For a continuous state trajectory:

$$\boxed{\boldsymbol{X}_{N_k+1}^{(k)} = \boldsymbol{X}_1^{(k+1)}, \qquad k = 1, \ldots, K-1.}$$

There are two implementation choices.

Explicit Continuity Constraints

Keep both interface state variables and impose

$$\boldsymbol{X}_{N_k+1}^{(k)} - \boldsymbol{X}_1^{(k+1)} = \boldsymbol{0}.$$

Shared State Variables

Use the same NLP variable for both sides:

$$\boldsymbol{X}_{N_k+1}^{(k)} \equiv \boldsymbol{X}_1^{(k+1)}.$$

This eliminates explicit continuity constraints and reduces the number of decision variables.

Control Behavior at Interfaces

Forward LGR has no control at the terminal point of interval k, but it does have a control at the initial point of interval $k+1$.

Thus, there is one interface control value:

$$\boldsymbol{U}_1^{(k+1)}.$$

This is especially useful when the control is discontinuous. The method does not force left and right control limits to be equal.

Endpoint Constraints

The discrete endpoint constraints are

$$\boxed{\boldsymbol{\phi}\left(\boldsymbol{X}_1^{(1)}, t_0, \boldsymbol{X}_{N_K+1}^{(K)}, t_f, \boldsymbol{p}\right) = \boldsymbol{0}.}$$

The initial state comes from the first node of the first interval. The final state comes from the additional terminal state point of the last interval.

Path Constraints

At every collocation point:

$$\boxed{\boldsymbol{c}_{\min} \le \boldsymbol{c}\left(\boldsymbol{X}_i^{(k)}, \boldsymbol{U}_i^{(k)}, t_i^{(k)}, \boldsymbol{p}\right) \le \boldsymbol{c}_{\max}.}$$

These constraints are imposed for

$$i = 1, \ldots, N_k, \qquad k = 1, \ldots, K.$$

Because constraints are enforced only at collocation nodes, the continuous-time solution should be checked between nodes after solving.

Discrete Nonlinear Programming Problem

The NLP decision vector contains:

- state values in every mesh interval;
- control values at every LGR node;
- free initial and final times;
- static parameters.

A generic decision vector is

$$\boldsymbol{Z} = \left[\mathrm{vec}(\boldsymbol{X}), \mathrm{vec}(\boldsymbol{U}), t_0, t_f, \boldsymbol{p}\right]^{\mathsf{T}}.$$

The NLP is

$$\min_{\boldsymbol{Z}} J_h(\boldsymbol{Z})$$

subject to

$$\begin{aligned} \boldsymbol{\Delta}^{(k)}(\boldsymbol{Z}) &= \mathbf{0}, \qquad && k = 1, \ldots, K, \\ \boldsymbol{\phi}_h(\boldsymbol{Z}) &= \mathbf{0}, \\ \boldsymbol{c}_{\min} \le \boldsymbol{c}_h(\boldsymbol{Z}) &\le \boldsymbol{c}_{\max}, \\ \boldsymbol{Z}_{\min} \le \boldsymbol{Z} &\le \boldsymbol{Z}_{\max}. \end{aligned}$$

Global Matrix Structure

Let

$$N_{\text{tot}} = \sum_{k=1}^{K} N_k.$$

The global differentiation matrix is block diagonal:

$$\boxed{\boldsymbol{D}_{\text{global}} = \text{blkdiag}\left(\boldsymbol{D}^{(1)}, \boldsymbol{D}^{(2)}, \ldots, \boldsymbol{D}^{(K)}\right).}$$

Unlike a single global pseudospectral differentiation matrix, this matrix is sparse.
The global collocation equations can be written schematically as

$$\boldsymbol{D}_{\text{global}}\boldsymbol{X} - \frac{t_f - t_0}{2}\boldsymbol{F}(\boldsymbol{X}, \boldsymbol{U}) = \mathbf{0}.$$

Sparsity Pattern

Each local collocation defect depends only on:

- state variables in the same interval;
- controls in the same interval;
- global time variables;
- static parameters.

Therefore, the Jacobian has a block-banded structure.
This sparsity is one of the principal computational advantages of multiple-interval methods.

Number of Decision Variables

Suppose:

- n_x states;
- n_u controls;
- n_p static parameters;
- N_k LGR points in interval k.

Without shared interface states, the state decision variables are

$$n_x \sum_{k=1}^{K} (N_k + 1).$$

The controls contribute

$$n_u \sum_{k=1}^{K} N_k.$$

With shared interface states, the number of unique state nodes is

$$1 + \sum_{k=1}^{K} N_k.$$

Thus,

$$\boxed{n_{\text{state}} = n_x \left(1 + \sum_{k=1}^{K} N_k \right).}$$

Defect Constraints

Define the defect at node i of interval k:

$$\boldsymbol{\Delta}_i^{(k)} = \sum_{j=1}^{N_k+1} D_{ij}^{(k)} \boldsymbol{X}_j^{(k)} - \frac{t_f - t_0}{2} \boldsymbol{F}_i^{(k)}.$$

The collocation conditions are

$$\boldsymbol{\Delta}_i^{(k)} = \mathbf{0}.$$

For all intervals, the number of defect equations is

$$n_x \sum_{k=1}^{K} N_k.$$

Integral Consistency Check

LGR quadrature also gives an integral form of the state update:

$$\boldsymbol{X}_{N_k+1}^{(k)} - \boldsymbol{X}_1^{(k)} \approx \frac{t_f - t_0}{2} \sum_{i=1}^{N_k} w_i^{(k)} \boldsymbol{F}_i^{(k)}.$$

This relation can be used as a consistency check.

Mesh Refinement

The mesh should adapt to the solution.
Two basic refinement operations are:

h-Refinement

Divide an interval into smaller intervals.
This is effective near:

- discontinuities;
- sharp control switches;
- path-constraint junctions;
- localized high-curvature regions.

p-Refinement

Increase polynomial degree N_k within an interval.
This is effective when the solution is smooth and the approximation error is distributed across the interval.

hp-Refinement

Combine interval subdivision and degree elevation.
Modern pseudospectral optimal control software often uses adaptive hp refinement.

Error Estimation

A common error estimator compares:

1. the polynomial state approximation;
2. a state reconstructed by integrating the dynamics using the approximate control.

Let

$$\boldsymbol{X}^{(k)}_{\text{poly}}(s)$$

be the collocation polynomial and

$$\boldsymbol{X}^{(k)}_{\text{int}}(s)$$

be an integrated state reconstruction.
Define

$$e_k = \max_{s \in \mathcal{S}_k} \left\| \boldsymbol{X}^{(k)}_{\text{poly}}(s) - \boldsymbol{X}^{(k)}_{\text{int}}(s) \right\|_\infty .$$

If

$$e_k > \epsilon,$$

the interval is refined.

Detecting Nonsmoothness

A smooth solution is typically characterized by rapidly decaying polynomial coefficients.
Suppose the local state approximation is written in an orthogonal basis:

$$X^{(k)}(\tau) = \sum_{j=0}^{N_k} a_j^{(k)} P_j(\tau).$$

If

$$|a_j^{(k)}|$$

decreases rapidly with j, p-refinement is appropriate.
If the coefficients decay slowly or stagnate, the interval may contain a nonsmooth feature and should be subdivided.

Switch Detection

For bang–bang problems, switch locations can be inferred from:

- rapid changes in the computed control;
- large defect errors;
- active-bound transitions;
- switching-function sign changes;
- concentration of polynomial oscillation.

Placing a mesh boundary near the switch dramatically improves accuracy.

Algorithmic Workflow

Algorithm: Multiple-Interval LGR Pseudospectral Method

```
Choose an initial mesh \{s_k\}_{k=0}^{K}.
Choose polynomial degrees N_k.
Compute standard LGR nodes and weights.
Map nodes and weights into each mesh interval.
Form local differentiation matrices.
Construct the NLP decision vector.
Impose collocation defects.
Impose state continuity.
Impose endpoint and path constraints.
Solve the NLP.
```

```
Estimate interval -wise discretization error.
if all interval errors are below tolerance:
    Accept the solution.
else:
    Apply h -, p -, or hp -refinement.
    Interpolate the previous solution onto the new mesh.
    Resolve the NLP.
end if
```

Practical Implementation Details

Initial Guess

The initial guess may be generated using:

- linear state interpolation;
- constant control;
- analytical reduced-order solutions;
- simulation-based trajectories;
- a previously solved nearby case;
- continuation in parameters or constraints.

Scaling

States, controls, constraints, and time should be scaled so that typical values are near unity.
For example:

$$\bar{x}_i = \frac{x_i - x_{i,\mathrm{ref}}}{s_{x_i}}.$$

Poor scaling can cause:

- inaccurate derivatives;
- slow convergence;
- misleading feasibility tolerances;
- failure of the NLP solver.

Derivative Computation

Derivatives may be obtained using:

- analytic formulas;
- automatic differentiation;
- complex-step differentiation;
- finite differences.

Automatic differentiation is usually preferred for large sparse problems.

MATLAB-Like Assembly Skeleton

```
function data = buildMultipleIntervalLGR(mesh, degree)

K = length(mesh) -1;

Dblocks = cell(K,1);
nodes   = cell(K,1);
weights = cell(K,1);

for k = 1:K

    Nk = degree(k);

    % Standard forward LGR data on [ -1,1]
    [tau, wHat, DHat] = lgrData(Nk);

    sL = mesh(k);
    sR = mesh(k+1);
    ds = sR -sL;

    % Map collocation nodes to current mesh interval
    s = 0.5*ds*tau + 0.5*(sR+sL);

    % Scale quadrature weights and differentiation matrix
    w = 0.5*ds*wHat;
    D = (2/ds)*DHat;

    nodes{k}   = s;
    weights{k} = w;
    Dblocks{k} = D;
end

data.nodes   = nodes;
data.weights = weights;
data.D       = blkdiag(Dblocks{:});

end
```

Defect Assembly Example

```
for k = 1:K

    Nk = degree(k);

    Xk = X{k};        % (Nk+1) -by -nx
    Uk = U{k};        % Nk -by -nu
    tk = timeMap(nodes{k}, t0, tf);

    Fk = dynamics(Xk(1:Nk,:), Uk, tk, p);
```

```
    defect{k} = D{k}*Xk ...
              - 0.5*(tf -t0)*Fk;
end
```

Advantages of Multiple-Interval LGR

1. Handles nonsmooth solutions more effectively than a global polynomial.
2. Preserves high-order accuracy within smooth intervals.
3. Produces sparse Jacobian and Hessian structures.
4. Avoids double-valued interface controls.
5. Supports adaptive mesh refinement.
6. Provides a natural framework for large-scale trajectory optimization.
7. Works well with sparse nonlinear programming solvers.

Limitations

1. Requires mesh-selection and error-estimation logic.
2. Can generate large NLPs.
3. Path constraints are enforced only at nodes unless additional checks are used.
4. Poorly selected meshes may miss short-duration events.
5. Very low-order intervals may reduce accuracy.
6. Very high-order intervals may become ill-conditioned.
7. Discontinuous states require multiphase formulations rather than simple continuity.

Multiple Phases versus Multiple Mesh Intervals

Multiple mesh intervals are used when the dynamics and state remain continuous.
Multiple phases are used when there are:

- discontinuous states;
- changes in dynamics;
- staging or separation;
- impact events;
- different control sets;
- phase-specific constraints.

For multiple phases, event constraints connect the terminal state of one phase to the initial state of another:

$$\boldsymbol{\psi}\left(\boldsymbol{x}_f^{(q)}, \boldsymbol{x}_0^{(q+1)}, \boldsymbol{p}\right) = \mathbf{0}.$$

Verification and Validation

A converged NLP solution is not automatically a trustworthy continuous-time solution.
Verification should include:

1. mesh refinement until the objective stabilizes;
2. independent integration of the dynamics;
3. endpoint residual checks;
4. path-constraint checking between nodes;
5. comparison across mesh strategies;
6. sensitivity to tolerances;
7. examination of control oscillations;
8. Hamiltonian or costate checks when available.

Common Mistakes

- forgetting the factor $(t_f - t_0)/2$;
- forgetting the interval scaling $2/\Delta s_k$ in $\boldsymbol{D}^{(k)}$;
- using standard LGR weights without interval scaling;
- imposing control continuity when it is not required;
- failing to impose state continuity;
- confusing global coordinate s with local coordinate τ;
- treating mesh points as NLP variables in a fixed-mesh transcription;
- applying one global differentiation matrix across discontinuities;
- failing to check path constraints between nodes;
- using excessive polynomial order in a nonsmooth interval.

Summary

The multiple-interval LGR method extends single-interval pseudospectral collocation to realistic optimal control problems.
The principal ideas are:

1. transform the physical time domain to a fixed global domain;
2. divide the transformed domain into mesh intervals;
3. map standard LGR points into each interval;
4. approximate the state locally using Lagrange polynomials;
5. enforce the dynamics at local LGR points;
6. approximate the cost using composite LGR quadrature;
7. enforce state continuity between intervals;

8. exploit the resulting block-sparse NLP structure;
9. refine the mesh until the discretization error is acceptable.

This method combines spectral accuracy in smooth regions with the flexibility required for nonsmooth controls and localized trajectory features.

Connection. Local interval equations become a useful computational method only after their states, controls, differentiation blocks, and continuity conditions are assembled globally.

Global Assembly of Multiple-Interval LGR Transcriptions

This section completes the derivation of the multiple-interval Legendre–Gauss–Radau (LGR) pseudospectral method. The preceding development established the local state approximation, local differentiation matrix, local quadrature rule, and collocation equations on each mesh interval. The present chapter assembles those local objects into a single global nonlinear programming problem.

The central result is that the multiple-interval method retains almost exactly the same mathematical form as the single-interval LGR method:

$$\boldsymbol{DX} = \frac{t_f - t_0}{2}\boldsymbol{F},$$

but the global differentiation matrix is sparse and is constructed from local interval matrices whose adjacent blocks share one state column.

The chapter develops:

- global stacking of LGR nodes, weights, states, and controls;
- shared state variables at mesh interfaces;
- the unique control representation at interval interfaces;
- assembly of the global differentiation matrix;
- compact quadrature and cost expressions;
- boundary and path constraints;
- the final sparse nonlinear programming formulation;
- implementation and verification procedures;
- motivation for adaptive mesh refinement.

Transformed Time Domain and Mesh

The physical time interval

$$t \in [t_0, t_f]$$

is mapped to the fixed domain

$$s \in [-1, 1]$$

using

$$s = \frac{2(t - t_0)}{t_f - t_0} - 1.$$

The inverse mapping is

$$t(s) = \frac{t_f - t_0}{2}s + \frac{t_f + t_0}{2}.$$

The transformed dynamics are

$$\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}s} = \frac{t_f - t_0}{2}\boldsymbol{f}\left(\boldsymbol{x}(s), \boldsymbol{u}(s), t(s), \boldsymbol{p}\right).$$

The transformed domain is divided into K mesh intervals:

$$-1 = s_0 < s_1 < \cdots < s_{K-1} < s_K = 1.$$

The kth interval is

$$\mathcal{S}_k = [s_{k-1}, s_k].$$

The mesh points are fixed during a single nonlinear programming solve. They may be changed between solves during mesh refinement.

Local LGR Approximation

Let interval k contain N_k LGR collocation points:

$$s_1^{(k)}, s_2^{(k)}, \ldots, s_{N_k}^{(k)}.$$

Forward LGR includes the left interval endpoint but excludes the right endpoint:

$$s_1^{(k)} = s_{k-1}, \qquad s_{N_k}^{(k)} < s_k.$$

The additional state point is

$$s_{N_k+1}^{(k)} = s_k.$$

The state approximation is

$$\boldsymbol{x}^{(k)}(s) \approx \boldsymbol{X}^{(k)}(s) = \sum_{i=1}^{N_k+1} \boldsymbol{X}_i^{(k)} \ell_i^{(k)}(s),$$

where

$$\ell_i^{(k)}(s) = \prod_{\substack{j=1 \\ j \neq i}}^{N_k+1} \frac{s - s_j^{(k)}}{s_i^{(k)} - s_j^{(k)}}.$$

The control is represented at only the N_k collocation points:

$$\boldsymbol{U}_i^{(k)} \approx \boldsymbol{u}\left(s_i^{(k)}\right), \qquad i = 1, \ldots, N_k.$$

Local Differentiation Matrix

Differentiating the state approximation gives

$$\frac{\mathrm{d}\boldsymbol{X}^{(k)}}{\mathrm{d}s} = \sum_{i=1}^{N_k+1} \boldsymbol{X}_i^{(k)} \frac{\mathrm{d}\ell_i^{(k)}}{\mathrm{d}s}.$$

At collocation node $s_j^{(k)}$,

$$\left.\frac{\mathrm{d}\boldsymbol{X}^{(k)}}{\mathrm{d}s}\right|_{s=s_j^{(k)}} = \sum_{i=1}^{N_k+1} D_{ji}^{(k)} \boldsymbol{X}_i^{(k)},$$

where

$$D_{ji}^{(k)} = \left.\frac{\mathrm{d}\ell_i^{(k)}}{\mathrm{d}s}\right|_{s=s_j^{(k)}}.$$

Thus,

$$\boldsymbol{D}^{(k)} \in \mathbb{R}^{N_k \times (N_k+1)}.$$

The local collocation equations are

$$\boxed{\sum_{i=1}^{N_k+1} D_{ji}^{(k)} \boldsymbol{X}_i^{(k)} = \frac{t_f - t_0}{2} \boldsymbol{f}\left(\boldsymbol{X}_j^{(k)}, \boldsymbol{U}_j^{(k)}, t_j^{(k)}, \boldsymbol{p}\right).}$$

These equations hold for

$$j = 1, \ldots, N_k.$$

State Continuity at Mesh Points

The final state of interval k and the initial state of interval $k+1$ represent the same physical state:

$$\boxed{\boldsymbol{X}_{N_k+1}^{(k)} = \boldsymbol{X}_1^{(k+1)}.}$$

This relation can be implemented in two ways.

Explicit Continuity Constraints

Retain separate variables and impose

$$\boldsymbol{X}_{N_k+1}^{(k)} - \boldsymbol{X}_1^{(k+1)} = \boldsymbol{0}.$$

Shared Interface Variables

Use one NLP variable for both quantities:

$$\boldsymbol{X}_{N_k+1}^{(k)} \equiv \boldsymbol{X}_1^{(k+1)}.$$

The shared-variable formulation is particularly elegant because it creates the overlapping-column structure of the global differentiation matrix.

Control Representation at Mesh Points

The forward LGR scheme does not contain a control at the right endpoint of interval k. It does contain a control at the left endpoint of interval $k+1$.

Therefore, the control at an interior mesh point has one representation:

$$\boldsymbol{U}_1^{(k+1)}.$$

There is no second control variable

$$\boldsymbol{U}_{N_k+1}^{(k)}$$

because the terminal state point is not a collocation point.

This property avoids an artificial double-valued control at the interface.

> **Important LGR Interface Property**
> At an interior mesh point, the state is shared between adjacent intervals, while the control is represented only from the interval beginning at that mesh point.

The only point on the full transformed domain without a control variable is the final endpoint:

$$s_K = 1.$$

Total Number of LGR Points

Define the total number of collocation points:

$$\boxed{N = \sum_{k=1}^{K} N_k.}$$

There are:

- N global collocation points;
- $N+1$ unique global state points;
- N global control points;

- N global quadrature weights.

The additional state point is the terminal point of the final interval.

Global Collocation-Point Vector

Stack the local LGR nodes in monotonically increasing order:

$$\widetilde{\boldsymbol{S}} = \begin{bmatrix} s_1^{(1)} \\ \vdots \\ s_{N_1}^{(1)} \\ s_1^{(2)} \\ \vdots \\ s_{N_2}^{(2)} \\ \vdots \\ s_1^{(K)} \\ \vdots \\ s_{N_K}^{(K)} \end{bmatrix} \in \mathbb{R}^N.$$

The complete state-node vector includes the terminal point:

$$\boldsymbol{S} = \begin{bmatrix} \widetilde{\boldsymbol{S}} \\ s_{N_K+1}^{(K)} \end{bmatrix} = \begin{bmatrix} \widetilde{\boldsymbol{S}} \\ 1 \end{bmatrix} \in \mathbb{R}^{N+1}.$$

All entries in $\widetilde{\boldsymbol{S}}$ are collocation points. The final entry of $\boldsymbol{S}$ is not a collocation point.

Global Weight Vector

For interval k, let the LGR quadrature weights be

$$w_1^{(k)}, \ldots, w_{N_k}^{(k)}.$$

Stack them:

$$\boxed{\boldsymbol{W} = \begin{bmatrix} w_1^{(1)} \\ \vdots \\ w_{N_1}^{(1)} \\ w_1^{(2)} \\ \vdots \\ w_{N_2}^{(2)} \\ \vdots \\ w_1^{(K)} \\ \vdots \\ w_{N_K}^{(K)} \end{bmatrix} \in \mathbb{R}^N.}$$

There is no weight corresponding to the terminal state point.

Global State Matrix

Assume there are n_x states. The unique state values are stacked as

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{X}_1^{(1)} \\ \vdots \\ \boldsymbol{X}_{N_1}^{(1)} \\ \boldsymbol{X}_1^{(2)} \\ \vdots \\ \boldsymbol{X}_{N_2}^{(2)} \\ \vdots \\ \boldsymbol{X}_1^{(K)} \\ \vdots \\ \boldsymbol{X}_{N_K}^{(K)} \\ \boldsymbol{X}_{N_K+1}^{(K)} \end{bmatrix}.$$

With one row per state node,

$$\boldsymbol{X} \in \mathbb{R}^{(N+1)\times n_x}.$$

The interface states are not duplicated. For example,

$$\boldsymbol{X}_{N_1+1}^{(1)} = \boldsymbol{X}_1^{(2)}$$

is represented by one shared row in the global state matrix.

Collocation-State Matrix

Path constraints and dynamics are evaluated only at collocation nodes. Define

$$\boxed{\widetilde{\boldsymbol{X}} = \boldsymbol{X}(1:N,:) \in \mathbb{R}^{N \times n_x}.}$$

Thus, $\widetilde{\boldsymbol{X}}$ excludes the final noncollocated state.

Global Control Matrix

Assume there are n_u controls. Stack the control values:

$$\boldsymbol{U} = \begin{bmatrix} \boldsymbol{U}_1^{(1)} \\ \vdots \\ \boldsymbol{U}_{N_1}^{(1)} \\ \boldsymbol{U}_1^{(2)} \\ \vdots \\ \boldsymbol{U}_{N_2}^{(2)} \\ \vdots \\ \boldsymbol{U}_1^{(K)} \\ \vdots \\ \boldsymbol{U}_{N_K}^{(K)} \end{bmatrix} \in \mathbb{R}^{N \times n_u}.$$

The number of control rows equals the number of global collocation points.

Global Dynamics Matrix

Define

$$\boldsymbol{F} = \begin{bmatrix} f\left(\boldsymbol{X}_1^{(1)}, \boldsymbol{U}_1^{(1)}, t_1^{(1)}, \boldsymbol{p}\right) \\ \vdots \\ f\left(\boldsymbol{X}_{N_1}^{(1)}, \boldsymbol{U}_{N_1}^{(1)}, t_{N_1}^{(1)}, \boldsymbol{p}\right) \\ \vdots \\ f\left(\boldsymbol{X}_{N_K}^{(K)}, \boldsymbol{U}_{N_K}^{(K)}, t_{N_K}^{(K)}, \boldsymbol{p}\right) \end{bmatrix}.$$

Therefore,

$$\boldsymbol{F} \in \mathbb{R}^{N \times n_x}.$$

Assembly of the Global Differentiation Matrix

Each local matrix is

$$\boldsymbol{D}^{(k)} \in \mathbb{R}^{N_k \times (N_k+1)}.$$

The final state column of interval k is the initial state column of interval $k+1$. Therefore, adjacent local matrices overlap in one global column.

The global matrix has dimension

$$\boxed{\boldsymbol{D} \in \mathbb{R}^{N\times(N+1)}.}$$

It is not a conventional block-diagonal matrix because adjacent blocks share one column. It is more accurately described as a block-sparse matrix with one-column overlap.

Two-Interval Example

For two intervals with N_1 and N_2 points:

$$\boldsymbol{D} = \begin{bmatrix} \boldsymbol{D}^{(1)} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{D}^{(2)} \end{bmatrix}_{\text{with shared interface column}}.$$

More explicitly, partition

$$\boldsymbol{D}^{(1)} = \begin{bmatrix} \boldsymbol{D}_A^{(1)} & \boldsymbol{d}_b^{(1)} \end{bmatrix},$$

where

$$\boldsymbol{D}_A^{(1)} \in \mathbb{R}^{N_1\times N_1}, \qquad \boldsymbol{d}_b^{(1)} \in \mathbb{R}^{N_1}.$$

Similarly,

$$\boldsymbol{D}^{(2)} = \begin{bmatrix} \boldsymbol{d}_a^{(2)} & \boldsymbol{D}_B^{(2)} \end{bmatrix}.$$

Then

$$\boxed{\boldsymbol{D} = \begin{bmatrix} \boldsymbol{D}_A^{(1)} & \boldsymbol{d}_b^{(1)} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{d}_a^{(2)} & \boldsymbol{D}_B^{(2)} \end{bmatrix}.}$$

The central column corresponds to the shared interface state.

General K-Interval Structure

The global matrix has the schematic form

$$\boldsymbol{D} = \begin{bmatrix} \boxed{\boldsymbol{D}^{(1)}} & & & \\ & \boxed{\boldsymbol{D}^{(2)}} & & \\ & & \ddots & \\ & & & \boxed{\boldsymbol{D}^{(K)}} \end{bmatrix},$$

where neighboring boxes overlap by one state column.

This structure yields a sparse matrix even when the total number of collocation points is large.

Global Collocation Equations

The local collocation equations from all mesh intervals combine into

$$\boxed{\boldsymbol{D}\boldsymbol{X} = \frac{t_f - t_0}{2}\boldsymbol{F}.}$$

Equivalently, define the global defect matrix:

$$\boxed{\boldsymbol{\Delta} = \boldsymbol{D}\boldsymbol{X} - \frac{t_f - t_0}{2}\boldsymbol{F}.}$$

The NLP enforces

$$\boldsymbol{\Delta} = \boldsymbol{0}.$$

The dimensions are

$$\begin{aligned}
\boldsymbol{D} &\in \mathbb{R}^{N\times(N+1)},\\
\boldsymbol{X} &\in \mathbb{R}^{(N+1)\times n_x},\\
\boldsymbol{F} &\in \mathbb{R}^{N\times n_x},\\
\boldsymbol{\Delta} &\in \mathbb{R}^{N\times n_x}.
\end{aligned}$$

The multiplication is therefore conformable.

Composite LGR Quadrature

Let the transformed running cost at each collocation point be

$$L_i^{(k)} = L\left(\boldsymbol{X}_i^{(k)}, \boldsymbol{U}_i^{(k)}, t_i^{(k)}, \boldsymbol{p}\right).$$

Stack the values:

$$\boldsymbol{L} = \begin{bmatrix} L_1^{(1)} \\ \vdots \\ L_{N_1}^{(1)} \\ \vdots \\ L_1^{(K)} \\ \vdots \\ L_{N_K}^{(K)} \end{bmatrix} \in \mathbb{R}^N.$$

The integral is approximated as

$$\int_{-1}^{1} L(s)\,\mathrm{d}s \approx \sum_{k=1}^{K}\sum_{i=1}^{N_k} w_i^{(k)} L_i^{(k)}.$$

Using the stacked vectors:

$$\boxed{\int_{-1}^{1} L(s)\,\mathrm{d}s \approx \boldsymbol{W}^{\mathsf{T}}\boldsymbol{L}.}$$

Discrete Cost Functional

The Bolza objective is

$$J = \Phi\left(\boldsymbol{x}(t_0), t_0, \boldsymbol{x}(t_f), t_f, \boldsymbol{p}\right) + \int_{t_0}^{t_f} L\left(\boldsymbol{x}, \boldsymbol{u}, t, \boldsymbol{p}\right)\,\mathrm{d}t.$$

The multiple-interval LGR approximation is

$$\boxed{J_h = \Phi\left(\boldsymbol{X}_1^{(1)}, t_0, \boldsymbol{X}_{N_K+1}^{(K)}, t_f, \boldsymbol{p}\right) + \frac{t_f - t_0}{2}\boldsymbol{W}^{\mathsf{T}}\boldsymbol{L}.}$$

This compact expression has exactly the same form as the single-interval LGR objective.

Boundary Constraints

The discrete endpoint constraints are

$$\boxed{\boldsymbol{\phi}\left(\boldsymbol{X}_1^{(1)}, t_0, \boldsymbol{X}_{N_K+1}^{(K)}, t_f, \boldsymbol{p}\right) = \boldsymbol{0}.}$$

Only the first state and final state enter the boundary constraints.
More general bounds may be written as

$$\boldsymbol{\phi}_{\min} \le \boldsymbol{\phi}\left(\boldsymbol{X}_1^{(1)}, t_0, \boldsymbol{X}_{N_K+1}^{(K)}, t_f, \boldsymbol{p}\right) \le \boldsymbol{\phi}_{\max}.$$

Path Constraints

Path constraints are evaluated at all collocation points:

$$\boxed{\boldsymbol{c}_{\min} \le \boldsymbol{c}\left(\widetilde{\boldsymbol{X}}, \boldsymbol{U}, \widetilde{\boldsymbol{S}}, t_0, t_f, \boldsymbol{p}\right) \le \boldsymbol{c}_{\max}.}$$

Here:

- $\widetilde{\boldsymbol{X}}$ excludes the terminal state;
- $\boldsymbol{U}$ has one row per collocation point;
- $\widetilde{\boldsymbol{S}}$ excludes the terminal point;

- physical time is recovered from $\widetilde{S}$.

The final state point is excluded because no control or collocation equation is defined there.

Physical-Time Vector

The global collocation-time vector is

$$\widetilde{\boldsymbol{t}} = \frac{t_f - t_0}{2}\widetilde{\boldsymbol{S}} + \frac{t_f + t_0}{2}\mathbf{1}_N.$$

The full state-time vector is

$$\boldsymbol{t} = \frac{t_f - t_0}{2}\boldsymbol{S} + \frac{t_f + t_0}{2}\mathbf{1}_{N+1}.$$

If t_0 or t_f is free, every nodal time depends on those decision variables.

Final Nonlinear Programming Problem

The complete multiple-interval LGR transcription is

$$\boxed{\min_{\boldsymbol{X},\boldsymbol{U},t_0,t_f,\boldsymbol{p}} J_h}$$

subject to

$$\begin{aligned}
\boldsymbol{D}\boldsymbol{X} - \frac{t_f - t_0}{2}\boldsymbol{F} &= \mathbf{0}, \\
\boldsymbol{\phi}\left(\boldsymbol{X}_1, t_0, \boldsymbol{X}_{N+1}, t_f, \boldsymbol{p}\right) &= \mathbf{0}, \\
\boldsymbol{c}_{\min} \le \boldsymbol{c}\left(\widetilde{\boldsymbol{X}}, \boldsymbol{U}, \widetilde{\boldsymbol{t}}, \boldsymbol{p}\right) &\le \boldsymbol{c}_{\max}, \\
\boldsymbol{Z}_{\min} \le \boldsymbol{Z} &\le \boldsymbol{Z}_{\max}.
\end{aligned}$$

A convenient decision vector is

$$\boldsymbol{Z} = \begin{bmatrix} \operatorname{vec}(\boldsymbol{X}) \\ \operatorname{vec}(\boldsymbol{U}) \\ t_0 \\ t_f \\ \boldsymbol{p} \end{bmatrix}.$$

Equivalence of Global and Multiple-Interval Forms

The single-interval LGR collocation equation is

$$\boldsymbol{D}_{\text{single}}\boldsymbol{X} = \frac{t_f - t_0}{2}\boldsymbol{F}.$$

The multiple-interval equation is

$$\boldsymbol{D}_{\text{multi}}\boldsymbol{X} = \frac{t_f - t_0}{2}\boldsymbol{F}.$$

The algebraic form is identical. The difference lies in the structure of $\boldsymbol{D}$:

- single interval: dense global differentiation matrix;
- multiple intervals: sparse local-block matrix with shared interface columns.

Likewise, the objective retains the form

$$J_h = \Phi_h + \frac{t_f - t_0}{2}\boldsymbol{W}^\top \boldsymbol{L}.$$

This structural equivalence simplifies implementation.

Jacobian Sparsity

The global defect constraints are

$$\boldsymbol{\Delta} = \boldsymbol{D}\boldsymbol{X} - \frac{t_f - t_0}{2}\boldsymbol{F}(\widetilde{\boldsymbol{X}}, \boldsymbol{U}, \widetilde{\boldsymbol{t}}, \boldsymbol{p}).$$

A defect row in interval k depends only on:

- states in interval k;
- controls in interval k;
- t_0 and t_f;
- global static parameters.

Therefore, the Jacobian with respect to state and control variables is block sparse.
The principal Jacobian terms are

$$\frac{\partial \boldsymbol{\Delta}}{\partial \boldsymbol{X}} = \boldsymbol{D} - \frac{t_f - t_0}{2}\frac{\partial \boldsymbol{F}}{\partial \boldsymbol{X}},$$

and

$$\frac{\partial \boldsymbol{\Delta}}{\partial \boldsymbol{U}} = -\frac{t_f - t_0}{2}\frac{\partial \boldsymbol{F}}{\partial \boldsymbol{U}}.$$

Number of Variables and Constraints

Assume:

- n_x state variables;
- n_u control variables;
- n_p static parameters;
- N total collocation points.

The state variables contribute

$$(N+1)n_x.$$

The control variables contribute

$$Nn_u.$$

Including time and parameters:

$$\boxed{n_z = (N+1)n_x + Nn_u + 2 + n_p.}$$

The defect equations contribute

$$\boxed{n_{\text{defect}} = Nn_x.}$$

Additional constraints arise from endpoint and path conditions.

Implementation Strategy

A robust implementation separates mesh construction from NLP evaluation.

Preprocessing

Before solving:

1. choose mesh points s_k;
2. choose degrees N_k;
3. compute standard LGR nodes and weights;
4. map nodes into each interval;
5. scale interval quadrature weights;
6. scale interval differentiation matrices;
7. assemble $\widetilde{\boldsymbol{S}}$, $\boldsymbol{S}$, $\boldsymbol{W}$, and $\boldsymbol{D}$;
8. create state and control index maps.

NLP Evaluation

At each NLP iteration:

1. unpack $\boldsymbol{X}$, $\boldsymbol{U}$, t_0, t_f, and $\boldsymbol{p}$;
2. compute the physical-time vector;
3. evaluate $\boldsymbol{F}$ and $\boldsymbol{L}$;
4. evaluate the cost;

5. evaluate defects;
6. evaluate endpoint constraints;
7. evaluate path constraints;
8. evaluate derivatives.

MATLAB Assembly of the Global Matrix

```
function D = assembleGlobalLGR(Dlocal, degree)

K = numel(degree);
N = sum(degree);

D = sparse(N,N+1);

rowStart = 1;
colStart = 1;

for k = 1:K

    Nk = degree(k);

    rows = rowStart:(rowStart+Nk -1);
    cols = colStart:(colStart+Nk);

    D(rows,cols) = Dlocal{k};

    rowStart = rowStart + Nk;

    % Advance by Nk, not Nk+1, because the last
    % state column is shared with the next interval.
    colStart = colStart + Nk;
end
end
```

MATLAB Stacking of Nodes and Weights

```
function [Scol,Sstate,W] = stackLGRData(nodes,weights,mesh)

K = numel(nodes);

Scol = [];
W = [];

for k = 1:K
    Scol = [Scol; nodes{k}(:)];
    W = [W; weights{k}(:)];
end

Sstate = [Scol; mesh(end)];
```

```
end
```

MATLAB Defect Function

```
function defect = lgrDefect(X,U,t0,tf,Scol,D,p)

N = size(U,1);

t = 0.5*(tf -t0)*Scol ...
  + 0.5*(tf+t0)*ones(N,1);

F = dynamics(X(1:N,:),U,t,p);

defect = D*X - 0.5*(tf -t0)*F;

end
```

MATLAB Objective Function

```
function J = lgrObjective(X,U,t0,tf,Scol,W,p)

N = size(U,1);

t = 0.5*(tf -t0)*Scol ...
  + 0.5*(tf+t0)*ones(N,1);

L = runningCost(X(1:N,:),U,t,p);

M = endpointCost(X(1,:),t0,X(end,:),tf,p);

J = M + 0.5*(tf -t0)*(W.'*L);

end
```

Indexing Example

Suppose

$$K = 3, \qquad N_1 = 3, \qquad N_2 = 4, \qquad N_3 = 2.$$

Then

$$N = 3 + 4 + 2 = 9.$$

The global state rows are:

$$
\begin{aligned}
1{:}4 &\quad \text{for interval 1,}\\
4{:}8 &\quad \text{for interval 2,}\\
8{:}10 &\quad \text{for interval 3.}
\end{aligned}
$$

Rows 4 and 8 are shared interface states.
The global differentiation matrix has size

$$9 \times 10.$$

The controls have nine rows. The weight vector has nine entries.

Integral State Consistency

The local state update can be checked using LGR quadrature:

$$\boldsymbol{X}_{N_k+1}^{(k)} - \boldsymbol{X}_1^{(k)} \approx \frac{t_f - t_0}{2} \sum_{i=1}^{N_k} w_i^{(k)} \boldsymbol{F}_i^{(k)}.$$

For the full horizon:

$$\boldsymbol{X}_{N+1} - \boldsymbol{X}_1 \approx \frac{t_f - t_0}{2} \boldsymbol{W}^{\mathsf{T}} \boldsymbol{F}.$$

For vector-valued dynamics, the weighted sum is applied columnwise.
This identity is useful for debugging quadrature and indexing.

Why the Final Point Has No Control

The LGR state polynomial requires $N_k + 1$ state values, but the dynamics are enforced only at N_k LGR points. The additional endpoint is needed to complete the polynomial and represent the terminal state.
Therefore:

$$\text{state nodes} = N + 1, \qquad \text{control nodes} = N.$$

A terminal control can be estimated after solving by extrapolation if needed, but it is not an independent NLP variable.

Path-Constraint Verification between Nodes

Enforcing a path constraint only at collocation points does not guarantee that it is satisfied between nodes.
After solving, evaluate the state and control polynomials on a dense grid:

$$s_{\text{check},m} \in [-1, 1].$$

Then verify

$$\boldsymbol{c}_{\min} \leq \boldsymbol{c}\left(\boldsymbol{x}_{\text{interp}}, \boldsymbol{u}_{\text{interp}}, t_{\text{interp}}\right) \leq \boldsymbol{c}_{\max}.$$

Violations may require:

- mesh refinement;
- additional constraint-check nodes;
- constraint tightening;
- higher polynomial degree.

Motivation for Mesh Refinement

The derivation above assumes that the mesh points and polynomial degrees are given. In practice, the important questions are:

- Where should mesh points be placed?
- How many intervals are required?
- What polynomial degree should be used in each interval?
- How should discontinuities be detected?
- When should an interval be subdivided?
- When should the polynomial degree be increased?

These questions motivate adaptive mesh-refinement algorithms.

h-, p-, and hp-Refinement

h-Refinement

Decrease interval width by subdividing:

$$\rightarrow [s_{k-1}, s_m] \cup [s_m, s_k].$$

This is suitable near discontinuities or sharp local features.

p-Refinement

Increase N_k while keeping the interval fixed.
This is suitable when the solution is smooth.

hp-Refinement

Adapt both interval width and polynomial degree.
A practical strategy is:

- use p-refinement when coefficient decay indicates smoothness;
- use h-refinement when nonsmoothness is detected;

- use both when the appropriate classification is uncertain.

Error Estimation Concept

Let $\boldsymbol{X}_{\text{coll}}^{(k)}(s)$ be the collocation polynomial. Construct an independent state approximation by integrating the dynamics:

$$\boldsymbol{X}_{\text{int}}^{(k)}(s) = \boldsymbol{X}_{1}^{(k)} + \frac{t_f - t_0}{2} \int_{s_{k-1}}^{s} \boldsymbol{f}\left(\boldsymbol{X}_{\text{coll}}^{(k)}(\sigma), \boldsymbol{U}^{(k)}(\sigma), t(\sigma)\right) \, \mathrm{d}\sigma.$$

Define

$$e_k = \max_{s \in \mathcal{S}_k} \frac{\left\| \boldsymbol{X}_{\text{int}}^{(k)}(s) - \boldsymbol{X}_{\text{coll}}^{(k)}(s) \right\|_\infty}{1 + \left\| \boldsymbol{X}_{\text{coll}}^{(k)}(s) \right\|_\infty}.$$

If

$$e_k > \epsilon,$$

refine interval k.

Warm Starting after Mesh Refinement

After changing the mesh:

1. interpolate the previous state solution onto the new state nodes;
2. interpolate the previous control solution onto the new collocation nodes;
3. preserve the previous t_0, t_f, and parameters;
4. project variables onto bounds if necessary;
5. solve the refined NLP.

Warm starting greatly improves efficiency.

Numerical Conditioning

Potential sources of poor conditioning include:

- very high polynomial degree;
- extremely small mesh intervals;
- badly scaled states;
- inconsistent derivative scaling;
- large differences among state magnitudes;
- nearly singular endpoint constraints.

A practical implementation should scale variables and avoid excessive polynomial degrees within a single interval.

Verification Checklist

A multiple-interval LGR solution should be checked using:

1. defect residuals;
2. endpoint residuals;
3. path constraints at and between nodes;
4. mesh-refinement convergence;
5. objective convergence;
6. independent forward integration;
7. state continuity at interfaces;
8. control behavior near switches;
9. consistency of matrix dimensions;
10. sensitivity to the initial mesh.

Common Implementation Errors

- duplicating interface states;
- failing to overlap one state column between adjacent $\boldsymbol{D}^{(k)}$ blocks;
- assigning a control variable to the final state point;
- attaching a quadrature weight to the final point;
- using $\boldsymbol{X}$ instead of $\widetilde{\boldsymbol{X}}$ in path constraints;
- forgetting the global factor $(t_f - t_0)/2$;
- double-scaling local differentiation matrices;
- using inconsistent node order;
- mapping local nodes incorrectly;
- constructing an $N \times N$ matrix instead of $N \times (N+1)$;
- imposing unnecessary control continuity;
- failing to check constraints between nodes.

Summary

The main results of this section are:

1. The total number of LGR collocation points is

$$N = \sum_{k=1}^{K} N_k.$$

2. There are $N+1$ unique state nodes and N control nodes.

3. Adjacent intervals share one state variable at each mesh point.
4. The forward LGR control has one value at each interior mesh point.
5. The global weight vector is obtained by stacking local weights.
6. The global differentiation matrix has size

$$N \times (N + 1).$$

7. Adjacent local differentiation blocks overlap by one state column.
8. The global defect equation is

$$\boldsymbol{D}\boldsymbol{X} - \frac{t_f - t_0}{2}\boldsymbol{F} = \boldsymbol{0}.$$

9. The discrete objective is

$$J_h = \Phi_h + \frac{t_f - t_0}{2}\boldsymbol{W}^\top \boldsymbol{L}.$$

10. The multiple-interval and single-interval methods have the same compact algebraic form.
11. Their principal difference is the sparse structure of the differentiation matrix.
12. Mesh refinement determines where intervals and polynomial degrees should be placed.

Connection. Once the global method is available, interpolation and integration error estimates can drive an adaptive choice between mesh subdivision and polynomial enrichment.

Adaptive Mesh Refinement for Pseudospectral Optimal Control

Adaptive mesh refinement is the process of repeatedly modifying the discretization until the numerical solution satisfies a prescribed accuracy requirement. In a multiple-interval pseudospectral method, the mesh is defined by:

- the locations of the mesh points;
- the number of mesh intervals;
- the polynomial degree in each interval.

The refinement process is iterative:

1. choose an initial mesh;
2. solve the nonlinear programming problem;
3. estimate the discretization error;
4. modify the mesh;
5. interpolate the previous solution onto the new mesh;
6. resolve the NLP;
7. repeat until the accuracy tolerance is satisfied.

This procedure is analogous to adaptive step-size integration for ordinary differential equations. In smooth regions, relatively large intervals or higher-order polynomials may provide excellent accuracy. Near rapid changes, switching points, singular arcs, boundary layers, or discontinuities, smaller intervals are often necessary.

This section develops an interpolation-and-integration error estimator for multiple-interval Legendre–Gauss–Radau (LGR) collocation and explains how the estimated error can guide h-, p-, and hp-refinement.

Initial Mesh and Polynomial Degrees

The transformed time domain is

$$s \in [-1, 1].$$

Choose mesh points

$$-1 = s_0 < s_1 < \cdots < s_{K-1} < s_K = 1.$$

The kth mesh interval is

$$\mathcal{S}_k = [s_{k-1}, s_k].$$

Assign N_k LGR collocation points to interval k:

$$s_1^{(k)}, \ldots, s_{N_k}^{(k)}.$$

The state is approximated using $N_k + 1$ points:

$$s_1^{(k)}, \ldots, s_{N_k}^{(k)}, s_{N_k+1}^{(k)},$$

where

$$s_{N_k+1}^{(k)} = s_k.$$

Thus, the discretization is determined by:

$$\mathcal{M} = \{s_0, \ldots, s_K; N_1, \ldots, N_K\}.$$

Mesh Refinement as an Iterative Process

Let $\mathcal{M}^{(r)}$ denote the mesh at refinement iteration r. The basic process is

$$\mathcal{M}^{(r)} \rightarrow \text{NLP solution} \rightarrow \text{error estimate} \rightarrow \mathcal{M}^{(r+1)}.$$

The iteration terminates when

$$e_k \leq \epsilon \qquad \text{for all } k = 1, \ldots, K,$$

where e_k is an interval-wise error indicator and ϵ is the required tolerance.

Core Difficulty
Solving the NLP on a fixed mesh is usually straightforward. The difficult questions are how to estimate the error and how to modify the mesh efficiently.

State Approximation on One Mesh Interval

Consider interval

$$\mathcal{S}_k = [s_{k-1}, s_k].$$

The LGR state polynomial is

$$\boldsymbol{X}^{(k)}(s) = \sum_{i=1}^{N_k+1} \boldsymbol{X}_i^{(k)} \ell_i^{(k)}(s),$$

where

$$\ell_i^{(k)}(s) = \prod_{\substack{j=1 \\ j \neq i}}^{N_k+1} \frac{s - s_j^{(k)}}{s_i^{(k)} - s_j^{(k)}}.$$

Because the state polynomial is known, it can be evaluated at any point in the interval.

Constructing an Enriched LGR Grid

To assess accuracy, create a new set of LGR points containing one additional collocation point:

$$M_k = N_k + 1.$$

Let the enriched LGR nodes be

$$\widetilde{s}_1^{(k)}, \ldots, \widetilde{s}_{M_k}^{(k)}.$$

The associated additional terminal state point is

$$\widetilde{s}_{M_k+1}^{(k)} = s_k.$$

Because the new LGR set has a different polynomial degree, most of its interior points differ from the original set.

Interpolated State on the Enriched Grid

Evaluate the original state polynomial at the enriched nodes:

$$\boxed{\boldsymbol{Y}_j^{(k)} = \boldsymbol{X}^{(k)}\left(\widetilde{s}_j^{(k)}\right), \qquad j = 1, \ldots, M_k + 1.}$$

Equivalently,

$$\boldsymbol{Y}_j^{(k)} = \sum_{i=1}^{N_k+1} \boldsymbol{X}_i^{(k)} \ell_i^{(k)}\left(\widetilde{s}_j^{(k)}\right).$$

Define the state interpolation matrix

$$I_{ji}^{(k)} = \ell_i^{(k)}\left(\widetilde{s}_j^{(k)}\right).$$

Then

$$\boxed{\boldsymbol{Y}^{(k)} = \boldsymbol{I}_x^{(k)} \boldsymbol{X}^{(k)}.}$$

The state interpolation is unique because the state polynomial is explicitly known.

Why Control Interpolation Is Different

The control variables in a direct collocation method are nodal samples:

$$\boldsymbol{U}_i^{(k)} \approx \boldsymbol{u}\left(s_i^{(k)}\right).$$

Unlike the state, the control is not necessarily defined by a unique polynomial approximation. Infinitely many continuous functions can pass through the same set of control samples.

Therefore, the control reconstruction is not unique.

> **Important Distinction**
> The state has an explicit interpolation polynomial. The control is generally available only as sampled data at collocation points.

Piecewise-Linear Control Reconstruction

A simple reconstruction is piecewise-linear interpolation.

Let

$$\boldsymbol{V}_j^{(k)}$$

denote the control evaluated at the enriched node

$$\widetilde{s}_j^{(k)}.$$

Then

$$\boxed{\boldsymbol{V}_j^{(k)} = \mathrm{PLInterp}\left(\left\{s_i^{(k)}, \boldsymbol{U}_i^{(k)}\right\}_{i=1}^{N_k}, \widetilde{s}_j^{(k)}\right).}$$

Piecewise-linear interpolation is not guaranteed to reproduce the exact control, but it is:

- simple;
- stable;
- local;
- consistent with the control samples;
- usually sufficient for error estimation.

Implicit Integration Form of the LGR Method

The LGR collocation method can be written in differentiation form or in an equivalent implicit integration form.

For the original grid in interval k:

$$\boldsymbol{X}_{2:N_k+1}^{(k)} = \mathbf{1}\left(\boldsymbol{X}_1^{(k)}\right)^{\mathsf{T}} + \frac{t_f - t_0}{2}\boldsymbol{A}^{(k)}\boldsymbol{F}_{1:N_k}^{(k)},$$

where:

- $\boldsymbol{A}^{(k)}$ is the LGR integration matrix;
- $\boldsymbol{F}_i^{(k)}$ is the dynamics evaluated at node i;
- $\mathbf{1}$ replicates the initial state across the rows.

For a scalar state, this becomes

$$X_j^{(k)} = X_1^{(k)} + \frac{t_f - t_0}{2} \sum_{i=1}^{N_k} A_{ji}^{(k)} F_i^{(k)}, \qquad j = 2, \ldots, N_k + 1.$$

Integrated State on the Enriched Grid

On the enriched grid, evaluate the dynamics using the interpolated state and reconstructed control:

$$\widetilde{\boldsymbol{F}}_j^{(k)} = \boldsymbol{f}\left(\boldsymbol{Y}_j^{(k)}, \boldsymbol{V}_j^{(k)}, \widetilde{t}_j^{(k)}, \boldsymbol{p}\right), \qquad j = 1, \ldots, M_k.$$

The enriched physical time is

$$\widetilde{t}_j^{(k)} = \frac{t_f - t_0}{2} \widetilde{s}_j^{(k)} + \frac{t_f + t_0}{2}.$$

Use the enriched LGR integration matrix $\boldsymbol{B}^{(k)}$:

$$\boxed{\boldsymbol{Z}_{2:M_k+1}^{(k)} = \mathbf{1}\left(\boldsymbol{Y}_1^{(k)}\right)^{\mathsf{T}} + \frac{t_f - t_0}{2} \boldsymbol{B}^{(k)} \widetilde{\boldsymbol{F}}_{1:M_k}^{(k)}.}$$

The initial value satisfies

$$\boldsymbol{Z}_1^{(k)} = \boldsymbol{Y}_1^{(k)} = \boldsymbol{X}_1^{(k)}.$$

Two Independent State Estimates

The enriched grid now contains two state estimates.

Interpolation-Based Estimate

$$\boldsymbol{Y}_j^{(k)} = \boldsymbol{X}^{(k)}\left(\widetilde{s}_j^{(k)}\right).$$

This estimate comes directly from the original state polynomial.

Dynamics-Based Estimate

$$\boldsymbol{Z}_j^{(k)}$$

is obtained by integrating the dynamics on the enriched grid using the interpolated state and reconstructed control.

If the original solution is sufficiently accurate, the two estimates should be close.

Absolute Error Indicator

Define the pointwise error:

$$\boldsymbol{\delta}_j^{(k)} = \left| \boldsymbol{Y}_j^{(k)} - \boldsymbol{Z}_j^{(k)} \right|.$$

For multiple states, the absolute value is taken componentwise.
A scalar interval indicator is

$$\boxed{e_k^{\text{abs}} = \max_{j=1,\dots,M_k+1} \left\| \boldsymbol{Y}_j^{(k)} - \boldsymbol{Z}_j^{(k)} \right\|_\infty .}$$

Relative Error Indicator

A relative measure is often more useful:

$$\boxed{e_k = \max_j \max_{r=1,\dots,n_x} \frac{\left| Y_{jr}^{(k)} - Z_{jr}^{(k)} \right|}{1 + \left| Y_{jr}^{(k)} \right|}.}$$

Alternatively, each state can be normalized using a problem-specific scale S_r:

$$e_k = \max_j \max_r \frac{\left| Y_{jr}^{(k)} - Z_{jr}^{(k)} \right|}{S_r}.$$

The interval is accepted if

$$e_k \le \epsilon.$$

Interpretation of the Error Distribution

The maximum error alone determines whether refinement is needed. The spatial distribution of the pointwise error helps determine which type of refinement should be used.
Let

$$\delta_j^{(k)} = \left\| \boldsymbol{Y}_j^{(k)} - \boldsymbol{Z}_j^{(k)} \right\|_\infty .$$

Two common patterns are:

Localized Error

The tolerance is violated only in a small region.
This may indicate:

- a switching point;
- a boundary layer;
- a sharp curvature change;
- a path-constraint junction;
- a discontinuity.

The preferred response is usually h-refinement.

Distributed Error

The tolerance is violated throughout most of the interval.
This suggests that the polynomial degree is insufficient over the entire interval.
The preferred response is usually p-refinement.

h-Refinement

In h-refinement, an interval is subdivided:

$$\rightarrow [s_{k-1}, s_{\text{new}}] \cup [s_{\text{new}}, s_k].$$

A natural splitting location is the enriched point with maximum error:

$$j^\star = \arg\max_j \delta_j^{(k)},$$

and

$$\boxed{s_{\text{new}} = \widetilde{s}_{j^\star}^{(k)}.}$$

Safeguards should prevent the new point from being too close to either endpoint.

p-Refinement

In p-refinement, the interval remains unchanged and the polynomial degree is increased:

$$\boxed{N_k^{\text{new}} = N_k + \Delta p.}$$

Typical values are

$$\Delta p = 1 \qquad \text{or} \qquad \Delta p = 2.$$

A small increase is preferred because:

- very high degrees may cause poor conditioning;
- the NLP becomes denser within the interval;
- the error response can be reassessed after the next solve.

Maximum Polynomial Degree

Impose an upper bound:

$$N_k \leq N_{\max}.$$

Typical choices might be

$$N_{\max} = 8, \quad 10, \quad 12, \quad \text{or} \quad 14.$$

If p-refinement is indicated but

$$N_k = N_{\max},$$

the interval should be subdivided instead.
Thus:

$$\boxed{N_k = N_{\max} \quad \Longrightarrow \quad \text{apply } h\text{-refinement.}}$$

***hp*-Refinement**

An hp method permits both:

- interval subdivision (h);
- polynomial-degree increase (p).

A simple decision rule is:

$$\text{Refinement type} = \begin{cases} \text{none}, & e_k \leq \epsilon, \\ h, & \text{error is localized}, \\ p, & \text{error is distributed and } N_k < N_{\max}, \\ h, & \text{error is distributed and } N_k = N_{\max}. \end{cases}$$

Localized-Error Metric

To automate the decision, define the set

$$\mathcal{V}_k = \left\{ j : \delta_j^{(k)} > \epsilon \right\}.$$

Define the violation fraction:

$$\rho_k = \frac{|\mathcal{V}_k|}{M_k + 1}.$$

One simple rule is:

$$\rho_k \leq \rho_{\text{crit}} \quad \Longrightarrow \quad \text{localized error.}$$

For example,

$$\rho_{\text{crit}} = 0.25.$$

Then:

- if fewer than 25% of enriched points violate the tolerance, use h-refinement;
- otherwise, use p-refinement unless $N_{\max}$ has been reached.

Alternative Smoothness Indicator

Another strategy is to inspect the decay of polynomial coefficients.
Write the state approximation in a Legendre basis:

$$X^{(k)}(s) = \sum_{n=0}^{N_k} a_n^{(k)} P_n(\tau(s)).$$

For a smooth function, the coefficients should decay rapidly:

$$|a_n^{(k)}| \to 0 \quad \text{rapidly as } n \to N_k.$$

If the decay is fast, p-refinement is appropriate. If the decay is slow or irregular, h-refinement is safer.

Adaptive Refinement Algorithm

Algorithm: Adaptive hp Mesh Refinement

```
Choose an initial mesh and interval degrees.
Set tolerance epsilon, degree increment Delta  p, and N_{\max}.
for r=0,1,2,:
    Solve the NLP on the current mesh.
    for each interval k:
        Set M_k=N_k+1.
        Construct the enriched LGR grid.
        Interpolate the state onto the enriched grid.
        Reconstruct the control using piecewise -linear interpolation.
        Evaluate the dynamics on the enriched grid.
        Integrate the dynamics using the enriched LGR integration matrix.
        Compute the pointwise and interval errors.
        if e_k<=epsilon:
            Keep interval k unchanged.
        else if the error is localized:
            Split interval k near the maximum error.
        else if N_k<N_{\max}:
            Set N_k< - N_k+Delta  p.
```

```
        else:
            Split interval k near the maximum error.
        end if
    end for
    if all intervals satisfy the tolerance:
        Stop.
    end if
    Interpolate the previous solution onto the new mesh.
end for
```

Warm Starting the Refined NLP

After changing the mesh, the old solution should be mapped to the new nodes.
For new state node s_{new}:

$$\boldsymbol{X}_{\text{guess}}(s_{\text{new}}) = \boldsymbol{X}_{\text{old}}(s_{\text{new}}).$$

For new control node s_{new}:

$$\boldsymbol{U}_{\text{guess}}(s_{\text{new}}) = \text{PLInterp}\left(\boldsymbol{U}_{\text{old}}, s_{\text{new}}\right).$$

Retain the previous values of:

- initial time;
- final time;
- static parameters;
- phase-link variables.

Warm starts usually reduce the number of NLP iterations significantly.

Pure h Methods

A pure h method keeps the polynomial degree fixed:

$$N_k = N_{\text{fixed}} \quad \text{for all intervals.}$$

Only interval widths are adapted.
Advantages:

- simple implementation;
- robust near discontinuities;
- predictable local polynomial degree;
- straightforward error control.

Disadvantages:

- may require many intervals;

- may produce unnecessarily large NLPs;
- can over-refine near isolated switching points.

Pure p Methods

A pure p method uses one interval or fixed interval boundaries and increases polynomial degree.
Advantages:

- spectral convergence for smooth solutions;
- compact NLPs;
- high accuracy with relatively few nodes.

Disadvantages:

- performs poorly near discontinuities;
- may exhibit Gibbs oscillations;
- may become ill-conditioned at high degree;
- cannot localize resolution.

Why hp Methods Are Attractive

The hp strategy combines the strengths of both methods:

- use high-order approximations in smooth regions;
- use additional intervals near nonsmooth features;
- avoid unnecessary global degree growth;
- avoid excessive interval proliferation;
- keep the NLP smaller for a prescribed accuracy.

The main difficulty is classification: determining whether the interval is smooth enough for p-refinement or should instead be divided.

NLP Size Considerations

Let

$$N = \sum_{k=1}^{K} N_k.$$

For n_x states and n_u controls, the principal number of nodal variables is

$$(N + 1)n_x + Nn_u.$$

Pure h-refinement may increase K rapidly and thereby increase N.
Pure p-refinement increases the density of local differentiation blocks.

A successful hp method seeks a balance:

$$\text{required accuracy} \quad \text{with minimal } N.$$

Bang–Bang Double Integrator Example

Consider

$$\dot{x}_1 = x_2,$$
$$\dot{x}_2 = u,$$

with bounded control

$$-1 \le u \le 1.$$

For a minimum-time transfer, the optimal control is typically bang–bang:

$$u(t) = \begin{cases} +1, & t < t_s, \\ -1, & t > t_s. \end{cases}$$

Then:

- x_2 is piecewise linear;
- x_1 is piecewise quadratic.

An ideal mesh places an interval boundary at t_s. Then each state segment can be represented exactly by a low-degree polynomial.

A pure h method may repeatedly cluster intervals around the switch. An effective hp method should identify the localized error, place a mesh boundary near the switch, and retain low polynomial degrees on either side.

Why Over-Refinement Is Wasteful

Suppose the exact solution is piecewise quadratic. A discretization with many low-order intervals may approximate it accurately, but the NLP can be much larger than necessary.

The goal of adaptive refinement is not merely:

$$e \le \epsilon.$$

A better goal is:

$$\boxed{\text{achieve } e \le \epsilon \text{ using as few variables as practical.}}$$

This is especially important when using general-purpose sparse NLP solvers.

MATLAB-Like Error Estimator

```
function [err,delta,Y,Z] = estimateIntervalError( ...
```

```
    X,U,sOld,sNew,t0,tf,Bnew,p)

% Interpolate state polynomial to enriched grid.
Y = interpolateStateLagrange(X,sOld,sNew);

% Reconstruct control using piecewise -linear interpolation.
V = interp1(sOld(1:end -1),U,sNew(1:end -1), ...
            'linear','extrap');

% Map enriched nodes to physical time.
tNew = 0.5*(tf -t0)*sNew(1:end -1) ...
     + 0.5*(tf+t0);

% Evaluate dynamics at enriched collocation points.
Fnew = dynamics(Y(1:end -1,:),V,tNew,p);

% Integrate dynamics from the initial state.
Z = zeros(size(Y));
Z(1,:) = Y(1,:);
Z(2:end,:) = Y(1,:) ...
    + 0.5*(tf -t0)*(Bnew*Fnew);

% Relative pointwise discrepancy.
scale = 1 + abs(Y);
delta = abs(Y -Z)./scale;

% Scalar interval error.
err = max(delta,[],'all');

end
```

MATLAB-Like Refinement Decision

```
function action = chooseRefinement(delta,tol,Nk,Nmax,rhoCrit)

viol = max(delta,[],2) > tol;
rho = nnz(viol)/numel(viol);

if ~any(viol)
    action.type = 'none';
    return
end

[~,jmax] = max(max(delta,[],2));

if rho <= rhoCrit
    action.type = 'h';
    action.splitIndex = jmax;
elseif Nk < Nmax
    action.type = 'p';
```

```
    action.newDegree = Nk + 2;
else
    action.type = 'h';
    action.splitIndex = jmax;
end

end
```

Safeguards for Mesh Splitting

A split point should not be too close to an endpoint.
Define

$$s_{\text{candidate}} = \widetilde{s}_{j^\star}^{(k)}.$$

Choose a minimum fractional distance η, such as

$$\eta = 0.1.$$

Then enforce

$$s_{\text{new}} \in \left[s_{k-1} + \eta \Delta s_k, s_k - \eta \Delta s_k \right].$$

This avoids creating extremely small intervals.

Stopping Criteria

The refinement loop may stop when:

1. every interval satisfies the error tolerance;
2. the maximum number of refinement iterations is reached;
3. the maximum number of nodes is reached;
4. the NLP fails repeatedly;
5. the objective and state trajectories no longer change materially.

A strong convergence test includes both error and solution stabilization:

$$\begin{aligned} \max_k e_k &\le \epsilon, \\ \frac{|J^{(r)} - J^{(r-1)}|}{1 + |J^{(r)}|} &\le \epsilon_J. \end{aligned}$$

Possible Failure Modes

Inaccurate Control Reconstruction

Piecewise-linear control interpolation may be poor near rapid changes or discontinuities.

Possible remedies:

- use one-sided interpolation near switches;
- place mesh points at detected discontinuities;
- use shape-preserving interpolation;
- refine before estimating.

Error Cancellation

The interpolated and integrated states may agree by coincidence.
Use multiple checks, including:

- independent forward integration;
- defect residuals on a dense grid;
- objective convergence;
- polynomial coefficient decay.

Repeated Refinement without Convergence

Possible causes include:

- incorrect dynamics;
- poor variable scaling;
- inconsistent boundary conditions;
- an under-resolved discontinuity;
- inaccurate derivatives;
- a defective NLP solution.

Practical Recommendations

1. Start with a modest number of intervals.
2. Use low-to-moderate polynomial degree.
3. Set a reasonable maximum degree.
4. Normalize state errors using physical scales.
5. Prefer h-refinement when nonsmoothness is suspected.
6. Prefer p-refinement only in clearly smooth intervals.
7. Warm start every refined solve.
8. Verify the final solution independently.
9. Monitor both accuracy and NLP size.
10. Retain the full refinement history.

Refinement History Table

A useful refinement log contains:

Iteration	Intervals	Total Nodes	Max Error	Objective	NLP Status
0	K_0	N_0	e_0	J_0	Converged
1	K_1	N_1	e_1	J_1	Converged
2	K_2	N_2	e_2	J_2	Converged

This table helps identify stagnation, excessive node growth, and objective instability.

Summary

Adaptive mesh refinement converts a fixed-mesh pseudospectral method into an accuracy-controlled computational procedure.

The principal steps are:

1. solve the NLP on a current mesh;
2. create an enriched LGR grid;
3. interpolate the state to that grid;
4. reconstruct the control;
5. integrate the dynamics on the enriched grid;
6. compare interpolated and integrated state values;
7. accept accurate intervals;
8. apply h-refinement to localized errors;
9. apply p-refinement to distributed smooth errors;
10. enforce a maximum polynomial degree;
11. interpolate the old solution onto the new mesh;
12. repeat until convergence.

Pure h methods are robust but may create large NLPs. Pure p methods are efficient for smooth solutions but perform poorly near discontinuities. An hp method seeks to combine robustness with compactness by placing resolution only where it is needed.

Exercises

1. **Global and local time transformations**

   The physical time interval is

   $$t \in [t_0, t_f],$$

   and the global transformed coordinate is

   $$s \in [-1, 1].$$

   The transformed domain is divided into K mesh intervals,

   $$-1 = s_0 < s_1 < \cdots < s_K = 1.$$

   For interval k, introduce a local coordinate

   $$\tau \in [-1, 1]$$

   that maps to

   $$s \in [s_{k-1}, s_k].$$

   (a) Derive the global affine transformations $s(t)$ and $t(s)$.
   (b) Derive the local affine transformations $s^{(k)}(\tau)$ and $\tau^{(k)}(s)$.
   (c) Derive the combined map $t^{(k)}(\tau)$.
   (d) Show that

   $$\frac{\mathrm{d}t}{\mathrm{d}\tau} = \frac{t_f - t_0}{4}\left(s_k - s_{k-1}\right).$$

   (e) Derive

   $$\frac{\partial t^{(k)}(\tau)}{\partial t_0}, \qquad \frac{\partial t^{(k)}(\tau)}{\partial t_f}.$$

   (f) For fixed mesh points, verify that these two derivatives sum to one.

2. **Local multiple-interval LGR transcription**

   On interval k, let

   $$\tau_1^{(k)}, \ldots, \tau_{N_k}^{(k)}$$

   be the left-LGR collocation points, and let

$$\tau_{N_k+1}^{(k)} = 1$$

be the noncollocated terminal state point.

Consider

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, t).$$

(a) Write the local state interpolation polynomial.

(b) Construct the local differentiation matrix

$$\boldsymbol{D}^{(k)} \in \mathbb{R}^{N_k \times (N_k+1)}.$$

(c) Derive the local collocation equations

$$\boldsymbol{D}^{(k)} \boldsymbol{X}^{(k)} = \frac{t_f - t_0}{4} (s_k - s_{k-1}) \boldsymbol{F}^{(k)}.$$

(d) State where the control variables are defined.

(e) Explain why the right endpoint of interval k has a state value but no control value belonging to interval k.

(f) Derive the state-continuity condition between intervals k and $k+1$.

(g) Compare explicit continuity constraints with the shared-interface-state formulation.

3. **Counting variables and points on a nonuniform mesh**

Consider

$$K = 4, \qquad N_1 = 3, \quad N_2 = 5, \quad N_3 = 4, \quad N_4 = 6.$$

(a) Compute the total number of collocation points.

(b) Compute the total number of unique state points when shared interface variables are used.

(c) Compute the total number of control points.

(d) For n_x states and n_u controls, derive the number of scalar state and control decision variables.

(e) Add n_p static parameters and free t_0, t_f, and derive the complete decision-vector dimension.

(f) Determine the number of scalar defect equations.

(g) If separate interface-state variables are retained, determine the number of additional state variables and continuity equations.

(h) Verify that the two interface formulations have the same number of degrees of freedom.

4. **Global assembly of the multiple-interval differentiation matrix**

Consider two mesh intervals with polynomial degrees N_1 and N_2.

(a) Write the two local matrices

$$\boldsymbol{D}^{(1)} \in \mathbb{R}^{N_1 \times (N_1+1)}, \qquad \boldsymbol{D}^{(2)} \in \mathbb{R}^{N_2 \times (N_2+1)}.$$

(b) Assemble the global differentiation matrix using a shared state at the interface.

(c) Show that its dimensions are

$$(N_1 + N_2) \times (N_1 + N_2 + 1).$$

(d) Identify the column shared by the two local blocks.

(e) Generalize the construction to K intervals.

(f) Show that, with

$$N = \sum_{k=1}^{K} N_k,$$

the global matrix satisfies

$$\boldsymbol{D} \in \mathbb{R}^{N \times (N+1)}.$$

(g) Explain why the matrix remains sparse as the number of intervals increases.

(h) Derive the global collocation equations in compact matrix form.

5. **Composite quadrature and endpoint-time derivatives**

Consider the Bolza objective

$$J = \Phi\left(\boldsymbol{x}(t_0), t_0, \boldsymbol{x}(t_f), t_f\right) + \int_{t_0}^{t_f} L\left(\boldsymbol{x}, \boldsymbol{u}, t\right) \, \mathrm{d}t.$$

For interval k, let $w_i^{(k)}$ be the LGR quadrature weights.

(a) Derive the composite quadrature approximation

$$J_h = \Phi + \frac{t_f - t_0}{4} \sum_{k=1}^{K} (s_k - s_{k-1}) \sum_{i=1}^{N_k} w_i^{(k)} L_i^{(k)}.$$

(b) Show that each local quadrature contribution is scaled by the physical duration of its mesh interval.

(c) Derive the nodal physical-time vector.

(d) Derive the derivatives of every nodal time with respect to t_0 and t_f.

(e) Derive $\partial J_h / \partial t_f$ when L depends explicitly on time.

(f) Derive $\partial J_h / \partial t_0$.

(g) Identify the terms arising from the scale factor, the Mayer term, and the explicit time dependence of L.

6. **Global NLP structure and Jacobian sparsity**

Consider a multiple-interval LGR transcription with N total collocation points, n_x states, n_u controls, and n_b endpoint equality constraints.

(a) Define a complete global decision vector.

(b) Write the global defect vector.

(c) Add the endpoint constraints.

(d) Add a path-constraint vector imposed at all collocation points.

(e) Derive the number of scalar defect equations.

(f) Describe which state and control blocks can appear in one local defect row.

(g) Explain why defects from nonadjacent mesh intervals have zero direct derivatives with respect to one another's local variables.

(h) Sketch the block sparsity pattern of the equality-constraint Jacobian.

(i) Explain how free t_0, free t_f, and static parameters create globally coupled columns without destroying the predominantly sparse structure.

7. **Interpolation-and-integration error estimator**

On mesh interval k, the original LGR approximation uses N_k collocation points. Construct an enriched LGR grid with

$$M_k = N_k + 1$$

collocation points.

(a) Construct the state interpolation matrix $\boldsymbol{I}^{(k)}$.

(b) Derive the interpolation-based enriched state estimate

$$\boldsymbol{Y}^{(k)} = \boldsymbol{I}^{(k)} \boldsymbol{X}^{(k)}.$$

(c) Explain why the state reconstruction is unique but the control reconstruction is not.

(d) Define a piecewise-linear reconstruction of the control on the enriched grid.

(e) Evaluate the dynamics on the enriched grid.

(f) Using the enriched integration matrix $\boldsymbol{B}^{(k)}$, derive the dynamics-based estimate

$$\boldsymbol{Z}^{(k)}_{2:M_k+1} = \mathbf{1} \left(\boldsymbol{Y}^{(k)}_1 \right)^{\mathsf{T}} + \frac{t_f - t_0}{4} \left(s_k - s_{k-1} \right) \boldsymbol{B}^{(k)} \boldsymbol{F}^{e(k)}.$$

(g) Define a pointwise absolute error.

(h) Define an interval-wise relative error indicator e_k.

(i) State the mesh-acceptance condition for a prescribed tolerance ϵ.

8. **Choosing between h- and p-refinement**

Let

$$\delta^{(k)}_j$$

denote the pointwise error samples on interval k, and let e_k be the maximum interval error.

(a) Define a localized-error metric based on the fraction of total error concentrated near the largest-error point.

(b) Explain why localized error suggests a switch, boundary layer, discontinuity, or sharp transition.

(c) Explain why distributed error on a smooth interval suggests increasing polynomial degree.

(d) Formulate a decision rule that selects h-refinement for localized error and p-refinement for distributed error.

(e) Modify the rule to enforce a maximum polynomial degree $N_{\max}$.

(f) State safeguards that prevent creation of excessively short mesh intervals.

(g) Explain why pure p-refinement may perform poorly near a bang–bang switching point.

(h) Explain why pure h-refinement may be unnecessarily expensive in a smooth region.

9. **Warm starting after mesh refinement**

Suppose interval k is refined in one of the following ways:

$$p\text{-refinement:} \qquad N_k \to N_k + \Delta N,$$

or

$$h\text{-refinement:} \qquad [s_{k-1}, s_k] \to [s_{k-1}, s_m] \cup [s_m, s_k].$$

(a) Derive how the old state polynomial is evaluated at the new state nodes for p-refinement.

(b) Derive how it is evaluated on both child intervals for h-refinement.

(c) Describe how to initialize new control values.

(d) Show how shared interface-state values should be assigned after an interval split.

(e) Explain how static parameters and endpoint times are transferred.

(f) State why warm starting normally reduces the number of NLP iterations.

(g) Identify circumstances in which interpolation of the previous solution may provide a poor warm start.

10. **Complete adaptive hp-refinement algorithm**

Formulate a complete adaptive solution procedure for a multiple-interval LGR transcription.

Your algorithm must include:

(a) construction of an initial mesh and polynomial-degree vector;

(b) solution of the NLP;

(c) reconstruction of state and control trajectories;

(d) construction of enriched grids;

(e) computation of interval error indicators;

(f) acceptance of intervals satisfying

$$e_k \le \epsilon;$$

(g) selection of h- or p-refinement for rejected intervals;

(h) enforcement of $N_{\max}$, minimum interval length, and maximum refinement iterations;

(i) warm starting of the refined NLP;

(j) stopping criteria based on error, feasibility, and stagnation;

(k) a refinement-history table containing iteration number, number of intervals, total collocation points, maximum error, objective value, and NLP status;

(l) verification by independent re-simulation and dense-grid checking of path constraints.

Finally, explain why over-refinement can increase computational cost and conditioning difficulty without materially improving the engineering conclusion.