COURSE NOTES

Control Co-Design

Compiled on Tuesday 21st July, 2026
Last repo commit: July 19, 2026 at 11:44 PM EDT (1c8a4e3)

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

About These Notes

Control Co-Design: Integrated Optimization of Plants and Controllers

by **Saeid Bayat** (saeidb@umich.edu; saeidbayat.com)

Google Chrome Recommended

For the best experience, please view these notes using **Google Chrome** (or another Chromium-based browser, like Arc). You may experience slower scrolling and performance issues if using Safari.

Course Under Development

These course notes are actively under development. Content, examples, activities, and figures may be revised as the course evolves. If you notice an error or have a suggestion, please contact me at saeidb@umich.edu.

Overview

Control co-design integrates decisions about the plant, controller, sensing, actuation, information, and system architecture. **These notes develop the theory and computational methods needed to design dynamic engineering systems as coordinated wholes rather than as a sequence of disciplinary handoffs.** Practical examples in Python and MATLAB connect the mathematical formulations to their engineering interpretation.

Course Roadmap

The nine chapters build from mathematical foundations to complete control co-design applications:

- **Chapter 0 — Mathematical Prerequisites:** Reviews the notation, calculus, linear algebra, differential equations, optimization language, discretization, and computational tools used throughout control co-design.
- **Chapter 1 — Introduction to Control Co-Design:** Introduces active systems, sequential and integrated design, plant–control coupling, representative applications, and responsible engineering practice.
- **Chapter 2 — Dynamic Systems and Feedback Control:** Develops dynamic-system models, state-space representations, feedback, stability, and the ways plant design changes closed-loop behavior.
- **Chapter 3 — Engineering Design Optimization:** Covers problem formulation, feasible design spaces, gradient-based and constrained methods, scaling, multidisciplinary coupling, and dynamic-system optimization.
- **Chapter 4 — Mathematical Formulation of Control Co-Design:** Combines plant and controller variables, trajectories, dynamics, objectives, path constraints, and boundary constraints in a common CCD formulation.
- **Chapter 5 — Control Co-Design Architectures and Coordination:** Compares single-pass, iterative sequential, nested, and simultaneous architectures and offers practical guidance for coordinating plant and controller optimization.
- **Chapter 6 — Optimal Control and Closed-Loop Control Co-Design:** Connects open-loop optimal control to implementable closed-loop designs through controller parameterization, realistic information sets, and model predictive control.

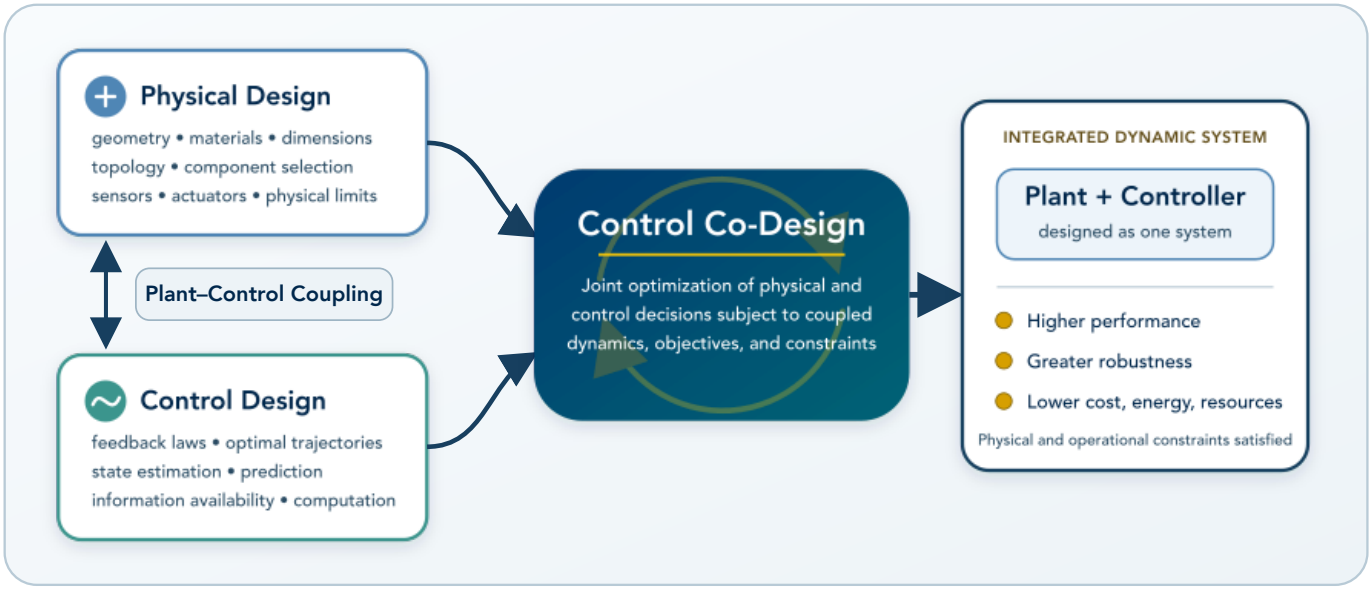


Figure 1: Physical and control design decisions influence one another through plant-control coupling and are jointly optimized to create an integrated dynamic system with higher performance, greater robustness, and lower resource use.

- **Chapter 7 — Numerical Methods for Control Co-Design:** Develops shooting, direct transcription, collocation, sparse derivatives, mesh refinement, and independent verification.
- **Chapter 8 — Practical CCD Studies and Advanced Topics:** Brings the course together through active-suspension and offshore-energy studies, uncertainty-aware design, multi-fidelity modeling, hardware implementation, reproducibility, and validation.

Prerequisites

Readers should be comfortable with differential equations, linear algebra, introductory control systems, numerical methods, and basic optimization. Examples are provided in both Python and MATLAB.

Why Write These Notes?

Control co-design sits at the intersection of dynamics, control, optimization, and physical-system design. Excellent resources exist in each area, but students are often left to assemble the connections for themselves. I wrote these notes to present those connections as one coherent story in which the physical system and its controller are designed as parts of a single engineered system.

My teaching begins with engineering questions rather than isolated techniques. Each chapter develops the mathematics needed to investigate those questions and then returns to examples where students can interpret the results physically. The goal is not merely to find an optimum, but to understand why the design changed, which assumptions produced the improvement, and whether that improvement is credible.

I treat these notes as an evolving teaching laboratory. Reproducible examples make abstract ideas inspectable, visual explanations support different ways of learning, and open questions encourage students to move from applying established methods toward research. My aim is to create a resource that is rigorous, intuitive, and useful when students begin designing real systems.

Acknowledgements and Feedback

I am deeply grateful to my PhD advisor, **Prof. James T. Allison**, Professor and Jerry S. Dobrovolny Faculty Scholar at the University of Illinois Urbana-Champaign. I learned the foundations of control co-design—and much of how I think about integrated engineering-system design—from him.

I also thank my postdoctoral advisor, **Prof. Lei Zuo**, Herbert C. Sadler Collegiate Professor of Engineering at the University of Michigan. Through his mentorship, I learned how control co-design can be applied to wave energy converters and other marine-energy systems.

I am grateful to **Prof. Daniel R. Herber**, Associate Professor of Systems Engineering at Colorado State University. His papers and openly available code have helped me understand and apply many important control co-design concepts.

Finally, I thank **Suraj Rampure**, teaching faculty in Computer Science and Engineering at the University of Michigan. At the University of Michigan's *Engineering Education Innovation Days* workshop, Suraj demonstrated how he uses Jupyter Book to create engaging and accessible course notes. His session inspired me to adopt this approach, and his course template provided a thoughtful starting point for organizing and presenting this book.

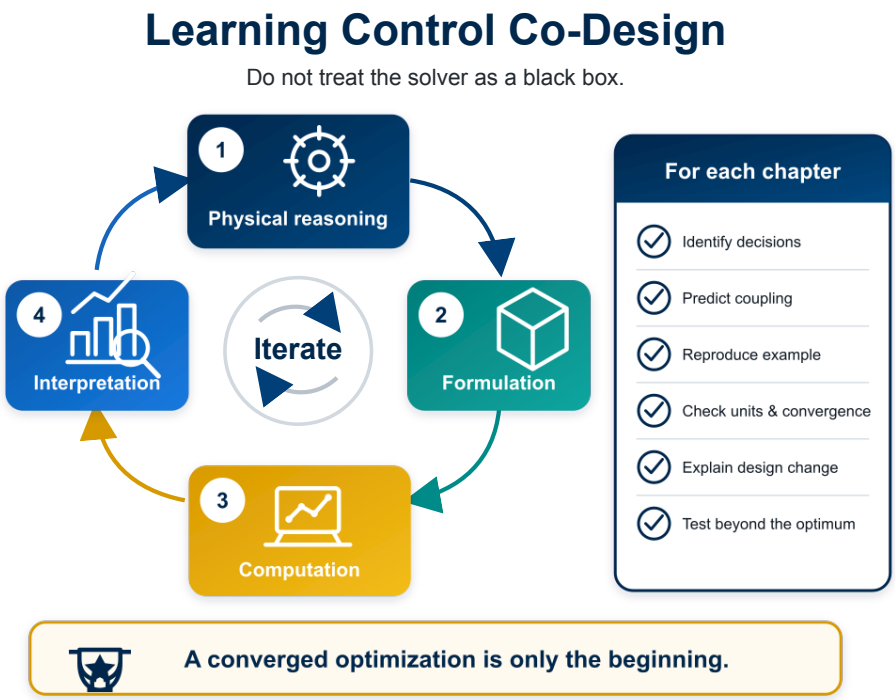
AI-assisted tools have been used selectively during drafting and visual development. I review and revise all published material and remain responsible for its accuracy, accessibility, and pedagogical quality.

If you have feedback or suggestions, please contact me at `saeidb@umich.edu`. I would be glad to hear from students, educators, researchers, and other readers who find these notes useful.

Accessibility

I want all students to be able to engage fully with these notes. The material uses text, equations, diagrams, and computational examples to provide multiple paths to understanding, and I aim to follow WCAG 2.1 AA practices. If you encounter an accessibility barrier, please contact me at `saeidb@umich.edu` so I can improve the resource.

Advice to Students



Control co-design is learned by moving repeatedly between physical reasoning, mathematical formulation, computation, and design interpretation. Do not treat the optimization solver as a black box.

For each chapter:

1. identify the physical and control decisions before reading the formulation;
2. predict the important coupling mechanisms;
3. reproduce the example in Python or MATLAB;
4. check units, constraints, scaling, and numerical convergence;
5. explain why the optimized design changed; and
6. test the result under conditions that were not optimized.

The activities and exercises are part of the course narrative. A converged optimization is only the beginning of the engineering argument.

Mathematical Prerequisites

Chapter 0: Mathematical Prerequisites

Mathematical and computational foundations for control co-design

Control co-design (CCD) combines plant design and controller design in one integrated problem. Reading, formulating, and solving CCD problems requires a modest but reliable toolkit: mathematical notation, calculus, linear algebra, differential equations, optimization language, time discretization, and basic computational workflows.

This prerequisite chapter does **not** replace a full course in calculus, linear algebra, dynamics, optimization, or programming. Instead, it provides a focused review of the concepts used throughout the rest of these notes, always from the viewpoint of their later role in CCD.

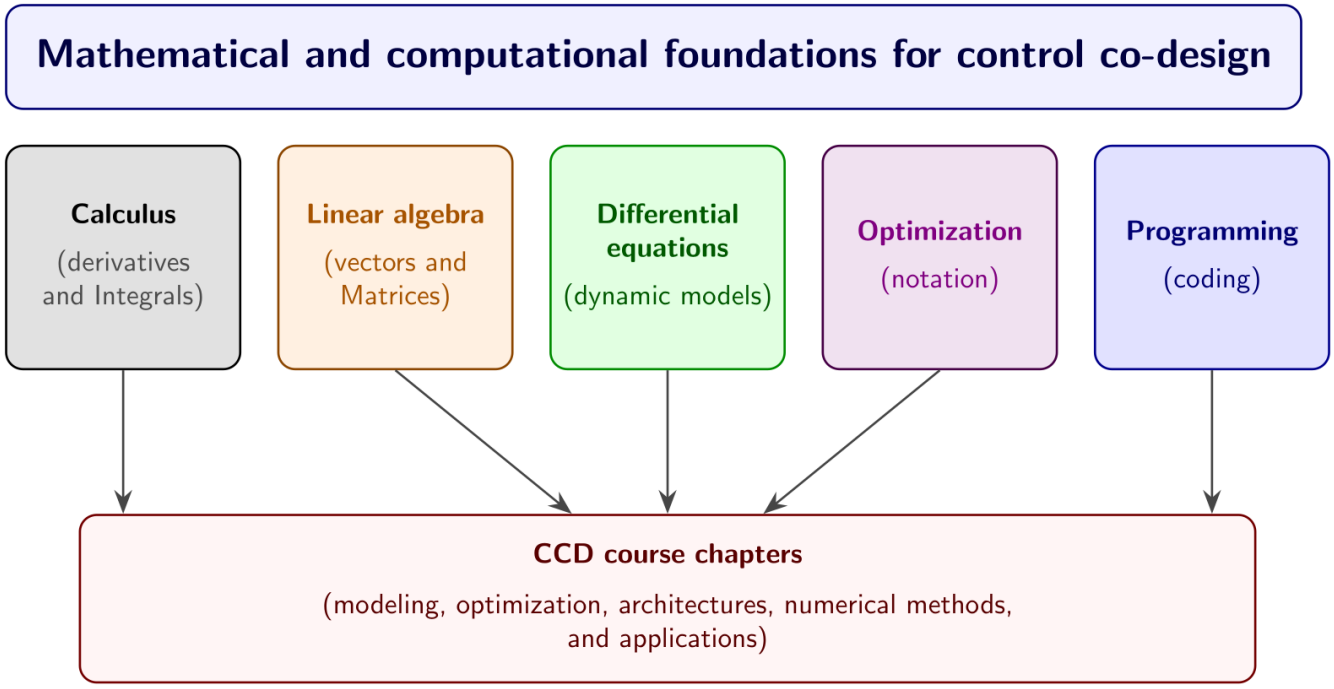


Figure 1: *
The mathematical and computational foundations that support modeling, optimization, numerical methods, and applications in the remainder of the course.

Learning objectives

After completing this chapter, you should be able to:

1. read and use the notation that appears throughout the course;
2. interpret derivatives, partial derivatives, gradients, Jacobians, and Hessians at an introductory level;
3. interpret definite integrals and basic numerical-integration formulas;
4. perform basic vector and matrix operations relevant to dynamic-system modeling;
5. convert a second-order differential equation into first-order state-space form;
6. read and write a basic optimization problem with variables, objectives, and constraints;
7. explain how continuous trajectories become finite-dimensional decision vectors; and
8. understand the minimal programming workflow used in simulation and optimization studies.

Chapter map

The chapter is organized as six focused sections:

- **Section 0.1 — Mathematical Notation and Engineering Functions:** Introduces scalars, vectors, matrices, trajectories, common CCD variables, indexing, summation, bounds, and functional relationships.
- **Section 0.2 — Calculus and Linear Algebra:** Reviews derivatives, gradients, Jacobians, Hessians, integrals, numerical integration, vectors, matrices, norms, eigenvalues, and positive definiteness.
- **Section 0.3 — Differential Equations and State-Space Models:** Converts a mass–spring–damper equation into first-order form and introduces nonlinear and linear state-space notation.
- **Section 0.4 — Optimization and Discretization:** Identifies the parts of an optimization problem and explains how continuous trajectories become finite decision vectors.
- **Section 0.5 — Computational Preparation and Readiness Project:** Presents basic Python and MATLAB simulation workflows and a small preparation exercise.
- **Section 0.6 — Summary and Problems:** Consolidates the chapter vocabulary and provides review problems across all prerequisite topics.

Readiness goal

If you can read vectors and matrices, compute simple derivatives and integrals, convert a second-order ODE to first-order form, identify the parts of an optimization problem, and follow a short simulation script, you are ready for the main body of the course.

0.1 Mathematical Notation and Engineering Functions

Control co-design formulations can look intimidating because they express many related ideas compactly. Becoming comfortable with the notation makes the modeling and optimization concepts in later chapters much easier to follow.

Mathematical notation and engineering variables

Scalars, vectors, matrices, and trajectories. These notes use the following conventions:

- Italic lowercase letters such as m, k, and t denote **scalars**.
- Bold lowercase letters such as $\mathbf{x}$ and $\mathbf{u}$ denote **vectors**.
- Uppercase letters such as A, B, and C denote **matrices**.
- Time-varying quantities are written explicitly, such as $x(t)$, $\mathbf{x}(t)$, or $u(t)$; these functions of time are **trajectories**.

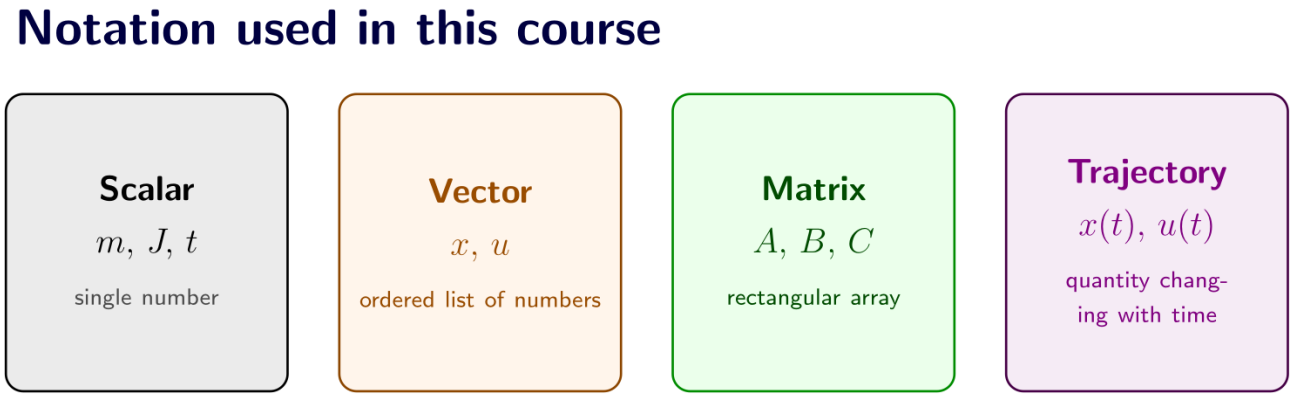


Figure 2: *
The main notation categories used throughout the course.

Common variables in control co-design. The following symbols appear frequently in later chapters:

Symbol	Meaning
$\mathbf{x}_p$	Plant design variables
$\mathbf{x}_c$	Control design variables or controller parameters
$\mathbf{x}(t)$	State trajectory
$\mathbf{u}(t)$	Control-input trajectory
$\mathbf{y}(t)$	Output trajectory or measured signals
J	Objective function or performance index
$g(\cdot)$	Inequality constraints
$h(\cdot)$	Equality constraints

For example, a plant-variable vector might be

$$\mathbf{x}_p = \begin{bmatrix} m & k & c \end{bmatrix}^T,$$

while a controller-parameter vector might be

$$\mathbf{x}_c = \begin{bmatrix} K_p & K_i & K_d \end{bmatrix}^T.$$

Subscripts and superscripts. Subscripts usually identify components, samples, or categories. For example,

$$\mathbf{x} = \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix}^T$$

means that $\mathbf{x}$ contains n components. In a time-discretized problem, x_i may instead mean the value of a variable at the ith time node; the surrounding context determines the intended meaning.

Superscripts may indicate powers, algorithm iterations, or labels. The notation

$$\mathbf{x}^{(k)}$$

usually denotes the value of $\mathbf{x}$ at iteration k of an algorithm, whereas x^2 denotes the square of a scalar.

Summation notation. Summation notation is common in discrete approximations and finite-dimensional optimization:

$$\sum_{i=1}^{N} a_i = a_1 + a_2 + \cdots + a_N.$$

If a scalar control input is sampled at N points, a simple discrete measure of control effort is

$$J = \sum_{i=1}^{N} u_i^2.$$

The index i can identify time nodes, operating conditions, load cases, uncertain scenarios, or experimental trials.

Sets, bounds, and intervals. Variable bounds are often written componentwise:

$$\mathbf{x}^L \le \mathbf{x} \le \mathbf{x}^U,$$

where $\mathbf{x}^L$ and $\mathbf{x}^U$ are lower and upper bounds. The notation $t \in [t_0, t_f]$ means that time belongs to the closed interval from t_0 to t_f.

Functions and engineering relationships

A function maps one quantity to another. In engineering, functions describe relationships among design variables, states, inputs, outputs, objectives, and constraints.

Single-variable and multivariable functions. A single-variable function has the form

$$y = f(x),$$

whereas a multivariable function may have the form

$$J = f(x_1, x_2, \ldots, x_n).$$

For example,

$$J(k, c) = k^2 + 2c^2 - kc$$

depends on two design variables: stiffness k and damping c.

Explicit and implicit relationships. An **explicit** relationship directly gives one variable as a function of others, such as

$$y = 3x + 2.$$

An **implicit** relationship combines variables in one equation, such as

$$x^2 + y^2 = 1.$$

Both forms occur in engineering models and constraints.

Linear, affine, and nonlinear relationships. A linear function preserves scaling and addition. A relationship such as $y = ax$ is linear, while

$$y = ax + b$$

is affine because of the constant offset b. A relationship such as

$$y = x^2 + \sin(x)$$

is nonlinear. Most CCD problems are nonlinear because plant and controller decisions interact through nonlinear dynamics, constraints, and performance measures.

Time-dependent functions. Dynamic systems are described using time-dependent functions such as $\mathbf{x}(t)$ and $\mathbf{u}(t)$. A trajectory is not one number: it assigns a value at every time in an interval. Later numerical methods will replace these continuous functions with finite sets of samples or coefficients.

Check your interpretation

Whenever you encounter a symbol, ask whether it represents a scalar, vector, matrix, or trajectory—and whether its subscript identifies a component, a sample, or a design category.

0.2 Calculus and Linear Algebra

Calculus describes rates, sensitivities, and accumulated quantities. Linear algebra provides the language for systems with many states, inputs, outputs, and design variables. Together, they support nearly every CCD formulation and numerical method in this course.

Derivatives and partial derivatives

Single-variable derivatives. The derivative of a scalar function $f(x)$ is written

$$\frac{df}{dx}.$$

It measures the rate at which f changes when x changes. In optimization, a derivative indicates how an objective or constraint responds to a small adjustment in a decision variable.

If

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

then

$$\frac{df}{dx} = 2x.$$

The second derivative,

$$\frac{d^2f}{dx^2},$$

measures how the slope changes and provides information about curvature.

Partial derivatives. When a function depends on several variables, a partial derivative describes its sensitivity to one variable while the others are held fixed:

$$\frac{\partial f}{\partial x_i}.$$

For

$$J(k, c) = k^2 + 2c^2 - kc,$$

the partial derivatives are

$$\frac{\partial J}{\partial k} = 2k - c, \qquad \frac{\partial J}{\partial c} = 4c - k.$$

Gradient. The gradient collects the first partial derivatives of a scalar function into a vector:

$$\nabla J(\mathbf{x}) = \begin{bmatrix} \frac{\partial J}{\partial x_1} \\ \frac{\partial J}{\partial x_2} \\ \vdots \\ \frac{\partial J}{\partial x_n} \end{bmatrix}.$$

For the function $J(k, c)$,

$$\nabla J(k, c) = \begin{bmatrix} 2k - c \\ 4c - k \end{bmatrix}.$$

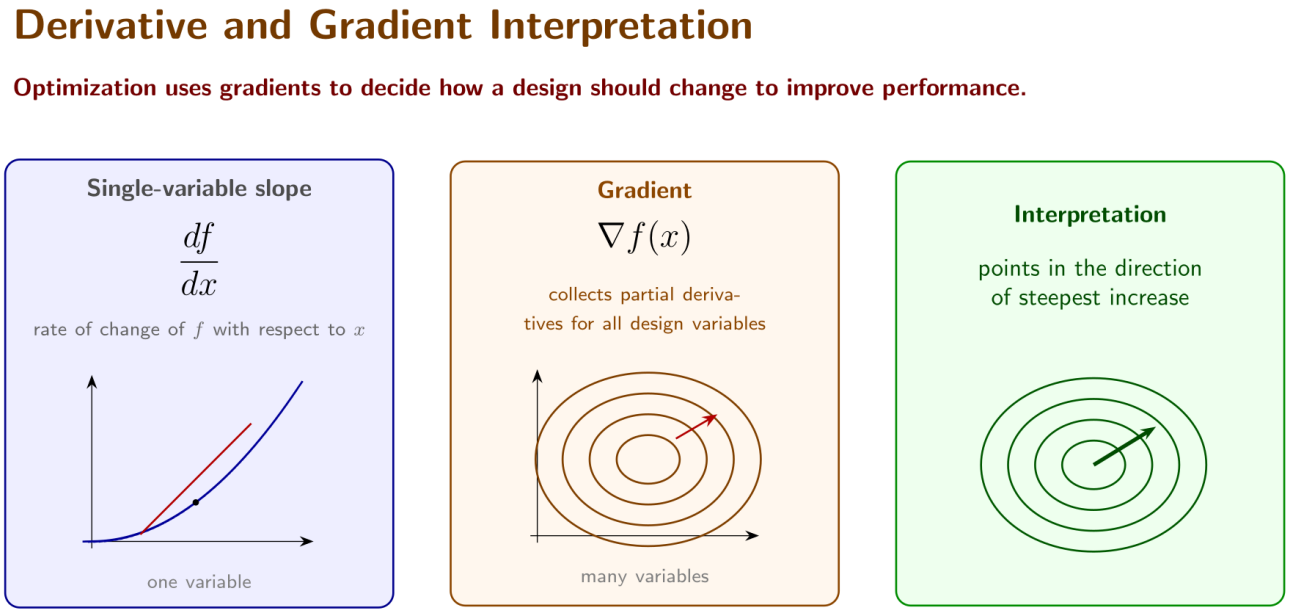


Figure 3: *
Derivatives and gradients provide the change information used by many optimization algorithms.

Chain rule. If one variable depends on another,

$$z = f(y), \qquad y = g(x),$$

the chain rule propagates derivatives through the nested relationship:

$$\frac{dz}{dx} = \frac{dz}{dy}\frac{dy}{dx}.$$

This idea is fundamental to sensitivity analysis and automatic differentiation.

Jacobian and Hessian. For a vector-valued function

$$\mathbf{r}(\mathbf{x}) = \begin{bmatrix} r_1(\mathbf{x}) \\ r_2(\mathbf{x}) \\ \vdots \\ r_m(\mathbf{x}) \end{bmatrix},$$

the **Jacobian** is the matrix of first derivatives:

$$\frac{\partial \mathbf{r}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial r_1}{\partial x_1} & \cdots & \frac{\partial r_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial r_m}{\partial x_1} & \cdots & \frac{\partial r_m}{\partial x_n} \end{bmatrix}.$$

The **Hessian** of a scalar function is the matrix of its second partial derivatives. Jacobians describe the local sensitivity of vector equations; Hessians describe the local curvature of scalar objectives or constraints.

Example: gradient of a design objective. Let

$$J(m, k) = m^2 + 0.5k^2 - mk.$$

Then

$$\frac{\partial J}{\partial m} = 2m - k, \qquad \frac{\partial J}{\partial k} = k - m,$$

so

$$\nabla J(m, k) = \begin{bmatrix} 2m - k \\ k - m \end{bmatrix}.$$

This calculation is the basic language of gradient-based optimization.

Integration and numerical integration

Definite integrals as accumulated quantities. A definite integral accumulates a quantity over an interval:

$$\int_{t_0}^{t_f} q(t)\, dt.$$

In engineering, integrals can represent accumulated energy, work, fuel, tracking error, or control effort. For example,

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

is a common measure of control effort.

Numerical integration. When time is discretized at $t_0, t_1, \ldots, t_N$, a left-rectangle approximation is

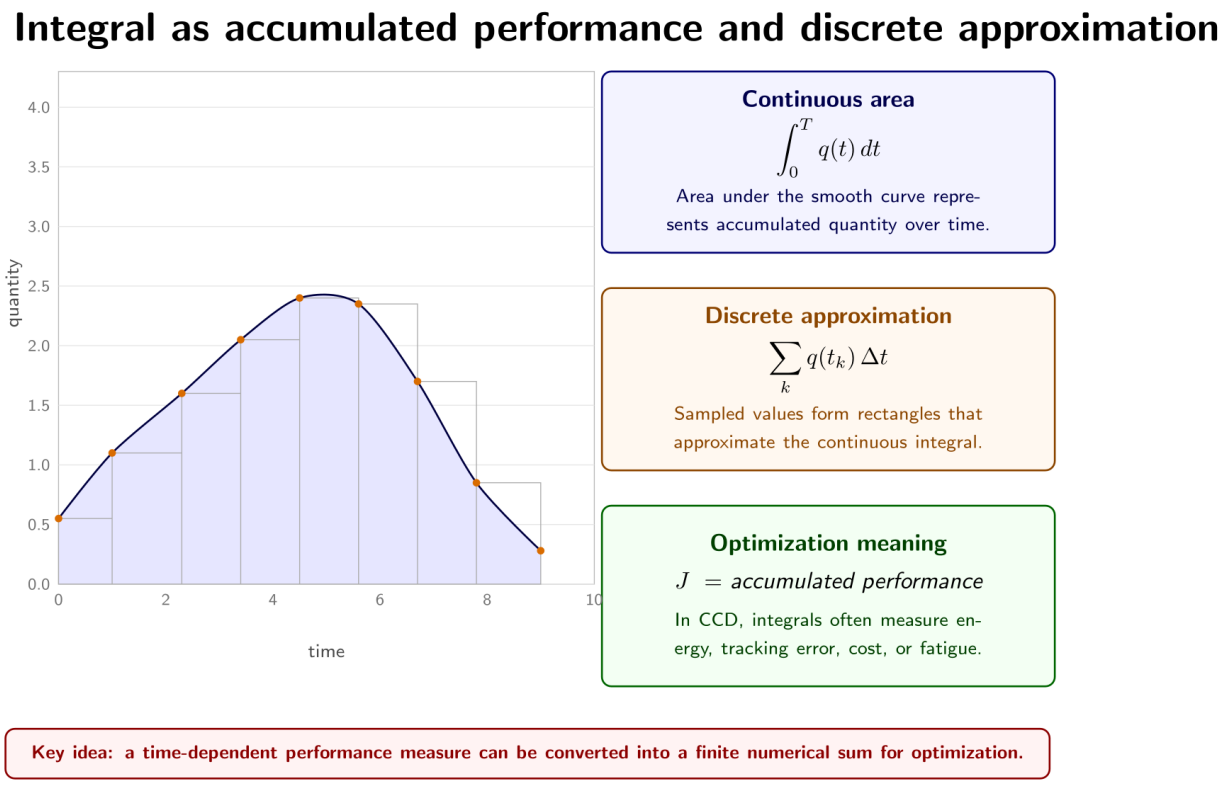


Figure 4: *
A definite integral measures accumulated area; numerical methods approximate it using sampled values.

$$\int_{t_0}^{t_f} q(t)\,dt \approx \sum_{i=0}^{N-1} q_i \Delta t_i,$$

where $q_i \approx q(t_i)$ and $\Delta t_i = t_{i+1} - t_i$.
The trapezoidal rule uses both endpoints of each interval:

$$\int_{t_0}^{t_f} q(t)\,dt \approx \sum_{i=0}^{N-1} \frac{\Delta t_i}{2}(q_i + q_{i+1}).$$

For the control-effort objective,

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

the trapezoidal approximation is

$$J \approx \sum_{i=0}^{N-1} \frac{\Delta t_i}{2}\left(u_i^2 + u_{i+1}^2\right).$$

This form appears later in direct-transcription methods.

Linear algebra

Vectors, matrices, and dimensions. A vector is an ordered list of numbers; a matrix is a rectangular array:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}, \qquad A = \begin{bmatrix} 1 & 0 \\ 2 & 3 \end{bmatrix}.$$

If $A \in \mathbb{R}^{m\times n}$, then A has m rows and n columns. The product $A\mathbf{x}$ is defined only when the number of columns in A equals the number of components in $\mathbf{x}$.

Important operations include vector and matrix addition, scalar multiplication, matrix multiplication, transpose A^T, and—when it exists—the inverse A^{-1}.

Identity matrices and linear systems. The identity matrix satisfies

$$I\mathbf{x} = \mathbf{x}.$$

A linear system has the form

$$A\mathbf{x} = \mathbf{b}.$$

If A is square and nonsingular, the mathematical solution can be written $\mathbf{x} = A^{-1}\mathbf{b}$. In computation, however, one normally uses a linear-system solver rather than forming the inverse explicitly.

Norms. A norm measures vector size. The Euclidean norm is

$$\|\mathbf{x}\|_2 = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}.$$

Norms appear in convergence tests, regularization terms, error measures, and engineering-performance metrics.

Eigenvalues and eigenvectors. An eigenvalue–eigenvector pair satisfies

$$A\mathbf{v} = \lambda\mathbf{v}.$$

In state-space models, eigenvalues help characterize dynamic behavior and stability.

Positive definiteness. A symmetric matrix Q is positive definite if

$$\mathbf{z}^T Q\mathbf{z} > 0 \qquad \text{for every nonzero } \mathbf{z}.$$

Positive-definite matrices occur frequently in quadratic objectives and Lyapunov analysis.

Computational practice

Check matrix dimensions before multiplying arrays, and solve $A\mathbf{x} = \mathbf{b}$ with a numerical linear-system solver instead of explicitly computing A^{-1}.

Activity 0.1: Composite Engineering Functions, Gradients, and Hessians

Activity 0.1: Composite Engineering Functions, Gradients, and Hessians

For a mass-spring-damper system, define

$$\omega_n(m,k) = \sqrt{\frac{k}{m}}, \qquad \zeta(m,k,c) = \frac{c}{2\sqrt{km}}.$$

Consider the dimensionless design objective

$$J(m,k,c) = \left(\frac{\omega_n - \omega_d}{\omega_d}\right)^2 + \beta\left(\frac{\zeta - \zeta_d}{\zeta_d}\right)^2 + \gamma\left(\frac{m}{m_{\text{ref}}}\right)^2,$$

where

$$\omega_d = 4\ \text{rad/s}, \qquad \zeta_d = 0.7, \qquad \beta = 2, \qquad \gamma = 0.05, \qquad m_{\text{ref}} = 2\ \text{kg}.$$

1. Derive

$$\frac{\partial \omega_n}{\partial m}, \qquad \frac{\partial \omega_n}{\partial k}.$$

2. Derive

$$\frac{\partial \zeta}{\partial m}, \qquad \frac{\partial \zeta}{\partial k}, \qquad \frac{\partial \zeta}{\partial c}.$$

3. Use the chain rule to derive the complete gradient $\nabla J(m, k, c)$.
4. Evaluate J and ∇J at

$$m = 2\ \mathrm{kg}, \qquad k = 24\ \mathrm{N/m}, \qquad c = 5\ \mathrm{N\,s/m}.$$

5. Let

$$\mathbf{d} = \frac{1}{\sqrt{6}} \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix}.$$

   Compute the directional derivative

$$D_{\mathbf{d}} J = \nabla J^T \mathbf{d}.$$

6. Derive the 3×3 Hessian matrix $H_J(m, k, c)$.
7. Evaluate the eigenvalues of the Hessian at the stated design and determine whether J is locally convex there.
8. Verify the analytical gradient using central finite differences with step sizes

$$h \in \{10^{-2}, 10^{-4}, 10^{-6}, 10^{-8}\}.$$

Activity 0.2: Numerical Integration and Observed Convergence Order

Activity 0.2: Numerical Integration and Observed Convergence Order

Consider

$$q(t) = e^{-t} \sin(3t), \qquad 0 \le t \le 2,$$

and define

$$I = \int_0^2 q(t)^2 \, dt.$$

1. Evaluate I analytically.
2. Approximate I using:
   (a) the left-rectangle rule;
   (b) the midpoint rule;
   (c) the composite trapezoidal rule; and

(d) composite Simpson's rule.

3. Use

$$N \in \{10, 20, 40, 80, 160\}$$

equal subintervals.

4. For each method, compute the absolute error

$$e_N = |I_N - I|.$$

5. Estimate the observed convergence order using

$$p_N = \frac{\log(e_N/e_{2N})}{\log 2}.$$

6. Compare the observed orders with the theoretical orders of the four methods.
7. Derive the composite trapezoidal approximation in matrix form:

$$I_N = \mathbf{w}^T \mathbf{q}^2,$$

where $\mathbf{w}$ is the quadrature-weight vector.

8. Explain why a highly accurate state trajectory does not automatically guarantee a highly accurate integral performance measure if the quadrature rule is inconsistent with the trajectory approximation.

Activity 0.3: Linear Algebra, Modal Coordinates, and Positive Definiteness

Activity 0.3: Linear Algebra, Modal Coordinates, and Positive Definiteness

Consider the undamped two-degree-of-freedom mechanical system

$$M\ddot{\mathbf{q}} + K\mathbf{q} = \mathbf{0},$$

where

$$M = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}, \qquad K = \begin{bmatrix} 6 & -2 \\ -2 & 4 \end{bmatrix}.$$

1. Verify that M and K are symmetric positive definite.
2. Solve the generalized eigenvalue problem

$$K\boldsymbol{\phi} = \omega^2 M\boldsymbol{\phi}.$$

3. Compute the two natural frequencies.

4. Normalize the mode shapes so that

$$\Phi^T M \Phi = I.$$

5. Verify that

$$\Phi^T K \Phi = \mathrm{diag}(\omega_1^2, \omega_2^2).$$

6. Introduce modal coordinates $\mathbf{q} = \Phi \boldsymbol{\eta}$ and derive the decoupled equations of motion.
7. For

$$\mathbf{q}(0) = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \qquad \dot{\mathbf{q}}(0) = \begin{bmatrix} 0 \\ 0 \end{bmatrix},$$

compute the initial modal coordinates.

8. Derive the exact displacement response $\mathbf{q}(t)$.
9. Compute the total mechanical energy

$$E(t) = \frac{1}{2} \dot{\mathbf{q}}^T M \dot{\mathbf{q}} + \frac{1}{2} \mathbf{q}^T K \mathbf{q}$$

and prove that it is constant.

10. Explain why positive definiteness of M and K matters physically and mathematically.

0.3 Differential Equations and State-Space Models

Dynamic systems are modeled with differential equations because their rates of change depend on current states, inputs, time, and physical parameters. CCD uses these equations both to predict system behavior and to constrain the optimization problem.

Ordinary differential equations

What is a differential equation?. A differential equation relates a function to one or more of its derivatives. A first-order ordinary differential equation (ODE) can be written

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

while a second-order ODE can be written

$$\ddot{x} = g(x, \dot{x}, u, t).$$

The derivative order refers to the highest derivative appearing in the equation.

Mass–spring–damper example. A standard engineering model is

$$m\ddot{x} + c\dot{x} + kx = u(t),$$

where

- m is mass,
- c is the damping coefficient,
- k is spring stiffness, and
- $u(t)$ is an external force or control input.

In a CCD problem, m, c, and k could be plant design variables, while $u(t)$ or the parameters of a feedback law could be control design variables.

Initial conditions. The differential equation alone does not determine a unique trajectory. A second-order initial-value problem normally also requires displacement and velocity at the initial time:

$$x(0) = x_0, \qquad \dot{x}(0) = v_0.$$

Converting a higher-order ODE to first-order form. First-order state-space form is central to simulation, control, and optimal control. Define

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

The first state equation is

$$\dot{x}_1 = x_2.$$

Solving the original mass–spring–damper equation for acceleration gives

$$\ddot{x} = -\frac{k}{m}x - \frac{c}{m}\dot{x} + \frac{1}{m}u,$$

so the second state equation is

$$\dot{x}_2 = -\frac{k}{m}x_1 - \frac{c}{m}x_2 + \frac{1}{m}u.$$

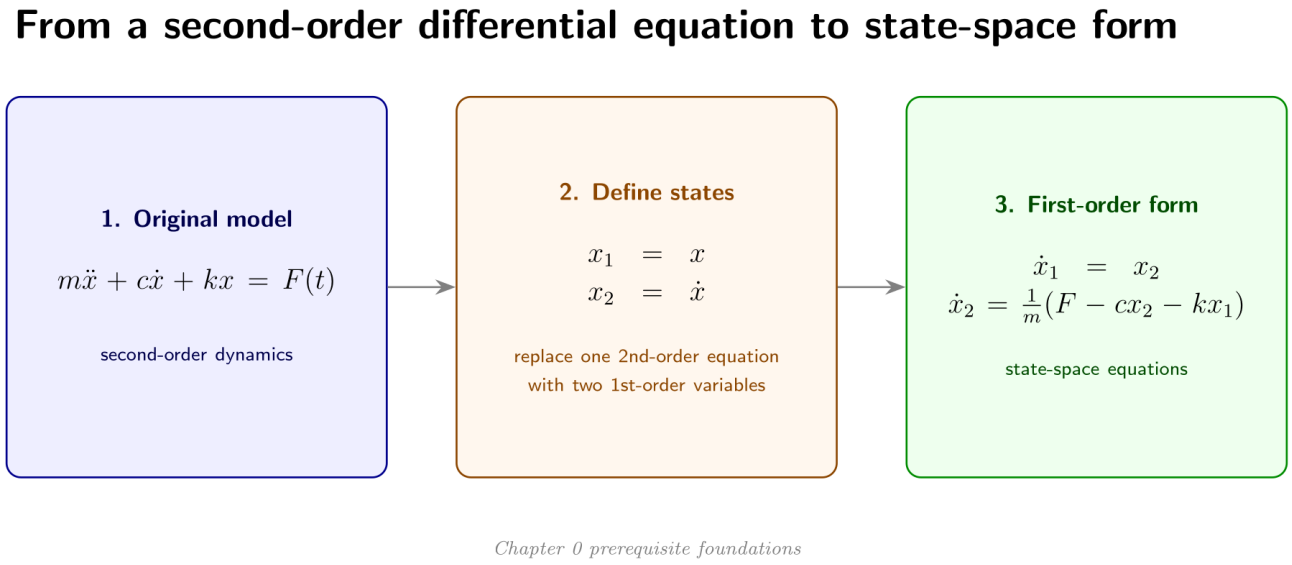


Figure 5: *
Second-order engineering models are commonly rewritten as systems of first-order state equations.

Example: state conversion. Collect the two states in a vector:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} x \\ \dot{x} \end{bmatrix}.$$

The dynamics can then be written compactly as

$$\dot{\mathbf{x}} = \begin{bmatrix} x_2 \\ -\frac{k}{m}x_1 - \frac{c}{m}x_2 + \frac{1}{m}u \end{bmatrix}.$$

This representation reappears in Chapter 2 and in later optimal-control formulations.

Introductory state-space notation

Nonlinear state-space form. A general nonlinear dynamic system is often written

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, t), \qquad \mathbf{y} = \mathbf{h}(\mathbf{x}, \mathbf{u}, t),$$

where

- $\mathbf{x}$ is the state vector,
- $\mathbf{u}$ is the input vector,
- $\mathbf{y}$ is the output vector,
- $\mathbf{f}$ defines the state dynamics, and
- $\mathbf{h}$ maps states and inputs to outputs.

Linear state-space form. For a linear time-invariant system,

$$\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u}, \qquad \mathbf{y} = C\mathbf{x} + D\mathbf{u}.$$

The matrices have compatible dimensions:

Matrix	Role
A	State or system matrix
B	Input matrix
C	Output matrix
D	Direct-feedthrough matrix

State-space notation is compact, supports multiple states and inputs, and works naturally with vector and matrix operations. It is therefore widely used in control theory and CCD.

Key connection

The plant parameters appear inside the dynamic equations. Changing a plant variable therefore changes the state-space model, the closed-loop response, and often the best controller.

Activity 0.4: Exact State-Space Response of a Forced Dynamic System

Activity 0.4: Exact State-Space Response of a Forced Dynamic System

Consider

$$m\ddot{x} + c\dot{x} + kx = u_0,$$

where u_0 is a constant input. Use

$$m = 1.5\ \text{kg}, \qquad c = 1.8\ \text{N s/m}, \qquad k = 12\ \text{N/m},$$
$$u_0 = 3\ \text{N}, \qquad x(0) = 0.2\ \text{m}, \qquad \dot{x}(0) = -0.1\ \text{m/s}.$$

1. Define

$$\mathbf{x} = \begin{bmatrix} x \\ \dot{x} \end{bmatrix}$$

and derive

$$\dot{\mathbf{x}} = A\mathbf{x} + Bu_0.$$

2. Compute the eigenvalues of A and classify the response as underdamped, critically damped, or overdamped.
3. Compute the equilibrium state $\mathbf{x}_{\text{eq}}$ associated with the constant input.
4. Define

$$\widetilde{\mathbf{x}} = \mathbf{x} - \mathbf{x}_{\text{eq}}$$

and derive the homogeneous shifted system.

5. Derive

$$\mathbf{x}(t) = e^{At}\left(\mathbf{x}(0) - \mathbf{x}_{\text{eq}}\right) + \mathbf{x}_{\text{eq}}.$$

6. Obtain explicit scalar expressions for $x(t)$ and $\dot{x}(t)$.
7. Compute the first time at which the displacement reaches its steady-state value.
8. Compute the maximum displacement over $0 \le t \le 5$.
9. Verify the analytical solution using a numerical ODE integrator.
10. Repeat the numerical solution using forward Euler with

$$h \in \{0.2, 0.1, 0.05, 0.025\}.$$

Estimate the observed global convergence order.

0.4 Optimization and Discretization

Optimization provides a precise language for stating what may change, what should improve, and what conditions must remain satisfied. Discretization then converts continuous-time functions into finite vectors that numerical algorithms can manipulate.

Basic optimization language

Optimization problem structure. A constrained optimization problem can be written as

$$\begin{aligned} \min_{\mathbf{x}} \quad & f(\mathbf{x}) \\ \text{subject to} \quad & g_i(\mathbf{x}) \le 0, \qquad i = 1, \dots, n_g, \\ & h_j(\mathbf{x}) = 0, \qquad j = 1, \dots, n_h, \\ & \mathbf{x}^L \le \mathbf{x} \le \mathbf{x}^U. \end{aligned}$$

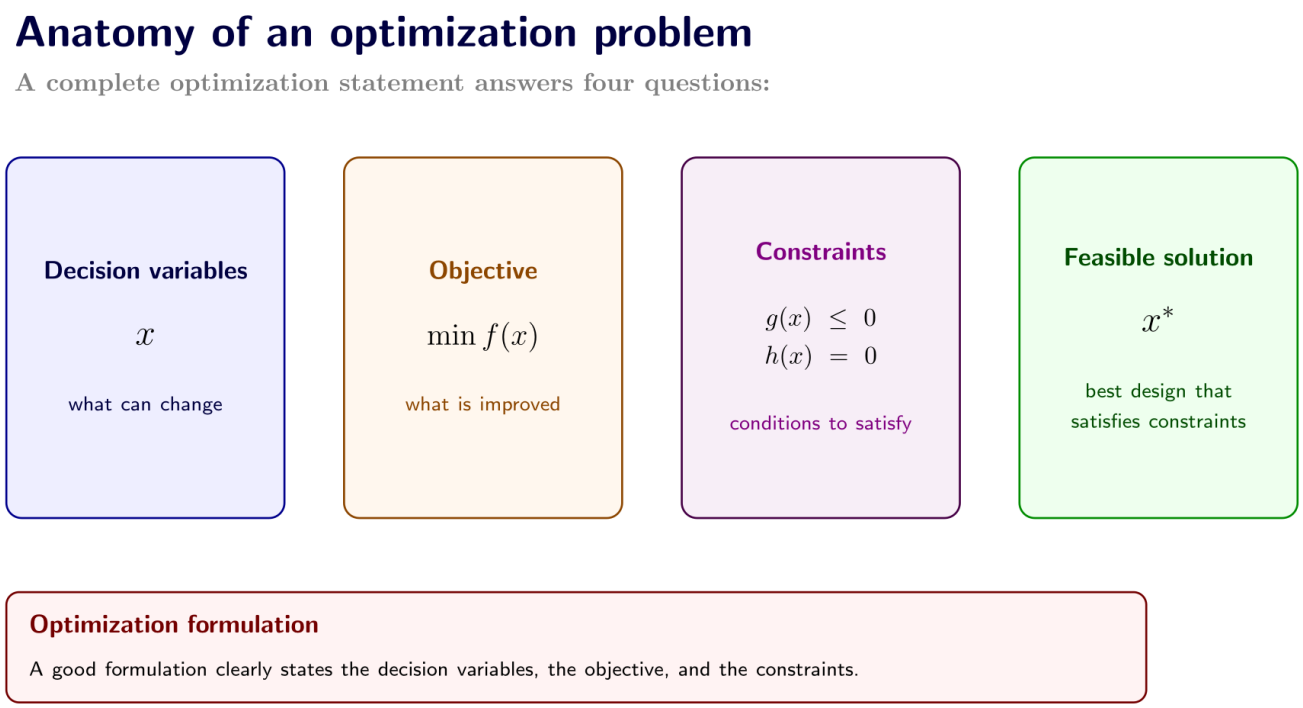


Figure 6: *
The principal parts of an optimization problem.

Core terms.

- **Decision variables:** Quantities the optimizer is allowed to change.
- **Objective function:** The quantity being minimized or maximized.
- **Constraints:** Conditions that a solution must satisfy.
- **Feasible point:** A point that satisfies every constraint.
- **Feasible region:** The set of all feasible points.
- **Optimal point:** A feasible point with the best objective value, according to the stated problem.

Example optimization statement. A simple engineering design problem is

$$\begin{aligned} \min_{m,k} \quad & J(m,k) = m^2 + 0.5k^2 \\ \text{subject to} \quad & m \geq 0.1, \\ & 10 \leq k \leq 1000. \end{aligned}$$

This statement specifies the quantities being chosen, the performance measure being improved, and the bounds that must hold.

How CCD builds on this structure. A CCD problem uses the same optimization language but adds dynamic-system structure. Its decision variables include plant and controller quantities, while its constraints often include state equations over time:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, \mathbf{x}_p, \mathbf{x}_c, t).$$

The objective may also depend on an entire trajectory rather than on a few static variables.

Discretization and sampled variables

Why discretization is needed. Many CCD problems are naturally continuous in time. Numerical optimizers, however, operate on finite-dimensional vectors rather than on arbitrary continuous functions. Continuous trajectories must therefore be approximated by samples or coefficients.

Time mesh. Divide a time interval into nodes:

$$t_0 < t_1 < \cdots < t_N.$$

The width of interval i is

$$\Delta t_i = t_{i+1} - t_i.$$

The mesh is **uniform** if every Δt_i is equal and **nonuniform** otherwise.

Sampled states and controls. At each node, approximate the state and control by

$$\mathbf{x}_i \approx \mathbf{x}(t_i), \qquad \mathbf{u}_i \approx \mathbf{u}(t_i).$$

If the control is scalar, its sampled trajectory becomes the finite vector

$$\mathbf{U} = \begin{bmatrix} u_0 & u_1 & \cdots & u_N \end{bmatrix}^T.$$

From differential equations to algebraic constraints. Discretization must also enforce the system dynamics. A simple forward-Euler approximation is

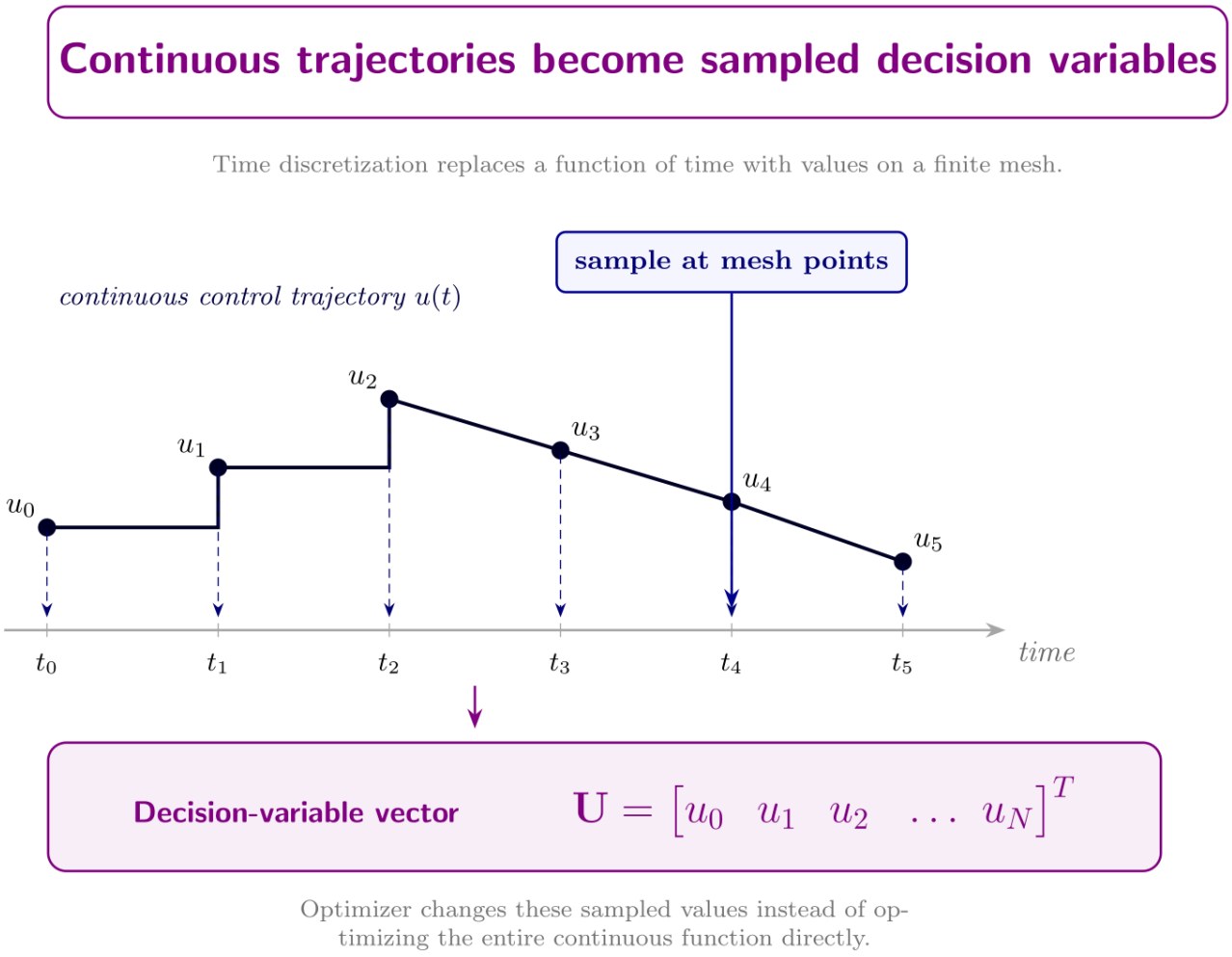


Figure 7: *
A continuous trajectory becomes a finite vector when sampled on a time mesh.

$$\dot{\mathbf{x}}(t_i) \approx \frac{\mathbf{x}_{i+1} - \mathbf{x}_i}{\Delta t_i}.$$

Substituting this approximation into

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

gives the algebraic relation

$$\mathbf{x}_{i+1} - \mathbf{x}_i - \Delta t_i \mathbf{f}(\mathbf{x}_i, \mathbf{u}_i, t_i) = \mathbf{0}.$$

Higher-accuracy methods use different approximations, but the central idea remains: a continuous-time model becomes a finite set of algebraic equations.

Why this matters later. This conversion underlies direct shooting, multiple shooting, direct transcription, and collocation. It also introduces important numerical questions:

- Is the mesh fine enough to represent the trajectory accurately?
- Does the discrete objective approximate the intended continuous objective?
- Do the discrete dynamics enforce the original ODE closely enough?
- Does refining the mesh change the apparent optimum?

Numerical diagnostic

A solution to the discretized problem is not automatically an accurate solution to the continuous problem. Mesh refinement and independent trajectory verification are essential later in the course.

Activity 0.5: Discretized Minimum-Energy Control Solved by Linear Algebra

Activity 0.5: Discretized Minimum-Energy Control Solved by Linear Algebra

Consider the double-integrator system

$$\dot{x}_1 = x_2, \qquad \dot{x}_2 = u, \qquad x_1(0) = x_2(0) = 0.$$

Use a time horizon $T = 1$ divided into four equal intervals of length $h = 1/4$. Assume the control is piecewise constant:

$$u(t) = u_i, \qquad t \in [ih, (i+1)h), \qquad i = 0, 1, 2, 3.$$

Use the exact zero-order-hold discrete dynamics

$$\begin{bmatrix} x_{1,i+1} \\ x_{2,i+1} \end{bmatrix} = \begin{bmatrix} 1 & h \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_{1,i} \\ x_{2,i} \end{bmatrix} + \begin{bmatrix} h^2/2 \\ h \end{bmatrix} u_i.$$

The terminal requirements are

$$x_1(1) = 1, \qquad x_2(1) = 0,$$

and the discrete control-effort objective is

$$J = h \sum_{i=0}^{3} u_i^2.$$

1. Propagate the dynamics symbolically and express the terminal state as

$$\mathbf{x}_4 = G\mathbf{U}, \qquad \mathbf{U} = \begin{bmatrix} u_0 & u_1 & u_2 & u_3 \end{bmatrix}^T.$$

2. Write the terminal requirements as $G\mathbf{U} = \mathbf{b}$.
3. Show that the problem is a minimum-norm equality-constrained quadratic problem.
4. Using Lagrange multipliers, prove that

$$\mathbf{U}^* = G^T(GG^T)^{-1}\mathbf{b}.$$

5. Compute all four optimal control values.
6. Compute the complete state trajectory at the five time nodes.
7. Verify the terminal conditions exactly.
8. Repeat the derivation for an arbitrary number N of equal intervals.
9. Explain how the dimensions of G, $\mathbf{U}$, and $\mathbf{b}$ change with N.
10. Compare the discrete solution with the continuous minimum-energy control obtained by calculus of variations or an optimal-control solver.

0.5 Computational Preparation and Readiness Project

CCD studies combine mathematical models with numerical simulation and optimization. You do not need advanced software-development experience to begin, but you should be comfortable reading and modifying short computational scripts.

Minimal MATLAB or Python preparation

Before continuing, you should be able to:

- define scalars, vectors, and matrices;
- write and call a simple function;
- evaluate a mathematical formula;
- simulate an ODE numerically;
- plot a result; and
- compute a discrete objective function.

A simple computational workflow

A basic modeling study follows a repeatable sequence:

1. Define parameters and initial conditions.
2. Write the model equations.
3. Simulate the model and plot the results.
4. Evaluate one or more performance measures.
5. Refine the model, parameters, or numerical settings.

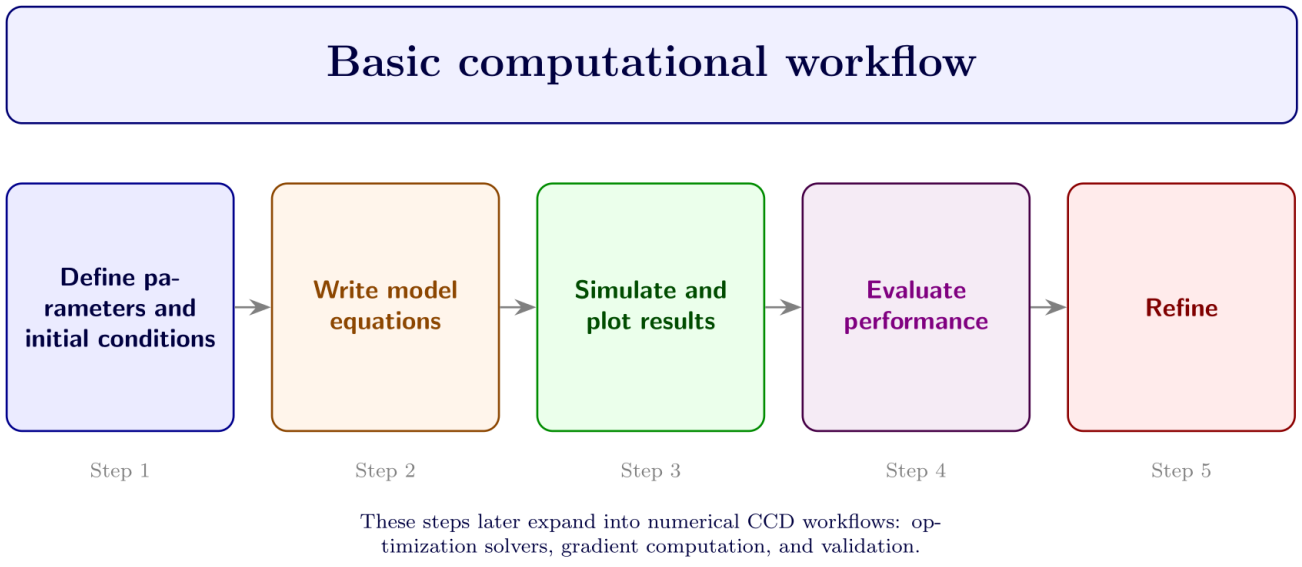


Figure 8: *
A basic workflow for modeling and simulation. Later CCD workflows add optimization solvers, gradient computation, uncertainty studies, and validation.

Python example

The following script defines a mass–spring–damper model and integrates it numerically. It is a preparation exercise rather than a full CCD implementation.

```
import numpy as np
from scipy.integrate import solve_ivp
import matplotlib.pyplot as plt

m = 1.0
c = 0.4
k = 4.0


def model(t, x):
    u = 1.0
    x1, x2 = x
    dx1 = x2
    dx2 = -(k / m) * x1 - (c / m) * x2 + (1 / m) * u
    return [dx1, dx2]


sol = solve_ivp(model, [0, 10], [0, 0], max_step=0.02)

plt.plot(sol.t, sol.y[0, :])
plt.xlabel("time")
plt.ylabel("displacement")
plt.show()
```

The script performs four basic tasks:

- specifies the model parameters m, c, and k;
- defines the first-order state equations;
- integrates the equations from an initial state; and
- plots the displacement trajectory.

MATLAB example

An analogous MATLAB script is

```
m = 1.0;
c = 0.4;
k = 4.0;

f = @(t,x) [x(2);
            -(k/m)*x(1) - (c/m)*x(2) + 1/m];

[t,x] = ode45(f,[0 10],[0;0]);

plot(t,x(:,1))
xlabel('time')
```

```
ylabel('displacement')
```

Both implementations express the same mathematical model. The syntax differs, but the modeling workflow is unchanged.

Computing a discrete performance measure

Suppose the continuous control-effort objective is

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

After sampling $u(t)$ on a time grid, a trapezoidal approximation can be computed in Python with

```
J = np.trapezoid(u**2, t)
```

or in MATLAB with

```
J = trapz(t, u.^2);
```

This simple pattern—simulate a trajectory, compute a performance measure, and compare alternatives—later becomes part of a numerical optimization loop.

Readiness mini-project

Before moving to Chapter 1, model and simulate a mass–spring–damper system and identify how its quantities would appear in a CCD formulation.

Suggested tasks.

1. Choose values for m, c, and k.
2. Write the second-order ODE and convert it to first-order state-space form.
3. Select initial conditions and a force input $u(t)$.
4. Simulate displacement and velocity.
5. Plot both state trajectories.
6. Compute at least one performance measure, such as peak displacement, RMS displacement, or control effort.
7. Explain which quantities could become decision variables and which conditions could become constraints.

Connection to a CCD formulation.

CCD element	Mass–spring–damper exa
Plant variables	m, c, and k
Control variables	Force-input history or co
States	Displacement and velocit
Possible objectives	Vibration suppression, c combination
Possible constraints	Actuator-force limits, di rameter bounds

Readiness checkpoint

If you can read vectors and matrices, compute simple derivatives and integrals, convert a second-order ODE to first-order form, understand the parts of an optimization problem, and follow a short simulation script, you are ready for the main body of the course.

Activity 0.6: State and Parameter Sensitivities for a Readiness Mini-Project

Activity 0.6: State and Parameter Sensitivities for a Readiness Mini-Project

Consider the mass-spring-damper system

$$m\ddot{q} + c\dot{q} + kq = u(t),$$

with

$$m = 1, \qquad q(0) = 1, \qquad \dot{q}(0) = 0, \qquad u(t) = e^{-t}.$$

The plant parameters and state vector are

$$\mathbf{p} = \begin{bmatrix} k \\ c \end{bmatrix}, \qquad \mathbf{x} = \begin{bmatrix} q \\ v \end{bmatrix}.$$

Define the performance measure

$$J(k, c) = \frac{1}{2} \int_0^5 \left(q(t)^2 + 0.1 v(t)^2 \right) dt + 0.01k^2 + 0.02c^2.$$

1. Derive the state-space model

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{p}, t).$$

2. Define the parameter-sensitivity vectors

$$\mathbf{s}_k = \frac{\partial \mathbf{x}}{\partial k}, \qquad \mathbf{s}_c = \frac{\partial \mathbf{x}}{\partial c}.$$

3. Derive the sensitivity equations

$$\dot{\mathbf{s}}_j = \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \mathbf{s}_j + \frac{\partial \mathbf{f}}{\partial p_j}, \qquad j \in \{k, c\}.$$

4. Derive the initial conditions for both sensitivity vectors.
5. Derive $\partial J / \partial k$ and $\partial J / \partial c$ in terms of the state and sensitivity trajectories.
6. Simulate the state and sensitivity equations simultaneously at

$$k = 4, \qquad c = 0.8.$$

7. Verify both derivatives using central finite differences.
8. Investigate the finite-difference error for

$$h \in \{10^{-2}, 10^{-4}, 10^{-6}, 10^{-8}\}.$$

9. Use the computed gradient to perform ten steepest-descent iterations with a backtracking line search, subject to

$$0.5 \le k \le 10, \qquad 0.05 \le c \le 4.$$

10. Report the final design, objective value, gradient norm, and state response.
11. Explain how this mini-project combines the main prerequisite topics of Chapter 0 and prepares students for later CCD formulations.

0.6 Summary and Problems

Chapter summary

This prerequisite chapter introduced the mathematical and computational foundations used throughout the course. It reviewed notation for scalars, vectors, matrices, and trajectories; functions, derivatives, gradients, Jacobians, Hessians, and integrals; essential linear-algebra concepts; the conversion of a second-order differential equation to first-order state-space form; basic optimization language; the discretization of continuous trajectories; and the minimal coding workflow required for simulation and optimization studies.

The aim was not to replace full courses in calculus, linear algebra, dynamics, optimization, or programming. It was to establish the vocabulary and tools needed to study control co-design clearly and efficiently. Chapter 1 now builds on this foundation to explain why plants and controllers should often be optimized together rather than separately.

Key terms

Mathematical language	Dynamics and computation	Optimization
Scalar	Trajectory	Decision variable
Vector	Ordinary differential equation	Objective function
Matrix	Initial condition	Equality constraint
Summation	State-space model	Inequality constraint
Derivative	Time discretization	Feasible point
Partial derivative	Sampled variable	Feasible region
Gradient	Simulation	Optimal point
Jacobian	Computational workflow	Performance index
Hessian	Numerical integration	Design bounds
Definite integral	Trapezoidal rule	Sensitivity

Problems

1. **Second-order sensitivity of a coupled objective.** Let $J(p,c) = \frac{1}{2}(ap^2 + 2bpc + dc^2) + \alpha e^{p-c}$, where $a, d > 0$ and $ad - b^2 > 0$. Derive the second-order Taylor model of J about an arbitrary point (p_0, c_0) and determine an explicit condition on α and a rectangular domain $\mathcal{D}$ that guarantees strict convexity throughout $\mathcal{D}$.
2. **KKT system for a constrained design.** Consider $\min_{x_1,x_2}(x_1-2)^2 + 3(x_2+1)^2$ subject to $x_1^2 + x_2^2 \le 4$ and $x_1 + x_2 \ge 1$. Derive and solve the complete Karush–Kuhn–Tucker system, then certify the global optimizer using the geometry and convexity of the problem.
3. **State transition and controllability.** For $\dot{\mathbf{x}} = A\mathbf{x} + Bu$ with $A = \begin{bmatrix} 0 & 1 \\ -k/m & -c/m \end{bmatrix}$ and $B = \begin{bmatrix} 0 \\ 1/m \end{bmatrix}$, derive the finite-horizon controllability Gramian $W_c(T)$ in terms of the matrix exponential and prove that $W_c(T) > 0$ for every $T > 0$ when $m > 0$.
4. **Lyapunov certificate with a design parameter.** For $A(q) = \begin{bmatrix} 0 & 1 \\ -q & -2 \end{bmatrix}$, solve $A(q)^T P + PA(q) = -I$ analytically for the symmetric matrix P and determine the complete range of q for which this equation supplies a positive-definite Lyapunov certificate.
5. **Trajectory sensitivity.** The nonlinear system $\dot{x} = -px^3 + u(t)$ has $x(0) = x_0$ and objective $J(p) = \int_0^T x(t;p)^2\,dt$. Derive a forward-sensitivity initial-value problem whose solution gives dJ/dp without finite differences.
6. **Index-one differential-algebraic model.** A constrained mechanism is modeled by $M(q,p)\ddot{q} + h(q,\dot{q},p) +$

$G(q)^T\lambda = Bu$ with holonomic constraint $\phi(q) = 0$. Derive a first-order index-one DAE for (q, v, λ) and state a nonsingularity condition that makes the algebraic multiplier locally unique.

7. **Error of trajectory quadrature.** For $J = \int_0^T q(t)\,dt$ with $q \in C^2[0,T]$, derive a global error bound for the composite trapezoidal rule on a nonuniform mesh and use it to construct a mesh-spacing condition that guarantees $|J - J_h| \le \varepsilon$.

8. **Direct transcription of a dynamic optimization problem.** Transcribe $\min_u \int_0^T (x^2 + \rho u^2)\,dt$ subject to $\dot{x} = -ax + bu$, $x(0) = x_0$, and $|u| \le u_{\max}$ using trapezoidal defects on N intervals, giving the resulting nonlinear program in matrix form and its exact sparsity pattern.

9. **Adjoint gradient.** For $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, p)$ and $J(p) = \Phi(\mathbf{x}(T), p) + \int_0^T L(\mathbf{x}, \mathbf{u}, p)\,dt$, derive the continuous adjoint equation, terminal condition, and a single integral expression for dJ/dp that eliminates the state sensitivity.

10. **Nondimensionalization and conditioning.** For $m\ddot{z} + c\dot{z} + kz = F$ with characteristic displacement z_0 and time $t_0 = \sqrt{m/k}$, derive the dimensionless dynamics and objective $J = \int_0^T (q_z z^2 + q_F F^2)\,dt$, then choose variable and residual scalings that make every coefficient and typical decision magnitude order one.

References and further reading

1. Martins, J. R. R. A., & Ning, A. (2021). *Engineering design optimization*. Cambridge University Press.

2. Boyd, S., & Vandenberghe, L. (2018). *Introduction to applied linear algebra: Vectors, matrices, and least squares*. Cambridge University Press.

3. Betts, J. T. (2010). *Practical methods for optimal control and estimation using nonlinear programming* (2nd ed.). Society for Industrial and Applied Mathematics. DOI: 10.1137/1.9780898718577

4. Allison, J. T., & Herber, D. R. (2014). Multidisciplinary design optimization of dynamic engineering systems. *AIAA Journal*, *52*(4), 691–710. DOI: 10.2514/1.J052182

Introduction to Control Co-Design

Chapter 1: Introduction to Control Co-Design

Why the physical system and controller should be designed together

> The best plant depends on the controller, and the best controller depends on the plant.

Modern engineering systems increasingly rely on sensing, computation, and actuation. A wind turbine changes blade pitch and generator torque in response to wind. An active suspension applies forces to isolate passengers from road disturbances. A robot coordinates its geometry, motors, sensors, and motion controller to complete a task. In each case, the physical system and controller jointly determine performance.

Conventional design often treats these parts sequentially: design the physical system first, freeze it, and then design a controller for the resulting plant. This workflow is convenient, but it can eliminate valuable combinations of physical and control decisions before they are evaluated.

Control co-design (CCD) treats the physical system and controller as parts of one integrated engineering design problem. This chapter develops the motivation, vocabulary, mathematics, and engineering judgment needed to recognize and formulate a CCD problem.

Learning objectives

After completing this chapter, you should be able to:

1. identify the plant, controller, sensors, actuators, states, outputs, and disturbances in an active system;
2. distinguish architecture, plant, and control as three design domains, and distinguish physical/control architecture from the numerical CCD architectures (nested, simultaneous) introduced later;
3. distinguish sequential design from control co-design;
4. explain plant-control coupling in physical and mathematical terms;
5. explain why a sequential design can be feasible yet system-suboptimal;
6. formulate a simple system-level objective containing plant and control decisions;
7. classify how a plant-design objective can represent, approximate, or ignore active control, and explain why that classification predicts sequential-design suboptimality;
8. recognize applications where CCD is likely to provide value;
9. distinguish CCD from controller tuning or merely combining variables in one software model;
10. place CCD within the broader landscape of control-inspired, co-optimization, and co-simulation methodologies; and
11. evaluate CCD claims with appropriate attention to uncertainty, implementation, and verification.

Chapter roadmap

Active systems → Sequential design → Control co-design → Plant-control coupling → example → Applications → Screening and responsible practice

The chapter begins with the anatomy of active systems, compares sequential and integrated design, introduces a compact CCD formulation, and develops coupling through a mass-spring-damper example. It then turns to applications, screening questions, limitations, and end-of-chapter exercises.

Actively Controlled Engineering Systems

The physical system is only part of the design

An engineering system is **active** when it senses its condition or environment and uses an actuator to modify its behavior. A useful system description contains five interacting elements:

1. **Physical plant:** the object or process being controlled, such as a vehicle suspension, wind turbine, robotic arm, building, aircraft, or energy converter.
2. **Sensors:** devices that measure position, velocity, strain, temperature, pressure, acceleration, current, or other quantities.
3. **Controller:** logic that uses measurements, models, references, and design parameters to determine a command.
4. **Actuators:** devices that convert the command into physical action such as force, torque, heat flow, voltage, valve position, or blade pitch.
5. **Environment:** external conditions and disturbances such as road roughness, wind, waves, payload changes, and uncertain loads.

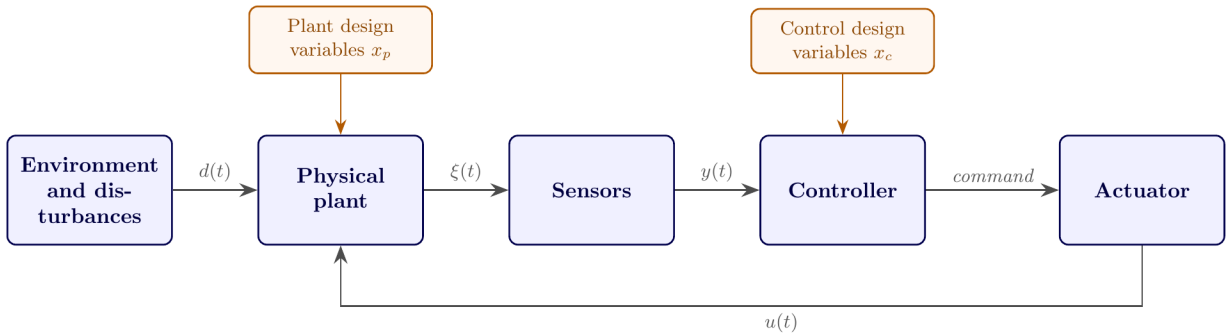


Figure 1.1: *
A generic actively controlled engineering system. Every block contributes to the behavior of the complete system.

Definition

An actively controlled engineering system is a physical system whose behavior is intentionally modified using sensing, computation, and actuation.

The controller is not an isolated software object. It acts through a physical actuator, receives imperfect measurements from physical sensors, and operates on a plant exposed to disturbances. Sensor bandwidth, actuator limits, power availability, computation, and plant dynamics therefore belong to the same engineering story.

Plant and control design variables

Engineers choose quantities called **design variables**. Once a system's architecture is fixed—an assumption examined critically later in this section—two broad groups of variables remain to be chosen.

Plant design variables $x_p \in \mathbb{R}^{n_p}$ describe the physical system. Examples include:

- geometry, shape, and material;
- mass, inertia, stiffness, and damping;

- gear ratio, generator rating, battery capacity, and actuator size; and
- sensor or actuator placement when it changes the physical system.

Control design variables $x_c \in \mathbb{R}^{n_c}$ describe how control action is generated. Examples include:

- PID or state-feedback gains;
- estimator and filter parameters;
- sampling time and model-predictive-control horizon;
- parameters of a control policy; and
- an entire control trajectory $u(t)$ in open-loop optimal control.

The boundary is sometimes context dependent. A sensor can be viewed as hardware, an information-architecture choice, or both. The label is less important than representing how the decision changes dynamics, information, performance, cost, or constraints.

A third design domain: architecture

Plant and control variables both presume that *which components exist and how they connect* has already been decided. That decision is itself a design choice, and a consequential one: **architecture** is the set of elements contained in a system and the relationships among them—which components are present, of what type, and how they are connected. Architecture design is normally a **discrete** decision (which components, in which arrangement), in contrast to the continuous sizing decisions of x_p and the trajectory or gain decisions of x_c.

A pick-and-place robotic manipulator makes the distinction concrete. Before any link length or joint controller can be chosen, a designer must decide the manipulator's architecture: how many links and joints it has, how they are connected (series or parallel kinematic chains), which joints carry an actuator and are therefore active, and whether extra structural components—an end-effector plate, a brace—are included. A two-link serial arm, a three-link serial arm, and a four-link parallel arm performing the identical pick-and-place task are three different architectures, each admitting its own plant variables (a two-link arm has two link-length variables; a four-link arm has four) and its own control problem (a different number of actuated joints to command). Only once an architecture is fixed do "the plant variables" and "the control variables" refer to a well-defined, finite list.

Two different meanings of "architecture"

This book uses the word *architecture* twice, for two unrelated ideas—an ambiguity worth resolving now rather than later. **Physical/control architecture** (this section) means which components exist and how they connect: a discrete structural or topological decision about the system itself. **CCD architecture**, the subject of a later chapter title, means which *numerical coordination strategy*—iterative sequential, nested, or simultaneous—is used to *solve* a plant-and-control optimization problem once the physical architecture is already fixed. A nested CCD architecture can be used to solve a two-link manipulator's plant-and-control problem, or a four-link manipulator's; the two uses of "architecture" describe different layers of the same design process and should not be conflated.

Architecture, plant, and control decisions are often assigned to different engineers or made at different times—an organizational habit rather than a technical necessity—and the same practical hazard identified for sequential plant-then-control design applies one level up: fixing an architecture before plant and control possibilities under alternative architectures have been explored can foreclose the best system-level design just as surely as fixing a plant before its controller does. A hybrid-electric vehicle's series, parallel, and power-split drivetrain architectures are a common example: each admits its own plant sizing and control problem, and comparing architectures fairly requires solving a genuine co-design problem within each one rather than reusing a plant or controller tuned for a different architecture.

Decisions also migrate between domains depending on how much a designer is willing to model. A structural topology decision can be relaxed into a continuous plant variable (density-based structural optimization treats "is material present here" as a continuous $[0, 1]$ variable rather than a discrete choice); a genuinely architectural decision—which power-take-off hardware a wave energy converter uses—is sometimes simplified into an assumed control structure so that standard control-design tools apply; and, in the sequential workflows criticized throughout this chapter, a control decision is sometimes reduced to tuning only the passive plant, leaving the feedback controller out of the problem entirely. None of these simplifications is wrong by itself; each one trades scope for tractability, and a CCD study should be explicit about which domain's flexibility it has traded away.

States, outputs, controls, and disturbances

Let the state, control input, disturbance, and output be

$$\xi(t) \in \mathbb{R}^{n_s}, \qquad u(t) \in \mathbb{R}^{n_u}, \qquad d(t) \in \mathbb{R}^{n_d}, \qquad y(t) \in \mathbb{R}^{n_y}.$$

A general state-space model is

$$\dot{\xi}(t) = f\big(t, \xi(t), u(t), x_p, d(t)\big),$$

$$y(t) = h\big(t, \xi(t), u(t), x_p, d(t)\big).$$

The physical design x_p appears inside the dynamic model. Changing a mass, dimension, stiffness, aerodynamic shape, or gear ratio changes the equations of motion and can therefore change which controller is best.

Checkpoint

For an active vehicle suspension, identify one plant variable, state, disturbance, sensor output, control input, and system-level performance metric. Then draw arrows showing which quantities directly influence one another.

Sequential Design and Control Co-Design

Why sequential design is common

Engineering organizations are divided into specialties for good reasons. Mechanical engineers develop structures and mechanisms; electrical engineers select sensors and power electronics; control engineers design feedback algorithms; software engineers implement them; and manufacturing teams evaluate cost and producibility.

Specialization, however, encourages a one-pass workflow:

1. design the physical plant using assumed operating conditions and an assumed control strategy;
2. freeze or largely fix the plant;
3. design the controller for that fixed plant; and
4. test the complete system and revise it if performance is unacceptable.

This workflow gives clear ownership, smaller models, and compatibility with established disciplinary tools. It may also be the only practical option after hardware is built. The difficulty is that freezing the plant can remove better system-level combinations before the final controller is known.

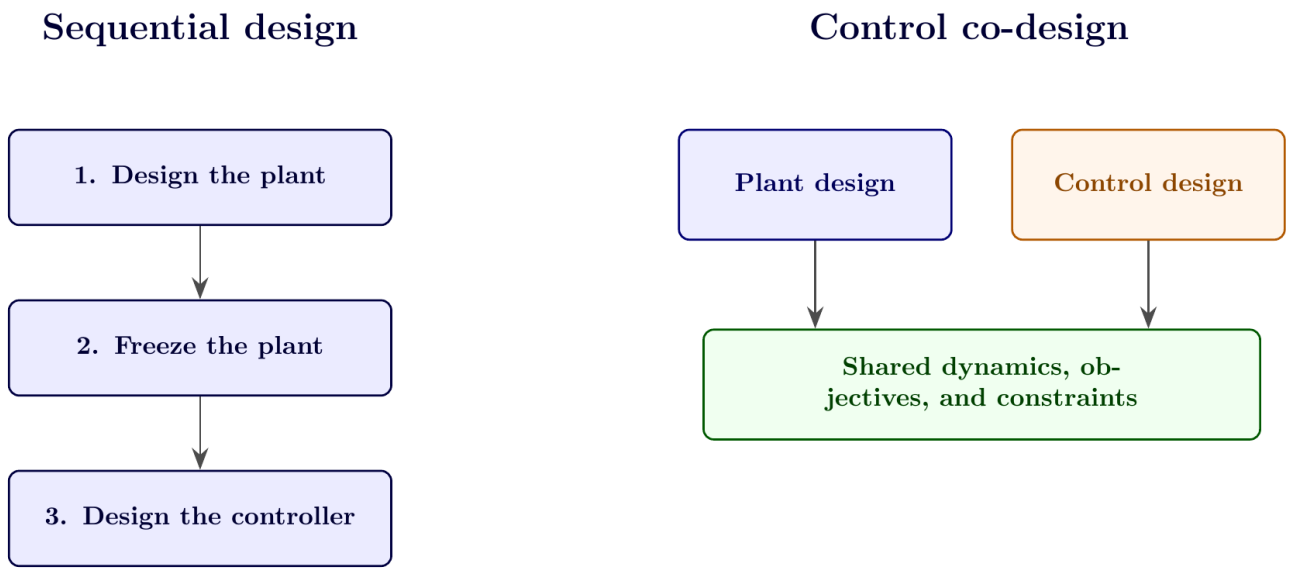


Figure 1.2: *
Sequential design fixes the plant before final controller design. CCD coordinates both sets of decisions through shared dynamics, objectives, and constraints.

Feasible does not mean system-optimal

A design is **feasible** when it satisfies its constraints. A design is **optimal** only relative to a specified objective, model, and feasible set. A sequential design can meet every safety and performance requirement yet still use excess material, require too much control effort, consume unnecessary energy, or deliver less performance than another feasible plant-controller pair.

Key idea

The criticism of sequential design is not that it always fails. Fixing one design domain too early can remove better system-level solutions from consideration.

Sequential design may be adequate when coupling is weak, the controller has little authority, one subsystem is fixed by regulation or legacy hardware, the plant barely affects the control problem, or the coupled model is too uncertain to justify added complexity.

Real consequences of ignoring coupling

The failure mode is not hypothetical. Two documented cases illustrate what happens when a physical system is finalized without full consideration of its interaction with active behavior.

When London's Millennium Bridge opened in June 2000, pedestrians crossing it excited unexpected lateral vibration. As the deck swayed, pedestrians unconsciously synchronized their footsteps to the motion to keep their balance, and that synchronized walking pumped more energy into the same lateral mode, amplifying the sway until the bridge had to be closed. The passive structural design had not accounted for how people—an uncontrolled but dynamically responsive part of the system—would interact with the deck's dynamics. The eventual fix retrofitted tuned mass and viscous dampers, an expensive redesign that earlier consideration of the coupled pedestrian-structure dynamics might have avoided or reduced.

A second case comes from agricultural equipment. Combine harvesters must hold their header at a narrow height above the ground to cut crops cleanly without hitting the soil, and header-height control performance is a real limit on how fast a harvester can operate. In at least one documented case, engineers found that further performance gains could not be reached by improving the controller alone: the physical-system design had not been developed with the header-height control problem in mind, and the control-relevant dynamics could only be improved by redesigning the physical header and its linkage—that is, by revisiting plant decisions that had already been frozen.

Key idea

Both cases share the same structure: a physical design was finalized using assumptions about behavior that turned out to depend on the very control (or uncontrolled human) response the design was supposed to accommodate. Neither failure required exotic engineering—both arose from ordinary sequential design applied to a system whose plant and control behavior were, in fact, coupled.

What control co-design means

Definition

Control co-design is the integrated design of an actively controlled engineering system in which physical-system and control-system decisions are coordinated using shared models, objectives, and constraints.

The definition contains four essential ideas:

- **Integrated:** plant and control decisions are not optimized independently and merely combined later.
- **Dynamic:** behavior through time is central to performance.
- **System-level:** objectives and constraints represent the mission and limitations of the complete system.
- **Coordinated:** the problem may be solved simultaneously, through nested optimization, or through a managed iterative strategy.

CCD is a specialized instance of a broader field called **multidisciplinary dynamic system design optimization (MDSDO)**: design optimization for systems in which the evolution of state through time is a critical element of performance, multiple disciplines or energy domains must be integrated, and the specific properties of dynamic systems are exploited to improve performance and problem tractability. MDSDO is itself a branch of multidisciplinary design optimization (MDO), the broader field concerned with optimizing systems whose performance depends on interacting disciplines (structural, aerodynamic, electrical, thermal, economic, and so on). CCD adds one more ingredient on top of MDSDO: the explicit presence of an active *controller* as a first-class design object, not merely a fixed regulator applied after the fact. Realistic CCD formulations may combine structures, fluids, electrical systems, thermal systems, economics, reliability, sensing, software, and manufacturing—so a working command of MDO ideas transfers directly.

Three complementary methodologies

CCD is often equated with the compact optimization statement introduced below, but that formal, mathematical style is only one of at least three complementary traditions engineers use to achieve integrated plant-control design:

1. **Control-inspired paradigms.** Practicing control engineers use frequency- and time-domain reasoning—Bode and Nichols diagrams, root locus, bandwidth, resonance, disturbance rejection, actuator-sensor colocation—to propose new mechanisms, sensor and actuator placements, and control structures directly from physical insight. This is the oldest and most engineering-intuitive route to CCD; the Wright brothers and Charles Brush, profiled later in this chapter, practiced it without ever writing an optimization problem.
2. **Co-optimization.** A formal mathematical methodology that poses plant and control decisions as variables in a single constrained optimization problem, typically nonlinear and often with dynamics enforced by an equality constraint. This is the most mature and most rigorously teachable of the three traditions, and it is the primary focus of this course starting with the mathematical statement below.
3. **Co-simulation.** Multiscale, multiphysics, high- or mixed-fidelity simulation—sometimes combined with data-driven or machine-learning models—used to explore designs iteratively when a closed-form or tractable optimization formulation is not yet available. Co-simulation becomes essential once models grow too complex or too expensive to embed directly inside an optimizer, a theme this course returns to in the discussion of model fidelity and surrogates.

These traditions are not mutually exclusive, and combining them tends to produce the most capable designs: control-inspired reasoning supplies engineering creativity and sanity checks, co-optimization supplies mathematical rigor and repeatability, and co-simulation supplies the ability to handle complexity that resists closed-form treatment. Two derived ideas sit at their intersections and are worth knowing by name even before they are developed later in the course: a **digital twin** couples co-simulation with real-time sensor data from the physical system, and **experimentally infused optimization** couples co-optimization with offline experimental data to reduce the uncertainty a numerical optimizer must otherwise absorb.

Checkpoint

Every CCD study also depends on the same five inputs regardless of which methodology is used: system objectives, a pre-design of the relevant components, physics-based models, real data (lab or field), and representative case studies. Missing or weak inputs propagate into a weak result no matter how sophisticated the chosen methodology is.

A first mathematical statement

A compact deterministic CCD problem is

$$
\begin{aligned}
\underset{x_p, x_c, \xi(\cdot)}{\text{minimize}} \quad & J\big(x_p, x_c, \xi(\cdot), u(\cdot)\big) \\
\text{subject to} \quad & \dot{\xi}(t) = f\big(t, \xi(t), u(t), x_p\big), \\
& g\big(t, \xi(t), u(t), x_p, x_c\big) \le 0, \\
& h\big(\xi(t_0), \xi(t_f), x_p, x_c\big) = 0, \\
& u(t) = \mathcal{K}(\text{available information}; x_c).
\end{aligned}
$$

The objective J measures system performance. The dynamic equation enforces physical consistency. Inequalities can impose stress, displacement, temperature, actuator, power, or safety limits. Boundary equations impose initial and terminal conditions. The final relation makes the controller's information explicit.

Common coordination strategies

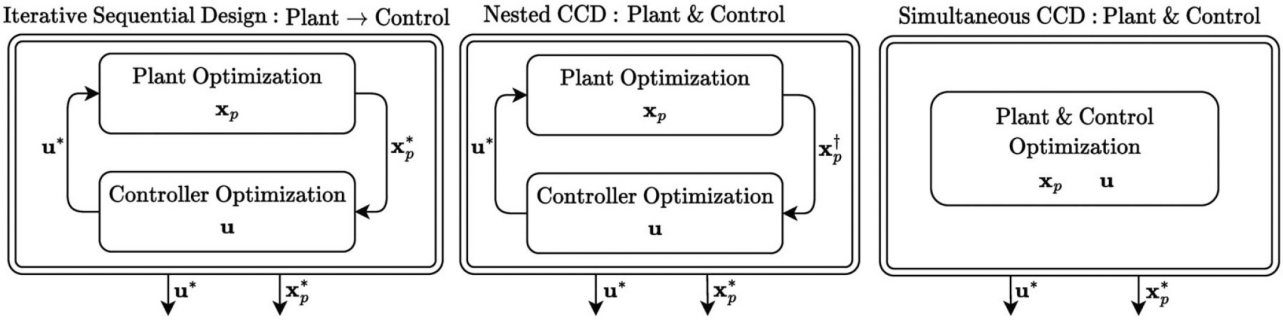


Figure 1.3: *
CCD is defined by integrated system design, not by a single numerical architecture.

- **Iterative sequential:** alternate plant and controller subproblems until a stopping rule is reached.
- **Nested:** solve an inner control problem for each candidate plant in an outer plant-design loop.
- **Simultaneous:** optimize plant variables, control variables, and often state trajectories in one problem.

All three attempt to account for plant-control coupling. A one-pass plant-then-controller workflow does not.

The iterative-sequential strategy has a precise mathematical identity: repeatedly optimizing the plant with the controller held fixed, then the controller with the plant held fixed, and alternating until neither subproblem changes, is an instance of **block coordinate descent (BCD)**—a general optimization algorithm that partitions the decision variables into blocks and optimizes one block at a time. If each subproblem is solved to a unique optimum and both blocks use the same, fully consistent system-level objective, BCD is guaranteed to converge to the same solution as the simultaneous problem. Two caveats matter in practice. First, if a coupling model is missing—if the plant subproblem is not actually re-solved against the true system objective once control is included—the iteration is not BCD at all; it is the one-pass sequential workflow criticized above, dressed up as "iteration." Second, even genuine BCD can converge slowly or fail to converge cleanly when the plant-control cross-coupling is strong: the number of iterations required grows sharply as the interaction between the plant and control blocks strengthens, and for non-convex or weakly identified coupling, BCD can cycle rather than converge. This is the same mechanism you will meet in Activity 1.1: the cross term in $J(p, c) = 4p^2 + 3c^2 + 4pc - \cdots$ is exactly the kind of plant-control interaction whose magnitude governs how many alternating passes are needed—and whether alternating is even a sound strategy at all.

One system-level objective

An active-suspension objective might combine comfort, road holding, effort, mass, and cost:

$$J = \int_{t_0}^{t_f} \left(w_a a_s^2(t) + w_r \Delta z_t^2(t) + w_u F^2(t) \right) dt + w_m M(x_p) + w_C C(x_p).$$

This does not imply that every system has only one goal. Multiobjective or constrained formulations are often appropriate. The important requirement is that the same system-level priorities govern both plant and control decisions.

Activity 1.1: Exact Sequential versus Simultaneous Co-Design

Activity 1.1: Exact Sequential versus Simultaneous Co-Design

Consider the coupled quadratic design problem

$$J(p, c) = 4p^2 + 3c^2 + 4pc - 12p - 10c + 20,$$

where p is a plant-design variable and c is a controller-design variable, subject to

$$0 \le p \le 3, \qquad 0 \le c \le 3.$$

1. Compute the Hessian of J and determine whether the problem is strictly convex.
2. In a single-pass sequential workflow, set the nominal controller to $\hat{c} = 0$. First optimize p, then freeze p and optimize c. Compute

$$p_{\text{seq}}, \qquad c_{\text{seq}}, \qquad J_{\text{seq}}.$$

3. Solve the simultaneous CCD problem

$$\min_{p,c} J(p, c)$$

analytically.

4. Compute the percentage improvement

$$\eta = \frac{J_{\text{seq}} - J_{\text{CCD}}}{J_{\text{seq}}} \times 100\%.$$

5. Repeat the sequential calculation using

$$\hat{c} \in \{0.5, 1, 2\}.$$

Determine how strongly the final sequential design depends on the assumed nominal controller.

6. Derive the value of $\hat{c}$ for which the single-pass sequential workflow reproduces the simultaneous optimum exactly.
7. Explain why this special value does not make sequential design generally equivalent to CCD.

Activity 1.2: System-Level Objective and Hidden Tradeoffs

Activity 1.2: System-Level Objective and Hidden Tradeoffs

An active-suspension study uses the weighted objective

$$J = w_a J_a + w_t J_t + w_u J_u + w_m J_m,$$

where

$$J_a = \int_0^T a_s(t)^2\,dt, \quad J_t = \int_0^T \Delta z_t(t)^2\,dt,$$
$$J_u = \int_0^T F(t)^2\,dt, \quad J_m = M(\mathbf{x}_p).$$

Two feasible designs produce the normalized metrics

$$\mathbf{J}^{(A)} = \begin{bmatrix} 0.60 \\ 1.10 \\ 1.45 \\ 0.75 \end{bmatrix}, \qquad \mathbf{J}^{(B)} = \begin{bmatrix} 0.85 \\ 0.80 \\ 0.70 \\ 1.10 \end{bmatrix}.$$

1. Determine the region of nonnegative weight vectors

$$\mathbf{w} = \begin{bmatrix} w_a & w_t & w_u & w_m \end{bmatrix}^T, \qquad \sum_i w_i = 1,$$

   for which design A is preferred.

2. Determine the region for which design B is preferred.
3. Show that neither design dominates the other in the Pareto sense.
4. Determine which design is selected for each weight vector:

$$\mathbf{w}^{(1)} = \begin{bmatrix} 0.55 & 0.20 & 0.15 & 0.10 \end{bmatrix}^T,$$
$$\mathbf{w}^{(2)} = \begin{bmatrix} 0.25 & 0.25 & 0.25 & 0.25 \end{bmatrix}^T,$$
$$\mathbf{w}^{(3)} = \begin{bmatrix} 0.20 & 0.25 & 0.15 & 0.40 \end{bmatrix}^T.$$

5. Suppose passenger comfort must satisfy the hard requirement $J_a \leq 0.75$. Determine which designs remain feasible.
6. Suppose actuator effort must satisfy $J_u \leq 1$. Determine which designs remain feasible.
7. Explain why moving an engineering requirement from the objective to a constraint can change the selected CCD solution even when the same physical metric is used.
8. Construct a third feasible design C that dominates either A or B, but not both.
9. Explain why reporting only the aggregate weighted objective can hide an unacceptable engineering compromise.

Plant-Control Coupling

Two groups of decisions are **coupled** when changing one group changes the best choice in the other. In CCD, the plant changes dynamics, actuator requirements, sensor signals, and achievable performance; the controller changes loads, motions, energy flow, fatigue, and the value of alternative plant designs.

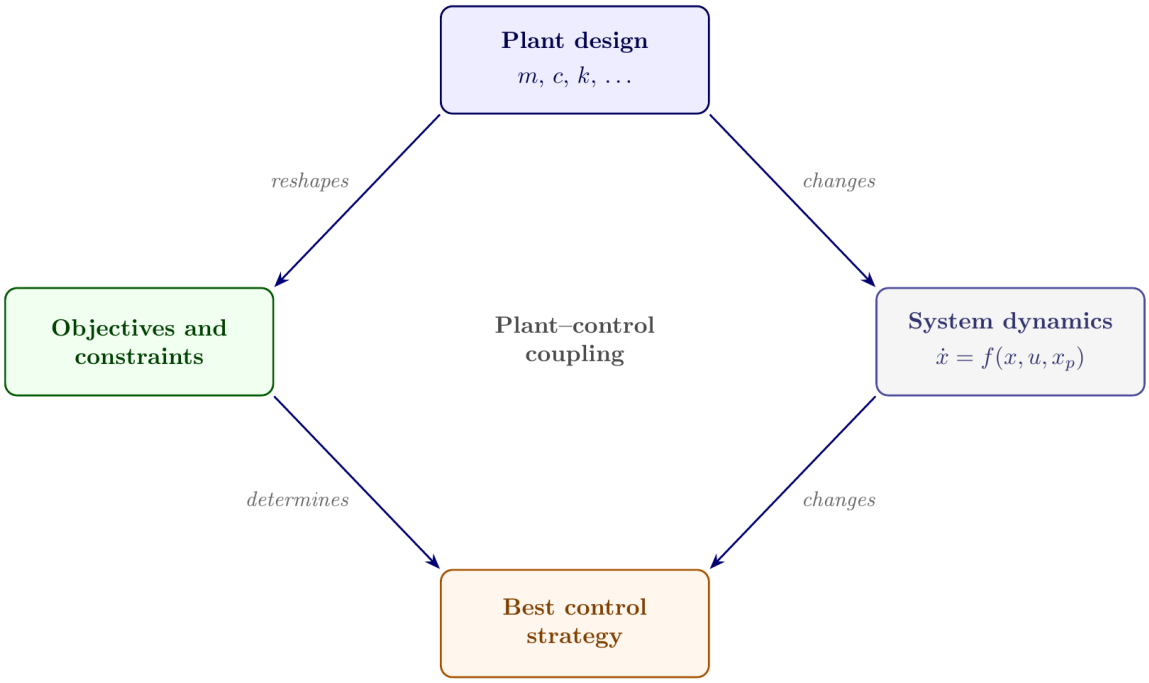


Figure 1.4: *
Plant decisions change the dynamics, the dynamics change the best controller, and controlled behavior changes which plant best serves the mission.

A mass-spring-damper example

Consider a mass m, spring k, damper c, actuator force $u(t)$, and disturbance $d(t)$:

$$m\ddot{x}(t) + c\dot{x}(t) + kx(t) = u(t) + d(t).$$

With proportional-derivative feedback,

$$u(t) = -K_p x(t) - K_d \dot{x}(t),$$

the closed-loop system becomes

$$m\ddot{x}(t) + (c + K_d)\dot{x}(t) + (k + K_p)x(t) = d(t).$$

Passive stiffness k and proportional feedback K_p both contribute to effective closed-loop stiffness. Passive damping c and derivative feedback K_d both contribute to effective damping. They are not interchangeable in practice: active action needs sensors, power, computation, bandwidth, and a functioning actuator. Nevertheless, they can partly substitute for or complement one another.

If k is chosen before K_p is considered, the design may contain more passive stiffness than needed once active control becomes available. If K_p is chosen without plant cost or actuator limits, the controller may demand unrealistic force or energy.

Allocation example

Suppose the design requires an effective stiffness

$$k + K_p = 6,$$

with normalized cost

$$C(k, K_p) = 0.08k^2 + 0.02K_p^2.$$

A passive-first design sets $K_p = 0$, giving $k = 6$ and

$$C_{\text{seq}} = 0.08(6)^2 = 2.88.$$

For co-design, substitute $K_p = 6 - k$:

$$C(k) = 0.08k^2 + 0.02(6 - k)^2.$$

Setting $dC/dk = 0$ yields $k = 1.2$ and $K_p = 4.8$, with

$$C_{\text{CCD}} = 0.08(1.2)^2 + 0.02(4.8)^2 = 0.576.$$

The numbers are deliberately simple. The lesson is general: when passive and active decisions contribute to the same system property, the least-cost allocation may be invisible to a one-pass sequential workflow.

Five ways a plant objective can misrepresent the system

The example above hides an important, more general question: when a plant subproblem is optimized on its own, exactly *which* objective is it optimizing—and how faithfully does that objective represent the true, complete system objective $\phi(\xi(t), u(t), x_p)$? Plant-design objectives used in practice fall into five cases, and only two of them are trustworthy:

Case	Accounts for control?	Uses full dynamics?	Objective used
1	No	Yes (passive dynamics)	Exact system objective $\phi(\xi(t), x_p)$
2	No	Yes (passive dynamics)	Approximation $\hat{\phi}(\xi(t), x_p)$
3	No	No (static analysis)	Approximation $\hat{\phi}(x_p)$
4	Yes	Yes	Exact system objective $\phi(\xi(t), u(t), x_p)$
5	Yes	Yes	Approximation $\hat{\phi}(\xi(t), u(t), x_p)$

Case 1 is what "sequential design" means when it is done as well as it can be done without control: the plant is optimized using the real system objective, honestly evaluated with passive dynamics ($u \equiv 0$). Case 4 is the

true co-design objective: the same real system objective, now evaluated with active control included—this is the objective J inside the compact CCD statement introduced earlier in this chapter. Cases 2, 3, and 5 substitute an *approximation* $\hat{\phi}$ for the true objective ϕ, most often because building a coupled dynamic model is expensive. Case 3 is the most severe simplification and the most common one in practice: it drops dynamics altogether and uses a static or steady-state proxy—mass, a gravity-balance condition, a frequency-domain margin—standing in for the real, time-dependent performance metric.

Why this matters

Nothing prevents a Case 2, 3, or 5 plant objective from producing a feasible, defensible-looking design. The problem is invisible from inside the plant subproblem: $\hat{\phi}$ has no mechanism for reporting how far it has drifted from ϕ. Coupling strength determines how much this drift costs. When coupling is weak, $\hat{\phi} \approx \phi$ and the approximation is nearly harmless. When coupling is strong—exactly the systems this course is about—the plant that minimizes $\hat{\phi}$ can be far from the plant that minimizes ϕ, which is the mathematical reason a feasible sequential design can be system-suboptimal.

In the worked allocation example, the passive-first design implicitly used a Case-3-style objective: it optimized k using a proxy that assumed $K_p = 0$, never evaluating what K_p would eventually need to be. A genuine CCD solution requires a Case 1 or Case 4 objective—the real system objective, honestly evaluated, whether or not active control is present in it.

Common forms of coupling

Coupling type	Plant-to-control effect	Control-to-plant effect
Dynamic	Mass, stiffness, damping, geometry, and operating point change the state equations.	Feedback changes effective dynamics, resonance, damping, and transient loads.
Constraint	Plant design changes actuator, temperature, stress, or motion limits.	Control trajectories determine whether stress, fatigue, stroke, and power limits are satisfied.
Economic	Plant size and material alter capital and maintenance cost.	Controller complexity, sensing, computation, and energy use alter lifecycle cost.
Information	Sensor placement and observability change what the controller can know.	Information needs can motivate different sensor locations, structures, or architectures.
Reliability	Passive stability and redundancy alter failure behavior.	Control can reduce loads but introduces hardware, software, and cyber failure modes.
Architecture	Components and connections determine available dynamic pathways.	Desired authority may motivate adding, removing, or relocating components and actuators.

Coupling strength

Coupling is strong when a small plant change causes a large change in the optimal controller, or when a small change in control capability causes a large change in the optimal plant. Strong coupling is common when:

- active control has substantial authority relative to passive forces;
- dynamic behavior dominates performance;

- plant constraints depend strongly on state and control trajectories;
- multiple time scales or energy domains interact;
- control exploits flexible, unstable, resonant, or energy-storing behavior; or
- sensing, sampling, delay, or preview limitations influence physical design.

Checkpoint

Why does coupling in the mass-spring-damper example weaken as allowable actuator force approaches zero? Why does unlimited actuator authority still fail to guarantee a practical design?

Activity 1.3: Quantifying Plant-Control Coupling

Activity 1.3: Quantifying Plant-Control Coupling

Consider the parameterized system-level objective

$$J(p, c; \gamma) = \frac{1}{2}ap^2 + \frac{1}{2}bc^2 + \gamma pc - r_p p - r_c c,$$

where $a > 0$, $b > 0$, and γ represents plant-control coupling strength.

1. Derive the condition on γ for J to be strictly convex.
2. Derive the optimal controller $c^*(p)$ for a fixed plant.
3. Compute the controller sensitivity

$$\frac{dc^*}{dp}.$$

4. Derive the optimal plant $p^*(c)$ for a fixed controller and compute

$$\frac{dp^*}{dc}.$$

5. Show that both sensitivities vanish when $\gamma = 0$.
6. Solve for the simultaneous optimum

$$p^*(\gamma), \qquad c^*(\gamma).$$

7. Use

$$a = 4, \qquad b = 3, \qquad r_p = 6, \qquad r_c = 5,$$

and evaluate the optimum for

$$\gamma \in \{0, 0.5, 1, 2, 3\}.$$

8. Plot

$$\left|\frac{dc^*}{dp}\right|$$

and the sequential performance loss as functions of γ.

9. Propose a dimensionless coupling index based on the Hessian cross-term and justify your choice.

Activity 1.4: Control Authority and the Value of Integration

Activity 1.4: Control Authority and the Value of Integration

Consider

$$m\ddot{x} + c\dot{x} + kx = u,$$

with

$$m = 1, \qquad x(0) = 1, \qquad \dot{x}(0) = 0.$$

Use the PD controller

$$u = -K_p x - K_d \dot{x}, \qquad 0 \le K_p \le K_{p,\max}, \qquad 0 \le K_d \le K_{d,\max},$$

with plant-design bounds

$$0.1 \le k \le 6, \qquad 0.05 \le c \le 3.$$

Define

$$A_{\mathrm{cl}} = \begin{bmatrix} 0 & 1 \\ -(k + K_p) & -(c + K_d) \end{bmatrix}.$$

For stable designs, let P solve

$$A_{\mathrm{cl}}^T P + P A_{\mathrm{cl}} + Q + K^T R K = 0,$$

where

$$Q = \begin{bmatrix} 1 & 0 \\ 0 & 0.05 \end{bmatrix}, \qquad R = 0.02, \qquad K = \begin{bmatrix} K_p & K_d \end{bmatrix}.$$

Define the total cost

$$J = \mathbf{x}_0^T P \mathbf{x}_0 + 0.02k^2 + 0.03c^2, \qquad \mathbf{x}_0 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}.$$

1. Derive the closed-loop stability conditions.
2. Explain why $\mathbf{x}_0^T P \mathbf{x}_0$ equals the infinite-horizon dynamic performance cost.

3. For

$$K_{p,\max} = K_{d,\max} \in \{0, 0.5, 1, 2, 4, 8\},$$

solve:

(a) the passive-first sequential problem; and

(b) the simultaneous CCD problem.

4. Plot the relative CCD advantage

$$\eta_{\mathrm{CCD}} = \frac{J_{\mathrm{seq}} - J_{\mathrm{CCD}}}{J_{\mathrm{seq}}}$$

against allowable control authority.

5. Determine whether the coupling becomes weak as the allowable controller gains approach zero.

6. Determine whether unlimited control authority drives the optimal passive stiffness and damping to their lower bounds.

7. Add the actuator-effort regularization

$$\beta(K_p^2 + K_d^2), \qquad \beta \in \{0, 10^{-3}, 10^{-2}, 10^{-1}\},$$

and repeat the study.

8. Explain why strong control authority does not automatically imply that the most active design is the most practical design.

A Quantitative Example

This example compares sequential design and CCD for a mass-spring-damper system. It demonstrates a mechanism, not a universal percentage improvement.

Model and objective

Let $m = 1$ and consider the free response from

$$x(0) = 1, \qquad \dot{x}(0) = 0.$$

The plant variables are spring stiffness k and passive damping c. The controller is

$$u(t) = -K_p x(t) - K_d \dot{x}(t),$$

with controller variables K_p and K_d. Define

$$J = \int_0^8 \left[x^2(t) + 0.05\dot{x}^2(t) + 0.02u^2(t)\right] dt + 0.02k^2 + 0.03c^2,$$

subject to

$$0.1 \le k \le 6, \quad 0.05 \le c \le 3, \quad 0 \le K_p \le 8, \quad 0 \le K_d \le 5.$$

The first two integral terms penalize motion, the third penalizes control effort, and the final terms are simplified plant costs.

Two workflows

Sequential workflow

1. Set $K_p = K_d = 0$ and optimize k and c.
2. Freeze the resulting plant.
3. Optimize K_p and K_d for the fixed plant using the full objective.

CCD workflow

1. Optimize k, c, K_p, and K_d together using the full objective.

The reported solutions are:

Method	k	c	K_p	K_d	J
Sequential	2.454	1.397	5.030	2.412	0.681
Control co-design	0.698	0.145	6.405	3.770	0.565

The co-designed objective is approximately 17% lower:

$$\frac{0.681 - 0.565}{0.681} \times 100\% \approx 17.0\%.$$

The important result is not only the lower objective. The optimized **plant is different**. The CCD solution uses less passive stiffness and damping and relies more on active control. The sequential plant was optimized to perform well without control and retained passive properties that were no longer optimal after the controller was added.

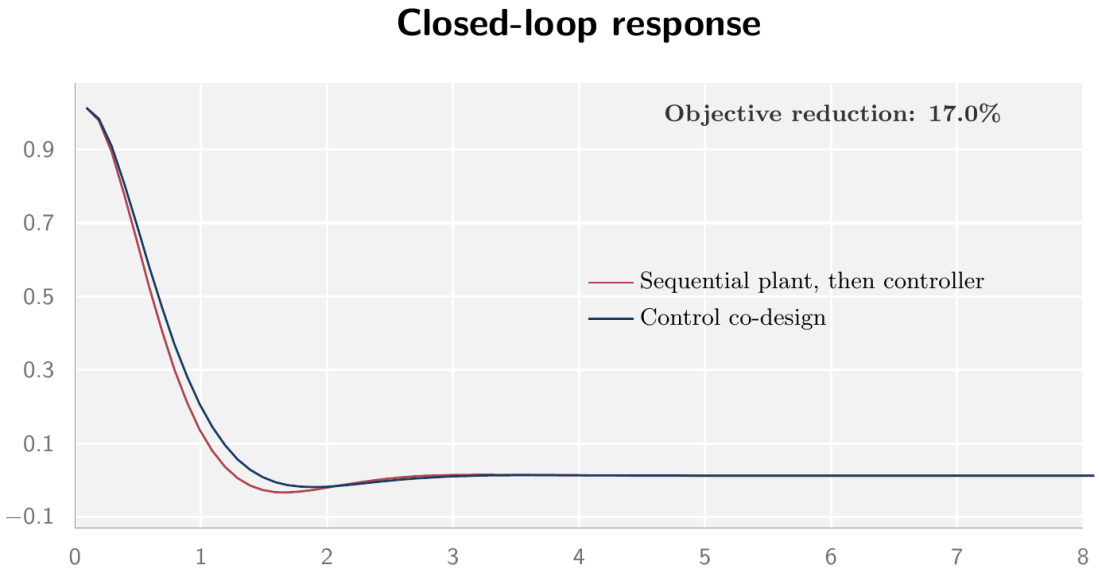


Figure 1.5: *
A coupled design landscape. The best controller depends on the plant, and the best plant depends on the controller.

Reproducible implementation

The following Python outline simulates the closed-loop response and evaluates the objective. It can be extended with `scipy.optimize.minimize` for both workflows.

```
import numpy as np
from scipy.integrate import solve_ivp

def evaluate(design, t_final=8.0):
    k, c, kp, kd = design

    def dynamics(t, state):
        x, v = state
        u = -kp*x - kd*v
        return [v, -c*v - k*x + u]  # m = 1

    t = np.linspace(0.0, t_final, 2001)
    sol = solve_ivp(dynamics, (0.0, t_final), [1.0, 0.0], t_eval=t,
                    rtol=1e -9, atol=1e -11)
    x, v = sol.y
    u = -kp*x - kd*v

    running_cost = x**2 + 0.05*v**2 + 0.02*u**2
    return np.trapz(running_cost, t) + 0.02*k**2 + 0.03*c**2
```

For a defensible computational study, report the optimizer, tolerances, initial guesses, bounds, integration grid, convergence status, and constraint residuals. Repeat the optimization from several initial guesses.

Interpreting the result

A co-designed system need not minimize every objective term. It minimizes the selected **total** objective. Here it accepts greater active-control responsibility in exchange for lower plant cost and a better system-level balance.

This example does **not** prove that CCD always improves performance by 17%, that software should replace passive hardware, or that the mathematical optimum is ready to build. A realistic study must also address:

- actuator saturation, bandwidth, power, and thermal limits;
- sensor noise, state estimation, and unmeasured states;
- uncertainty, unmodeled dynamics, and disturbance variation;
- failure modes and passive-safety requirements;
- hardware cost, maintenance, and reliability;
- causal real-time implementation; and
- higher-fidelity and experimental validation.

Key idea

CCD does not mean "replace hardware with software." It allocates functions across physical and control elements while respecting performance, cost, information, reliability, and implementation constraints.

Engineering Applications and CCD Screening

The formal field of CCD is relatively modern, but the engineering instinct is older. Designers have long traded passive behavior against control authority, sensing, and actuation.

Historical intuition

Wright brothers: controllability rather than passive stability. Nearly every aircraft designer before 1903 treated passive stability as a requirement: an airplane should fly straight on its own, with little or no pilot correction, so the wing surfaces and their arrangement were chosen to keep the center of pressure safely behind the center of gravity. The Wright brothers made the opposite choice. Their 1903 Flyer placed a small canard surface forward of the wing, shifting the center of pressure closer to the front of the aircraft and the center of gravity closer to the rear—making the airplane **statically unstable** in undisturbed flight. This was not an oversight; it was a deliberate physical-design decision made because the Wrights understood its dynamic consequence: an unstable airframe has faster, higher-bandwidth dynamics, so it responds to wind gusts and to the pilot's own control inputs more quickly than a stable one. The Wrights supplied the missing stability actively, through continuous pilot control learned first on gliders and then on bicycles—themselves unstable systems balanced by a trained rider. The plant decision (deliberately relocate the aerodynamic centers to speed up the dynamics) and the control decision (a trained human closing the loop in real time) were made as one coupled design, which is why the Wrights succeeded where stability-first competitors did not. The lesson is not that every system should be made unstable; it is that passive stability, agility, disturbance rejection, and control authority are system-level tradeoffs, and treating stability as non-negotiable can foreclose a better coupled design.

Charles Brush's wind turbine: control rather than added material. In 1887, Charles Brush—already known for DC dynamos, arc lighting, and lead-acid batteries—built the world's first automatically operating wind turbine for electricity generation, on his estate in Cleveland, Ohio. It was enormous for its time: a 17-meter-diameter rotor carrying 144 cedar blades, driving a generator that produced 12 kW at full load and 500 rpm, enough to charge 408 storage batteries and power roughly 350 incandescent lights, two arc lights, and three electric motors in Brush's mansion. In July 1887, a severe storm destroyed the turbine. Rebuilding gave Brush a genuine plant-versus-control choice: reinforce the tower and blades to survive the next storm, or use a control system to shed excess load before the structure was overstressed. He chose control. The rebuilt turbine used a large tail vane (18 m by 6 m), a smaller auxiliary vane mounted perpendicular to it, and a mechanism of weights and pulleys: under normal wind the large tail kept the rotor pointed into the wind for maximum energy capture, but under high wind the auxiliary vane dominated and turned the rotor away from the wind, automatically reducing mechanical loads and rotor speed without any operator present. Brush called it "self-regulating, requiring no person to attend it." It ran without failure for the next twenty years. The design substituted a control mechanism for the passive structural capacity a purely mechanical redesign would have required—an early, concrete instance of the same tradeoff this chapter has been building toward mathematically: control authority and passive capacity can substitute for one another, and the least-cost system may use less of the passive kind than a plant-only redesign would choose.

Modern application domains

Active vehicle suspension. Plant decisions include spring and damper properties, suspension geometry, unsprung mass, and actuator size. Control decisions include force commands, feedback gains, preview use, and estimation. Comfort, road holding, suspension travel, power, mass, cost, and passive safety must be evaluated together.

Wind turbines. Blade geometry, tower stiffness, drivetrain sizing, generator rating, and support structure interact with blade-pitch, generator-torque, and supervisory control. Control can reduce loads and increase energy capture, while structural changes alter modes and control bandwidth.

Floating offshore wind turbines make this coupling unusually direct and physical. A multi-megawatt rotor spinning at rate Ω_r with rotational inertia I_r carries substantial angular momentum $I_r\Omega_r$ about the rotor axis. When a wave changes the pitch angle β_p of the floating platform beneath the turbine, the spinning rotor responds the way any gyroscope does: conservation of angular momentum converts the platform's pitching motion into a torque that rotates the platform's yaw angle. The rotor, drivetrain, tower, floating platform, mooring system, and blade-pitch and torque control loops are therefore coupled through ordinary rigid-body mechanics, not merely through a shared performance objective—an aerodynamic disturbance (wind) and a hydrodynamic disturbance (waves) both reach the same rotor, and the rotor's own dynamics couple platform motion in one plane to structural loads in another. Programs such as the U.S. Department of Energy's ATLANTIS initiative were organized specifically around this observation: independent, disciplinary-best designs of the turbine and the floating platform cannot reach an optimal combined system, because the coupling mechanism is physical, not a modeling artifact that better bookkeeping could remove. A related, already-commercialized example is pitch-control design for fixed-tower turbines: a well co-designed pitch controller can substantially reduce tower-base bending loads and fatigue relative to a controller designed after the tower, which can in turn justify a lighter, less expensive tower—an active-versus-passive substitution in the same spirit as Brush's turbine, applied with modern optimization tools.

Robotics and mechatronics. All three design domains are visible at once in a robotic manipulator. Architecture decisions fix how many links and joints the manipulator has, how they are connected, and which joints are actuated—a two-link serial arm, a three-link serial arm, and a four-link parallel arm are different architectures for the same task, each admitting a different number of plant and control variables. Within a chosen architecture, link length, stiffness, mass, joint placement, gear ratio, motor size, and sensor placement are plant decisions, while trajectory planning and feedback control are control decisions; together they determine speed, accuracy, energy, payload, and robustness. Comparing architectures fairly requires solving the plant-and-control co-design problem within each candidate architecture, not reusing one architecture's tuned controller to judge another.

Marine and wave energy. Device geometry, hydrodynamics, power-take-off sizing, mooring, generator design, and control affect resonance, captured energy, extreme loads, and survivability. Control can change effective damping and stiffness, but realistic force, power, and information constraints are essential.

Other domains. CCD also appears in aircraft and spacecraft, active structures, buildings and microgrids, electric vehicles, thermal management, manufacturing systems, biomedical devices, and networked systems.

When CCD is most valuable

CCD deserves serious consideration when several of the following are true:

1. **Dynamic performance is central.** Transients, vibration, trajectory tracking, stability, fatigue, or energy conversion dominate the mission.
2. **Control authority is meaningful.** The actuator can materially alter loads, effective damping, resonance, or energy flow.
3. **Plant constraints depend on trajectories.** Stress, fatigue, temperature, stroke, battery degradation, and comfort depend on controlled time histories.
4. **Passive and active behavior can complement one another.** Neither a purely passive nor purely active solution is clearly best.
5. **Architecture is still flexible.** Components, connections, sensor locations, and actuator locations remain design choices.

6. **Information affects physical design.** Sampling, estimation, delay, communication, and preview change the preferred plant.

Practical screening checklist

Question	Why it matters
Does plant design materially change the dynamics?	The best controller may change across plant designs.
Can control materially change loads, energy, or feasibility?	The best plant may depend on the controller.
Are important constraints trajectory dependent?	Control may change plant constraint satisfaction.
Can passive and active elements substitute or complement one another?	Integrated allocation may reduce total cost.
Are actuator, sensor, information, or architecture choices flexible?	Early flexibility creates larger CCD opportunities.
Is a trustworthy coupled model available or developable?	Results are only as meaningful as the model and formulation.
Is the expected lifecycle benefit worth the added effort?	CCD carries modeling, software, and computational costs.

Screening exercise

Choose an active system familiar to you. Answer each question in the checklist with "yes," "no," or "uncertain," and support every answer with one physical mechanism or piece of evidence.

Activity 1.5: When Is Control Co-Design Worth Using?

Activity 1.5: When Is Control Co-Design Worth Using?

A design team may choose either a sequential process or a CCD process. Define the expected net value of CCD as

$$N = V - C - R,$$

where V is the expected lifecycle value of performance improvement, C is the additional modeling and computational cost of CCD, and R is the expected cost of model or implementation failure.
For a stylized dynamic system, assume

$$V = V_0 \left[1 - \exp\left(-\alpha A \Gamma\right)\right],$$

where $A \geq 0$ is normalized control authority and $\Gamma \geq 0$ is normalized plant-control coupling strength. Also assume

$$C = C_0 + C_1 n_p n_c + C_2 N_s,$$

where n_p and n_c are the numbers of plant and controller variables and N_s is the number of uncertainty scenarios, and

$$R = R_0 \exp(-\beta F), \qquad 0 \le F \le 1,$$

where F is a normalized model-fidelity and validation score. Use

$$V_0 = 100, \quad \alpha = 0.8, \quad C_0 = 5,$$
$$C_1 = 0.15, \quad C_2 = 0.5, \quad R_0 = 40, \quad \beta = 3.$$

1. Derive the break-even condition $N = 0$.
2. For

$$n_p = 8, \qquad n_c = 5, \qquad N_s = 10,$$

   compute the minimum product $A\Gamma$ required for CCD to have positive expected net value when $F = 0.8$.
3. Repeat the calculation for

$$F \in \{0.3, 0.5, 1\}.$$

4. Plot the break-even boundary in the (A, Γ) plane.
5. Determine how the break-even boundary changes when the number of uncertainty scenarios increases.
6. Explain why high coupling and high control authority favor CCD.
7. Explain why poor model fidelity can eliminate the expected value of CCD even when the numerical optimization predicts a large performance improvement.
8. Use the model to compare:
   (a) a legacy machine with nearly fixed hardware and weak control authority;
   (b) a flexible robot with strong sensing and actuation;
   (c) a new floating wind platform with uncertain hydrodynamics; and
   (d) a regulated product with fixed plant architecture.
9. Discuss the limitations of using a single scalar net-value model to decide whether CCD should be pursued.

What CCD Is Not—and How to Practice It Responsibly

Common misconceptions

"CCD means tuning a controller for several candidate plants.". That activity can support CCD, but it becomes co-design only when plant decisions are coordinated with control decisions under a shared system-level objective and constraints. Controller tuning after plant selection remains sequential.

"Putting all variables in one optimizer automatically creates CCD.". A variable list is not a formulation. The model must represent how plant and control decisions affect dynamics, objectives, information, and constraints. Integrated code with missing physics is not integrated engineering.

"CCD always requires simultaneous optimization.". CCD can use simultaneous, nested, iterative, or architecture-based strategies. The defining feature is coordinated system design, not one solver architecture.

"Open-loop optimal control is directly implementable feedback.". Open-loop optimal control (OLOC) can reveal performance limits and useful dynamic behavior, but it may assume future information unavailable in real time. Implementation requires causal feedback, sensing, estimation, computation, and robustness analysis. Controllers can be classified by how much of that information they actually have when they act:

- **Complete horizon:** the controller has full information about the environment over the entire time horizon in advance—the assumption behind OLOC. This is appropriate for offline planning problems (a spacecraft reentry trajectory, a manipulator path) but not for a controller that must act causally against a disturbance it cannot foresee.
- **Instantaneous:** the controller has information only about the current instant, the classical assumption behind ordinary feedback control (tracking and regulation).
- **Limited horizon:** the controller has information over a short window into the future and re-solves at each step—the assumption behind model predictive control (MPC). Limited-horizon control is a practical middle ground: it can approach complete-horizon performance as the horizon grows, and it recovers instantaneous, purely reactive behavior as the horizon shrinks toward zero.

The gap between complete-horizon and instantaneous information is not a modeling nuisance—it has a measurable performance cost. In one documented active-suspension co-design study, an idealized fully active open-loop benchmark was progressively made more realistic in four steps: an ideal active actuator, an idealized semi-active damper, a semi-active damper with a realistic hysteretic force-current model, and finally a full-state-feedback closed-loop controller with optimized gains acting on real-time measurements. Each step degraded the achievable objective—by roughly $1.8\times$, then $2.6\times$, then $2.1\times$ relative to the previous step—so that the final, genuinely implementable closed-loop design was about $9.7\times$ worse than the open-loop upper bound on performance. None of these steps involved a design mistake; each one simply removed an assumption (unlimited information, an idealized actuator, unconstrained control) that was not physically available. This is why a lower OLOC-based objective value is a **benchmark**, not a deliverable: it tells you the best that could ever be achieved if implementation were free, which is a useful number to know but not the number a real controller can deliver. Model predictive control's tunable prediction horizon gives one systematic way to study exactly how much of that $9.7\times$ gap can be recovered as more (but still finite) future information is made available to the controller—a question you will revisit quantitatively in later chapters.

"A lower numerical objective proves a better product.". A lower value is meaningful only for the chosen model, scenarios, objective, constraints, and numerical accuracy. An optimizer can exploit unrealistic assumptions or omitted failure modes.

Limitations and engineering responsibilities

Modeling burden. A useful CCD model must be dynamically meaningful and flexible with respect to plant variables. A fixed-plant control model may be accurate but unable to predict how geometry changes dynamics. A plant-design model may allow geometry changes but simplify the dynamics that matter for control.

Computational burden. Dynamic optimization can require many simulations and thousands of time-dependent variables. Efficient derivatives, sparsity, model reduction, surrogate models, and suitable coordination strategies can become essential.

Formulation sensitivity. The optimized design can change substantially with objective weights, scenarios, bounds, and constraints. Engineers must justify these choices and test sensitivity rather than treating them as neutral inputs.

Uncertainty and robustness. A nominally optimal pair may fail when parameters, disturbances, sensors, or models differ from assumptions. Manufacturing variation, environmental uncertainty, noise, degradation, and unmodeled dynamics must eventually be included.

Implementation gap. An elegant controller may be impossible to implement because of sampling, communication, causality, computation, sensor availability, or actuator limits. CCD should progress from idealized exploration toward realizable closed-loop control and hardware validation.

Organizational challenge. Cross-disciplinary teams must agree on the mission, model interfaces, variable ownership, data standards, and verification procedures. Mathematical integration does not remove the need for engineering communication.

Four kinds of credibility

A trustworthy CCD result requires **physical credibility**, **formulation credibility**, **numerical credibility**, and **implementation credibility**.

A practical CCD mindset

Before choosing a solver, ask:

1. What is the mission of the complete system?
2. Which decisions are genuinely free?
3. How do those decisions influence the dynamics?
4. What information does the controller actually possess?
5. Which physical limits are essential?
6. How will the result be verified?

A practical workflow is:

1. define the system mission and requirements;
2. identify plant, control, information, and architecture decisions;
3. construct a coupled dynamic model;

4. define system-level objectives and constraints;
5. select a coordination and numerical strategy;
6. solve and verify the optimization problem;
7. interpret the physical-control tradeoffs; and
8. move toward realizable control and higher-fidelity validation.

Verification should include unit checks, baseline reproduction, comparison with a sequential formulation, multiple starting points, constraint and trajectory inspection, and validation with higher-fidelity models or experiments.

Activity 1.6: Ideal Actuator versus Implementable Actuator

Activity 1.6: Ideal Actuator versus Implementable Actuator

Consider the plant

$$\dot{x} = -x + f_a + d(t), \qquad x(0) = 0, \qquad d(t) = \sin(4t).$$

An idealized study assumes $f_a(t) = u_c(t)$. A more realistic actuator satisfies

$$\dot{f}_a = \omega_a(u_c - f_a), \qquad |u_c(t)| \le F_{\max},$$

with actuator design bounds

$$0.5 \le F_{\max} \le 5, \qquad 2 \le \omega_a \le 50.$$

Minimize

$$J = \int_0^{10} \left(x(t)^2 + 0.02u_c(t)^2\right) dt + 0.05F_{\max}^2 + \frac{0.1}{\omega_a}.$$

1. For the ideal actuator, derive the control command that would cancel the disturbance exactly while maintaining $x(t) = 0$.
2. Determine the minimum ideal actuator capacity required for exact disturbance cancellation.
3. For the finite-bandwidth actuator, derive the steady-state sinusoidal amplitude and phase of $f_a(t)$ when

$$u_c(t) = -\sin(4t).$$

4. Derive the resulting steady-state amplitude of $x(t)$.
5. Evaluate the state-response amplitude for

$$\omega_a \in \{2, 4, 8, 16, 32, 50\}.$$

6. Determine the smallest actuator bandwidth for which the state amplitude is within 5% of the ideal-actuator prediction.

7. Optimize $F_{\max}$ and ω_a for the realistic model using a suitable numerical method.
8. Explain why a low objective obtained with the ideal actuator model is not sufficient evidence of a useful CCD result.
9. Identify at least three other implementation effects that could invalidate an apparently strong CCD solution.

Chapter Summary, Key Terms, and Problems

Chapter summary

- Active engineering systems combine a plant, sensors, controller, actuators, and environment.
- Architecture, plant, and control are three distinct design domains—which components exist and how they connect, their continuous sizing, and how control action is generated—and the same premature-fixing hazard that motivates CCD at the plant-control level applies one level up, to architecture.
- Sequential design fixes the plant before the final controller is known and can eliminate better plant-controller combinations; the Millennium Bridge and combine-harvester header-height cases show this is a documented engineering failure mode, not a hypothetical one.
- CCD coordinates physical and control decisions through shared dynamics, objectives, information assumptions, and constraints, and is a specialization of the broader MDSDO/MDO landscape.
- CCD itself spans three complementary methodologies—control-inspired reasoning, mathematical co-optimization and co-simulation—that this course develops primarily through co-optimization.
- Plant-control coupling exists when changing the plant changes the best controller and changing control capability changes the best plant; the five plant-objective cases explain precisely how an incomplete objective can hide this dependence.
- Iterated sequential design is a block-coordinate-descent algorithm; it can reach the CCD optimum only when both subproblems use a consistent, complete objective, and even then convergence slows as plant-control coupling strengthens.
- Passive and active elements can substitute for or complement one another, but differ in reliability, information, energy, and implementation requirements.
- CCD is especially valuable when dynamic performance is central, control authority is meaningful, constraints depend on trajectories, and architecture remains flexible.
- Open-loop optimal control gives a performance benchmark, not a deliverable; the gap to an implementable, limited-information controller can be an order of magnitude and is governed by how much information the controller is assumed to have (complete, instantaneous, or limited horizon).
- A lower objective value is meaningful only within a credible model, formulation, numerical solution, and implementation pathway.

Key terms

Term	Meaning
Active system	A physical system whose behavior is modified using sensing, computation, and actuation.
Architecture	The elements (components) contained in a system and the relationships among them; a discrete decision distinct from, and prior to, plant sizing and control design. Not to be confused with "CCD architecture" (nested, simultaneous), a numerical solution strategy.
Plant	The physical object or process being controlled.
Plant design variable	A decision describing physical form, parameters, or components, given a fixed architecture.
Control design variable	A decision describing how control action is generated.
State	A variable needed to describe the internal dynamic condition.
Control input	A commanded physical action applied by an actuator.
Disturbance	An external input affecting behavior but not selected by the controller.
Sequential design	A workflow in which one design domain is fixed before another is optimized.
Control co-design	Integrated design of plant and control decisions for a complete active system.
Plant-control coupling	Dependence of optimal plant decisions on control decisions and vice versa.
Control authority	The ability of an actuator and controller to alter system behavior.
System-level objective	A performance measure representing the mission and tradeoffs of the complete system.
Feasible design	A design satisfying all stated constraints.
System-suboptimal	Feasible, but not optimal for the complete integrated problem.
MDSDO	Multidisciplinary dynamic system design optimization: MDO specialized to systems whose time-evolving state is critical to performance. CCD is an MDSDO problem that also treats the controller as a first-class design object.
Plant-objective case	One of five ways a plant-design objective can (mis)represent the true system objective, ranging from exact (Cases 1 and 4) to static or approximate (Cases 2, 3, and 5).
Block coordinate descent (BCD)	An optimization algorithm that alternates optimizing disjoint blocks of variables; iterated sequential design is a BCD instance and converges to the CCD optimum only under specific conditions.
Information horizon	The span of time over which a controller has usable information when it acts: complete (offline, full foresight), instantaneous (classical feedback), or limited (MPC-style receding horizon).

Problems

1. **Quantifying plant–control coupling.** An actively controlled oscillator satisfies $m\ddot{x} + c\dot{x} + kx = u + w$ and uses $u = -K_p x - K_d \dot{x}$. For $J = \int_0^\infty (q_x x^2 + q_v \dot{x}^2 + r u^2)\, dt + \gamma_m m + gamma_k k$, derive a local cross-sensitivity measure that quantifies coupling between (m, k) and (K_p, K_d) at a stable design and explain how it predicts the potential value of CCD.

2. **Sequential versus simultaneous design.** For the static surrogate $J(p, c) = \frac{1}{2}a(p - p_0)^2 + bpc + \frac{1}{2}d(c - c_0)^2$ with $a, d > 0$ and $ad > b^2$, derive the exact simultaneous optimizer and the optimizer produced by one plant-then-control sequential pass, then obtain a closed-form expression for their objective gap.

3. **Control-authority allocation.** A positioning system has effective stiffness $k + K_p = q > 0$ and cost $C(k, K_p) = ak^2 + bK_p^2 + \gamma k K_p$, with $k \geq 0$ and $0 \leq K_p \leq \bar{K}_p$. Derive the globally optimal passive–active allocation as a piecewise function of $(a, b, \gamma, q, \bar{K}_p)$.

4. **A system-level objective from physical units.** A battery-electric vehicle has longitudinal dynamics $m\dot{v} = F_t - \frac{1}{2}\rho C_d A v^2 - C_r m g - m g \sin\theta$ and battery power $P_b = F_t v/(\eta_d \eta_m)$. Construct one dimensionally consistent Bolza objective that trades trip time, electrical energy, battery mass, and terminal-speed error, and justify a normalization that makes its weights interpretable.

5. **Architecture screening under common assumptions.** Compare passive, semi-active, and fully active suspension architectures for $m_s \ddot{z}_s = -k_s(z_s - z_u) - c_s(\dot{z}_s - \dot{z}_u) + F_a$ under the same road input, packaging envelope, ride metric, and actuator-power model by formulating a single mixed-discrete CCD problem whose feasible sets make the comparison fair.

6. **Failure of an incomplete plant objective.** Let the true objective be $J(p, c) = (p-1)^2 + (c-p)^2 + \rho c^2$, while a plant team minimizes only $J_p(p) = (p-1)^2$ before the control team selects c. Derive the resulting sequential design and the simultaneous CCD design, then determine for which $\rho > 0$ the relative performance loss exceeds ten percent.

7. **Passive safety as a coupled requirement.** The actuator in $m\ddot{x} + c\dot{x} + kx = u$ may fail at an unknown time, after which $u = 0$. Formulate a CCD problem that minimizes nominal closed-loop performance while guaranteeing a prescribed exponential decay rate after failure, expressing the passive-safety condition as a constraint on the plant parameters.

8. **Information as a design variable.** A vehicle suspension controller receives a preview $w(t + \tau)$ of road displacement at sensing cost $C_s(\tau) = c_0 + c_1 \tau^2$, while the best achievable closed-loop cost is $J^*(p, \tau)$. Formulate a value-of-information test based on $\partial J^*/\partial \tau$ that determines whether the optimal design uses zero, finite, or maximum available preview.

9. **Iterated sequential design.** For a twice continuously differentiable strongly convex objective $J(p, c)$, represent alternating exact minimization over p and c as block coordinate descent and derive a local linear convergence factor in terms of the Hessian blocks H_{pp}, H_{pc}, and H_{cc}.

10. **CCD study definition.** For a two-link robot with $M(q, p)\ddot{q} + C(q, \dot{q}, p)\dot{q} + g(q, p) = B(p)u$, formulate a reproducible control co-design study with link dimensions as plant variables and a feedback policy as the control design, giving one complete objective, physically meaningful constraints, an information pattern, and a sequential baseline in one unified mathematical statement.

References and further reading

1. Garcia-Sanz, M. (2019). Control co-design: An engineering game changer. *Advanced Control for Applications: Engineering and Industrial Systems, 1*(1), Article e18. DOI: 10.1002/adc2.18

2. Allison, J. T., & Herber, D. R. (2014). Multidisciplinary design optimization of dynamic engineering systems. *AIAA Journal, 52*(4), 691–710. DOI: 10.2514/1.J052182

3. Allison, J. T., Guo, T., & Han, Z. (2014). Co-design of an active suspension using simultaneous dynamic optimization. *Journal of Mechanical Design, 136*(8), Article 081003.

4. Herber, D. R., & Allison, J. T. (2019). Nested and simultaneous solution strategies for general combined plant and control design problems. *Journal of Mechanical Design, 141*(1), Article 011402. DOI: 10.1115/1.4040705

5. Bayat, S., Peterson, C., Lee, Y. H., Iori, J., & Allison, J. T. (2026). Advancing wind turbines through control co-design: An integrative review. *Applied Energy, 416*, Article 127951. ScienceDirect article

6. Deshmukh, A. P., Herber, D. R., & Allison, J. T. (2015). Bridging the gap between open-loop and closed-loop control in co-design: A framework for complete optimal plant and control architecture design. In *2015 American Control Conference (ACC)* (pp. 4916–4922).

7. Bayat, S., & Allison, J. T. (2026). Control co-design with varying available information applied to vehicle suspensions. *ASME Journal of Dynamic Systems, Measurement, and Control, 148*(1), Article 011013. DOI: 10.1115/1.4069918

8. Herber, D. R. (2017). *Advances in Combined Architecture, Plant, and Control Design* (Doctoral dissertation). University of Illinois at Urbana-Champaign.

Completion checkpoint

You are ready for the next chapter when you can convert a conventional plant-then-controller workflow into a coupled problem statement and explain which physical mechanisms create value—or risk—from integration.

Dynamic Systems and Feedback Control

Chapter 2: Dynamic Systems and Feedback Control

Modeling, analysis, and feedback ideas needed for control co-design

> A controller can only improve a system through the dynamics of the plant it acts on. To do control co-design well, we must understand those dynamics clearly.

Control co-design (CCD) is fundamentally about **dynamic systems**. A plant is not just a static object with dimensions and mass. It moves, stores energy, dissipates energy, responds to disturbances, and interacts with sensors and actuators over time. Controllers also act over time: they measure, decide, and command. Because both the plant and controller affect time-domain behavior, CCD requires fluency in dynamic-system modeling and feedback ideas.

This chapter develops that foundation. We begin with physical system modeling and the language of states, inputs, outputs, and disturbances. We then introduce state-space representation, discuss open- and closed-loop systems, explain stability, and show how feedback modifies system behavior. The chapter closes by connecting these ideas back to plant design: changing stiffness, mass, damping, geometry, inertia, or actuator placement changes the system dynamics, which in turn changes what controller is best.

Learning objectives

After completing this chapter, you should be able to:

1. explain why dynamic-system models are needed in control co-design;
2. identify states, inputs, outputs, and disturbances for a physical system;
3. derive a state-space model from differential equations;
4. distinguish open-loop and closed-loop systems;
5. explain equilibrium, stability, and asymptotic stability for linear systems;
6. show how feedback changes closed-loop dynamics; and
7. explain how plant design choices modify the dynamic model and therefore influence control design.

Chapter roadmap

Physical model → States and signals → State space → Feedback → Stability → Plant–control interaction

Physical System Modeling

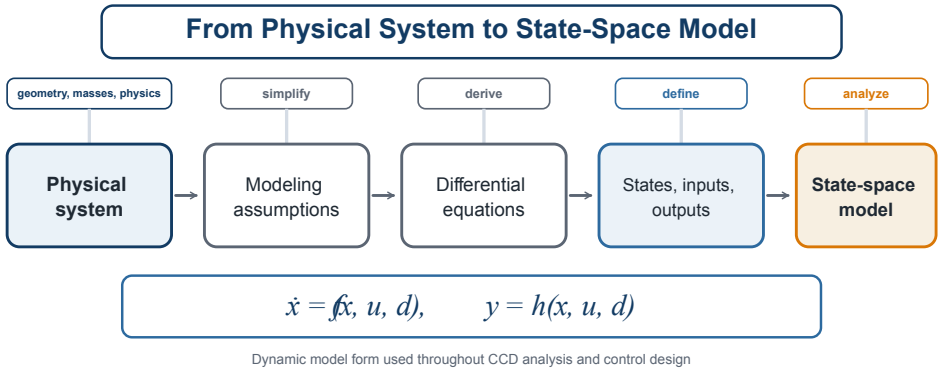
Why modeling matters

Every control problem begins with a model, whether that model is explicit or implicit. Even when a controller is tuned experimentally, the engineer still carries an internal mental model of what the system does. In CCD, the model plays an even larger role: it is the bridge between plant design variables and control design variables.

A good model should be:

- **physically meaningful**, so that design-variable changes have understandable effects;
- **simple enough to analyze and optimize**, especially in early design stages;
- **rich enough to capture the dominant dynamics** that matter for control and performance; and
- **well matched to the decisions being made**. A conceptual design study does not need the same fidelity as a certification-grade simulation.

There is no perfect model. There are only models that are more or less useful for a given decision. The art of dynamic modeling lies in retaining the right physics while excluding details that do not materially affect the design question.



A typical modeling workflow for dynamic systems used in CCD and control analysis. The process begins with the physical system and ends with a dynamic model that can be analyzed, simulated, and used for design.

A common modeling pipeline begins with a physical system and its governing physics. We then make assumptions: rigid body or flexible body? Linear or nonlinear? Lumped parameters or distributed parameters? Small motions or large motions? These assumptions lead to differential equations. Finally, we define states, inputs, outputs, and disturbances and express the model in a convenient form such as state space.

Levels of model fidelity

Dynamic models are often built at different levels of fidelity:

1. **Conceptual models** capture the dominant energy-storage and dissipation mechanisms. They are excellent for teaching, intuition, and early optimization.
2. **Intermediate models** add realism such as actuator dynamics, sensor dynamics, multiple degrees of freedom, or saturation.
3. **High-fidelity models** incorporate flexible modes, nonlinearities, aerodynamic or hydrodynamic effects, contact, constraints, and implementation details.

Conceptual and intermediate models build the understanding needed to use high-fidelity models responsibly.

Coupling across energy domains: multidisciplinary analysis

Many CCD-relevant systems are not built from a single set of governing equations. A robotic system, for instance, couples mechanical dynamics with electrical actuator dynamics; a marine energy device couples hydrodynamics with the electrical or hydraulic dynamics of its power take-off. If each energy domain were modeled independently, the interaction between domains would be lost — an electric motor's rotor state could appear to change instantaneously, when in fact the mechanical load it drives constrains how fast it can actually respond.

Suppose two energy domains, labeled α and β, each have their own state trajectory, $\boldsymbol{\xi}_\alpha(t)$ and $\boldsymbol{\xi}_\beta(t)$, governed by their own derivative functions:

$$\dot{\boldsymbol{\xi}}_\alpha(t) = \mathbf{f}_\alpha\big(\boldsymbol{\xi}_\alpha(t), \boldsymbol{\xi}_\beta(t), \mathbf{u}_\alpha(t), t\big),$$

$$\dot{\boldsymbol{\xi}}_\beta(t) = \mathbf{f}_\beta\big(\boldsymbol{\xi}_\alpha(t), \boldsymbol{\xi}_\beta(t), \mathbf{u}_\beta(t), t\big).$$

The two domains are coupled because $\dot{\boldsymbol{\xi}}_\alpha$ depends on $\boldsymbol{\xi}_\beta$ and $\dot{\boldsymbol{\xi}}_\beta$ depends on $\boldsymbol{\xi}_\alpha$. Building and solving this kind of coupled model — capturing how each energy domain affects the dynamics of the others — is called **multidisciplinary analysis (MDA)**. Once both domains are captured, the stacked state $\boldsymbol{\xi} = [\boldsymbol{\xi}_\alpha^T \; \boldsymbol{\xi}_\beta^T]^T$ and stacked input $\mathbf{u} = [\mathbf{u}_\alpha^T \; \mathbf{u}_\beta^T]^T$ recombine the two coupled equations into a single first-order model $\dot{\boldsymbol{\xi}}(t) = \mathbf{f}(\boldsymbol{\xi}(t), \mathbf{u}(t), t)$ of the same form used throughout this chapter. The DC motor of Activity 2.1 is exactly this construction: its electrical current equation and mechanical speed equation are coupled through the back-EMF term $K_e\omega$ and the torque term $K_t i$, and stacking (θ, ω, i) into one state vector is an MDA step performed almost automatically.

CCD connection

MDA is not optional bookkeeping. A control or plant design decision made using only one domain's equations, with the other domain's feedback path left out, is analyzing a different (and usually easier) system than the one that will actually be built.

When the model is not a clean ODE: differential-algebraic equations. Combining domains, or imposing a physical or design requirement as an equality among the states, is not always reducible to $\dot{\boldsymbol{\xi}} = \mathbf{f}(\boldsymbol{\xi}, \mathbf{u}, t)$ without extra work. Sometimes a modeling assumption, a rigid physical constraint, or a path constraint that has become active removes a degree of freedom from the system and introduces an algebraic — not differential — relationship among the variables. The resulting model, in semi-explicit form, is

$$\dot{\boldsymbol{\xi}}(t) = \mathbf{f}\big(\boldsymbol{\xi}(t), \boldsymbol{\gamma}(t), \mathbf{u}(t), t\big), \qquad \mathbf{0} = \mathbf{f}_a\big(\boldsymbol{\xi}(t), \boldsymbol{\gamma}(t), \mathbf{u}(t), t\big),$$

where $\boldsymbol{\gamma}(t)$ is an **algebraic variable**: unlike a state, its own time derivative never appears in the model, and its value at time t is instead pinned down by the algebraic equation $\mathbf{f}_a(\cdot) = \mathbf{0}$. A system of this form is called a **differential-algebraic equation (DAE)**. If the algebraic equation can, at least locally, be solved for $\boldsymbol{\gamma}(t)$ — equivalently, if the Jacobian of $\mathbf{f}_a$ with respect to $\boldsymbol{\gamma}$ is nonsingular — the DAE is called **index-1**, where the index counts how many times the algebraic equation must be differentiated with respect to time before the whole system reduces to an ordinary differential equation. An index-1 DAE can, in principle, always be converted to the explicit state-space form used elsewhere in this chapter; the conversion is simply not free, and some solution methods work directly with the DAE instead of performing it.

This situation is common, not exotic. A rigid mechanical constraint between two bodies, an incompressible fluid, or an inequality constraint (on stress, temperature, position, or actuator force) that has just become active can each remove a degree of freedom this way. In every one of these cases, some variable that used to be a

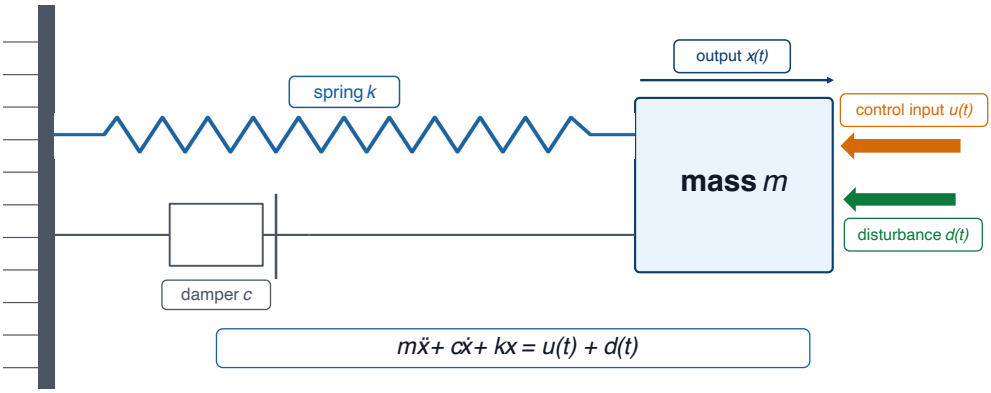
free state must become an algebraic variable, satisfying a constraint rather than its own dynamics. In actively controlled systems, control inputs are normally treated as independent quantities, while the algebraic variables introduced by active constraints must still satisfy the underlying physics.

CCD connection

Whether a model reduces cleanly to $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u})$ or is naturally a DAE is itself a modeling decision with consequences for CCD. Direct-transcription optimization methods, used widely for combined plant-and-control design, can pose the differential and algebraic equations as equality constraints directly, without first eliminating $\gamma(t)$ by hand — one reason DAE-aware formulations matter for later chapters on CCD architectures and numerical solution strategies.

A first example: mass–spring–damper system

The mass–spring–damper system contains nearly every idea we need: inertia, restoring force, damping, actuation, disturbance response, and measurable output.



A single-degree-of-freedom mass–spring–damper system with a control input and a disturbance.

Applying Newton's second law gives

$$m\ddot{x}(t) + c\dot{x}(t) + kx(t) = u(t) + d(t),$$

where m is mass, c is the damping coefficient, k is spring stiffness, $x(t)$ is displacement, $u(t)$ is the control force, and $d(t)$ is an external disturbance force.

CCD connection

Changing m, c, or k changes the dynamic behavior. Since a controller acts through those dynamics, the best controller also changes. This is the essence of plant–control interaction.

Activity 2.1: Full and Reduced Models of a DC Motor

Activity 2.1: Full and Reduced Models of a DC Motor

An armature-controlled DC motor is modeled by

$$J\dot{\omega} + b\omega = K_t i - \tau_L,$$
$$L\dot{i} + Ri = v - K_e\omega.$$

Use

$$J = 0.02\ \mathrm{kg\,m^2}, \qquad b = 0.08\ \mathrm{N\,m\,s},$$

and

$$L = 0.015\ \mathrm{H}, \qquad R = 1.2\ \Omega, \qquad K_t = K_e = 0.25.$$

1. Using

$$\mathbf{x} = \begin{bmatrix} \theta & \omega & i \end{bmatrix}^T,$$

derive the complete state-space model with voltage v as the control input and load torque τ_L as the disturbance.

2. Derive the characteristic polynomial of the unforced state matrix.
3. Explain why one eigenvalue is located at the origin when shaft position is included as a state.
4. Remove the position state and derive the two-state speed-current subsystem.
5. Prove that the speed-current subsystem is asymptotically stable for positive

$$J,\ b,\ L,\ R,\ K_t,\ K_e.$$

6. Set $L\dot{i} \approx 0$ and derive the reduced first-order speed model.
7. Derive the reduced-model pole and compare it with the slow eigenvalue of the full speed-current model.
8. Simulate the full and reduced models for a 12-V step input and zero load torque. Compare:
   (a) motor speed;
   (b) current;
   (c) steady-state speed; and
   (d) dominant time constant.
9. Repeat the comparison for

$$L \in \{0.005,\ 0.015,\ 0.08\}\ \mathrm{H}.$$

10. Define a quantitative criterion for deciding when the reduced model is acceptable.
11. Explain how changing rotor inertia J, inductance L, or torque constant K_t changes both plant performance and the control-design problem.

States, Inputs, Outputs, and Disturbances

States

A **state** is the minimum information needed at time t so that, together with future inputs, we can predict future behavior. Dynamic systems have memory; the state records the information needed to represent that memory.

For many mechanical systems, positions and velocities are natural states. For electrical systems, currents and voltages across energy-storing elements are common states. For thermal systems, temperatures may be states.

For the mass–spring–damper example, a convenient state choice is

$$x_1(t) = x(t), \qquad x_2(t) = \dot{x}(t).$$

These two variables are sufficient to predict future motion when future inputs are known.

Inputs

An **input** is a signal that can influence the system. In control problems, we usually distinguish between:

- **control inputs**, which are commanded by the controller; and
- **exogenous inputs**, such as disturbances or reference signals.

For the mass–spring–damper system, $u(t)$ is the control input and $d(t)$ is a disturbance. For a vehicle, steering angle or brake torque may be control inputs, while road irregularities and wind gusts are disturbances.

Outputs

An **output** is a quantity we care to measure, estimate, regulate, track, or report. Outputs do not need to equal states. Sometimes an output is a state, such as position. Sometimes it is a combination of states, such as acceleration or strain.

A suspension designer may care about body acceleration, tire deflection, and suspension travel. A wind-turbine designer may care about rotor speed, generated power, and tower-base bending moment. These are outputs because they relate to performance and constraints.

Disturbances

A **disturbance** is an external influence that affects the dynamics but is not under direct control. Disturbances may be measurable, partially measurable, or unmeasurable. Typical examples include road bumps, wave excitation, wind variations, payload changes, and unmodeled loads.

Disturbance rejection is one of the main reasons feedback is valuable. If disturbances were absent and the model were exact, open-loop control would often be enough. Real systems are rarely that kind.

Examples from engineering systems

System	Possible states	Control inputs	Outputs of interest	Disturbances
Mass–spring–damper	Displacement, velocity	Force u	Displacement, velocity, acceleration	External force
Active suspension	Body and wheel positions and velocities	Actuator force	Ride comfort, road holding, suspension travel	Road profile
Robot arm	Joint angles, joint rates	Motor torques	End-effector position, tracking error	Payload changes, contact forces
Wind turbine	Rotor speed, structural states, actuator states	Pitch angle, generator torque	Power, loads, speed regulation	Wind speed and turbulence
Marine energy device	Heave and pitch states, PTO states	PTO force or damping command	Absorbed power, motions, loads	Wave excitation

Key idea

States describe the system's internal memory. Inputs influence the system. Outputs are what we monitor or care about. Disturbances are external influences that alter behavior. CCD depends on identifying these quantities clearly because plant and control decisions can change them.

State-Space Representation

Why state space?

Single differential equations and transfer functions are useful, but state-space representation is more general and more natural for CCD. It handles:

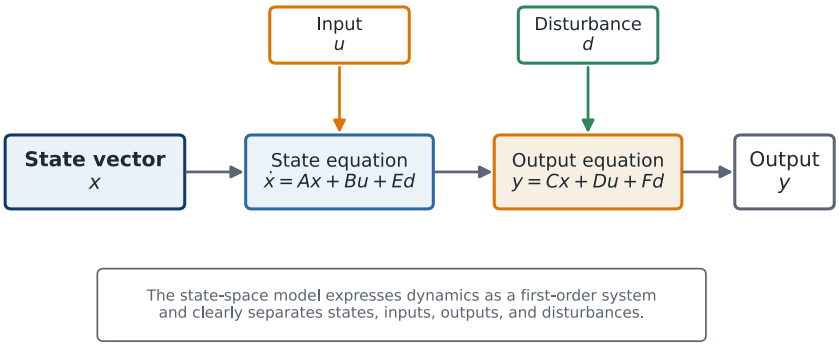
- multi-input and multi-output systems;
- systems with many states;
- nonlinear dynamics;
- time-varying parameters;
- state constraints and modern controller design methods; and
- optimization formulations used in control and CCD.

The general nonlinear continuous-time state-space model is

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), \mathbf{d}(t), t), \qquad \mathbf{y}(t) = \mathbf{h}(\mathbf{x}(t), \mathbf{u}(t), \mathbf{d}(t), t).$$

For a linear time-invariant (LTI) system, this becomes

$$\dot{\mathbf{x}}(t) = A\mathbf{x}(t) + B\mathbf{u}(t) + E\mathbf{d}(t), \qquad \mathbf{y}(t) = C\mathbf{x}(t) + D\mathbf{u}(t) + F\mathbf{d}(t).$$



The state-space viewpoint separates internal dynamic variables from control inputs, disturbances, and measured outputs.

When the model does not start in this form

The explicit state-space forms in () and () assume that every derivative $\dot{\mathbf{x}}(t)$ can be written directly as a function of the states, inputs, and disturbances. That assumption is convenient, but it is not automatic. As introduced in the previous section, coupling two energy domains through multidisciplinary analysis, or activating a path constraint on stress, position, or actuator force, can introduce an algebraic variable $\gamma(t)$ tied to the states and inputs through an algebraic equation rather than a differential one, producing an index-1 differential-algebraic equation (DAE) instead of a plain ODE.

Getting from a DAE to the state-space form used in this chapter requires eliminating $\gamma(t)$ — solving the algebraic constraint for $\gamma(t)$ in terms of $\xi(t)$ and $\mathbf{u}(t)$, then substituting the result back into the differential equation. For an index-1 DAE this elimination is always possible in principle, but it can be algebraically involved, and it must be redone whenever the constraint that generated $\gamma(t)$ changes (for example, when a different inequality

constraint becomes the active one). Some numerical and optimization methods used later in this course work with the DAE directly rather than performing this elimination, which is one reason the distinction between an ODE model and a DAE model matters beyond bookkeeping.

Key idea

State-space representation is the destination, not always the starting point. A physically derived model may arrive as a DAE; deriving the A, B, C, D matrices (or the nonlinear $\mathbf{f}(\cdot)$, $\mathbf{h}(\cdot)$) implicitly assumes that elimination step has already been done.

Converting the mass–spring–damper model

Start from () and define

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

Then

$$\dot{x}_1 = x_2, \qquad \dot{x}_2 = -\frac{k}{m}x_1 - \frac{c}{m}x_2 + \frac{1}{m}u + \frac{1}{m}d.$$

Therefore,

$$\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\end{bmatrix} = \begin{bmatrix}0 & 1\\ -k/m & -c/m\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix} + \begin{bmatrix}0\\ 1/m\end{bmatrix}u + \begin{bmatrix}0\\ 1/m\end{bmatrix}d.$$

If the output is displacement only,

$$y = \begin{bmatrix}1 & 0\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix}.$$

If the output includes displacement and velocity, then $\mathbf{y} = [x_1\ x_2]^T$ and $C = I$.

The plant design variables m, c, and k appear directly in the system and input matrices. This is why state-space models are so useful for CCD: changing design variables changes the matrices.

Example: a DC motor

Let θ be shaft angle, $\omega = \dot{\theta}$ shaft speed, and i armature current. A basic armature-controlled DC motor model is

$$J\dot{\omega} + b\omega = K_t i - \tau_L, \qquad L\dot{i} + Ri = v - K_e\omega,$$

where J is inertia, b is viscous friction, L is inductance, R is resistance, K_t and K_e are motor constants, v is applied voltage, and τ_L is load torque.

With $\mathbf{x} = [\theta\ \omega\ i]^T$,

$$\dot{\mathbf{x}} = \begin{bmatrix}0 & 1 & 0\\ 0 & -b/J & K_t/J\\ 0 & -K_e/L & -R/L\end{bmatrix}\mathbf{x} + \begin{bmatrix}0\\ 0\\ 1/L\end{bmatrix}v + \begin{bmatrix}0\\ -1/J\\ 0\end{bmatrix}\tau_L.$$

This model naturally combines mechanical and electrical physics.

Linearization

Many real systems are nonlinear. Control design is often performed around an operating point using a linearized model. If

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}),$$

and $(\mathbf{x}_0, \mathbf{u}_0)$ is an equilibrium or operating point, a first-order Taylor expansion gives

$$\delta\dot{\mathbf{x}} = A\,\delta\mathbf{x} + B\,\delta\mathbf{u},$$

where

$$A = \left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right|_{(\mathbf{x}_0,\mathbf{u}_0)}, \qquad B = \left.\frac{\partial \mathbf{f}}{\partial \mathbf{u}}\right|_{(\mathbf{x}_0,\mathbf{u}_0)}.$$

Linearization matters in CCD because plant design variables can change both the operating point and the local dynamics.

Activity 2.2: Nonlinear Pendulum Modeling, Linearization, and Local Feedback

Activity 2.2: Nonlinear Pendulum Modeling, Linearization, and Local Feedback

A torque-actuated pendulum is described by

$$ml^2\ddot{\theta} + b\dot{\theta} + mgl\sin\theta = u,$$

where $\theta = 0$ is the downward equilibrium and $\theta = \pi$ is the upright equilibrium. Use

$$m = 1.2\ \text{kg}, \qquad l = 0.8\ \text{m}, \qquad b = 0.12\ \text{N m s}, \qquad g = 9.81\ \text{m/s}^2,$$

with the actuator limit

$$|u(t)| \le 8\ \text{N m}.$$

1. Define

$$x_1 = \theta, \qquad x_2 = \dot{\theta},$$

and derive the nonlinear state-space model

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, u).$$

2. Determine all equilibrium pairs $(\mathbf{x}_e, u_e)$ for

$$-2\pi \le \theta_e \le 2\pi.$$

3. Linearize the system about the downward equilibrium and derive the matrices A_d and B_d.
4. Linearize the system about the upright equilibrium and derive the matrices A_u and B_u.
5. Compute the open-loop eigenvalues of both linearized systems and classify their local stability.
6. Design the upright state-feedback law

$$u = -K \begin{bmatrix} \theta - \pi \\ \dot{\theta} \end{bmatrix}$$

so that the linearized closed-loop poles are located at

$$-3 \pm 4j.$$

7. Derive the resulting closed-loop linearized matrix and verify its eigenvalues analytically.
8. Simulate both the nonlinear and linearized closed-loop systems for

$$\theta(0) = \pi + \Delta\theta, \qquad \dot{\theta}(0) = 0,$$

with

$$\Delta\theta \in \{0.02,\ 0.10,\ 0.30,\ 0.60\}.$$

9. Determine the largest tested perturbation for which the linearized model predicts the nonlinear response with less than 5% normalized root-mean-square error over the first five seconds.
10. Repeat the nonlinear simulations with actuator saturation and explain why local pole placement does not guarantee large-angle stabilization.

Activity 2.3: Exact State Transition of a Mass–Spring–Damper System

Activity 2.3: Exact State Transition of a Mass–Spring–Damper System

Consider

$$m\ddot{x} + c\dot{x} + kx = 0,$$

with

$$m = 2\ \mathrm{kg}, \qquad k = 18\ \mathrm{N/m}, \qquad x(0) = 1, \qquad \dot{x}(0) = 0.$$

Use the state vector

$$\mathbf{x} = \begin{bmatrix} x \\ \dot{x} \end{bmatrix}.$$

1. Derive the state matrix $A(c)$.
2. Derive the characteristic polynomial and the two eigenvalues as functions of c.
3. Determine the damping values corresponding to:
   (a) underdamped motion;
   (b) critical damping; and
   (c) overdamped motion.
4. For the underdamped case, derive a closed-form expression for
   $$e^{At}$$
   using the Cayley–Hamilton theorem.
5. Use the result to derive exact expressions for $x(t)$ and $\dot{x}(t)$.
6. Repeat the derivation for the critically damped case, taking care of the repeated eigenvalue.
7. For
   $$c \in \{2,\ 12,\ 20\}\ \mathrm{N\,s/m},$$
   compute
   $$\omega_n, \qquad \zeta, \qquad \lambda_1, \qquad \lambda_2.$$
8. For each damping value, determine analytically or numerically:
   (a) the first zero-crossing time;
   (b) the time of the first displacement extremum after $t = 0$; and
   (c) the 2% settling time.
9. Verify the analytical state-transition solution using a high-accuracy numerical ODE solver.
10. Explain why increasing c beyond critical damping can make the slowest mode slower even though the system contains more damping.

Open-Loop and Closed-Loop Systems

Open-loop operation

An **open-loop controller** generates a command without using measurements to correct its action during operation. It may use a model or a planned trajectory, but it does not adapt to actual measured output in real time.

Open-loop control can work well when:

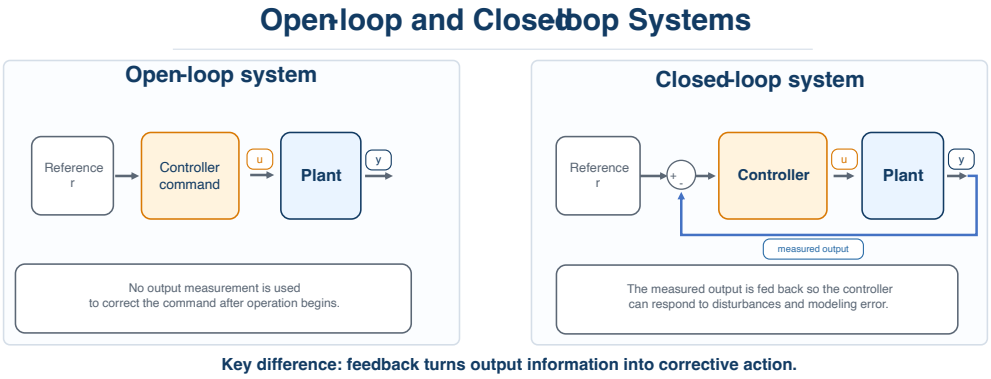
- the model is accurate;
- disturbances are negligible;
- repeatability is high; and
- performance requirements are modest.

Examples include timed traffic lights, simple washing-machine cycles, or a precomputed robot motion in a highly predictable setting.

Closed-loop operation

A **closed-loop controller** uses measurements to compare what is happening with what should happen and adjusts the input accordingly. Feedback can:

- reject disturbances;
- reduce sensitivity to modeling error;
- stabilize unstable or weakly damped plants; and
- improve tracking and regulation.



Open-loop and closed-loop structures. Closed-loop control feeds measured output back to the controller so commands can be corrected in real time.

What is actually being designed: trajectories versus gains

The open-loop/closed-loop distinction changes more than implementation detail — it changes what a "control design" even consists of. With no feedback mechanism, an open-loop control design is typically an entire time trajectory chosen in advance: for instance, the torque history commanded to a robotic-manipulator joint over a fixed motion, decided once and executed without regard to how the joint actually behaves along the way. With a feedback mechanism in place, a closed-loop control design instead consists of the parameters of that mechanism — gains, an observer, or another rule that maps the currently available information into an action — such as the gain matrix K in the state-feedback law (), or the gain matrices of an infinite-horizon linear-quadratic regulator using full-state feedback.

This distinction carries directly into CCD problem formulations. An open-loop CCD problem optimizes a plant together with a control trajectory $\mathbf{u}(\cdot)$; a closed-loop CCD problem optimizes a plant together with a feedback rule (a gain matrix, or more generally the map $\mathcal{K}$ from available information to a command). The two are not interchangeable: a plant tuned around a fixed, precomputed trajectory can look very different from a plant tuned to work well under a feedback law that has to react to whatever the system actually does.

Error-driven feedback

A standard feedback idea is to compute an error

$$e(t) = r(t) - y(t),$$

where $r(t)$ is a reference and $y(t)$ is the measured output. The controller then uses $e(t)$ or the measured state to choose the command. In its simplest form, this may be proportional control,

$$u(t) = K_p e(t).$$

More advanced architectures include proportional–integral–derivative (PID), state feedback, observer-based feedback, and model predictive control.

Why open-loop control often fails in practice

Suppose an open-loop command for a mass–spring–damper system is designed using nominal parameters. If the actual mass is slightly larger or the disturbance differs from the prediction, the motion may deviate from its intended path. A feedback controller corrects based on what actually happens, not only on what was predicted.

CCD consequence

A plant that looks best under open-loop assumptions may not be best when feedback implementation and uncertainty are considered. CCD studies therefore often need a closed-loop model, not only an optimized open-loop trajectory.

Stability and Feedback

Equilibrium points

An **equilibrium point** is a state at which the system can remain indefinitely if the input is held appropriately and no perturbation moves it away. For

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}),$$

a pair $(\mathbf{x}_e, \mathbf{u}_e)$ is an equilibrium if

$$\mathbf{f}(\mathbf{x}_e, \mathbf{u}_e) = \mathbf{0}.$$

For the uncontrolled mass–spring–damper system with no disturbance, the origin $(x, \dot{x}) = (0, 0)$ is an equilibrium.

What stability means

Engineers often explain stability informally:

- If a small perturbation leaves the system near the equilibrium, the equilibrium is **stable**.
- If the system returns to the equilibrium, it is **asymptotically stable**.
- If trajectories move away, the equilibrium is **unstable**.

For an LTI system, these questions reduce to the eigenvalues of the matrix A. For

$$\dot{\mathbf{x}} = A\mathbf{x},$$

the origin is:

- **asymptotically stable** if every eigenvalue of A has strictly negative real part;
- **unstable** if any eigenvalue has positive real part; and
- potentially **marginally stable**, or in need of a more careful test, when eigenvalues lie on the imaginary axis.

For the mass–spring–damper state matrix

$$A = \begin{bmatrix} 0 & 1 \\ -k/m & -c/m \end{bmatrix},$$

the characteristic equation is

$$\lambda^2 + \frac{c}{m}\lambda + \frac{k}{m} = 0.$$

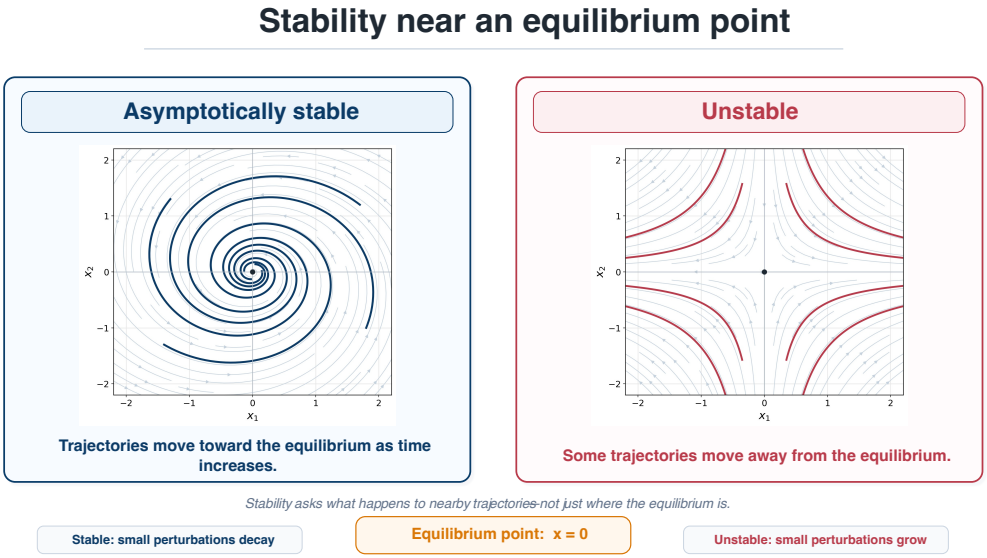
If $m > 0$, $c > 0$, and $k > 0$, the eigenvalues have negative real part and the equilibrium is asymptotically stable.

Natural frequency and damping ratio

For a second-order system,

$$\omega_n = \sqrt{\frac{k}{m}}, \qquad \zeta = \frac{c}{2\sqrt{km}}.$$

The natural frequency ω_n measures the speed of the undamped oscillatory dynamics, while the damping ratio ζ measures damping relative to critical damping. Greater stiffness generally raises natural frequency, greater damping raises damping ratio, and greater mass tends to lower natural frequency.



Stability is ultimately about trajectories. Nearby trajectories approach an asymptotically stable equilibrium, while some trajectories move away from an unstable equilibrium.

How feedback changes dynamics

Consider

$$\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u}.$$

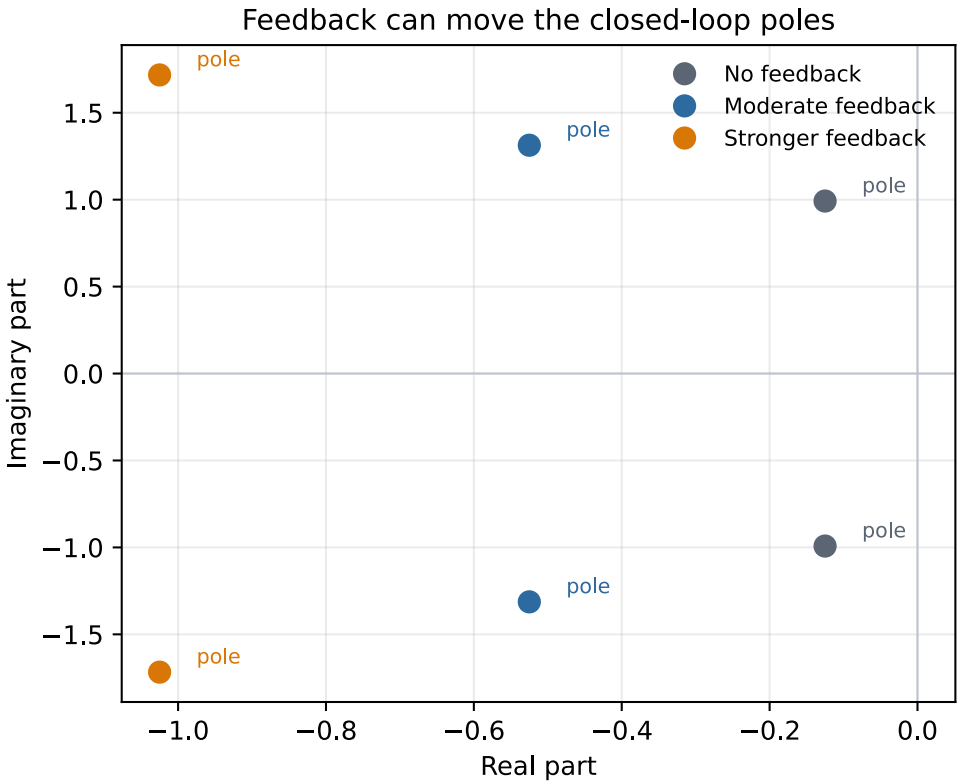
With state feedback

$$\mathbf{u} = -K\mathbf{x},$$

the closed-loop system becomes

$$\dot{\mathbf{x}} = (A - BK)\mathbf{x}.$$

Feedback reshapes the effective dynamics by changing the system matrix from A to $A - BK$.



Increasing feedback can shift the closed-loop poles. Pole movement changes response speed, oscillation, and stability margin.

If the plant design changes A or B, the same feedback matrix K no longer produces the same dynamics. Conversely, if K changes, the value of a plant design may change.

A proportional–derivative example

For the mass–spring–damper system, choose

$$u(t) = -K_p x(t) - K_d \dot{x}(t).$$

Substituting this law into () gives

$$m\ddot{x}(t) + (c + K_d)\dot{x}(t) + (k + K_p)x(t) = d(t).$$

The proportional term behaves like added stiffness and the derivative term behaves like added damping. The controller can improve performance, but only through the plant to which it is attached.

Activity 2.4: Lyapunov Analysis of a Nonlinear Feedback System

Activity 2.4: Lyapunov Analysis of a Nonlinear Feedback System

Consider the nonlinear oscillator

$$m\ddot{x} + c\dot{x} + kx + \alpha x^3 = u,$$

where

$$m > 0, \qquad \alpha > 0.$$

Apply the PD controller

$$u = -K_p x - K_d \dot{x}.$$

1. Write the nonlinear closed-loop state-space model using

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

2. Determine all equilibrium points as functions of $k + K_p$.
3. Consider the candidate Lyapunov function

$$V(x_1, x_2) = \frac{1}{2} m x_2^2 + \frac{1}{2}(k + K_p) x_1^2 + \frac{1}{4} \alpha x_1^4.$$

   Derive $\dot{V}$ along closed-loop trajectories.
4. Prove global asymptotic stability of the origin when

$$k + K_p > 0, \qquad c + K_d > 0.$$

5. Explain what changes when

$$k + K_p < 0.$$

   Determine the additional equilibria and classify them locally.
6. Replace the ideal controller with

$$u = \mathrm{sat}\left(-K_p x - K_d \dot{x}, F_{\max}\right).$$

7. Derive the region in the $(x, \dot{x})$ plane in which the actuator remains unsaturated:

$$|K_p x + K_d \dot{x}| \le F_{\max}.$$

8. Find the largest Lyapunov sublevel set

$$\Omega_\rho = \{(x, \dot{x}) : V(x, \dot{x}) \le \rho\}$$

   that is guaranteed to lie inside the unsaturated region.
9. Verify the analytical stability conclusions numerically for at least four initial conditions inside and outside Ω_ρ.
10. Explain why local linear eigenvalues cannot establish the global stability properties obtained from the Lyapunov analysis.

How Plant Design Changes System Dynamics

Design variables appear in the model

Plant design variables may include masses, inertias, stiffnesses, damping coefficients, link lengths, actuator placement, structural dimensions, sensor placement, and many other quantities. They can appear:

- in the mass matrix;
- in stiffness and damping terms;
- in aerodynamic or hydrodynamic coefficients;
- in actuator leverage or authority; and
- in sensor output equations.

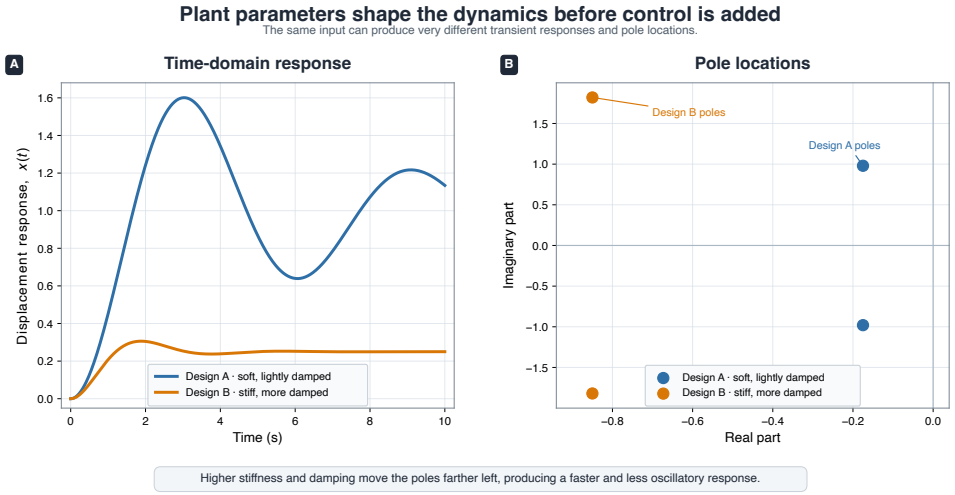
Plant design therefore changes the dynamic equations, the state-space matrices, and the input–output relationships.

Mass, stiffness, and damping

For the mass–spring–damper system:

- increasing m lowers natural frequency and usually slows the response;
- increasing k raises natural frequency and can make the system faster but more oscillatory if damping is not increased appropriately; and
- increasing c reduces oscillation, although excessive damping may slow the response and increase energy dissipation.

These are physical design tradeoffs, not merely abstract parameter changes.



Changing plant parameters changes dynamic behavior and shifts pole locations. Even before feedback is added, plant design choices alter response speed, overshoot, damping, and frequency content.

Examples from CCD application areas

CCD ideas of this kind extend well beyond the four areas developed below — they have been applied to systems as varied as chemical reactors, aircraft, underwater vehicles, wastewater treatment plants, bicycles, motorcycles, and composite smart antennas. The coupling principle is the same in every case: a controller can only reshape the dynamic pathways that the plant provides.

Vehicles. Spring and damper selection strongly affect ride comfort and road holding. A soft suspension may improve isolation but require aggressive actuation. A stiff suspension may reduce large motions but worsen comfort. The value of active control depends on the passive design.

Robots. Link lengths, structural stiffness, inertia distribution, and actuator placement affect natural frequencies, control bandwidth, and energy use. A heavier arm may require stronger actuators and respond more slowly. A compliant structure may need advanced feedback to suppress vibration.

Wind turbines. Tower flexibility, blade mass distribution, and drivetrain inertia influence rotor-speed dynamics and structural loading. Blade-pitch and torque controllers must be matched to those dynamics. If a redesign changes dominant modes, the controller tuning must also change.

A commercial direct-drive, variable-speed, pitch-controlled 1.65 MW wind turbine illustrates the point concretely. Redesigning the pitch controller to co-design rotor-speed regulation together with active tower-vibration damping produced not only smooth, robust rotor-speed control, but also a significant reduction in tower vibration and the mechanical fatigue it causes. Because the controller was absorbing loads that would otherwise have to be carried structurally, the manufacturer could design a tower considerably cheaper than a competitor's while also improving reliability and maintenance characteristics. The plant that is genuinely cheapest is not the one designed in isolation — it is the one designed knowing how much of the load-carrying job the controller will take on.

Marine energy systems. Buoy geometry, power-take-off characteristics, and mooring properties influence heave, pitch, resonance, and absorbed power. Since waves are dynamic disturbances, the plant design shapes which control strategies are possible or beneficial.

CCD viewpoint

Plant design is not only about static mass or cost. It changes the dynamic pathways through which feedback acts. This is why physical and control design variables belong in the same design study.

Activity 2.5: Allocation of Closed-Loop Dynamics between Plant and Feedback

Activity 2.5: Allocation of Closed-Loop Dynamics between Plant and Feedback

Consider

$$m\ddot{x} + c\dot{x} + kx = u,$$

with the PD feedback law

$$u = -K_p x - K_d \dot{x}.$$

Use $m = 2$ kg. The desired closed-loop dynamics are

$$s^2 + 2\zeta_d \omega_d s + \omega_d^2 = 0,$$

where

$$\zeta_d = 0.7, \qquad \omega_d = 4\ \mathrm{rad/s}.$$

1. Show that all plant–controller combinations producing the desired poles must satisfy

$$k + K_p = m\omega_d^2,$$

and

$$c + K_d = 2m\zeta_d\omega_d.$$

2. Consider the allocation cost

$$C = \alpha_k k^2 + \alpha_c c^2 + \alpha_p K_p^2 + \alpha_d K_d^2,$$

with

$$\alpha_k = 0.04, \qquad \alpha_c = 0.08, \qquad \alpha_p = 0.01, \qquad \alpha_d = 0.02.$$

Derive the minimum-cost allocation analytically.

3. Repeat the derivation subject to the passive-safety requirements

$$k \geq 8\ \mathrm{N/m}, \qquad c \geq 1.5\ \mathrm{N\,s/m}.$$

4. For the initial condition

$$x(0) = 1, \qquad \dot{x}(0) = 0,$$

derive the closed-loop state trajectory.

5. Derive the control trajectory

$$u(t) = -K_p x(t) - K_d \dot{x}(t)$$

for an arbitrary feasible allocation.

6. Show that different allocations can produce identical closed-loop poles and identical state trajectories but different control-force histories.

7. Compute

$$E_u = \int_0^\infty u(t)^2\, dt$$

for:

(a) the unconstrained minimum-cost allocation;
(b) the passive-safety-constrained allocation; and
(c) the fully passive allocation $K_p = K_d = 0$.

8. Suppose the actuator fails at $t = 1$ and

$$u(t) = 0, \qquad t \geq 1.$$

Simulate and compare the three designs.

9. Explain why matching closed-loop poles alone is insufficient for choosing a plant–controller design.

Activity 2.6: Modal Dynamics and Actuator Placement in a Two-Mass System

Activity 2.6: Modal Dynamics and Actuator Placement in a Two-Mass System

Two unit masses are connected by springs as described by

$$M\ddot{\mathbf{q}} + C\dot{\mathbf{q}} + K\mathbf{q} = B_f u,$$

where

$$M = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \qquad K = k \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix},$$

and proportional damping is used:

$$C = \beta K.$$

Use

$$k = 20\ \mathrm{N/m}, \qquad \beta = 0.015\ \mathrm{s}.$$

Consider three actuator placements:

$$B_f^{(1)} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \qquad B_f^{(2)} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \qquad B_f^{(3)} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}.$$

1. Derive the four-state model using

$$\mathbf{x} = \begin{bmatrix} \mathbf{q} \\ \dot{\mathbf{q}} \end{bmatrix}.$$

2. Solve the generalized eigenvalue problem

$$K\boldsymbol{\phi}_j = \omega_j^2 M\boldsymbol{\phi}_j$$

and compute the natural frequencies and normalized mode shapes.

3. Transform the mechanical equations to modal coordinates

$$\mathbf{q} = \Phi \boldsymbol{\eta}.$$

4. For each actuator placement, compute the modal input vector

$$\widetilde{B}_f = \Phi^T B_f.$$

5. Identify which modes are weakly actuated or completely unactuated by each placement.
6. Apply the initial condition

$$\mathbf{q}(0) = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \qquad \dot{\mathbf{q}}(0) = \mathbf{0},$$

   and determine which mode is initially excited.
7. For each actuator placement, design full-state feedback

$$u = -K_x \mathbf{x}$$

   to move the controllable closed-loop poles as far left as possible while satisfying

$$\|K_x\|_2 \le 25.$$

8. Compare the closed-loop responses and control efforts for the three actuator placements.
9. Repeat the modal analysis after changing the second mass to

$$m_2 = 1.4\ \mathrm{kg}.$$

10. Explain how a physical design change can alter mode shapes, actuator authority, and the controller that performs best.

Examples

Example 2.1: derive and analyze a state-space model

Consider a mass–spring–damper system with $m = 2$ kg, $c = 3$ N s/m, and $k = 8$ N/m. The control input is force u and the measured output is displacement $y = x$.

Step 1: Write the dynamic equation.

$$2\ddot{x} + 3\dot{x} + 8x = u.$$

Step 2: Define the states.

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

Step 3: Write the first-order model.

$$\dot{x}_1 = x_2, \qquad \dot{x}_2 = -4x_1 - 1.5x_2 + 0.5u,$$

so

$$\dot{\mathbf{x}} = \begin{bmatrix} 0 & 1 \\ -4 & -1.5 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0 \\ 0.5 \end{bmatrix} u, \qquad y = \begin{bmatrix} 1 & 0 \end{bmatrix} \mathbf{x}.$$

Step 4: Analyze stability. The eigenvalues are the roots of

$$\lambda^2 + 1.5\lambda + 4 = 0,$$

which are

$$\lambda = -0.75 \pm j1.854.$$

Their real parts are negative, so the open-loop equilibrium is asymptotically stable.

Step 5: Interpret the result. The system oscillates, but the oscillations decay. If the designer raises stiffness to $k = 18$ without changing damping, the oscillation frequency increases and the best feedback tuning may change.

Example 2.2: effect of PD feedback

Suppose Example 2.1 uses

$$u = -6x - 4\dot{x}.$$

The closed-loop equation is

$$2\ddot{x} + (3 + 4)\dot{x} + (8 + 6)x = 0,$$

with characteristic equation

$$2\lambda^2 + 7\lambda + 14 = 0.$$

The closed-loop poles have a more negative real part than the open-loop poles, indicating increased damping and faster decay. Feedback improves the response by altering the effective dynamics.

Example 2.3: identify signals in cruise control

Consider a simple vehicle moving along a straight road:

$$m\dot{v} = F_t - c_r v - mg\sin\theta,$$

where v is vehicle speed, F_t is traction force, c_r is a drag or rolling-resistance coefficient, and θ is road slope. A sensible description is:

- **state:** vehicle speed v;
- **control input:** traction force F_t or throttle command;
- **output:** vehicle speed v; and
- **disturbance:** road slope θ.

This first-order model reminds us that a useful state set need not contain many variables.

Common modeling mistakes

1. **Treating outputs as states without justification.** Outputs may equal states, but they do not have to.
2. **Ignoring disturbances.** This usually creates unrealistic conclusions about controller performance.
3. **Using too many states.** More states are not always better; the state set should be minimal and meaningful.
4. **Forgetting units.** Dynamic equations must be dimensionally consistent.
5. **Assuming plant design changes only constraints or cost.** In CCD, plant design often changes the differential equations themselves.
6. **Confusing stability with performance.** A stable system is not necessarily fast, efficient, comfortable, or optimal.

Chapter Summary and Problems

Summary

This chapter introduced the dynamic-system ideas needed for CCD. Physical modeling leads to differential equations, and state-space representation organizes those equations using states, inputs, outputs, and disturbances. Open-loop control relies on prediction, while closed-loop control uses feedback to respond to actual behavior. Stability describes what trajectories do near an equilibrium, and feedback changes stability and response by changing the closed-loop dynamics. Most importantly for CCD, plant design variables change the dynamic model and therefore change which controller is best.

The central message

Controllers do not act on abstract equations; they act on physical systems with specific dynamics. When the plant changes, the control problem changes too.

Key terms

Dynamic system; state; control input; output; disturbance; differential equation; multidisciplinary analysis (MDA); differential-algebraic equation (DAE); index-1 DAE; algebraic constraint; state-space representation; linearization; open-loop system; closed-loop system; feedback; equilibrium; stability; asymptotic stability; eigenvalue; natural frequency; damping ratio; closed-loop poles.

Problems

1. **Nonlinear modeling and local dynamics.** A torque-actuated pendulum satisfies $ml^2\ddot{\theta}+b\dot{\theta}+mgl\sin\theta = u$. Derive its nonlinear state-space model and the exact linearization about the upright equilibrium, then determine the state-feedback gains that place the linearized poles at a prescribed stable conjugate pair.
2. **Plant design and controllability.** A flexible two-mass system obeys $M(p)\ddot{q} + C\dot{q} + K(p)q = b(r_a)u$, where p changes stiffness and r_a is actuator location. Derive a modal controllability metric and use it to state a quantitative co-design criterion that prevents any retained mode from becoming weakly actuated.
3. **LQR-dependent plant selection.** For $\dot{x} = \begin{bmatrix} 0 & 1 \\ -k/m & -c/m \end{bmatrix} x + \begin{bmatrix} 0 \\ 1/m \end{bmatrix} u$ and $J = \int_0^\infty (x^T Q x + ru^2)dt + \gamma m$, derive the algebraic-Riccati sensitivity equations needed to compute the total derivative of the optimized closed-loop cost with respect to m.
4. **Robust stability over a design family.** The closed-loop matrix is $A_{cl}(p) = A_0 + pA_1 - BK$ for $p \in [p_-, p_+]$. Formulate and justify a common quadratic Lyapunov inequality that certifies a uniform exponential decay rate over the entire interval, reducing the infinite family to finitely many matrix inequalities when the dependence is affine.
5. **Observer–plant interaction.** A sensor placement variable s changes the output matrix $C(s)$ in $\dot{x} = A(p)x + Bu$, $y = C(s)x + v$. Derive the observability Gramian and formulate a joint placement–estimator design constraint that bounds the worst-direction state-estimation uncertainty over a finite horizon.
6. **Sampled-data feedback.** For $\dot{x} = Ax + Bu$ with zero-order-hold control $u(t) = -Kx(kh)$ on $kh \le t < (k+1)h$, derive the exact discrete closed-loop matrix and determine the largest sampling period h that preserves asymptotic stability for a specified numerical pair (A, B, K).
7. **Disturbance attenuation.** The quarter-car model is $M\ddot{q} + C\dot{q} + Kq = B_u u + B_w w$, with performance output $z = C_z x + D_z u$. Derive the bounded-real linear matrix inequality that certifies $\|T_{w\to z}\|_\infty < \gamma$ and identify which matrix terms change when suspension stiffness is a design variable.
8. **Region of attraction under actuator saturation.** For $\dot{x} = Ax + B\,\mathrm{sat}(-Kx, u_{\max})$, derive an invariant ellip-

soidal inner approximation $\mathcal{E}(P) = \{x : x^T P x \le 1\}$ by combining a Lyapunov inequality with constraints ensuring the feedback remains unsaturated inside $\mathcal{E}(P)$.

9. **Unmodeled flexible dynamics.** A nominal rigid-body plant $G_0(s) = 1/(Js^2)$ is augmented by a flexible mode $G_f(s) = \omega_f^2/(s^2 + 2\zeta_f\omega_f s + \omega_f^2)$. Derive a robust-stability restriction on controller bandwidth using multiplicative uncertainty and show how increasing structural stiffness can enlarge the admissible bandwidth.

10. **Dynamic feasibility of an index-one DAE.** An electromechanical actuator satisfies $L\dot{i} + Ri + k_e\omega = v$, $J\dot{\omega} = k_t i - \tau_L$, and the algebraic saturation law $0 = i - \mathrm{sat}(i_c, i_{\max}(T))$. Formulate a consistent state–algebraic representation and derive the conditions under which an equilibrium and its local linearization are well defined.

References and further reading

1. Allison, J. T., & Herber, D. R. (2014). Multidisciplinary design optimization of dynamic engineering systems. *AIAA Journal*, *52*(4), 691–710. DOI: 10.2514/1.J052182

2. Allison, J. T., Guo, T., & Han, Z. (2014). Co-design of an active suspension using simultaneous dynamic optimization. *Journal of Mechanical Design*, *136*(8), Article 081003.

3. Garcia-Sanz, M. (2019). Control co-design: An engineering game changer. *Advanced Control for Applications: Engineering and Industrial Systems*, *1*(1), Article e18. DOI: 10.1002/adc2.18

4. Herber, D. R., & Allison, J. T. (2019). Nested and simultaneous solution strategies for general combined plant and control design problems. *Journal of Mechanical Design*, *141*(1), Article 011402. DOI: 10.1115/1.4040705

5. Martins, J. R. R. A., & Ning, A. (2021). *Engineering design optimization*. Cambridge University Press.

6. Franklin, G. F., Powell, J. D., & Emami-Naeini, A. (2019). *Feedback control of dynamic systems* (8th ed.). Pearson.

7. Herber, D. R. (2017). *Advances in Combined Architecture, Plant, and Control Design* (Doctoral dissertation). University of Illinois at Urbana-Champaign.

Engineering Design Optimization

Chapter 3: Engineering Design Optimization

Formulation, search, numerical reliability, and multidisciplinary coupling

> Optimization does not replace engineering judgment. It organizes engineering judgment into a precise search over decisions, performance measures, models, and constraints.

Control co-design requires more than knowing how dynamic systems behave. It also requires a disciplined method for choosing among many possible plant and controller designs. Engineering design optimization provides that method. An optimization problem defines what decisions may change, how designs are compared, which designs are physically or operationally acceptable, and how a numerical algorithm searches for an improved solution.

This chapter introduces the optimization concepts needed throughout the rest of the course: design variables, objectives, constraints, feasible design spaces, local and global optima, gradient-based search, constrained optimality, scaling, numerical conditioning, multidisciplinary design optimization (MDO), and optimization of dynamic systems.

Learning objectives

After completing this chapter, you should be able to:

1. identify design variables, objectives, equality constraints, inequality constraints, and bounds;
2. write an engineering design problem in standard mathematical form;
3. interpret feasible, infeasible, active, and inactive constraints geometrically;
4. distinguish local and global optima and explain why initialization matters;
5. describe gradient descent, line search, Newton, and quasi-Newton methods;
6. explain why scaling and numerical conditioning influence reliability;
7. identify disciplines and coupling variables in an MDO problem; and
8. formulate problems whose objectives and constraints depend on dynamic simulations.

Chapter roadmap

Formulation → Feasible space → Optimality → Gradient methods → Scaling → MDO → Dynamic optimization

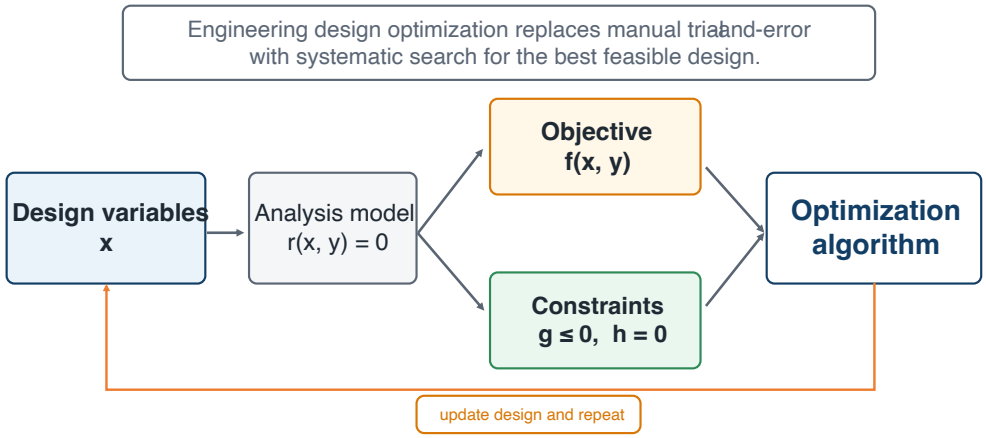
Formulating an Engineering Design Optimization Problem

What is engineering design optimization?

Engineering design optimization is the systematic selection of design decisions to improve a quantitative measure of performance while satisfying requirements. It automates the search within the familiar engineering loop

$$\text{propose design} \rightarrow \text{analyze design} \rightarrow \text{judge result} \rightarrow \text{modify design}.$$

The engineer still decides what can change, what performance means, which models are credible, and which constraints are essential. The algorithm proposes design updates using numerical information.



The major elements of an engineering design optimization problem.

The four distinct elements are:

1. **Design variables** describe the choices available to the designer.
2. An **analysis model** predicts how a candidate design behaves.
3. An **objective** assigns a scalar measure of desirability.
4. **Constraints** determine whether the design is acceptable.

The algorithm is not the problem itself. It is the numerical method used to search the problem defined by the engineer.

Optimization versus analysis

An engineering analysis asks, "What happens for this design?" An optimization study asks, "Which design should be selected from the allowed set?"

A simulation may compute stress, temperature, power, displacement, or cost for one design. Optimization wraps that model in an iterative search. A model suitable for one accurate analysis may be too slow, noisy, discontinuous, or fragile for thousands of evaluations.

Formulation comes first

A numerical optimizer can solve the problem it is given, but it cannot decide whether that problem represents the real mission. Poor objectives or missing constraints can produce a mathematically optimal but unacceptable design:

- minimizing mass without a stiffness constraint may produce excessive deflection;

- maximizing absorbed energy without a force limit may demand an unrealizable actuator;
- minimizing cost without a reliability requirement may produce a fragile system; and
- optimizing nominal performance without disturbance scenarios may produce poor closed-loop behavior.

Key idea

The most important optimization decision is often not which algorithm to use. It is how to formulate the problem so that the mathematical optimum corresponds to an engineering design worth building.

Standard mathematical form

A continuous optimization problem is commonly written as

$$\begin{aligned} \underset{\mathbf{x}\in\mathbb{R}^n}{\text{minimize}} \quad & f(\mathbf{x}) \\ \text{subject to} \quad & g_i(\mathbf{x}) \le 0, \qquad i = 1, \dots, m, \\ & h_j(\mathbf{x}) = 0, \qquad j = 1, \dots, p, \\ & \mathbf{x}_L \le \mathbf{x} \le \mathbf{x}_U. \end{aligned}$$

Here, $\mathbf{x}$ is the vector of design variables, f is the objective, g_i are inequality constraints, h_j are equality constraints, and $\mathbf{x}_L, \mathbf{x}_U$ are bounds. Maximizing P is equivalent to minimizing $-P$.

Design variables

Design variables should correspond to real decisions. They may represent dimensions, materials, masses, stiffnesses, damping, controller gains, actuator sizes, sensor locations, sampling times, topology, or architecture.

- **Continuous:** any real value within a range.
- **Integer:** whole-number values, such as the number of blades.
- **Binary:** whether a component exists.
- **Categorical:** a selection such as material or gearbox type.
- **Functional:** a function of time or space, such as $u(t)$ or $t(s)$.

This chapter emphasizes continuous variables because gradient-based methods are central to large CCD problems.

Objectives

A useful objective reflects the mission, changes meaningfully with decisions, and can be computed reliably. Common examples include mass, cost, energy, tracking error, fatigue damage, risk, or negative profit.

Dynamic-system objectives often have terminal and integral terms:

$$J = \Phi(\mathbf{z}(t_f)) + \int_{t_0}^{t_f} L(\mathbf{z}(t), \mathbf{u}(t), \mathbf{x}, t)\, dt.$$

Multiple measures can be combined as $J = \sum_{i=1}^{q} w_i J_i$, but weights encode value judgments and are sensitive to units and scaling.

Constraints and bounds

Constraints express nonnegotiable requirements such as stress, temperature, actuator, displacement, stability, geometry, manufacturability, conservation, performance, or reliability limits.

For $\sigma(\mathbf{x}) \leq \sigma_{\max}$, a normalized inequality is

$$g_\sigma(\mathbf{x}) = \frac{\sigma(\mathbf{x})}{\sigma_{\max}} - 1 \leq 0.$$

Equality constraints $h_j(\mathbf{x}) = 0$ may represent conservation, compatibility, or coupled-analysis residuals. Bounds $x_{L,k} \leq x_k \leq x_{U,k}$ should be physically defensible: overly wide bounds expose invalid model regions, while narrow bounds can hide better designs.

Example 3.1: cylindrical pressure vessel

For a vessel with radius r, thickness t, and fixed length L, minimize material volume while meeting volume and thin-wall stress requirements:

$$\begin{aligned} \underset{r,t}{\text{minimize}} \quad & 2\pi r L t \\ \text{subject to} \quad & 1 - \frac{\pi r^2 L}{V_{\min}} \leq 0, \\ & \frac{pr}{\sigma_{\max} t} - 1 \leq 0, \\ & r_L \leq r \leq r_U, \\ & t_L \leq t \leq t_U. \end{aligned}$$

Here, r and t are decisions; stress and volume are predicted quantities. The stress equation belongs in the model, while the allowable stress belongs in the constraint set.

Checkpoint

For a system you know, list three true design decisions, two predicted outputs, one objective, and three nonnegotiable constraints.

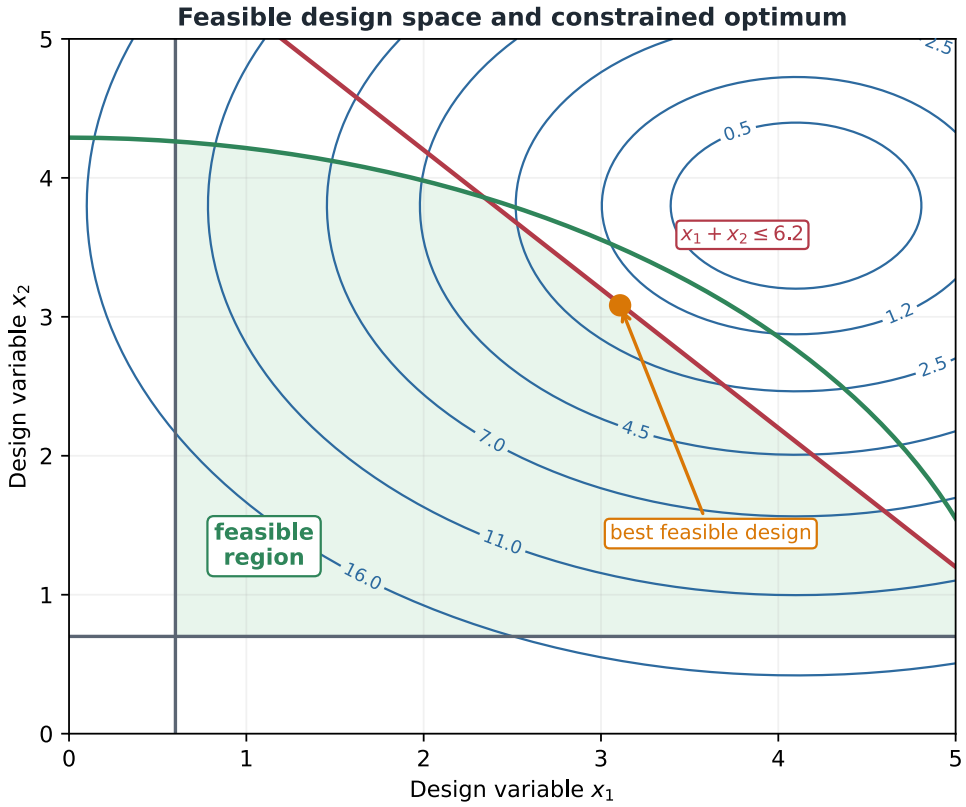
Feasible Design Spaces and Optima

Feasible and infeasible designs

A design is **feasible** if it satisfies every constraint. The feasible set is

$$\mathcal{F} = \{\mathbf{x} : \mathbf{g}(\mathbf{x}) \leq \mathbf{0},\ \mathbf{h}(\mathbf{x}) = \mathbf{0},\ \mathbf{x}_L \leq \mathbf{x} \leq \mathbf{x}_U\}.$$

A design outside $\mathcal{F}$ is infeasible. The unconstrained best design is often infeasible; constraints determine how far the optimizer may move toward improved performance.



Objective contours connect equally good designs, constraints carve out the feasible region, and the optimum often lies on one or more boundaries.

Active constraints and margin

At a candidate $\mathbf{x}^*$, an inequality is:

- **active** if $g_i(\mathbf{x}^*) = 0$;
- **inactive** if $g_i(\mathbf{x}^*) < 0$; and
- **violated** if $g_i(\mathbf{x}^*) > 0$.

Active constraints often reveal the design drivers. A normalized margin may be defined as

$$M_i = -g_i(\mathbf{x}).$$

Then $M_i > 0$ indicates margin, $M_i = 0$ activity, and $M_i < 0$ violation. Reporting margins is much more informative than reporting only the objective.

Empty and disconnected feasible sets

A formulation may have no feasible design because requirements conflict, or it may contain disconnected feasible regions. Failure to find a feasible design can mean:

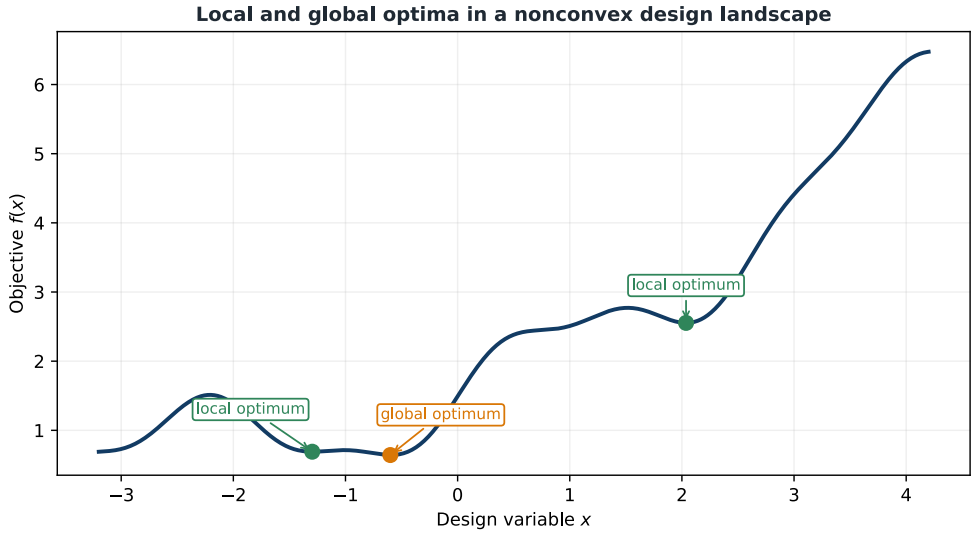
- the requirements are genuinely impossible;
- bounds are too restrictive;
- equations or units are wrong;
- the solver cannot reach a feasible region; or
- poor scaling obstructs constraint satisfaction.

The correct response is diagnosis, not automatic constraint relaxation.

Local and global optima

A feasible $\mathbf{x}^*$ is a **local minimum** if it has no better feasible neighbor in some small region. It is a **global minimum** if

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



A local algorithm can converge to different solutions from different starting points.

Nonconvexity arises from nonlinear physics, resonance, bifurcation, contact, switching, saturation, discrete architectures, geometric constraints, multidisciplinary coupling, and time-dependent control. A successful solver run is not proof of global optimality.

In a convex minimization problem, every local optimum is global. Most CCD problems are not convex, but convex models remain useful inside algorithms such as sequential quadratic programming.

Practical strategies for nonconvex problems

- Run a local solver from multiple initial guesses.
- Select diverse starts using engineering intuition.
- Solve a relaxed or low-fidelity problem first.
- Use continuation or homotopy to introduce difficulty gradually.

- Combine global exploration with local refinement.
- Inspect low-dimensional objective and constraint landscapes.

Global methods may improve exploration but typically require far more model evaluations and still may not guarantee the global optimum for a difficult black-box problem.

Activity 3.1: Local and Global Minima in a Tilted Double-Well Problem

Activity 3.1: Local and Global Minima in a Tilted Double-Well Problem

Consider the nonconvex function

$$f(x, y) = (x^2 - 1)^2 + (y - x)^2 + 0.15x.$$

1. Derive the gradient and Hessian.
2. Show that every stationary point satisfies

$$y = x$$

and

$$4x^3 - 4x + 0.15 = 0.$$

3. Compute all real roots of the cubic equation to at least eight significant digits.
4. Classify every stationary point using the Hessian eigenvalues.
5. Identify the local minima, the saddle point, and the global minimum.
6. Apply gradient descent with an Armijo backtracking line search from the initial points

$$(-2, 2), \quad (-1, 0), \quad (0, 0), \quad (1, 0), \quad (2, -2).$$

7. Repeat using Newton's method with line search.
8. Construct a basin-of-attraction plot on

$$-2 \le x \le 2, \qquad -2 \le y \le 2.$$

9. Explain why convergence from several initial guesses provides evidence about nonconvexity but does not prove global optimality.
10. Add the constraint

$$x + y \ge 0$$

and determine the constrained global minimum using KKT conditions.

Gradient-Based Optimization

The gradient

For a scalar objective $f(\mathbf{x})$,

$$\nabla f(\mathbf{x}) = \begin{bmatrix} \partial f/\partial x_1 & \partial f/\partial x_2 & \cdots & \partial f/\partial x_n \end{bmatrix}^T.$$

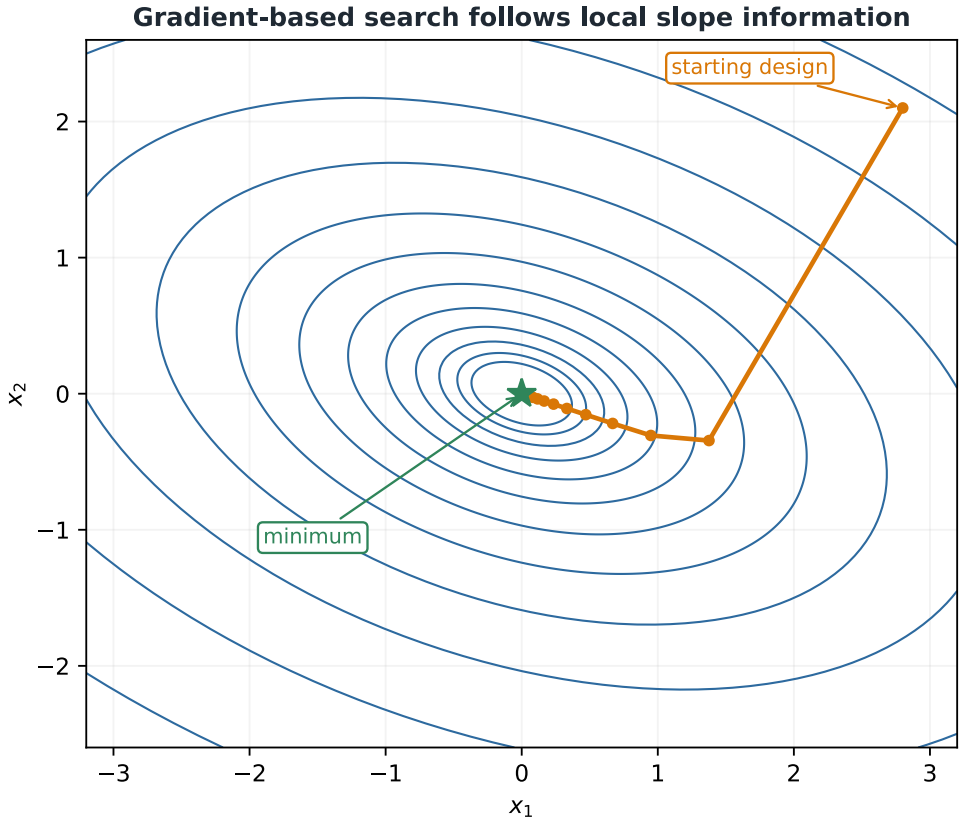
The gradient points in the direction of steepest local increase in Euclidean coordinates, so $-\nabla f$ points toward steepest local decrease.

Gradient descent

The basic update is

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \nabla f(\mathbf{x}_k),$$

where $\alpha_k > 0$ is the step length.



Each update uses local slope information. Small steps are slow, while overly large steps may overshoot or diverge.

Line search and trust regions

A line-search method chooses a direction $\mathbf{p}_k$ and seeks an acceptable solution to

$$\underset{\alpha>0}{\text{minimize}} \quad f(\mathbf{x}_k + \alpha \mathbf{p}_k).$$

The direction may come from steepest descent, Newton, or quasi-Newton information. A practical line search seeks sufficient reduction rather than an exact one-dimensional optimum. Trust-region methods instead restrict the step to a region where a local model is considered reliable.

Newton and quasi-Newton methods

A second-order Taylor approximation is

$$f(\mathbf{x}_k + \mathbf{p}) \approx f(\mathbf{x}_k) + \nabla f(\mathbf{x}_k)^T \mathbf{p} + \frac{1}{2}\mathbf{p}^T H(\mathbf{x}_k)\mathbf{p}.$$

Minimizing this quadratic model gives the Newton step

$$H(\mathbf{x}_k)\mathbf{p}_k = -\nabla f(\mathbf{x}_k).$$

Newton's method can converge rapidly near a well-behaved optimum, but exact Hessians may be expensive and the step may fail if the Hessian is indefinite or ill-conditioned. Quasi-Newton methods such as BFGS estimate Hessian information from changes in gradients.

Computing derivatives

Gradient-based algorithms are only as reliable as the derivatives they receive. Four practical families of methods exist, and they differ in cost, accuracy, and required access to the model.

Finite differences. The forward-difference formula follows from a truncated Taylor series,

$$\frac{\partial f}{\partial x_j} = \frac{f(\mathbf{x} + h\hat{e}_j) - f(\mathbf{x})}{h} + O(h),$$

which is first-order accurate; a central difference reaches $O(h^2)$ at roughly twice the cost. Finite differences face a *step-size dilemma*: shrinking h reduces truncation error, but below a certain size subtractive cancellation in finite-precision arithmetic dominates and the estimate degrades—for double precision, the best forward-difference step is typically near 10^{-8}. Because each design direction requires a separate perturbed model evaluation, the cost of a full Jacobian scales linearly with the number of design variables. Finite differences require no access to source code, so they remain the default when a model is a black box, but they are neither the most accurate nor the most efficient option when the model can be modified.

Complex-step differentiation. Perturbing a design variable along the imaginary axis instead of the real axis,

$$\frac{\partial f}{\partial x_j} = \frac{\operatorname{Im}\left(f(\mathbf{x} + ih\hat{e}_j)\right)}{h} + O(h^2),$$

avoids subtraction entirely, so there is no cancellation error. The step size can then be shrunk toward machine precision (values near 10^{-200} are typical) without any loss of accuracy, so the resulting derivative matches the precision of the function itself. The cost still scales linearly with the number of design variables, comparable to a central difference, but the method requires source-code access and correct handling of complex arithmetic (including redefined absolute-value and comparison logic).

Algorithmic (automatic) differentiation. AD applies the chain rule mechanically to every elementary operation in the source code, without ever forming a symbolic expression. In *forward mode*, one input is seeded and its derivative is propagated through every subsequent line of code, so the cost scales with the number of inputs and is independent of the number of outputs. In *reverse mode*, one output is seeded and its derivative is propagated backward to every variable it depends on, so the cost scales with the number of outputs and is independent of the number of inputs—but it requires storing intermediate values from a forward pass before the backward

sweep can run. Reverse-mode AD is therefore especially attractive when a problem has many design variables and few objectives or constraints, which is common in large CCD problems.

Direct and adjoint sensitivity methods. When a model is expressed as governing residual equations, implicit differentiation of the residuals with respect to the states yields a linear system whose solution gives the total derivatives. Solving that system once per design variable is the *direct* method; solving it once per output is the *adjoint* method—mirroring the forward/reverse trade-off of AD, but applied at the level of the governing equations rather than lines of code.

Key idea

No single derivative method dominates in every case. The right choice depends on the number of design variables and outputs, whether source code can be modified, and how much accuracy the optimizer needs to converge reliably.

Example 3.2: quadratic minimization

Consider

$$f(x_1, x_2) = 2x_1^2 + x_1x_2 + x_2^2 - 6x_1 - 5x_2.$$

Its gradient is

$$\nabla f = \begin{bmatrix} 4x_1 + x_2 - 6 \\ x_1 + 2x_2 - 5 \end{bmatrix}.$$

Setting $\nabla f = 0$ gives $x_1^* = 1$ and $x_2^* = 2$. The Hessian

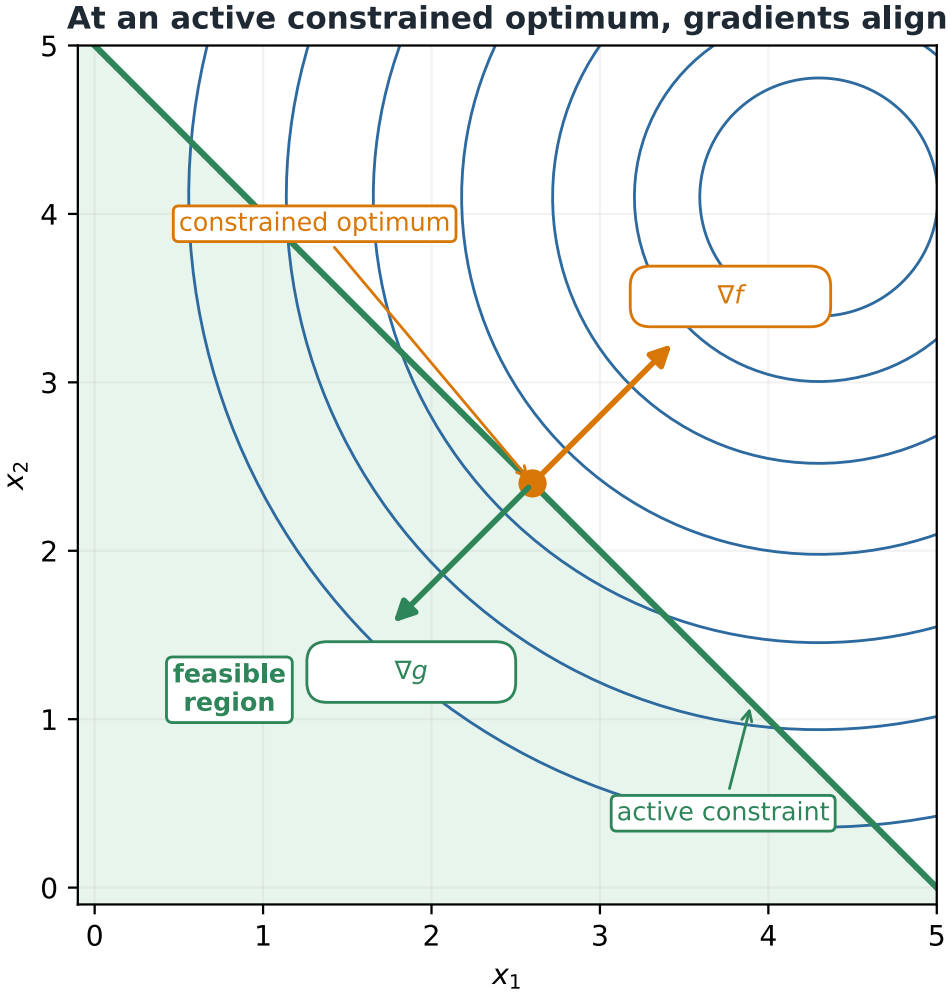
$$H = \begin{bmatrix} 4 & 1 \\ 1 & 2 \end{bmatrix}$$

is positive definite, so this stationary point is the unique global minimum of the convex quadratic.

Constrained Optimization, Scaling, and Conditioning

Why $\nabla f = 0$ is insufficient

For an unconstrained differentiable optimum, $\nabla f(\mathbf{x}^*) = 0$ is necessary. At a constrained optimum, the gradient need not vanish because active constraints block motion in improving directions.



At an active boundary, the objective gradient is balanced by the active constraint gradient.

Lagrangian and KKT conditions

Define the Lagrangian

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\nu}) = f(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i g_i(\mathbf{x}) + \sum_{j=1}^{p} \nu_j h_j(\mathbf{x}).$$

Under suitable regularity conditions, a local optimum satisfies the Karush–Kuhn–Tucker conditions:

$$
\begin{aligned}
\nabla_{\mathbf{x}} \mathcal{L} &= 0, \\
g_i(\mathbf{x}) &\le 0, \\
h_j(\mathbf{x}) &= 0, \\
\lambda_i &\ge 0, \\
\lambda_i g_i(\mathbf{x}) &= 0.
\end{aligned}
$$

The last condition is **complementary slackness**: an inequality is either inactive with zero multiplier, or active and potentially influential.

A multiplier measures the local value of relaxing its constraint. A large multiplier indicates that a small relaxation could significantly improve the objective and can help prioritize requirements.

Major constrained algorithms

- **Penalty and augmented-Lagrangian methods** add measures of constraint violation to the objective.
- **Sequential quadratic programming (SQP)** solves a sequence of quadratic constrained approximations.
- **Interior-point methods** approach the solution from within the feasible region using barrier terms.

Modern nonlinear-programming solvers often use SQP or interior-point methods. CCD users should understand the required inputs and returned diagnostics even if they do not implement these solvers.

Gradient-based and gradient-free solvers in CCD practice

SQP and interior-point methods are gradient-based: they exploit derivative information and typically converge in far fewer function evaluations than derivative-free alternatives, but they converge to the local optimum nearest the initial guess. Gradient-free algorithms—such as covariance matrix adaptation evolution strategy (CMA-ES), genetic algorithms (GA), and particle swarm optimization (PSO)—require only function evaluations, tolerate noisy or discontinuous objectives, and search more globally, but the number of evaluations they need tends to grow rapidly with the number of design variables, so they scale poorly to high-dimensional problems even after they have located a promising region.

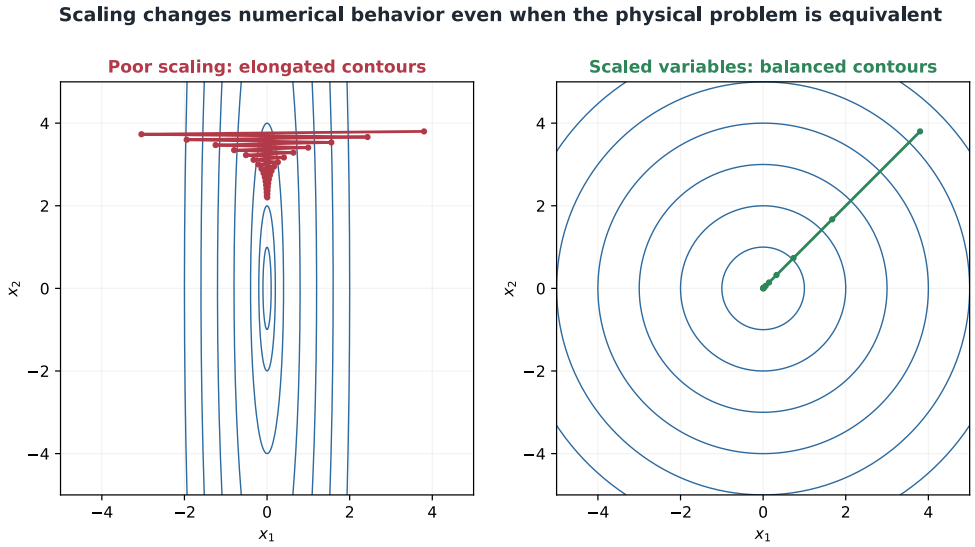
A recent review of wind-turbine control co-design practice illustrates how these families are actually combined rather than chosen exclusively. Because CCD problems often combine a low-dimensional, potentially multimodal plant-design search with a much higher-dimensional control-trajectory search, a common hybrid strategy assigns a gradient-free optimizer (CMA-ES, GA, or PSO) to the plant-design variables and a gradient-based optimizer (SQP or an interior-point method) to the control-design variables, matching each discipline to the solver family best suited to its dimensionality and smoothness. Widely used implementations include CMA-ES and COBYLA on the gradient-free side, and SNOPT (a sparse SQP implementation) and IPOPT (an interior-point implementation) on the gradient-based side.

Key idea

Solver choice is itself a CCD design decision. It should follow from the dimensionality and smoothness of the plant and control subproblems, not from habit or default settings.

Why scaling matters

Mathematically equivalent formulations can behave very differently numerically. A solver may handle variables near 10^{-6} and 10^{6} unevenly, especially when using common step and convergence tolerances.



Scaling produces a numerically more balanced landscape.

Variable, objective, and constraint scaling

A physical variable may be represented as

$$x_i = x_{i,\mathrm{ref}} + s_i \tilde{x}_i,$$

with typical $\tilde{x}_i$ values near one. Bound-based scaling maps a variable to $[0, 1]$:

$$\tilde{x}_i = \frac{x_i - x_{L,i}}{x_{U,i} - x_{L,i}}.$$

Functions can similarly be scaled using meaningful nonzero references:

$$\tilde{f} = \frac{f}{f_{\mathrm{ref}}}, \qquad \tilde{g}_i = \frac{g_i}{g_{i,\mathrm{ref}}}.$$

Conditioning and tolerances

Conditioning describes solution sensitivity to small changes in data. Poor conditioning can cause slow convergence, sensitivity to tolerances, inaccurate finite differences, unstable linear algebra, and large design changes from small numerical errors.

Scaling improves algorithm behavior but cannot remove genuine physical ill-conditioning. If two variables have nearly identical effects, the design may be fundamentally poorly identifiable.

Solver tolerances operate in the provided coordinates. A residual tolerance of 10^{-6} has a different meaning for a normalized constraint than for a dimensional force balance. Scale first, then select tolerances.

Example 3.3: poorly scaled quadratic

For

$$f(x_1, x_2) = 10^6 x_1^2 + x_2^2,$$

the curvature in x_1 is a million times larger than in x_2. Gradient descent needs a small step for stability in x_1,

making progress in x_2 slow. Define

$$\tilde{x}_1 = 1000x_1, \qquad \tilde{x}_2 = x_2.$$

Then $f = \tilde{x}_1^2 + \tilde{x}_2^2$, which has balanced curvature.

Key idea

Consistent scaling does not change the physical problem. It changes how the numerical algorithm sees the problem, often turning an unreliable search into a robust one.

Activity 3.2: KKT Analysis of a Minimum-Mass Cantilever Beam

Activity 3.2: KKT Analysis of a Minimum-Mass Cantilever Beam

A rectangular cantilever beam has width b, height h, length L, elastic modulus E, material density ρ, and an end load P. Its mass is

$$m(b,h) = \rho L b h.$$

The maximum bending stress and tip displacement are

$$\sigma(b,h) = \frac{6PL}{bh^2}, \qquad \delta(b,h) = \frac{4PL^3}{Ebh^3}.$$

The optimization problem is

$$\begin{aligned} \min_{b,h} \quad & \rho L b h, \\ \text{subject to} \quad & \frac{6PL}{bh^2} - \sigma_{\max} \le 0, \\ & \frac{4PL^3}{Ebh^3} - \delta_{\max} \le 0, \\ & b_{\min} \le b \le b_{\max}, \\ & h_{\min} \le h \le h_{\max}. \end{aligned}$$

Use

$$L = 1.2\ \text{m}, \qquad P = 8\ \text{kN}, \qquad E = 70\ \text{GPa},$$

$$\rho = 2700\ \text{kg/m}^3, \qquad \sigma_{\max} = 120\ \text{MPa}, \qquad \delta_{\max} = 4\ \text{mm},$$

and

$$0.01 \le b \le 0.12\ \text{m}, \qquad 0.02 \le h \le 0.20\ \text{m}.$$

1. Write the Lagrangian, including multipliers for the stress, displacement, and bound constraints.
2. Write all KKT conditions: stationarity, primal feasibility, dual feasibility, and complementary slackness.

3. Assume initially that the stress and displacement constraints are active and all bounds are inactive. Solve the two active constraints analytically for b and h.
4. Check whether the resulting design satisfies the variable bounds.
5. Compute the associated Lagrange multipliers and determine whether the assumed active set satisfies dual feasibility.
6. Enumerate all physically plausible active sets involving:
   (a) stress only;
   (b) displacement only;
   (c) stress and displacement; and
   (d) one performance constraint and one variable bound.
7. Determine the globally optimal feasible design by comparing all valid KKT candidates.
8. Interpret each nonzero multiplier as the local value of relaxing its corresponding engineering requirement.

Activity 3.3: Scaling, Conditioning, and Gradient-Method Convergence

Activity 3.3: Scaling, Conditioning, and Gradient-Method Convergence

Consider the quadratic optimization problem

$$\min_{\mathbf{x}\in\mathbb{R}^3} \quad f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T H\mathbf{x} - \mathbf{b}^T\mathbf{x},$$

where

$$H = \begin{bmatrix} 10^8 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 10^{-4} \end{bmatrix}, \qquad \mathbf{b} = \begin{bmatrix} 10^4 \\ 1 \\ 10^{-2} \end{bmatrix}.$$

1. Compute the exact minimizer

$$\mathbf{x}^* = H^{-1}\mathbf{b}.$$

2. Compute the 2-norm condition number $\kappa_2(H)$.
3. For fixed-step gradient descent,

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \alpha\nabla f\left(\mathbf{x}^{(k)}\right),$$

   derive the largest stable step size.
4. Derive the optimal fixed step size

$$\alpha^* = \frac{2}{\lambda_{\max} + \lambda_{\min}},$$

and the corresponding worst-case error contraction factor

$$q^* = \frac{\kappa_2(H) - 1}{\kappa_2(H) + 1}.$$

5. Estimate the number of iterations required to reduce the error norm by a factor of 10^{-6}.
6. Introduce the scaled variables

$$\mathbf{x} = S\mathbf{z}, \qquad S = \begin{bmatrix} 10^{-4} & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 10^{2} \end{bmatrix}.$$

7. Derive the scaled Hessian

$$\widetilde{H} = S^T H S$$

and compute its condition number.

8. Compare gradient descent, Newton's method, and BFGS on the original and scaled formulations from

$$\mathbf{x}^{(0)} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}.$$

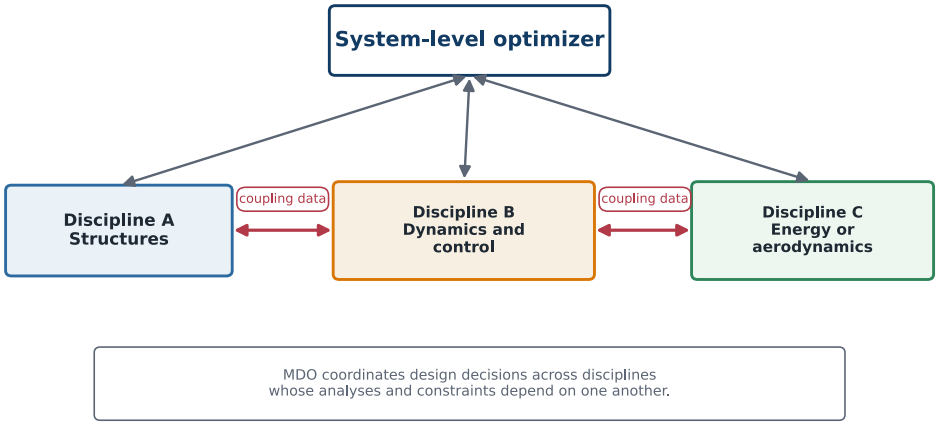
9. Explain why variable scaling changes numerical behavior without changing the underlying physical optimum.
10. Give one example of physical ill-conditioning that cannot be eliminated by simple coordinate scaling.

Multidisciplinary Design Optimization

Why MDO is needed

Large engineering systems involve interacting disciplines. A wind turbine, for example, combines structures, aerodynamics, dynamics, controls, electrical systems, economics, and manufacturing. Improving one discipline in isolation may worsen another.

Multidisciplinary design optimization coordinates decisions across coupled disciplines. CCD is a specialized dynamic-system MDO problem in which plant and control design are dominant interacting disciplines.



A conceptual MDO problem. Coupling variables communicate interactions between disciplines.

Coupling variables

A coupling variable is an output from one discipline that becomes an input to another. Examples include:

- structural deflection affecting aerodynamic loads;
- aerodynamic loads affecting structural stress;
- plant dynamics affecting controller performance;
- control forces affecting fatigue and actuator sizing; and
- thermal behavior affecting electrical efficiency.

Coupling creates implicit dependencies. One design change may influence many disciplines through several paths.

Coupled analysis equations

A two-discipline model can be written as

$$\mathbf{y}_1 = \mathbf{F}_1(\mathbf{x}, \mathbf{y}_2), \qquad \mathbf{y}_2 = \mathbf{F}_2(\mathbf{x}, \mathbf{y}_1).$$

A consistent multidisciplinary state satisfies both equations simultaneously. Consistency may be enforced by an inner coupled solver or directly within the optimization formulation.

MDO architectures: MDF, IDF, and ATC

An MDO *architecture* specifies which variables the optimizer controls directly and how multidisciplinary consistency is enforced. Architectures fall into two broad families: **monolithic**, which solve a single optimization

problem, and **distributed**, which decompose that problem into coordinated subproblems.

Multidisciplinary feasible (MDF). MDF is the architecture closest to a single-discipline problem: the optimizer sees only the original design variables, objective, and constraints, exactly as in earlier sections of this chapter. The difference is that evaluating the objective and constraints at each iteration requires converging the coupled system first, typically with a fixed-point iteration such as Gauss–Seidel. Every design the optimizer considers during the search is therefore multidisciplinary feasible—an advantage if the optimization is stopped early—but the coupled solve nested inside every function call can be expensive, and computing gradients requires *coupled* (total) derivatives rather than simple partial derivatives.

Individual discipline feasible (IDF). IDF instead gives each discipline an independent copy of the coupling variables it needs, called *target* variables, and lets each discipline solve its own equations without waiting on the others. The optimizer becomes responsible for driving these copies toward the actual coupling values by adding *consistency constraints*,

$$h_i^c = \hat{u}_i^t - \hat{u}_i = 0, \qquad i = 1, \ldots, m,$$

to the problem, where $\hat{u}_i^t$ is the target copy supplied to discipline i and $\hat{u}_i$ is the coupling variable that discipline actually computes. IDF trades a larger optimization problem (more variables and constraints) for disciplines that can be solved independently, in parallel, and it often converges better with gradient-based algorithms than the nested fixed-point iteration inside MDF. The cost is that multidisciplinary feasibility is only guaranteed once the optimization itself has converged, not at every intermediate iteration.

Analytical target cascading (ATC). ATC plays a role similar to IDF's consistency constraints but enforces them with penalty terms in the objective instead of equality constraints,

$$\min \quad f_0(x_0, \hat{u}^t) + \sum_{i=1}^{m} \Phi_i\left(x_{0i}^t - x_0,\ \hat{u}_i^t - \hat{u}_i\right),$$

where Φ_i is a penalty (commonly quadratic) that is driven toward zero as the penalty weights are increased across outer iterations. ATC originated as a way to cascade targets down a hierarchy of design requirements and later became a general MDO architecture; it converges to the same optimum as MDF and IDF only if the penalty weights are increased enough that every consistency penalty vanishes. Other distributed architectures—such as collaborative optimization and bilevel integrated system synthesis—use the same target/consistency idea with different coordination mechanics, but MDF, IDF, and ATC illustrate the essential trade-off: how tightly consistency is enforced during the search versus how independently each discipline can be solved.

Allison and Herber's review of MDO for dynamic engineering systems applies these same MDF/IDF ideas directly to CCD problems, and highlights a complication that rarely arises in static MDO: coupling variables are often entire time histories rather than scalars. In one example, a block-diagram model of a vane-airflow sensor is partitioned into a drag-force block and a response-dynamics block linked by drag-torque and angular-position signals; formulating this partition as IDF requires the optimizer to specify complete discretized torque and position trajectories as target coupling variables, since the signals connecting the two blocks are themselves time-varying. In a second example—an electric-vehicle design problem that separates motor design from vehicle/powertrain design—the coupling variables are function-valued motor properties (such as a torque–speed curve) rather than plain time histories; because directly discretizing such function-valued coupling variables would again inflate the problem dimension, reduced-dimension representations of these coupling variables are needed to keep the decomposition practical. Both examples reinforce the same lesson for CCD: whether a coupling variable is a scalar, a full trajectory, or an intermediate function-valued quantity should drive the choice of architecture and coupling representation, not just the architecture's usual convergence trade-offs.

The central lesson is to recognize when analyses depend on one another and represent those dependencies consistently—not merely to memorize architecture names.

CCD as MDO

Changing plant design can alter state-space matrices, natural frequencies, damping, actuator authority, control effort, and closed-loop performance. Changing controller design can alter performance, load histories, actuator sizing and power, and the value of passive stiffness, damping, or mass.

Plant–control coupling

These reciprocal effects mean the plant and controller cannot be evaluated as independent disciplines. Their consistency and shared performance must be handled at the system level.

Activity 3.4: Total Derivatives through a Coupled Multidisciplinary Model

Activity 3.4: Total Derivatives through a Coupled Multidisciplinary Model

Consider two coupled disciplines:

$$r_1 = y_1 - x_1^2 - \frac{1}{2}y_2 = 0,$$
$$r_2 = y_2 - \sin x_2 - \frac{1}{4}y_1 = 0.$$

The objective and constraint are

$$f = (y_1 - 1)^2 + (y_2 - 0.5)^2 + 0.1(x_1^2 + x_2^2),$$

and

$$g = y_1 + y_2 - 2 \le 0.$$

The design bounds are

$$-1.5 \le x_1 \le 1.5, \qquad -2 \le x_2 \le 2.$$

1. Write the coupled residual system in the compact form

$$\mathbf{r}(\mathbf{x}, \mathbf{y}) = \mathbf{0}.$$

2. Solve the coupled equations analytically for

$$y_1(\mathbf{x}), \qquad y_2(\mathbf{x}).$$

3. Derive the total state sensitivity using the implicit-function theorem:

$$\frac{d\mathbf{y}}{d\mathbf{x}} = -\left(\frac{\partial \mathbf{r}}{\partial \mathbf{y}}\right)^{-1} \frac{\partial \mathbf{r}}{\partial \mathbf{x}}.$$

4. Derive the total derivatives

$$\frac{df}{d\mathbf{x}}, \qquad \frac{dg}{d\mathbf{x}}.$$

5. Evaluate the analytical total derivatives at

$$\mathbf{x} = \begin{bmatrix} 0.8 \\ 0.4 \end{bmatrix}.$$

6. Verify the derivatives using complex-step differentiation.
7. Show that using only the partial derivative

$$\frac{\partial f}{\partial \mathbf{x}}$$

while ignoring

$$\frac{d\mathbf{y}}{d\mathbf{x}}$$

gives an incorrect search direction.

8. Formulate and solve the problem using a multidisciplinary feasible architecture in which the coupling equations are converged at every design iteration.
9. Reformulate the same problem using an individual-discipline feasible architecture in which y_1 and y_2 are optimization variables and the coupling equalities are consistency constraints.
10. Compare the two formulations in terms of variable count, constraint count, derivative structure, and convergence.

Activity 3.5: OpenMDAO Benchmark using the Sellar Coupled Problem

Activity 3.5: OpenMDAO Benchmark using the Sellar Coupled Problem

Consider the classical two-discipline Sellar problem:

$$\begin{aligned} y_1 &= z_1^2 + z_2 + x - 0.2y_2, \\ y_2 &= \sqrt{y_1} + z_1 + z_2. \end{aligned}$$

The optimization problem is

$$\begin{aligned} \min_{x,z_1,z_2} \quad & f = x^2 + z_2 + y_1 + e^{-y_2}, \\ \text{subject to} \quad & 3.16 - y_1 \leq 0, \\ & y_2 - 24 \leq 0, \end{aligned}$$

with bounds

$$0 \leq x \leq 10, \qquad -10 \leq z_1 \leq 10, \qquad 0 \leq z_2 \leq 10.$$

1. Identify:

   (a) local design variables;
   (b) shared design variables;
   (c) discipline outputs;
   (d) coupling variables; and
   (e) system-level objective and constraints.

2. Derive the coupled residual equations.
3. Derive the fixed-point iteration that results from solving the disciplines in Gauss–Seidel order.
4. Determine the local convergence condition for the coupling iteration from the spectral radius of the iteration Jacobian.
5. Implement the multidisciplinary feasible formulation in OpenMDAO, using a nonlinear block Gauss–Seidel or Newton solver for the coupled analysis.
6. Supply exact partial derivatives for both disciplines and use OpenMDAO's total-derivative check.
7. Solve the optimization using at least three different initial designs.
8. Reformulate the problem using an individual-discipline feasible architecture by promoting y_1 and y_2 to design variables and adding consistency constraints.
9. Compare the two architectures in terms of:

   (a) number of design variables;
   (b) number of constraints;
   (c) coupling-solver iterations;
   (d) optimizer iterations;
   (e) total derivative accuracy; and
   (f) final objective value.

10. Perturb the discipline equations by replacing

$$-0.2y_2$$

with

$$-\beta y_2,$$

where

$$0.1 \leq \beta \leq 0.8.$$

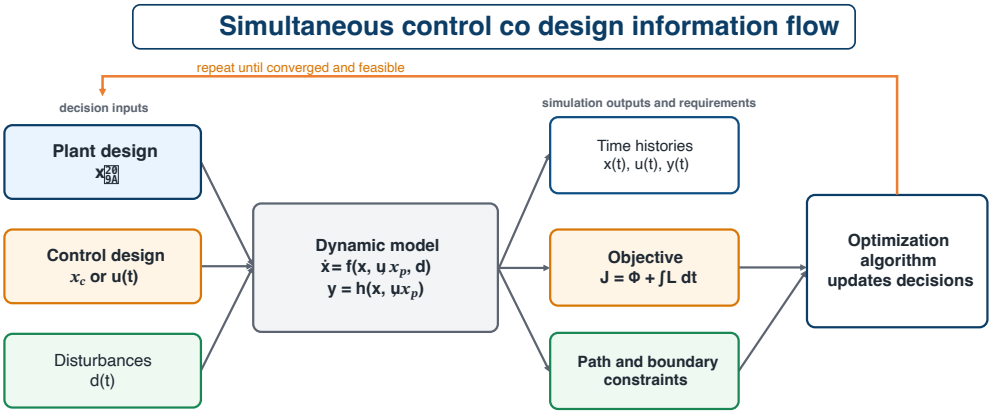
Determine how coupling strength affects fixed-point convergence and optimization performance.

11. Explain why a converged multidisciplinary analysis does not by itself prove that the optimization result is globally optimal or numerically reliable.

Optimization of Dynamic Systems

Why dynamic optimization is different

In static optimization, the model returns quantities for one design condition. In dynamic optimization, the model generates trajectories over time. Objectives and constraints may depend on the entire trajectory rather than on one scalar state.



Optimization of a dynamic system: design decisions enter the model, whose time histories determine performance and feasibility.

General formulation

A dynamic optimization problem can be written as

$$
\begin{aligned}
\underset{\mathbf{x},\mathbf{u}(\cdot),\mathbf{z}(\cdot)}{\text{minimize}} \quad & \Phi(\mathbf{z}(t_f),\mathbf{x}) + \int_{t_0}^{t_f} L(\mathbf{z}(t),\mathbf{u}(t),\mathbf{x},t)\,dt \\
\text{subject to} \quad & \dot{\mathbf{z}}(t) = \mathbf{f}(\mathbf{z}(t),\mathbf{u}(t),\mathbf{x},\mathbf{d}(t),t), \\
& \mathbf{c}(\mathbf{z}(t),\mathbf{u}(t),\mathbf{x},t) \le \mathbf{0}, \\
& \mathbf{b}(\mathbf{z}(t_0),\mathbf{z}(t_f),\mathbf{x}) = \mathbf{0}, \\
& \mathbf{x}_L \le \mathbf{x} \le \mathbf{x}_U.
\end{aligned}
$$

The objective combines a terminal term Φ, such as final error or energy, and an integral term L, such as accumulated tracking error, fatigue, energy, or control effort.

Path and boundary constraints

Path constraints must hold throughout the time interval:

$$
\mathbf{c}(\mathbf{z}(t),\mathbf{u}(t),\mathbf{x},t) \le \mathbf{0}, \qquad t \in [t_0, t_f].
$$

Examples include actuator, displacement, temperature, contact-force, power, and stress limits. A violation may occur between sampled times, so the discretization must be fine enough to detect it.

Boundary constraints apply at the initial or final time, for example

$$
\mathbf{z}(t_0) = \mathbf{z}_0, \qquad \|\mathbf{z}(t_f) - \mathbf{z}_{\text{target}}\| \le \varepsilon.
$$

Simulation-based optimization

A common workflow is to select design variables, simulate the dynamic model, compute objectives and constraints from the resulting trajectories, and return them to the optimizer. This works naturally when control is parameterized by a few variables. Optimizing an entire control trajectory requires specialized optimal-control transcription methods.

A small CCD formulation

For

$$m\ddot{q} + c\dot{q} + kq = u(t) + d(t),$$

let c, k, K_p, and K_d be decisions and use $u = -K_p q - K_d \dot{q}$. One possible objective is

$$J = \int_0^T \left(w_q q^2 + w_v \dot{q}^2 + w_u u^2 \right) dt + w_c c + w_k k.$$

Possible constraints include $|q(t)| \le q_{\max}$, $|u(t)| \le u_{\max}$, and bounds on c, k, K_p, and K_d. This is already a CCD problem because plant and controller variables are optimized together.

Activity 3.6: Discrete Adjoint Sensitivities for Simulation-Based Optimization

Activity 3.6: Discrete Adjoint Sensitivities for Simulation-Based Optimization

Consider the nonlinear time-marching model

$$x_{k+1} = x_k + h\left(-p_1 x_k - p_2 x_k^3 + u_k\right), \qquad k = 0, \dots, N-1,$$

with

$$x_0 = 1, \qquad h = 0.05, \qquad N = 100.$$

The design variables are

$$\mathbf{z} = \begin{bmatrix} p_1 & p_2 & u_0 & u_1 & \cdots & u_{N-1} \end{bmatrix}^T.$$

Minimize

$$\begin{aligned} J =& \frac{1}{2}\left(x_N - x_{\text{target}}\right)^2 \\ &+ \frac{h}{2}\sum_{k=0}^{N-1}\left(q x_k^2 + r u_k^2\right) + \frac{\gamma}{2}\left(p_1^2 + p_2^2\right), \end{aligned}$$

where

$$x_{\text{target}} = 0, \qquad q = 1, \qquad r = 0.02, \qquad \gamma = 0.01.$$

1. Derive the direct sensitivity recursion for

$$\frac{\partial x_k}{\partial z_j}.$$

2. Determine the computational complexity of forming the complete gradient using direct sensitivities.
3. Introduce a discrete adjoint sequence λ_k and derive its backward recursion.
4. Derive

$$\frac{\partial J}{\partial p_1}, \qquad \frac{\partial J}{\partial p_2}, \qquad \frac{\partial J}{\partial u_k}.$$

5. Implement direct sensitivity, adjoint sensitivity, forward finite difference, central finite difference, and complex-step gradients.
6. Perform a directional-derivative test using a random normalized direction $\mathbf{v}$:

$$\frac{J(\mathbf{z} + \epsilon \mathbf{v}) - J(\mathbf{z})}{\epsilon} \approx \nabla J(\mathbf{z})^T \mathbf{v}.$$

7. Compare gradient errors for

$$10^{-2} \leq \epsilon \leq 10^{-14}.$$

8. Compare computational cost as N increases from 100 to $10\,000$.
9. Explain why the adjoint method is especially attractive when the number of design variables is large and the number of scalar outputs is small.
10. Use an optimization solver to minimize J subject to

$$0.1 \leq p_1 \leq 3, \qquad 0 \leq p_2 \leq 2, \qquad -2 \leq u_k \leq 2.$$

A Practical Optimization Workflow

1. Define the mission and requirements

State what the system must accomplish, under which operating conditions, and how success will be measured. Do not begin with the solver.

2. Choose the design representation

Identify variables and bounds. Confirm that every variable corresponds to a realizable decision and that the representation is not unnecessarily redundant.

3. Build and verify the analysis model

Check equations, units, baseline behavior, and sensitivity to variables. An optimizer will exploit every model defect it can find.

4. Define objective and constraints

Separate preferences from requirements. Normalize quantities where appropriate and include limits that prevent physically meaningless exploitation.

5. Explore before optimizing

Use parameter sweeps, plots, and hand calculations to understand trends. Check smoothness and inspect likely active constraints.

6. Scale variables and functions

Bring typical values near order one. Select solver tolerances only after scaling.

7. Use more than one initial design

For nonlinear problems, compare several starts. Preserve the initial designs and solver settings for reproducibility.

8. Verify the solution

A trustworthy optimum should satisfy every constraint, have an understood termination message, exhibit acceptable optimality residuals, survive small perturbations, have interpretable active constraints, reproduce under independent simulation, and remain credible under higher-fidelity analysis.

9. Interpret the engineering result

Ask which variables moved, which constraints became active, what tradeoffs the optimum reveals, how uncertainty affects it, and whether the design is manufacturable, controllable, maintainable, and safe.

Audit checkpoint

For any optimization result, identify the active constraints, termination condition, initial guess, scaling choices, and evidence that the result is not a numerical artifact. Without these items, the result is not fully auditable.

Common formulation and solver mistakes

1. **Choosing dependent or redundant variables**, creating singularity and poor conditioning.

2. **Using an objective that does not reflect the mission.**
3. **Omitting essential constraints**, producing unrealistic optima.
4. **Mixing dimensional constraints with wildly different magnitudes.**
5. **Trusting one initial guess in a nonconvex problem.**
6. **Ignoring derivative quality**, causing false convergence.
7. **Failing to handle simulation failures** in unstable or invalid regions.
8. **Reporting only the objective** rather than the design, margins, trajectories, and diagnostics.
9. **Assuming convergence means correctness.** A solver can converge to an infeasible, poorly scaled, or incorrectly modeled solution.

Chapter Summary and Problems

Summary

Engineering design optimization formalizes a search using design variables, an objective, a predictive model, and constraints. The feasible space contains acceptable designs, and active constraints often explain the optimum. Nonlinear engineering problems can contain multiple local optima, so initialization and exploration matter. Gradient-based algorithms use derivative information efficiently; constrained methods balance improvement against active requirements. Scaling and conditioning strongly influence numerical reliability. MDO extends optimization across coupled disciplines, while dynamic-system optimization introduces trajectories, integral objectives, and path constraints.

Central lesson

Optimization is valuable when its formulation, model, derivatives, scaling, convergence evidence, and engineering interpretation are all credible. A solver result alone is not a design decision.

Key terms

Design variable; objective function; equality constraint; inequality constraint; bound; feasible design; infeasible design; active constraint; local optimum; global optimum; convexity; gradient; Hessian; line search; Newton method; quasi-Newton method; KKT conditions; Lagrange multiplier; scaling; conditioning; finite differences; complex-step differentiation; algorithmic (automatic) differentiation; direct and adjoint sensitivity methods; multidisciplinary design optimization; coupling variable; multidisciplinary feasible (MDF); individual discipline feasible (IDF); analytical target cascading (ATC); gradient-free optimization; CMA-ES; genetic algorithm; particle swarm optimization; path constraint; boundary constraint.

Problems

1. **KKT interpretation of an engineering optimum.** Minimize $J(t,r) = \rho\pi r^2 t + \alpha/t$ subject to $\sigma(t,r) = Fr/(2\pi r^2 t) \le \sigma_y/S_f$, $r \le r_{\max}$, and $t \ge t_{\min}$. Derive every KKT point and classify the globally optimal pressure-vessel design as the parameters vary.

2. **Scaling and solver conditioning.** A design vector contains length in millimeters, stiffness in newtons per meter, mass in kilograms, and a dimensionless gain, while constraint residuals span eight orders of magnitude. Construct diagonal variable and residual scalings from characteristic physical values and derive how those scalings transform the gradient, Jacobian, Hessian, and KKT system condition number.

3. **Second-order optimality in a nonconvex design.** For $f(x,y) = (x^2-1)^2 + \beta(y-x)^2$ subject to $x+y = 1$ and $x \ge 0$, derive all feasible stationary points and use the reduced Hessian to determine their local optimality as a function of $\beta > 0$.

4. **Adjoint sensitivity of a multidisciplinary analysis.** Coupling variables satisfy $y_1 = a_1(x, y_2)$ and $y_2 = a_2(x, y_1)$, while $J = J(x, y_1, y_2)$. Derive an adjoint system that computes dJ/dx with one linear solve independent of the number of design variables.

5. **MDF versus IDF equivalence.** For the two-discipline coupling model $y_1 = x_1 + \alpha y_2$ and $y_2 = x_2 + \beta y_1$, formulate multidisciplinary-feasible and individual-discipline-feasible optimization architectures and prove their local equivalence when $1 - \alpha\beta \neq 0$ and all consistency constraints are satisfied.

6. **Pareto geometry.** Consider $f_1(x) = (x-1)^2$ and $f_2(x) = \gamma(x+1)^2$ on $x \in [-2, 2]$. Derive the complete Pareto set and Pareto front, then characterize which Pareto points cannot be recovered by a strictly positive weighted-sum scalarization if a nonconvex perturbation $-\delta\cos(3\pi x)$ is added to both objectives.

7. **Reliability-based design optimization.** A stress limit state is $g(A, F, Y) = Y - F/A$ with independent $F \sim \mathcal{N}(\mu_F, \sigma_F^2)$ and $Y \sim \mathcal{N}(\mu_Y, \sigma_Y^2)$. Derive a first-order reliability constraint for $\mathbb{P}[g \le 0] \le 10^{-4}$ and solve

the minimum-area design including the sensitivity of the optimum to the target reliability index.

8. **Dynamic-system design optimization.** A thermal storage device follows $C(p)\dot{T} = -h(p)(T - T_a) + q(t)$ with $T(0) = T_0$. Formulate and solve for the design p that minimizes material cost plus integrated temperature deviation while satisfying $T(t) \le T_{\max}$, using an adjoint to derive the continuous total derivative.

9. **Trust-region SQP step.** For a nonlinear program with twice differentiable objective $f(x)$, equalities $c(x) = 0$, and inequalities $g(x) \le 0$, derive the exact trust-region SQP subproblem and a merit-function acceptance ratio that remains meaningful when the linearized constraints are inconsistent.

10. **Bilevel design sensitivity.** Let $\psi(p) = \min_c J(p, c)$ subject to $h(p, c) = 0$ and $g(p, c) \le 0$. Under LICQ, strict complementarity, and second-order sufficiency, derive $d\psi/dp$ from the lower-level Lagrangian and identify the precise regularity loss that makes the outer objective nonsmooth.

References and further reading

1. Martins, J. R. R. A., & Ning, A. (2021). *Engineering design optimization*. Cambridge University Press.

2. Nocedal, J., & Wright, S. J. (2006). *Numerical optimization* (2nd ed.). Springer. DOI: 10.1007/978-0-387-40065-5

3. Boyd, S., & Vandenberghe, L. (2004). *Convex optimization*. Cambridge University Press. DOI: 10.1017/CBO97805118

4. Allison, J. T., & Herber, D. R. (2014). Multidisciplinary design optimization of dynamic engineering systems. *AIAA Journal, 52*(4), 691–710. DOI: 10.2514/1.J052182

5. Herber, D. R., & Allison, J. T. (2019). Nested and simultaneous solution strategies for general combined plant and control design problems. *Journal of Mechanical Design, 141*(1), Article 011402. DOI: 10.1115/1.4040705

6. Bayat, S., Peterson, C., Lee, Y. H., Iori, J., & Allison, J. T. (2026). Advancing wind turbines through control co-design: An integrative review. *Applied Energy, 416*, Article 127951. DOI: 10.1016/j.apenergy.2026.127951

Mathematical Formulation of Control Co-Design

Chapter 4: Mathematical Formulation of Control Co-Design

Variables, trajectories, objectives, dynamics, and engineering constraints

> A control co-design problem becomes solvable only after the engineering question is written as an optimization problem with clear variables, objectives, and constraints.

The previous chapters explained why control co-design (CCD) matters, developed the required dynamic-system concepts, and introduced engineering optimization. We can now answer a practical question: **How do we write a CCD problem mathematically?**

A weak formulation can make a good engineering idea impossible to solve. A clear formulation reveals problem structure, design tradeoffs, and appropriate computational methods. In CCD, it must describe plant and controller variables, time-dependent state and control trajectories, physical dynamics, objectives, and path and boundary constraints.

Learning objectives

After completing this chapter, you should be able to:

1. identify plant variables, control variables, state trajectories, and control trajectories;
2. express system dynamics as equality constraints;
3. distinguish and write path and boundary constraints;
4. explain Lagrange, Mayer, and Bolza objectives;
5. write a general continuous-time CCD formulation; and
6. explain how a dynamic formulation differs from static optimization.

Chapter roadmap

Engineering question → Variables and trajectories → Dynamics → Constraints → Objective → Continuous-time CCD problem

Formulation and Design Variables

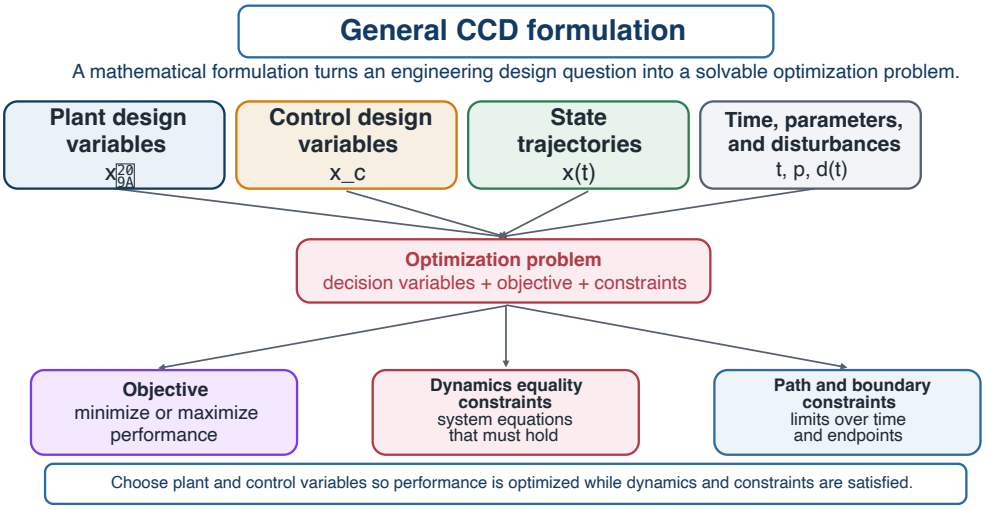
Why formulation matters

A CCD study begins as an engineering question:

> What tower geometry and pitch-control tuning should reduce wind-turbine loads while maintaining power quality?
>
> What suspension parameters and feedback gains should balance ride comfort, road holding, and actuator effort?

To turn such questions into optimization problems, we must identify:

- the **decision variables**;
- the quantity to **minimize or maximize**;
- the **physical laws** the design must obey;
- the **engineering limits** that cannot be violated; and
- the **time interval** over which the problem applies.



The main ingredients of a mathematical CCD formulation.

A formulation determines what the optimizer is allowed to do. Poorly defined variables, a misleading objective, or missing physics can produce a numerically polished but physically meaningless result.

Plant design variables

Plant variables describe physical-system, structural, mechanism, or hardware decisions. Denote them by

$$\mathbf{x}_p \in \mathbb{R}^{n_p}.$$

Examples include suspension stiffness and damping, structural dimensions, inertia and mass distribution, actuator or sensor placement, buoy or rotor geometry, and power-take-off parameters. These decisions often enter the dynamic model directly through mass, stiffness, damping, actuation, or output equations.

Typical examples are:

- **Vehicle suspension:** k_s, c_s, and tire stiffness k_t.

- **Robot arm:** link lengths, gear ratios, motor sizes, and joint stiffness.
- **Wind turbine:** blade properties, tower diameter, and drivetrain inertia.
- **Marine energy device:** buoy radius, draft, PTO characteristics, and mooring stiffness.

Plant variables usually have defensible bounds,

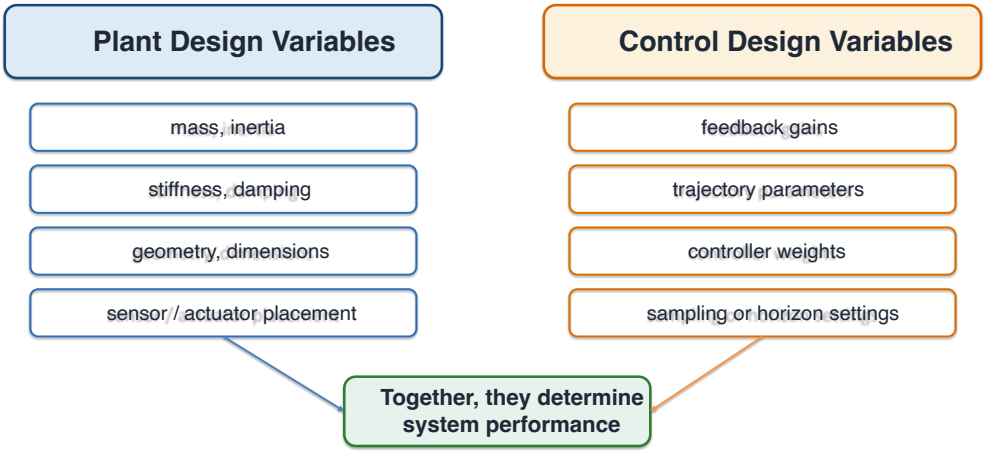
$$\mathbf{x}_p^L \leq \mathbf{x}_p \leq \mathbf{x}_p^U,$$

reflecting manufacturability, safety, packaging, cost, and physical realism.

Control design variables

Control variables define how the controller behaves. Denote them by

$$\mathbf{x}_c \in \mathbb{R}^{n_c}.$$



Plant and control variables are distinct, but both affect system performance and must be chosen together.

Control variables may include PID or state-feedback gains, weighting matrices, trajectory parameters, feedforward coefficients, switching thresholds, sampling times, prediction horizons, or coefficients in a control-law parameterization.

Parameterized control and free trajectories

There are two common control representations:

1. **Parameterized control design:** a law depends on finite parameters $\mathbf{x}_c$, for example

$$u(t) = -K(\mathbf{x}_c)x(t).$$

2. **Direct trajectory optimization:** the function $u(t)$ itself is a time-varying decision.

Either can be used in CCD. The appropriate choice depends on the problem structure, intended controller implementation, and numerical method.

Partitioning the control-variable set

It is often useful to be explicit about which part of $\mathbf{x}_c$ is time-independent and which is a genuine trajectory. Writing

$$\mathbf{x}_c = \begin{bmatrix} \mathbf{p} \\ \mathbf{u} \end{bmatrix},$$

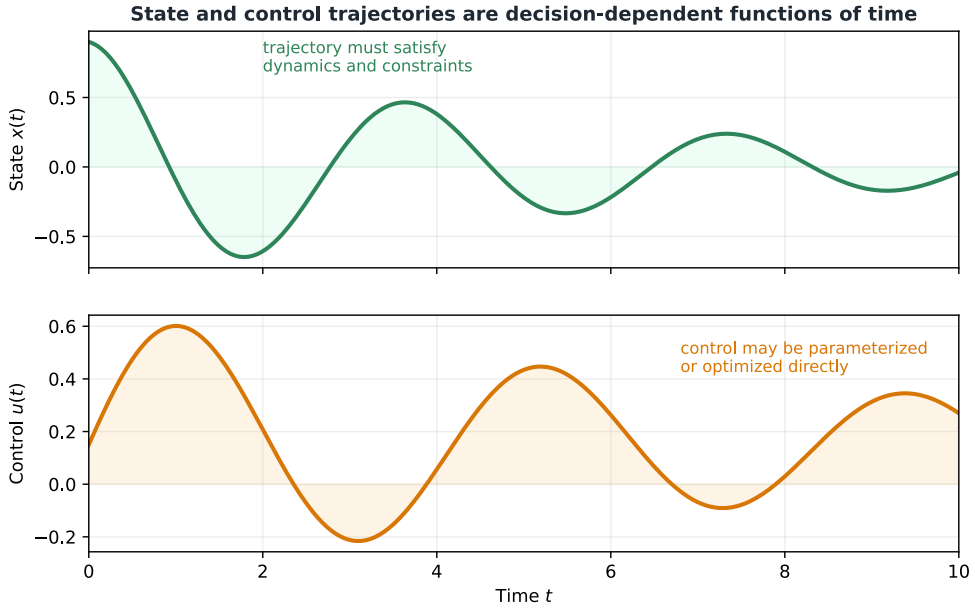
separates **control parameters** $\mathbf{p}$—time-independent quantities such as feedback gains, weighting-matrix entries, or a prediction horizon—from **open-loop control (OLC) variables** $\mathbf{u}(\cdot)$, the time-varying trajectory itself. A formulation may include only $\mathbf{p}$ (a fully parameterized feedback law), only $\mathbf{u}(\cdot)$ (direct trajectory optimization with no parametric structure), or both at once, for example a parameterized feedback law whose reference trajectory is itself optimized. The general continuous-time CCD formulation developed later in this chapter accommodates all three cases without change; only the content of $\mathbf{x}_c$ differs.

State Trajectories and Dynamic Equality Constraints

Time-dependent decisions

Dynamic optimization involves functions of time. The state and control trajectories are

$$\mathbf{x}(t) \in \mathbb{R}^{n_x}, \qquad \mathbf{u}(t) \in \mathbb{R}^{n_u}, \qquad t \in [t_0, t_f].$$



State and control trajectories must evolve consistently with the system dynamics and constraints.

Trajectories matter because comfort, energy, tracking, stress, and control effort depend on behavior over an interval—not merely one instant. Initial and terminal conditions may be fixed or constrained:

$$\mathbf{x}(t_0) = \mathbf{x}_0, \qquad \mathbf{x}(t_f) \in \mathcal{X}_f.$$

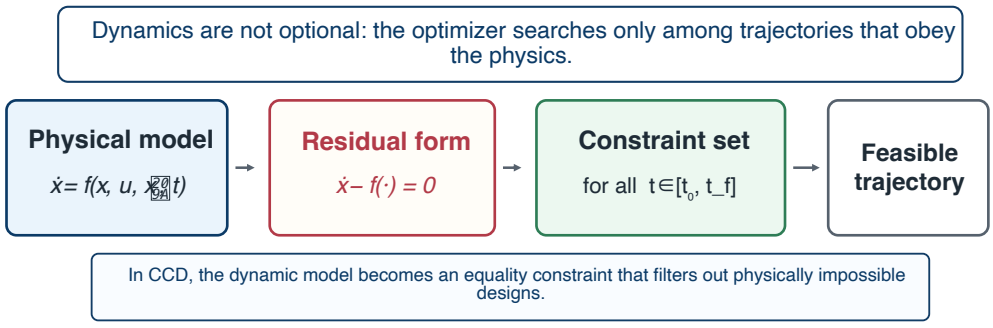
The final time may itself be fixed or optimized.

Dynamics as equality constraints

The dynamics enforce physical laws. A general continuous-time model is

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), \mathbf{x}_p, \mathbf{x}_c, \mathbf{d}(t), t).$$

This is an equality constraint restricting the feasible trajectory set.



The optimizer must search among trajectories that satisfy the physical model.

In residual form,

$$\dot{\mathbf{x}}(t) - \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), \mathbf{x}_p, \mathbf{x}_c, \mathbf{d}(t), t) = \mathbf{0}, \qquad t \in [t_0, t_f].$$

Mass–spring–damper example

Let $\mathbf{x}_p = [m, c, k]^T$, $x_1 = x$, and $x_2 = \dot{x}$. Then

$$\dot{x}_1 = x_2, \qquad \dot{x}_2 = -\frac{k}{m}x_1 - \frac{c}{m}x_2 + \frac{1}{m}u.$$

These equations define the feasible state trajectories for any candidate plant and control input.

Other equality constraints

CCD formulations may also contain algebraic equalities for kinematic loop closure, geometric compatibility, steady-state equations, periodicity, and actuator or sensor calibration. The equality set can therefore be larger than the state equations alone.

Differential-algebraic form

When the dynamics contain algebraic equalities in addition to the state derivatives, the model is a **differential-algebraic equation (DAE)** rather than a plain ODE. In semi-explicit form,

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \boldsymbol{\gamma}(t), \mathbf{u}(t), \mathbf{x}_p, \mathbf{x}_c, t), \qquad \mathbf{0} = \mathbf{f}_a(\mathbf{x}(t), \boldsymbol{\gamma}(t), \mathbf{u}(t), \mathbf{x}_p, \mathbf{x}_c, t),$$

where $\boldsymbol{\gamma}(t)$ is an **algebraic variable**: a quantity that must satisfy the algebraic constraint $\mathbf{f}_a(\cdot) = \mathbf{0}$ at every instant but has no derivative of its own appearing in the model. Solving the algebraic equation in Eq. () for $\boldsymbol{\gamma}(t)$ requires that the Jacobian $\partial \mathbf{f}_a / \partial \boldsymbol{\gamma}$ be nonsingular; a DAE with this property is called **index-1**, where the *index* counts how many times the algebraic equations must be differentiated before the system reduces to an ordinary differential equation. Index-1 DAEs are the most tractable case and the one nearly all CCD dynamic-optimization software targets.

Algebraic variables arise naturally in CCD. Kinematic loop closure and geometric compatibility, mentioned above, are two sources; another is a path inequality constraint that becomes active during part of the trajectory. Every time an inequality constraint is pinned to its bound, one degree of freedom is lost: a state variable must become dependent on the others through the newly active algebraic equation. In actively controlled systems, the control input is the natural candidate to absorb this loss—an actuator command that saturates, for instance, stops behaving as a free trajectory and becomes an algebraic variable determined by the active bound—while the state variables continue to evolve according to the underlying physics. Each additional active inequality constraint can raise the DAE index and make the numerical solution correspondingly harder.

Activity 4.1: Formulating Actuator Dynamics, Rate Limits, and Energy Limits

Activity 4.1: Formulating Actuator Dynamics, Rate Limits, and Energy Limits

Consider the plant

$$\dot{x}_1 = x_2, \qquad m\dot{x}_2 = -kx_1 - cx_2 + f_a + d(t),$$

where $f_a(t)$ is the actual actuator force. The actuator command is $u_c(t)$, but the actuator has first-order dynamics:

$$\dot{f}_a = \frac{1}{\tau_a}\left(u_c - f_a\right).$$

The actuator capacity and bandwidth are design variables:

$$0.5 \le F_{\max} \le 5, \qquad 0.02 \le \tau_a \le 0.3.$$

The actuator is subject to

$$|u_c(t)| \le F_{\max}, \qquad |f_a(t)| \le F_{\max}, \qquad |\dot{f}_a(t)| \le R_{\max},$$

and

$$\int_0^T f_a(t)^2 dt \le E_{\max}.$$

1. Construct an augmented state vector that includes the actuator dynamics.
2. Introduce an energy-accumulation state $e(t)$ such that

$$\dot{e} = f_a^2, \qquad e(0) = 0.$$

3. Rewrite the energy limit as a terminal boundary constraint.
4. Rewrite the actuator-rate constraint entirely in terms of u_c, f_a, and τ_a, without using $\dot{f}_a$ explicitly.
5. Write the complete augmented dynamics in residual form

$$\mathbf{r}\left(\dot{\mathbf{x}}, \mathbf{x}, u_c, \mathbf{x}_p, t\right) = \mathbf{0}.$$

6. Identify which of the following are plant variables, control trajectories, states, path constraints, and boundary constraints:

$$k, \quad c, \quad F_{\max}, \quad \tau_a, \quad u_c(t), \quad f_a(t), \quad e(t).$$

7. Explain why imposing only

$$|u_c(t)| \le F_{\max}$$

is not sufficient to guarantee that the actual actuator force is physically feasible.

8. Add the actuator-design penalty

$$J_{\text{act}} = \alpha_F F_{\max}^2 + \frac{\alpha_\tau}{\tau_a},$$

and explain the physical tradeoff represented by the second term.

Path and Boundary Constraints

Path constraints

A path constraint must hold throughout the time interval:

$$\mathbf{g}(\mathbf{x}(t), \mathbf{u}(t), \mathbf{x}_p, \mathbf{x}_c, \mathbf{d}(t), t) \leq \mathbf{0}, \qquad t \in [t_0, t_f].$$

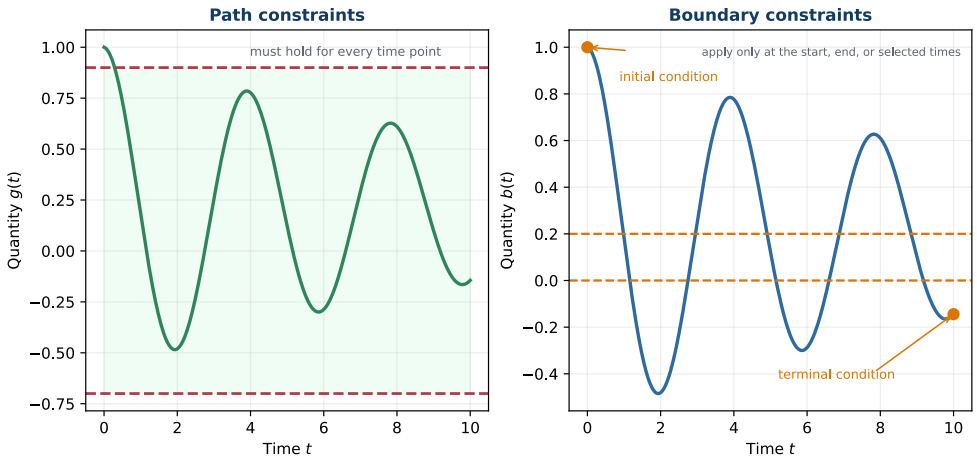
Examples include actuator saturation, displacement and angle limits, structural loads, temperature, speed, and safety envelopes.

Boundary constraints

Boundary constraints apply at the start, end, or designated points:

$$\mathbf{b}(\mathbf{x}(t_0), \mathbf{x}(t_f), \mathbf{u}(t_0), \mathbf{u}(t_f), \mathbf{x}_p, \mathbf{x}_c, t_0, t_f) = \mathbf{0}.$$

They can enforce initial states, final positions, periodicity, terminal performance, endpoint compatibility, or mission completion. Boundary inequalities are also possible.



Path constraints apply throughout time; boundary constraints apply at selected points.

For example,

$$|u(t)| \leq u_{\max}, \qquad \forall t,$$

is an actuator path constraint, while

$$q(t_f) = q_f$$

is a terminal boundary constraint.

Key idea

Path constraints protect the design at all times. Boundary constraints enforce what must happen at the beginning, end, or another designated point. The distinction affects both modeling and numerical solution.

Physical-design constraints

Some path-like constraints depend only on the plant design and the trajectory it produces—not on an instantaneous operating limit such as actuator saturation. It is common in the CCD literature to separate these out as a dedicated **physical-constraint function**,

$$\mathbf{g}_p(\mathbf{x}(t), \mathbf{x}_p, t) \leq \mathbf{0},$$

distinct from the general path constraint $\mathbf{g}(\cdot)$ above. $\mathbf{g}_p(\cdot)$ captures requirements such as material stress or fatigue limits, buckling, deflection, and packaging or geometric envelopes—quantities that describe whether the *hardware itself* is physically realizable and durable, as opposed to quantities that describe instantaneous *operating* behavior such as actuator saturation or a safety envelope. The distinction is one of engineering bookkeeping rather than mathematics: both are inequality constraints evaluated along the trajectory, and a complete CCD formulation must respect both.

An example makes the distinction concrete. In a quarter-car active-suspension co-design study, the plant vector expands well beyond an abstract stiffness and damping pair to the physical spring and damper geometry—wire diameter, coil diameter, pitch, number of active coils, valve diameter, damper piston diameter, and damper stroke. The resulting $\mathbf{g}_p(\cdot)$ set includes a spring-buckling constraint, a fatigue criterion combining mean and alternating shear stress, a damper-fluid thermal limit, a maximum damper seal pressure, and several packaging and clearance constraints tying the spring and damper geometry to the available pocket length in the vehicle. The examples section later in this chapter revisits this system in detail.

Lagrange, Mayer, and Bolza Objectives

The objective defines what "good" means. CCD objectives may represent energy, ride quality, tracking error, power, structural load, cost, or combinations of goals.

Lagrange form

The Lagrange form accumulates a running cost:

$$J = \int_{t_0}^{t_f} L(\mathbf{x}(t), \mathbf{u}(t), \mathbf{x}_p, \mathbf{x}_c, t)\, dt.$$

It is appropriate for total energy, integrated tracking error, mean-squared acceleration, or accumulated control effort. For example,

$$J = \int_{t_0}^{t_f} \left(q_1 x(t)^2 + q_2 \dot{x}(t)^2 + r u(t)^2\right) dt.$$

Mayer form

The Mayer form depends only on terminal quantities:

$$J = \Phi(\mathbf{x}(t_f), \mathbf{x}_p, \mathbf{x}_c, t_f).$$

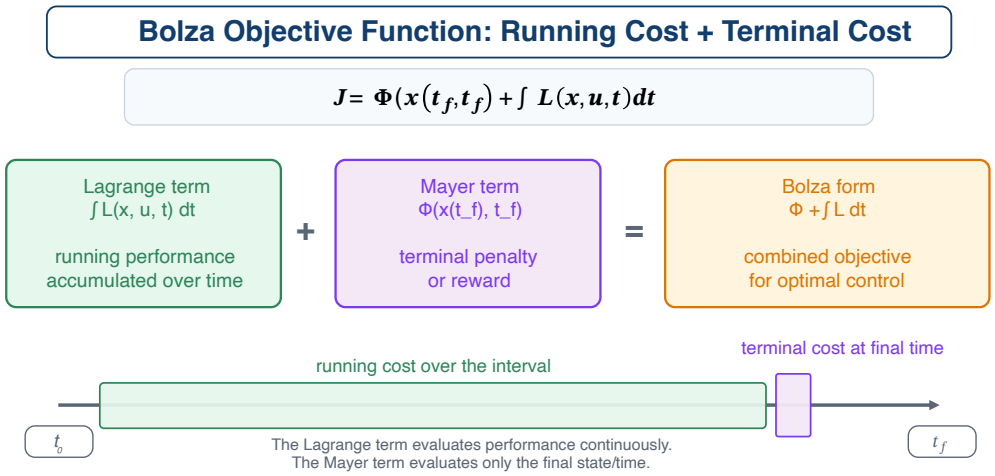
Examples include minimizing final error or final time and maximizing final stored energy:

$$J = (x(t_f) - x_f)^2.$$

Bolza form

The Bolza form combines terminal and running terms:

$$J = \Phi(\mathbf{x}(t_f), \mathbf{x}_p, \mathbf{x}_c, t_f) + \int_{t_0}^{t_f} L(\mathbf{x}(t), \mathbf{u}(t), \mathbf{x}_p, \mathbf{x}_c, t)\, dt.$$



Lagrange, Mayer, and Bolza objective forms.

The forms are closely related. Lagrange and Mayer problems are special cases of Bolza. A Bolza problem can also be rewritten in Mayer form by introducing an auxiliary state. Keeping the conceptual distinction makes engineering formulations easier to interpret.

A more general terminal term. Some CCD formulations let the terminal-cost term depend on the initial state as well as the final state,

$$J = \mathcal{M}(\mathbf{x}(t_0), \mathbf{x}(t_f), \mathbf{x}_p, \mathbf{x}_c, t_0, t_f) + \int_{t_0}^{t_f} L(\mathbf{x}(t), \mathbf{u}(t), \mathbf{x}_p, \mathbf{x}_c, t)\, dt.$$

This more general Mayer term $\mathcal{M}(\cdot)$ is useful whenever performance depends on how the state has changed between endpoints rather than on the terminal state alone—for example, a periodicity requirement, or an objective that rewards net displacement over a maneuver. The $\Phi(\mathbf{x}(t_f), \cdots)$ form used above is the common special case in which $\mathcal{M}(\cdot)$ does not depend on $\mathbf{x}(t_0)$.

Activity 4.2: Conversion among Lagrange, Mayer, and Bolza Forms

Activity 4.2: Conversion among Lagrange, Mayer, and Bolza Forms

Consider the dynamic optimization problem

$$\min_{u(\cdot)} \quad J = \int_0^{t_f} \left[x(t)^2 + \rho u(t)^2\right] dt,$$

subject to

$$\dot{x}(t) = ax(t) + bu(t), \qquad x(0) = x_0.$$

1. Introduce an auxiliary state $z(t)$ and convert the Lagrange objective into an equivalent Mayer objective.
2. State the initial condition for $z(t)$ and write the augmented dynamics.
3. Prove that

$$z(t_f) = \int_0^{t_f} \left[x(t)^2 + \rho u(t)^2\right] dt.$$

4. Now consider the Bolza objective

$$J = \frac{1}{2} q_f x(t_f)^2 + \int_0^{t_f} \left[x(t)^2 + \rho u(t)^2\right] dt.$$

   Convert it into a pure Mayer problem.

5. Add the integral energy constraint

$$\int_0^{t_f} u(t)^2 dt \le E_{\max}.$$

   Introduce another auxiliary state and rewrite the integral inequality as a terminal boundary constraint.

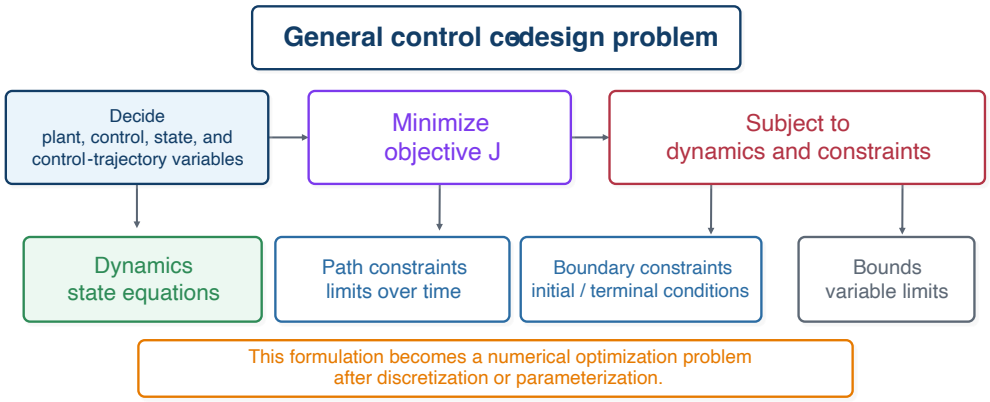
6. Show that the resulting augmented system contains no integral objective or integral constraint.
7. Explain why the Mayer transformation is mathematically exact but may still affect numerical conditioning and scaling.
8. Generalize the transformation to the vector running cost

$$\mathbf{L}\left(\mathbf{x}, \mathbf{u}, \mathbf{x}_p, t\right) \in \mathbb{R}^{n_L}.$$

A General Continuous-Time CCD Formulation

Combining the pieces gives the broad formulation

$$
\begin{aligned}
\underset{\mathbf{x}_p,\mathbf{x}_c,\mathbf{x}(\cdot),\mathbf{u}(\cdot)}{\text{minimize}} \quad & J = \Phi(\mathbf{x}(t_f), \mathbf{x}_p, \mathbf{x}_c, t_f) + \int_{t_0}^{t_f} L(\mathbf{x}(t), \mathbf{u}(t), \mathbf{x}_p, \mathbf{x}_c, t)\, dt \\
\text{subject to} \quad & \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), \mathbf{x}_p, \mathbf{x}_c, \mathbf{d}(t), t), \\
& \mathbf{g}(\mathbf{x}(t), \mathbf{u}(t), \mathbf{x}_p, \mathbf{x}_c, \mathbf{d}(t), t) \leq \mathbf{0}, \\
& \mathbf{b}(\mathbf{x}(t_0), \mathbf{x}(t_f), \mathbf{u}(t_0), \mathbf{u}(t_f), \mathbf{x}_p, \mathbf{x}_c, t_0, t_f) = \mathbf{0}, \\
& \mathbf{x}_p^L \leq \mathbf{x}_p \leq \mathbf{x}_p^U, \\
& \mathbf{x}_c^L \leq \mathbf{x}_c \leq \mathbf{x}_c^U, \qquad t \in [t_0, t_f].
\end{aligned}
$$



Decision variables, objective, physical dynamics, and constraint sets together define the CCD problem.

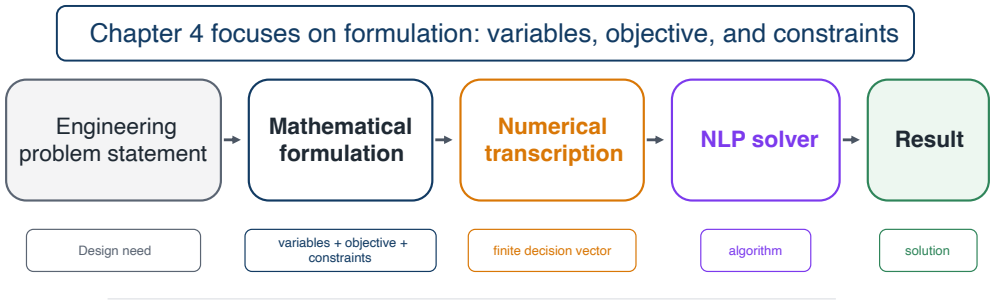
The objective measures performance. The differential equations encode physical behavior. Path constraints impose time-dependent engineering limits. Boundary constraints enforce endpoint requirements. Bounds restrict finite-dimensional plant and controller decisions.

Recovering passive design as a special case

The general formulation above specializes naturally. If the controller is removed from the decision set entirely—no $\mathbf{x}_c$, no $\mathbf{u}(\cdot)$, and the dynamics are evaluated with $\mathbf{u} \equiv \mathbf{0}$—the same structure reduces to a **passive physical-system design** problem: minimize an objective built from $\Phi(\mathbf{x}(t_f), \mathbf{x}_p, t_f)$ (or a Lagrange or Bolza generalization of it) subject to $\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{x}_p, t)$ and the physical-constraint set $\mathbf{g}_p(\mathbf{x}(t), \mathbf{x}_p, t) \leq \mathbf{0}$ introduced earlier. Restoring $\mathbf{x}_c$ and $\mathbf{u}(\cdot)$ as decisions recovers the full simultaneous co-design problem above. The two formulations share the same dynamics structure, the same objective forms, and the same physical-constraint function; only the decision-variable set differs. Seeing passive design as a restriction of the general CCD formulation—rather than as a separately posed, unrelated problem—is what makes it meaningful to compare a passive-only design against a co-designed one on a common objective.

Continuous-time and numerical formulations

The continuous-time problem is infinite-dimensional because state and control trajectories are functions. Numerical solution requires a finite representation through parameterization, sampling, or transcription.



Formulation translates an engineering statement into a solvable optimization problem.

Mathematical formulation precedes numerical transcription and solution.

The numerical stage typically parameterizes trajectories, enforces dynamics through simulation or defect equations, and approximates integrals by quadrature. Those choices should come after the engineering formulation is correct.

Activity 4.3: Complete Continuous-Time CCD Formulation with Free Final Time

Activity 4.3: Complete Continuous-Time CCD Formulation with Free Final Time

Consider the controlled mass–spring–damper system

$$m\ddot{q}(t) + c\dot{q}(t) + kq(t) = u(t) + d(t),$$

where

$$m = 1.5\ \text{kg}, \qquad d(t) = 2e^{-0.5t}\sin(3t).$$

The plant design variables are

$$0.5 \le k \le 8, \qquad 0.1 \le c \le 4, \qquad 0.5 \le F_{\max} \le 6,$$

and the final time is also a design variable:

$$2 \le t_f \le 8.$$

The control trajectory satisfies

$$|u(t)| \le F_{\max}.$$

The initial and terminal conditions are

$$q(0) = 1, \qquad \dot{q}(0) = 0,$$

and

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

The displacement and velocity must satisfy

$$|q(t)| \le 1.2, \qquad |\dot{q}(t)| \le 2.5.$$

Minimize

$$J = \int_0^{t_f} \left[q(t)^2 + 0.1\dot{q}(t)^2 + 0.02u(t)^2 \right] dt + 0.03k^2 + 0.02c^2 + 0.04F_{\max}^2 + 0.2t_f.$$

1. Define a first-order state vector and derive the state equations in the form

$$\dot{\mathbf{x}} = \mathbf{f}\left(\mathbf{x}, u, \mathbf{x}_p, t\right).$$

2. Define the complete plant-design vector, control trajectory, state trajectory, and free-time variable.
3. Write the problem in the general Bolza form

$$\begin{aligned} \min_{\mathbf{x}_p, u(\cdot), \mathbf{x}(\cdot), t_f} \quad & \Phi\left(\mathbf{x}(0), \mathbf{x}(t_f), \mathbf{x}_p, t_f\right) + \int_0^{t_f} L\left(\mathbf{x}, u, \mathbf{x}_p, t\right) dt, \\ \text{subject to} \quad & \dot{\mathbf{x}} - \mathbf{f}\left(\mathbf{x}, u, \mathbf{x}_p, t\right) = \mathbf{0}, \end{aligned}$$

   together with all path, boundary, and bound constraints.
4. Classify every constraint as one of the following:
   (a) dynamic equality constraint;
   (b) path inequality constraint;
   (c) boundary equality constraint; or
   (d) simple bound.
5. Determine whether $F_{\max}$ is redundant when $u(t)$ is also bounded. Explain why it is still a meaningful plant or actuator-design variable in this formulation.
6. Replace the terminal equalities with the terminal tolerance

$$\left\|\mathbf{x}(t_f)\right\|_2 \le 10^{-2},$$

   and write the equivalent scalar inequality constraint.
7. Explain which terms in the objective discourage the optimizer from selecting the largest possible actuator, the stiffest possible spring, or the longest possible maneuver time.

Activity 4.4: Time Normalization for a Free-Final-Time CCD Problem

Activity 4.4: Time Normalization for a Free-Final-Time CCD Problem

Consider the general free-final-time CCD problem

$$\min_{\mathbf{x}_p, \mathbf{x}_c, \mathbf{u}(\cdot), t_f} \quad \Phi\left(\mathbf{x}(t_f), \mathbf{x}_p, \mathbf{x}_c, t_f\right) + \int_{t_0}^{t_f} L\left(\mathbf{x}, \mathbf{u}, \mathbf{x}_p, \mathbf{x}_c, t\right) dt,$$

subject to

$$\dot{\mathbf{x}} = \mathbf{f}\left(\mathbf{x}, \mathbf{u}, \mathbf{x}_p, \mathbf{x}_c, t\right).$$

Introduce normalized time

$$\tau = \frac{t - t_0}{t_f - t_0}, \qquad 0 \le \tau \le 1.$$

1. Derive

$$\frac{dt}{d\tau} = t_f - t_0.$$

2. Show that the transformed dynamics are

$$\frac{d\mathbf{x}}{d\tau} = (t_f - t_0)\mathbf{f}\left(\mathbf{x}, \mathbf{u}, \mathbf{x}_p, \mathbf{x}_c, t(\tau)\right).$$

3. Transform the integral objective to the fixed interval $\tau \in [0, 1]$.
4. Transform the path constraint

$$\mathbf{g}\left(\mathbf{x}(t), \mathbf{u}(t), \mathbf{x}_p, t\right) \le \mathbf{0}.$$

5. Transform the explicit time-dependent disturbance

$$d(t) = A\sin(\omega t + \phi).$$

6. For the scalar system

$$\dot{x} = -px + u, \qquad x(0) = 1, \qquad x(t_f) = 0,$$

write the complete normalized-time formulation when

$$J = t_f + \int_0^{t_f} \left(x^2 + 0.1u^2\right) dt.$$

7. Explain why t_f appears in both the transformed dynamics and the transformed running cost.
8. State the condition that must be imposed to prevent the optimizer from selecting $t_f = t_0$.

Activity 4.5: Multi-Scenario Robust CCD Formulation

Activity 4.5: Multi-Scenario Robust CCD Formulation

A controlled system is described under scenario s by

$$\dot{\mathbf{x}}_s = \mathbf{f}\left(\mathbf{x}_s, \mathbf{u}_s, \mathbf{x}_p, \boldsymbol{\theta}_s, t\right), \qquad s = 1, \ldots, N_s,$$

where $\mathbf{x}_p$ is a common plant design and $\boldsymbol{\theta}_s$ contains scenario-dependent parameters. Each scenario has objective

$$J_s = \Phi_s\left(\mathbf{x}_s(t_f), \mathbf{x}_p\right) + \int_0^{t_f} L_s\left(\mathbf{x}_s, \mathbf{u}_s, \mathbf{x}_p, t\right) dt.$$

The robust objective is

$$J_R = \sum_{s=1}^{N_s} p_s J_s + \lambda \max_s J_s,$$

where

$$p_s \geq 0, \qquad \sum_{s=1}^{N_s} p_s = 1.$$

1. Write the complete multi-scenario CCD problem when the plant design $\mathbf{x}_p$ is common to all scenarios but each scenario has its own open-loop control trajectory $\mathbf{u}_s(t)$.
2. Introduce an epigraph variable η and replace

$$\max_s J_s$$

   with smooth algebraic inequalities.
3. Write the resulting objective and all epigraph constraints.
4. Add scenario-wise path constraints

$$\mathbf{g}_s\left(\mathbf{x}_s, \mathbf{u}_s, \mathbf{x}_p, t\right) \leq \mathbf{0}.$$

5. Add the common actuator-capacity variable $F_{\max}$ and impose

$$|u_s(t)| \leq F_{\max}, \qquad s = 1, \ldots, N_s.$$

6. Now assume that one causal feedback controller

$$\mathbf{u}_s(t) = \boldsymbol{\pi}\left(\mathbf{y}_s(t); \mathbf{x}_c\right)$$

must be shared by all scenarios. Rewrite the decision-variable set.

7. Explain the difference between scenario-dependent open-loop controls and a common feedback law from the viewpoint of information availability and implementability.
8. If each scenario has n_x states, n_u controls, and $N+1$ transcription nodes, determine the total number of trajectory decision variables for:
   (a) scenario-dependent open-loop controls; and
   (b) common parameterized feedback with n_c controller parameters.

Activity 4.6: Multiphase Hybrid Control Co-Design Formulation

Activity 4.6: Multiphase Hybrid Control Co-Design Formulation

A robotic system performs two phases:

1. a free-motion phase; and
2. a contact phase.

During phase 1,

$$\dot{\mathbf{x}}^{(1)} = \mathbf{f}^{(1)}\left(\mathbf{x}^{(1)}, \mathbf{u}^{(1)}, \mathbf{x}_p, t\right), \qquad t \in [t_0, t_s].$$

At contact, the state undergoes the jump

$$\mathbf{x}^{(2)}(t_s^+) = \boldsymbol{\Delta}\left(\mathbf{x}^{(1)}(t_s^-), \mathbf{x}_p\right).$$

During phase 2,

$$\dot{\mathbf{x}}^{(2)} = \mathbf{f}^{(2)}\left(\mathbf{x}^{(2)}, \mathbf{u}^{(2)}, \mathbf{x}_p, t\right), \qquad t \in [t_s, t_f].$$

The switching time t_s and final time t_f are decision variables. The objective is

$$\begin{aligned} J = &\int_{t_0}^{t_s} L^{(1)}\left(\mathbf{x}^{(1)}, \mathbf{u}^{(1)}, \mathbf{x}_p, t\right) dt \\ &+ \int_{t_s}^{t_f} L^{(2)}\left(\mathbf{x}^{(2)}, \mathbf{u}^{(2)}, \mathbf{x}_p, t\right) dt \\ &+ \Phi\left(\mathbf{x}^{(2)}(t_f), \mathbf{x}_p, t_f\right). \end{aligned}$$

1. Write the full multiphase CCD decision set, including plant variables, both state trajectories, both control trajectories, t_s, and t_f.
2. Write the dynamic equality constraints for both phases.
3. Write the phase-linkage constraint associated with the jump map.
4. Add the event condition

$$h_{\text{contact}}\left(\mathbf{x}^{(1)}(t_s^-), \mathbf{x}_p\right) = 0.$$

5. Add the time-ordering constraints

$$t_0 < t_s < t_f.$$

6. Suppose the normal contact force in phase 2 is

$$F_n = \psi\left(\mathbf{x}^{(2)}, \mathbf{u}^{(2)}, \mathbf{x}_p\right).$$

   Write the unilateral-contact path constraint.

7. Suppose the impact impulse must satisfy

$$I\left(\mathbf{x}^{(1)}(t_s^-), \mathbf{x}_p\right) \leq I_{\max}.$$

   Classify this as a path or boundary constraint and justify your answer.

8. Convert both phases to fixed normalized-time intervals

$$\tau_1, \tau_2 \in [0, 1].$$

   Derive the scaled dynamics for each phase.

9. Explain why using one continuous dynamic model without the jump map would produce an incorrect formulation.

10. Describe how the formulation would change if the contact sequence itself were an architecture decision rather than fixed in advance.

Worked CCD Formulations

Example 4.1: mass–spring–damper CCD

Choose spring stiffness k and a proportional–derivative controller

$$u(t) = -K_p x(t) - K_d \dot{x}(t)$$

to reduce motion and effort for a system with fixed m and c.

Decisions:

$$\mathbf{x}_p = [k], \qquad \mathbf{x}_c = [K_p, K_d].$$

States and dynamics: With $x_1 = x$ and $x_2 = \dot{x}$,

$$\dot{x}_1 = x_2, \qquad \dot{x}_2 = -\frac{k}{m} x_1 - \frac{c}{m} x_2 + \frac{1}{m} u.$$

Substituting the controller produces a closed-loop model in k, K_p, and K_d.

Objective:

$$J = \int_{t_0}^{t_f} \left(q_x x_1(t)^2 + q_v x_2(t)^2 + r u(t)^2 \right) dt.$$

Bounds and path constraint:

$$0 < k_{\min} \le k \le k_{\max}, \quad 0 \le K_p \le K_{p,\max}, \quad 0 \le K_d \le K_{d,\max}, \quad |u(t)| \le u_{\max}.$$

This is a complete, if small, CCD formulation.

Example 4.2: active suspension

For a quarter-car suspension, let

$$\mathbf{x}_p = [k_s, c_s]^T, \qquad \mathbf{x}_c = [K_1, K_2, K_3, K_4]^T.$$

A possible objective is

$$J = \int_{t_0}^{t_f} \left(w_1 a_b(t)^2 + w_2 z_s(t)^2 + w_3 u(t)^2 \right) dt,$$

where a_b is body acceleration and z_s is suspension deflection. The quarter-car equations enforce the dynamics; path constraints can limit suspension travel, tire load, and actuator force. Plant and controller variables are optimized together while the state trajectory obeys the model.

Example 4.2 revisited: physically concrete plant constraints. The abstract stiffness-and-damping version above is a reasonable first pass, but real suspension co-design studies replace k_s and c_s with the physical geometry that produces them, and replace the generic path constraint $\mathbf{g}(\cdot) \le \mathbf{0}$ with a list of named engineering limits. In one such study, the quarter-car plant vector is the helical-spring and telescopic-damper geometry

$$\mathbf{x}_p = [d,\ D,\ p,\ N_a,\ D_0,\ D_p,\ D_s]^T,$$

where d is spring wire diameter, D is spring coil diameter, p is spring pitch, N_a is the number of active coils, D_0 is the damper valve diameter, D_p is the damper piston diameter, and D_s is the damper stroke; spring stiffness k_s and damping rate c_s become *dependent* quantities computed from this geometry rather than independent design variables. The Lagrange objective weighs road holding, ride comfort, and control cost,

$$J = \int_{t_0}^{t_f} \left(r_1 (z_{us}(t) - z_0(t))^2 + r_2 \ddot{z}_s(t)^2 + r_3 u(t)^2 \right) dt,$$

with weights on the order of $r_1 = 10^5$, $r_2 = 0.5$, and $r_3 = 10^{-5}$ chosen so that all three terms have comparable numerical magnitude—an ordinary but essential scaling step for any multi-term Lagrange objective.

The physical-constraint set $\mathbf{g}_p(\mathbf{x}_p) \leq \mathbf{0}$ for this system is not one generic bound but a list of named engineering limits, including:

- a **buckling constraint** relating free spring length to coil diameter;
- **packaging constraints** requiring the spring and damper to fit within a fixed pocket length (about 0.40 m) and outer diameter, with clearance around the coaxial damper;
- a **rattlespace constraint** bounding peak suspension travel, evaluated from a full vehicle simulation over a ramp road input rather than from a closed-form bound;
- a **spring stress and fatigue constraint** using a Soderberg criterion that combines mean and alternating shear stress computed from the mean and amplitude of the axial spring force under a rough-road simulation;
- a **damper thermal constraint** limiting the maximum damper fluid temperature (about 390 K), predicted by a lumped heat-transfer model of the damper fluid, steel housing, and surrounding air, to prevent fluid fade and seal damage;
- a **damper pressure constraint** limiting maximum damper pressure (about 4.75×10^6 Pa) to protect the seals; and
- a **damper velocity constraint** limiting piston velocity to protect internal valve components.

Several of these—rattlespace, fatigue, and the thermal limit—depend on the full state trajectory under a specified road disturbance, not merely on $\mathbf{x}_p$ in isolation, so they are best classified as path constraints evaluated along a simulated response rather than as simple bounds. The general lesson is that a physically faithful CCD formulation of even a "simple" quarter-car suspension has more than a dozen named engineering constraints, most of them unit-bearing and traceable to a specific failure mode, rather than one abstract $\mathbf{g}(\cdot) \leq \mathbf{0}$.

Example 4.3: classify an objective

Suppose

$$J = -P(t_f) + \int_{t_0}^{t_f} \left(0.02\theta_p(t)^2 + 0.1 M_t(t)^2 \right) dt,$$

where $P(t_f)$ is final delivered energy, θ_p is blade-pitch angle, and M_t is tower-base moment. The terminal term $-P(t_f)$ is Mayer; the integral is Lagrange; together they form a Bolza objective.

From Formulation to Numerical Optimization

Finite-dimensional transcription

A computer cannot optimize arbitrary functions directly. A continuous-time formulation is converted into a nonlinear program by:

- parameterizing state and control trajectories;
- sampling trajectories at finite points;
- enforcing dynamics through simulation or defect equations; and
- approximating integral objectives numerically.

For example, quadrature gives

$$\int_{t_0}^{t_f} L(t)\,dt \approx \sum_{i=1}^{N} w_i L(t_i).$$

Later chapters develop these transcription choices. The important point here is that reliable numerical methods begin with a sound mathematical formulation.

CCD viewpoint

Static design variables are usually numbers. Dynamic CCD variables include both finite-dimensional design parameters and time-dependent trajectories. The formulation must unify engineering design optimization with dynamic-system modeling.

Common formulation mistakes

1. **Leaving out essential physics.** The result may not transfer to the real system.
2. **Confusing variables and parameters.** A quantity is a decision variable only if the optimizer may change it.
3. **Using an objective that misses the engineering goal.** Minimizing control effort alone, for example, may destroy tracking.
4. **Treating path constraints as endpoint constraints.** This permits dangerous intermediate violations.
5. **Over-constraining the problem.** Excessively tight limits may eliminate all feasible designs.
6. **Forgetting variable bounds.** The optimizer may enter unrealistic model regions.
7. **Mixing open- and closed-loop representations.** A feedback-parameterized controller must be reflected consistently in the dynamics and decisions.

Chapter Summary and Problems

Summary

A CCD formulation identifies plant and controller decisions, state and control trajectories, performance objectives, physical equality constraints, time-dependent path constraints, endpoint boundary constraints, and variable bounds. When algebraic equalities accompany the state derivatives, the dynamics form a differential-algebraic equation, and an active inequality path constraint can raise its index by converting a state or control into a dependent algebraic variable. Physical-design limits such as stress, fatigue, and packaging are often collected in a dedicated physical-constraint function, separate from operating limits such as actuator saturation. Lagrange objectives measure accumulated behavior, Mayer objectives measure terminal behavior, and Bolza objectives combine both. The resulting continuous-time problem is infinite-dimensional and must later be transcribed into a finite nonlinear program.

Central lesson

Formulation bridges engineering intuition and numerical computation. It states precisely what can change, what should improve, and what must remain true.

Key terms

Plant design variables; control design variables; control parameters; open-loop control (OLC) variables; state trajectory; control trajectory; equality constraint; path constraint; boundary constraint; physical-constraint function; differential-algebraic equation (DAE); algebraic variable; algebraic constraint; index-1; running cost; terminal cost; Lagrange objective; Mayer objective; Bolza objective; feasible trajectory; bounds; formulation; dynamic optimization.

Problems

1. **Canonical continuous-time CCD formulation.** An active oscillator satisfies $m\ddot{x} + c\dot{x} + kx = u + w$, where (m, k, c) are bounded plant variables and $u(\cdot)$ is the control trajectory. Formulate a complete Bolza CCD problem that minimizes mass, vibration, and control energy while enforcing displacement, velocity, force, terminal-state, and passive-failure constraints.

2. **Nondimensional CCD model.** For the oscillator in Problem 1, use $t_0 = \sqrt{m_0/k_0}$, x_0, and $F_0 = k_0 x_0$ to derive a dimensionless state equation, objective, design bounds, and path constraints, identifying every independent dimensionless group.

3. **Variable-final-time formulation.** A point mass obeys $\dot{r} = v$, $m\dot{v} = u - c_d v|v|$ and must travel from $(0, 0)$ to $(L, 0)$. Transform a free-final-time CCD problem over $[0, t_f]$ to the fixed domain $\tau \in [0, 1]$ and derive the transformed dynamics and objective when m, actuator rating, and t_f are decisions.

4. **Path-constraint tangency.** For $\dot{x}_1 = x_2$ and $\dot{x}_2 = f(x) + bu$, impose the state-only path constraint $x_1(t) \leq x_{\max}$. Derive its relative degree, boundary-control law, and entry tangency conditions required for a nonzero-duration constrained arc.

5. **Index-one DAE co-design.** A circuit model has $E(p)\dot{x} = A(p)x + Bu$ with singular $E(p)$ and algebraic variables embedded in x. Formulate a CCD problem that preserves the DAE rather than eliminating it and state regularity conditions on the matrix pencil $sE - A$ and initial conditions that ensure a unique admissible trajectory.

6. **Hybrid architecture dynamics.** A powertrain switches between modes $q \in \{1, 2\}$ with $\dot{x} = f_q(x, u, p)$ and reset $x^+ = R_{12}(x^-, p)$ at switching time t_s. Formulate the hybrid CCD problem with p, $u(\cdot)$, and t_s as decisions and derive the interior transversality condition at t_s.

7. **Integral versus pointwise requirements.** For actuator temperature $\dot{T} = -a(p)(T - T_a) + b(p)u^2$, prove by

counterexample that an energy constraint $\int_0^{t_f} u^2 dt \leq E_{\max}$ does not generally enforce $T(t) \leq T_{\max}$, then formulate the correct thermal path constraint within a CCD problem.

8. **Uncertain CCD formulation.** The dynamics are $\dot{x} = f(x, u, p, \theta)$ with random, time-invariant parameter $\theta \sim \mathcal{N}(\bar{\theta}, \Sigma)$. Construct a chance-constrained CCD formulation requiring $\mathbb{P}[g(x(t), u(t), p, \theta) \leq 0 \; \forall t] \geq 1 - \epsilon$ and derive a conservative finite-grid deterministic approximation using first-order propagation and risk allocation.

9. **Feedback-policy parameterization.** Replace an open-loop trajectory by $u(t) = \pi(x(t), t; c)$ in a nonlinear CCD problem. Derive the closed-loop state and design sensitivities with respect to (p, c) and show exactly where the controller Jacobian $\partial\pi/\partial x$ enters.

10. **First-order necessary conditions for CCD.** For a general Bolza problem with plant vector p, dynamics $\dot{x} = f(x, u, p)$, equality path constraints $h = 0$, inequality path constraints $g \leq 0$, and endpoint constraint $\Psi = 0$, derive the augmented Hamiltonian system and the stationarity condition with respect to the time-invariant plant variables.

References and further reading

1. Allison, J. T., & Herber, D. R. (2014). Multidisciplinary design optimization of dynamic engineering systems. *AIAA Journal*, *52*(4), 691–710. DOI: 10.2514/1.J052182

2. Allison, J. T., Guo, T., & Han, Z. (2014). Co-design of an active suspension using simultaneous dynamic optimization. *Journal of Mechanical Design*, *136*(8), Article 081003. DOI: 10.1115/1.4027335

3. Herber, D. R., & Allison, J. T. (2019). Nested and simultaneous solution strategies for general combined plant and control design problems. *Journal of Mechanical Design*, *141*(1), Article 011402. DOI: 10.1115/1.4040705

4. Garcia-Sanz, M. (2019). Control co-design: An engineering game changer. *Advanced Control for Applications: Engineering and Industrial Systems*, *1*(1), Article e18. DOI: 10.1002/adc2.18

5. Martins, J. R. R. A., & Ning, A. (2021). *Engineering design optimization*. Cambridge University Press.

6. Bryson, A. E., Jr., & Ho, Y.-C. (1975). *Applied optimal control: Optimization, estimation, and control*. Hemisphere Publishing Corporation.

7. Betts, J. T. (2010). *Practical methods for optimal control and estimation using nonlinear programming* (2nd ed.). Society for Industrial and Applied Mathematics. DOI: 10.1137/1.9780898718577

Control Co-Design Architectures and Coordination

Chapter 5: Control Co-Design Architectures and Coordination

Sequential, iterative, nested, and simultaneous solution strategies

> A CCD architecture is not the engineering problem itself. It is the organization used to solve that problem.

We now know what a control co-design (CCD) problem is and how to write a continuous-time formulation. A new question arises: **How should we organize the solution process?**

A plant and controller may be optimized one after another, alternated repeatedly, embedded in a nested architecture, or solved together in one simultaneous problem. These architectures often target the same engineering goal, but differ in coordination strength, computational burden, numerical difficulty, software integration, and organizational cost.

Learning objectives

After completing this chapter, you should be able to:

1. describe the information flow in the main CCD architectures;
2. distinguish single-pass from iterative sequential design;
3. explain nested and simultaneous CCD;
4. distinguish mathematical equivalence from practical numerical behavior;
5. evaluate computational and implementation tradeoffs; and
6. select an architecture for a given engineering application.

Chapter roadmap

Single-pass sequential → Iterative sequential → Nested CCD → Simultaneous CCD → Tradeoffs and selection

Architecture Landscape and Single-Pass Design

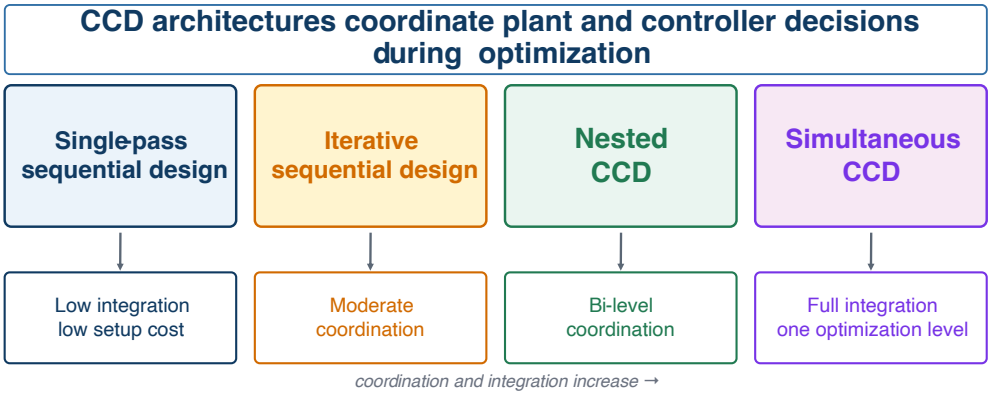
Why architecture matters

A generic CCD problem may be written as

$$\underset{\mathbf{x}_p,\mathbf{x}_c}{\text{minimize}} \quad J(\mathbf{x}_p,\mathbf{x}_c) \qquad \text{subject to dynamic and engineering constraints.}$$

This statement does not say how to organize the optimization. An architecture answers:

- Which subproblem is solved first?
- Which variables are fixed while others change?
- Does plant optimization call controller optimization as an inner loop?
- Are all decisions handled at one level or several?
- How often is information exchanged?



The main CCD architectures differ in coordination strength and implementation cost.

Engineering teams may already have separate plant and controller tools, or may be limited by simulation cost, derivative availability, and software structure. If plant–control coupling is weak, a simple method may be enough. Strong coupling makes weak coordination more likely to miss good designs.

Main idea

Different architectures may address the same underlying CCD formulation while coordinating its variables in very different ways.

CCD architectures within the broader MDO taxonomy

The four architectures introduced in this chapter are not invented in isolation. Each has a direct analogue in the general multidisciplinary design optimization (MDO) literature, which classifies formulations by how they handle the coupling between disciplinary analyses.

- **Multidisciplinary feasible (MDF)** formulations perform all analysis inside the optimization loop, so every point the optimizer visits is consistent with the system's governing equations before optimality is reached. Nested CCD is a special case of MDF: the plant design is proposed by the (outer) optimizer, the associated response is obtained for that candidate plant — here, through an inner optimal-control solve rather than a single simulation pass — and the outer optimizer sees only a single reduced objective function value in return.

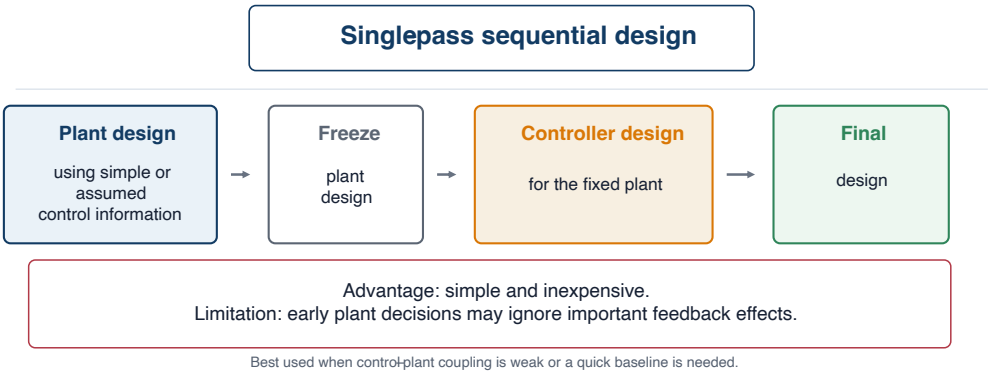
- **Individual disciplinary feasible (IDF)** formulations instead promote the variables that couple subsystems to the status of independent optimization variables, adding equality constraints that only need to hold at convergence rather than at every iterate. Applied to a dynamic system, this can mean treating a subset of state variables, or entire trajectory segments as in multiple shooting, as optimization variables in their own right. IDF sacrifices guaranteed feasibility during the search in exchange for decoupling the underlying analyses, which is what enables coarse-grained parallel computation across subsystems.
- **Distributed, multilevel formulations** go further and distribute the optimization itself, not merely the analysis, across subproblems. Augmented Lagrangian coordination (ALC) is one such formulation, described in the MDO literature as a nonhierarchical generalization of analytical target cascading (ATC), an earlier multilevel coordination strategy. In ALC, each subsystem — for example, each time segment of a discretized trajectory — solves its own local optimization problem using a local copy of the shared coupling variables, and a coordination algorithm drives the local copies toward agreement through an augmented Lagrangian penalty rather than a hard equality constraint.

Where this chapter's architectures sit

Single-pass and iterative sequential design are not standard MDO architectures at all — they solve reduced problems that need not honor the full system-level objective, which is exactly the weak-coordination property criticized earlier in this course. Nested CCD is essentially MDF applied to a plant-versus-control partition of the design variables. Simultaneous CCD, once transcribed into a nonlinear program, has an MDF-like character too: the discretized dynamics appear as ordinary equality constraints handled by the same optimizer that resolves the design variables, rather than by a separate inner solve. IDF-style and ALC-style decompositions are the architectures to reach for when a single monolithic dynamic model cannot be assembled or solved directly — a case the four architectures developed in this chapter do not, by themselves, address.

Single-pass sequential design

The plant is designed first using a nominal controller or simplified control assumption. The chosen plant is then fixed while the controller is optimized.



Single-pass sequential design makes one pass through the two stages.

A simple mathematical representation is

$$\mathbf{x}_p^* = \arg\min_{\mathbf{x}_p} J_p(\mathbf{x}_p; \hat{\mathbf{x}}_c), \qquad \mathbf{x}_c^* = \arg\min_{\mathbf{x}_c} J_c(\mathbf{x}_p^*, \mathbf{x}_c),$$

where $\hat{\mathbf{x}}_c$ is an assumed controller.

Advantages.

- Easy to explain and implement.
- Reuses existing disciplinary workflows.
- Requires little optimization infrastructure.
- Can work when plant–control coupling is weak.

Limitations. Early physical decisions ignore how an improved controller might alter plant tradeoffs. A passive structure may be oversized because feedback load reduction was not considered. The method is simple, but coordination is weak.

Example. If suspension stiffness and damping are selected using passive metrics and an active controller is added afterward, the process may miss a softer plant that delivers better comfort under feedback while still meeting handling constraints.

Activity 5.1: Exact Comparison of Four CCD Architectures

Activity 5.1: Exact Comparison of Four CCD Architectures

Consider the unconstrained control co-design problem

$$\min_{p,c} \; J(p,c) = \frac{1}{2}\left(6p^2 + 6pc + 4c^2\right) - 8p - 6c,$$

where p is a plant-design variable and c is a controller-design variable.

1. Verify that J is strictly convex.
2. In a single-pass sequential design, first optimize p using the nominal controller $\hat{c} = 0$, and then optimize c for the fixed value of p. Compute

$$p_{\text{seq}}, \qquad c_{\text{seq}}, \qquad J_{\text{seq}}.$$

3. Formulate the nested problem

$$\phi(p) = \min_{c} J(p,c), \qquad \min_{p} \phi(p).$$

   Derive $c^*(p)$ and the reduced objective $\phi(p)$.
4. Solve the nested problem analytically.
5. Solve the simultaneous problem by imposing

$$\nabla J(p,c) = \mathbf{0}.$$

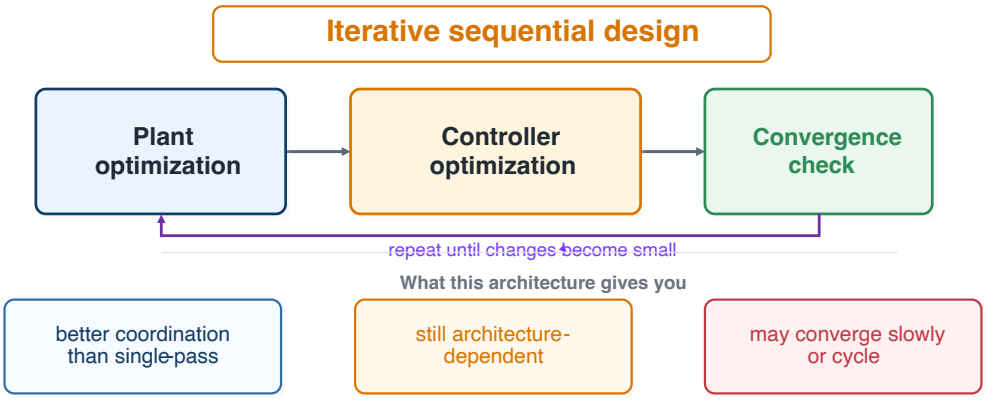
6. Show that the nested and simultaneous solutions are identical.
7. Compute the relative performance loss of the single-pass sequential design:

$$\eta_{\text{seq}} = \frac{J_{\text{seq}} - J_{\text{sim}}}{|J_{\text{sim}}|}.$$

8. Explain why strict convexity is important for the equivalence observed in this problem.

Iterative Sequential Design

Iterative sequential design alternates plant and controller optimization until changes become small.



Repeated information exchange provides more coordination than a single pass.

An abstract update is

$$\mathbf{x}_p^{(k+1)} = \arg\min_{\mathbf{x}_p} J(\mathbf{x}_p, \mathbf{x}_c^{(k)}), \qquad \mathbf{x}_c^{(k+1)} = \arg\min_{\mathbf{x}_c} J(\mathbf{x}_p^{(k+1)}, \mathbf{x}_c).$$

The plant step uses the current controller, and the controller step uses the updated plant.

Advantages

- Can substantially improve on a single-pass design.
- Preserves separate plant and controller subproblems.
- Reuses legacy models and tools.
- Is easier to implement than full simultaneous CCD.

Limitations

- Convergence is not guaranteed.
- Results can depend on update order.
- The process may cycle or converge slowly.
- The final point can differ from a fully coordinated optimum.

In strongly coupled problems, each improvement in one discipline changes the best response of the other, making repeated alternation expensive.

Example

A wind-turbine team may alternate structural redesign and pitch-controller retuning. Performance can improve gradually, but many costly simulation-based iterations may be required.

Activity 5.2: Convergence of Iterative Sequential Design

Activity 5.2: Convergence of Iterative Sequential Design

Consider the quadratic CCD problem

$$J(\mathbf{p},\mathbf{c}) = \frac{1}{2}\mathbf{p}^T A\mathbf{p} + \mathbf{p}^T B\mathbf{c} + \frac{1}{2}\mathbf{c}^T C\mathbf{c} - \mathbf{a}^T\mathbf{p} - \mathbf{b}^T\mathbf{c},$$

where A and C are symmetric positive-definite matrices. An iterative sequential method performs

$$\mathbf{p}^{(k+1)} = \arg\min_{\mathbf{p}} J\left(\mathbf{p}, \mathbf{c}^{(k)}\right),$$
$$\mathbf{c}^{(k+1)} = \arg\min_{\mathbf{c}} J\left(\mathbf{p}^{(k+1)}, \mathbf{c}\right).$$

1. Derive the explicit update equations

$$\mathbf{p}^{(k+1)} = A^{-1}\left(\mathbf{a} - B\mathbf{c}^{(k)}\right),$$
$$\mathbf{c}^{(k+1)} = C^{-1}\left(\mathbf{b} - B^T\mathbf{p}^{(k+1)}\right).$$

2. Let $\mathbf{c}^*$ denote the simultaneous optimum. Show that the controller error satisfies

$$\mathbf{c}^{(k+1)} - \mathbf{c}^* = M\left(\mathbf{c}^{(k)} - \mathbf{c}^*\right),$$

   where

$$M = C^{-1}B^T A^{-1}B.$$

3. Prove that the iterative sequential method converges for every initial controller if

$$\rho(M) < 1,$$

   where $\rho(\cdot)$ denotes spectral radius.

4. Use

$$A = \begin{bmatrix} 4 & 1 \\ 1 & 3 \end{bmatrix}, \qquad C = \begin{bmatrix} 3 & 0.5 \\ 0.5 & 2 \end{bmatrix},$$

   and

$$B = \begin{bmatrix} 1.8 & 0.4 \\ 0.2 & 1.2 \end{bmatrix}, \qquad \mathbf{a} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}, \qquad \mathbf{b} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}.$$

   Compute $\rho(M)$ and determine whether the iterative sequential method converges.

5. Starting from

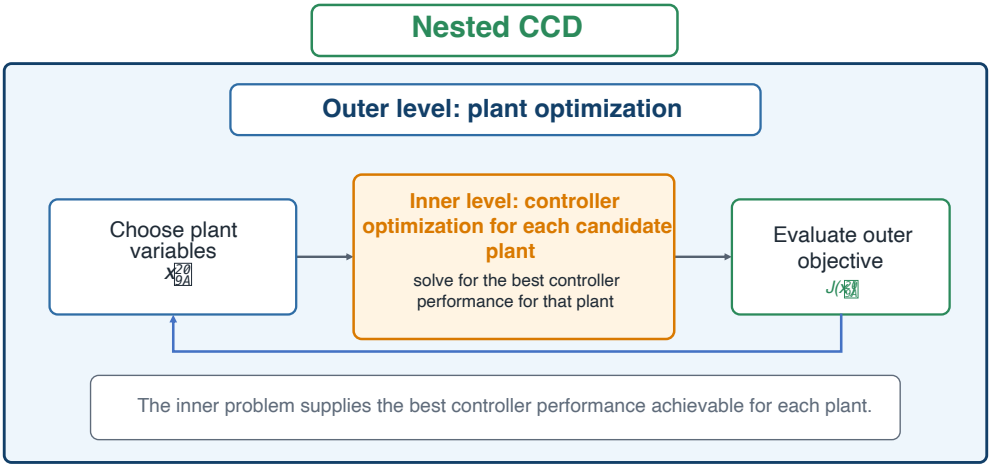
$$\mathbf{c}^{(0)} = \begin{bmatrix} 0 \\ 0 \end{bmatrix},$$

perform ten alternating iterations and compare the result with the simultaneous optimum.

6. Explain how the strength of plant–controller coupling is reflected in $\rho(M)$.

Nested Control Co-Design

Nested CCD uses a **bi-level** structure. The outer optimization chooses plant variables; for each plant candidate, an inner optimization finds its best controller.



Each plant is evaluated through an inner controller-optimization problem.

An idealized formulation is

$$\underset{\mathbf{x}_p}{\text{minimize}} \quad \phi(\mathbf{x}_p), \qquad \phi(\mathbf{x}_p) = \underset{\mathbf{x}_c}{\text{minimize}} \quad J(\mathbf{x}_p, \mathbf{x}_c),$$

subject to the relevant plant, controller, dynamic, and engineering constraints.

The outer loop does not directly choose $\mathbf{x}_c$; it sees the best achievable performance returned by the inner controller solve. This evaluates each plant fairly under the chosen controller structure.

Advantages

- Stronger coordination than sequential approaches.
- Natural separation of plant and controller roles.
- Clear interpretation of plant quality under best achievable control.
- Compatibility with established controller-tuning tools.

Limitations

Every outer evaluation may require a complete inner optimization. Additional challenges include:

- noise or discontinuity when inner solves are incomplete;
- difficulty differentiating through the inner optimum;
- sensitivity to inner-solver reliability; and
- potentially very high total simulation cost.

Example

For active suspension, the outer loop varies stiffness and damping. For each candidate, the inner loop computes optimal feedback gains. The outer loop compares plants using the optimized closed-loop performance returned by the controller solver.

A precise bi-level formulation

The idealized statement above hides an important detail: the inner minimization only makes sense if a feasible controller exists for the candidate plant. Write the inner-loop feasible set for a given plant candidate $\mathbf{x}_p^\dagger$ as $\Lambda(\mathbf{x}_p^\dagger)$, the set of controller variables $\mathbf{x}_c$ satisfying the dynamic, path, and boundary constraints associated with that plant. The nested strategy's *induced region* is then the set of pairs $(\mathbf{x}_p, \mathbf{x}_c)$ such that $\mathbf{x}_p$ satisfies its own constraints and $\mathbf{x}_c$ minimizes $J(\mathbf{x}_p, \cdot)$ over $\Lambda(\mathbf{x}_p)$.

This induced region is generally a strict subset of the simultaneous problem's feasible set: a pair $(\mathbf{x}_p, \mathbf{x}_c)$ may satisfy every constraint yet never be produced by the nested strategy if $\mathbf{x}_c$ is feasible but not the *optimal* response to $\mathbf{x}_p$. For the nested and simultaneous strategies to be equivalent, $\Lambda(\mathbf{x}_p')$ must be nonempty for every plant design $\mathbf{x}_p'$ the outer loop is allowed to propose — otherwise the inner optimization has nothing to return, and the reduced objective $\phi(\mathbf{x}_p')$ is undefined. This "outer-loop feasibility" property is not automatic and, for general co-design problems, is not guaranteed even by classical results such as linear-system controllability once realistic path constraints are present. Practical nested implementations often add an explicit outer-loop feasibility constraint on $\mathbf{x}_p$ so that only plant designs with a workable inner problem are ever offered to the optimizer.

Why this matters in practice

An infeasible inner controller optimization is not a corner case. In a documented active-suspension co-design study, uniformly sampling plant designs within their simple box bounds produced an infeasible inner-loop optimal-control problem 44% of the time; requiring feasibility with respect to the full constraint set reduced this to about 0.033%, but a nested implementation that does not screen for this can still fail outright on an otherwise reasonable starting point. Equivalence and Computational Tradeoffs returns to this issue with the full comparison it comes from.

Optimality conditions for the nested strategy

The inner controller problem, for a fixed candidate plant, is an ordinary optimal control problem. Adjoining the dynamics and any path constraints with costates $\boldsymbol{\lambda}(t)$ and multipliers $\boldsymbol{\mu}(t)$ forms the Hamiltonian $H = \mathcal{L} + \boldsymbol{\lambda}^T f + \boldsymbol{\mu}^T C$, and Pontryagin's minimum principle supplies the usual costate dynamics, control stationarity, complementary slackness, and transversality conditions at t_0 and t_f — all evaluated at the candidate plant. These conditions are exactly the necessary conditions the inner loop must satisfy.

The outer-loop problem is finite-dimensional in $\mathbf{x}_p$ alone, so its necessary conditions follow from ordinary KKT theory once the *total* derivative of the reduced objective $\phi(\mathbf{x}_p) = J(\mathbf{x}_p, \mathbf{x}_c^*(\mathbf{x}_p))$ is available:

$$\frac{d\phi}{d\mathbf{x}_p} = \frac{\partial J}{\partial \mathbf{x}_p} + \frac{\partial J}{\partial \mathbf{x}_c}\frac{d\mathbf{x}_c^*}{d\mathbf{x}_p}.$$

If the inner problem's necessary conditions hold exactly, the second term can be shown to vanish by the envelope theorem — Activity 5.4 works through this cancellation explicitly for a quadratic inner problem with an equality-constrained inner solution. In that idealized case, the outer-loop gradient reduces to $\partial J/\partial \mathbf{x}_p$ evaluated along the optimal inner trajectory, and the bookkeeping for $d\mathbf{x}_c^*/d\mathbf{x}_p$ can, in principle, be skipped. In practice this shortcut is dangerous: a numerical inner solver only ever converges to a nonzero optimality or feasibility residual, so the envelope-theorem cancellation is only approximate, and dropping the correction term can silently corrupt the outer-loop gradient. This is the numerical effect Activity 5.4 asks you to quantify directly by sweeping the inner solver's tolerance.

When the inner problem is a linear-quadratic dynamic optimization problem — quadratic objective, linear time-invariant dynamics, no path constraints, infinite time horizon — the inner loop collapses to a single algebraic Riccati equation, and the optimal feedback gain has the closed form

$$\mathbf{u}^* = -K^*\xi, \qquad K^* = R^{-1}B^T P^*,$$

where P^* is the unique positive-definite solution of $P^*A + A^T P^* - P^*BR^{-1}B^T P^* + Q = 0$. This is precisely the structure exploited in Activity 5.3, and it is one of two special inner-loop forms — the other being a general finite-horizon linear-quadratic dynamic optimization problem, solvable after discretization as a convex quadratic program — that make nested CCD attractive in practice: the inner loop can be solved to high accuracy and speed by a tailored algorithm rather than a general-purpose nonlinear optimal control solver.

Activity 5.3: Nested LQR Control Co-Design

Activity 5.3: Nested LQR Control Co-Design

Consider the scalar plant

$$\dot{x} = -px + u, \qquad x(0) = 1,$$

where

$$0.1 \le p \le 3$$

is a plant-design variable. Use the feedback controller

$$u = -Kx, \qquad K \ge 0.$$

The infinite-horizon system-level objective is

$$J(p, K) = \int_0^\infty \left(x(t)^2 + r\,u(t)^2\right) dt + \alpha p^2,$$

with

$$r = 0.2, \qquad \alpha = 0.08.$$

1. Show that the closed-loop state is

$$x(t) = e^{-(p+K)t}.$$

2. Derive the finite-dimensional objective

$$J(p, K) = \frac{1 + rK^2}{2(p + K)} + \alpha p^2.$$

3. For fixed p, solve the inner controller problem and show that

$$K^*(p) = -p + \sqrt{p^2 + \frac{1}{r}}.$$

4. Derive the nested value function

$$\phi(p) = J\left(p, K^*(p)\right).$$

5. Independently derive the same inner solution using the scalar algebraic Riccati equation.
6. Minimize $\phi(p)$ over the admissible interval and compute

$$p^*_{\text{nested}}, \qquad K^*_{\text{nested}}.$$

7. Solve the simultaneous problem

$$\min_{p,K} J(p, K)$$

directly and verify that it produces the same optimum.

8. Perform a single-pass sequential design by first setting $K = 0$ and optimizing p, and then optimizing K for the fixed plant. Compare the sequential and coordinated solutions.

Activity 5.4: Differentiating a Nested Value Function

Activity 5.4: Differentiating a Nested Value Function

Consider the nested CCD problem

$$\min_{\mathbf{p}} \ \phi(\mathbf{p}) = \frac{1}{2}\mathbf{p}^T Q\mathbf{p} + \min_{\mathbf{c}} \left[\frac{1}{2}\mathbf{c}^T H\mathbf{c} + \mathbf{p}^T D\mathbf{c}\right],$$

subject to the inner equality constraint

$$A\mathbf{c} = \mathbf{b} + E\mathbf{p},$$

where H is symmetric positive definite and A has full row rank.

1. Write the KKT system for the inner problem:

$$\begin{bmatrix} H & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} \mathbf{c}^* \\ \boldsymbol{\lambda}^* \end{bmatrix} = \begin{bmatrix} -D^T\mathbf{p} \\ \mathbf{b} + E\mathbf{p} \end{bmatrix}.$$

2. Differentiate the KKT system with respect to $\mathbf{p}$ and derive

$$\frac{d\mathbf{c}^*}{d\mathbf{p}}, \qquad \frac{d\boldsymbol{\lambda}^*}{d\mathbf{p}}.$$

3. Derive the total derivative of the nested objective,

$$\frac{d\phi}{d\mathbf{p}}.$$

4. Show how the envelope theorem eliminates the explicit $d\mathbf{c}^*/d\mathbf{p}$ term when the inner KKT conditions are satisfied exactly.
5. Explain why this simplification may fail numerically when the inner problem is terminated with a large optimality or feasibility residual.
6. Implement the nested gradient for a randomly generated positive-definite H and verify it using complex-step or central finite differences.
7. Investigate how the outer-gradient error changes when the inner solver tolerance is varied from 10^{-3} to 10^{-10}.

Activity 5.5: KKT Reformulation of a Constrained Bilevel CCD Problem

Activity 5.5: KKT Reformulation of a Constrained Bilevel CCD Problem

Consider the nested problem

$$\min_{0\le p\le 3} \ \phi(p) = (p-2)^2 + \min_{0\le c\le 1} \left[(c-p)^2 + 0.1c^2\right].$$

1. Solve the unconstrained inner problem and obtain

$$c_{\mathrm{uc}}(p).$$

2. Derive the exact piecewise inner solution $c^*(p)$.
3. Derive the piecewise value function $\phi(p)$ and identify every point at which its derivative changes expression.
4. Solve the nested problem analytically.
5. Introduce Lagrange multipliers μ_L and μ_U for

$$-c \le 0, \qquad c - 1 \le 0,$$

and write the inner KKT conditions:

$$\begin{aligned} 2(c-p) + 0.2c - \mu_L + \mu_U &= 0, \\ \mu_L c &= 0, \\ \mu_U (c-1) &= 0, \\ \mu_L, \mu_U &\ge 0. \end{aligned}$$

6. Reformulate the bilevel problem as a single-level mathematical program with complementarity constraints.
7. Enumerate the possible active sets and recover the nested optimum from the KKT reformulation.
8. Solve the direct simultaneous problem

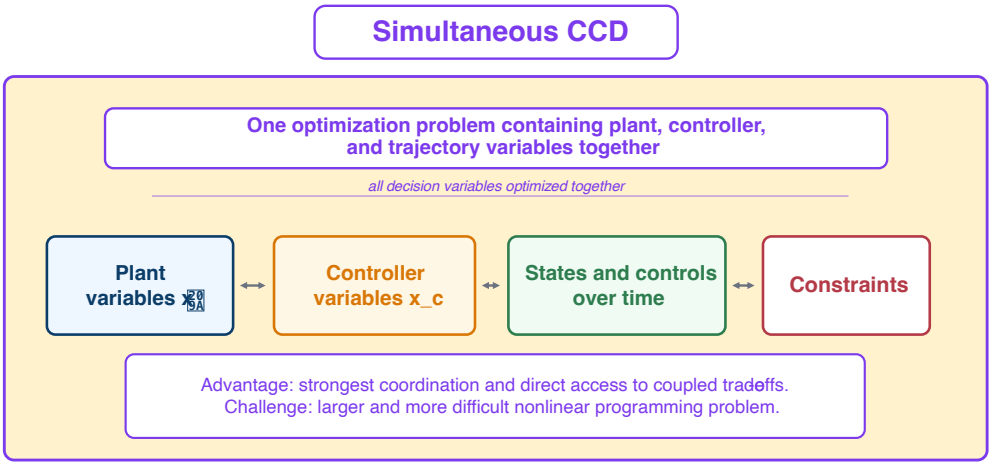
$$\min_{0\le p\le 3,\ 0\le c\le 1} (p-2)^2 + (c-p)^2 + 0.1c^2$$

and compare its solution with the nested result.

9. Explain why complementarity constraints create numerical difficulties for conventional nonlinear programming solvers.

Simultaneous Control Co-Design

Simultaneous CCD optimizes plant, controller, and often trajectory variables in one integrated problem.



All coupled decisions are optimized together.

A generic continuous-time statement is

$$\underset{\mathbf{x}_p,\mathbf{x}_c,\mathbf{x}(\cdot),\mathbf{u}(\cdot)}{\text{minimize}} \quad J(\mathbf{x}_p,\mathbf{x}_c,\mathbf{x},\mathbf{u}) \qquad \text{subject to all model and engineering constraints.}$$

After transcription, this becomes one large nonlinear program. The optimizer can change physical and control decisions together while accounting for the resulting trajectories and constraints.

Advantages

- Strongest coordination.
- Direct access to plant–control tradeoffs.
- Unified mathematical structure.
- Natural use of sparse gradient-based optimization.

Limitations

The integrated nonlinear program can be large and difficult. Successful implementation may require:

- careful scaling;
- strong initial guesses;
- reliable derivatives;
- specialized nonlinear-programming tools; and
- sophisticated software integration.

Simultaneous CCD offers a direct route to a coordinated optimum, especially under strong coupling, but it is not automatically the easiest or most reliable architecture.

Optimality conditions

Because the simultaneous problem folds the plant variables into the same optimization as the state trajectory, its necessary conditions can be derived directly with Pontryagin's minimum principle by treating the plant variables as an additional, time-invariant part of an augmented state:

$$\boldsymbol{\Theta} = \begin{bmatrix} \boldsymbol{\xi} \\ \mathbf{x}_p \end{bmatrix}, \qquad \dot{\boldsymbol{\Theta}} = \begin{bmatrix} f \\ \mathbf{0} \end{bmatrix}.$$

Treating $\mathbf{x}_p$ as a free initial condition of this augmented state folds the plant-design decision into the same costate machinery used for the true states, at the cost of an additional transversality-type stationarity condition that couples the plant costate to the partial derivatives of the terminal cost and boundary constraints with respect to $\mathbf{x}_p$, integrated together with the running-cost and dynamic sensitivities over the whole horizon. The essential result is that the simultaneous strategy's necessary conditions are the ordinary costate dynamics, control stationarity, and complementary-slackness conditions for the states and controls, augmented with one additional stationarity condition on $\mathbf{x}_p$ that cannot be separated from the necessary conditions on the rest of the trajectory. This is the formal sense in which simultaneous CCD has the strongest coordination: in the nested strategy, by contrast, the outer-loop condition on $\mathbf{x}_p$ only sees the trajectory indirectly, through the already-optimized reduced objective $\phi(\mathbf{x}_p)$.

From infinite-dimensional problem to finite nonlinear program

Simultaneous CCD is almost always solved after **direct transcription (DT)**: the state and control trajectories are approximated at a finite set of mesh points, and the dynamic constraint is replaced by a large but sparse set of algebraic *defect* equations enforced at those points. The plant variables $\mathbf{x}_p$ enter this nonlinear program as ordinary optimization variables alongside the discretized states and controls, which is what makes the "unified mathematical structure" and "natural use of sparse gradient-based optimization" advantages listed above concrete — a sparse NLP solver can exploit the resulting Jacobian sparsity pattern directly, without any special-purpose inner solve.

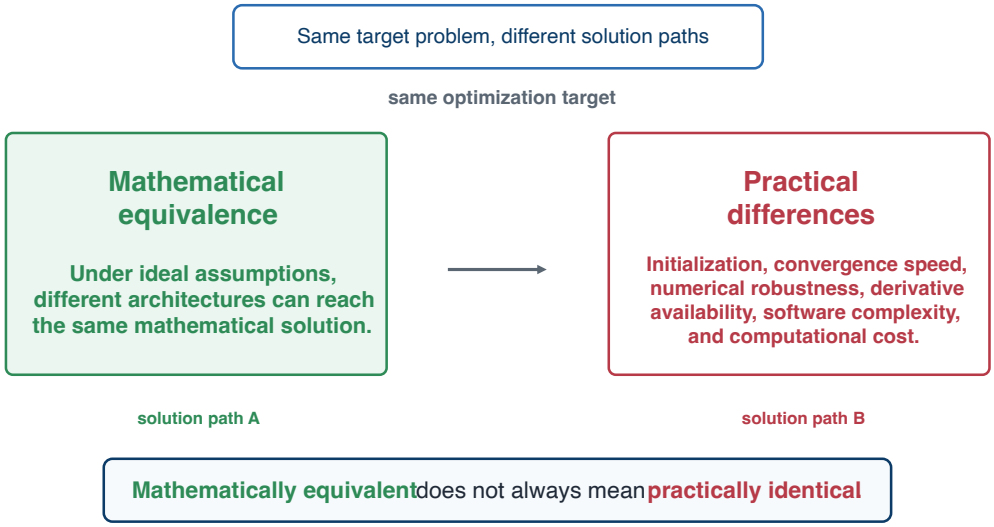
A structural limitation

Not every co-design problem can be posed as an efficient simultaneous quadratic-program formulation. If a plant variable multiplies the state inside the dynamics — a *bilinear* dynamic constraint such as $\dot{\xi} = -k\xi$ with k a plant-design variable — fixing k keeps the dynamics linear, but once k is a joint optimization variable the corresponding defect constraints become quadratic rather than linear, so the convex quadratic-program structure that makes linear-quadratic inner loops so efficient no longer applies directly to the simultaneous formulation. The nested strategy sidesteps this cleanly: its inner loop only ever sees one fixed value of k at a time, so the inner problem stays exactly linear-quadratic even though the plant variable multiplies the state.

Equivalence and Computational Tradeoffs

Mathematical equivalence

Nested and simultaneous CCD may target the same underlying problem. If the nested inner controller optimization is solved exactly for every plant and the outer optimization finds the global optimum, it should identify the same pair $(\mathbf{x}_p^*, \mathbf{x}_c^*)$ as a perfect simultaneous method.



Architectures may be mathematically equivalent under ideal assumptions while behaving differently in actual computation.

Why practical behavior differs

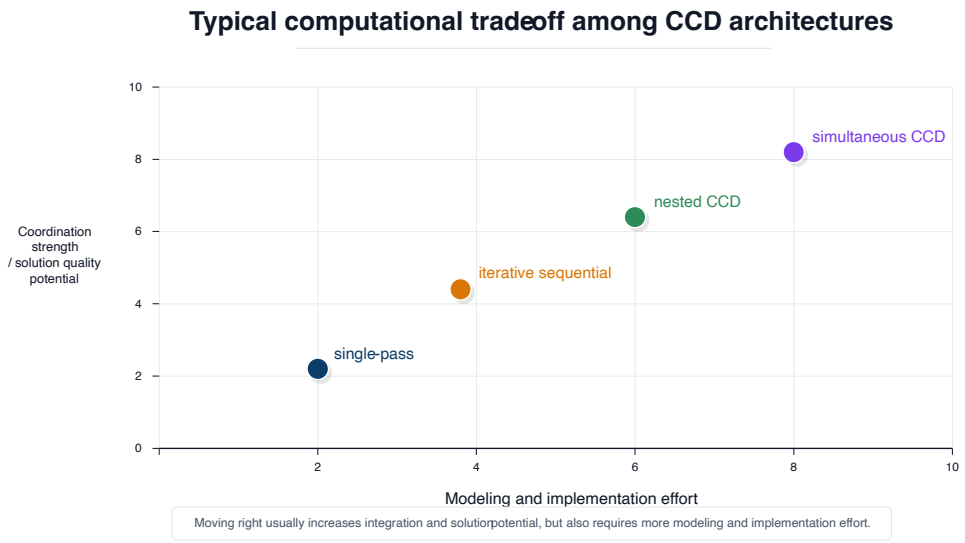
Real optimization is not exact:

- **Local minima:** architectures may converge to different basins.
- **Initialization:** sensitivity differs across formulations.
- **Approximate inner solves:** nested objectives may be evaluated inconsistently.
- **Derivative quality:** simultaneous methods often depend heavily on accurate Jacobians.
- **Scaling:** a large unified nonlinear program can be difficult to balance.
- **Stopping criteria:** different tolerances produce different practical endpoints.

Important distinction

Two architectures may be equivalent as statements of an ideal solution yet very different as computational procedures. For engineering work, the practical difference often matters more than formal equivalence.

Coordination versus complexity



Greater coordination often comes with greater implementation and computational complexity.

The typical pattern is:

- **Single-pass sequential:** lowest setup cost and weakest coordination.
- **Iterative sequential:** moderate effort and coordination.
- **Nested CCD:** strong coordination with potentially expensive repeated solves.
- **Simultaneous CCD:** strongest coordination and often the most demanding integrated formulation.

Simulation cost. Long simulations make repeated nested outer-loop evaluations expensive. A simultaneous solve may avoid repeated full re-optimization but introduces a much larger single problem.

Derivative availability. Reliable gradients can make simultaneous methods highly effective. When derivatives are unavailable, sequential or nested strategies—possibly with derivative-free outer loops—may be easier.

Software and organizational cost. A team with separate plant and controls groups may adopt iterative or nested approaches more readily than a unified simultaneous framework.

Robustness versus optimality. A simpler architecture may be chosen because it is understandable, robust, and compatible with existing tools. Maximum performance may justify the added effort of simultaneous CCD.

Evidence from a controlled comparison

Much of the folklore about which architecture is "faster" compares implementations that were not built with comparable care — a fast simultaneous run with hand-coded analytic derivatives against a nested run with a crude finite-difference outer loop, or the reverse. A more recent controlled study of an active-suspension CCD problem — a quarter-car model with two masses, four states, a controllable force actuator, and physically parameterized spring and damper design variables, solved by direct transcription — deliberately varied only the implementation choices (derivative method, mesh density, and solver tolerances) while holding the problem formulation and solver family fixed, to separate genuine architectural differences from implementation quality.

Two results stand out.

Derivative method dominates the comparison. With enough mesh points ($N_t \geq 600$) and symbolic derivatives, simultaneous CCD was the fastest and most accurate strategy tested. But symbolic derivatives are a best case that many real dynamic models cannot supply. The complex-step method — nearly as accurate as symbolic differentiation — was on average about $10\times$ slower, and ordinary real-valued (central or forward) finite differences were a further 5–$10\times$ slower than complex-step. Once the comparison is restricted to derivative methods that are actually available for a general nonlinear plant model, the ranking can flip: the nested strategy becomes competitive with, and can be faster than, simultaneous CCD by an order of magnitude or more.

Inner-loop infeasibility is common, not exotic. Uniformly sampling candidate plant designs within their simple box bounds produced an infeasible inner-loop optimal control problem 44% of the time; requiring feasibility with respect to the full constraint set (not just the box bounds) reduced this to about 0.033%, but a naive nested implementation that does not anticipate this can fail outright — in this study, a previously published starting point turned out to be exactly one of the infeasible candidates. A hybrid scheme — one generation of a genetic algorithm to locate a plant design with a feasible inner problem, followed by a gradient-based optimizer — was one practical way to avoid this failure mode.

Reading the folklore carefully

"Nested is better" and "simultaneous is better" are both defensible conclusions from real published comparisons, because those comparisons typically differ in implementation quality, not just in architecture. A fair, implementation-controlled comparison found that the ranking flips depending on whether accurate derivatives are available — an implementation and modeling detail, not a property of the architecture itself.

This connects directly to the outer-loop feasibility constraint introduced for nested CCD: adding such a constraint is exactly the kind of implementation safeguard that keeps the nested strategy from failing on the large fraction of naively sampled plant candidates that would otherwise have no valid inner solution.

Activity 5.6: Computational Benchmark of CCD Architectures

Activity 5.6: Computational Benchmark of CCD Architectures

Consider the controlled oscillator

$$\dot{\mathbf{x}} = \begin{bmatrix} 0 & 1 \\ -k & -c \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u,$$

with feedback

$$u = -K_p x_1 - K_d x_2.$$

The design bounds are

$$0.2 \leq k \leq 5, \qquad 0.05 \leq c \leq 3,$$

and

$$0 \leq K_p \leq 10, \qquad 0 \leq K_d \leq 8.$$

Define

$$A_{\mathrm{cl}} = \begin{bmatrix} 0 & 1 \\ -(k + K_p) & -(c + K_d) \end{bmatrix},$$

and

$$Q = \begin{bmatrix} 10 & 0 \\ 0 & 1 \end{bmatrix}, \qquad R = 0.05.$$

For every stable design, let P satisfy

$$A_{\mathrm{cl}}^T P + P A_{\mathrm{cl}} + Q + \begin{bmatrix} K_p \\ K_d \end{bmatrix} R \begin{bmatrix} K_p & K_d \end{bmatrix} = 0.$$

Use

$$\Sigma_0 = \begin{bmatrix} 1 & 0 \\ 0 & 0.25 \end{bmatrix},$$

and define

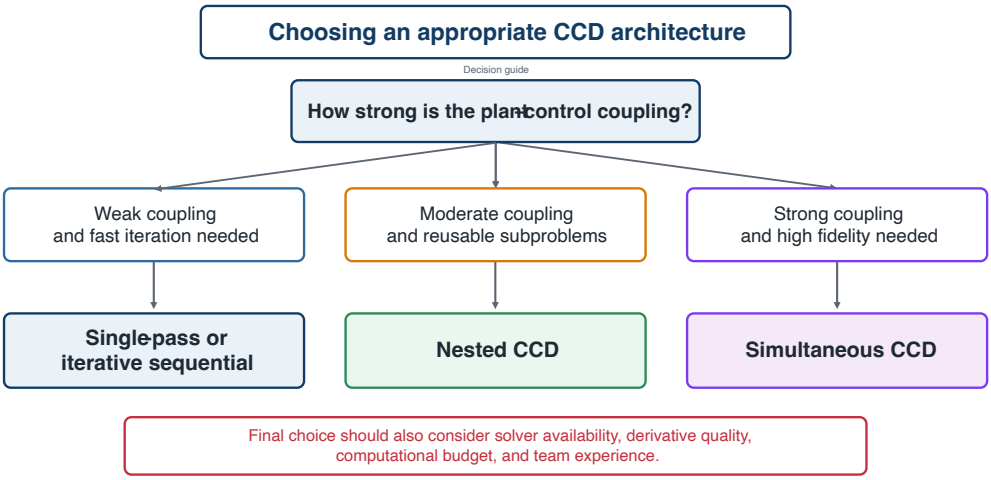
$$J(k, c, K_p, K_d) = \mathrm{tr}(P\Sigma_0) + 0.02k^2 + 0.02c^2 + 0.005K_p^2 + 0.005K_d^2.$$

1. Derive the closed-loop stability conditions.
2. Implement a single-pass sequential architecture:
   (a) optimize k and c with $K_p = K_d = 0$; and
   (b) freeze the plant and optimize K_p and K_d.
3. Implement iterative sequential design by alternating exact or numerically converged plant and controller subproblems until
$$\left\| \mathbf{z}^{(j+1)} - \mathbf{z}^{(j)} \right\|_2 < 10^{-6}.$$
4. Implement nested CCD:
$$\min_{k,c} \left[\min_{K_p, K_d} J(k, c, K_p, K_d) \right].$$
5. Implement simultaneous CCD:
$$\min_{k,c,K_p,K_d} J(k, c, K_p, K_d).$$
6. Use at least ten initial guesses for the nested and simultaneous problems. MATLAB `fmincon`, IPOPT, SNOPT, or an equivalent constrained optimizer may be used.
7. Compare
$$J^*, \qquad k^*, \qquad c^*, \qquad K_p^*, \qquad K_d^*,$$
together with function evaluations, derivative evaluations, CPU time, and sensitivity to initialization.

8. Repeat the nested study with inner optimality tolerances 10^{-3}, 10^{-6}, and 10^{-9}. Quantify the effect of imperfect inner solves on the outer optimum.
9. Determine whether the best nested and simultaneous solutions agree to within 10^{-5} in objective value.
10. Explain any disagreement in terms of local minima, inner-loop accuracy, derivative quality, scaling, or solver termination criteria.

Choosing an Appropriate Architecture

No architecture is always best. Selection depends on coupling strength, simulation cost, derivatives, model structure, available software, and the goal of the study.



Plant–control coupling is central to architecture choice, but it is not the only consideration.

Favor single-pass sequential design when

- coupling is weak;
- a quick baseline is needed;
- ease of implementation dominates; and
- the controller has modest influence on plant tradeoffs.

Favor iterative sequential design when

- some coupling exists but full integration is impractical;
- separate plant and controller tools must remain;
- repeated coordination is affordable; and
- moderate improvement over a baseline is desired.

Favor nested CCD when

- controller optimization is reliable and well defined;
- every plant should be evaluated under best achievable control;
- the structure naturally suggests outer plant and inner control loops; and
- controller-tool reuse is important.

Favor simultaneous CCD when

- plant–control coupling is strong;
- maximum coordinated performance is sought;
- accurate gradients and modern nonlinear-programming tools are available; and
- unified model integration is feasible.

A sharper criterion: what derivatives can you actually get?

A controlled comparison on an active-suspension co-design problem suggests a more specific, checkable version of "favor simultaneous when coupling is strong": ask first whether the dynamic model can supply symbolic or complex-step derivatives with respect to the plant and control variables. If it can, simultaneous CCD's coordination advantage tends to show up as a real computational advantage too, especially at fine time discretization. If only real-valued finite differences are available — common with legacy, black-box, or multiphysics simulation codes — nested CCD, using a tailored and reliable inner-loop solver (a closed-form LQR solve, or a quadratic program when the inner problem is linear-quadratic), can be faster by an order of magnitude despite its repeated inner solves. This turns the derivative-availability question already listed below from a soft preference into the single most decisive factor observed in a rigorous, implementation-controlled study.

A companion checklist item

Before committing to nested CCD, check that the inner problem is feasible for every plant design the outer loop might propose, or add an explicit outer-loop feasibility constraint. In a documented case, uniform sampling within simple design bounds left 44% of candidate plants without a feasible inner solution.

Questions to ask before choosing

1. How expensive is each simulation?
2. Are accurate derivatives available?
3. How nonlinear and nonconvex is the problem?
4. How many variables and trajectory points are involved?
5. Which legacy tools must be preserved?
6. Is the aim a quick baseline or the best achievable design?

Examples and Architecture Comparison

Example 5.1: active suspension

Let plant variables be $\mathbf{x}_p = [k_s, c_s]^T$ and controller gains be $\mathbf{x}_c = [K_1, K_2, K_3, K_4]^T$.

- **Single-pass:** choose k_s, c_s using passive metrics, then tune gains.
- **Iterative sequential:** alternate plant redesign and gain tuning until performance stabilizes.
- **Nested:** solve an inner controller problem for every (k_s, c_s) candidate.
- **Simultaneous:** optimize all six decisions in one problem.

The engineering system remains the same; only the solution organization changes.

Example 5.2: wind turbine

Plant variables may include tower and structural parameters, while control variables include pitch settings. When feedback strongly influences structural loads, nested or simultaneous methods become attractive because the best plant depends on controller capability.

Example 5.3: marine energy device

Buoy geometry and PTO properties can be plant decisions, while reactive or latching parameters define control. Strong hydrodynamic–control coupling may make nested or simultaneous coordination substantially better than a purely sequential process.

Example 5.4: a controlled comparison on active suspension

A quarter-car active-suspension CCD problem — sprung and unsprung mass connected by a force actuator, with a physically parameterized spring ($\mathbf{x}_p$ including wire diameter, helix diameter, pitch, and number of active coils) and a physically parameterized damper — was solved to a converged reference solution using both nested and simultaneous strategies at a fine mesh ($N_t = 5000$) and tight tolerances. Both strategies converged to essentially the same design: an objective value of $o^* \approx 2.0677$, with the simultaneous strategy giving an effective spring constant $k_s^* \approx 2.366 \times 10^4$ N/m and damping $c_s^* \approx 839.8$ Ns/m, within 2% of the nested strategy's values, so the resulting optimal state and control trajectories were nearly indistinguishable between the two strategies. This is a directly verified instance of the "same ideal solution" claim from the previous section — but the two strategies took very different amounts of computation to reach it, depending entirely on which derivative methods were available.

Example 5.5: how the architecture ranking scales with mesh density

A strain-actuated solar-array co-design test problem, solved by direct transcription at three levels of mesh accuracy, shows how sensitive a nested-versus-simultaneous comparison is to the efficiency of the inner-loop solver. A naive, general-purpose nested implementation was always many orders of magnitude slower than a nested implementation that recognized the inner loop's linear-quadratic structure and solved it as a quadratic program. Comparing that efficient nested (QP) implementation against simultaneous CCD:

Mesh accuracy	Simultaneous runtime	Nested (QP) runtime
Low ($N_t = 10$)	0.06 s	0.16 s
Medium ($N_t = 100$)	6.54 s	1.22 s
High ($N_t = 1{,}000$)	11,464.33 s	17.63 s

At low accuracy, simultaneous CCD was actually faster, though both solve times were small in absolute terms. At medium and high accuracy, the efficient nested implementation was dramatically faster — by roughly $650\times$ at

the highest mesh density tested — while both strategies converged to plant designs within a fraction of a percent of each other and of the closed-form solution available for this test problem. The lesson is not that nested is unconditionally faster: the advantage of nested CCD here is entirely contingent on having an efficient method for the *specific* inner-loop problem structure, exactly as the derivative-method dependence in Example 5.4 made the simultaneous-versus-nested ranking contingent on derivative availability rather than on architecture alone.

Qualitative comparison

Architecture	Coordination	Implementation	Typical burden	Good use case
Single-pass sequential	Low	Low	Low	Weak coupling, quick baseline
Iterative sequential	Moderate	Low–moderate	Moderate	Reusable separate tools
Nested CCD	High	Moderate	High repeated inner solves	Best-control evaluation of each plant
Simultaneous CCD	Very high	High	Large integrated solve	Strong coupling and unified optimization

Common mistakes

1. **Assuming simultaneous always means better.** Poor scaling or derivatives can defeat its theoretical advantages.
2. **Assuming equivalence means identical computation.** Formal equivalence does not eliminate local minima or numerical error.
3. **Ignoring setup cost.** Elegant integration may not fit a short project.
4. **Using a single pass under strong coupling.** Important design opportunities may remain hidden.
5. **Ignoring nested inner-solve cost.** Controller optimization can dominate runtime.
6. **Neglecting initialization.** Architecture and initialization often interact strongly.
7. **Comparing architectures with mismatched implementation quality.** A published "nested is better" or "simultaneous is better" conclusion often reflects which implementation had accurate derivatives and a reliable inner solver, not an intrinsic property of the architecture. A fair comparison holds implementation quality fixed and varies only the architecture.

Chapter Summary and Problems

Summary

Single-pass design is simple and inexpensive but weakly coordinated. Iterative sequential design exchanges information repeatedly while preserving disciplinary subproblems. Nested CCD evaluates each plant through an inner optimal controller and can be computationally costly. Simultaneous CCD solves all coupled decisions in one problem and provides the strongest coordination, at the cost of numerical and software complexity.

Architectures can target the same ideal solution yet behave differently because of initialization, local minima, derivative quality, scaling, approximate solves, and stopping criteria. Architecture choice should reflect both mathematical coupling and practical engineering constraints. A controlled, implementation-fair comparison shows that two of these practical factors — derivative-method availability and inner-loop feasibility — can matter more than the choice of architecture itself, and that nested CCD's bi-level structure connects directly to the multidisciplinary feasible (MDF) formulation used more broadly in multidisciplinary design optimization.

Key terms

CCD architecture; coordination strategy; single-pass sequential design; iterative sequential design; nested CCD; simultaneous CCD; outer loop; inner loop; bi-level optimization; one-level optimization; mathematical equivalence; practical difference; coupling strength; computational tradeoff; legacy tools; derivative availability; multidisciplinary feasible (MDF); individual disciplinary feasible (IDF); augmented Lagrangian coordination (ALC); analytical target cascading (ATC); induced region; outer-loop feasibility constraint; inner-loop feasibility; derivative method (symbolic, complex-step, finite difference); direct transcription; algebraic Riccati equation.

Problems

1. **Convergence of iterated sequential design.** Let $J(p,c) = \frac{1}{2}p^T H_{pp} p + p^T H_{pc} c + \frac{1}{2}c^T H_{cc} c - b_p^T p - b_c^T c$ with a positive-definite joint Hessian. Derive the exact iteration matrix for alternating plant and control minimization and prove the necessary and sufficient spectral condition for linear convergence.
2. **Nested value-function gradient.** The inner optimal-control problem defines $\psi(p) = \min_{x,u} J(p,x,u)$ subject to transcription constraints $c(p,x,u) = 0$ and $g(p,x,u) \le 0$. Under strong regularity, derive $\nabla_p \psi$ from the inner KKT multipliers and show its equivalence to simultaneous first-order stationarity.
3. **Failure of nested differentiability.** Consider $\psi(p) = \min_c (c^2 - p)^2 + \rho c^2$. Determine all lower-level minimizers and derive the points at which the value function or optimizer mapping loses differentiability, explaining the consequence for a gradient-based outer loop.
4. **Simultaneous transcription structure.** For $\dot{x} = f(x,u,p)$ transcribed by a one-step defect on N intervals, order the NLP variables by time and derive the block sparsity pattern of its constraint Jacobian and Lagrangian Hessian, including the dense columns created by time-invariant plant variables.
5. **Nested infeasibility restoration.** The inner problem is infeasible for some p because $g(p,x,u) \le 0$ cannot be satisfied. Construct an exact-penalty feasibility-restoration inner problem that defines a finite outer value for every bounded p, and derive a condition on the penalty weight under which feasible local minima of the original problem are preserved.
6. **Fair comparison of coordination strategies.** For the active quarter-car equations $M(p)\ddot{q} + C(p)\dot{q} + K(p)q = B_u u + B_w w$, specify a numerical experiment that compares nested and simultaneous CCD under identical discretization, tolerances, derivatives, starts, bounds, and feasibility tests, reducing the comparison to one statistically defensible performance metric.
7. **Augmented-Lagrangian coordination.** Split a CCD problem into plant and control subproblems coupled through a trajectory copy $y_p(t) = y_c(t)$. Derive an augmented-Lagrangian iteration with multiplier and penalty updates and state a residual-based stopping condition that certifies coupling consistency.

8. **Architecture selection as mixed-integer CCD.** A suspension may contain a passive spring, damper, active actuator, and tuned-mass absorber, each selected by a binary variable. Formulate a mixed-integer dynamic optimization problem with logical sizing bounds, a component-count penalty, common ride and handling metrics, and physically valid dynamics for every selected topology.
9. **Computational break-even analysis.** A nested solve requires n_o outer iterations and an average of n_i inner NLP solves, whereas a simultaneous solve has n_z variables and a sparse KKT factorization cost proportional to n_z^α. Derive a break-even inequality that includes derivative cost, warm-start savings, parallelism, and the probability of inner infeasibility.
10. **Architecture-selection rule.** Given estimates of coupling strength, inner-problem failure rate, derivative availability, transcription size, and required solution accuracy, construct a quantitative decision rule that selects single-pass sequential, iterated sequential, nested, or simultaneous CCD and justify the rule using conditioning and convergence arguments.

References and further reading

1. Allison, J. T., & Herber, D. R. (2014). Multidisciplinary design optimization of dynamic engineering systems. *AIAA Journal*, *52*(4), 691–710. DOI: 10.2514/1.J052182
2. Herber, D. R., & Allison, J. T. (2019). Nested and simultaneous solution strategies for general combined plant and control design problems. *Journal of Mechanical Design, 141*(1), Article 011402. DOI: 10.1115/1.4040705
3. Garcia-Sanz, M. (2019). Control co-design: An engineering game changer. *Advanced Control for Applications: Engineering and Industrial Systems, 1*(1), Article e18. DOI: 10.1002/adc2.18
4. Martins, J. R. R. A., & Ning, A. (2021). *Engineering design optimization*. Cambridge University Press.
5. Betts, J. T. (2010). *Practical methods for optimal control and estimation using nonlinear programming* (2nd ed.). Society for Industrial and Applied Mathematics. DOI: 10.1137/1.9780898718577
6. Sundarrajan, A. K., & Herber, D. R. (2021). Towards a fair comparison between the nested and simultaneous control co-design methods using an active suspension case study. In *2021 American Control Conference (ACC)* (pp. 358–365).

Optimal Control and Closed-Loop Control Co-Design

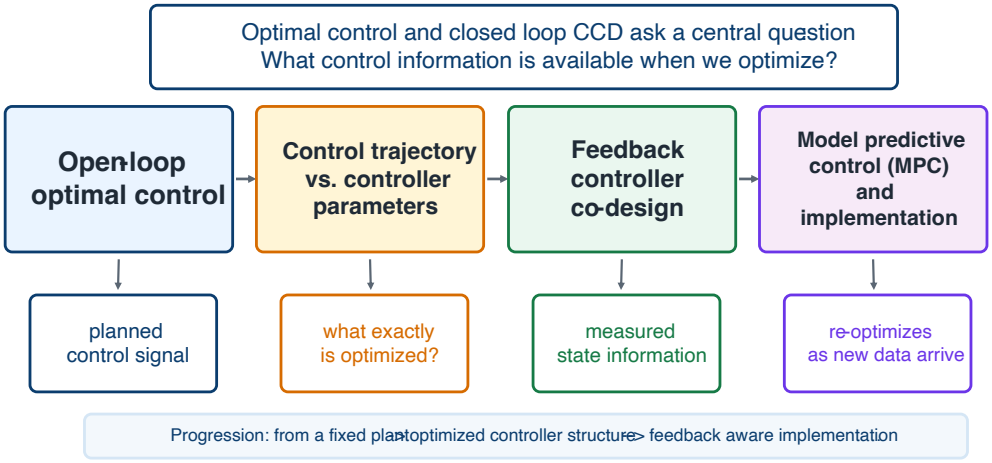
Chapter 6: Optimal Control and Closed-Loop Control Co-Design

From ideal control trajectories to implementable feedback policies

> The key distinction is information: an open-loop optimizer assumes a planned future, while a closed-loop controller reacts to measurements as the future unfolds.

Previous chapters formulated CCD problems and compared solution architectures. We now ask: **What kind of control object is being optimized?**

An optimizer may choose a complete control history $\mathbf{u}(t)$ over a finite interval, as in optimal control, or choose parameters of a feedback law. The first often gives an ideal performance benchmark; the second usually yields an implementable controller. This chapter connects those viewpoints through feedback-controller co-design, information availability, and model predictive control (MPC).



The chapter's main thread runs from ideal open-loop optimization to implementable closed-loop control.

Learning objectives

After completing this chapter, you should be able to:

1. explain open-loop optimal control and its role in CCD;
2. distinguish control-trajectory and controller-parameter optimization;
3. formulate feedback-controller co-design;
4. explain the limitations of perfect future information;
5. classify information available during operation;
6. explain receding-horizon MPC; and
7. describe paths from OLOC solutions to implementable feedback controllers.

Open-Loop Optimal Control

A **control trajectory** is a signal planned over time. A **controller** is a rule mapping available information to actions during operation. Disturbance, noise, and model uncertainty make exact future behavior unknowable, so most practical systems ultimately require feedback.

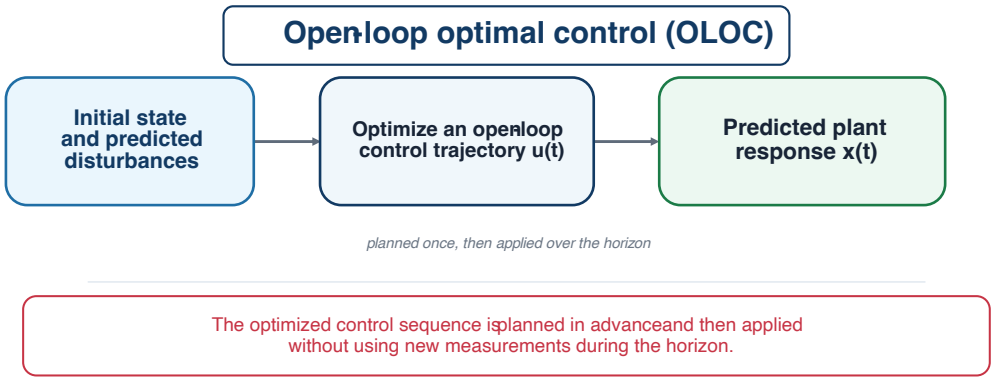
Main idea

Open-loop optimal control provides useful ideal benchmarks and design insight, but engineered systems usually need feedback that responds to measured information.

Basic formulation

Open-loop optimal control (OLOC) chooses $\mathbf{u}(t)$ over $t \in [t_0, t_f]$ using a model, initial condition, and assumed future information:

$$\begin{aligned} \underset{\mathbf{u}(\cdot),\mathbf{x}(\cdot)}{\text{minimize}} \quad & J = \Phi(\mathbf{x}(t_f)) + \int_{t_0}^{t_f} L(\mathbf{x}(t), \mathbf{u}(t), t)\, dt \\ \text{subject to} \quad & \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t), \\ & \mathbf{x}(t_0) = \mathbf{x}_0, \\ & \mathbf{c}(\mathbf{x}(t), \mathbf{u}(t), t) \le \mathbf{0}, \\ & \mathbf{b}(\mathbf{x}(t_f)) \le \mathbf{0}. \end{aligned}$$



The open-loop trajectory is planned first and then applied as planned.

OLOC can be reasonable when the horizon is short, uncertainty is small, the result is a benchmark, or the trajectory is repeatedly recomputed as in MPC.

Role in CCD

OLOC can answer:

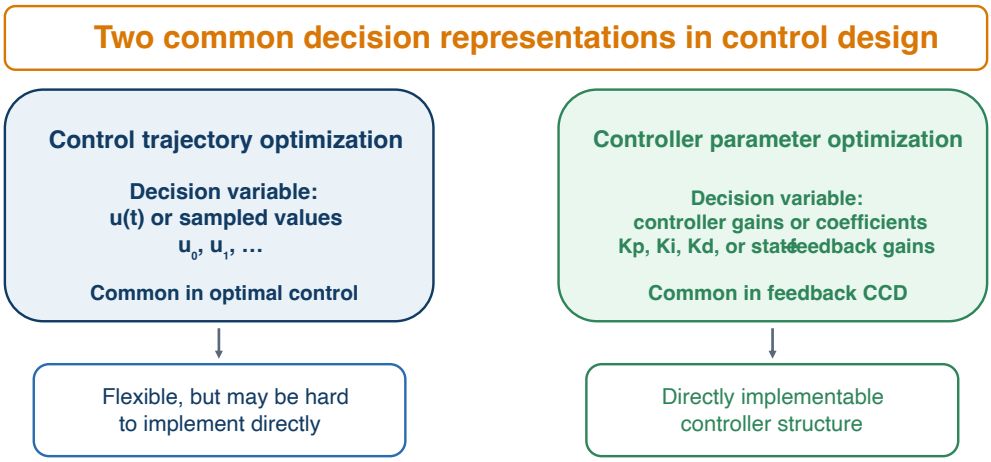
- What is the best achievable performance for a candidate plant under an assumed scenario?
- How much energy can ideally be extracted or saved?
- What qualitative features should control action have?
- Which plant properties make the control task easier?

For a wave-energy device, optimizing PTO force against a fixed recorded wave provides an upper-bound energy benchmark. It does not automatically produce a realizable controller, but it exposes valuable plant and control structure.

Control Trajectories versus Controller Parameters

Two different control-related decision sets are common:

1. the time-varying signal $\mathbf{u}(t)$ or its sampled values; and
2. parameters $\mathbf{x}_c$ in a policy $\mathbf{u} = \boldsymbol{\kappa}(\mathbf{y}; \mathbf{x}_c)$.



Trajectory optimization chooses a signal; controller-parameter optimization chooses a rule that generates the signal from information.

Optimizing trajectories

A discretized trajectory may use decisions

$$\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_N.$$

This offers high flexibility, reveals best-case performance, and matches optimal-control theory. However, the result may be specific to one disturbance realization, difficult to implement directly, and does not itself define feedback.

Optimizing controller parameters

Choose a realizable structure

$$\mathbf{u}(t) = \boldsymbol{\kappa}(\mathbf{y}(t); \mathbf{x}_c),$$

where $\mathbf{y}(t)$ contains available measurements. Parameters may be PID gains, state-feedback gains, estimator settings, basis weights, or MPC tuning weights.

This representation is directly implementable, explicitly feedback-based, and often lower dimensional. Its limitation is structural: a restricted controller may sacrifice ideal performance, and selecting the controller architecture is itself a design decision.

Why this matters for plant design

A plant optimized under unrestricted open-loop control may not remain best when paired with a realistic feedback structure. The control representation changes the attainable closed-loop behavior and can therefore change the optimal physical design.

Activity 6.1: Open-Loop versus Feedback Control Co-Design

Activity 6.1: Open-Loop versus Feedback Control Co-Design

Consider the mass-normalized system

$$\dot{x}_1 = x_2, \qquad \dot{x}_2 = -kx_1 - cx_2 + u + d(t).$$

The plant variables are

$$0.5 \le k \le 8, \qquad 0.1 \le c \le 3,$$

and the actuator-capacity variable is

$$0.5 \le F_{\max} \le 5.$$

Use the feedback law

$$u(t) = \operatorname{sat}\left(-K_p x_1(t) - K_d x_2(t), F_{\max}\right),$$

with

$$0 \le K_p \le 10, \qquad 0 \le K_d \le 8.$$

Minimize

$$J = \frac{1}{N_s}\sum_{s=1}^{N_s}\int_0^8 \left[x_{1,s}^2 + 0.1x_{2,s}^2 + 0.02u_s^2\right] dt + 0.01k^2 + 0.02c^2 + 0.04F_{\max}^2.$$

Use the scenarios

$$\mathbf{x}_0^{(1)} = \begin{bmatrix}1\\0\end{bmatrix}, \qquad \mathbf{x}_0^{(2)} = \begin{bmatrix}0.5\\1\end{bmatrix}, \qquad \mathbf{x}_0^{(3)} = \begin{bmatrix}-1\\0.5\end{bmatrix},$$

with

$$d_1(t) = 0, \qquad d_2(t) = 0.4\sin(2t), \qquad d_3(t) = 0.3\sin(3t).$$

1. Derive the unsaturated closed-loop dynamics.
2. Derive the conditions on k, c, K_p, and K_d for asymptotic stability.
3. Add the passive-safety constraint

$$\zeta_{\text{passive}} = \frac{c}{2\sqrt{k}} \ge 0.15.$$

4. Optimize

$$k, \quad c, \quad F_{\max}, \quad K_p, \quad K_d$$

simultaneously.

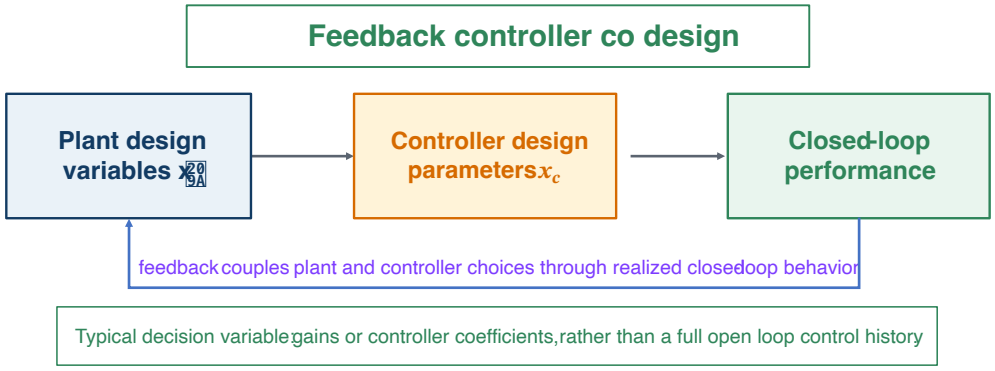
5. For the optimized plant, solve a separate OLOC problem in GPOPS-II or Dymos for each disturbance scenario.
6. Compute the relative implementation gap

$$G = \frac{J_{\text{feedback}} - J_{\text{OLOC}}}{J_{\text{OLOC}}}.$$

7. Explain why the plant optimized with unrestricted OLOC may differ from the plant optimized with the parameterized feedback controller.

Feedback-Controller Co-Design

Feedback-controller co-design optimizes physical variables together with parameters of a realizable feedback controller.



Performance is assessed in closed loop under a parameterized feedback law.

A generic formulation is

$$\begin{aligned} \underset{\mathbf{x}_p,\mathbf{x}_c}{\text{minimize}} \quad & J(\mathbf{x}_p, \mathbf{x}_c) \\ \text{subject to} \quad & \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), \mathbf{x}_p, \mathbf{w}(t)), \\ & \mathbf{u}(t) = \boldsymbol{\kappa}(\mathbf{y}(t); \mathbf{x}_c), \\ & \mathbf{y}(t) = \mathbf{h}(\mathbf{x}(t), \mathbf{x}_p), \\ & \text{state, control, and design constraints.} \end{aligned}$$

The optimizer chooses policy parameters rather than every future control value. The realized response depends on the plant, controller, disturbance, and available measurements.

Why closed-loop co-design is more realistic

Feedback can reject disturbances, reduce sensitivity to model mismatch, and react as state deviations occur. Vehicles, robots, wind turbines, marine devices, and industrial processes therefore operate primarily in closed loop.

For a wind turbine, structural variables and pitch-controller gains can be optimized together. During operation, the controller uses measured rotor speed and other signals to respond to changing wind conditions.

Activity 6.2: Sensor and Output-Feedback Co-Design

Activity 6.2: Sensor and Output-Feedback Co-Design

Consider the stochastic plant

$$\dot{\mathbf{x}} = A(p)\mathbf{x} + Bu + \mathbf{w},$$

where

$$A(p) = \begin{bmatrix} 0 & 1 \\ -p & -0.4 \end{bmatrix}, \qquad B = \begin{bmatrix} 0 \\ 1 \end{bmatrix},$$

and $1 \le p \le 4$ is a plant-design variable.
The process-noise covariance is

$$W = \begin{bmatrix} 0 & 0 \\ 0 & 0.04 \end{bmatrix}.$$

Three sensor architectures are available:

$$C_1 = \begin{bmatrix} 1 & 0 \end{bmatrix}, \qquad V_1 = 0.01, \qquad C_{\text{sensor},1} = 0.20,$$

$$C_2 = \begin{bmatrix} 0 & 1 \end{bmatrix}, \qquad V_2 = 0.04, \qquad C_{\text{sensor},2} = 0.10,$$

and

$$C_3 = I, \qquad V_3 = \begin{bmatrix} 0.01 & 0 \\ 0 & 0.04 \end{bmatrix}, \qquad C_{\text{sensor},3} = 0.35.$$

Use the control-performance weights

$$Q = \begin{bmatrix} 10 & 0 \\ 0 & 1 \end{bmatrix}, \qquad R = 0.2.$$

1. Compute the observability matrix for each sensor architecture and determine whether the system is observable for all feasible p.
2. For each value of p, solve the LQR algebraic Riccati equation and compute the state-feedback gain $K(p)$.
3. For each sensor architecture, solve the Kalman-filter Riccati equation

   $$A\Sigma + \Sigma A^T - \Sigma C^T V^{-1} C\Sigma + W = 0$$

   and compute

   $$L = \Sigma C^T V^{-1}.$$

4. Construct the LQG controller

   $$u = -K\hat{\mathbf{x}}.$$

5. For each sensor architecture, optimize p using

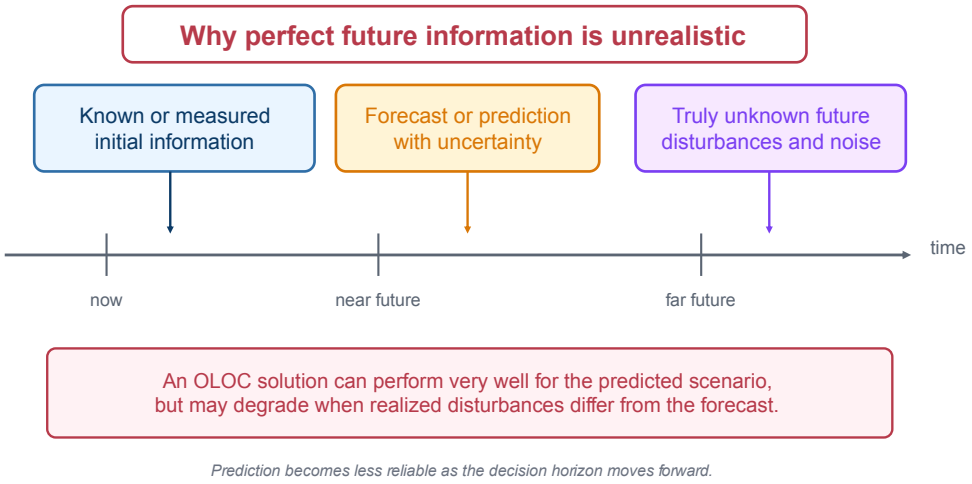
   $$J_{\text{total}} = \mathbb{E}\left[\int_0^{20} \left(\mathbf{x}^T Q \mathbf{x} + R u^2\right) dt\right] + 0.03p^2 + C_{\text{sensor}}.$$

6. Estimate the expected cost using at least 200 Monte Carlo noise realizations.
7. Identify the optimal plant–sensor–controller combination.
8. Explain why the separation principle does not imply that sensor selection and plant design can be performed independently.

Future Information and Information Availability

The hidden open-loop assumption

Many OLOC studies assume an exact future disturbance or reference history: a wave record, road profile, or wind trace. This is convenient but often unrealistic.



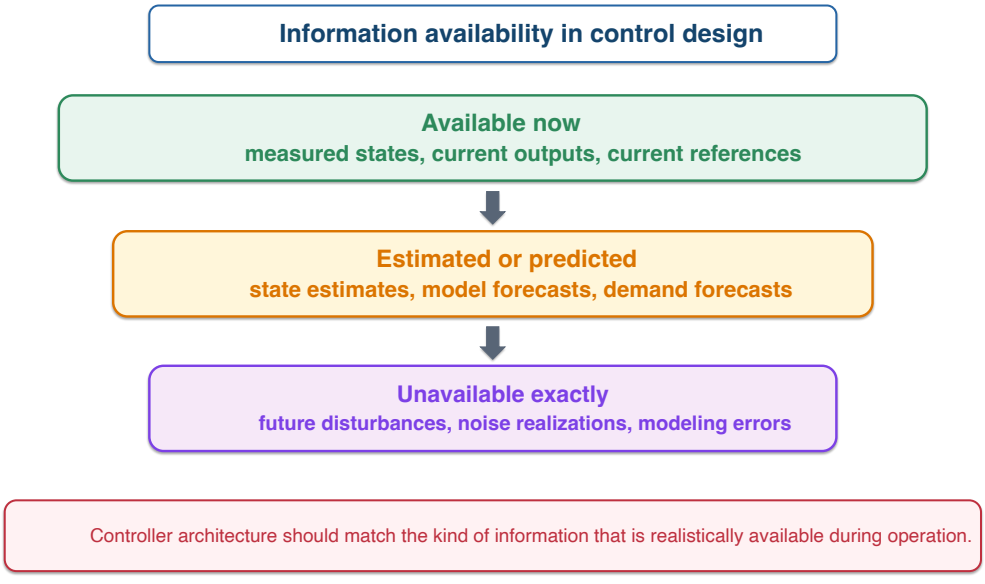
Perfect future information is rarely available.

An optimizer with exact foresight may exploit details a real controller could never know. Its performance should be interpreted as an upper bound, benchmark, or source of design insight—not automatically as implementable performance.

Forecasts can still be useful. Weather forecasts, route preview, and demand prediction provide future information with uncertainty. Forecast is not certainty, and a practical controller must account for error.

Control as an information problem

The key question is: **What information is available when an action must be chosen?**



A controller can use only realistically measured, estimated, or predicted information.

Three useful categories are:

1. **Available now:** current measurements, state estimates, and commands.

2. **Estimated or predicted:** observer states, previews, and uncertain forecasts.
3. **Unavailable exactly:** future disturbances, future sensor noise, and future modeling errors.

A formal information-horizon taxonomy

A co-design framework developed for semi-active suspension design gives these three categories precise names and ties each to a recognizable class of problem:

- **Complete horizon:** the control problem—whether OLOC or a closed-loop controller—is solved with complete information about the environment (exogenous inputs, disturbances, the physical model) over the entire time horizon. This is a strong assumption, but it is standard practice in offline optimization problems such as robotic-manipulator path planning and space-shuttle reentry trajectory design, where the full mission profile is known before the trajectory is ever computed.
- **Instantaneous:** environment information is available only at the current instant. Closed-loop control for tracking and regulation is the textbook example. With only instantaneous information, achieving an acceptable level of performance over the *entire* horizon—for instance, satisfying a path constraint—can be genuinely difficult.
- **Limited horizon:** the control problem is solved for only a small portion of the horizon, and that partial solution is recomputed at regular intervals during operation. Look-ahead control of wind turbines and fuel-optimal routing of hauling trucks or railways are representative examples. This is the category MPC belongs to: it is a limited-horizon strategy that can approach the performance of a complete-horizon solution while retaining the practicality of instantaneous information, making it a genuine middle ground rather than a compromise between two unrelated extremes.

This taxonomy sharpens the three-part classification above: "available now" is instantaneous information; "estimated or predicted" is limited-horizon information, with the horizon length itself a design choice; and "unavailable exactly" is precisely what a complete-horizon OLOC study assumes away.

Implications for CCD

Plant decisions can change information availability. Sensor placement can improve observability; mechanical redesign can make important modes easier to measure; actuator selection changes authority; robust physical design can reduce sensitivity to uncertain disturbances.

Important viewpoint

A control architecture should not assume more information than the real system can obtain. Practical control selects actions from the right information set, not from impossible foresight.

Activity 6.3: Perfect-Information OLOC versus Causal Feedback

Activity 6.3: Perfect-Information OLOC versus Causal Feedback

Consider

$$\dot{x}(t) = -0.8x(t) + u(t) + w(t), \qquad x(0) = 1, \qquad 0 \le t \le 6,$$

with

$$-3 \le u(t) \le 3.$$

The nominal disturbance is

$$w_0(t) = 0.4\sin(1.5t) + 0.2\sin(4t).$$

Minimize

$$J = \frac{1}{2}x(6)^2 + \frac{1}{2}\int_0^6 \left[x(t)^2 + 0.05u(t)^2\right] dt.$$

1. Write the Hamiltonian and derive the costate equation.
2. Show that the unconstrained optimal control is

$$u^*(t) = -\frac{\lambda(t)}{0.05},$$

   and write the corresponding saturated optimal-control law.
3. State the terminal condition for the costate.
4. Solve the two-point boundary-value problem using GPOPS-II, Dymos, or a shooting method.
5. Design a finite-horizon causal LQR controller

$$u(t) = -K(t)x(t)$$

   by solving the Riccati equation backward in time.
6. Apply both controllers to the perturbed disturbance

$$w(t) = w_0(t) + 0.15\sin(2.3t + \phi),$$

   for

$$\phi \in \left\{0, \frac{\pi}{2}, \pi, \frac{3\pi}{2}\right\}.$$

7. Compare the nominal and perturbed costs. Quantify the performance advantage caused by perfect future disturbance information.

Activity 6.4: Disturbance Preview and Information Availability

Activity 6.4: Disturbance Preview and Information Availability

Consider the discrete-time system

$$\mathbf{x}_{k+1} = A\mathbf{x}_k + Bu_k + Ew_k,$$

where

$$A = \begin{bmatrix} 1 & h \\ 0 & 1 \end{bmatrix}, \qquad B = E = \begin{bmatrix} h^2/2 \\ h \end{bmatrix}, \qquad h = 0.1.$$

Use the horizon $N = 60$ and the objective

$$J = \mathbf{x}_N^T Q_f \mathbf{x}_N + \sum_{k=0}^{N-1} \left(\mathbf{x}_k^T Q \mathbf{x}_k + R u_k^2 \right),$$

with

$$Q = \begin{bmatrix} 10 & 0 \\ 0 & 1 \end{bmatrix}, \qquad Q_f = 10Q, \qquad R = 0.1.$$

The disturbance is

$$w_k = 0.8 \sin(0.25k) + 0.3 \sin(0.9k).$$

Assume that at time k, the controller knows only the next L disturbance values.

1. Assume the value function has the affine-quadratic form

$$V_k(\mathbf{x}) = \mathbf{x}^T P_k \mathbf{x} + 2\mathbf{s}_k^T \mathbf{x} + c_k.$$

   Derive the dynamic-programming recursion for P_k, $\mathbf{s}_k$, and c_k.

2. Show that the optimal control has the form

$$u_k = -K_k \mathbf{x}_k - u_{k,\mathrm{ff}},$$

   where the feedforward term depends on the available disturbance preview.

3. Implement controllers for $L = 0,\ 2,\ 5,\ 10,\ 20,\ 60$.

4. Plot the optimal cost as a function of preview length.

5. Compute the marginal value of preview:

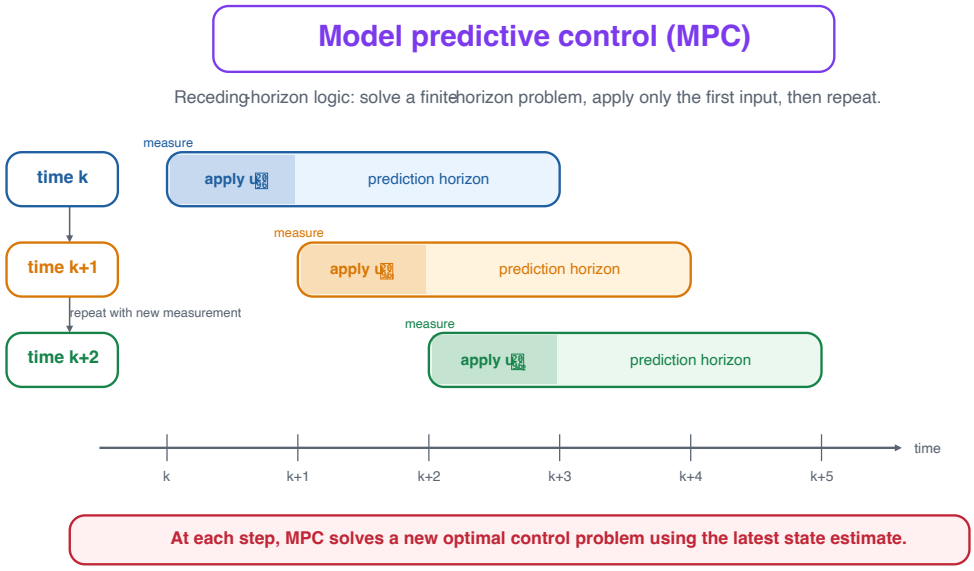
$$\Delta J_L = J_L - J_{L+1}.$$

6. Determine the preview length beyond which additional information provides less than a 1% improvement.

7. Explain how the optimal physical design of a suspension or energy converter could change as L changes.

Model Predictive Control

Model predictive control bridges optimal control and practical feedback. At each time step, MPC solves a finite-horizon problem using the current state estimate and possibly short-term forecasts. It applies only the first action, advances one step, and solves again.



The receding-horizon strategy repeatedly updates a local open-loop plan.

Why MPC is feedback

Each internal optimization is open loop, but the overall controller is closed loop because every new solve uses updated measurements or estimates.

At time k:

1. measure or estimate $\mathbf{x}_k$;
2. solve a finite-horizon optimal-control problem;
3. obtain $\mathbf{u}_k, \ldots, \mathbf{u}_{k+N-1}$;
4. apply only $\mathbf{u}_k$; and
5. advance and repeat.

Advantages

- Incorporates a predictive model directly.
- Handles multivariable systems naturally.
- Enforces state and input constraints.
- Updates from measurements, creating feedback.
- Can use preview and short-term forecasts.

Limitations

- Requires online optimization.
- Depends on model quality.
- Is sensitive to horizon length and discretization.

- Can be difficult to execute in real time for fast or large systems.

MPC in CCD

Plant design may be optimized with MPC prediction horizon, control horizon, weighting matrices, and estimator settings. This gives a more implementable closed-loop formulation than pure OLOC while retaining optimization-based constraint handling.

Prediction horizon as an information-availability dial

A 2026 vehicle-suspension study frames the MPC prediction horizon explicitly as a tunable "information availability" setting rather than a fixed implementation detail. Two hyperparameters govern how much information the controller effectively has: the prediction horizon p (an integer number of discrete steps) and the control sampling time T_s, which together give a prediction horizon length in time of $T_p = pT_s$. A control horizon $m \leq p$ specifies how many of those steps the control signal is allowed to vary over; setting $m = p$ keeps comparisons fair as p is increased. With a very short horizon, MPC behaves close to a simple feedback controller; as p grows, together with a finer model and more frequent updates, MPC approaches the accuracy of OLOC—though, unlike OLOC, it still incorporates no explicit knowledge of the past.

This makes p (and T_s) genuine control-design variables that can themselves be optimized inside a **nested CCD architecture**: an outer loop uses the covariance matrix adaptation evolution strategy (CMA-ES)—a gradient-free, stochastic, population-based global optimizer well suited to nonconvex, nonseparable, and noisy objectives—to choose plant design variables, while an inner loop evaluates each candidate plant by running a full closed-loop MPC (or OLOC) simulation and returning the resulting objective value.

State estimation with a hybrid Kalman filter

MPC needs a state estimate at every sampling instant, but the plant is naturally modeled with continuous-time dynamics while sensor data arrives at discrete instants. A **hybrid Kalman filter** matches this mismatch: it propagates a continuous-time model,

$$\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u} + \mathbf{w}(t), \qquad \mathbf{w}(t) \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}),$$

forward between measurements, and performs a discrete-time update using

$$\mathbf{y}_k = C\mathbf{x}_k + \mathbf{v}_k, \qquad \mathbf{v}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_k),$$

whenever a new measurement $\mathbf{y}_k$ arrives. The updated state estimate initializes the next MPC solve, and the controller never sees the true state directly—only the sensed outputs $\mathbf{y}_k$. Because the Kalman gain and error covariance in this linear setting are not themselves functions of the measured data, they can be computed offline and, after an initial transient, replaced by a constant steady-state gain instead of the full time-varying gain, substantially reducing online computation with little effect on estimation accuracy.

Activity 6.5: Nested Plant–MPC Co-Design

Activity 6.5: Nested Plant–MPC Co-Design

Consider the sampled system

$$\mathbf{x}_{k+1} = A(k_s)\mathbf{x}_k + Bu_k + Ew_k,$$

where

$$A(k_s) = \begin{bmatrix} 1 & h \\ -k_s h & 1 - 0.4h \end{bmatrix}, \qquad B = E = \begin{bmatrix} 0 \\ h \end{bmatrix}, \qquad h = 0.1.$$

The plant variables are

$$0.5 \le k_s \le 5, \qquad 0.5 \le F_{\max} \le 3.$$

At every time step, use MPC to solve

$$\min_{\{u_i\}} \sum_{i=0}^{N_p-1} \left(\mathbf{x}_i^T Q \mathbf{x}_i + R u_i^2 \right) + \mathbf{x}_{N_p}^T P \mathbf{x}_{N_p},$$

subject to

$$\mathbf{x}_{i+1} = A(k_s)\mathbf{x}_i + B u_i,$$

and

$$|u_i| \le F_{\max}, \qquad |x_{1,i}| \le 1.$$

Use

$$Q = \begin{bmatrix} 10 & 0 \\ 0 & 1 \end{bmatrix}, \qquad R = 0.05.$$

1. Derive the condensed prediction equation

$$\mathbf{X} = \mathcal{A}\mathbf{x}_k + \mathcal{B}\mathbf{U}.$$

2. Write the MPC problem as a quadratic program

$$\min_{\mathbf{U}} \frac{1}{2}\mathbf{U}^T H \mathbf{U} + \mathbf{f}^T \mathbf{U}.$$

3. Implement a nested CCD architecture in which the outer optimization chooses k_s and $F_{\max}$, while the inner problem performs closed-loop MPC simulation.
4. Use prediction horizons $N_p = 5,\ 10,\ 20,\ 40$.
5. Minimize

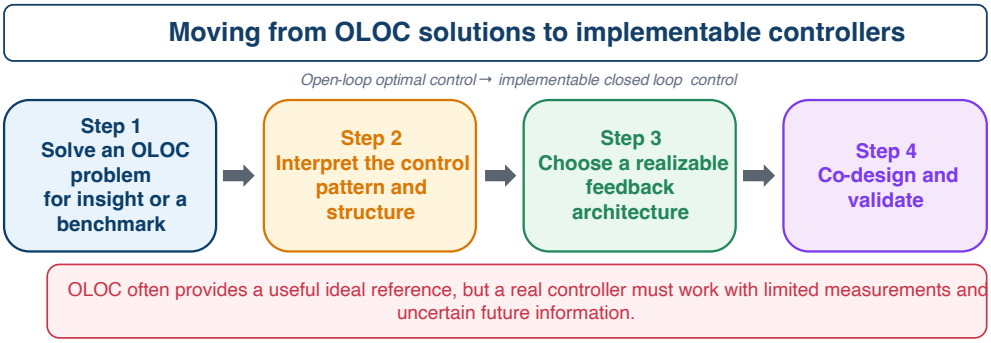
$$J_{\mathrm{CCD}} = \sum_{k=0}^{100} \left(\mathbf{x}_k^T Q \mathbf{x}_k + R u_k^2 \right) + 0.02 k_s^2 + 0.05 F_{\max}^2.$$

6. Compare the optimal plant for each prediction horizon.
7. Compare the MPC result with a full-horizon OLOC solution obtained using GPOPS-II or Dymos.
8. Introduce a 20% stiffness-model error and determine whether recursive feasibility is maintained.
9. Explain why prediction horizon should be regarded as a control-design variable in CCD.

From OLOC to Implementable Control

A productive development path is:

1. solve an ideal OLOC problem;
2. study its control patterns and performance;
3. choose a realizable controller structure that captures the important behavior; and
4. co-design the plant with that implementable controller.



An ideal trajectory can guide—but should not be confused with—the final controller.

What OLOC can teach us

An optimal trajectory can reveal:

- when control action should be active;
- whether it behaves like stiffness, damping, or phase lead;
- required control bandwidth and authority; and
- which physical plant properties are most valuable.

Typical transitions

- OLOC → PID or gain-scheduled feedback.
- OLOC → LQR or state feedback.
- OLOC → MPC.
- OLOC → a rule-based controller informed by optimal behavior.
- OLOC → a parameterized policy fitted to optimal trajectories.

OLOC remains valuable because it supplies a best-case reference, explains ideal behavior, benchmarks practical controllers, and helps separate plant limitations from controller limitations. It is often best treated as an intermediate design-analysis tool.

A three-stage design framework

One way to operationalize the OLOC-to-implementable-control transition is a three-stage process developed for dynamic-system architecture design:

1. **Stage 1 — plant architecture design, co-designed with OLC.** The physical architecture and its design variables are optimized together with an unrestricted control trajectory $\mathbf{u}(t)$. No actuator or control-architecture assumptions are made, so this stage searches the broadest design space and produces a system-optimal benchmark plant.

2. **Stage 2 — controller architecture design, co-designed with CLC.** A specific, implementable control architecture (for example, full-state feedback with optimized gains) is selected, and the plant is re-optimized jointly with that controller's finite-dimensional parameter vector.
3. **Stage 3 — digital controller design.** The continuous-time control law produced in Stage 2 is realized as a digital controller, addressing sampling, discretization, and computation.

Mechanical engineers typically own Stage 1, while control engineers own Stages 2 and 3. Iteration between stages—and revision of a stage's own formulation—is expected as insights from later stages feed back into earlier ones.

A worked progression: semi-active suspension. A trailing-arm quarter-car suspension study shows how performance degrades, in a controlled and explainable way, as each stage adds a layer of realism. Each row below is informed by the solution of the previous one:

Problem	Control representation	Φ_*	Change
P0 — passive	fixed linear damping, no active or semi-active control	5.80×10^{-4}	an idealized passive benchmark that cannot actually be realized, because underlying geometric nonlinearities keep true passive damping nonlinear
P1 — active OLC	unrestricted actuator force $u(t)$ (Stage 1)	2.06×10^{-4}	benchmark: best achievable performance with no actuator or architecture restriction
P2 — semi-active OLC, ideal damper	force restricted to always dissipate energy, $F(u) = -u(t)\,(\dot{z}_s - \dot{z}_{us})$, $u(t) \geq 0$	3.72×10^{-4}	about $1.80\times$ worse than P1
P3 — semi-active OLC, realistic MR-damper model	current-to-force map $F(I, \dot{z}_s - \dot{z}_{us})$ identified from a physical damper	9.58×10^{-4}	about $2.57\times$ worse than P2
P4 — semi-active CLC, full-state feedback (Stage 2)	optimized feedback gains $\mathbf{K}$ acting on measured/estimated states, damper current $I(t)$ as the physical output, $0 \leq \mathbf{K}\boldsymbol{\xi}(t) \leq 1$	2.00×10^{-3}	about $2.08\times$ worse than P3; about $9.70\times$ worse than the P1 benchmark

The P2 solution was used to select and size an actual component: a Lord 8041-1 magnetorheological (MR) damper, with a continuous current operating range of 0-1 A and a maximum stroke of 74 mm. That damper's hysteretic, current-dependent force-velocity behavior was characterized in the laboratory to build the P3 surrogate model, $F(I, \dot{z}_s - \dot{z}_{us})$. In P4, all states were assumed measurable—mass accelerations from accelerometers, relative displacements from linear encoders, and velocities obtained by integrating the acceleration signals—and the feedback gains were optimized subject to the damper's saturation limits. No single step in P1 $\rightarrow$ P4 involved a design mistake: each one simply removed an assumption (unlimited actuator authority, an idealized semi-active component, or unlimited state/future information) that a real system cannot provide, and the resulting performance gap is the accumulated cost of implementability.

Activity 6.6: Converting an OLOC Trajectory into TVLQR Feedback

Activity 6.6: Converting an OLOC Trajectory into TVLQR Feedback

Consider a torque-controlled pendulum. Let $\theta = 0$ denote the upright position and $\theta = \pi$ the downward position.
The dynamics are

$$\dot{\theta} = \omega, \qquad \dot{\omega} = \frac{g}{l}\sin\theta - \frac{b}{ml^2}\omega + \frac{1}{ml^2}u.$$

Use

$$m = 1\ \text{kg}, \qquad l = 1\ \text{m}, \qquad b = 0.05\ \text{N m s},$$

and

$$g = 9.81\ \text{m/s}^2, \qquad |u(t)| \le 8\ \text{N m}.$$

The boundary conditions are

$$\mathbf{x}(0) = \begin{bmatrix} \pi \\ 0 \end{bmatrix}, \qquad \mathbf{x}(4) = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

Minimize

$$J = \int_0^4 \left(0.001\omega(t)^2 + 0.01u(t)^2\right) dt.$$

1. Solve the swing-up OLOC problem using GPOPS-II or Dymos.
2. Denote the optimal trajectory by

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

   Linearize the dynamics along the optimal trajectory and show that

   $$A(t) = \begin{bmatrix} 0 & 1 \\ \frac{g}{l}\cos\theta^*(t) & -\frac{b}{ml^2} \end{bmatrix}, \qquad B(t) = \begin{bmatrix} 0 \\ \frac{1}{ml^2} \end{bmatrix}.$$

3. Solve the time-varying Riccati equation

   $$-\dot{S} = A^T S + SA - SBR^{-1}B^T S + Q,$$

   with terminal condition

   $$S(4) = Q_f.$$

4. Implement the trajectory-tracking controller

$$u(t) = \mathrm{sat}\left[u^*(t) - R^{-1}B(t)^T S(t)\left(\mathbf{x}(t) - \mathbf{x}^*(t)\right)\right].$$

5. Compare pure open-loop implementation and TVLQR feedback for the perturbed initial conditions

$$\theta(0) = \pi + \Delta\theta, \qquad \Delta\theta \in \{-0.15, -0.10, 0.10, 0.15\}.$$

6. Repeat with

$$10\% \text{ error in } m, \qquad 10\% \text{ error in } l,$$

and additive measurement noise.

7. Report

$$\|\mathbf{x}(4)\|_2, \qquad \max_t |u(t)|, \qquad J.$$

8. Determine the largest initial perturbation for which the TVLQR controller successfully reaches the upright neighborhood

$$|\theta(4)| \le 0.05, \qquad |\omega(4)| \le 0.1.$$

9. Explain why the OLOC trajectory remains valuable even though it is not robust enough for direct implementation.

Examples and Comparison

Example 6.1: active suspension

For a known short road disturbance, OLOC can optimize actuator force $u(t)$ to reduce body acceleration and suspension deflection. This yields a best-case force history for that road profile. The result may reveal when strong actuation is needed. A final controller instead maps measured suspension deflection and body velocity to force, and its gains are tuned or co-designed in closed loop.

Example 6.2: wind turbine

Optimizing blade-pitch trajectory against a known wind trace gives an ideal load-reduction benchmark. A real turbine does not know the future gust exactly, so an implementable controller uses rotor-speed and load measurements, perhaps with preview. MPC or gain-scheduled feedback is a more realistic final architecture.

Example 6.3: marine energy device

OLOC can optimize PTO force for a recorded wave sequence and estimate theoretical energy capture. A practical controller must operate from measured or forecast waves. Closed-loop CCD redesigns geometry or PTO hardware together with that realizable controller.

Example 6.4: prediction horizon and architecture complexity in a quarter-car study

A study of six quarter-car active-suspension architectures makes the OLOC-versus-MPC trade-off concrete. Each architecture connects the road input to a single translational-force actuator through a different number of auxiliary masses, springs, and dampers—from a control-only case with no additional plant design variables up to a six-design-variable architecture. All six architectures share the objective

$$\psi_d = \int_{t_0}^{t_f} \left(w_1 (z_U - \delta)^2 + w_2 \ddot{z}_S^2 + w_3 F^2 \right) dt,$$

combining road-handling performance, passenger comfort (sprung-mass acceleration), and a control-effort penalty. For each architecture, the plant variables and the controller's prediction horizon p are optimized together in a nested CCD loop—once with OLOC in the inner loop, and once with MPC at $p = 40, 10, 5$ for a fixed control sampling time.

Two findings stand out:

- **Plant complexity cannot fully substitute for information.** Moving from a simple two-design-variable architecture to the richest six-design-variable architecture markedly improves the open-loop and long-horizon objective values. Yet even the richer plant, evaluated with a short prediction horizon, remains far from its own open-loop bound—while still outperforming the simpler architecture operating fully open loop. A more sophisticated plant design can partially compensate for limited prediction capability, but the controller's information horizon remains a first-order factor, not a detail to optimize last.
- **Disturbance frequency amplifies the OLOC/MPC gap.** At low road-disturbance frequencies (around 2 Hz), OLOC- and MPC-based co-design produce objective values within about 4-5% of each other and nearly identical optimized plant designs. As the disturbance frequency rises toward 20 Hz, the objective-value gap exceeds 18% and the relative difference between the two optimized plant designs grows to roughly 650%. Faster disturbance content relative to the controller's update rate makes the prediction-horizon and sampling-time choices matter far more.

Comparison

Approach	Main decision	Strength	Limitation
Open-loop optimal control	Full trajectory $\mathbf{u}(t)$	Flexible benchmark	May need unrealistic future information
Controller-parameter optimization	Gain or coefficient vector $\mathbf{x}_c$	Directly implementable	Limited by chosen structure
Feedback-controller CCD	Plant and controller parameters	Realistic closed-loop design	More restrictive than free trajectories
MPC	Repeated finite-horizon plans	Feedback with constraints and prediction	Requires online optimization and a good model

Common mistakes

1. **Confusing a trajectory with a controller.** A planned signal is not a feedback law.
2. **Treating OLOC as automatically implementable.** It may rely on impossible information.
3. **Ignoring the real information set.** Controllers use only measured, estimated, or credibly forecast quantities.
4. **Assuming feedback removes all uncertainty.** Feedback mitigates but does not eliminate noise and mismatch.
5. **Calling MPC purely open loop.** Repeated measurement-based replanning creates feedback.
6. **Assuming parameterized controllers are always inferior.** A realizable feedback policy can outperform a nonimplementable ideal signal on the uncertain real system.
7. **Treating the MPC prediction horizon as fixed.** Prediction horizon and control sampling time can be optimized alongside plant variables in a nested CCD loop; treating them as fixed implementation details after the fact overlooks a first-order design choice.

Chapter Summary and Problems

Summary

OLOC directly optimizes a control trajectory and is valuable for ideal performance benchmarks, qualitative insight, and plant-design studies. It can also assume future information unavailable to a real controller. Controller-parameter optimization instead designs an implementable feedback policy. Feedback-controller CCD combines physical and policy decisions and evaluates them under measurements and realized disturbances. MPC bridges these views by repeatedly solving a short open-loop problem using updated information.

Key terms

Open-loop optimal control; control trajectory; controller parameterization; feedback controller; feedback-controller co-design; future information; information availability; measured state; estimated state; forecast; implementable controller; model predictive control; receding horizon; prediction horizon; benchmark solution; realizability; complete horizon; instantaneous information; limited horizon; digital controller realization; prediction horizon as a design variable; nested CCD; covariance matrix adaptation evolution strategy (CMA-ES); hybrid Kalman filter.

Problems

1. **Pontryagin analysis with a plant variable.** Minimize $J = \frac{1}{2}q_f x(T)^2 + \int_0^T \frac{1}{2}(qx^2 + ru^2 + \gamma p^2)dt$ subject to $\dot{x} = -px + u$, $x(0) = x_0$, and $p \in [p_-, p_+]$. Derive the costate, optimal control, boundary conditions, and stationarity condition determining an interior optimal p.
2. **LQR control co-design.** For $\dot{x} = A(p)x + B(p)u$ and infinite-horizon cost $\int_0^\infty (x^T Q(p)x + u^T Ru)dt + C(p)$, derive a gradient of the optimized value with respect to p using differentiated Riccati and Lyapunov equations without differentiating the feedback gain explicitly.
3. **Static output-feedback co-design.** The controller is $u = -Ky$ with $y = C(s)x$, where sensor placement s changes C. Formulate an H_2 plant–sensor–controller co-design problem and derive first-order stationarity conditions using the closed-loop controllability and observability Gramians.
4. **Stabilizing model predictive control.** For $x_{k+1} = A(p)x_k + B(p)u_k$ with polyhedral state and input constraints, formulate a finite-horizon MPC law with terminal cost and terminal set, then derive a sufficient decrease inequality proving recursive feasibility and asymptotic stability for every admissible plant design.
5. **Value of preview.** A disturbance-preview controller knows $w_{k:k+N_p}$ for $x_{k+1} = Ax_k + Bu_k + Ew_k$. Derive the finite-horizon optimal control law and an explicit expression for the reduction in quadratic cost relative to the no-preview law as a function of N_p.
6. **Information-horizon monotonicity.** Let $J_N^*(p)$ be the optimal expected cost when a controller has an N-step information horizon. Prove that $J_{N+1}^*(p) \leq J_N^*(p)$ under nested policy classes and construct a counterexample showing that the corresponding optimal plant design need not vary monotonically with N.
7. **Open-loop to closed-loop performance gap.** For $\dot{x} = ax + bu + w$ with bounded disturbance $|w| \leq \bar{w}$, derive the worst-case tracking-cost gap between a nominal open-loop control optimized for $w = 0$ and a stabilizing linear feedback law, including actuator saturation in the bound.
8. **Estimator-aware CCD.** For $\dot{x} = A(p)x + Bu + Gw$ and $y = C(s)x + v$, formulate a joint plant–sensor–LQG design and derive the coupled Riccati sensitivities showing how plant and sensor decisions affect both regulation and estimation cost.
9. **Actuator sizing under saturation.** The scalar system $\dot{x} = -ax + bu$ uses $u = \mathrm{sat}(-Kx, U)$, and actuator mass is $m_a(U) = m_0 + \alpha U$. Determine the globally optimal rating U for a prescribed initial-state distribu-

tion and quadratic regulation objective by deriving a one-dimensional optimality condition that accounts for switching between saturated and unsaturated arcs.

10. **Robust closed-loop CCD.** The uncertain plant is $x_{k+1} = (A(p)+H\Delta E)x_k+B(p)u_k$ with $\Delta^T\Delta \preceq I$. Formulate a min–max state-feedback CCD problem and derive an LMI-based sufficient condition that jointly certifies robust stability, an upper bound on worst-case quadratic cost, and admissible plant bounds.

References and further reading

1. Betts, J. T. (2010). *Practical methods for optimal control and estimation using nonlinear programming* (2nd ed.). Society for Industrial and Applied Mathematics. DOI: 10.1137/1.9780898718577
2. Bryson, A. E., Jr., & Ho, Y.-C. (1975). *Applied optimal control: Optimization, estimation, and control*. Hemisphere Publishing Corporation.
3. Allison, J. T., & Herber, D. R. (2014). Multidisciplinary design optimization of dynamic engineering systems. *AIAA Journal, 52*(4), 691–710. DOI: 10.2514/1.J052182
4. Herber, D. R., & Allison, J. T. (2019). Nested and simultaneous solution strategies for general combined plant and control design problems. *Journal of Mechanical Design, 141*(1), Article 011402. DOI: 10.1115/1.4040705
5. Garcia-Sanz, M. (2019). Control co-design: An engineering game changer. *Advanced Control for Applications: Engineering and Industrial Systems, 1*(1), Article e18. DOI: 10.1002/adc2.18
6. Mayne, D. Q., Rawlings, J. B., Rao, C. V., & Scokaert, P. O. M. (2000). Constrained model predictive control: Stability and optimality. *Automatica, 36*(6), 789–814. DOI: 10.1016/S0005-1098(99)00214-9
7. Deshmukh, A. P., Herber, D. R., & Allison, J. T. (2015). Bridging the gap between open-loop and closed-loop control in co-design: A framework for complete optimal plant and control architecture design. In *2015 American Control Conference (ACC)* (pp. 4916-4922).
8. Bayat, S., & Allison, J. T. (2026). Control co-design with varying available information applied to vehicle suspensions. *ASME Journal of Dynamic Systems, Measurement, and Control, 148*(1), Article 011013. DOI: 10.1115/1.4069918

Numerical Methods for Control Co-Design

Chapter 7: Numerical Methods for Control Co-Design

Discretization, transcription, sparse optimization, derivatives, and verification

> The numerical method, mesh, scaling, derivatives, and verification procedure can change the design reported as optimal.

CCD problems are often infinite-dimensional because states and controls are functions of time. Computers cannot optimize arbitrary continuous functions directly. Numerical methods replace them with finite variables and algebraic constraints, producing a nonlinear program (NLP).

This chapter develops time discretization, direct and multiple shooting, direct transcription, collocation and defect constraints, sparse NLP structure, derivative computation, scaling, initialization, mesh refinement, and numerical verification.

Learning objectives

After completing this chapter, you should be able to:

1. convert a continuous CCD problem into a finite NLP;
2. construct and assess a computational mesh;
3. compare direct shooting, multiple shooting, and transcription;
4. write Euler, trapezoidal, and Hermite–Simpson defects;
5. explain direct-transcription sparsity;
6. compare numerical and exact derivative methods;
7. perform mesh-refinement and independent-simulation checks; and
8. organize a reliable CCD implementation workflow.

Chapter roadmap

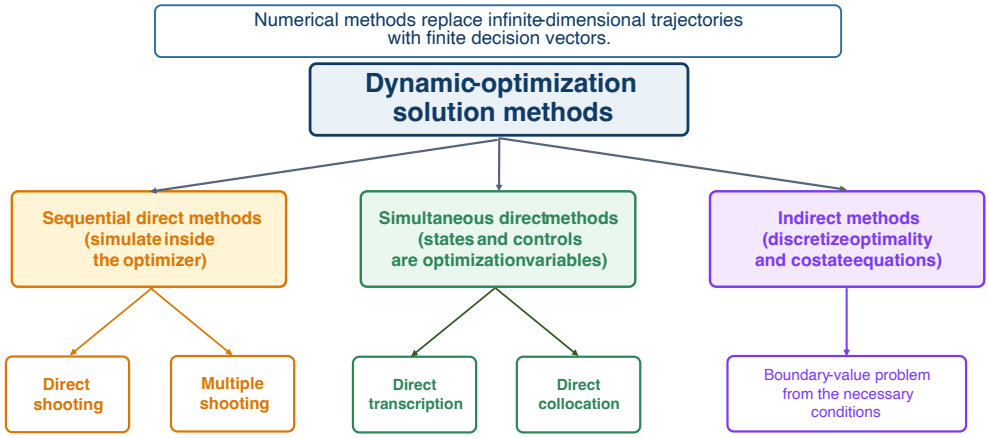
Continuous problem → Mesh → Shooting or transcription → Sparse NLP → Derivatives → Refinement and verification

From Continuous CCD to a Computable Problem

A continuous finite-horizon CCD formulation has finite plant and controller parameters but functional state and control decisions. Numerical approximation produces

$$\begin{aligned} \underset{\mathbf{z}\in\mathbb{R}^{n_z}}{\text{minimize}} \quad & F(\mathbf{z}) \\ \text{subject to} \quad & \mathbf{g}_e(\mathbf{z}) = \mathbf{0}, \\ & \mathbf{g}_i(\mathbf{z}) \le \mathbf{0}, \\ & \mathbf{z}^L \le \mathbf{z} \le \mathbf{z}^U, \end{aligned}$$

where $\mathbf{z}$ includes sampled states and controls plus plant and controller variables.



The numerical-method landscape for dynamic optimization.

Indirect methods derive costate, stationarity, and transversality conditions and solve a boundary-value problem. They can be accurate and theoretically revealing, but costate initialization and active path constraints are difficult.

Direct methods parameterize controls, states, or both and solve an NLP. They are usually more robust for engineering problems with bounds, path constraints, black-box components, and plant–control coupling.

In **sequential direct methods**, a simulator computes states from optimized controls and parameters. In **simultaneous direct methods**, state samples are also decision variables and algebraic defects enforce the dynamics.

Terminology warning

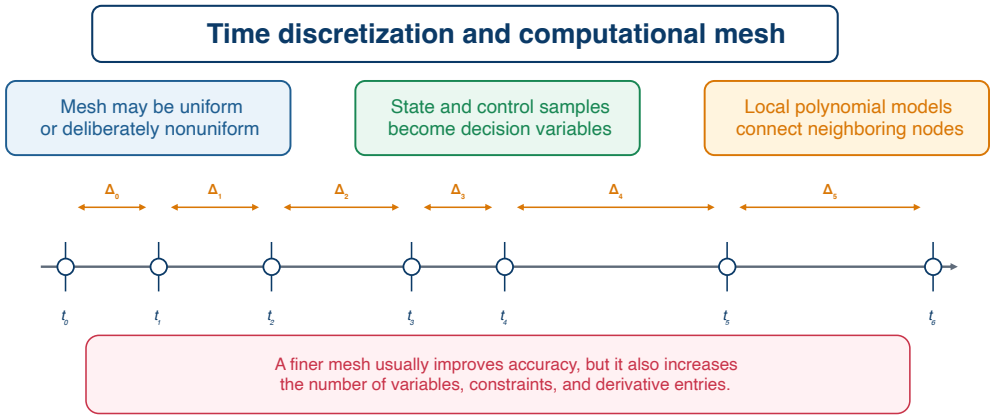
Sequential and simultaneous numerical methods describe how dynamic equations are solved. They are different from sequential and simultaneous CCD architectures, which describe how plant and controller decisions are coordinated.

Time discretization

Choose a mesh

$$t_0 < t_1 < \cdots < t_N = t_f, \qquad h_k = t_{k+1} - t_k,$$

and define $\mathbf{x}_k \approx \mathbf{x}(t_k)$ and $\mathbf{u}_k \approx \mathbf{u}(t_k)$.



The mesh converts continuous trajectories into finite samples and local approximations.

For n_x states and n_u controls, node variables contribute approximately $(N+1)(n_x+n_u)$ scalars.

Uniform meshes suit smooth behavior. Nonuniform meshes place points near transients, switching, active-set changes, impacts, or boundary layers. The mesh must resolve the physics, not merely divide the horizon conveniently.

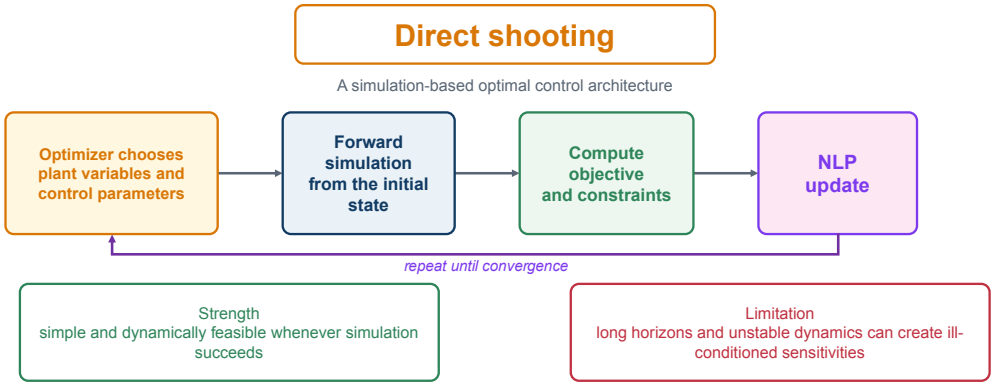
Local representations use piecewise-constant, piecewise-linear, trapezoidal, or Hermite–Simpson polynomials. Global or pseudospectral methods use high-order Legendre or Chebyshev approximations. Local methods refine naturally near nonsmooth behavior; global methods can converge very rapidly for smooth solutions.

Common control representations are piecewise constant, piecewise linear, and basis expansions. The choice should match actuator physics and the selected integration scheme.

Direct and Multiple Shooting

Direct shooting

Parameterize control as $\mathbf{u}(t;\mathbf{q})$. The optimizer selects $\mathbf{q}$ and design variables; a forward integrator simulates the dynamics and returns objective and constraint values.



Direct shooting embeds a full forward simulation in every optimization evaluation.

Advantages include simple simulator integration, automatically integrated states, relatively few NLP variables, and reuse of mature ODE/DAE solvers.

Its main weakness is long-range sensitivity. If

$$\mathbf{S}(t) = \frac{\partial \mathbf{x}(t)}{\partial \mathbf{q}},$$

then

$$\dot{\mathbf{S}} = \mathbf{f}_x \mathbf{S} + \mathbf{f}_u \frac{\partial \mathbf{u}}{\partial \mathbf{q}}, \qquad \mathbf{S}(t_0) = \mathbf{0}.$$

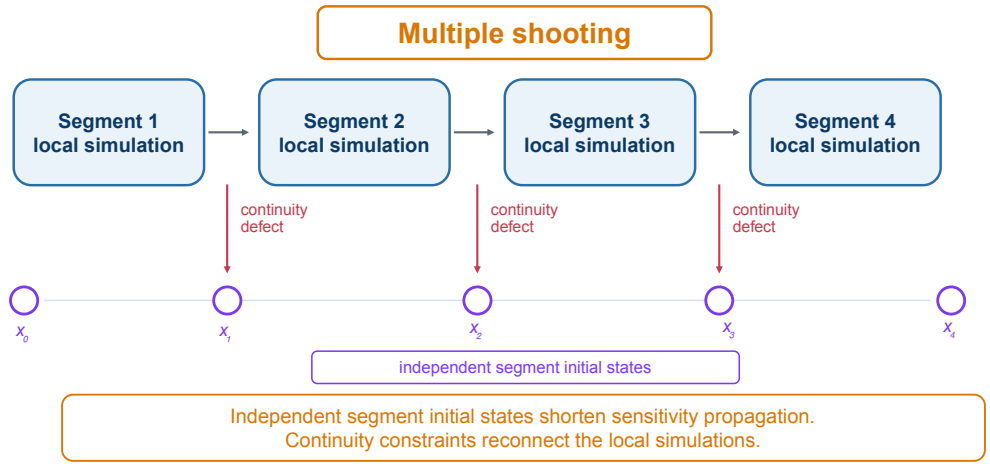
For unstable or long-horizon systems, sensitivities may grow or decay severely, making terminal and path constraints difficult. Shooting is most suitable for stable systems, short horizons, low-dimensional controls, and moderate terminal requirements.

Numerical warning

Accurate forward simulation does not imply a well-conditioned optimization problem.

Multiple shooting

Multiple shooting splits the horizon into segments with independent initial-state decisions $\mathbf{X}_k$.



Short local simulations limit sensitivity propagation.

Let $\mathbf{\Phi}_k(\mathbf{X}_k, \mathbf{q}_k, \mathbf{x}_p, \mathbf{x}_c)$ be the simulated endpoint of segment k. Continuity requires

$$\mathbf{r}_k = \mathbf{X}_{k+1} - \mathbf{\Phi}_k(\mathbf{X}_k, \mathbf{q}_k, \mathbf{x}_p, \mathbf{x}_c) = \mathbf{0}.$$

Multiple shooting handles unstable dynamics and tight terminal constraints better, permits internal state guesses, and produces sparse continuity Jacobians. It adds state variables, constraints, and several local simulations. It lies between single shooting and direct transcription: integration remains inside each segment, while algebraic continuity reconnects the horizon.

Activity 7.1: Multiple Shooting versus Direct Collocation for an Unstable System

Activity 7.1: Multiple Shooting versus Direct Collocation for an Unstable System

Consider

$$\dot{\mathbf{x}} = \begin{bmatrix} 0 & 1 \\ 4 & 0 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u,$$

with

$$\mathbf{x}(0) = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \qquad \mathbf{x}(3) = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

Minimize

$$J = \int_0^3 \left(\mathbf{x}^T \mathbf{x} + 0.05 u^2 \right) dt, \qquad |u| \le 5.$$

1. Derive the exact state-transition equation over one shooting interval:

$$\mathbf{x}_{i+1} = e^{A\Delta t} \mathbf{x}_i + \int_0^{\Delta t} e^{A(\Delta t - \tau)} B \, u_i \, d\tau.$$

2. Formulate a 10-segment multiple-shooting nonlinear program with piecewise-constant control.
3. Formulate a Hermite–Simpson transcription using 30 intervals.
4. Solve both formulations.

5. Compare Jacobian sparsity, condition number, optimizer iterations, and sensitivity to initial guesses.
6. Explain why direct single shooting performs poorly for this unstable system.

Activity 7.2: Hypersensitive Optimal-Control Problem

Activity 7.2: Hypersensitive Optimal-Control Problem

Consider

$$\dot{x} = -x^3 + u,$$

with

$$x(0) = 1, \qquad x(T) = 1.5, \qquad T = 50.$$

Minimize

$$J = \frac{1}{2}\int_0^T \left(x^2 + u^2\right)\, dt.$$

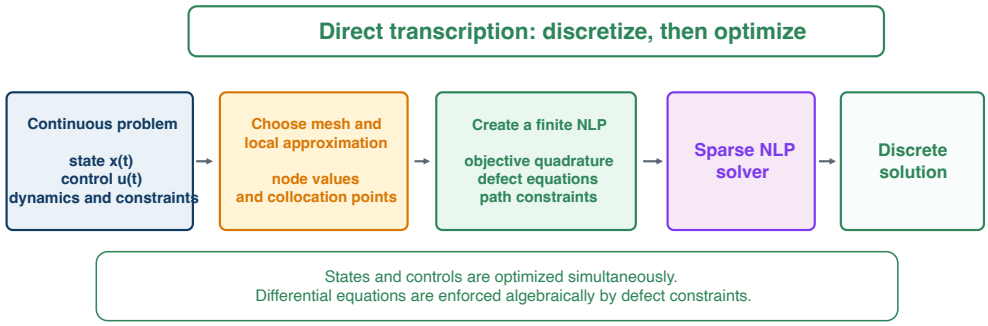
1. Solve the problem using direct single shooting.
2. Solve it using multiple shooting with 5, 10, and 20 segments.
3. Solve it using GPOPS-II or Dymos with an adaptive collocation mesh.
4. Plot the state and control near:
   (a) the initial boundary layer;
   (b) the long interior arc; and
   (c) the terminal boundary layer.
5. Compare the methods in terms of:
   (a) convergence;
   (b) sensitivity to the initial guess;
   (c) Jacobian conditioning;
   (d) mesh size; and
   (e) computation time.
6. Independently integrate the collocation solution and report

$$e_x = \max_t \left|x_{\text{collocation}}(t) - x_{\text{verification}}(t)\right|.$$

7. Explain why this problem is difficult for single shooting even though it contains only one state.

Direct Transcription

Direct transcription treats state and control samples as optimization variables and replaces differential equations with algebraic defects.



Discretize first, then optimize all trajectory and design variables together.

A time-ordered decision vector may be

$$\mathbf{z} = [\mathbf{x}_p, \mathbf{x}_c, \mathbf{x}_0, \mathbf{u}_0, \ldots, \mathbf{x}_N, \mathbf{u}_N]^T.$$

A generic transcription is

$$\begin{aligned} \underset{\mathbf{z}}{\text{minimize}} \quad & \widehat{\Phi}(\mathbf{z}) + \sum_{k=0}^{N} w_k L(\mathbf{x}_k, \mathbf{u}_k, \mathbf{x}_p, \mathbf{x}_c, t_k) \\ \text{subject to} \quad & \boldsymbol{\zeta}_k(\mathbf{x}_k, \mathbf{u}_k, \mathbf{x}_{k+1}, \mathbf{u}_{k+1}, \mathbf{x}_p, \mathbf{x}_c) = \mathbf{0}, \\ & \mathbf{c}(\mathbf{x}_k, \mathbf{u}_k, \mathbf{x}_p, \mathbf{x}_c, t_k) \leq \mathbf{0}, \\ & \mathbf{b}(\mathbf{x}_0, \mathbf{x}_N, \mathbf{x}_p, \mathbf{x}_c) = \mathbf{0}, \\ & \mathbf{z}^L \leq \mathbf{z} \leq \mathbf{z}^U. \end{aligned}$$

Direct transcription is valuable for CCD because it handles nonlinear plant-dependent dynamics, unstable systems, strong coupling, state/control path limits, and terminal constraints while exposing local sparsity.

Activity 7.3: Plant and Actuator Co-Design of a Dynamic System

Activity 7.3: Plant and Actuator Co-Design of a Dynamic System

Consider

$$\dot{x}_1 = x_2, \qquad m\dot{x}_2 = -kx_1 - cx_2 + u,$$

where $m = 1$ and $c = 0.5$. The spring stiffness k and actuator capacity $F_{\max}$ are plant-design variables:

$$0.5 \leq k \leq 8, \qquad 0.2 \leq F_{\max} \leq 5.$$

The control satisfies

$$|u(t)| \leq F_{\max}.$$

Use

$$\mathbf{x}(0) = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \qquad \mathbf{x}(t_f) = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \qquad t_f = 4.$$

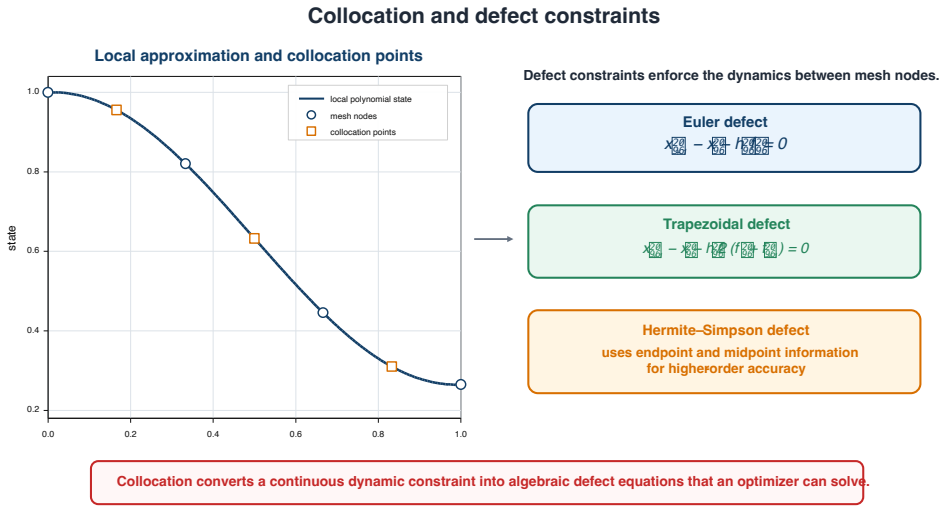
Minimize

$$J = \int_0^{t_f} \left(x_1^2 + 0.1 x_2^2 + 0.01 u^2 \right) \, dt + 0.03 F_{\max}^2 + 0.002 k^2.$$

1. Nondimensionalize the states, control, time, and design variables.
2. Formulate the simultaneous direct-collocation problem.
3. Solve the problem using GPOPS-II or Dymos.
4. Determine whether the actuator constraint is active.
5. Repeat the solution after multiplying the $F_{\max}^2$ penalty by 0.1, 1, and 10.
6. Explain how the actuator-cost weight changes both k^* and $F_{\max}^*$.

Collocation and Defect Constraints

A **defect** measures disagreement between discrete state variables and the state change predicted by the dynamics. Zero defect means local dynamic consistency under the chosen approximation.



Higher-order schemes use more information within each interval.

Euler forward

$$\zeta_k^{\mathrm{EF}} = \mathbf{x}_{k+1} - \mathbf{x}_k - h_k \mathbf{f}_k = \mathbf{0}.$$

Euler forward is globally first order and typically needs a fine mesh.

Trapezoidal rule

$$\zeta_k^{\mathrm{TR}} = \mathbf{x}_{k+1} - \mathbf{x}_k - \frac{h_k}{2}(\mathbf{f}_k + \mathbf{f}_{k+1}) = \mathbf{0}.$$

It is globally second order and implicit. Implicitness is natural in transcription because the NLP solver determines states and defects together.

Hermite–Simpson

Define

$$\mathbf{x}_{k+1/2} = \frac{\mathbf{x}_k + \mathbf{x}_{k+1}}{2} + \frac{h_k}{8}(\mathbf{f}_k - \mathbf{f}_{k+1}), \qquad \mathbf{u}_{k+1/2} = \frac{\mathbf{u}_k + \mathbf{u}_{k+1}}{2}.$$

Then

$$\zeta_k^{\mathrm{HS}} = \mathbf{x}_{k+1} - \mathbf{x}_k - \frac{h_k}{6}(\mathbf{f}_k + 4\mathbf{f}_{k+1/2} + \mathbf{f}_{k+1}) = \mathbf{0}.$$

Hermite–Simpson offers high accuracy, local sparsity, and moderate implementation complexity.

Zero-order-hold and higher-order single-step defects

For linear time-invariant dynamics $\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u}$ with piecewise-constant control on an interval, the defect can be made exact rather than approximate using the matrix exponential:

$$\zeta_k^{\mathrm{ZOH}} = \mathbf{x}_{k+1} - e^{Ah_k}\mathbf{x}_k - \left(\int_0^{h_k} e^{A(h_k-\tau)}\,d\tau\right) B\mathbf{u}_k = \mathbf{0}.$$

This zero-order-hold defect introduces no truncation error for the assumed control interpolation, at the cost of a matrix exponential per interval and the loss of exactness once the dynamics become nonlinear or plant-parameter-dependent. Runge–Kutta families such as the classical fourth-order method extend the single-step idea to nonlinear dynamics by combining several interior stage evaluations per interval, trading additional function evaluations for higher order than Euler forward or the trapezoidal rule.

Pseudospectral collocation and the differentiation matrix

Pseudospectral methods approximate the state on an interval with a single global Lagrange interpolating polynomial through N collocation points and enforce the dynamics at those points using a **differentiation matrix** D, where differentiating the interpolant and evaluating at the collocation points gives $\dot{\mathbf{x}}(\tau_k) \approx \sum_j D_{kj}\mathbf{x}_j$. Three families of collocation points are in common use, each built from roots of Legendre polynomials: **Legendre–Gauss (LG)** points (both endpoints excluded), **Legendre–Gauss–Radau (LGR)** points (one endpoint included), and **Legendre–Gauss–Lobatto (LGL)** points (both endpoints included).

The three schemes are not merely cosmetic variations. The LG and LGR differentiation matrices are rectangular and full rank, so the collocated dynamics $D\mathbf{X} = \mathbf{F}$ can be rewritten equivalently in an integral form using a corresponding integration matrix — the differential and integral transcriptions agree exactly. The LGL differentiation matrix, in contrast, is square but **singular**.

This rank deficiency is not just a numerical curiosity: it propagates into the discrete costate (adjoint) system obtained from the Karush–Kuhn–Tucker conditions of the collocated NLP. Because the transformed adjoint systems for LG and LGR collocation inherit the full-rank differentiation matrix, they are themselves full rank and determine a unique discrete costate. The LGL transformed adjoint system inherits the singular differentiation matrix, so its costate equations possess a nontrivial null space; the recovered LGL costate approximation can oscillate around the true continuous costate even while the state and control trajectories converge normally. This defect is specific to LGL collocation — it does not occur with LG or LGR — and has been demonstrated by direct comparison against a known analytic costate on a benchmark problem.

Practical implication

If a CCD study needs a trustworthy costate or adjoint sensitivity (for example, to cross-check stationarity against an indirect formulation), prefer LG or LGR collocation over LGL. An oscillatory LGL costate does not necessarily mean the state and control solution is wrong — but it does mean the costate cannot be read off directly.

They can converge very rapidly for smooth trajectories but need multiple intervals or adaptation near discontinuities, and production transcription software typically varies both the number of intervals and the polynomial degree per interval to reach a target accuracy (see the next section).

Feature	Local collocation	Global/pseudospectral
Approximation	Low order per interval	High order over large intervals
Sparsity	Strong local bands	Denser within each polynomial interval
Smooth solutions	Algebraic mesh convergence	Potentially spectral convergence
Nonsmooth solutions	Local refinement	Multiple intervals or adaptation
Implementation	Relatively direct	Specialized matrices and quadrature

Quadrature, Path Constraints, and Sparse NLPs

Objective quadrature

The running cost must be discretized consistently with the trajectory approximation:

$$\text{rectangle:} \quad \int_{t_k}^{t_{k+1}} L\,dt \approx h_k L_k,$$

$$\text{trapezoidal:} \quad \int_{t_k}^{t_{k+1}} L\,dt \approx \frac{h_k}{2}(L_k + L_{k+1}),$$

$$\text{Simpson:} \quad \int_{t_k}^{t_{k+1}} L\,dt \approx \frac{h_k}{6}(L_k + 4L_{k+1/2} + L_{k+1}).$$

A low-order quadrature rule can limit the accuracy of an otherwise high-order transcription.

Path constraints between nodes

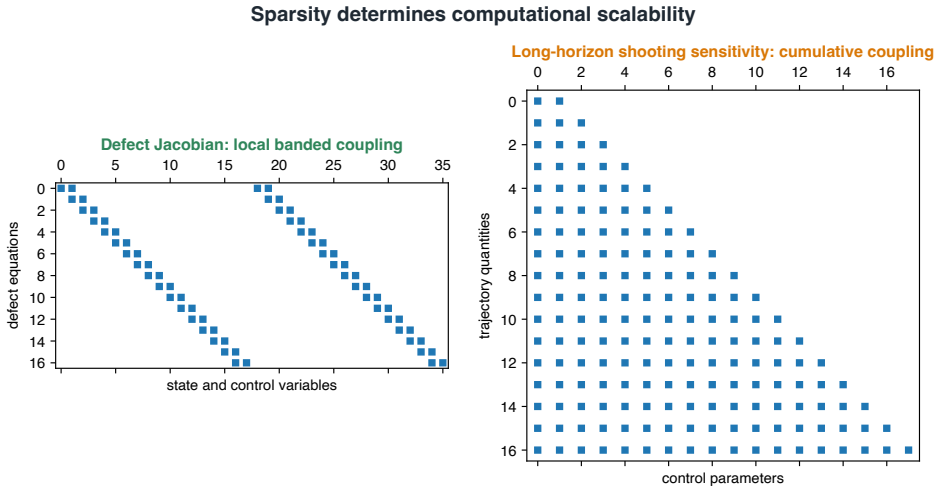
Imposing $\mathbf{c}(\mathbf{x}_k, \mathbf{u}_k, \ldots) \leq \mathbf{0}$ only at nodes does not guarantee continuous feasibility. Reconstructed trajectories can overshoot. Remedies include dense-grid verification, extra enforcement points, margins, local refinement, and interpolation-error estimates.

Boundary constraints depend on selected endpoint nodes and do not create the same between-node issue.

Why large NLPs remain manageable

A problem with 30 states, 10 controls, and 500 intervals has about 20,040 trajectory variables. Dimension alone is not decisive; **sparsity** is.

A local defect depends mainly on $\mathbf{x}_k$, $\mathbf{u}_k$, $\mathbf{x}_{k+1}$, $\mathbf{u}_{k+1}$, and global design variables. Its Jacobian touches only a small portion of the decision vector.



Local time dependence produces exploitable sparse structure.

Sparse SQP and interior-point solvers benefit from exact sparsity patterns, sparse derivatives, scaled variables and constraints, and bounds separated from general constraints.

Decision-vector ordering changes fill-in in the solver's linear systems. Condensing can eliminate states, reducing dimension but potentially destroying sparsity. Partial condensing balances these effects.

Where plant design variables live in the NLP

A production multi-phase transcription typically separates the decision vector into phase-dependent blocks — the state, control, and integral values on the mesh of each phase — and an explicit vector of **static parameters**: time-independent quantities that do not vary along the trajectory. Concretely, a widely used transcription formulation stacks the decision vector as the phase blocks followed by the static parameters appended at the end, with dedicated bounds and constraints on the static-parameter block alone.

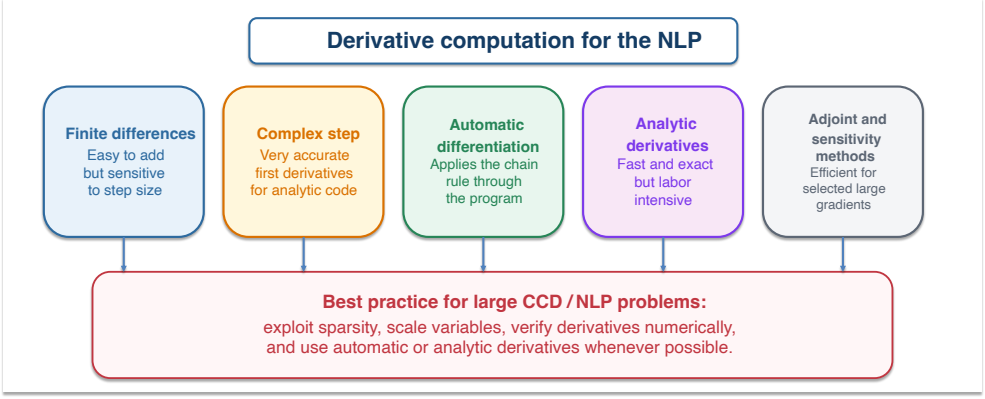
This static-parameter vector is precisely the mechanism through which plant-design variables enter an off-the-shelf transcription code: a plant parameter such as a stiffness, mass, or gain is declared as a static parameter, and the dynamics, path constraints, and objective at every collocation point in every phase are then allowed to depend on it. Because a static parameter can affect the residual at every mesh point, its column in the constraint Jacobian is **dense** even though the state and control columns remain locally banded — the same structural asymmetry noted for the design-parameter column p in the Hermite–Simpson Jacobian activity later in this chapter. The composite defect Jacobian is therefore block-diagonal (or block-banded) in the trajectory variables, with a small number of dense columns appended for the static parameters.

Automated generation of the sparse NLP or QP

For the sizable subclass of CCD subproblems with linear dynamics and a quadratic objective — the case that recurs whenever an inner-loop or nested CCD architecture repeatedly re-solves a control subproblem for a fixed plant design — every matrix in the transcribed problem is a constant (state-, control-, and parameter-independent) sparse matrix. In that case the sparse Hessian and constraint Jacobian blocks can be assembled directly from the problem's symbolic structure, without any finite-differencing or automatic differentiation: each objective term, defect constraint, or path constraint contributes a known sparse block (banded for single-step defects such as Euler forward, trapezoidal, or Hermite–Simpson; block-structured for pseudospectral defects built from a differentiation matrix), and an automated procedure can assemble these blocks by row/column bookkeeping alone. This is attractive for nested CCD, where the inner control-design QP is solved many times per outer plant-design iteration and assembly overhead matters.

Derivative Computation

Large CCD problems typically require gradient-based optimization: objective gradients, constraint Jacobians, and sometimes Lagrangian Hessians.



Major derivative-computation options for nonlinear programming.

Finite differences

$$\frac{\partial F}{\partial z_i} \approx \frac{F(\mathbf{z} + h\mathbf{e}_i) - F(\mathbf{z})}{h}$$

is easy to implement but balances truncation against cancellation and may require one evaluation per column. Sparse coloring perturbs structurally independent columns together.

Before coloring can be applied, the solver needs to know which functions depend on which variables. One practical dependency-detection technique evaluates each constraint or objective function once with a single input component set to `NaN`; IEEE floating-point arithmetic propagates `NaN` through every expression that actually depends on that component, so a single perturbed evaluation reveals an entire row of the first-derivative sparsity pattern. Repeating this for each input component (or exploiting problem structure to batch several at once) yields the full Jacobian sparsity pattern with a small number of extra function evaluations; squaring the resulting $0/1$ pattern gives a conservative estimate of the Hessian sparsity. This dependency map is what lets sparse finite-differencing compute only the structurally nonzero derivative entries, and it is what makes a single transcription implementation usable with both quasi-Newton NLP solvers, which need only the gradient and constraint Jacobian, and full-Newton solvers, which additionally need the sparse Lagrangian Hessian.

Complex-step differentiation

For analytic code,

$$\frac{\partial F}{\partial z_i} \approx \frac{\operatorname{Im} F(\mathbf{z} + ih\mathbf{e}_i)}{h}.$$

It avoids subtractive cancellation and is excellent for verification, provided complex numbers propagate correctly and nonanalytic operations are avoided.

Analytic derivatives

For a trapezoidal defect,

$$\boldsymbol{\zeta}_k = \mathbf{x}_{k+1} - \mathbf{x}_k - \frac{h_k}{2}(\mathbf{f}_k + \mathbf{f}_{k+1}),$$

representative blocks are

$$\frac{\partial \zeta_k}{\partial \mathbf{x}_k} = -I - \frac{h_k}{2}\mathbf{f}_{x,k}, \quad \frac{\partial \zeta_k}{\partial \mathbf{x}_{k+1}} = I - \frac{h_k}{2}\mathbf{f}_{x,k+1},$$

with analogous control blocks. These formulas expose local sparsity directly.

Automatic differentiation and sensitivities

Automatic differentiation applies the chain rule to code and can provide machine-precision sparse Jacobians and Hessians. Forward mode suits few independent variables; reverse mode suits scalar outputs with many inputs.

Shooting derivatives often use direct or adjoint sensitivities. Direct sensitivities propagate one system per parameter; adjoints can be efficient for many parameters and few scalar outputs.

Verification

Compare implemented derivatives against complex step or careful finite differences at random points. A normalized discrepancy is

$$E_D = \frac{\|\mathbf{d}_{\text{implemented}} - \mathbf{d}_{\text{reference}}\|_2}{\max(1, \|\mathbf{d}_{\text{reference}}\|_2)}.$$

Derivative errors can cause slow convergence, false convergence, or convergence to the wrong point.

Activity 7.4: Analytical Hermite–Simpson Jacobian

Activity 7.4: Analytical Hermite–Simpson Jacobian

Consider the nonlinear, parameter-dependent system

$$\dot{x}_1 = x_2, \qquad \dot{x}_2 = -p \sin x_1 + u,$$

where p is a plant-design variable. For interval i, define

$$\mathbf{f}_i = \mathbf{f}(\mathbf{x}_i, u_i, p), \qquad \mathbf{f}_{i+1} = \mathbf{f}(\mathbf{x}_{i+1}, u_{i+1}, p),$$

and

$$\mathbf{x}_{i+1/2} = \frac{\mathbf{x}_i + \mathbf{x}_{i+1}}{2} + \frac{h_i}{8}\left(\mathbf{f}_i - \mathbf{f}_{i+1}\right).$$

The Hermite–Simpson defect is

$$\mathbf{d}_i = \mathbf{x}_{i+1} - \mathbf{x}_i - \frac{h_i}{6}\left(\mathbf{f}_i + 4\mathbf{f}_{i+1/2} + \mathbf{f}_{i+1}\right).$$

1. Derive

$$\frac{\partial \mathbf{d}_i}{\partial \mathbf{x}_i}, \qquad \frac{\partial \mathbf{d}_i}{\partial \mathbf{x}_{i+1}}, \qquad \frac{\partial \mathbf{d}_i}{\partial u_i}, \qquad \frac{\partial \mathbf{d}_i}{\partial u_{i+1}}, \qquad \frac{\partial \mathbf{d}_i}{\partial p}.$$

2. Assemble the Jacobian sparsity pattern for N intervals.

3. Determine the exact number of nonzero entries.
4. Explain why the column associated with p is globally dense while the state and control blocks remain locally sparse.
5. Verify the analytical Jacobian using complex-step differentiation.

Activity 7.5: Derivative Verification for a Shooting Formulation

Activity 7.5: Derivative Verification for a Shooting Formulation

Consider

$$\dot{x} = -x^3 + u, \qquad x(0) = 1, \qquad 0 \le t \le 2.$$

Parameterize $u(t)$ using $N = 20$ piecewise-constant values:

$$\mathbf{U} = \begin{bmatrix} u_0 & u_1 & \cdots & u_{19} \end{bmatrix}^T.$$

Minimize

$$J(\mathbf{U}) = x(2)^2 + 0.05 \int_0^2 u(t)^2 \, dt.$$

1. Derive the forward-sensitivity equations

$$\frac{d}{dt}\left(\frac{\partial x}{\partial u_j}\right) = -3x^2 \frac{\partial x}{\partial u_j} + \frac{\partial u}{\partial u_j}.$$

2. Derive the gradient $\nabla_{\mathbf{U}} J$.
3. Compute the gradient using:
   (a) forward finite differences;
   (b) central finite differences;
   (c) complex-step differentiation; and
   (d) sensitivity equations or automatic differentiation.
4. Perform a directional-derivative test using a random normalized vector $\mathbf{p}$.
5. Plot derivative error for step sizes from 10^{-2} to 10^{-14}.
6. Explain the truncation-error and cancellation-error regions in the resulting plot.

Scaling, Refinement, and Verification

Scaling and initialization

Use scaled variables and residuals,

$$\widehat{z}_i = \frac{z_i - z_i^{\mathrm{ref}}}{s_i}, \qquad \widehat{g}_j = \frac{g_j}{r_j},$$

so typical magnitudes are comparable. Dynamic defects need state-specific scales because position, velocity, temperature, and rotation have different units.

Useful initial guesses come from baseline simulation, simple boundary interpolation, known operating controls, simplified problems, and nearby parameter cases. Continuation gradually tightens constraints, increases disturbances, activates nonlinear physics, reduces regularization, or refines the mesh while warm-starting from the previous solution.

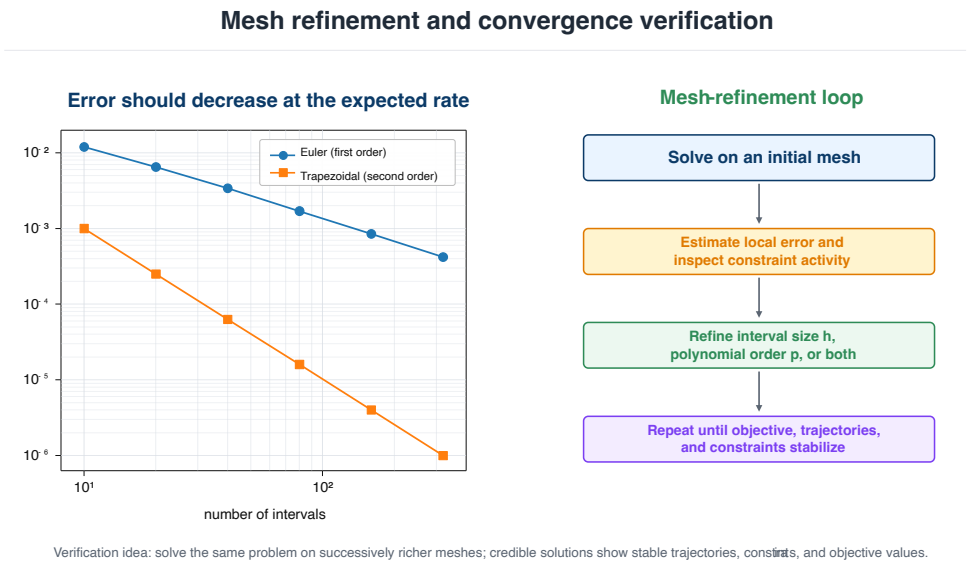
Why scaling changes what "feasible" means

Consider the scalar linear system $\dot{y} = ay$, $y(0) = y_0$, discretized with a trapezoidal defect. Most NLP solvers accept an equality defect once its *absolute* value falls within a solver tolerance ϵ (a common default is $\epsilon = 10^{-6}$). Because the defect is linear in y, the *relative* error that this absolute tolerance permits scales with $1/y_0$: for a state with $y_0 = 10^{-6}$, a defect within ϵ can correspond to a relative error of order unity, while rescaling y to unit initial magnitude drives the same absolute tolerance down to a relative error of order ϵ. Two systems that are mathematically equivalent up to a change of units can therefore behave completely differently under the same solver settings — scaling state, control, and time to be $O(1)$ is what makes the solver's tolerance mean what it appears to mean.

A practical automatic-scaling recipe applies an affine map to each variable so that it lies in a fixed reference interval (for example $[-1/2, 1/2]$), then scales each constraint by approximately the reciprocal of its gradient norm sampled at a few candidate points, so that both the scaled variables and the scaled Jacobian entries are $O(1)$ before the NLP solver ever sees them. Defect constraints inherit the scale factor of the state they enforce, so that dynamic consistency is checked in the same relative terms across states with very different physical units.

Mesh refinement

One mesh is not enough evidence.



Mesh refinement tests whether the reported design is stable under improved approximation.

- h**-refinement:** add intervals, especially near nonsmooth behavior.
- p**-refinement:** increase polynomial degree in smooth regions.
- hp**-refinement:** combine both based on local smoothness.

Error indicators include model-versus-reconstruction derivative mismatch, low- versus high-order estimates, coefficient decay, constraint overshoot, independent-simulation error, and changes in objective and design.

Production pseudospectral codes automate hp-refinement as an explicit decision rule rather than trial and error. One published strategy compares the ratio of maximum to mean curvature of the state solution within a mesh interval against a user-specified threshold: exceeding the threshold increases the polynomial degree in that interval, while staying below it subdivides the interval into more, lower-degree intervals. A related strategy instead exploits the (near-)exponential convergence of orthogonal collocation directly: it estimates the polynomial degree needed to reach the target accuracy within an interval, raises the degree toward a user-specified upper limit $N_{\max}$ if that estimate is attainable, and otherwise subdivides. Software exposing this kind of combined interval-count/degree adaptation commonly labels the configuration hp-$(N_{\min}, N_{\max})$, naming the allowed minimum and maximum polynomial degree per mesh interval. Either rule automates exactly the "add intervals near nonsmooth behavior, add degree in smooth regions" heuristic above, driven by a re-solve-and-re-estimate loop around the current mesh.

Monitor several convergence quantities:

$$E_J^{(r)} = \frac{|J^{(r)} - J^{(r-1)}|}{\max(1, |J^{(r)}|)}, \qquad E_z^{(r)} = \frac{\|\mathbf{z}_d^{(r)} - \mathbf{z}_d^{(r-1)}\|_2}{\max(1, \|\mathbf{z}_d^{(r)}\|_2)},$$

along with maximum defect and path-constraint violation.

Independent simulation

Reconstruct the optimized control and simulate the original dynamics with a separate high-accuracy integrator. Compare trajectories, terminal values, objectives, and constraints. This reveals insufficient meshes, incorrect defects, inconsistent control interpolation, and hidden violations.

Inspect primal and dual infeasibility, stationarity, complementarity, step acceptance, regularization, and restoration warnings. A solver can terminate because it cannot progress—not because it found a credible optimum.

Verification warning

Small defects at mesh points do not guarantee small continuous-time error. Check reconstructed trajectories between nodes.

Activity 7.6: Minimum-Time Control with an Active State Constraint

Activity 7.6: Minimum-Time Control with an Active State Constraint

Consider the double integrator

$$\dot{x}_1 = x_2, \qquad \dot{x}_2 = u,$$

with boundary conditions

$$\mathbf{x}(0) = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \qquad \mathbf{x}(t_f) = \begin{bmatrix} 1 \\ 0 \end{bmatrix},$$

and constraints

$$-1 \le u(t) \le 1, \qquad |x_2(t)| \le 0.6.$$

Minimize

$$J = t_f.$$

1. Derive the optimal-control structure and identify all bang and state-constrained arcs.
2. Compute the switching times and optimal final time analytically.
3. Solve the problem using GPOPS-II or Dymos.
4. Plot $x_1(t)$, $x_2(t)$, and $u(t)$, and mark every switching point.
5. Repeat the numerical solution using 20, 40, and 80 mesh intervals, and report the error in t_f.
6. Verify the final solution using independent high-accuracy integration.

Example and Implementation Workflow

Mass–spring–damper direct transcription

Consider

$$m\ddot{q} + c\dot{q} + kq = u(t) + d(t),$$

with plant decisions k, c, state $\mathbf{x} = [q\ v]^T$, and

$$\dot{\mathbf{x}} = \begin{bmatrix} v \\ -kq/m - cv/m + (u+d)/m \end{bmatrix}.$$

Use

$$J = \int_0^T (w_q q^2 + w_v v^2 + w_u u^2)dt + w_k k + w_c c$$

with bounds on q, u, k, and c.

The decision vector is

$$\mathbf{z} = [k, c, q_0, v_0, u_0, \ldots, q_N, v_N, u_N]^T.$$

Trapezoidal defects are

$$0 = q_{k+1} - q_k - \frac{h_k}{2}(v_k + v_{k+1}),$$

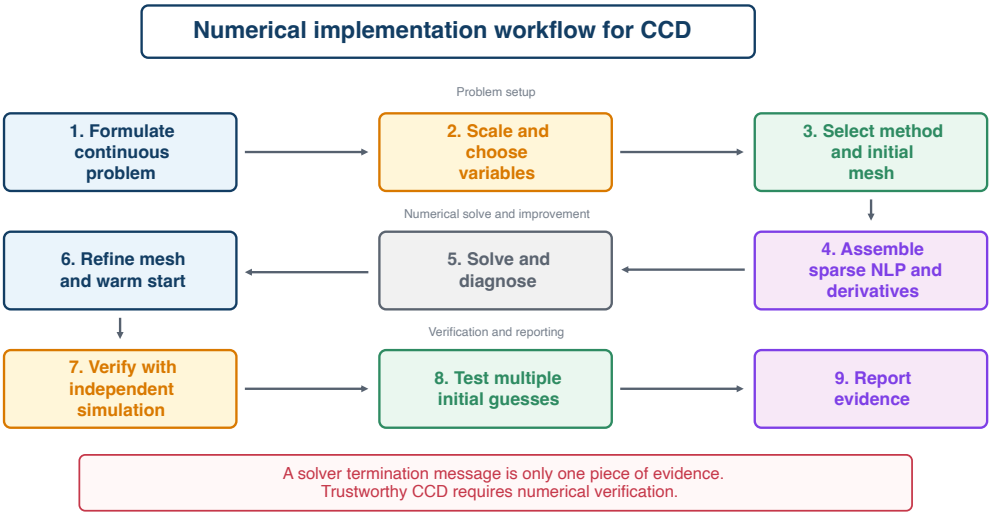
$$0 = v_{k+1} - v_k - \frac{h_k}{2}(f_{v,k} + f_{v,k+1}).$$

The second is nonlinear because plant decisions multiply state variables. Use trapezoidal objective quadrature, node bounds, and dense between-node checks.

Note the structure this creates for a nested CCD architecture: for any *fixed* plant design (k, c), the dynamics are linear and the objective is quadratic in the remaining state and control variables, so the inner control-design transcription is a linear-quadratic dynamic-optimization problem whose sparse Hessian and defect Jacobian can be assembled directly from constant matrices rather than by finite-differencing or automatic differentiation at every inner solve. Only the outer sweep over (k, c) reintroduces nonlinearity. Recognizing this fixed-plant/variable-control structure is often what makes repeated inner-loop transcription affordable inside a nested plant search.

A credible study uses at least three meshes, design and objective convergence, dense constraint checks, independent simulation, derivative verification, and multiple initial guesses.

Numerical workflow



The loop from refinement back to formulation and method selection is intentional.

1. Formulate the continuous problem.
2. Scale variables, objective terms, and residuals.
3. Select shooting, multiple shooting, or transcription.
4. Choose the mesh, polynomial order, control interpolation, and quadrature.
5. Assemble sparse functions and verified derivatives.
6. Solve and inspect feasibility, stationarity, and warnings.
7. Refine and warm start.
8. Verify with independent simulation and dense constraint checks.
9. Test nonconvexity using multiple starts and continuation.
10. Report method, mesh, scaling, solver, tolerances, derivatives, initialization, convergence, residuals, and verification.

Method selection

Method	Best suited for	Main strength	Main limitation
Direct shooting	Stable systems, short horizons	Simple simulator connection	Long-horizon conditioning
Multiple shooting	Unstable systems, terminal constraints	Short sensitivity propagation	Added states and continuity equations
Local collocation	General nonlinear path-constrained CCD	Sparse and locally refinable	Large NLP and trajectory initialization
Pseudospectral	Smooth high-accuracy trajectories	Rapid smooth convergence	Awkward near discontinuities
Indirect	Known active sets and theoretical analysis	High accuracy and costates	Difficult initialization and constraints

Chapter Summary and Problems

Summary

Shooting optimizes control parameters and simulates states, but may be ill-conditioned over long or unstable horizons. Multiple shooting shortens sensitivity propagation using segment states and continuity constraints. Direct transcription optimizes states and controls simultaneously and replaces dynamics with sparse defects. Collocation schemes trade polynomial order, mesh size, and robustness. Accurate sparse derivatives, scaling, initialization, continuation, refinement, dense constraint checks, and independent simulation are essential for numerical credibility.

Key terms

Time discretization; mesh; control parameterization; direct shooting; multiple shooting; direct transcription; collocation; defect; quadrature; Euler forward; trapezoidal rule; Hermite–Simpson; zero-order hold; pseudospectral method; Legendre–Gauss (LG), Legendre–Gauss–Radau (LGR), and Legendre–Gauss–Lobatto (LGL) points; differentiation matrix; static parameters; sparse NLP; Jacobian sparsity; hp-adaptive mesh refinement; automatic differentiation; complex step; dependency detection; mesh refinement; warm start; continuation; independent simulation.

Problems

1. **Single-shooting sensitivity and conditioning.** For $\dot{x} = f(x, u(t; c), p)$ with terminal residual $r = x(T) - x_T$, derive the exact Jacobian $\partial r/\partial(p, c)$ from variational equations and show why its norm can grow exponentially with the horizon for unstable dynamics.

2. **Multiple-shooting Jacobian.** Partition $[0, T]$ into N shooting intervals with independent initial states s_i. Derive the continuity constraints and their block-bidiagonal Jacobian, then prove how this structure limits sensitivity growth relative to single shooting.

3. **Hermite–Simpson transcription.** Transcribe $\min \Phi(x_N) + \int_0^T L(x, u, p)dt$ subject to $\dot{x} = f(x, u, p)$ using Hermite–Simpson collocation, deriving the midpoint state, midpoint dynamics, defect equation, and Simpson quadrature contribution for one generic interval.

4. **Legendre–Gauss–Radau collocation.** On one mesh interval, derive the LGR differentiation matrix and quadrature weights from the Lagrange basis, then express the state defects and endpoint update in a form suitable for a sparse nonlinear program.

5. **Sparse KKT complexity.** A direct transcription has n_x states, n_u controls, n_p global plant variables, and N intervals. Derive the asymptotic nonzero counts of the constraint Jacobian and KKT matrix and compare time-ordered sparse factorization with dense factorization.

6. **Derivative verification.** For a collocation NLP residual $F(z)$, derive truncation and roundoff error models for forward, central, complex-step, and algorithmic derivatives, then prescribe a numerical test that distinguishes a coding error from finite-difference noise.

7. **Goal-oriented mesh refinement.** Given collocation defects $d_i(t)$ and an adjoint $\lambda(t)$ for the objective, derive an adjoint-weighted local error indicator and a mesh-refinement rule that targets objective error rather than state error alone.

8. **hp-adaptive decision.** Suppose the estimated local error behaves as $C_p h^{p+1}$ for a smooth solution but only $C_s h^{\nu}$ near a nonsmooth control switch. Construct an hp-adaptive rule that chooses polynomial enrichment or interval subdivision from observed coefficient decay and justify its asymptotic behavior.

9. **Post-solution optimality audit.** For a transcribed CCD nonlinear program, define a single normalized verification metric combining primal feasibility, dual feasibility, complementarity, mesh defect, and objective stability under refinement, with scaling that prevents any physical unit from dominating the audit.

10. **Numerical convergence study.** For $\min \int_0^1 (x^2 + 10^{-2}u^2)dt$ subject to $\dot{x} = -px + u$, $x(0) = 1$, $x(1) = 0$, $0.1 \le p \le 3$, and $|u| \le 2$, design a reproducible comparison of trapezoidal, Hermite–Simpson, and LGR transcription that estimates each method's observed convergence order and separates discretization error from NLP termination error.

References and further reading

1. Herber, D. R. (2017). *Advances in combined architecture, plant, and control design* (Doctoral dissertation, University of Illinois at Urbana–Champaign). Illinois Digital Environment for Access to Learning and Scholarship. https://www.ideals.illinois.edu/items/105359
2. Betts, J. T. (2010). *Practical methods for optimal control and estimation using nonlinear programming* (2nd ed.). Society for Industrial and Applied Mathematics. DOI: 10.1137/1.9780898718577
3. Biegler, L. T. (2010). *Nonlinear programming: Concepts, algorithms, and applications to chemical processes*. Society for Industrial and Applied Mathematics. DOI: 10.1137/1.9780898719383
4. Kelly, M. (2017). An introduction to trajectory optimization: How to do your own direct collocation. *SIAM Review, 59*(4), 849–904. DOI: 10.1137/16M1062569
5. Rao, A. V. (2009). A survey of numerical methods for optimal control. *Advances in the Astronautical Sciences, 135*, 497–528.
6. Garg, D., Patterson, M., Hager, W. W., Rao, A. V., Benson, D. A., & Huntington, G. T. (2010). A unified framework for the numerical solution of optimal control problems using pseudospectral methods. *Automatica, 46*(11), 1843–1851. DOI: 10.1016/j.automatica.2010.06.048
7. Patterson, M. A., & Rao, A. V. (2014). GPOPS-II: A MATLAB software for solving multiple-phase optimal control problems using hp-adaptive Gaussian quadrature collocation methods. *ACM Transactions on Mathematical Software, 41*(1), Article 1, 1–37. DOI: 10.1145/2558904
8. Nocedal, J., & Wright, S. J. (2006). *Numerical optimization* (2nd ed.). Springer. DOI: 10.1007/978-0-387-40065-5
9. Allison, J. T., & Herber, D. R. (2014). Multidisciplinary design optimization of dynamic engineering systems. *AIAA Journal, 52*(4), 691–710. DOI: 10.2514/1.J052182
10. Herber, D. R., & Allison, J. T. (2019). Nested and simultaneous solution strategies for general combined plant and control design problems. *Journal of Mechanical Design, 141*(1), Article 011402. DOI: 10.1115/1.4040705

Practical CCD Studies and Advanced Topics

Chapter 8: Practical CCD Studies and Advanced Topics

Applications, robustness, model fidelity, implementation, reproducibility, and validation

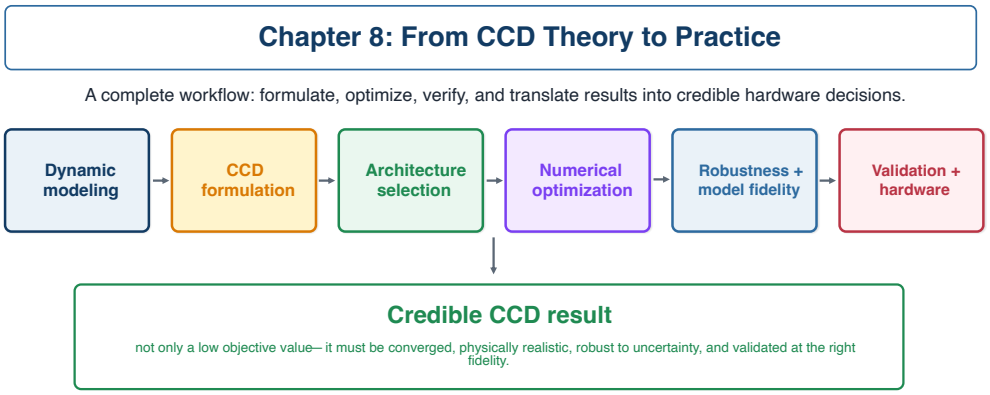
> A practical CCD study connects mathematical optimization to engineering evidence. Its final design must remain credible when the model, environment, solver, and hardware are examined critically.

Earlier chapters developed the theory and computational tools of control co-design (CCD). This chapter integrates those ideas through two applications. An active vehicle suspension provides a complete tutorial in plant–control coupling, trajectory-dependent objectives, path constraints, alternative CCD architectures, uncertainty, and implementation limits. A larger wind–wave energy application shows how the same ideas scale to multiphysics systems with expensive models, many operating conditions, and multiple control layers.

Learning objectives

After completing this chapter, you should be able to:

1. formulate and interpret a complete active-suspension CCD study;
2. compare sequential, nested, and simultaneous results fairly;
3. extend CCD formulations to multiple scenarios and uncertain quantities;
4. choose suitable model fidelity for each design stage;
5. use surrogate and multi-fidelity models to accelerate CCD;
6. identify constraints that connect optimized designs to hardware;
7. construct a reproducible numerical and experimental validation plan; and
8. organize a practical CCD workflow from problem definition to validated design.



Chapter 8 integrates the full CCD workflow from formulation to validated hardware.

What makes a CCD study practical?

A mathematically correct formulation is necessary, but insufficient. A practical study establishes a defensible chain of evidence:

1. the design question is meaningful;
2. the model represents important physics and information flow;
3. objectives and constraints reflect engineering priorities;
4. the selected CCD architecture is solved reliably;

5. the result is converged and reproducible;
6. the design is robust to realistic uncertainty;
7. higher-fidelity models and experiments support the conclusion; and
8. the controller and hardware can be implemented safely.

A failure at any link can invalidate the result. For example, a simultaneous solution may exploit nonexistent actuator bandwidth; a robust design may use an unrealistic uncertainty distribution; or a surrogate-assisted optimum may fail when checked with the original model.

Practical definition of success

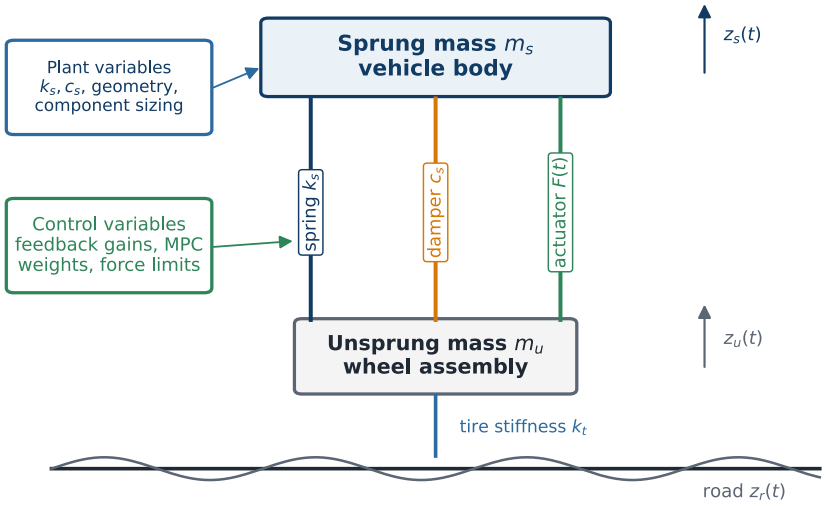
A successful CCD study identifies a coordinated plant–control design and supplies enough numerical, physical, and experimental evidence for another engineer to understand, reproduce, challenge, and continue it.

Active-Suspension Model and Formulation

Engineering purpose

A vehicle suspension must isolate the body from road disturbances while maintaining tire contact and acceptable suspension travel. Passive springs and dampers provide inherent behavior. An active actuator can improve comfort and handling, but introduces power, sensing, control, cost, and reliability requirements. Plant choices change the dynamics and best control action; control action changes motion, tire load, travel, and the value of plant designs.

Quarter-car model



The sprung mass m_s represents one-quarter of the vehicle body; m_u represents the wheel and associated components. With road displacement $z_r(t)$, suspension stiffness k_s, damping c_s, active force $F(t)$, tire stiffness k_t, and upward displacement positive,

$$m_s\ddot{z}_s = -k_s(z_s - z_u) - c_s(\dot{z}_s - \dot{z}_u) + F,$$

$$m_u\ddot{z}_u = k_s(z_s - z_u) + c_s(\dot{z}_s - \dot{z}_u) - F - k_t(z_u - z_r).$$

A convenient state vector is

$$\mathbf{x} = [z_s, \dot{z}_s, z_u, \dot{z}_u]^T.$$

Important outputs include body acceleration $\ddot{z}_s$ for comfort, tire deflection $z_u - z_r$ for road holding, suspension travel $z_s - z_u$, actuator force and energy, and component mass, cost, and packaging.

Plant and control decisions

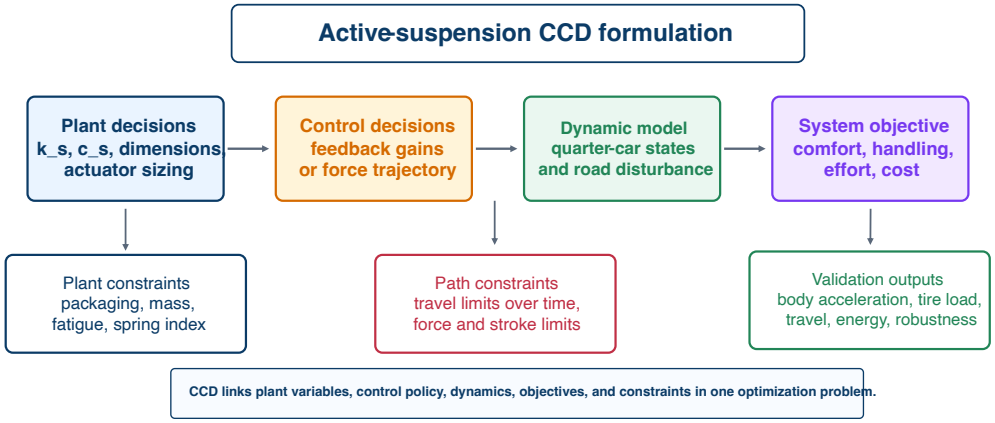
A low-fidelity plant vector may be

$$\mathbf{x}_p = [k_s, c_s, F_{\max}]^T.$$

At higher fidelity, stiffness and damping follow from geometry. Decisions can include spring wire and coil diameter, active coil count, damper piston diameter, orifice area, and actuator rating. A fully detailed suspension study makes these choices explicit. One published quarter-car formulation defines the plant vector as $\mathbf{x}_p = [d, D, p, N_a, D_o, D_p, D_s]^T$, where d and D are the spring wire and helix diameters, p is the coil pitch, N_a is the number of active coils, D_o is the damper valve diameter, D_p is the damper piston diameter, and D_s is the damper stroke. The spring constant follows from $k_s = d^4G/[8D^3N_a(1 + 1/(2C^2))]$ with spring index $C = D/d$, and the damper constant follows from an orifice-flow relation in D_p, D_o, and the damper fluid properties. Resulting plant constraints include a spring-index bound roughly $4 \le C \le 12$ (springs with $C < 4$ are difficult to form; $C > 12$ can tangle), a buckling limit on free length versus helix diameter, a packaging bound on outer spring diameter, a Soderberg fatigue criterion combining mean and alternating shear stress over the load history, and a damper-fluid thermal model relating heat generation ($q_{\text{gen}} = c_s\dot{\xi}_3^2$, with ξ_3 the relative damper velocity) to a bounded fluid temperature that guards against seal damage and damping fade. Comparing a sequential design (plant optimized as a passive system, then a fixed open-loop force trajectory optimized for that plant) against a fully simultaneous direct-transcription solution of this same detailed model, the simultaneous result improved the system objective by nearly 20%, but it also commanded substantially higher peak control force (roughly 2400 N versus 1400 N for the sequential design) and required more than an order of magnitude more function evaluations to solve. Coordinated designs are therefore not free: performance gains and added actuation and computational cost trade off together.

Control decisions depend on the representation:

- **Open-loop:** sampled forces $F_0, \ldots, F_N$.
- **State feedback:** gains in $F = -K\mathbf{x}$.
- **Output feedback:** gains using measurable signals.
- **MPC:** weights, prediction horizon, sample time, and estimator parameters.



System-level objective and constraints

A representative finite-horizon objective is

$$J(\mathbf{x}_p, \mathbf{x}_c) = \int_{t_0}^{t_f} \Big[w_h(z_u - z_r)^2 + w_c\ddot{z}_s^2 + w_u F^2 + w_r(z_s - z_u)^2\Big]dt + w_m M(\mathbf{x}_p) + w_C C(\mathbf{x}_p).$$

The terms represent handling, comfort, control effort, suspension travel, plant mass, and cost. Quantities with

firm allowable values are often clearer as constraints. For example,

$$-r_{\max} \le z_s(t) - z_u(t) \le r_{\max},$$

$$|F(t)| \le F_{\max}, \qquad |\dot{F}(t)| \le \dot{F}_{\max}.$$

Plant feasibility must also capture spring index and manufacturability, buckling, free and solid height, stress and fatigue, packaging, actuator stroke, velocity, power, thermal capacity, and bandwidth. A force limit alone can permit impossible high-frequency actuation.

Activity 8.1: Full Active-Suspension CCD Benchmark

Activity 8.1: Full Active-Suspension CCD Benchmark

Consider the quarter-car suspension model

$$\begin{aligned} m_s\ddot{z}_s &= -k_s(z_s - z_u) - c_s(\dot{z}_s - \dot{z}_u) + f_a, \\ m_u\ddot{z}_u &= k_s(z_s - z_u) + c_s(\dot{z}_s - \dot{z}_u) - k_t(z_u - z_r) - f_a, \end{aligned}$$

with actuator dynamics

$$\dot{f}_a = \omega_a(f_c - f_a).$$

Use

$$m_s = 300\ \text{kg}, \qquad m_u = 40\ \text{kg}, \qquad k_t = 190{,}000\ \text{N/m}, \qquad \omega_a = 60\ \text{rad/s}.$$

The plant-design bounds are

$$\begin{aligned} 10{,}000 \le k_s &\le 40{,}000\ \text{N/m}, \\ 500 \le c_s &\le 3500\ \text{N s/m}, \\ 500 \le F_{\max} &\le 4000\ \text{N}. \end{aligned}$$

Use the causal feedback controller

$$f_c(t) = \text{sat}[-K_1(z_s - z_u) - K_2(\dot{z}_s - \dot{z}_u) - K_3(z_u - z_r) - K_4\dot{z}_u, F_{\max}],$$

with gain bounds

$$\begin{aligned} 0 \le K_1 \le 60{,}000, \quad 0 \le K_2 \le 6000, \\ 0 \le K_3 \le 80{,}000, \quad 0 \le K_4 \le 6000. \end{aligned}$$

The road input is a half-cosine bump:

$$z_r(t) = \begin{cases} \dfrac{h_b}{2}\left[1 - \cos\left(\dfrac{2\pi(t - t_b)}{T_b}\right)\right], & t_b \le t \le t_b + T_b, \\ 0, & \text{otherwise}, \end{cases}$$

where

$$h_b = 0.05\ \text{m}, \qquad t_b = 0.25\ \text{s}, \qquad T_b = 0.10\ \text{s}.$$

Use the system-level objective

$$J = \frac{1}{T}\int_0^T \left[0.45\left(\frac{\ddot{z}_s}{2.5}\right)^2 + 0.20\left(\frac{z_u - z_r}{0.015}\right)^2 + 0.20\left(\frac{z_s - z_u}{0.05}\right)^2 + 0.10\left(\frac{f_c}{3000}\right)^2\right] dt + 0.05\left(\frac{F_{\max}}{3000}\right)^2 + 0.02\left(\frac{k_s}{25{,}000}\right)^2 + 0.02\left(\frac{c_s}{2000}\right)^2, \qquad T = 2\ \text{s}.$$

Impose

$$|z_s - z_u| \le 0.08\ \text{m}, \qquad |z_u - z_r| \le 0.025\ \text{m}, \qquad |f_c| \le F_{\max}, \qquad |f_a| \le F_{\max}.$$

1. Derive the five-state nonlinear closed-loop model.
2. Formulate the single-pass sequential design:
   (a) optimize k_s, c_s, and $F_{\max}$ using a fixed nominal controller; and
   (b) freeze the plant and optimize $K_1, \ldots, K_4$.
3. Formulate the nested design
$$\min_{k_s, c_s, F_{\max}} \left[\min_{K_1,\ldots,K_4} J\right].$$
4. Formulate the simultaneous CCD problem.
5. Solve all three formulations using identical dynamics, constraints, integration tolerances, initial conditions, and objective weights.
6. Use at least ten initial guesses for the nested and simultaneous formulations.
7. Compare the best designs in terms of
$$J^*, \quad k_s^*, \quad c_s^*, \quad F_{\max}^*, \quad K_1^*, \ldots, K_4^*.$$
8. Verify every candidate design using an independent high-accuracy forward simulation and a time grid at least ten times finer than the optimization grid.
9. Explain any disagreement between the nested and simultaneous results in terms of local minima, inner-loop convergence, derivatives, scaling, and solver termination criteria.

Comparing Suspension CCD Architectures

Fair comparison

Sequential, nested, and simultaneous methods should use the same physical model, scenarios, system objective, bounds, path and boundary constraints, discretization accuracy, and convergence requirements. Initialization effort and solver tolerances should also be comparable. Otherwise, differences may reflect formulation rather than coordination.

Sequential formulation

A passive-first baseline is

$$\mathbf{x}_p^{\text{seq}} = \arg\min_{\mathbf{x}_p} J(\mathbf{x}_p, \mathbf{x}_c = 0),$$

$$\mathbf{x}_c^{\text{seq}} = \arg\min_{\mathbf{x}_c} J(\mathbf{x}_p^{\text{seq}}, \mathbf{x}_c).$$

It answers a practical question: how much value is lost when physical suspension selection precedes active-control design?

Nested and simultaneous formulations

Nested CCD evaluates each plant under its best achievable controller:

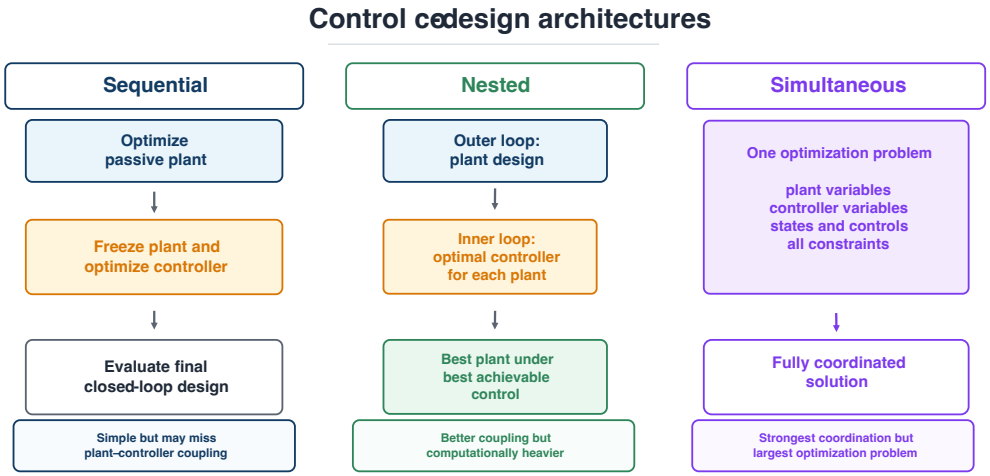
$$\min_{\mathbf{x}_p} \ \psi(\mathbf{x}_p), \qquad \psi(\mathbf{x}_p) = \min_{\mathbf{x}_c, \mathbf{x}(\cdot), \mathbf{u}(\cdot)} J(\mathbf{x}_p, \mathbf{x}_c).$$

The inner problem enforces control and dynamic constraints; the outer problem enforces plant constraints. This is conceptually attractive but potentially expensive.

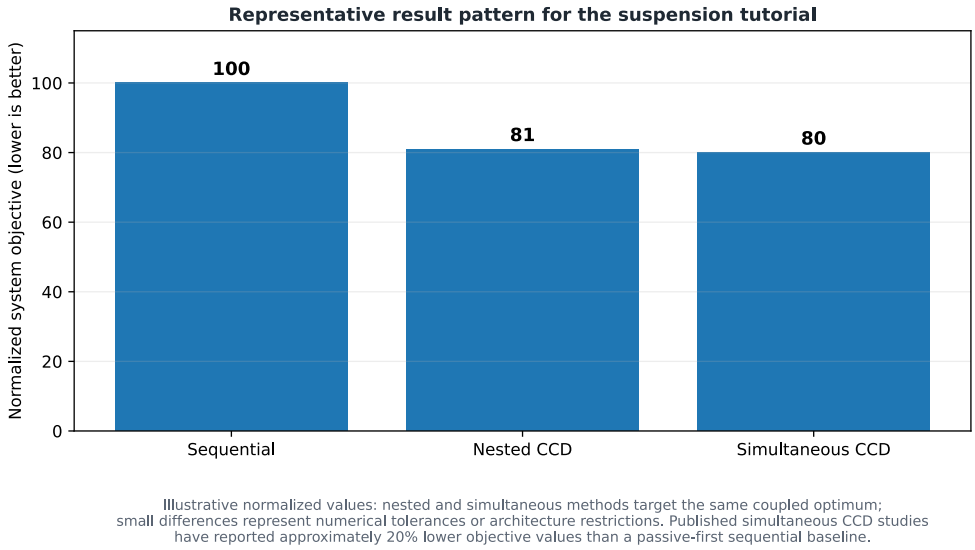
Simultaneous CCD solves

$$\min_{\mathbf{x}_p, \mathbf{x}_c, \mathbf{x}(\cdot), \mathbf{u}(\cdot)} J(\mathbf{x}_p, \mathbf{x}_c)$$

subject to dynamics, plant constraints, path constraints, and controller relations. After transcription, every plant, controller, state, and control variable appears in one sparse nonlinear program.



Representative result pattern



With the sequential result normalized to 100, a representative pattern places nested and simultaneous results near 80. These values are instructional, not a reproduction of one dataset. Under ideal conditions, nested and simultaneous formulations can target the same coupled optimum; differences may arise from inner-solve accuracy, local minima, controller restrictions, or numerical tolerances.

Interpretation should examine whether the coordinated plant becomes softer or stiffer, how damping changes, actuator force and energy, active constraints, improvement by objective component, and performance on unseen road profiles. Active control may support passive elements that look unattractive alone but work well in the closed-loop system.

Do not report only the aggregate objective

A weighted value can hide unacceptable tradeoffs. Report individual objective terms, active constraints, trajectories, computation measures, and physical design values.

Verified performance and computational trade-offs

A detailed active-suspension case study built exactly the model above — spring wire diameter, helix diameter, pitch, and active coil count for the plant, damper valve and piston diameter and stroke for the actuator — and evaluated a ramp-input load case together with a rough-road load case built from measured road-roughness data, using sequential, nested, and simultaneous strategies under matched dynamics, constraints, and tolerances. Two findings deserve emphasis.

First, which architecture is fastest numerically depends heavily on the derivative method available, not only on the coordination strategy. With symbolic derivatives and a sufficiently fine time mesh, the simultaneous formulation was the fastest approach, roughly ten times faster than the same formulation using the complex-step method (whose accuracy was itself close to symbolic). When only complex-step or finite-difference derivatives were available, the nested formulation became faster — in some cases by an order of magnitude — because its inner-loop subproblem reduces to a small, well-structured quadratic program while the outer loop's derivative burden stays low; increasing outer- and inner-loop accuracy in the nested strategy moved solution time from single-digit seconds to several minutes.

Second, the nested strategy carries a structural hazard that a purely theoretical equivalence argument can hide: sampling plant designs uniformly within the stated bounds, roughly 44% produced an infeasible inner-loop optimal control problem. A naive nested implementation started from such a plant design would fail to converge

before the outer loop ever assessed it.

Derivative method interacts with architecture choice

A claim that "simultaneous beats nested" or the reverse is incomplete without stating which derivative method (symbolic, complex-step, or finite difference) was used and at what mesh density. The faster architecture can flip when only the derivative method changes, and a fair comparison must report both.

Extending the comparison to architecture

The plant-and-control comparison generalizes to include architecture — which passive and active components exist and how they connect — as a third coordinated design layer, consistent with the three design domains introduced earlier in the course. One suspension study organized this extension as a trilevel solution strategy: an outer level proposes a candidate architecture from a catalog of components (masses, springs, dampers, a force actuator, and junctions), a middle level solves the outer-loop plant-design problem for that candidate architecture with a multistart search, and an inner level solves the resulting control subproblem, nested inside the plant level, which is itself nested inside the architecture level.

Applied to a quarter-car suspension with fixed sprung and unsprung mass, six architectures were compared on a common objective combining handling, comfort, and control effort: a canonical passive spring-damper design, a pure active design (a force actuator alone), a canonical active design (spring and damper in parallel with a force actuator), an active design augmented with a tuned-mass dynamic absorber, and two novel architectures discovered by the outer-level architecture search. The canonical passive design performed worst of the six. A novel passive architecture using seven additional components (extra masses, springs, and dampers) reduced the objective by about 12% relative to the canonical passive design, driven mainly by improved handling. Among active designs, the canonical active architecture improved on the pure-active design by about 3.5%, while adding a dynamic absorber gave no measurable benefit — the optimizer drove the absorber mass to its lower bound, effectively removing it from the system. A second novel active architecture, again using seven additional components, gave the best overall result: about 13% better than the canonical active design, and achieved with the smallest peak control force of any active variant considered.

These architecture-level results reinforce a point from Chapter 1: coordinating control with plant *and* architecture decisions can outperform coordinating control with plant decisions alone, but only when the added architectural complexity is justified by measured performance rather than assumed. A complexity metric that simply counts additional components makes this tradeoff explicit rather than hiding it inside a single aggregate objective.

Uncertainty and Robustness

Nominal CCD assumes a fixed model and operating condition. Real systems encounter environmental variability, payload changes, manufacturing tolerances, sensor noise and delay, model-form error, and actuator saturation or derating. A nominal optimum near a constraint boundary may therefore be fragile.

Scenario formulation

For scenario s with uncertain quantities $\boldsymbol{\theta}_s$ and weight p_s,

$$\min_{\mathbf{x}_p, \mathbf{x}_c} \sum_{s=1}^{N_s} p_s J(\mathbf{x}_p, \mathbf{x}_c; \boldsymbol{\theta}_s)$$

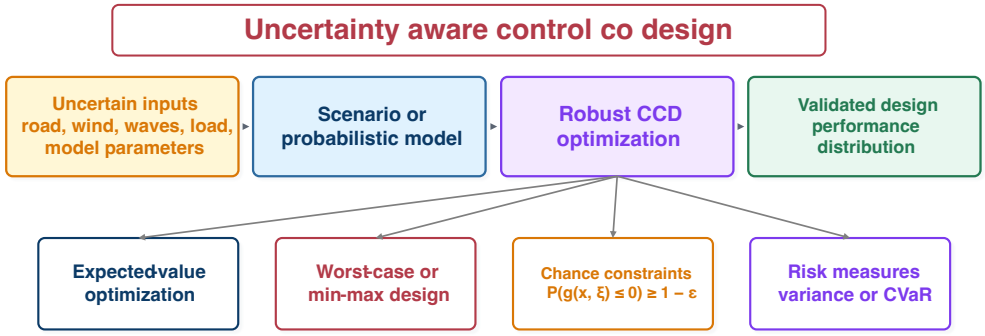
subject to required scenario constraints

$$g(\mathbf{x}_p, \mathbf{x}_c; \boldsymbol{\theta}_s) \le 0, \qquad s = 1, \ldots, N_s.$$

Suspension scenarios might combine bump, rough-road, pothole, and sinusoidal inputs; light, nominal, and heavy payloads; tire-property changes; sensor delay; and actuator derating.

Risk-aware alternatives

- **Worst case:** $\min_{\mathbf{x}_p, \mathbf{x}_c} \max_{\boldsymbol{\theta} \in \Theta} J(\mathbf{x}_p, \mathbf{x}_c; \boldsymbol{\theta})$ protects against the most damaging defined case, but may be conservative.
- **Mean–variance:** $\min\ \mathbb{E}[J] + \lambda\, \mathrm{Var}(J)$ balances average and dispersion.
- **Chance constraint:** $\mathbb{P}(g(\mathbf{x}_p, \mathbf{x}_c; \boldsymbol{\theta}) \le 0) \ge 1 - \epsilon$ imposes a satisfaction probability.
- **CVaR:** penalizes the upper tail of poor outcomes and reveals more than mean performance alone.



Goal: design choices that remain reliable when operating conditions and model parameters vary.

A taxonomy of uncertain CCD formulations

Uncertainty can be represented in three fundamentally different ways: probabilistically, when a distribution is known or can be estimated; crisply, as a bounded set with no probability measure attached; or possibilistically, through a fuzzy membership function, for cases where even the shape and size of a bounding set are not well known. This representation choice interacts with a second distinction: aleatory uncertainty is irreducible variability inherent to the phenomenon itself (manufacturing variation across nominally identical parts cannot be reduced by acquiring more knowledge), while epistemic uncertainty reflects limited knowledge that could in principle be reduced by more data, more testing, or a better model (an unvalidated model-form assumption is epistemic).

Crossing these representations with risk attitude yields several specialized uncertain-CCD (UCCD) formula-

tions, all built from the same nominal, all-at-once CCD problem. A **stochastic-expectation** formulation minimizes the expected objective subject to expected-value constraints and is risk-neutral. A **stochastic chance-constrained** formulation instead requires each constraint to be satisfied with at least a target probability, $\mathbb{P}[g_i \leq 0] \geq 1 - \mathbb{P}_{f,i}$, generalizing the chance constraint above. A **probabilistic robust** formulation penalizes both the expected value and the standard deviation of the objective and constraints together, trading multiobjective weight for reduced sensitivity — the mean–variance idea generalized to constraints as well as the objective. A **worst-case robust** formulation replaces the probabilistic description entirely with a bounded uncertainty set and requires feasibility for every realization in that set. **Fuzzy expected-value** and **possibilistic chance-constrained** formulations use possibility measures instead of probability measures when only vague, expert-elicited information about the uncertainty is available, such as linguistic descriptions of an expected operating range.

The worst-case robust formulation (WCR-UCCD) is structurally a bi-level, min–max problem. The outer level chooses plant, control, and state variables to minimize a deterministic objective, while an inner-level maximization searches the uncertainty set for the worst-case realization of each constraint,

$$\Phi_i = \max_{\boldsymbol{\theta} \in \Theta} \; g_i(\mathbf{x}_p, \mathbf{x}_c; \boldsymbol{\theta}),$$

and the outer problem enforces $\Phi_i \leq 0$ for every constraint i. Because uncertain objectives and constraints play an interchangeable role in a UCCD formulation, an uncertain objective can always be moved into the constraint set through an **epigraph reformulation**: introduce a new scalar decision variable v and the constraint $o(\cdot) - v \leq 0$, then minimize v. This lets every complication introduced by uncertainty be handled uniformly within the inequality constraints, rather than requiring separate machinery for the objective.

Two further distinctions are useful when formulating a UCCD problem. Uncertainty that acts through the same channel as the control input is called *matched* (or lumped); uncertainty that enters the dynamics through a different channel is called *mismatched*, and generally cannot be compensated by feedback in the same direct way. Equality constraints also split into two types under uncertainty: *Type I* constraints describe the physics of the system (the dynamics and any algebraic relations) and must be satisfied at every point in the uncertainty space considered; *Type II* constraints describe a design requirement that cannot be strictly satisfied once its arguments are uncertain, and so must instead be relaxed — typically to a statement about its expected value while its variance is minimized.

Choosing an uncertainty representation is a modeling decision

The representation is not fixed by the physics alone. When the risk associated with a specific criterion is severe enough that no constraint violation can be tolerated, a worst-case robust formulation may be preferable even when distributional information is available — the choice reflects how much conservatism the design can afford, not only how much is known about the uncertainty.

Verification under uncertainty

Even a nominal optimization should be tested with Monte Carlo simulation or a structured validation set. Report mean and worst objective, constraint-violation probability, percentiles of key outputs, sensitivities, and observed failure modes. Keep validation scenarios independent from those used to tune the design whenever possible.

Activity 8.2: Robust Suspension Design Using Expected Value, Worst Case, and CVaR

Activity 8.2: Robust Suspension Design Using Expected Value, Worst Case, and CVaR

Use the suspension model from Activity 8.1. Define the uncertain sprung mass and bump height as

$$m_s \in \{240, 300, 360\}\ \text{kg}, \qquad h_b \in \{0.03, 0.05, 0.07\}\ \text{m}.$$

This produces nine equally probable scenarios. Let the scenario objective be

$$J_s(\mathbf{x}_p, \mathbf{x}_c), \qquad s = 1, \dots, 9.$$

1. Formulate the expected-value problem

$$\min_{\mathbf{x}_p, \mathbf{x}_c} \frac{1}{9} \sum_{s=1}^{9} J_s.$$

2. Formulate the worst-case problem using an epigraph variable η:

$$\begin{aligned} \min_{\mathbf{x}_p, \mathbf{x}_c, \eta} \quad & \eta, \\ \text{subject to} \quad & J_s \le \eta, \qquad s = 1, \dots, 9. \end{aligned}$$

3. Formulate the combined mean-worst-case problem

$$\min \quad \frac{1}{9} \sum_{s=1}^{9} J_s + \lambda \eta, \qquad \lambda \in \{0, 0.1, 0.5, 1, 2\}.$$

4. Formulate the empirical CVaR objective at confidence level $\alpha = 0.9$ using auxiliary variables ξ_s:

$$\text{CVaR}_\alpha = \eta + \frac{1}{(1-\alpha) N_s} \sum_{s=1}^{N_s} \xi_s,$$

   subject to

$$\xi_s \ge J_s - \eta, \qquad \xi_s \ge 0.$$

5. Require every path constraint to hold in all nine scenarios.
6. Solve the nominal, expected-value, worst-case, and CVaR designs.
7. Compare

$$J_{\text{nominal}}, \qquad \mathbb{E}[J], \qquad \max_s J_s, \qquad \text{CVaR}_{0.9}\,.$$

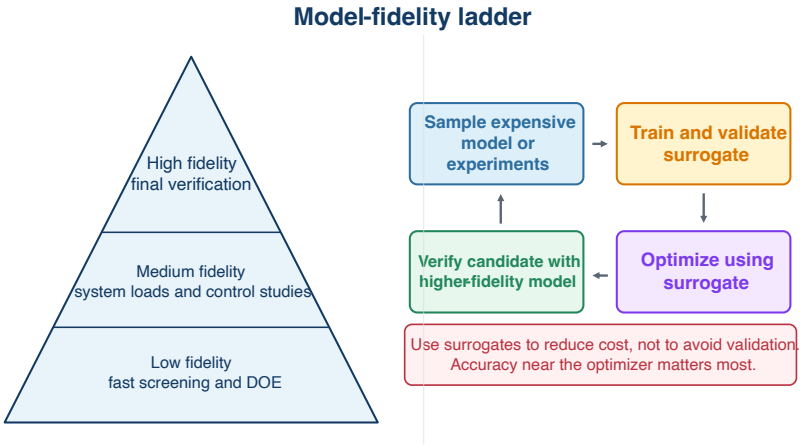
8. Quantify the nominal-performance penalty of robustness.
9. Determine which plant and controller variables change most strongly when moving from nominal to robust design.

10. Explain why a robust design can have a worse nominal objective but still be the more credible engineering design.

Model Fidelity and Surrogate Models

A fidelity ladder

No single model is best for every stage. Model fidelity should rise as the design narrows and the claim becomes stronger.



- **Low fidelity** supports broad exploration, sensitivity studies, architecture screening, and initialization.
- **Medium fidelity** adds nonlinearities, actuator and sensor models, realistic loading, and implementation details.
- **High fidelity** provides cross-checks with detailed multiphysics simulation, hardware-in-the-loop (HIL), prototypes, or experiments.

The key question is not whether a model is "accurate" in general, but whether it is sufficiently accurate for the design decision. The optimum can exploit omitted physics, so discrepancy must be assessed near promising designs, not only near a baseline.

Surrogate-assisted CCD

When repeated high-fidelity simulations are expensive, a surrogate workflow can:

1. sample plant, control, and operating variables;
2. run high-fidelity closed-loop simulations;
3. fit a reduced-order model, response surface, Gaussian process, neural surrogate, or derivative-function surrogate;
4. validate within the intended domain;
5. optimize with the surrogate;
6. evaluate candidates in the original model; and
7. add infill samples where error or improvement is large.

Trust regions, error indicators, and conservative margins can reduce damaging extrapolation. A surrogate for a scalar metric may be insufficient for controller design: transient trajectories, stability, peaks, and path constraints can require a dynamic surrogate.

Multi-fidelity reasoning

Low-fidelity models can rank designs or provide trends while a smaller number of high-fidelity cases correct bias and validate candidates. A discrepancy model can be written schematically as

$$y_H(\mathbf{d}) = y_L(\mathbf{d}) + \delta(\mathbf{d}),$$

where $\mathbf{d}$ collects design and operating variables. The correction δ is useful only within a validated domain and must not conceal structural model errors.

Comparing data-driven low-fidelity approaches: a floating-wind example

Not every surrogate needs a physics-based reduced-order structure. A recent comparison for a floating offshore wind turbine (the IEA-15MW reference turbine on a semisubmersible platform, simulated in OpenFAST through the WEIS toolbox with the open-source ROSCO controller) contrasted three data-driven ways of building a low-fidelity closed-loop "plant" for control optimization:

- **Subspace system identification** (the n4sid/SSARX algorithm) fits a discrete-time linear time-invariant state-space model directly from input–output time series, with model order chosen by a sensitivity study.
- **Deep learning** trains a recurrent network (an LSTM layer with a small number of cells and a fully connected output layer) to map inputs to outputs at each time step.
- A **derivative function surrogate model (DFSM)** identifies a continuous-time state-space model whose states are physically meaningful turbine quantities (platform pitch, tower-top displacement, generator speed, and their time derivatives) rather than an abstract identified state, and schedules the state-space matrices on wind speed using a linear parameter-varying (LPV) structure, interpolating between per-wind-speed models identified from simulated time series.

All three low-fidelity models ran far faster than the original high-fidelity simulation, which took nearly 20 minutes per load case: the n4sid model averaged around 2.5 seconds, the DFSM around 25 seconds, and the LSTM around 70 seconds. Simulation speed alone, however, was a poor guide to usefulness. The n4sid model was cheapest to evaluate but had the highest variance in closed-loop response accuracy across load cases — accurate for some, poor for others — partly because its identified states carry no direct physical meaning, which also makes it awkward to extend into a physically scheduled LPV surrogate. The LSTM had lower average error but still showed considerable case-to-case variance and no guaranteed stability. The DFSM balanced simulation time, accuracy, and consistency across load cases better than either alternative, and, unlike the subspace model, could be scheduled meaningfully on wind speed because its states correspond to physical turbine quantities that the LPV weighting can be built around.

The value of a DFSM for control co-design specifically was tested with a small design of experiments over blade-pitch-controller natural frequency and damping ratio, $\mathbf{x}_c = [\omega_{\mathrm{PC}}, \zeta_{\mathrm{PC}}]^T$: a full-factorial sample of 25 controller points, each evaluated at five wind speeds with several turbulent seeds, was run in both the high-fidelity simulator and the DFSM. The full high-fidelity sweep cost roughly 250 CPU-hours; the same sweep using the DFSM cost about 5 CPU-hours — close to a fifty-times speedup — and the DFSM's predicted tower-base damage-equivalent-load design space had the same qualitative shape and trends as the high-fidelity design space, even though the DFSM underpredicted the absolute load level and was evaluated at controller values it had not been trained on. For early-stage design-space screening, reproducing the *shape* of a response surface at a fraction of the cost can be more valuable than an unbiased absolute prediction, provided the bias is characterized before the model is trusted for a final decision.

Activity 8.3: Surrogate-Assisted Wind-Turbine CCD with Adaptive Infill

Activity 8.3: Surrogate-Assisted Wind-Turbine CCD with Adaptive Infill

Consider a simplified floating-wind CCD problem with plant and controller variables

$$\mathbf{x}_p = \begin{bmatrix} D_p & K_m & C_m \end{bmatrix}^T, \qquad \mathbf{x}_c = \begin{bmatrix} K_\Omega & K_\theta & K_{\dot{\theta}} \end{bmatrix}^T,$$

where D_p is a platform-dimension parameter and K_m and C_m are effective mooring stiffness and damping parameters. A high-fidelity simulator returns

$$\mathbf{y} = \begin{bmatrix} P_{\text{mean}} & \theta_{\text{rms}} & M_{\text{tower,max}} & T_{\text{moor,max}} \end{bmatrix}^T.$$

The system-level objective is

$$J_{\text{HF}} = -\frac{P_{\text{mean}}}{P_{\text{ref}}} + 0.3\left(\frac{\theta_{\text{rms}}}{\theta_{\text{ref}}}\right)^2 + 0.2\left(\frac{M_{\text{tower,max}}}{M_{\text{ref}}}\right)^2 + 0.1\left(\frac{T_{\text{moor,max}}}{T_{\text{ref}}}\right)^2 + C_{\text{plant}}(\mathbf{x}_p).$$

1. Construct a Latin-hypercube design of experiments containing at least 150 plant-controller samples.
2. Train separate Gaussian-process or radial-basis surrogates for all four outputs.
3. Split the data into training, validation, and test sets, and report

$$R^2, \qquad \text{RMSE}, \qquad \max|e|.$$

4. Formulate the surrogate CCD problem, including high-fidelity output constraints

$$M_{\text{tower,max}} \le M_{\text{allow}}, \qquad T_{\text{moor,max}} \le T_{\text{allow}}.$$

5. Solve the surrogate optimization from at least twenty initial guesses.
6. Evaluate the best five surrogate candidates in the high-fidelity model.
7. Define a combined infill criterion

$$\mathcal{I}(\mathbf{z}) = \widehat{J}(\mathbf{z}) - \kappa\sigma_J(\mathbf{z}) + \rho V(\mathbf{z}),$$

   where σ_J is predictive uncertainty and V is a predicted constraint-violation penalty.
8. Add at least five adaptive infill points per iteration and repeat until the high-fidelity objective changes by less than 0.5%.
9. Compare the surrogate optimum and final high-fidelity optimum.
10. Explain why a surrogate that predicts only annual energy or mean power is insufficient when transient loads appear in the constraints.

Activity 8.4: Multi-Fidelity CCD with a Trust-Region Correction Model

Activity 8.4: Multi-Fidelity CCD with a Trust-Region Correction Model

Let $J_L(\mathbf{z})$ be a low-fidelity CCD objective and $J_H(\mathbf{z})$ the corresponding high-fidelity objective, where

$$\mathbf{z} = \begin{bmatrix} \mathbf{x}_p \\ \mathbf{x}_c \end{bmatrix}.$$

Define the discrepancy

$$\delta(\mathbf{z}) = J_H(\mathbf{z}) - J_L(\mathbf{z}).$$

At iteration k, construct the corrected model

$$m_k(\mathbf{z}) = J_L(\mathbf{z}) + \widehat{\delta}_k(\mathbf{z}),$$

and solve

$$\min_{\mathbf{z}} \quad m_k(\mathbf{z}) \qquad \text{subject to} \qquad \left\| \mathbf{z} - \mathbf{z}^{(k)} \right\|_2 \le \Delta_k.$$

1. Derive the predicted reduction

$$\Delta m_k = m_k(\mathbf{z}^{(k)}) - m_k(\mathbf{z}^{(k)} + \mathbf{s}_k).$$

2. Derive the actual high-fidelity reduction

$$\Delta J_{H,k} = J_H(\mathbf{z}^{(k)}) - J_H(\mathbf{z}^{(k)} + \mathbf{s}_k).$$

3. Define the trust-region ratio

$$\rho_k = \frac{\Delta J_{H,k}}{\Delta m_k}.$$

4. Propose explicit rules for:

   (a) accepting or rejecting the trial design;

   (b) shrinking the trust region; and

   (c) expanding the trust region.

5. Implement the method for a low-fidelity suspension or wind model and a higher-fidelity validation model.

6. Use both additive and multiplicative corrections:

$$J_H \approx J_L + \widehat{\delta}, \qquad J_H \approx \widehat{\beta} J_L.$$

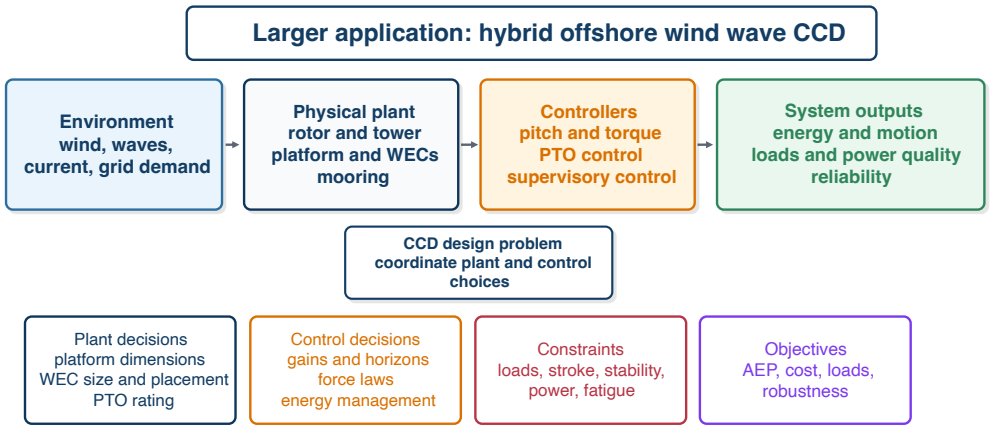
7. Compare the two correction strategies in terms of

$$\text{high-fidelity calls}, \quad \text{final objective}, \quad \text{constraint satisfaction}.$$

8. Determine whether the low-fidelity optimum lies inside a region where the correction model is accurate.
9. Explain why unconstrained optimization of a corrected surrogate can be unreliable without a trust region or another mechanism that controls extrapolation.

Wind and Wave Energy CCD

Floating wind and wave-energy systems combine aerodynamics, hydrodynamics, structural dynamics, moorings, power electronics, and multiple controllers. Their stochastic environment affects annual energy, extreme loads, fatigue, platform motion, power quality, and reliability.



Plant and control decisions

Candidate plant variables include rotor and blade properties, tower dimensions, platform geometry and mass distribution, mooring layout and stiffness, wave-energy-converter (WEC) geometry and placement, power-take-off (PTO) force and stroke ratings, and storage or conversion capacity.

Control decisions may include generator-torque and blade-pitch gains, platform-motion or load feedback, WEC PTO damping or optimal-force parameters, MPC horizons and weights, supervisory power sharing, and fault-handling thresholds.

Multi-scenario objective

A representative objective combines energy, loads, motion, power variation, and cost:

$$\begin{aligned}\min_{\mathbf{x}_p,\mathbf{x}_c} J = & -w_E E_{\text{ann}}(\mathbf{x}_p,\mathbf{x}_c) + w_L D_{\text{fat}}(\mathbf{x}_p,\mathbf{x}_c) \\ & + w_M M_{\text{motion}}(\mathbf{x}_p,\mathbf{x}_c) + w_P P_{\text{var}}(\mathbf{x}_p,\mathbf{x}_c) + w_C C_{\text{cap}}(\mathbf{x}_p).\end{aligned}$$

Annual behavior is approximated with weighted wind–wave bins, turbulent seeds, or representative scenarios. Constraints can address blade, tower, and mooring loads; platform pitch, heave, and surge; WEC angle, stroke, and PTO force; electrical power; closed-loop stability; clearance; survivability; and fault conditions.

Multi-fidelity strategy

A practical hierarchy uses:

1. frequency-domain or linear models for screening;
2. reduced or identified dynamic models for controller optimization;
3. engineering time-domain simulation for detailed design; and
4. CFD, high-order structural models, or experiments for selected verification.

A low-fidelity optimum is a candidate hypothesis, not final evidence.

Sequential versus integrated outcomes

A sequential process may optimize platform stability, add WECs, and then tune controllers. Integrated CCD may instead select a platform that moves more but enables better energy capture, locate WECs to reduce turbine motion, balance PTO rating against structural-load mitigation, or coordinate wind, wave, and storage subsystems. The system optimum can therefore differ qualitatively from individually optimized subsystems.

Quantitative evidence of integrated benefit

Across published wind turbine CCD studies, integrated formulations generally outperform sequential design when plant and control are strongly coupled, but the reported margin varies with how tightly the two domains actually interact for the specific system studied. A nested formulation applied to a spar-buoy floating turbine reported more than 11% improvement in annual energy production relative to a sequential baseline; a related linear-parameter-varying-based nested formulation on the same class of platform achieved about 3.4% reduction in levelized cost of energy. At larger scale, a 22 MW semisubmersible floating turbine study found that simultaneous CCD reduced structural mass by about 2% compared to a sequential approach while maintaining comparable performance — a smaller margin that illustrates a general pattern: the size of the CCD benefit depends not only on the chosen coordination architecture but on how strongly the system's plant and control decisions are coupled in the first place. When coupling is weak or dominated by a single subsystem, a sensitivity study focused on tower design reported only about 0.53% reduction in cost of energy from coordinated design, even though structural changes (a lighter, taller tower under fatigue-driven constraints) were still identified. A CCD study should therefore report, and not merely assume, the strength of plant–control coupling before claiming a large expected benefit.

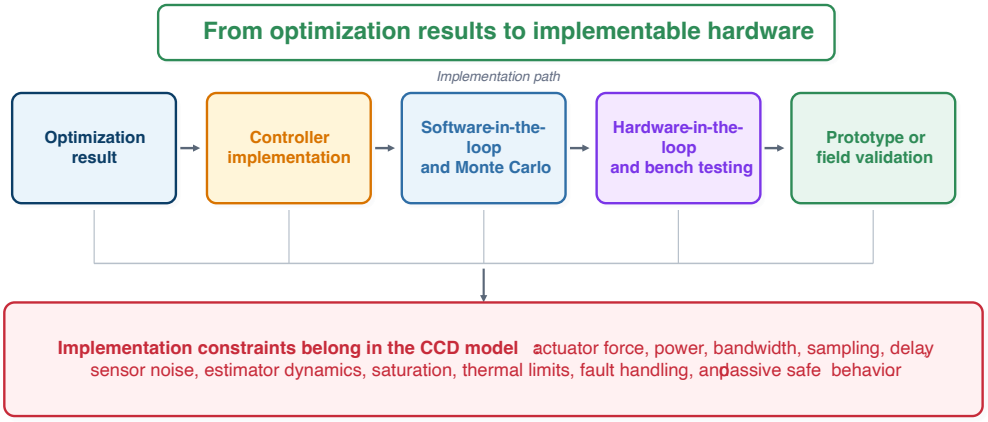
Research gaps and a path to standardization

A recent review of wind turbine CCD identified a consistent set of open gaps across the literature: a predominant reliance on fixed-structure PI/PID controllers that restricts the achievable control-design space; isolated, rather than joint, optimization of pitch and torque control; no consensus on which objective function (AEP, LCOE, mass, or cost) should drive a CCD study; sequential, discipline-specific workflows persisting even when a fully coupled simulation tool is already available; no systematic method for quantifying how strongly plant and control disciplines actually couple in a given design or for deciding how to decompose a large CCD problem into tractable subproblems; a persistent tension between model fidelity and computational cost; uncertainty that is often ignored across all CCD phases; and a shortage of validated, interpretable AI-based surrogates and controllers. The review also noted that current CCD studies rarely report the information needed for one study to be compared against another — consistent quantitative performance metrics, sensitivity analyses, and computational cost are inconsistently documented.

To close this gap, the review proposed three priorities for the field: establishing standard, reproducible benchmark problems for wind turbine CCD; developing comparative guidelines for how to decompose a wind turbine design problem based on a systematic, quantified analysis of design coupling rather than intuition; and extending the scope of wind turbine CCD studies toward more fully coupled tower–platform–control formulations rather than partial or sequential treatments. These priorities echo the practical-study checklist introduced at the start of this chapter: a credible CCD result needs not only a coordinated design, but evidence that lets another engineer reproduce, challenge, and extend it.

From Optimization to Implementable Hardware

A numerical optimum is an idealized mathematical object. Implementation requires physical dimensions and components, a digital controller, sensor and estimator specifications, real-time software, communications, and safe saturation and fault behavior.



Constraints to model

A closed-loop CCD formulation should consider:

- actuator saturation, rate, stroke, bandwidth, and efficiency;
- sampling and computational delay;
- sensor bandwidth, resolution, bias, and noise;
- state-estimation dynamics;
- anti-windup and saturation logic;
- electrical power and thermal limits;
- quantization and communications; and
- behavior when control is disabled.

Passive safety

When sensing, computation, or actuation fails, the passive plant should remain stable and avoid catastrophic behavior. The resulting constraints may reduce nominal performance but improve deployability.

Controller realization

An open-loop optimal trajectory can inform a realizable controller but normally should not be implemented directly unless future inputs are known. Practical transitions include fitting low-order feedback to optimal trajectories, gain scheduling, embedding the problem in MPC, identifying switching or feedforward rules, and simplifying learned policies under explicit safety constraints.

A physical example: from open-loop control to a laboratory MR damper

One suspension study made the transition from open-loop optimal control (OLC) trajectories to closed-loop control (CLC) concrete using a physical, reconfigurable trailing-arm suspension testbed whose geometric plant variables could be adjusted on the bench. The design process moved through a sequence of increasingly realistic and increasingly constrained control representations, each solved with the same road input and an objective that minimized sprung-mass acceleration and tire deflection: an unstructured active-force open-loop trajectory gave

the best possible, but unrealizable, performance benchmark; a semi-active open-loop damping-force trajectory — assuming an idealized damper able to supply any commanded force at a given velocity — came reasonably close to that benchmark; a semi-active open-loop current trajectory, constrained to the current-versus-force-versus-velocity behavior of a specific, laboratory-characterized magnetorheological (MR) damper, performed measurably worse than the idealized semi-active case because real MR dampers are hysteretic and can only dissipate, not supply, energy; and finally a full-state feedback controller acting on that same MR damper gave the most realizable, but least performant, design in the sequence.

The actuator selected for the physical build was a Lord 8041-1 MR damper, operated over a 0–1 A continuous current range with a maximum stroke of 74 mm. Its force–velocity–current behavior was characterized experimentally in the laboratory and fit with a smooth surrogate model before being used as a hard constraint in both the open-loop and closed-loop problems. Each step in this sequence traded some performance for realizability, and the overall pattern — ideal active, then idealized semi-active, then structured semi-active with a real component model, then closed-loop feedback — is a template for moving any CCD result from an idealized optimal-control benchmark toward a design that can be built and tested.

Bench validation closes the loop

A physical, reconfigurable testbed lets each stage of this sequence be checked against hardware rather than only against a higher-fidelity simulation. Reconfigurable geometric plant variables on the bench let the same optimization formulation drive both the numerical study and the physical build.

Validation stages

A staged campaign can proceed through:

1. software-in-the-loop simulation;
2. Monte Carlo uncertainty tests;
3. processor-in-the-loop or real-time execution;
4. hardware-in-the-loop testing;
5. component bench tests;
6. scaled or full prototypes; and
7. field validation.

Each stage should address a named modeling or implementation risk and use predefined acceptance criteria.

Activity 8.5: From Optimized Suspension to Implementable Hardware

Activity 8.5: From Optimized Suspension to Implementable Hardware

Use the optimized active-suspension design from Activity 8.1. Replace the ideal implementation with the sampled-data controller

$$f_{c,k} = \mathrm{sat}[-K\widehat{\mathbf{x}}_k, F_{\max}],$$

which is updated every T_s seconds and held constant between updates. The sensor measurements satisfy

$$\mathbf{y}_k = C\mathbf{x}_k + \mathbf{v}_k,$$

where $\mathbf{v}_k$ is zero-mean measurement noise. The command is delayed by n_d sampling intervals, and the actuator also satisfies

$$|\dot{f}_a| \le R_{\max}.$$

1. Derive a discrete-time model using zero-order hold.
2. Construct a state observer or Kalman filter for the available measurements.
3. Implement the delayed command

$$f_{c,k}^{\text{applied}} = f_{c,k-n_d}.$$

4. Impose the actuator-rate limit using

$$|f_{a,k+1} - f_{a,k}| \le R_{\max} T_s.$$

5. Evaluate

$$T_s \in \{0.001, 0.005, 0.01, 0.02, 0.05\} \text{ s}$$

and

$$n_d \in \{0, 1, 2, 5\}.$$

6. Determine the largest sampling time and delay for which all path constraints remain satisfied.
7. Add a passive-safety requirement: when the actuator is disabled, the passive suspension must satisfy

$$|z_s - z_u| \le 0.10 \text{ m}.$$

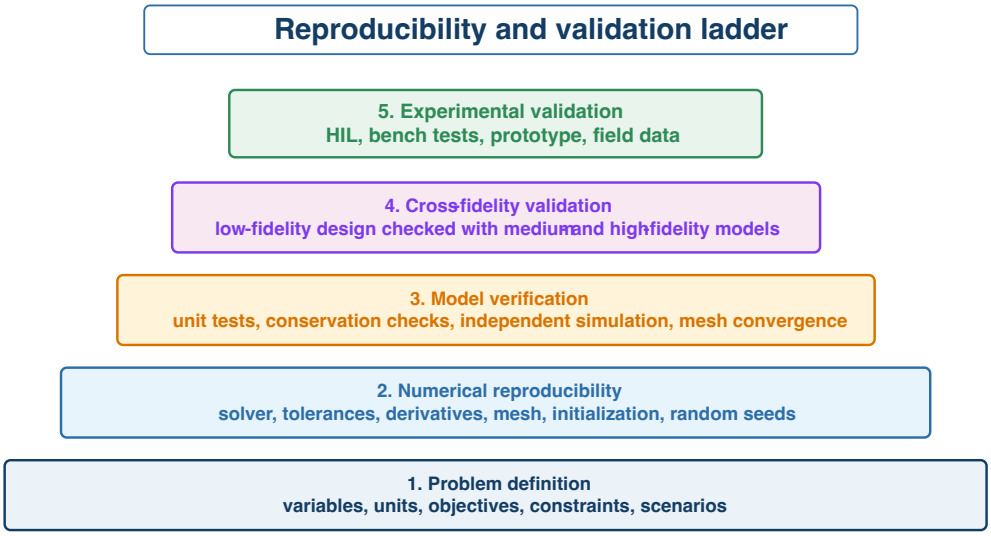
8. Re-optimize the plant and controller while including sampling time, command delay, observer dynamics, force-rate limits, and passive safety.
9. Compare the ideal and implementable CCD designs in terms of

$$J, \quad k_s, \quad c_s, \quad F_{\max}, \quad \text{constraint margin}.$$

10. Design a software-in-the-loop, hardware-in-the-loop, and bench-test validation sequence for the final design.

Reproducibility and Validation

CCD studies depend on model versions, meshes, derivatives, tolerances, initial guesses, scaling, random seeds, and data processing. Recording these choices allows others to recreate the design and diagnose differences.



Minimum reproducibility package

A complete package should include:

- governing equations, assumptions, variables, units, bounds, and scaling;
- objective terms, weights, constraints, scenarios, and input data;
- model and software versions;
- solver algorithms, tolerances, and linear-solver settings;
- derivative methods and sparsity information;
- mesh, time step, and convergence study;
- initial guesses, seeds, and multistart strategy;
- optimized designs and trajectories;
- postprocessing scripts and validation cases; and
- limitations and known failure modes.

Numerical verification

Before assigning physical meaning, confirm acceptable feasibility and optimality residuals, mesh convergence, between-node feasibility in an independent simulation, derivative accuracy, consistency across starts, and freedom from hidden scaling or bound errors.

Model validation

Validation asks whether the mathematical model predicts the real or higher-fidelity system well enough for the design decision. Compare trajectories, natural frequencies and damping, loads and energy, constraint margins, stability and bandwidth, and—critically—model error near the optimized design.

Validation is not a single score. Small average error can hide critical peaks or rare failures. Match validation evidence to the design claim: fatigue claims require fatigue-driving loads; feasibility claims require extreme trajectories and constraint margins.

Avoid validation leakage

Do not repeatedly tune a model or surrogate on the cases used to claim independent validation. Preserve a final test set or use additional high-fidelity and experimental cases not used during model construction.

The standardization gap in practice

The reproducibility and validation demands described above are not yet routine practice in every CCD subfield. A recent review of wind turbine CCD found that industrial adoption remains limited in part because CCD studies rarely report enough information for one study to be compared against another: quantitative performance metrics, sensitivity analyses, and computational cost are inconsistently documented, and there is no standardized way to quantify how strongly a given system's plant and control disciplines actually couple. The review identified this as a distinct research gap from model fidelity or uncertainty treatment — even a numerically well-verified individual study can fail to accumulate into transferable engineering knowledge if each published study uses different objectives, different constraint sets, and different reporting conventions. The remedy the review proposed for the field is exactly the reproducibility package emphasized in this chapter: standard, reproducible benchmark problems that let independent groups reproduce a claimed result and directly compare coordination strategies on identical terms, rather than reproducing only the qualitative conclusion that "coordination helps."

Activity 8.6: Independent Reproduction and Validation Audit

Activity 8.6: Independent Reproduction and Validation Audit

A research team reports an optimized CCD design

$$\mathbf{z}^* = \begin{bmatrix} \mathbf{x}_p^* \\ \mathbf{x}_c^* \end{bmatrix},$$

with objective J_{reported}, maximum path-constraint violation below 10^{-6}, and a claimed 18% improvement over a sequential baseline. Perform an independent reproduction and validation audit.

1. Define a reproducibility package containing:
   (a) governing equations;
   (b) units and variable definitions;
   (c) initial and boundary conditions;
   (d) objective weights;
   (e) all constraints and bounds;
   (f) software versions;
   (g) solver settings;
   (h) initialization;
   (i) random seeds; and
   (j) raw and processed data.
2. Recompute the reported objective using an independent implementation.
3. Define the relative objective discrepancy

$$e_J = \frac{|J_{\text{reproduced}} - J_{\text{reported}}|}{\max(1, |J_{\text{reported}}|)}.$$

4. Independently simulate the reported control law and compute the maximum trajectory discrepancy

$$e_x = \max_t \left\| \mathbf{x}_{\text{reported}}(t) - \mathbf{x}_{\text{independent}}(t) \right\|_\infty .$$

5. Re-evaluate every path constraint on a dense grid and report

$$v_{\max} = \max_{t,j} g_j(\mathbf{x}(t), \mathbf{u}(t)) .$$

6. Repeat the optimization using:
   (a) at least ten initial guesses;
   (b) two mesh densities;
   (c) two NLP solvers or algorithms; and
   (d) two derivative methods.
7. Reconstruct the sequential baseline using exactly the same model, objective, constraints, tolerances, and information assumptions as the CCD problem.
8. Compute the verified improvement

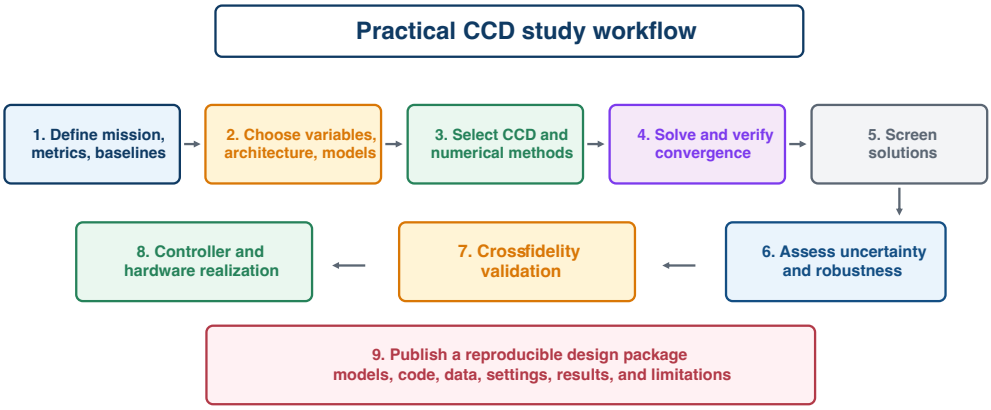
$$\eta_{\text{verified}} = \frac{J_{\text{seq,verified}} - J_{\text{CCD,verified}}}{J_{\text{seq,verified}}}.$$

9. Test both designs under uncertainty and at a higher model-fidelity level.
10. Define explicit acceptance criteria for

$$e_J, \qquad e_x, \qquad v_{\max}, \qquad \eta_{\text{verified}}.$$

11. Classify the final conclusion as one of the following:
    (a) numerically reproduced and physically validated;
    (b) numerically reproduced but not physically validated;
    (c) partially reproduced; or
    (d) not reproduced.
12. Explain why an optimizer success message and a low reported objective are insufficient evidence for a credible CCD result.

Practical CCD Workflow and Examples



Nine-step workflow

1. **Define mission, metrics, and baselines.** Quantify the value of coordination against at least one fair conventional design.
2. **Choose variables, architecture, and models.** Identify flexible plant, control, information, and architecture decisions; document model limits.
3. **Select CCD and numerical architectures.** Coordination and discretization are separate choices.
4. **Solve and verify convergence.** Use scaling, derivatives, sparse structure, multiple starts, and mesh refinement; record settings.
5. **Screen the result physically.** Inspect values, active bounds, trajectories, actuator demand, stability, and failure modes.
6. **Assess uncertainty and robustness.** Test scenarios and uncertain parameters beyond nominal optimization.
7. **Perform cross-fidelity and independent validation.** Use detailed models, independent implementations, or new data.
8. **Realize controller and hardware.** Implement sampling, estimation, actuator dynamics, saturation logic, and real-time limits.
9. **Publish a reproducible package.** Enable others to recreate the study and understand its limitations.

Example 8.1: suspension architectures

Using the suspension objective, compare:

- **Sequential:** optimize k_s and c_s with $F = 0$, freeze the plant, then optimize feedback or force trajectory.
- **Nested:** solve the specified control problem for every proposed plant.
- **Simultaneous:** transcribe states and controls and solve one sparse NLP containing plant and trajectory decisions.

Report total objective and every component, plant and controller values, peak travel and force, acceleration and tire-deflection trajectories, computation measures, mesh convergence, and independent simulation. Similar nested and simultaneous solutions support the theoretical equivalence and both implementations. Differences require investigation of inner convergence, local minima, feasibility, discretization, and controller representation.

Example 8.2: robust suspension

For three road profiles and three payloads, define nine scenarios with probabilities p_s:

$$J_R(\mathbf{x}_p, \mathbf{x}_c) = \sum_{s=1}^{9} p_s J_s(\mathbf{x}_p, \mathbf{x}_c) + \lambda \max_s J_s(\mathbf{x}_p, \mathbf{x}_c).$$

The first term rewards average performance; the second penalizes a weak worst case. Enforce actuator and travel limits in all scenarios. Compare nominal and robust designs on all cases, including the nominal penalty, worst-case improvement, violations, and design changes.

Example 8.3: surrogate-assisted wind CCD

From high-fidelity histories of rotor speed, platform pitch, tower moment, generator power, and controller signals:

1. sample wind, wave, plant, and controller conditions;
2. run high-fidelity closed-loop simulations;
3. identify a reduced or derivative-function surrogate;
4. validate histories and performance metrics;
5. optimize controller or plant–controller design;
6. re-evaluate the optimum at high fidelity;
7. add infill cases near disagreement or promising designs; and
8. complete multi-seed and extreme-condition validation.

A metric-only surrogate is inadequate when transient loads govern decisions or constraints.

Common failure modes

1. unfair baselines with different models, objectives, or constraints;
2. ideal actuators with unlimited rate, energy, bandwidth, or thermal capacity;
3. overfitting to one environmental realization;
4. unverified mesh or tolerance convergence;
5. surrogate extrapolation;
6. metric-only validation that misses peaks or constraints;
7. hidden assumptions about states, preview, or computation;
8. incomplete code, data, units, and settings; and
9. reporting a low objective without its physical mechanism.

Chapter Summary and Problems

Summary

The active-suspension tutorial connected a dynamic model to a system-level CCD formulation and showed how to compare sequential, nested, and simultaneous architectures. The wind–wave application extended the principles to expensive multiphysics systems, many operating conditions, and multiple controllers. Scenario, worst-case, chance-constrained, and risk-aware methods addressed uncertainty. Staged fidelity and surrogates supported efficient exploration without replacing high-fidelity checks. Hardware constraints, reproducibility, and validation completed the path from an optimization result to credible engineering evidence.

CCD is not complete when the optimizer stops. It is complete when the coordinated design is explained, numerically verified, tested against uncertainty, checked at suitable fidelity, translated into realizable hardware and control, and documented reproducibly.

Key terms

Practical CCD study; active suspension; fair baseline; system-level objective; nominal optimum; robust optimization; scenario optimization; chance constraint; CVaR; aleatory and epistemic uncertainty; uncertain control co-design (UCCD); worst-case robust CCD (WCR-UCCD); epigraph reformulation; matched and mismatched uncertainty; model fidelity; surrogate model; derivative function surrogate model (DFSM); linear parameter-varying (LPV) surrogate; multi-fidelity optimization; infill; cross-fidelity validation; software-in-the-loop; hardware-in-the-loop; passive safety; trilevel architecture-plant-control design; reproducibility; verification; validation.

Problems

1. **Detailed active-suspension CCD.** A quarter car satisfies $m_s\ddot{z}_s = -k_s(z_s - z_u) - c_s(\dot{z}_s - \dot{z}_u) + F$, $m_u\ddot{z}_u = k_s(z_s - z_u) + c_s(\dot{z}_s - \dot{z}_u) - k_t(z_u - r) - F$. Formulate a simultaneous CCD problem over spring geometry, damping, actuator rating, and $F(t)$ that minimizes ride acceleration and control energy while enforcing suspension travel, tire load, fatigue, buckling, packaging, and passive-failure constraints.

2. **Chance-constrained suspension travel.** Let uncertain parameters $\theta = [m_s, k_t, c_s]^T \sim \mathcal{N}(\bar{\theta}, \Sigma)$ and $g(t, \theta) = |z_s(t) - z_u(t)| - r_{\max}$. Derive a first-order, two-sided deterministic approximation guaranteeing $\mathbb{P}[g(t_j, \theta) \leq 0 \; \forall j] \geq 0.999$ using sensitivity propagation and an explicit allocation of risk across time nodes.

3. **Surrogate selection under a fixed budget.** A DFSM surrogate has structural bias B_D^2, variance $\alpha p/N$, and evaluation cost c_D, while an LSTM has bias $B_L^2 < B_D^2$, variance $\beta W/N$, and cost $c_L > c_D$. Derive a budget-dependent decision boundary that selects the model giving the lowest expected validation error after accounting for training, optimization evaluations, and iterative infill.

4. **Trust-region multi-fidelity CCD.** With low- and high-fidelity objectives $J_L(z)$ and $J_H(z)$, construct an additive first-order discrepancy model inside $\|z - z_k\| \leq \Delta_k$ and derive fully-linear error constants and an acceptance rule sufficient for convergence to first-order stationarity of J_H.

5. **Fatigue-limited spring design.** A helical spring has $k_s = Gd^4/(8D^3N_a)$ and corrected shear stress $\tau = K_w(D/d)8FD/(\pi d^3)$. Formulate and solve the minimum-mass spring design subject to target stiffness, Soderberg fatigue, spring-index, buckling, and packaging constraints, identifying the active set as load amplitude varies.

6. **Floating-wind-turbine co-design.** A linear parameter-varying model is $\dot{x} = A(V, p)x + B_u(V, p)u + B_w(V, p)w$, where p contains platform and tower variables. Formulate a multi-operating-point CCD problem minimizing levelized cost of energy while constraining fatigue-equivalent loads, platform pitch, generator speed, actuator usage, stability margin, and annual-energy production.

7. **Hardware-aware implementation.** A digital controller has sample period h, computation delay τ_d, quantization step Δ_y, and actuator rate limit $\dot{u}_{\max}$. Derive an augmented closed-loop model and one robust fea-

sibility certificate that guarantees stability and constraint satisfaction for all admissible delays and quantization errors.

8. **Statistical validation of CCD benefit.** Given paired sequential and CCD performance samples over M common uncertainty realizations and N multistart solutions per formulation, derive a hierarchical bootstrap confidence interval for the percentage improvement and a preregistered acceptance rule that accounts for optimization-path variability, feasibility violations, and multiple metrics.

9. **Architecture-value attribution.** For design domains $D = \{\text{architecture}, \text{plant}, \text{control}\}$, define a coalition value as the verified improvement obtained when only domains in $S \subseteq D$ are optimized. Derive all three Shapley values and specify the minimum set of controlled optimization runs required to estimate them without confounding domain interactions.

10. **Reproducible end-to-end study.** For an actively controlled engineering system of your choice with explicit dynamics $\dot{x} = f(x, u, p, w)$, design one complete computational experiment that compares a sequential baseline with CCD and yields a defensible claim through formulation disclosure, derivative checks, mesh convergence, multistart analysis, uncertainty validation, high-fidelity verification, and archived solver metadata.

References and further reading

1. Allison, J. T., Guo, T., & Han, Z. (2014). Co-design of an active suspension using simultaneous dynamic optimization. *Journal of Mechanical Design, 136*(8), Article 081003.

2. Allison, J. T., & Herber, D. R. (2014). Multidisciplinary design optimization of dynamic engineering systems. *AIAA Journal, 52*(4), 691–710. DOI: 10.2514/1.J052182

3. Deshmukh, A. P., Herber, D. R., & Allison, J. T. (2015). Bridging the gap between open-loop and closed-loop control in co-design: A framework for complete optimal plant and control architecture design. In *2015 American Control Conference (ACC)* (pp. 4916–4922).

4. Herber, D. R. (2017). *Advances in combined architecture, plant, and control design* (Doctoral dissertation, University of Illinois at Urbana–Champaign). Illinois Digital Environment for Access to Learning and Scholarship. https://www.ideals.illinois.edu/items/105359

5. Herber, D. R., & Allison, J. T. (2019). Nested and simultaneous solution strategies for general combined plant and control design problems. *Journal of Mechanical Design, 141*(1), Article 011402. DOI: 10.1115/1.4040705

6. Sundarrajan, A. K., & Herber, D. R. (2021). Towards a fair comparison between the nested and simultaneous control co-design methods using an active suspension case study. In *2021 American Control Conference (ACC)* (pp. 358–365).

7. Azad, S., & Herber, D. R. (2023). An overview of uncertain control co-design formulations. *Journal of Mechanical Design, 145*(9), Article 091709. DOI: 10.1115/1.4062753

8. Martins, J. R. R. A., & Ning, A. (2021). *Engineering design optimization*. Cambridge University Press.

9. Betts, J. T. (2010). *Practical methods for optimal control and estimation using nonlinear programming* (2nd ed.). Society for Industrial and Applied Mathematics. DOI: 10.1137/1.9780898718577

10. Lee, Y. H., Bayat, S., & Allison, J. T. (2025). Wind turbine control co-design using dynamic system derivative function surrogate model (DFSM) based on OpenFAST linearization. *Applied Energy, 396*, Article 126203. ScienceDirect article

11. Sundarrajan, A. K., & Herber, D. R. (2025). Comparison of data-driven modeling approaches for control optimization of floating offshore wind turbines. In *Proceedings of the ASME 2025 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference (IDETC-CIE2025)*, Article DETC2025-167963.

12. Bayat, S., Peterson, C., Lee, Y. H., Iori, J., & Allison, J. T. (2026). Advancing wind turbines through control co-design: An integrative review. *Applied Energy, 416*, Article 127951. ScienceDirect article

13. Mayne, D. Q., Rawlings, J. B., Rao, C. V., & Scokaert, P. O. M. (2000). Constrained model predictive control: Stability and optimality. *Automatica, 36*(6), 789–814. DOI: 10.1016/S0005-1098(99)00214-9