॥ श्री गणेशाय नमः ॥

Optimal Control

Varad Vaidya

October 1, 2025

Abstract

Contents

Lecture 1: Introduction to State-Space Dynamics

1.1 Continuous-Time Dynamics

Welcome to the first lecture on Optimal Control! Today we will begin our journey by formalizing the concept of a dynamical system. The core idea is to represent the evolution of a system over time using a mathematical model.

Definition 1.1 (State-Space Representation). A general continuous-time smooth dynamical system can be described by a first-order ordinary differential equation (ODE) of the form:

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t), t) \tag{1.1}$$

where:

- $\mathbf{x}(t) \in \mathbb{R}^n$ is the **state vector** of the system at time t. It is a complete summary of the system's history, meaning that the state at time t is sufficient to predict the future evolution of the system given the future inputs.
- $\mathbf{u}(t) \in \mathbb{R}^m$ is the **input vector** (or control vector) at time t. These are the external signals we can manipulate to influence the system's behavior.
- $f : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R} \to \mathbb{R}^n$ is a smooth function called the **dynamics function** or vector field. It maps the current state, input, and time to the time derivative of the state.

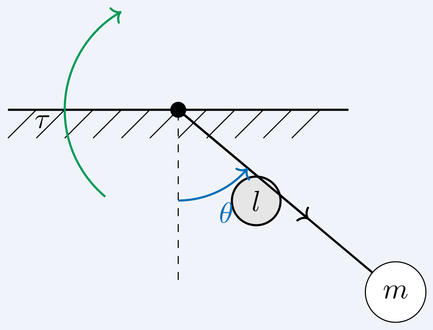
For many systems, especially mechanical ones, the state vector $\mathbf{x}$ has a specific structure. It's often composed of the system's configuration and its velocity.

Notation. For a mechanical system, we typically define the state as:

$$\mathbf{x} = \begin{bmatrix} \mathbf{q} \\ \mathbf{v} \end{bmatrix}$$

where $\mathbf{q}$ is the **configuration** or "pose" of the system, and $\mathbf{v}$ is its corresponding **velocity**. Note that the configuration space is not always a simple vector space like $\mathbb{R}^k$; it can be a more complex manifold, like a circle S^1 or the special orthogonal group $SO(3)$ for rotations.

Example 1.1 (The Simple Pendulum). Let's consider a simple pendulum, which consists of a point mass m attached to a massless rod of length l, pivoting frictionlessly. A torque τ can be applied at the pivot.



The equation of motion for this system can be derived from Newton's second law for rotation, $\sum \tau = I\alpha$, where $I = ml^2$ is the moment of inertia and $\alpha = \ddot{\theta}$ is the angular acceleration. The torques acting on the mass are the applied torque τ and the torque due to gravity, $-mgl\sin(\theta)$. This gives us:

$$ml^2\ddot{\theta} + mgl\sin(\theta) = \tau \tag{1.2}$$

To put this into our state-space form, we define our state and input:

- Configuration: $q = \theta \in S^1$ (the space of angles, a circle)
- Velocity: $v = \dot{\theta} \in \mathbb{R}$
- State: $\mathbf{x} = \begin{bmatrix}\theta \\ \dot{\theta}\end{bmatrix} \in S^1 \times \mathbb{R}$ (a cylinder)
- Input: $u = \tau \in \mathbb{R}$

Now we can write the dynamics $\dot{\mathbf{x}} = f(\mathbf{x}, u)$:

$$\dot{\mathbf{x}} = \frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix}\theta \\ \dot{\theta}\end{bmatrix} = \begin{bmatrix}\dot{\theta} \\ \ddot{\theta}\end{bmatrix}$$

From Equation 1.2, we can solve for $\ddot{\theta}$:

$$\ddot{\theta} = \frac{1}{ml^2}(\tau - mgl\sin(\theta))$$

Substituting this back, we get the state-space representation:

$$\dot{\mathbf{x}} = f(\mathbf{x}, u) = \begin{bmatrix} x_2 \\ \frac{1}{ml^2}(u - mgl\sin(x_1)) \end{bmatrix} \tag{1.3}$$

where we have used $x_1 = \theta$ and $x_2 = \dot{\theta}$.

1.2 Control-Affine Systems

A very common and important class of nonlinear systems are those that are "affine" in the control input.

Definition 1.2 (Control-Affine System). A system is called **control-affine** if its dynamics can be written in the form:

$$\dot{\mathbf{x}} = f_0(\mathbf{x}) + B(\mathbf{x})\mathbf{u} \tag{1.4}$$

Here, $f_0(\mathbf{x})$ is called the **drift vector field**, representing the system's natural dynamics when no control is applied. The matrix $B(\mathbf{x})$ is the **input Jacobian**, which describes how the control input $\mathbf{u}$ affects the state's velocity at a given state $\mathbf{x}$.

Note. Most mechanical systems can be expressed in this form. For our pendulum example Equation 1.3, we can separate the terms to see its control-affine structure:

$$\dot{\mathbf{x}} = \underbrace{\begin{bmatrix} x_2 \\ -\frac{g}{l}\sin(x_1) \end{bmatrix}}_{f_0(\mathbf{x})} + \underbrace{\begin{bmatrix} 0 \\ \frac{1}{ml^2} \end{bmatrix}}_{B(\mathbf{x})} u$$

Notice that in this case, the input Jacobian B is constant.

1.3 Manipulator Dynamics and Euler-Lagrange

The dynamics of robotic manipulators (and mechanical systems in general) have a well-defined structure.

Theorem 1.1 (Manipulator Equation)**.** The dynamics of a fully-actuated mechanical system can be written as:

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) = \mathbf{B}(\mathbf{q})\mathbf{u} + \tau_{ext} \tag{1.5}$$

where:

- $\mathbf{M}(\mathbf{q})$ is the symmetric, positive-definite **mass matrix**.
- $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}}$ represents Coriolis and centrifugal forces.
- $\mathbf{g}(\mathbf{q})$ is the vector of gravitational forces.
- $\mathbf{B}(\mathbf{q})$ is the input mapping, and τ_{ext} are other external forces.

This is a second-order ODE. To convert it to a first-order state-space form, we can again let $\mathbf{x} = [\mathbf{q}^\top, \dot{\mathbf{q}}^\top]^\top$. Then:

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{\mathbf{q}} \\ \mathbf{M}(\mathbf{q})^{-1}(\mathbf{B}(\mathbf{q})\mathbf{u} + \tau_{ext} - \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} - \mathbf{g}(\mathbf{q})) \end{bmatrix}$$

This is also in the control-affine form.

Intuition. This structured form is not arbitrary; it is a direct consequence of the Euler-Lagrange equations from classical mechanics. The Lagrangian $\mathcal{L}$ of a system is defined as the difference between its kinetic energy T and potential energy U:

$$\mathcal{L}(\mathbf{q}, \dot{\mathbf{q}}) = T(\mathbf{q}, \dot{\mathbf{q}}) - U(\mathbf{q})$$

For mechanical systems, the kinetic energy is $T = \frac{1}{2}\dot{\mathbf{q}}^\top \mathbf{M}(\mathbf{q})\dot{\mathbf{q}}$. The Euler-Lagrange equation then gives the dynamics:

$$\frac{\mathrm{d}}{\mathrm{dt}}\left(\frac{\partial \mathcal{L}}{\partial \dot{\mathbf{q}}}\right) - \frac{\partial \mathcal{L}}{\partial \mathbf{q}} = \tau_{\text{gen}}$$

where τ_{gen} are the generalized forces acting on the system (like control inputs and external forces). Working through this derivation yields the manipulator Equation 1.5.

1.4 Linear Systems

A particularly simple yet powerful class of systems are linear systems.

Definition 1.3 (Linear Time-Varying System)**.** A system is linear if its dynamics can be written as:

$$\dot{\mathbf{x}}(t) = \mathbf{A}(t)\mathbf{x}(t) + \mathbf{B}(t)\mathbf{u}(t) \tag{1.6}$$

If the matrices $\mathbf{A}$ and $\mathbf{B}$ are constant, the system is called **Linear Time-Invariant (LTI)**.

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

Linear systems are fundamental in control theory, not just because they are easy to analyze, but because they can serve as local approximations of nonlinear systems.

As previously seen. We can linearize a nonlinear system $\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u})$ around a nominal trajectory $(\bar{\mathbf{x}}(t), \bar{\mathbf{u}}(t))$. Let $\delta\mathbf{x} = \mathbf{x} - \bar{\mathbf{x}}$ and $\delta\mathbf{u} = \mathbf{u} - \bar{\mathbf{u}}$. A first-order Taylor expansion gives:

$$\dot{\bar{\mathbf{x}}} + \delta\dot{\mathbf{x}} \approx f(\bar{\mathbf{x}}, \bar{\mathbf{u}}) + \underbrace{\left.\frac{\partial f}{\partial \mathbf{x}}\right|_{\bar{\mathbf{x}},\bar{\mathbf{u}}}}_{\mathbf{A}(t)} \delta\mathbf{x} + \underbrace{\left.\frac{\partial f}{\partial \mathbf{u}}\right|_{\bar{\mathbf{x}},\bar{\mathbf{u}}}}_{\mathbf{B}(t)} \delta\mathbf{u}$$

Since $\dot{\bar{\mathbf{x}}} = f(\bar{\mathbf{x}}, \bar{\mathbf{u}})$ (it's a valid trajectory), the dynamics of the error $\delta\mathbf{x}$ are approximately linear:

$$\delta\dot{\mathbf{x}} \approx \mathbf{A}(t)\delta\mathbf{x} + \mathbf{B}(t)\delta\mathbf{u}$$

This process of linearization is a cornerstone of control design for nonlinear systems, and we will revisit it many times in this course.

Lecture 2: Equilibria, Stability, and Simulation

As previously seen. In our last lecture, we introduced the state-space representation for continuous-time dynamical systems, $\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u})$. We examined the structure of control-affine systems and general manipulator dynamics derived from Euler-Lagrange principles. We concluded by defining linear systems and showing how they arise from the linearization of nonlinear systems.

2.1 Equilibrium Points

A fundamental concept in analyzing dynamical systems is the notion of an equilibrium point, a state where the system can remain indefinitely if undisturbed.

Definition 2.1 (Equilibrium Point). An equilibrium point (or fixed point) $\mathbf{x}^\star$ of a continuous-time system $\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u})$ for a constant control input $\mathbf{u}^\star$ is a state that satisfies:

$$f(\mathbf{x}^\star, \mathbf{u}^\star) = 0 \tag{2.1}$$

This means that if the system starts at $\mathbf{x}^\star$ with control $\mathbf{u}^\star$, its state derivative is zero, and it will not move.

Example 2.1 (Pendulum Equilibria). Consider the unforced pendulum ($u = 0$). Its dynamics are:

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{\theta} \\ -\frac{g}{l}\sin(\theta) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

This system of equations requires $\dot{\theta} = 0$ and $\sin(\theta) = 0$. This occurs when $\theta = k\pi$ for any integer k. The two distinct physical equilibria are:

- $\mathbf{x}_1^\star = [0, 0]^\top$: The pendulum is hanging straight down, at rest.
- $\mathbf{x}_2^\star = [\pi, 0]^\top$: The pendulum is balanced perfectly upright, at rest.

A basic control problem is to create a new equilibrium point. For instance, can we make the pendulum hang at $\theta = \pi/2$? We need to find a constant control $u^\star$ such that $f([\pi/2, 0]^\top, u^\star) = 0$.

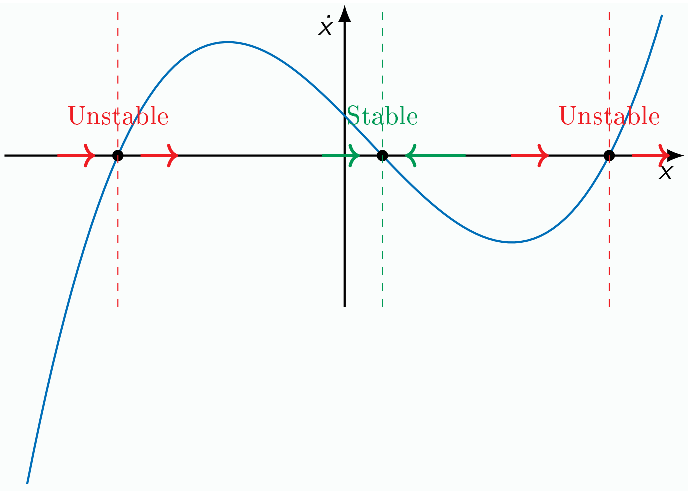
$$\dot{\mathbf{x}} = \begin{bmatrix} 0 \\ \frac{1}{ml^2}(u^\star - mgl\sin(\pi/2)) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

This requires $u^\star - mgl(1) = 0$, so $u^\star = mgl$. A constant torque can indeed hold the pendulum at this new equilibrium.

2.2 Stability of Equilibria

It's not enough to know where the equilibria are; we need to know if they are stable. If the system is perturbed slightly from an equilibrium, does it return, or does it move away?

Intuition (1D Systems). For a 1D system $\dot{x} = f(x)$, we can visualize stability by plotting $\dot{x}$ vs. x.



An equilibrium $x^\star$ is **locally stable** if, for points x near $x^\star$, the dynamics push the state back towards $x^\star$. This happens when the slope of $f(x)$ at $x^\star$ is negative.

- If $\frac{\partial f}{\partial x}|_{x^\star} < 0$, the equilibrium is stable.
- If $\frac{\partial f}{\partial x}|_{x^\star} > 0$, the equilibrium is unstable.
- If $\frac{\partial f}{\partial x}|_{x^\star} = 0$, the analysis is inconclusive (marginally stable).

This concept generalizes to higher dimensions. For a system $\dot{\mathbf{x}} = f(\mathbf{x})$, we linearize the dynamics around an equilibrium $\mathbf{x}^\star$. Let $\delta\mathbf{x} = \mathbf{x} - \mathbf{x}^\star$.

$$\delta\dot{\mathbf{x}} \approx \left.\frac{\partial f}{\partial \mathbf{x}}\right|_{\mathbf{x}^\star} \delta\mathbf{x} = \mathbf{A}\delta\mathbf{x}$$

The stability of the nonlinear system near $\mathbf{x}^\star$ is determined by the stability of this linear system.

Theorem 2.1 (Stability by Linearization). An equilibrium point $\mathbf{x}^\star$ of $\dot{\mathbf{x}} = f(\mathbf{x})$ is:

- **Asymptotically Stable** if all eigenvalues of the Jacobian $\mathbf{A} = \frac{\partial f}{\partial \mathbf{x}}|_{\mathbf{x}^\star}$ have strictly negative real parts ($\text{Re}(\lambda_i) < 0$).
- **Unstable** if at least one eigenvalue of $\mathbf{A}$ has a strictly positive real part ($\text{Re}(\lambda_i) > 0$).
- **Marginally Stable** if all eigenvalues have non-positive real parts ($\text{Re}(\lambda_i) \leq 0$) and at least one has a zero real part.

Example 2.2 (Pendulum Stability Analysis). The Jacobian of the unforced pendulum dynamics is:

$$\mathbf{A} = \frac{\partial f}{\partial \mathbf{x}} = \begin{bmatrix} 0 & 1 \\ -\frac{g}{l}\cos(\theta) & 0 \end{bmatrix}$$

1. **At $\mathbf{x}_1^\star = [0,0]^\top$ (hanging down)**:

$$\mathbf{A}|_{\theta=0} = \begin{bmatrix} 0 & 1 \\ -\frac{g}{l} & 0 \end{bmatrix}$$

The eigenvalues are $\lambda = \pm i\sqrt{g/l}$. Since the real parts are zero, this equilibrium is **marginally stable**. A small push will cause it to oscillate forever (in this frictionless model). If we added any damping (e.g., $u = -K_d\dot{\theta}$), the eigenvalues would move into the left-half plane, making it asymptotically stable.

2. **At $\mathbf{x}_2^\star = [\pi, 0]^\top$ (upright):**

$$\mathbf{A}|_{\theta=\pi} = \begin{bmatrix} 0 & 1 \\ \frac{g}{l} & 0 \end{bmatrix}$$

The eigenvalues are $\lambda = \pm\sqrt{g/l}$. Since there is one positive real eigenvalue, this equilibrium is **unstable**. Any small perturbation will cause the pendulum to fall.

2.3 Discrete-Time Dynamics and Simulation

Analytically solving nonlinear ODEs is rarely possible. We rely on numerical simulation to understand their behavior. This requires converting the continuous-time model into a discrete-time one.

Definition 2.2 (Discrete-Time System). An explicit discrete-time system has the form:

$$\mathbf{x}_{k+1} = f_d(\mathbf{x}_k, \mathbf{u}_k) \tag{2.2}$$

where $\mathbf{x}_k$ is the state at time step k, and f_d is the discrete-time dynamics function that maps the current state and input to the next state.

The process of converting a continuous-time model f to a discrete-time model f_d is called **discretization** or **integration**.

2.3.1 The Forward Euler Method

The simplest way to discretize is to approximate the derivative $\dot{\mathbf{x}}$ with a finite difference: $\dot{\mathbf{x}} \approx \frac{\mathbf{x}_{k+1}-\mathbf{x}_k}{h}$, where h is the time step.

$$\frac{\mathbf{x}_{k+1} - \mathbf{x}_k}{h} = f(\mathbf{x}_k, \mathbf{u}_k) \Rightarrow \mathbf{x}_{k+1} = \mathbf{x}_k + hf(\mathbf{x}_k, \mathbf{u}_k)$$

This is the **Forward Euler** integration scheme.

Code 2.1 (Julia Notebook: Forward Euler Simulation).
The provided Julia notebook first simulates the pendulum using this method. When you run this code, you will observe that the amplitude of the pendulum's oscillation grows with each swing. Eventually, it "blows up" with the angle going to infinity. This is because the Forward Euler method is not energy-preserving. At each step, it adds a small amount of energy to this conservative system, leading to catastrophic instability. This is a critical lesson: **never use Forward Euler for simulating mechanical systems**.

2.3.2 Stability of Discrete-Time Systems

Stability for discrete-time systems is defined by how the system behaves under repeated application of the map f_d. For a linear discrete-time system $\mathbf{x}_{k+1} = \mathbf{A}_d\mathbf{x}_k$, stability requires that $\lim_{k\to\infty} \mathbf{A}_d^k\mathbf{x}_0 = 0$. This condition is met if and only if all eigenvalues of $\mathbf{A}_d$ are strictly inside the unit circle in the complex plane.

Theorem 2.2 (Discrete-Time Stability). A discrete-time linear system $\mathbf{x}_{k+1} = \mathbf{A}_d\mathbf{x}_k$ is stable if and only if $|\lambda_i| < 1$ for all eigenvalues λ_i of $\mathbf{A}_d$.

For the Forward Euler method, the discrete-time Jacobian is $\mathbf{A}_d = \frac{\partial f_d}{\partial \mathbf{x}} = \mathbf{I} + h\mathbf{A}$.

Code 2.2 (Julia Notebook: Stability of Forward Euler).
The notebook analyzes the stability of the Forward Euler discretization of the pendulum around its stable equilibrium ($\theta = 0$). It computes the eigenvalues of $\mathbf{A}_d = \mathbf{I} + h\mathbf{A}|_{\theta=0}$. The result shows that the magnitude of these eigenvalues is always slightly greater than 1 for any $h > 0$. This analytically confirms why the simulation is unstable: the discretization method itself is unstable for this type of system (systems with purely imaginary eigenvalues).

2.3.3 The Runge-Kutta 4 (RK4) Method

A much better explicit integrator is the 4th-order Runge-Kutta (RK4) method. Instead of taking a single step in the direction of the derivative at the start of the interval, it evaluates the derivative at several points within the interval to get a much more accurate estimate of the next state.

Algorithm 1: RK4 Step

```
Input: Current state $\mathbf{x}_k$, time step $h$
Output: Next state $\mathbf{x}_{k+1}$
1 $\mathbf{k}_1 = f(\mathbf{x}_k)$
2 $\mathbf{k}_2 = f(\mathbf{x}_k + \frac{h}{2}\mathbf{k}_1)$
3 $\mathbf{k}_3 = f(\mathbf{x}_k + \frac{h}{2}\mathbf{k}_2)$
4 $\mathbf{k}_4 = f(\mathbf{x}_k + h\mathbf{k}_3)$
5 $\mathbf{x}_{k+1} = \mathbf{x}_k + \frac{h}{6}(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4)$
```

Code 2.3 (Julia Notebook: RK4 Simulation and Stability).
The notebook also simulates the pendulum with RK4. The result is a stable oscillation. The energy is much better conserved, and the amplitude remains constant. Analyzing the eigenvalues of the RK4 discrete Jacobian $\mathbf{A}_d$ shows their magnitudes are extremely close to 1. This indicates that RK4 is a much more stable and accurate choice for this type of simulation. The slight deviation from 1 explains the very slow energy drift that still occurs over very long simulations.

Note. The notebook also includes a **Backward Euler** integrator. This is an *implicit* method, meaning it solves an equation $\mathbf{x}_{k+1} = \mathbf{x}_k + hf(\mathbf{x}_{k+1})$ at each step. Implicit methods are generally very stable, but often introduce "numerical damping," causing energy to decrease even in a conservative system. This can be useful for stiff systems or when stability is paramount, but it may not accurately reflect the physics.

2.4 Key Takeaways on Simulation

- Choosing the right numerical integrator is crucial for obtaining meaningful simulation results.
- Explicit methods like Forward Euler can be unstable and add energy to conservative systems. Avoid them.
- Higher-order methods like RK4 provide a much better balance of accuracy, stability, and computational cost for many problems in robotics.

- Always sanity-check your simulations by monitoring physical quantities that should be conserved, such as total energy or momentum.

2.5 Supplementary Concepts

2.5.1 Lyapunov Stability Theory

Consider an autonomous system $\dot{\mathbf{x}} = f(\mathbf{x})$ with an equilibrium $\mathbf{x}^\star$ (i.e. $f(\mathbf{x}^\star) = 0$). Define the error $\mathbf{e} = \mathbf{x} - \mathbf{x}^\star$ and rewrite about the origin (w.l.o.g. take $\mathbf{x}^\star = 0$ by translation). We recall several nested stability notions.

Definition 2.3 (Stability Notions). Let $\dot{\mathbf{x}} = f(\mathbf{x})$ with equilibrium at 0.

- (Lyapunov or *stable in the sense of Lyapunov*): For every $\varepsilon > 0$ there exists $\delta(\varepsilon) > 0$ s.t. $\|\mathbf{x}(0)\| < \delta \Rightarrow \|\mathbf{x}(t)\| < \varepsilon$ for all $t \geq 0$.
- (Asymptotically Stable): Stable, and $\exists\, \delta' > 0$ s.t. $\|\mathbf{x}(0)\| < \delta' \Rightarrow \lim_{t\to\infty} \mathbf{x}(t) = 0$.
- (Exponentially Stable): $\exists\, c > 0, \alpha > 0, \delta'' > 0$ s.t. $\|\mathbf{x}(0)\| < \delta'' \Rightarrow \|\mathbf{x}(t)\| \leq ce^{-\alpha t}\|\mathbf{x}(0)\|$ for all $t \geq 0$.
- (Globally Asymptotically / Exponentially Stable): The above properties hold for all initial conditions (no radius restriction).

Definition 2.4 (Positive (Semi)Definite, Radially Unbounded). A continuous scalar function $V : \mathbb{R}^n \to \mathbb{R}$ is

- Positive definite if $V(0) = 0$ and $V(\mathbf{x}) > 0$ for all $\mathbf{x} \neq 0$.
- Positive semidefinite if $V(\mathbf{x}) \geq 0$ and $V(0) = 0$.
- Radially unbounded (proper) if $\|\mathbf{x}\| \to \infty \Rightarrow V(\mathbf{x}) \to \infty$.

Definition 2.5 (Lyapunov Function). A continuously differentiable V is a (strict) Lyapunov function for the equilibrium if it is positive definite and its orbital derivative $\dot{V}(\mathbf{x}) := \nabla V(\mathbf{x})^\top f(\mathbf{x})$ is negative definite (or negative semidefinite for weaker conclusions) in a neighborhood $\mathcal{D}$ of the origin.

Theorem 2.3 (Lyapunov Stability Theorem). If there exists a positive definite function V with $\dot{V}(\mathbf{x}) \leq 0$ (negative semidefinite) in a neighborhood $\mathcal{D}$ of the origin, then the equilibrium is Lyapunov stable. If additionally $\dot{V}(\mathbf{x}) < 0$ (negative definite) for all $\mathbf{x} \in \mathcal{D} \setminus \{0\}$, the equilibrium is asymptotically stable. If V and $-\dot{V}$ satisfy quadratic bounds $\alpha_1\|\mathbf{x}\|^2 \leq V(\mathbf{x}) \leq \alpha_2\|\mathbf{x}\|^2$ and $\dot{V}(\mathbf{x}) \leq -\alpha_3\|\mathbf{x}\|^2$, exponential stability follows.

Theorem 2.4 (LaSalle Invariance Principle). Let $\Omega \subset \mathbb{R}^n$ be compact, positively invariant, and suppose $V : \Omega \to \mathbb{R}$ is continuous, C^1 on interior, with $\dot{V} \leq 0$ on Ω. Let $E = \{\mathbf{x} \in \Omega : \dot{V}(\mathbf{x}) = 0\}$ and let $\mathcal{M}$ be the largest invariant subset of E. Then every trajectory starting in Ω approaches $\mathcal{M}$ as $t \to \infty$. If $\mathcal{M} = \{0\}$, the origin is asymptotically stable. If $\Omega = \mathbb{R}^n$ and V is proper, global asymptotic stability can be concluded.

Linear Systems and Lyapunov Equation. For $\dot{\mathbf{x}} = \mathbf{A}\mathbf{x}$ with $\mathbf{A}$ Hurwitz (all eigenvalues with negative real parts), for any symmetric positive definite $\mathbf{Q}$ there exists a unique symmetric positive definite $\mathbf{P}$ solving the Lyapunov equation

$$\mathbf{A}^\top \mathbf{P} + \mathbf{P}\mathbf{A} = -\mathbf{Q}.$$

Then $V(\mathbf{x}) = \mathbf{x}^\top \mathbf{P}\mathbf{x}$ is a quadratic Lyapunov function certifying exponential stability.

2.5.2 Basin / Region of Attraction

Definition 2.6 (Basin (Region) of Attraction)**.** For an asymptotically stable equilibrium $\mathbf{x}^\star$, its (open) basin of attraction $\mathcal{B}(\mathbf{x}^\star)$ is the set of initial conditions whose trajectories converge to $\mathbf{x}^\star$:

$$\mathcal{B}(\mathbf{x}^\star) := \{\mathbf{x}_0 \in \mathbb{R}^n : \lim_{t\to\infty} \phi_t(\mathbf{x}_0) = \mathbf{x}^\star\},$$

where ϕ_t denotes the flow map. If $\mathcal{B}(\mathbf{x}^\star) = \mathbb{R}^n$, the equilibrium is globally asymptotically stable.

Lyapunov Level-Set Inner Approximations. Suppose V is a Lyapunov function certifying asymptotic stability on domain $\mathcal{D}$. Any sublevel set $\mathcal{L}_c := \{\mathbf{x} : V(\mathbf{x}) \leq c\}$ contained in $\mathcal{D}$ with $\dot{V} < 0$ for nonzero points inside provides an inner approximation of $\mathcal{B}(0)$. One may enlarge c until tangency with $\dot{V} = 0$ occurs. For polynomial systems, sum-of-squares (SOS) optimization can automate this search.

Nonlinear Pendulum Example. Including viscous damping $\dot{\theta} = \omega$, $\dot{\omega} = -\frac{g}{l}\sin\theta - k\omega$ ($k > 0$) makes $\theta = 0$ asymptotically stable. The (mechanical) energy $E = \frac{1}{2}ml^2\omega^2 + mgl(1 - \cos\theta)$ decreases monotonically ($\dot{E} = -kml^2\omega^2 \leq 0$). The basin of attraction is all of $\mathbb{R}^2$ (global) when friction prevents escape; without damping it was only (marginally) stable, not attracting.

2.5.3 Poincaré–Bendixson Theorem

In planar continuous-time autonomous systems ($n = 2$), the long-term behaviors are heavily constrained. The Poincaré–Bendixson Theorem rules out chaos (which requires dimension ≥ 3).

Theorem 2.5 (Poincaré–Bendixson)**.** Let $\dot{\mathbf{x}} = f(\mathbf{x})$ with $f \in C^1$ on an open set $\mathcal{U} \subset \mathbb{R}^2$. Suppose a trajectory $\phi_t(\mathbf{x}_0)$ remains in a compact set $\mathcal{K} \subset \mathcal{U}$ for all $t \geq 0$ and contains no equilibrium points. Then the ω-limit set of $\mathbf{x}_0$ is a periodic orbit (limit cycle). More generally, any nonempty compact ω-limit set that contains only finitely many equilibria is either (i) an equilibrium, (ii) a periodic orbit, or (iii) a finite union of equilibria and connecting heteroclinic orbits.

Consequences. In 2D one cannot have strange attractors. For physical robot subsystems reduced to two states (e.g. simplified balancing planes), observed sustained oscillations typically imply a limit cycle certified by the theorem.

Application Template. To use Poincaré–Bendixson:

1. Find a forward-invariant compact set $\mathcal{K}$ (e.g. by trapping region, energy bounds).
2. Show the trajectory stays in $\mathcal{K}$ and does not converge to an equilibrium (e.g. by Dulac's criterion or sign of divergence arguments).
3. Conclude existence of a periodic orbit (limit cycle).

Example (Van der Pol). $\ddot{x} - \mu(1 - x^2)\dot{x} + x = 0$ with $\mu > 0$ can be written as a planar system possessing a unique stable limit cycle; trajectories enter a compact trapping region and cannot settle at the sole equilibrium (the origin is unstable for $\mu > 0$).

2.5.4 Controllability and Observability

We restrict to linear time-invariant (LTI) systems $\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}$, $\mathbf{y} = \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u}$, with $\mathbf{A} \in \mathbb{R}^{n\times n}$, $\mathbf{B} \in \mathbb{R}^{n\times m}$, $\mathbf{C} \in \mathbb{R}^{p\times n}$.

Definition 2.7 (Reachable / Controllable). The system is (state) controllable if for any $\mathbf{x}_0, \mathbf{x}_f$ there exists a finite time T and input $\mathbf{u}(t)$ steering $\mathbf{x}(0) = \mathbf{x}_0$ to $\mathbf{x}(T) = \mathbf{x}_f$. It is (origin) reachable if each state can be reached from the origin. For LTI systems these notions coincide.

Definition 2.8 (Observable). The pair $(\mathbf{A}, \mathbf{C})$ is observable if any two distinct initial states $\mathbf{x}_1(0) \neq \mathbf{x}_2(0)$ produce different output trajectories on some finite interval; equivalently the initial state is uniquely determined from knowledge of $\mathbf{y}(t)$ and $\mathbf{u}(t)$ over a finite horizon.

Kalman Rank Conditions. Define the controllability matrix

$$\mathcal{C} = \begin{bmatrix} \mathbf{B} & \mathbf{A}\mathbf{B} & \mathbf{A}^2\mathbf{B} & \cdots & \mathbf{A}^{n-1}\mathbf{B} \end{bmatrix} \in \mathbb{R}^{n\times nm}$$

and the observability matrix

$$\mathcal{O} = \begin{bmatrix} \mathbf{C} \\ \mathbf{C}\mathbf{A} \\ \mathbf{C}\mathbf{A}^2 \\ \vdots \\ \mathbf{C}\mathbf{A}^{n-1} \end{bmatrix} \in \mathbb{R}^{np\times n}.$$

Then $(\mathbf{A}, \mathbf{B})$ is controllable iff $\operatorname{rank}(\mathcal{C}) = n$; $(\mathbf{A}, \mathbf{C})$ is observable iff $\operatorname{rank}(\mathcal{O}) = n$.

PBH (Popov–Belevitch–Hautus) Tests. An equivalent spectral test: $(\mathbf{A}, \mathbf{B})$ is controllable iff $\operatorname{rank}\begin{bmatrix} \lambda\mathbf{I} - \mathbf{A} & \mathbf{B} \end{bmatrix} = n$ for all $\lambda \in \mathbb{C}$. Likewise $(\mathbf{A}, \mathbf{C})$ is observable iff $\operatorname{rank}\begin{bmatrix} \lambda\mathbf{I} - \mathbf{A} \\ \mathbf{C} \end{bmatrix} = n$ for all $\lambda \in \mathbb{C}$.

Stabilizability and Detectability. If uncontrollable modes (eigenvalues) all lie in the open left-half plane, the system is stabilizable (we can design feedback to stabilize). Dually, if unobservable modes all decay, the system is detectable (an observer can be designed with asymptotic error convergence). These relaxed properties are sufficient for many optimal control designs (e.g. LQR requires stabilizability / detectability, not full controllability / observability).

Energy / Gramian Characterization. Over horizon $[0, T]$, the controllability Gramian is

$$\mathbf{W}_c(T) = \int_0^T e^{\mathbf{A}t}\mathbf{B}\mathbf{B}^\top e^{\mathbf{A}^\top t}\, dt.$$

If $\mathbf{W}_c(T)$ is nonsingular for some (hence all sufficiently large) $T > 0$, the system is controllable. Minimum input energy to reach $\mathbf{x}_f$ from 0 in time T is $\mathbf{x}_f^\top \mathbf{W}_c(T)^{-1}\mathbf{x}_f$. Similarly, the observability Gramian $\mathbf{W}_o(T) = \int_0^T e^{\mathbf{A}^\top t}\mathbf{C}^\top\mathbf{C}e^{\mathbf{A}t}\, dt$ is nonsingular iff $(\mathbf{A}, \mathbf{C})$ is observable.

Connection to Optimal Control. Existence of a unique positive semidefinite solution to the Algebraic Riccati Equation (ARE) for LQR hinges on stabilizability and detectability. The resulting optimal feedback $\mathbf{u} = -\mathbf{K}\mathbf{x}$ ensures closed-loop Lyapunov stability via the ARE as a Lyapunov equation for the cost-to-go.

Summary. Lyapunov functions certify (local/global, asymptotic/exponential) stability and approximate basins via level sets; planar systems admit only equilibria and limit cycles as nontrivial recurrent sets (Poincaré–Bendixson), and linear controllability/observability (with their relaxed forms) determine if we can stabilize or reconstruct system states—all foundational for designing and analyzing optimal controllers.

Lecture 3: Derivatives, Root Finding, and Minimization

3.1 A Note on Notation: Derivatives as Linear Operators

To proceed, we must first establish a clear and consistent notation for derivatives, treating them as linear operators that describe the local change of a function.

Notation. Let $f : \mathbb{R}^n \to \mathbb{R}^m$ be a differentiable function. The **Jacobian** of f at a point $\mathbf{x}$ is the unique $m \times n$ matrix, denoted $\frac{\partial f}{\partial \mathbf{x}}$, that satisfies the first-order Taylor approximation:

$$f(\mathbf{x} + \delta \mathbf{x}) \approx f(\mathbf{x}) + \left.\frac{\partial f}{\partial \mathbf{x}}\right|_{\mathbf{x}} \delta \mathbf{x} \tag{3.1}$$

This convention ensures that the chain rule for Jacobians works as expected with standard matrix multiplication. If we have $h(\mathbf{z}) = f(g(\mathbf{z}))$, then:

$$\frac{\partial h}{\partial \mathbf{z}} = \left.\frac{\partial f}{\partial \mathbf{y}}\right|_{g(\mathbf{z})} \frac{\partial g}{\partial \mathbf{z}}$$

For a scalar-valued function $f : \mathbb{R}^n \to \mathbb{R}$, its Jacobian $\frac{\partial f}{\partial \mathbf{x}}$ is a $1 \times n$ row vector. For convenience, we define the **gradient** vector, $\nabla f(\mathbf{x})$, as the transpose of the Jacobian.

$$\nabla f(\mathbf{x}) := \left(\frac{\partial f}{\partial \mathbf{x}}\right)^\top \quad \text{(an } n \times 1 \text{ column vector)}$$

The **Hessian** matrix is the Jacobian of the gradient, resulting in an $n \times n$ matrix of second partial derivatives.

$$\nabla^2 f(\mathbf{x}) := \frac{\partial}{\partial \mathbf{x}} = \frac{\partial^2 f}{\partial \mathbf{x}^2} \quad \text{(an } n \times n \text{ matrix)}$$

With this, the second-order Taylor expansion of a scalar function is:

$$f(\mathbf{x} + \delta \mathbf{x}) \approx f(\mathbf{x}) + (\nabla f(\mathbf{x}))^\top \delta \mathbf{x} + \frac{1}{2} \delta \mathbf{x}^\top (\nabla^2 f(\mathbf{x})) \delta \mathbf{x} \tag{3.2}$$

3.2 Root Finding

A fundamental problem in numerical methods is root finding: for a given function $g : \mathbb{R}^n \to \mathbb{R}^n$, find a point $\mathbf{x}^\star$ such that $g(\mathbf{x}^\star) = 0$.

Example 3.1. Finding the equilibrium point of a dynamical system $\dot{\mathbf{x}} = f(\mathbf{x})$ is a root-finding problem. Similarly, as we saw last lecture, solving the implicit equation in a Backward Euler step, $\mathbf{x}_{k+1} - \mathbf{x}_k - hf(\mathbf{x}_{k+1}) = 0$, is also a root-finding problem for $\mathbf{x}_{k+1}$.

3.2.1 Fixed-Point Iteration

A closely related problem is finding a **fixed point**, i.e., an $\mathbf{x}^\star$ such that $g(\mathbf{x}^\star) = \mathbf{x}^\star$. We can always convert a root-finding problem $r(\mathbf{x}) = 0$ to a fixed-point problem by defining $g(\mathbf{x}) = \mathbf{x} - r(\mathbf{x})$.

The simplest method for finding a fixed point is **fixed-point iteration**:

$$\mathbf{x}_{k+1} = g(\mathbf{x}_k)$$

This method is only guaranteed to converge if the fixed point is stable (i.e., $|\text{eig}(\frac{\partial g}{\partial \mathbf{x}})| < 1$) and the initial guess is within the basin of attraction. The convergence rate is typically linear, which can be quite slow.

3.2.2 Newton's Method

A much more powerful and faster method is **Newton's method**. It works by iteratively solving a linearized version of the root-finding problem. Given a guess $\mathbf{x}_k$, we seek a correction $\delta\mathbf{x}$ such that $g(\mathbf{x}_k + \delta\mathbf{x}) = 0$. We linearize g around $\mathbf{x}_k$:

$$g(\mathbf{x}_k + \delta\mathbf{x}) \approx g(\mathbf{x}_k) + \left.\frac{\partial g}{\partial \mathbf{x}}\right|_{\mathbf{x}_k} \delta\mathbf{x} = 0$$

Solving for the correction $\delta\mathbf{x}$ gives the Newton step:

$$\delta\mathbf{x} = -\left(\frac{\partial g}{\partial \mathbf{x}}\right)^{-1} g(\mathbf{x}_k) \tag{3.3}$$

The next guess is then $\mathbf{x}_{k+1} = \mathbf{x}_k + \delta\mathbf{x}$. This process is repeated until convergence.

Algorithm 3: Newton's Method

```
Input: Function $g(\mathbf{x})$, initial guess $\mathbf{x}_0$, tolerance $\epsilon$
Output: Root $\mathbf{x}^\star$
1 $\mathbf{x}_k \leftarrow \mathbf{x}_0$
2 while $\|\mathbf{g}(\mathbf{x}_k)\| > \epsilon$ do
3     Compute Jacobian $\mathbf{J} = \frac{\partial g}{\partial \mathbf{x}}|_{\mathbf{x}_k}$
4     Solve the linear system $\mathbf{J}\delta\mathbf{x} = -g(\mathbf{x}_k)$ for $\delta\mathbf{x}$
5     $\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k + \delta\mathbf{x}$
6 return $\mathbf{x}_k$
```

Code 3.1 (Julia Notebook: Root Finding for Backward Euler).
The 'root-finding.ipynb' notebook provides a perfect comparison of fixed-point iteration and Newton's method. Both are used to solve the Backward Euler equation for the pendulum. The notebook plots the error (norm of the residual) at each iteration for both methods.

- **Fixed-point iteration** shows a slow, steady, linear decrease in error on a semi-log plot. It takes many iterations to reach high precision.
- **Newton's method** exhibits its characteristic **quadratic convergence**. The error decreases extremely rapidly, with the number of correct digits roughly doubling at each step. It typically converges to machine precision in just a few iterations.

This demonstrates the power of Newton's method. While each step is more expensive (requiring a Jacobian and a linear solve, is of the order $O(n^3)$), the drastically fewer iterations required make it far more efficient overall.

3.3 Unconstrained Minimization

We now turn to the problem of finding a local minimum of a smooth scalar function $f : \mathbb{R}^n \to \mathbb{R}$. A **first-order necessary condition** for a point $\mathbf{x}^\star$ to be a local minimum is that the gradient vanishes:

$$\nabla f(\mathbf{x}^\star) = 0 \tag{3.4}$$

This turns the minimization problem into a root-finding problem on the gradient! We can directly apply Newton's method.

Definition 3.1 (Newton's Method for Minimization)**.** To find a point where $\nabla f(\mathbf{x}) = 0$, we apply the Newton's method recipe. The "function" we are finding the root of is now $\nabla f(\mathbf{x})$. The "Jacobian" of this function is the Hessian, $\nabla^2 f(\mathbf{x})$. The Newton step is therefore:

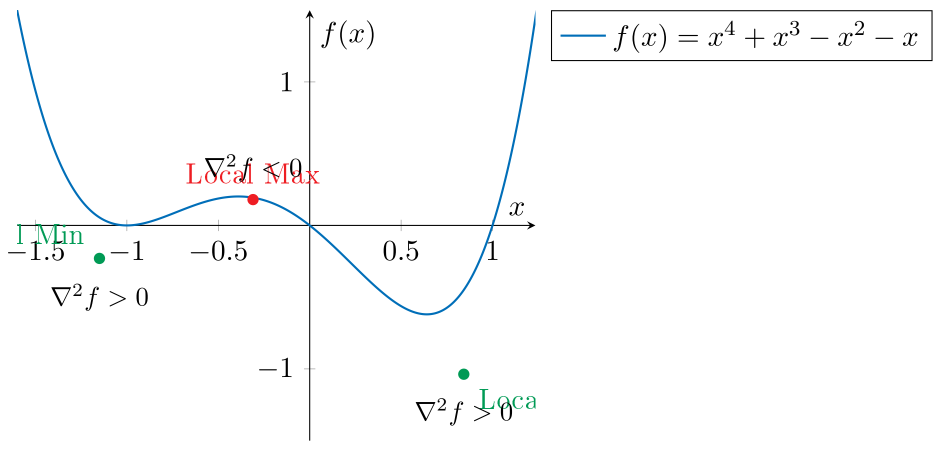
$$\delta\mathbf{x} = -(\nabla^2 f(\mathbf{x}))^{-1}\nabla f(\mathbf{x}) \tag{3.5}$$

The intuition here is that we are fitting a quadratic model to the function at the current point (based on the second-order Taylor expansion) and then jumping directly to the minimum of that quadratic model.

However, there are pitfalls. The condition $\nabla f(\mathbf{x}^\star) = 0$ applies to local maxima and saddle points as well as minima. A standard Newton step can easily converge to the wrong type of point.

A **second-order sufficient condition** for $\mathbf{x}^\star$ to be a strict local minimum is that $\nabla f(\mathbf{x}^\star) = 0$ and the Hessian $\nabla^2 f(\mathbf{x}^\star)$ is **positive definite** (all its eigenvalues are positive). A Newton step is a *descent direction* (i.e., a step that decreases the function value) if and only if the Hessian is positive definite. If the Hessian is negative definite (at a maximum), Newton's method becomes an *ascent* method.

Code 3.2 (Julia Notebook: The Pitfall of Standard Newton's Method)**.**
The 'minimization.ipynb' notebook clearly illustrates this problem. It attempts to minimize the function $f(x) = x^4 + x^3 - x^2 - x$.



When an initial guess of $x_0 = 0$ is used, the Hessian $\nabla^2 f(0)$ is negative. The notebook shows that the standard Newton step moves from $x = 0$ towards the local maximum, not the minimum.

3.3.1 Regularization (Damped Newton's Method)

To fix this, we can **regularize** the Hessian. The goal is to modify the Hessian $\mathbf{H} = \nabla^2 f(\mathbf{x})$ to ensure it is positive definite, thus guaranteeing a descent direction. A simple and effective method is Levenberg-Marquardt style damping:

$$\text{If } \mathbf{H} \text{ is not positive definite, replace it with } \mathbf{H}' = \mathbf{H} + \beta\mathbf{I}$$

where $\beta > 0$ is a damping parameter. We can increase β until $\mathbf{H}'$ becomes positive definite. This has two effects:

1. It guarantees the resulting step $\delta\mathbf{x} = -(\mathbf{H}')^{-1}\nabla f(\mathbf{x})$ is a descent direction.

2. As $\beta \to \infty$, the step becomes $\delta\mathbf{x} \approx -\frac{1}{\beta}\nabla f(\mathbf{x})$, which is a small step in the steepest descent direction.

This method adaptively blends between the fast Newton step (when the function is locally convex) and the robust but slow gradient descent step (when it is not).

Code 3.3 (Julia Notebook: Regularized Newton's Method).
The notebook implements this regularization. When starting from $x_0 = 0$, the algorithm detects the negative Hessian, adds damping, and computes a new step. The resulting step is now correctly pointed towards the local minimum. This demonstrates how regularization makes Newton's method a robust tool for nonlinear optimization.

3.4 Supplementary Concepts

We adopt the setting of unconstrained smooth optimization $\min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x})$ with $f \in C^2$ unless otherwise specified.

3.4.1 Convexity and Strong Convexity

Definition 3.2 (Convex Function). Let $\mathcal{D} \subseteq \mathbb{R}^n$ be convex. A function $f : \mathcal{D} \to \mathbb{R}$ is convex if for all $\mathbf{x}, \mathbf{y} \in \mathcal{D}$ and $\theta \in [0, 1]$:

$$f(\theta\mathbf{x} + (1-\theta)\mathbf{y}) \le \theta f(\mathbf{x}) + (1-\theta) f(\mathbf{y}).$$

It is *strictly convex* if the inequality is strict whenever $\mathbf{x} \ne \mathbf{y}$ and $\theta \in (0, 1)$.

Definition 3.3 (Strong Convexity). For $m > 0$, f is m-strongly convex if

$$f(\mathbf{y}) \ge f(\mathbf{x}) + \nabla f(\mathbf{x})^\top(\mathbf{y}-\mathbf{x}) + \frac{m}{2}\|\mathbf{y}-\mathbf{x}\|^2, \quad \forall \mathbf{x}, \mathbf{y}.$$

Theorem 3.1 (Second-Order Characterizations). Assume $f \in C^2$ on an open convex domain.

- f convex $\Leftrightarrow$ $\nabla^2 f(\mathbf{x})$ is positive semidefinite (PSD) for all $\mathbf{x}$.
- f m-strongly convex $\Leftrightarrow$ $\nabla^2 f(\mathbf{x}) \succeq m\mathbf{I}$ for all $\mathbf{x}$ (i.e., all eigenvalues $\ge m$).

Theorem 3.2 (Alternative Strong Convexity Characterizations). Let f be differentiable and $m > 0$. The following are equivalent:

1. f is m-strongly convex.
2. For all $\mathbf{x}, \mathbf{y}$, $(\nabla f(\mathbf{x}) - \nabla f(\mathbf{y}))^\top(\mathbf{x}-\mathbf{y}) \ge m\|\mathbf{x}-\mathbf{y}\|^2$.
3. For all $\mathbf{x}, \mathbf{y}$, $f(\mathbf{y}) \ge f(\mathbf{x}) + \nabla f(\mathbf{x})^\top(\mathbf{y}-\mathbf{x}) + \frac{m}{2}\|\mathbf{y}-\mathbf{x}\|^2$.

Consequences. Strong convexity implies uniqueness of the global minimizer. Plain convexity only guarantees that any local minimum is global (but possibly non-unique). For L-smooth functions (gradient Lipschitz with constant L), gradient descent with step size $\alpha \in (0, 2/L)$ converges; if additionally m-strongly convex, the convergence is linear with rate $1 - m/L$.

3.4.2 Optimality Conditions (Unconstrained)

Let $f \in C^2$. A point $\mathbf{x}^\star$ is a (local) minimizer only if the following hold.

Theorem 3.3 (First-Order Necessary Condition)**.** If $\mathbf{x}^\star$ is a local minimizer, then $\nabla f(\mathbf{x}^\star) = 0$.

Theorem 3.4 (Second-Order Necessary Condition)**.** If $\mathbf{x}^\star$ is a local minimizer and $f \in C^2$, then $\nabla f(\mathbf{x}^\star) = 0$ and $\nabla^2 f(\mathbf{x}^\star)$ is PSD.

Theorem 3.5 (Second-Order Sufficient Condition)**.** If $\nabla f(\mathbf{x}^\star) = 0$ and $\nabla^2 f(\mathbf{x}^\star)$ is positive definite (PD), then $\mathbf{x}^\star$ is a strict local minimizer. If $\nabla^2 f(\mathbf{x}) \succeq m\mathbf{I}$ globally ($m > 0$), then $\mathbf{x}^\star$ is the unique global minimizer.

Saddle Points. If the Hessian has both positive and negative eigenvalues at a stationary point, Newton's method may be attracted there; regularization or negative curvature exploitation is needed to escape.

3.4.3 Positive Definiteness, Cholesky Factorization, and On-the-Spot Padding

Definition 3.4 (Positive (Semi)Definiteness)**.** A symmetric matrix $\mathbf{H} \in \mathbb{R}^{n\times n}$ is PSD if $\mathbf{v}^\top \mathbf{H}\mathbf{v} \geq 0$ for all $\mathbf{v}$; PD if the inequality is strict for all nonzero $\mathbf{v}$.

Cholesky Factorization. A symmetric matrix $\mathbf{H}$ is PD $\Leftrightarrow$ it admits a (unique) Cholesky factorization $\mathbf{H} = \mathbf{R}^\top\mathbf{R}$ with $\mathbf{R}$ upper triangular with positive diagonal. Attempting Cholesky is therefore a practical test: breakdown (negative pivot) indicates loss of definiteness.

Lecture 4: Constrained Minimization

4.1 Globalization Strategy I: Line Search

While regularization ensures we take steps in the right direction, it doesn't control how far we step. The full Newton step $\delta\mathbf{x} = -(\nabla^2 f)^{-1}\nabla f$ is derived from minimizing a quadratic approximation of the function. If the function is not well-approximated by a quadratic, this step can overshoot the actual minimum.

A **line search** is a procedure to find an appropriate step size $\alpha \in (0, 1]$ to scale the Newton direction $\delta\mathbf{x}$, such that the update $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha\delta\mathbf{x}$ makes sufficient progress.

Definition 4.1 (Armijo Backtracking Line Search). A simple and effective line search strategy is the Armijo rule. It ensures that the actual reduction in the function value is at least a fraction of the reduction predicted by the linear approximation.

Algorithm 4: Armijo Backtracking Line Search

```
Input: Current point $\mathbf{x}$, search direction $\delta\mathbf{x}$, parameters $b \in (0,1), c \in (0,1)$
Output: Step size $\alpha$
1 $\alpha \leftarrow 1.0$
2 while $f(\mathbf{x} + \alpha\delta\mathbf{x}) > f(\mathbf{x}) + b\alpha(\nabla f(\mathbf{x}))^\top \delta\mathbf{x}$ do
3     $\alpha \leftarrow c\alpha$
4 return $\alpha$
```

Note. The term $(\nabla f(\mathbf{x}))^\top \delta\mathbf{x}$ is the directional derivative, which represents the slope of the function along the search direction. The condition checks if our actual progress is better than a certain fraction (b) of the progress we would expect from a linear model. Typical values are $b \approx 10^{-4}$ and $c = 0.5$.

Code 4.1 (Julia Notebook: Backtracking Newton's Method).
The ‘minimization.ipynb‘ notebook implements this backtracking line search. When applied, it prevents the large, unproductive steps that the pure Newton method might take, especially when far from the optimum. This combination of **regularization** (to pick a good direction) and **line search** (to pick a good distance) makes Newton's method a robust and powerful "globalized" algorithm for finding local minima.

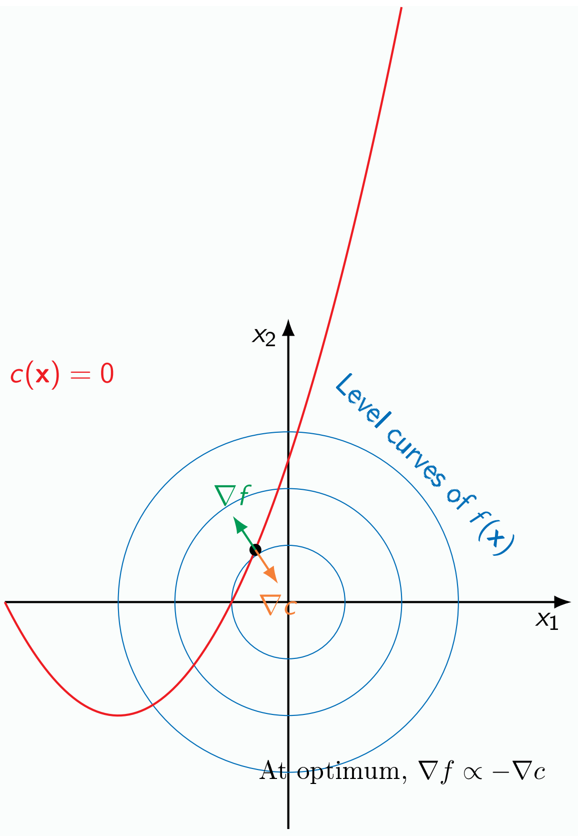
4.2 Equality-Constrained Minimization

We now consider problems of the form:

$$\begin{aligned} \min_{\mathbf{x}\in\mathbb{R}^n} \quad & f(\mathbf{x}) \\ \text{subject to} \quad & \mathbf{c}(\mathbf{x}) = 0 \end{aligned}$$

where $\mathbf{c} : \mathbb{R}^n \to \mathbb{R}^m$ is a set of m equality constraints.

Intuition. At a constrained optimum $\mathbf{x}^\star$, you cannot make further progress by moving along the constraint surface. This means the gradient of the objective function, $\nabla f(\mathbf{x}^\star)$, must be orthogonal to the constraint surface. The gradient of the constraint function, $\nabla c(\mathbf{x}^\star)$, is also orthogonal to the constraint surface. Therefore, the two gradients must be parallel.



This geometric condition implies that at the optimum, $\nabla f(\mathbf{x}^\star)$ must be a linear combination of the constraint gradients.

Definition 4.2 (The Lagrangian)**.** This leads to the first-order necessary conditions for optimality. There must exist a vector of **Lagrange multipliers** $\lambda \in \mathbb{R}^m$ such that:

$$\nabla f(\mathbf{x}^\star) + (\frac{\partial \mathbf{c}}{\partial \mathbf{x}})^\top \lambda = 0 \tag{4.1}$$

We can elegantly combine this condition with the original feasibility condition, $\mathbf{c}(\mathbf{x}) = 0$, by defining the **Lagrangian** function, $\mathcal{L} : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$:

$$\mathcal{L}(\mathbf{x}, \lambda) = f(\mathbf{x}) + \lambda^\top \mathbf{c}(\mathbf{x}) \tag{4.2}$$

The optimality conditions are now equivalent to finding a stationary point of the Lagrangian:

$$\nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x}, \lambda) = \nabla f(\mathbf{x}) + (\frac{\partial \mathbf{c}}{\partial \mathbf{x}})^\top \lambda = 0$$
$$\nabla_{\lambda} \mathcal{L}(\mathbf{x}, \lambda) = \mathbf{c}(\mathbf{x}) = 0$$

This is a root-finding problem for the concatenated variable vector $(\mathbf{x}, \lambda)$. We can solve it using Newton's method! Applying Newton's method yields the following linear system, known as the Karush-Kuhn-Tucker (KKT) system:

$$\begin{bmatrix} \nabla^2_{\mathbf{xx}} \mathcal{L} & (\frac{\partial \mathbf{c}}{\partial \mathbf{x}})^\top \\ \frac{\partial \mathbf{c}}{\partial \mathbf{x}} & 0 \end{bmatrix} \begin{bmatrix} \delta \mathbf{x} \\ \delta \lambda \end{bmatrix} = - \begin{bmatrix} \nabla_{\mathbf{x}} \mathcal{L} \\ \mathbf{c}(\mathbf{x}) \end{bmatrix} \tag{4.3}$$

4.2.1 The Gauss-Newton Approximation

The full Hessian of the Lagrangian is $\nabla^2_{\mathbf{xx}}\mathcal{L} = \nabla^2 f(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i \nabla^2 c_i(\mathbf{x})$. Another way to write this, that allows for using jacobian-vector product tricks from autodifferentiation, is:

$$\nabla^2_{\mathbf{xx}}\mathcal{L} = \nabla^2 f(\mathbf{x}) + \frac{\partial}{\partial x}\left[\left(\frac{\partial c}{\partial x}\right)^{\top} \lambda\right]$$

Computing the second derivatives of the constraints (the "constraint curvature" term) can be expensive. The **Gauss-Newton method** (or Sequential Quadratic Programming, SQP) approximates this Hessian by simply dropping the constraint curvature term:

$$\nabla^2_{\mathbf{xx}}\mathcal{L} \approx \nabla^2 f(\mathbf{x})$$

This approximation often works very well, leading to cheaper iterations that still make good progress.

Code 4.2 (Julia Notebook: Equality Constrained Optimization).
The ‘equality-constraints.ipynb‘ notebook solves a simple quadratic minimization problem subject to a nonlinear equality constraint. It compares the full Newton method with the Gauss-Newton method.

- **Full Newton**: Converges very quickly (quadratically) to the solution from a good initial guess.
- **Gauss-Newton**: Also converges, but may take a few more iterations since its model of the problem is less accurate. However, each iteration is computationally cheaper. In many large-scale robotics problems, this trade-off makes Gauss-Newton the preferred method.

The notebook also shows a case where the full Newton method can get "stuck" if the second-order constraint curvature term is problematic, while the simpler Gauss-Newton approximation avoids this issue and successfully finds the solution.

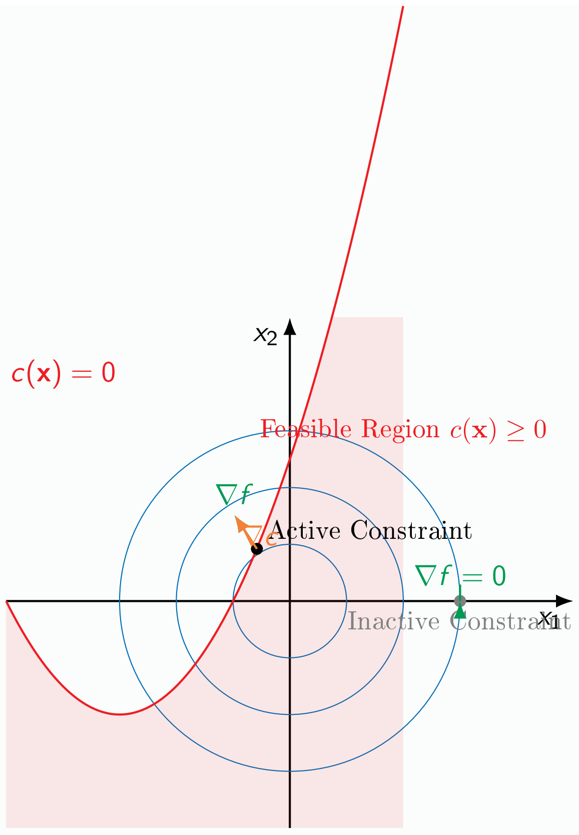
4.3 Inequality-Constrained Minimization

Finally, we consider the general case:

$$\begin{aligned} \min_{\mathbf{x}\in\mathbb{R}^n} \quad & f(\mathbf{x}) \\ \text{subject to} \quad & \mathbf{c}(\mathbf{x}) \geq 0 \end{aligned}$$

At the solution $\mathbf{x}^\star$, some constraints may be **active** ($c_i(\mathbf{x}^\star) = 0$) while others are **inactive** ($c_i(\mathbf{x}^\star) > 0$).

Intuition. Inactive constraints don't affect the solution locally, so we can ignore them. Active constraints behave like equality constraints. However, there is a key difference: for an inequality constraint, the gradient ∇f must point "away" from the feasible region. This means ∇f must be a *non-negative* linear combination of the active constraint gradients.



Theorem 4.1 (Karush-Kuhn-Tucker (KKT) Conditions)**.** The first-order necessary conditions for a point $\mathbf{x}^\star$ to be a local minimum are that there exists a vector of Lagrange multipliers $\lambda^\star$ satisfying:

1. **Stationarity**: $\nabla_{\mathbf{x}}\mathcal{L}(\mathbf{x}^\star, \lambda^\star) = \nabla f(\mathbf{x}^\star) + (\frac{\partial \mathbf{c}}{\partial \mathbf{x}})^\top \lambda^\star = 0$
2. **Primal Feasibility**: $\mathbf{c}(\mathbf{x}^\star) \geq 0$
3. **Dual Feasibility**: $\lambda^\star \geq 0$
4. **Complementary Slackness**: $\lambda_i^\star c_i(\mathbf{x}^\star) = 0$ for all $i = 1, \ldots, m$.

The complementary slackness condition is a clever way of stating that if a constraint is inactive ($c_i > 0$), its corresponding multiplier must be zero ($\lambda_i = 0$). This mathematically enforces our intuition. Modern algorithms for solving such problems, like interior-point methods, are designed to find points that satisfy these KKT conditions.

4.4 Supplementary Material: Advanced Line Search Concepts

The backtracking line search using the Armijo rule is simple and effective. However, to fully appreciate the theory of optimization, it's crucial to understand the landscape of line search conditions and the convergence guarantees they provide.

4.4.1 The Challenge of Exact Line Search

At each iteration of a descent method, we have a current point $\mathbf{x}_k$ and a descent direction $\mathbf{p}_k$. The ideal step size α would be the one that solves the one-dimensional optimization problem:

$$\alpha_k = \arg\min_{\alpha>0} \phi(\alpha) = \arg\min_{\alpha>0} f(\mathbf{x}_k + \alpha \mathbf{p}_k) \tag{4.4}$$

Solving this subproblem exactly is often computationally expensive, requiring many function and gradient evaluations. The cost of finding the exact minimum might be comparable to the cost of several iterations of the main algorithm. Therefore, in practice, we use **inexact line searches** that aim to find a step size that is "good enough" with minimal effort. The following conditions define what "good enough" means.

4.4.2 A Tour of Inexact Line Search Conditions

The Armijo (Sufficient Decrease) Condition This is the condition we have already seen. It ensures that the step size α leads to a decrease in the objective function that is at least a fraction of the decrease predicted by the linear approximation at $\mathbf{x}_k$.

$$f(\mathbf{x}_k + \alpha \mathbf{p}_k) \leq f(\mathbf{x}_k) + c_1 \alpha \nabla f(\mathbf{x}_k)^\top \mathbf{p}_k \tag{4.5}$$

with $c_1 \in (0, 1)$. A typical value for c_1 is 10^{-4}. While this condition prevents steps that are too long, it does not prevent steps that are excessively short. A backtracking algorithm based solely on this rule might produce a sequence of very small steps and converge slowly.

The Wolfe Conditions To rule out unacceptably short steps, the Armijo condition is often paired with a **curvature condition**. The combination is known as the **Wolfe conditions**.

1. **Sufficient Decrease (Armijo):** Same as (4.5).
2. **Curvature Condition:**

   $$\nabla f(\mathbf{x}_k + \alpha \mathbf{p}_k)^\top \mathbf{p}_k \geq c_2 \nabla f(\mathbf{x}_k)^\top \mathbf{p}_k \tag{4.6}$$

   with $c_2 \in (c_1, 1)$. This condition ensures that the slope of $\phi(\alpha)$ at the new point is less negative than the initial slope, which means we have made reasonable progress and are not at a point where the function is still decreasing rapidly. It forces the step length α to be in a region where the function is flatter.

A step length satisfying both conditions is guaranteed to exist for many common functions, provided $\mathbf{p}_k$ is a descent direction.

The Strong Wolfe Conditions A slight modification leads to the **Strong Wolfe conditions**, which constrain the slope to be closer to zero, forcing α to be closer to a stationary point of $\phi(\alpha)$.

1. **Sufficient Decrease (Armijo):** Same as (4.5).
2. **Strong Curvature Condition:**

$$|\nabla f(\mathbf{x}_k + \alpha \mathbf{p}_k)^\top \mathbf{p}_k| \leq c_2 |\nabla f(\mathbf{x}_k)^\top \mathbf{p}_k| \tag{4.7}$$

This is particularly useful in quasi-Newton methods (like BFGS), where it helps ensure that the Hessian approximation matrix remains positive definite.

The Goldstein Conditions The Goldstein conditions are another pair of inequalities that ensure both sufficient decrease and that the step is not too short.

$$f(\mathbf{x}_k) + (1 - c)\alpha \nabla f(\mathbf{x}_k)^\top \mathbf{p}_k \leq f(\mathbf{x}_k + \alpha \mathbf{p}_k) \tag{4.8}$$

$$f(\mathbf{x}_k + \alpha \mathbf{p}_k) \leq f(\mathbf{x}_k) + c\alpha \nabla f(\mathbf{x}_k)^\top \mathbf{p}_k \tag{4.9}$$

with $0 < c < 1/2$. The first inequality ensures α is not too large, while the second (the Armijo condition) ensures it's not too small. A disadvantage is that the first inequality might exclude the actual minimizer of $\phi(\alpha)$, which is why the Wolfe conditions are generally preferred in modern algorithms.

4.4.3 Convergence of Backtracking Line Search

We can prove that a simple backtracking algorithm using the Armijo rule will terminate and that the overall optimization algorithm will converge. This analysis relies on the smoothness of the objective function, specifically that its gradient is Lipschitz continuous.

Definition 4.3 (Lipschitz Continuity of the Gradient). A function f has a Lipschitz continuous gradient if there exists a constant $L > 0$ (the Lipschitz constant) such that for all $\mathbf{x}, \mathbf{y}$ in the domain:

$$\|\nabla f(\mathbf{x}) - \nabla f(\mathbf{y})\| \leq L\|\mathbf{x} - \mathbf{y}\| \tag{4.10}$$

This is a measure of the smoothness of the gradient. For twice-differentiable functions, it is related to the maximum eigenvalue of the Hessian. A key consequence, derived from the Mean Value Theorem, is the following quadratic upper bound on the function:

$$f(\mathbf{x} + \mathbf{p}) \leq f(\mathbf{x}) + \nabla f(\mathbf{x})^\top \mathbf{p} + \frac{L}{2}\|\mathbf{p}\|^2 \tag{4.11}$$

Proof of Termination for Backtracking Line Search Here, we prove that the backtracking line search algorithm (using the Armijo rule) finds a step size α that satisfies the condition in a finite number of iterations, assuming the gradient ∇f is Lipschitz continuous.

The Armijo condition is given by:

$$f(\mathbf{x}_k + \alpha \mathbf{p}_k) \leq f(\mathbf{x}_k) + c_1 \alpha \nabla f_k^\top \mathbf{p}_k$$

From the definition of Lipschitz continuity, we have the quadratic upper bound on f shown in (4.11). Applying this with $\mathbf{p} = \alpha \mathbf{p}_k$, we get:

$$f(\mathbf{x}_k + \alpha \mathbf{p}_k) \leq f(\mathbf{x}_k) + \alpha \nabla f_k^\top \mathbf{p}_k + \frac{L}{2}\alpha^2\|\mathbf{p}_k\|^2$$

For the Armijo condition to hold, it is sufficient that the upper bound on $f(\mathbf{x}_k + \alpha \mathbf{p}_k)$ satisfies the condition. We therefore seek an α such that:

$$f(\mathbf{x}_k) + \alpha \nabla f_k^\top \mathbf{p}_k + \frac{L}{2}\alpha^2\|\mathbf{p}_k\|^2 \leq f(\mathbf{x}_k) + c_1 \alpha \nabla f_k^\top \mathbf{p}_k$$

Subtracting $f(\mathbf{x}_k)$ from both sides and rearranging gives:

$$(1 - c_1)\alpha \nabla f_k^\top \mathbf{p}_k + \frac{L}{2}\alpha^2\|\mathbf{p}_k\|^2 \leq 0$$

Since $\alpha > 0$, we can divide by it:

$$(1 - c_1)\nabla f_k^\top \mathbf{p}_k + \frac{L}{2}\alpha\|\mathbf{p}_k\|^2 \leq 0$$

Now, we solve for α. Since $\mathbf{p}_k$ is a descent direction, we know $\nabla f_k^\top \mathbf{p}_k < 0$. Also, $c_1 \in (0, 1)$ means $1 - c_1 > 0$.

$$\alpha \leq \frac{2(1 - c_1)(-\nabla f_k^\top \mathbf{p}_k)}{L\|\mathbf{p}_k\|^2} \tag{4.12}$$

This inequality shows that any α that is sufficiently small (i.e., below the positive threshold on the right-hand side) will satisfy the Armijo condition. The backtracking algorithm starts with an initial α (e.g., $\alpha = 1$) and multiplies it by a contraction factor $c \in (0, 1)$ until the condition is met. Since each step reduces α, it is guaranteed to eventually fall below this threshold. Therefore, the backtracking line search terminates in a finite number of steps. This proves that the backtracking line search is a well-posed and practical compromise, avoiding the expense of exact search while still guaranteeing sufficient progress towards the minimum.

Lecture 5: Solving Inequality-Constrained Problems

5.1 Challenges with Inequality Constraints

Recall the KKT conditions for a problem $\min f(\mathbf{x})$ s.t. $\mathbf{c}(\mathbf{x}) \geq 0$:

1. **Stationarity**: $\nabla f(\mathbf{x}^*) - (\frac{\partial \mathbf{c}}{\partial \mathbf{x}})^\top \lambda^* = 0$ (note the sign change on λ is a common convention for $\geq$ constraints)
2. **Primal Feasibility**: $\mathbf{c}(\mathbf{x}^*) \geq 0$
3. **Dual Feasibility**: $\lambda^* \geq 0$
4. **Complementary Slackness**: $\lambda \odot c(\mathbf{x}^*) = 0$

Role of Complementary Slackness ("Switching"). For each constraint component $c_i(\mathbf{x}) \geq 0$ with multiplier $\lambda_i \geq 0$, the complementarity condition $\lambda_i c_i(\mathbf{x}^*) = 0$ enforces an automatic switch:

- (Inactive case) If at the solution $c_i(\mathbf{x}^*) > 0$, then necessarily $\lambda_i = 0$. The stationarity condition then contains no contribution from this constraint, so locally this constraint behaves as if it were absent; i.e., the problem reduces (locally) to an unconstrained one in that direction.
- (Active case) If $c_i(\mathbf{x}^*) = 0$, the multiplier can be strictly positive $\lambda_i > 0$. In that case the gradient contribution $-\lambda_i \nabla c_i(\mathbf{x}^*)$ appears in stationarity, and c_i effectively acts like an equality constraint: motion that would violate $c_i \geq 0$ is opposed by a restoring first-order term. The boundary is therefore "sticky": any feasible descent direction must be tangent to $c_i(\mathbf{x}) = 0$.

Thus, complementary slackness encodes a discrete (active/inactive) combinatorial choice inside a continuous system of equations. This logical structure is the main obstacle to naively applying Newton's method directly to all KKT conditions: the feasible manifold changes dimension depending on which constraints are active.

The presence of the inequality conditions (2 and 3) and the complementarity constraint (4) prevents us from directly applying Newton's method to solve this system as a standard root-finding problem. We need more sophisticated approaches.

5.2 Methods for Inequality-Constrained Optimization

5.2.1 Active-Set Methods

These methods work by maintaining a "guess" of which constraints will be active (equal to zero) at the solution.

1. Solve the equality-constrained problem assuming the current active set is correct.
2. Check the KKT conditions. If any Lagrange multipliers for the active set are negative, it means the objective could be improved by making that constraint inactive. Remove it from the active set.
3. If any inactive constraints are violated, add them to the active set.
4. Repeat until all conditions are satisfied.

Active-set methods work well for problems where the active set doesn't change much between iterations, such as Quadratic Programs (QPs).

Consequences of a Wrong Active Set Guess. Suppose the true optimal active set is $\mathcal{A}^* = \{i : c_i(\mathbf{x}^*) = 0\}$.

- (Missing an active constraint) If we temporarily exclude some $j \in \mathcal{A}^*$ from the working set, we solve a relaxation. The intermediate solution $\tilde{\mathbf{x}}$ will typically violate feasibility with $c_j(\tilde{\mathbf{x}}) < 0$ (if constraints were encoded as $c_i(\mathbf{x}) \geq 0$), flagging the need to add constraint j.
- (Including an inactive constraint) If we include some $k \notin \mathcal{A}^*$, we may obtain a Lagrange multiplier $\lambda_k < 0$. Violated dual feasibility indicates the constraint should be removed, since maintaining it would artificially restrict descent directions.

These tests (primal violation for missing constraints, negative multipliers for superfluous constraints) drive the update logic.

Combinatorial Nature and Complexity. In the worst case with m inequality constraints there are 2^m possible active sets. Enumerating them is exponential. Active-set methods avoid this by iteratively refining a single guess, but they can still require many iterations if the active set changes frequently or if the problem is ill-conditioned. They are most effective when the optimal active set is small or changes little between similar problems (e.g., in sequential QPs for trajectory optimization).

5.2.2 Penalty and Augmented Lagrangian Methods

These methods transform the constrained problem into an unconstrained one by adding a penalty term to the objective that discourages constraint violation. For a problem $\min f(\mathbf{x})$ s.t. $c(x) \geq 0$, the penalty formulation is:

$$\min_{\mathbf{x}} f(\mathbf{x}) + \frac{\rho}{2}(\min(0, c(x)))^2$$

The solver gradually increases the penalty weight $\rho \to \infty$. A major drawback is that large ρ values lead to an ill-conditioned Hessian, making the problem difficult to solve accurately. Additionally, at the constraint boundary, the Hessian is techinicaly not defined, so second order methods like Newton's method can struggle, even though they might work in practice. One of the recent modifications to

The **Augmented Lagrangian** method improves upon this by introducing an estimate of the Lagrange multipliers, avoiding the need for $\rho \to \infty$. This is a very powerful and popular technique.

5.2.3 The Barrier Problem

Consider the problem $\min f(\mathbf{x})$ s.t. $\mathbf{c}(\mathbf{x}) \geq 0$. We first introduce **slack variables** $\mathbf{s} \geq 0$ to convert the general inequality into an equality:

$$\begin{aligned} \min_{\mathbf{x},\mathbf{s}} \quad & f(\mathbf{x}) \\ \text{s.t.} \quad & \mathbf{c}(\mathbf{x}) - \mathbf{s} = 0 \\ & \mathbf{s} \geq 0 \end{aligned}$$

Now, we enforce the non-negativity constraint $\mathbf{s} \geq 0$ by adding a logarithmic barrier term to the objective. This creates a new equality-constrained subproblem, parameterized by the barrier parameter $\rho > 0$:

$$\min_{\mathbf{x},\mathbf{s}} \quad f(\mathbf{x}) - \rho \sum_i \log(s_i) \quad \text{s.t.} \quad \mathbf{c}(\mathbf{x}) - \mathbf{s} = 0 \tag{5.1}$$

The $-\log(s_i)$ term acts as a barrier: as $s_i \to 0^+$, its value goes to infinity, preventing the solver from ever making a slack variable non-positive. This keeps the iterates strictly "interior" to the feasible set.

5.3 Interior-Point Methods

Code 5.1 (Julia Notebook: Interior-Point Method).
The 'interior-point.ipynb' notebook implements a simplified interior-point method to solve a quadratic program with a linear inequality constraint.

- **Formulation**: Instead of slack variables, it uses a clever change of variables, $s = \sqrt{\rho}e^{\sigma}$ and $\lambda = \sqrt{\rho}e^{-\sigma}$, to implicitly enforce positivity and the relaxed complementarity condition. This transforms the problem into finding the roots of a system of two equations in $(\mathbf{x}, \sigma)$.
- **Central Path Following**: The notebook demonstrates the concept of the central path. By starting with a large ρ and solving, and then using that solution as a warm start for a smaller ρ, you can trace the path of solutions. The plot shows the solver taking a step towards the constraint boundary with a large ρ, and then converging directly to the true solution as ρ is decreased to a small value (e.g., 10^{-8}).
- **Robustness**: The solver uses Newton's method with a line search to robustly solve the root-finding problem at each ρ value.

This method is the engine behind powerful, open-source solvers like IPOPT, which are widely used in robotics for problems like trajectory optimization.

5.4 Quadratic Programs (QPs)

A particularly important class of optimization problems is the Quadratic Program (QP), which involves minimizing a quadratic objective subject to linear constraints.

$$\begin{aligned} \min_{\mathbf{x}} \quad & \frac{1}{2}\mathbf{x}^\top \mathbf{Q}\mathbf{x} + \mathbf{q}^\top \mathbf{x} \\ \text{s.t.} \quad & \mathbf{A}\mathbf{x} \geq \mathbf{b} \\ & \mathbf{C}\mathbf{x} = \mathbf{d} \end{aligned}$$

If the matrix $\mathbf{Q}$ is positive definite, the problem is convex, meaning it has a single global minimum. QPs can be solved extremely quickly (often in kHz) by specialized solvers, making them a cornerstone of many real-time control and estimation algorithms, including Model Predictive Control (MPC) and contact-implicit optimal control.