# EECS 106B/206B
## Robotic Manipulation and Interaction

**Valmik Prabhu**

**Spring 2020**

# Unit 2: Control

1. Linear Stability and Control
2. Lyapunov Stability
3. PID Control
4. A Survey of Nonlinear Control Techniques

# Goals of this lecture

Define a couple useful mathematical tools that will help us understand control

- Positive definite matrices and functions
- Matrix Diagonalization and the Jordan Normal Form

Summarize the basics of stabilization and control of linear systems

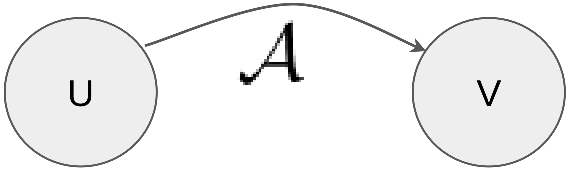
- Hurwitz Stability
- Lyapunov Equation
- State Feedback Control

# SOME USEFUL MATH

Content draws from Roberto Horowitz’s ME 232 Slides, some of Claire Tomlin’s EE 221A notes, and Kameshwar Poolla’s Linear Algebra Primer

# What is a Linear Map (aka Linear Operator)?

A linear operator $\mathcal{A} : U \to V$ is just a linear function*




Range Space: $\mathcal{R}(\mathcal{A}) := \{v | v = \mathcal{A}(u), u \in U\} \subset V$

Null Space: $\mathcal{N}(\mathcal{A}) := \{u | \mathcal{A}(u) = \emptyset v\} \subset U$

The null space and range space are themselves linear operators.

All linear operators can be expressed by matrix multiplication!

$$\mathcal{A} : v = Au$$

We can also examine all matrices as linear operators

*at least for our purposes

# What is a matrix

A matrix is a collection of vectors: $A = [v_1, v_2, \cdots v_m], v_i \in \mathbb{R}^n, u \in \mathbb{R}^m$

Thus, left-multiplying a vector by the matrix is just a weighted sum of the columns.

$$Ax = x_1 v_1 + x_2 v_2 \cdots + x_m v_m$$

A set of vectors is *linearly dependent* iff $\exists \{\alpha_1 \cdots \alpha_m\} \ s.t. \sum_{i=1}^{m} \alpha_i v_i = 0$

Since $\mathcal{R}(\mathcal{A})$ and $\mathcal{N}(\mathcal{A})$ are linear operators, they're also matrices

- Rank(A) := dimension (number of columns) of $\mathcal{R}(\mathcal{A})$
- Nullity(A) := dimension of $\mathcal{N}(\mathcal{A})$

# Rank Nullity Theorem:

$$\text{Let } \mathcal{A} : U \to V$$

$$\text{Rank}(\mathcal{A}) + \text{Nullity}(\mathcal{A}) = \dim(U)$$

This is one of the most important theorems in linear algebra!

# Eigenvalues and Eigenvectors

*Eigenvectors* are the “principal axes” of a matrix. Applying the matrix to an eigenvector will only scale it, not rotate it.

$$Av = \lambda v \longrightarrow (A - I\lambda)v = \emptyset$$

The amount an eigenvector gets stretched is its corresponding *eigenvalue*.

If all the eigenvectors of a matrix span $\mathbb{R}^n$ they’re called an *eigenbasis*

# Matrix Diagonalization

If the eigenvectors of a matrix form an eigenbasis, the matrix can be *diagonalized*

$$Av = v\lambda \longrightarrow AV = V\Lambda \longrightarrow A = V\Lambda V^{-1}$$

Where

$$V = \begin{bmatrix} v_1 & v_2 & \cdots & v_n \end{bmatrix} \qquad \Lambda = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_n \end{bmatrix}$$

# Jordan Normal Form

Algebraic Multiplicity: The number of times an eigenvalue λ appears

Geometric Multiplicity: The rank of the eigenvectors corresponding to λ

If algebraic multiplicity > geometric multiplicity for any λ, an eigenbasis cannot be constructed, and thus the matrix cannot be diagonalized.

Instead we use the Jordan Normal Form.

$$A = VJV^{-1}$$

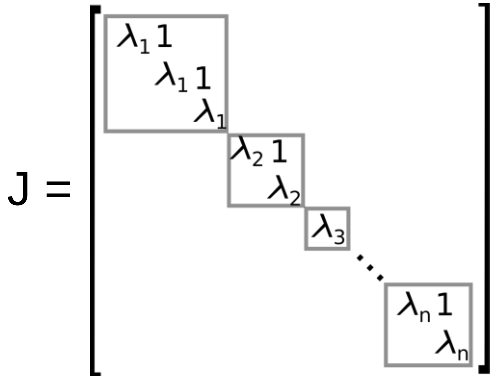



Figure from: Jacob Scholbach via Wikipedia. https://en.wikipedia.org/wiki/Jordan_normal_form#/media/File:Jordan_blocks.svg

# Matrix Properties

Symmetric / Hermitian: $A = A^T, \; A = A^*$

- All hermitian matrices can be represented as $A = B^T B$

Positive Definite: $A \succ 0$

- All EVs of A are greater than zero. $x^T A x > 0, \forall x \neq \emptyset$

Positive Semidefinite: $A \succeq 0$

- All EVs of A are greater than or equal to zero. $x^T A x \geq 0, \forall x \neq \emptyset$

Negative Definite / Semidefinite: $A \prec 0, \; A \preceq 0$

- Same as above, but EVs are negative

# INTRO TO LINEAR DYNAMICAL SYSTEMS

Content drawn from Claire Tomlin's EE221A Notes, Roberto Horowitz's ME 232 Slides, and Kameshwar Poolla's ME 132 Notes

# Linear Autonomous Systems

$$\dot{x} = Ax$$

# Linear Systems from Differential Equations

We can take any linear differential equation and turn it into a matrix

$$x^{(3)} = 3\ddot{x} + 7\dot{x} + 2x \longrightarrow \begin{bmatrix} \dot{x} \\ \ddot{x} \\ x^{(3)} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 2 & 7 & 3 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \ddot{x} \end{bmatrix}$$

What about this differential equation?

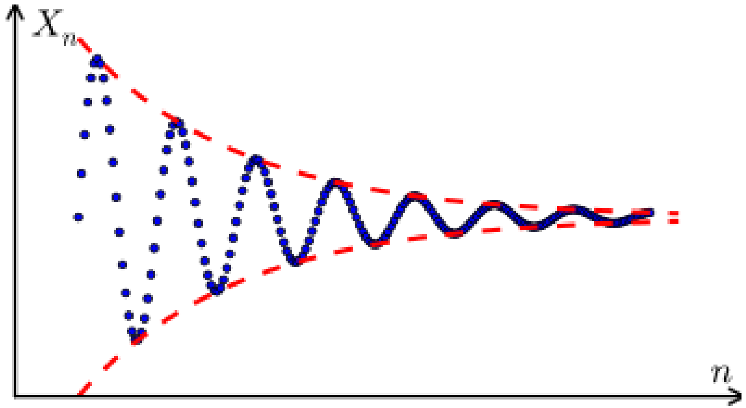
$$\cancel{x^{(3)} = 3\ddot{x} + 7\dot{x} + 2x + x^2}$$

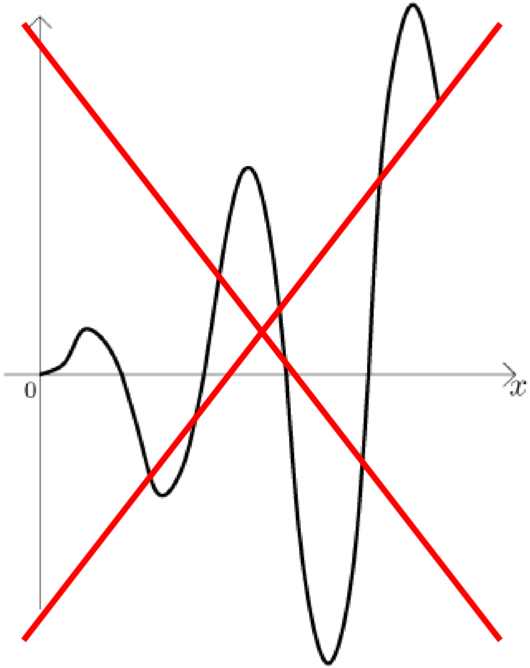
This differential equation is not linear

# Defining Stability

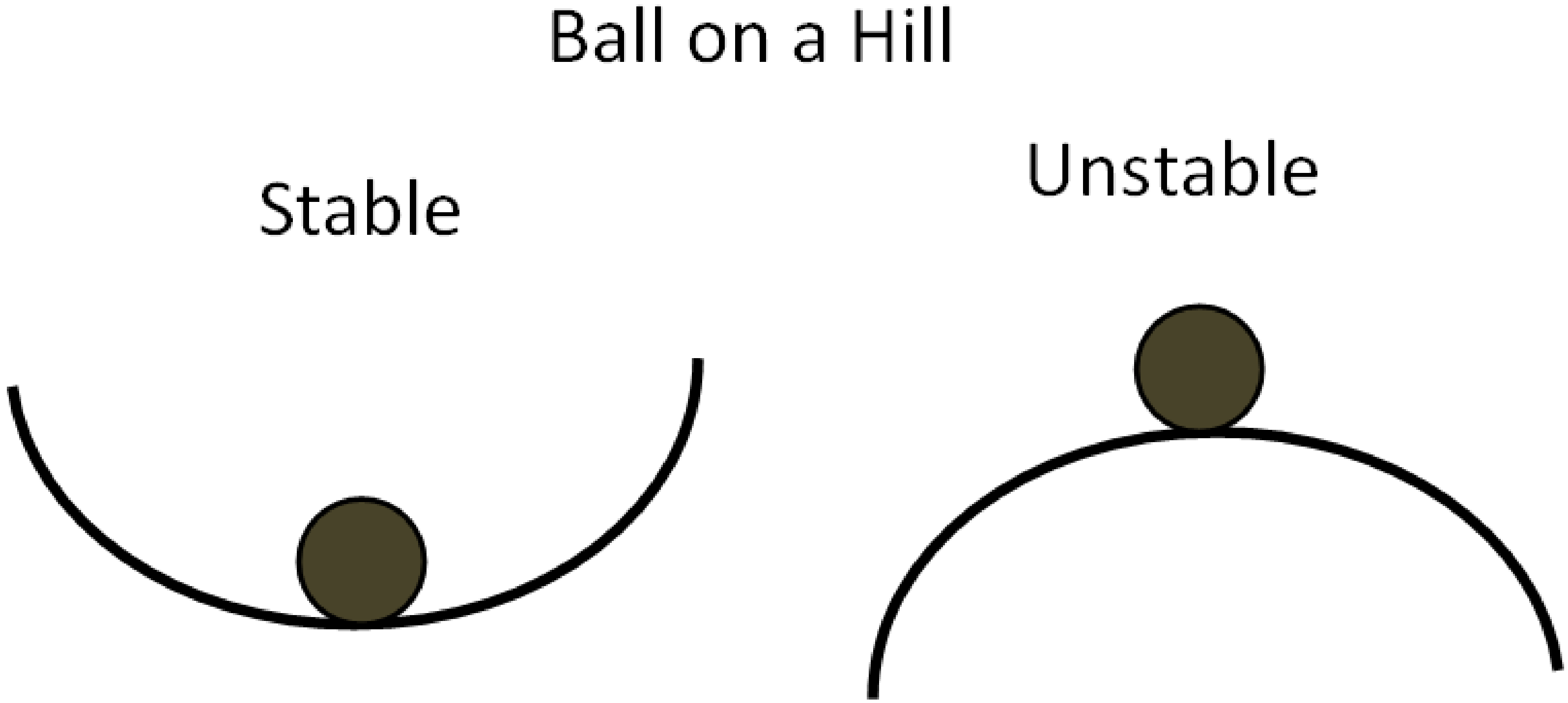
The origin is stable if small perturbations are not amplified.

- Linear Systems: Routh-Hurwitz Stability
- Nonlinear Systems: Lyapunov Stability



# Example



# Integrating with the Matrix Exponential

Just like we did before, we can use the matrix exponential to integrate

$$\dot{x} = Ax \longrightarrow x(t) = e^{At}x(0)$$

Our goal is **stability** or convergence; As time goes to infinity, we want our system state to not go to infinity

$$|| \lim_{t \to \infty} e^{At}x(0)||_2 \neq \infty$$

# Matrix Diagonalization and the Exponential

If our dynamics is diagonalizable, we can pull out the change of basis

$$e^{V\Lambda V^{-1}} = Ve^{\lambda}V^{-1} = \begin{bmatrix} e^{\lambda_1} & 0 & \cdots & 0 \\ 0 & e^{\lambda_2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & e^{\lambda_n} \end{bmatrix}$$

Hurwitz Stability!

$$\begin{cases} \lambda > 0 & \lim_{t\to\infty} e^{\lambda t} \to \infty, \quad \text{unstable} \\ \lambda = 0 & \lim_{t\to\infty} e^{\lambda t} \to 1, \quad \text{marginally stable} \\ \lambda < 0 & \lim_{t\to\infty} e^{\lambda t} \to 0, \quad \text{stable} \end{cases}$$

# Jordan Normal Form and the Matrix Exponential

If the matrix can't be diagonalized, use Jordan form. When you take the exponential, you can break up all the Jordan blocks, and integrate each separately.

$$\exp\left(\begin{bmatrix} \lambda & 1 \\ 0 & \lambda \end{bmatrix} t\right) = \begin{bmatrix} e^{\lambda t} & te^{\lambda t} \\ 0 & e^{\lambda t} \end{bmatrix}$$

This means that repeated eigenvalues on the imaginary axis (real part zero) that are not diagonalizable are not stable.

# Lyapunov Stability

The Lyapunov Equation looks at the change in system “energy”. We can define an energy function, which must be positive definite

$$V(x) \succ 0$$

Then we look at how the energy of that function changes over time

$$\frac{d}{dt}V(x) = \frac{\partial V}{\partial x}\dot{x} = L_f V$$

If our energy decreases at every point x, then our system is globally stable

# Deriving the Lyapunov Equation

We choose a quadratic energy function

$$V(x) = x^T P x,\ \ P \succ 0$$

And differentiate to get

$$\dot{V}(x) = x^T (A^T P + PA) x = x^T Q x$$

If Q is negative definite, then the system is stable

If Q is negative semi-definite, then  the system is marginally stable, or *stable in the sense of Lyapunov (SISL)*

# Driven Systems

$$\dot{x} = Ax + Bu$$

# Transfer Function Notation

Rather than using matrices, we can use transfer functions and examine control systems in the frequency domain

$$x^{(3)} = 2\ddot{x} + 7\dot{x} + 3x + u$$

$$s^3X = 2s^2X + 7sX + 3X + U$$

$$X = \frac{1}{s^3 - 2s^2 + 7s + 3}U$$

Control done in the frequency domain is called classical controls. We won't be doing much of it in this class, but you'll likely see it in some papers.

# State Feedback Control

In state feedback control, we make the control a linear function of state

$$u = -Kx$$

Our driven system now becomes an autonomous system with

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

By varying K, we can actually change all the eigenvalues of our new system, thus allowing us to (theoretically) determine both stability and rate of convergence for all modes of the system.

# LQR - Linear Quadratic Regulator

The “optimal” way to find K

Define cost matrices Q, R

$$Q \in \mathbb{R}^{n \times n},\ \ R \in \mathbb{R}^{u \times u},\ \ Q, R \succ 0$$

LQR finds a K which minimizes the cost function

$$\int_0^\infty (x^T Q x + u^T R u + 2x^T N u) dt$$

Use command “lqr” in Matlab

# What I didn’t cover

- How do we model and reject disturbances and noise?
- Our state feedback control design requires that we have x. What happens when we don’t?
- How do we get these models in the first place?
- How do we do this stuff fast computationally?
- How do we quantify (and qualify) controller performance?
- How do we apply this to nonlinear systems (100% of systems)?

# Sources

EE 221A Course Notes, Fa 2019, Claire Tomlin

ME 232 Course Slides, Fa 2018, Roberto Horowitz

ME 132 Course Notes, Sp 2016, Kameshwar Poolla

Linear Algebra Primer (from ME 232), Fa 2018, Kameshwar Poolla