EECS C106B / 206B Robotic Manipulation and Interaction **Spring 2020**

Lecture 3: Linear Autonomous Systems and Stability

Scribes: Son Tran and Jay Monga

3.1 Lecture Goals

- Defining a couple useful mathematical tools that will help us understand control.
- Summarizing the basics of stabilization and control of linear systems.

3.2 Lecture Outline

Now that we're done with 106A review, we can get into our next topic of **Control** (Valmik's favorite)! To fully dive into the material, there are a wide variety of classes to take, however such an intense level of scrutiny is not required for this course. With this lecture, we gain familiarity with linear systems and their control as to prepare ourselves for further engagement with controls in later lectures. Without further ado, here's what's covered in the lecture:

- Maths
  - Linear Algebra: Linear Maps, Diagonalization, and Jordan Normal Form
  - Positive Definite Functions and Matrices
- Linear Systems and Control
  - Differential Equations and Transfer Functions
  - Hurwitz and Lyapunov Stability
  - State Feedback Control

3.3 Linear Algebra Coverage

Depending on your background, some amount of this material may be a review. Whether to learn the material or just receive a refresher, going through this should be useful. We start our coverage by introducing the **linear map**.

Definition 3.3.1. Linear Map (or Operator): A linear map is just a linear function from a domain to a co-domain. Note: both domain and co-domain must be vector spaces. For our purposes, we will consider linear maps and operators to be the same thing.

We will often define mappings in a format like $\mathcal{A} : U \to V$, which means that $\mathcal{A}$ is a linear operator that will take vectors in U and put them in V somehow.

Linear functions should satisfy superposition and scaling with respect to input and output. We now look at two of the most important spaces associated with a linear operator: the **range space** and the **null space**.

Definition 3.3.2. Range Space: The range space is the subset of the co-domain which gets mapped to from the domain by the the linear operator. For a linear operator $\mathcal{A} : U \to V$, we can define the range space $\mathcal{R}(\mathcal{A})$ as $\mathcal{R}(\mathcal{A}) = \{v | v = \mathcal{A}(u), u \in U\}$

Definition 3.3.3. Null Space: The null space is the subset of the domain which gets mapped to 0 by the linear operator. The non-trivial null space, which is what we usually care about, is this set excluding the 0 vector. For a linear operator $\mathcal{A} : U \to V$, we can define the null space $\mathcal{N}(\mathcal{A})$ as $\mathcal{N}(\mathcal{A}) = \{u | \mathcal{A}(u) = 0\}$

It's important to note that in a general sense, range space deals with the space of output and null space deals with the space of inputs.

Now that we have an abstract notion of what linear maps are and the important spaces they are associated with, it would be nice if we had some tangible structure that can help us visualize what we're actually dealing with. This is when a **matrix** is useful.

Definition 3.3.4. Matrix: We will define a matrix as an ordered collection of vectors. An $m \times n$ matrix will be made up of n vectors, each one being m-dimensional.

We can define a linear operation with a matrix via left-multiplication by a vector: Each entry of the vector weights the appropriate column of the matrix before they are all added. Example: for an $m \times n$ matrix $A = [v_1, v_2, \ldots, v_n]$ and vector $x = [x_1, x_2, \ldots, x_n]^T$, $Ax = x_1 v_1 + x_2 v_2 + \ldots + x_n v_n = \sum_{i=1}^{n} x_i v_i$. An important note is that the number of columns in the matrix much match the dimension of the domain, and the number of rows in the matrix much match the dimension of the co-domain.

The great part about matrices is that any linear operator (even like the derivative for example) has an equivalent matrix representation. Although matrices offer a more digestible view of the input-output properties of a linear operator, we want a way to remove "redundant information" and actually get minimal sets of vectors that can generate the entire space we care about, called a **basis**. In order to do so, we need to establish some notion of what it means to be redundant, which we will call **linear dependence**.

Definition 3.3.5. Linear Dependence: For a set of vectors $S = \{x_1, x_2, \ldots, x_n\}$, we say that S is linear dependent if we can assign weight (not all 0) to all vectors in S such that the weighted sum of these vectors is 0, or $\exists \{a_1, a_2, \ldots, a_n\} s.t. \exists a_i \neq 0 \wedge a_1 x_1 + a_2 x_2 + \ldots + a_n x_n = \sum_{i=1}^{n} a_i x_i = 0$

Another way to view linear dependence is that if we have a set of vectors such that any vector in the set can be generated by a **linear combination** of other vectors in the set, that set is linearly dependent. We may simply define a set of vectors as linearly independent iff it's not linearly dependent. Let's also define what a basis is for good measure.

Definition 3.3.6. Basis: For a given vector space V, as basis is V is a set of linear independent vectors in V who's linear combination can generate any vector in V.

It can be proven that while the basis for a vector space may not be unique, they will all have the same size, which is why we use the number of vectors in any basis for a vector space to define the **dimension** of that space.

Now that we have these established ideas, let's put them to use. We can examine the crucial range and null spaces in context of the size of maximally linearly independent set, or basis, for each. We will define **rank** and **nullity** to refer to respective dimensions of the range and null space of a linear operator or matrix.

Definition 3.3.7. Rank: For a linear operator $\mathcal{A} : U \to V$, Rank($\mathcal{A}$) is the dimension of $\mathcal{R}(\mathcal{A})$, the range space of $\mathcal{A}$.

Definition 3.3.8. Nullity: For a linear operator $\mathcal{A} : U \to V$, Nullity($\mathcal{A}$) is the dimension of $\mathcal{N}(\mathcal{A})$, the null space of $\mathcal{A}$.

If you made a matrix that was comprised of basis vectors from $\mathcal{R}(\mathcal{A})$, the number of the columns in that matrix would be Rank$(\mathcal{A})$; similarly the number of columns in a matrix comprised of basis vectors of $\mathcal{N}(\mathcal{A})$ would be Nullity$(\mathcal{A})$. For a matrix A, we can get a basis for $\mathcal{R}(A)$ and thus determine Rank(A) by taking all linearly independent column vectors in A.

With the definitions of rank and nullity come one of the most important theorems in linear algebra: the rank-nullity theorem! Take Math 110 if you want to prove foundational linear algebra theorems like these.

Theorem 3.3.1 (Rank-Nullity Theorem)**.** For a linear operator $\mathcal{A} : U \to V$, Rank$(\mathcal{A})$ + Nullity$(\mathcal{A})$ = dim(U), or the dimension of the range space + the dimension of the null space of a linear operator is equal to the dimension of its domain.

This may not make intuitive sense right away, so here's something to think about. Let's say that we had a linear operator whose domain was 3 dimensional, nullity and rank both 2. The rank-nullity theorem tells us that this shouldn't be possible, so let's kind of figure out why. Nullity of 2 tells us that we can describe the null space, part of the domain, with two basis vectors. Fine. What about the rank? We also have 2 basis vectors in the co-domain that characterize that space. These two vectors should have corresponding linearly independent vectors in the domain which they are mapped to by, so now we have come across 4 linearly independent vectors in the domain! This kind of feels like having 4 different "directions" that produce distinct actions, which doesn't feel right in a 3 dimensional space where it seems like we should only have 3 different "directions" that produce distinct actions.

Here's an example to build intuition. Consider the matrix $A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$. It's domain and co-domain are both 3 dimensional. In addition, its range space is the 1 dimensional space generated by the span of $\{ \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \}$, while its null space is the 2 dimensional space generated by the span of $\{ \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \}$

This practically covers the basic linear algebra knowledge we need before we start talking about the more interesting stuff. Let's move on to that now.

Matrix multiplication is too hard. This whole linear combination business, scaling and adding, is just too much work. What if we didn't have to do all that work? What if we just had to ... scale? This is a role of **eigenvectors** and **eigenvalues**, to break down a complex matrix linear operation into simpler scaling operations.

Definition 3.3.9. Eigenvalues and Eigenvectors: For matrix A, a vector $v \neq 0$ that satisfies $Av = \lambda v$ with scalar λ is said to be an eigenvector of A with corresponding eigenvalue λ.

What's going on here? This is saying that we can have vectors for which the application of a linear operator only scales the vector! You've probably experienced many examples of this, maybe without noticing: for a rotation matrix, a vector along the axis of rotation is an eigenvector with eigenvalue 1, for the derivative operator on polynomials, 0-degree polynomials are eigenvectors with eigenvalue 0, and we can even think of any vector in the non-trivial nullspace as an eigenvector with eigenvalue 0.

Eigenvectors can also be thought of as principal axes of a matrix, the directions along which the operator only scales or stretches, and we can also then think of eigenvalues as the amount of scaling or stretching in each appropriate direction.

You may have the question on how do we actually find these magical eigen-stuff. All eigenvalues of a matrix can be found with a well-defined procedure: First, we notice that we are looking for the λ which satisfy

$Av = \lambda v$. It will be helpful to manipulate this to

$$Av = \lambda v \implies Av - \lambda v = 0 \implies (A - \lambda I)v = 0$$

Now we will look at what this equation is saying: We are selecting a value of λ such that there exists a non-trivial null space of $A - \lambda I$. There is a non-trivial null space of $A - \lambda I$ iff $\det(A - \lambda I) = 0$, so since we can express $\det(A - \lambda I)$ as a polynomial function of λ called the **characteristic polynomial**, the roots of this characteristic polynomial yield us the eigenvalues of A. For a given eigenvalue λ, we can find the space of corresponding eigenvectors called the **eigenspace** by solving for the non-trivial null space of $A - \lambda I$.

Awesome, we know that there are some special vectors for which applying a linear operator to them is pretty easy, because it's just scaling. What if we want to apply a linear operator to something that's not one of our special vectors and take advantage of this property? The answer is to express our vector in terms of eigenvectors via a change of basis transformation. We can do this when a full basis of eigenvectors, known as an **eigenbasis**, exists for the space our linear operator works in. This case occurs iff the linear operator is said to be **diagonalizable**.

Definition 3.3.10. Eigenbasis: An eigenbasis for an $n \times n$ matrix A would be a set of n linearly independent eigenvectors for A which form a basis for $\mathbb{R}^n$ or whatever n dimensional space A operates in.

Definition 3.3.11. Diagonalizable: A matrix is diagonalizable iff there exists a basis such that the matrix is completely diagonal when expressed within that basis; this basis would be an eigenbasis.

The relationship between full basis of eigenvectors and diagonalization can be better understood with the following work: For a matrix A, say were able to make an eigenbasis of n linearly independent eigenvectors. Compile the eigenbasis into a matrix $V = [v_1, v_2, \dots, v_n]$ and corresponding eigenvalues into a matrix

$$\Lambda = \begin{bmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & 0 & \dots 0 \\ \dots & \dots & \dots & \dots \\ 0 & \dots & 0 & \lambda_n \end{bmatrix}$$

for which each v_i, λ_i satisfy $Av_i = \lambda_i v_i$. We may extend this into $AV = V\Lambda$, and since V is invertible, this then means $A = V\Lambda V^{-1}$. Now we see the role of an eigenbasis in showing A's similarity with a diagonal matrix of eigenvalues.

If an $n \times n$ matrix A has n linearly independent eigenvectors, we can compile these into an eigenbasis and thus A would be diagonalizable. This is definitely not always the case; consider the matrix $\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$. If you were to go through the normal procedure to find eigenvalues via roots of the characteristic polynomial, you would see that there is a repeated root of $\lambda = 1$; repeated roots in the characteristic polynomial may be a sign that something's up and there might not be a full basis of eigenvectors. Sure enough, the null space of $A - \lambda I$ is just 1 dimensional, meaning we won't be able to complete a set of two linearly independent eigenvectors. This matrix sure enough has one dimension of eigenvectors, those in the direction of $\begin{bmatrix} 1 \\ 0 \end{bmatrix}$ with eigenvalue 1, but lacks a concrete second dimension. In cases like these, we consider a different matrix decomposition, the **Jordan Normal Form**, which exists for all matrices.

Definition 3.3.12. Jordan Normal Form For an $n \times n$ matrix A, we can define its Jordan normal form as a collection of Jordan blocks arranged along the diagonal of the matrix. Let's say A has m linearly independent eigenvectors. Each Jordan block looks like

$$J_i = \begin{bmatrix} \lambda_i & 1 & \dots & 0 & 0 & 0 \\ 0 & \lambda_i & 1 & \dots & 0 & 0 \\ 0 & 0 & \lambda_i & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & \lambda_i & 1 \end{bmatrix}$$

and the final Jordan form J may something like

$$J = \begin{bmatrix} J_1 & 0 & \dots & 0 & \\ 0 & J_2 & \dots & 0 & \\ \dots & \dots & \dots & \dots & \\ 0 & 0 & \dots & 0 & J_m \end{bmatrix}$$

Note: For $i \neq j$, we may have $\lambda_i = \lambda_j$. The number of distinct Jordan blocks is determined solely by the number of linearly independent eigenvectors we have for A.

As previously stated, we can not relate every matrix to a diagonal matrix with respect to a similarity transformation, although we can relate every matrix to its Jordan cannonical form with similarity transformations. This is due to the fact that the similarity transform for Jordan form uses a basis of generalized eigenvectors, which will always exist for any square matrix. A fun fact is that when the algebraic multiplicity (multiplicity of the root in characteristic polynomial) of each eigenvalue is equal to the number of linearly independent eigenvectors we can get for that eigenvalue, the operator is actually diagonalizable and the Jordan form for the operator is a diagonal matrix of eigenvalues. Essentially, when we can diagonalize, the Jordan Canonical form and eigenvector decomposition are equivalent.

3.3.1 Matrix properties

- Symmetric/Hermitian: A matrix A is symmetric/hermitian when $A = A^T / A = A^*$
  - All hermitian matrices can be represented as $A = B^T B$.
- Positive Definite: A matrix A is positive definite denoted as $A \succ 0$ when all eigenvalues are greater than or equal to zero, i.e. $x^T A x > 0, \forall x \neq 0$.
- Positive Semi-definite: A matrix A is positive semidefinite ($A \succeq 0$) when all eigenvalues are greater than or equal to zero, i.e. $x^T A x \geq 0, \forall x \neq 0$.
- Negative Definite: $A \prec 0 \implies x^T A x < 0, \forall x \neq 0$
- Negative Semi-definite: $A \preceq 0 \implies x^T A x \leq 0, \forall x \neq 0$.

3.4 Introduction to Linear Dynamical Systems

3.4.1 Linear Autonomous Systems (No Inputs)

We can model a linear dynamical system as the following differential equation: $\dot{x} = Ax$. Also, we can take any linear differential equation and turn it into a matrix:

$$\dddot{x} = 3\ddot{x} + 7\dot{x} + 2x \implies \begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dddot{x} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 2 & 7 & 3 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \ddot{x} \end{bmatrix}$$

However, we cannot turn the following differential equation into a matrix, $\dddot{x} = 3\ddot{x} + 7\dot{x} + 2x + x^2$, since this differential equation is not linear.

3.4.2 Stability

A system is stable (over some range of inputs) if the outputs to those inputs do not go to infinity.

In this course, we will use the following two theory of stability for linear and non-linear systems:

- Linear Systems: Hurwitz Stability
- Non-linear Systems: Lyapunov Stability

The following graphs are examples of two dynamical systems:

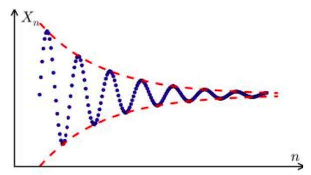

Figure 3.1: An example of a stable system

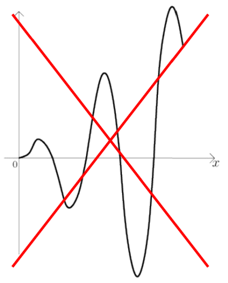

Figure 3.2: An example of an unstable system

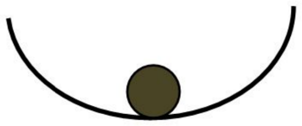

Figure 3.3: A stable system

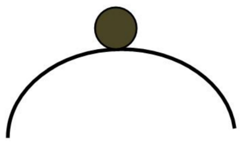

Figure 3.4: An unstable system

3.4.2.1 Matrix Exponential

The matrix exponential is an invaluable tool for solving linear differential equations. Given anything in the form of $\dot{x} = Ax$, we know we can find the solution as $x(t) = e^{At}x(0)$. How is this relevant to stability? We want our system state to not go to infinity, i.e. $\lim_{t\to\infty} e^{At}x(0) \neq \infty$.

To analyze the behavior of the above solution when t approaches infinity, we use the fact that if our dynamics (A) is diagonalizable, we can pull out the change of basis, i.e.:

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

We have three cases for stability of a dynamical system:

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

In the case that the matrix cannot be diagonalized, we use Jordan form. When we take the exponential, we can break up all the Jordan blocks, and integrate each separately. The following formula can be used to compute the matrix exponential for each Jordan block:

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

Looking at the above formula, we can see that when the real part of eigenvalues are zero then the systems are unstable since the value $te^{\lambda t}$ approaches infinity.

3.4.2.2 Lyapunov Stability

The Lyapunov equation looks at the change in system "energy." We can define an energy function such that it is 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 systems is globally stable.

For example, if we choose a quadratic energy function: $V(x) = x^T Px, P \succ 0$. We differentiate to get:

$$\begin{aligned} \dot{V}(x) &= \dot{x}^T Px + x^T P\dot{x} && (3.1) \\ &= x^T A^T Px + x^T PAx && (3.2) \\ &= x^T(A^T P + PA)x = x^T Qx && (3.3) \end{aligned}$$

If Q is negative definite, then the system is stable since it means the energy decreases at every point x.

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

3.4.3 Driven Systems

To model systems with inputs, we can use the following differential equation: $\dot{x} = Ax + Bu$

3.4.4 Transfer Function Notation

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

$$\dddot{x} = 2\ddot{x} + 7\dot{x} + 3x + u \tag{3.4}$$

$$s^3X = 2s^2X + 7sX + 3X + U \tag{3.5}$$

$$X = \frac{1}{s^3 - 2s^2 + 7s + 3}U \tag{3.6}$$

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

3.4.5 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:

$$\dot{x} = Ax + Bu \tag{3.7}$$

$$= Ax + B(-Kx) \tag{3.8}$$

$$= Ax - BKx \tag{3.9}$$

$$= (A - BK)x \tag{3.10}$$

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.

3.4.6 Linear Quadratic Regulator

LQR is the "optimal" way to find K. It is "optimal" in the sense that it is the best K that satisfies all of our constraints.

Define cost matrices $Q \in \mathbb{R}^{n\times n}$, $R \in \mathbb{R}^{u\times u}$, $Q, R \succ 0$. Usually, we set $R = I$ and manually tune Q.

LQR finds a K which minimizes the cost function:

$$\int_0^\infty (x^TQx + u^TRu + 2u^TNu)dt$$

Usually, we can ignore N (setting to 0).

In Matlab, we can use the command *lqr* to find run the LQR.