EE16B — Midterm 2 Review

Presented by: George Higgins Hutchinson, Naomi Sagan, Maran Palaniappan

Authors: George Higgins Hutchinson, Parth Nobel, Patrick Wang, Matteo Ciccozzi, Jaymo Kang, Brianna Zhang, Rehan Durrani, Sarah Sun

November 10, 2019

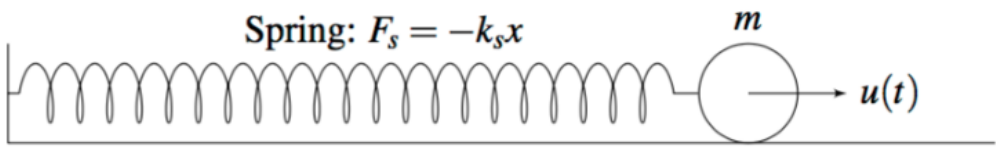
Disclaimer

This is an unofficial review session and HKN is not affiliated with this course. All of the topics we're reviewing will reflect the material you have covered, our experiences in EE16B, and past exams. We make no promise that what we cover will necessarily reflect the content of this midterm. While some course staff members may be among the presenters, this review session is still not official.

Overview

State Space Modeling: Example

Assume we have the following spring system:



We can model the system as a linear continuous time state space model:

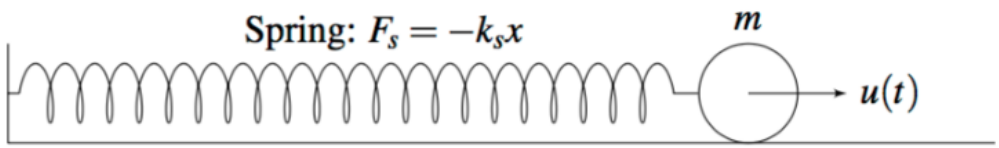
$$\frac{d}{dt}\vec{x}(t) = A\vec{x}(t) + \vec{b}u(t)$$
$$\vec{y}(t) = C\vec{x}(t)$$

in which:

$$\vec{x}(t) = \begin{bmatrix} x(t) \\ v(t) \end{bmatrix}, A = \begin{bmatrix} 0 & 1 \\ -\frac{k_s}{m} & 0 \end{bmatrix}, \vec{b} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \text{and } C = \begin{bmatrix} 0 & 1 \end{bmatrix}$$

State Space Modeling: Example

Assume we have the following spring system:



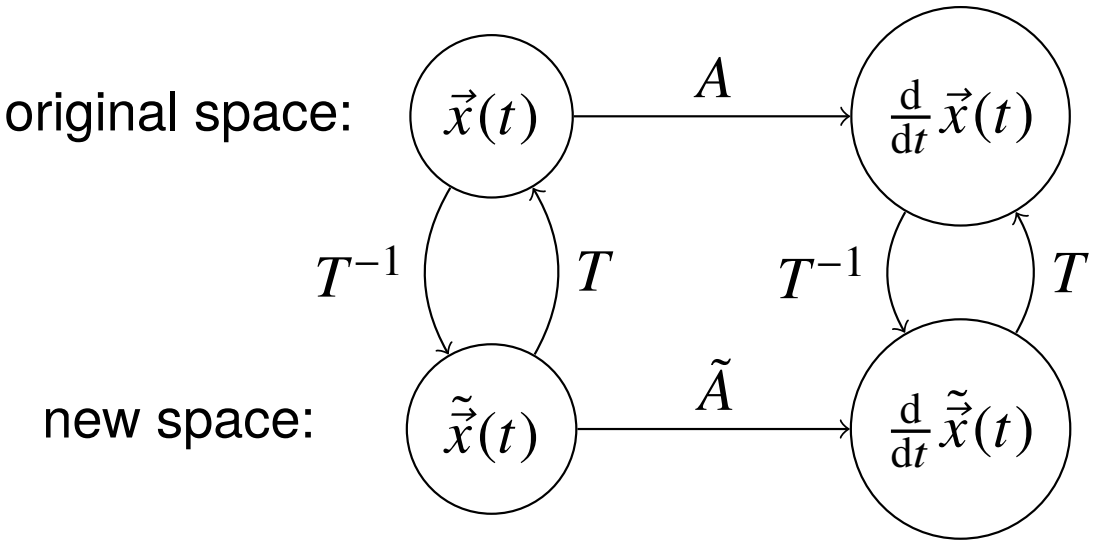
We can model the system as a linear continuous time state space model:

$$\frac{d}{dt}\vec{x}(t) = A\vec{x}(t) + \vec{b}u(t)$$
$$\vec{y}(t) = C\vec{x}(t)$$

in which:

$$\vec{x}(t) = \begin{bmatrix} x(t) \\ v(t) \end{bmatrix}, A = \begin{bmatrix} 0 & 1 \\ -\frac{k_s}{m} & 0 \end{bmatrix}, \vec{b} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \text{ and } C = \begin{bmatrix} 0 & 1 \end{bmatrix}$$

State Space Modeling:



Always apply operations to right side first

$$\tilde{A} = T^{-1}AT$$

$$\tilde{\vec{x}}(t) = T^{-1}\vec{x}(t)$$

State Space Modeling Procedure:

1. Set up differential equation of the form: $\frac{\mathrm{d}}{\mathrm{d}t}\vec{x}(t) = A\vec{x}(t) + \vec{b}u(t)$
2. Find eigenvectors $\vec{v}_i$ of A; let $T = \begin{bmatrix} \vec{v_1} & \vec{v_2} & \cdots & \vec{v_n} \end{bmatrix}$
3. Convert $\vec{x}(t)$ to $\tilde{\vec{x}}(t)$ using: $\tilde{\vec{x}}(t) = T^{-1}\vec{x}(t)$
4. Find λ_i of A; let $\tilde{A} = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & 0 & \vdots \\ \vdots & 0 & \ddots & 0 \\ 0 & \cdots & 0 & \lambda_n \end{bmatrix}$
5. Solve $\frac{d}{dt}\tilde{\vec{x}}(t) = \tilde{A}\tilde{\vec{x}}(t) + T\vec{b}u(t)$
6. Convert solution back to $\vec{x}(t)$

State Space Modeling Procedure:

1. Set up differential equation of the form: $\frac{\mathrm{d}}{\mathrm{d}t}\vec{x}(t) = A\vec{x}(t) + \vec{b}u(t)$
2. Find eigenvectors $\vec{v}_i$ of A; let $T = \begin{bmatrix} \vec{v_1} & \vec{v_2} & \cdots & \vec{v_n} \end{bmatrix}$
3. Convert $\vec{x}(t)$ to $\tilde{\vec{x}}(t)$ using: $\tilde{\vec{x}}(t) = T^{-1}\vec{x}(t)$
4. Find λ_i of A; let $\tilde{A} = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & 0 & \vdots \\ \vdots & 0 & \ddots & 0 \\ 0 & \cdots & 0 & \lambda_n \end{bmatrix}$
5. Solve $\frac{d}{dt}\tilde{\vec{x}}(t) = \tilde{A}\tilde{\vec{x}}(t) + T\vec{b}u(t)$
6. Convert solution back to $\vec{x}(t)$

State Space Modeling Procedure:

1. Set up differential equation of the form: $\frac{\mathrm{d}}{\mathrm{d}t}\vec{x}(t) = A\vec{x}(t) + \vec{b}u(t)$
2. Find eigenvectors $\vec{v}_i$ of A; let $T = \begin{bmatrix} \vec{v_1} & \vec{v_2} & \cdots & \vec{v_n} \end{bmatrix}$
3. Convert $\vec{x}(t)$ to $\tilde{\vec{x}}(t)$ using: $\tilde{\vec{x}}(t) = T^{-1}\vec{x}(t)$
4. Find λ_i of A; let $\tilde{A} = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & 0 & \vdots \\ \vdots & 0 & \ddots & 0 \\ 0 & \cdots & 0 & \lambda_n \end{bmatrix}$
5. Solve $\frac{d}{dt}\tilde{\vec{x}}(t) = \tilde{A}\tilde{\vec{x}}(t) + T\vec{b}u(t)$
6. Convert solution back to $\vec{x}(t)$

State Space Modeling Procedure:

1. Set up differential equation of the form: $\frac{\mathrm{d}}{\mathrm{d}t}\vec{x}(t) = A\vec{x}(t) + \vec{b}u(t)$
2. Find eigenvectors $\vec{v}_i$ of A; let $T = \begin{bmatrix} \vec{v_1} & \vec{v_2} & \cdots & \vec{v_n} \end{bmatrix}$
3. Convert $\vec{x}(t)$ to $\tilde{\vec{x}}(t)$ using: $\tilde{\vec{x}}(t) = T^{-1}\vec{x}(t)$
4. Find λ_i of A; let $\tilde{A} = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & 0 & \vdots \\ \vdots & 0 & \ddots & 0 \\ 0 & \cdots & 0 & \lambda_n \end{bmatrix}$
5. Solve $\frac{d}{dt}\tilde{\vec{x}}(t) = \tilde{A}\tilde{\vec{x}}(t) + T\vec{b}u(t)$
6. Convert solution back to $\vec{x}(t)$

State Space Modeling Procedure:

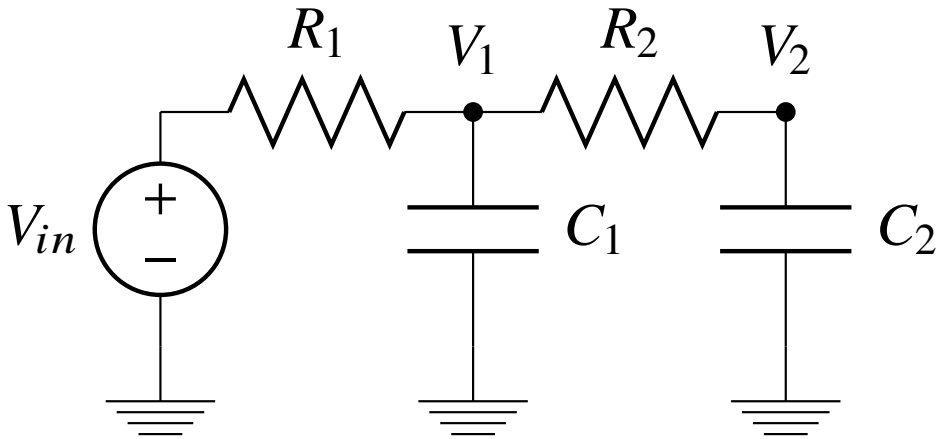
1. Set up differential equation of the form: $\frac{\mathrm{d}}{\mathrm{d}t}\vec{x}(t) = A\vec{x}(t) + \vec{b}u(t)$
2. Find eigenvectors $\vec{v}_i$ of A; let $T = \begin{bmatrix} \vec{v_1} & \vec{v_2} & \cdots & \vec{v_n} \end{bmatrix}$
3. Convert $\vec{x}(t)$ to $\tilde{\vec{x}}(t)$ using: $\tilde{\vec{x}}(t) = T^{-1}\vec{x}(t)$
4. Find λ_i of A; let $\tilde{A} = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & 0 & \vdots \\ \vdots & 0 & \ddots & 0 \\ 0 & \cdots & 0 & \lambda_n \end{bmatrix}$
5. Solve $\frac{d}{dt}\tilde{\vec{x}}(t) = \tilde{A}\tilde{\vec{x}}(t) + T\vec{b}u(t)$
6. Convert solution back to $\vec{x}(t)$

State Space Modeling Procedure:

1. Set up differential equation of the form: $\frac{\mathrm{d}}{\mathrm{d}t}\vec{x}(t) = A\vec{x}(t) + \vec{b}u(t)$
2. Find eigenvectors $\vec{v}_i$ of A; let $T = \begin{bmatrix} \vec{v_1} & \vec{v_2} & \cdots & \vec{v_n} \end{bmatrix}$
3. Convert $\vec{x}(t)$ to $\tilde{\vec{x}}(t)$ using: $\tilde{\vec{x}}(t) = T^{-1}\vec{x}(t)$
4. Find λ_i of A; let $\tilde{A} = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & 0 & \vdots \\ \vdots & 0 & \ddots & 0 \\ 0 & \cdots & 0 & \lambda_n \end{bmatrix}$
5. Solve $\frac{d}{dt}\tilde{\vec{x}}(t) = \tilde{A}\tilde{\vec{x}}(t) + T\vec{b}u(t)$
6. Convert solution back to $\vec{x}(t)$

State Space Modeling Example:

Given the following circuit:



in which $R_1 = 2\,\Omega$, $R_2 = \frac{8}{3}\Omega$, $C_1 = 1\,\text{C}$, $C_2 = \frac{3}{2}\text{C}$

solve equations for V_1 and V_2

Overview

Stability and Controllability:

given:

$\vec{x}(i+1) = A\vec{x}(i) + Bu(i)$

$\vec{y}(i) = C\vec{x}(i)$

in which:

$\vec{x}$ is our state,

$\vec{u}$ is our input,

$\vec{y}$ is what we can observe:

Stability (Discrete time):

Discrete time model:
if $|\lambda_i| < 1$ for all λ_i of A, system is stable
Intuition:

1. If any $|\lambda_i| > 1$, state vector is increasing each time step so will be infinitely magnified over time
2. If any $|\lambda_i| = 1$, state vector is constant each time step so if we were to integrate this, it would eventually blow up to infinity

Stability (Continuous time):

Continuous time model:
if the real parts of all eigenvalues of A are strictly negative, system is stable
Intuition:

1. If real part of eigenvalue is positive, state vector is increasing over time and will be infinitely magnified over time

Controllability:

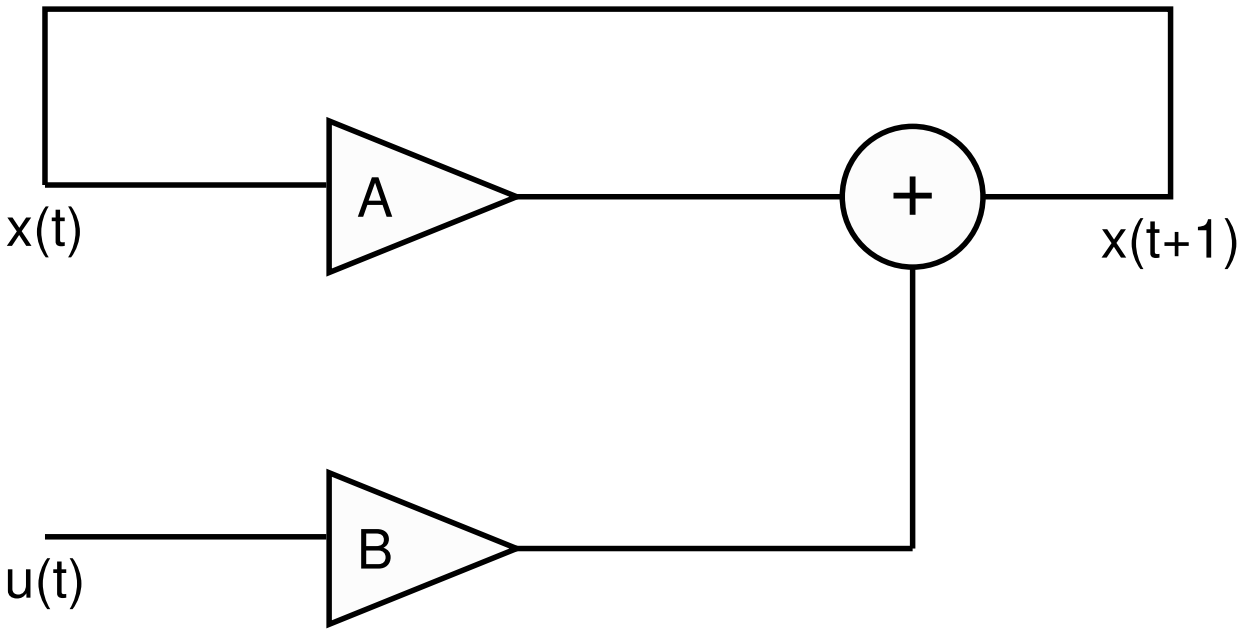
if $\begin{bmatrix} B & AB & \dots & A^{n-1}B \end{bmatrix}$ spans R^n, then system is controllable

Intuition: If you can go to any location in an n-dimensional space in infinite steps, that means you can go to any location in that space in n steps as well, implying that the matrix is full rank.

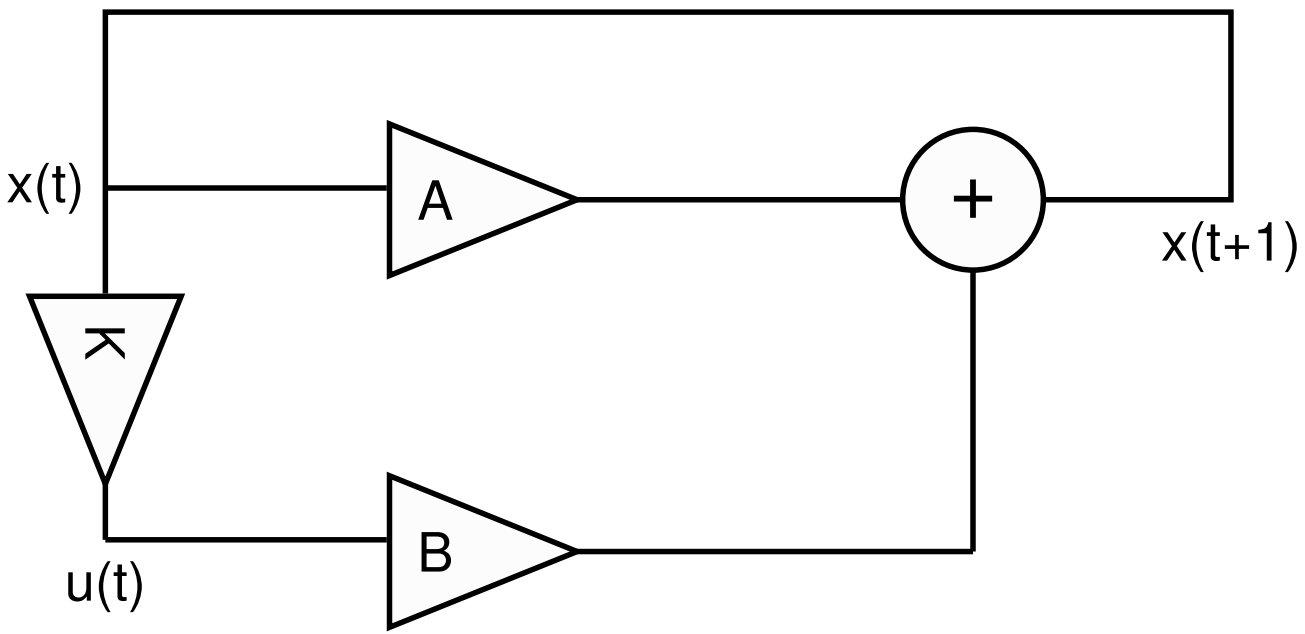
Feedback:

1. If system is controllable, we can set: $u(t) = K\vec{x}(t)$
   Plugging in, we get: $\vec{x}(t+1) = (A + BK)\vec{x}(t)$
   We can find the eigenvalues of (A + BK) to check for stability
2. If this system has noise, we can modify the equation to be:
   $\vec{x}(t+1) = (A + BK)\vec{x}(t) + \vec{\omega}(t)$
   Where $\vec{\omega}$ is noise.

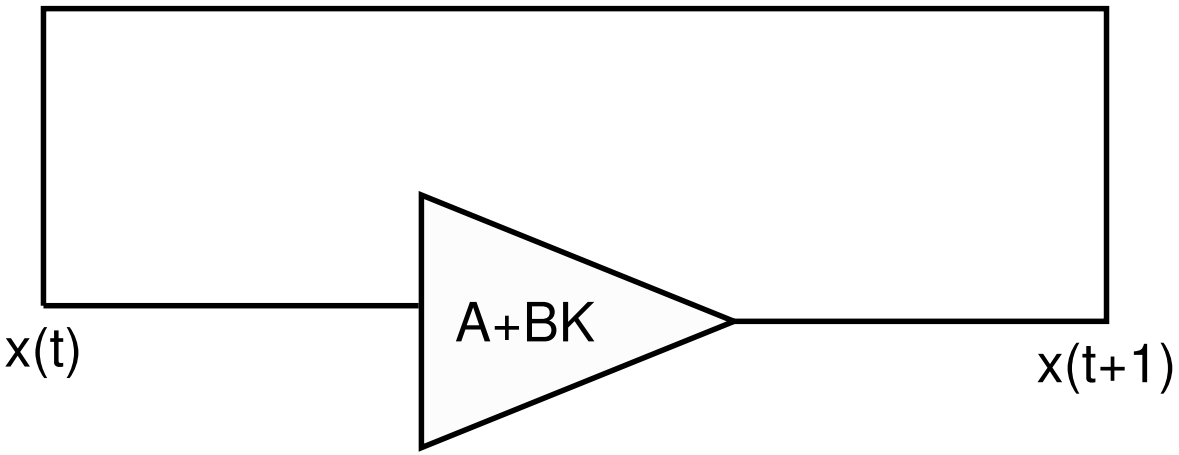
Feedback Flow Diagram:



Feedback Flow Diagram:



Feedback Flow Diagram:



Stability and Controllability Example:

given the following system:

$$\vec{x}[t+1] = \begin{bmatrix} -5 & 0 \\ 7 & 6 \end{bmatrix} \vec{x}[t] + \begin{bmatrix} 2 \\ -1 \end{bmatrix} u[t]$$

Stability Check:

Remember: $\vec{x}[t+1] = A\vec{x}[t] + \vec{\omega}[t] = U\Lambda U^T \vec{x}[t] + \vec{\omega}[t]$

$\lambda = 6, -5$

System is unstable

Stability Check:

Remember: $\vec{x}[t+1] = A\vec{x}[t] + \vec{\omega}[t] = U\Lambda U^T \vec{x}[t] + \vec{\omega}[t]$

$\lambda = 6, -5$

System is unstable

Controllability Check:

$$\begin{bmatrix} AB & B \end{bmatrix} = \begin{bmatrix} -10 & 2 \\ 8 & -1 \end{bmatrix}$$ which spans R^n

System is controllable

* Use results from previous slides to figure this out.

Controllability Check:

$$\begin{bmatrix} AB & B \end{bmatrix} = \begin{bmatrix} -10 & 2 \\ 8 & -1 \end{bmatrix}$$ which spans R^n

System is controllable

* Use results from previous slides to figure this out.

Overview

Eigenvalue Placement

Why?

- Recall that we are always interested in determining if a given system is BIBO (bounded input bounded output) stable.
- More precisely, if we have a system described by $\vec{x}(t+1) = A\vec{x}(t) + Bu(t) + \vec{\omega}(t)$ we would like the eigenvalues of $A \in \mathbb{R}^{n \times n}$, to satisfy the following property : $|\lambda_i| < 1$.
- So what if we have a λ that does not satisfy this property?
- This is where eigenvalue placement comes into play!
- Assuming the system is controllable, we will use closed loop controls to change the eigenvalues such that they satisfy this property.

How?

- Assume e.g. a DT system. Input: $u[t]$ If the system is controllable then we can use feedback, which means that we can let the input depend on the output, $\vec{x}[t]$.
- We would like to change the matrix multiplying $\vec{x}[t]$ such that $|\lambda_i| < 1$, so let's see what happens when we let $u[t] = K\vec{x}[t]$, where $K \in \mathbb{R}^{1\times n}$.
- Using this input we have:

$$\begin{aligned}\vec{x}[t+1] &= A\vec{x}[t] + Bu[t] + \vec{\omega}[t] \\ &= A\vec{x}[t] + BK\vec{x}[t] + \vec{\omega}[t] \\ &= (A + BK)\vec{x}[t] + \vec{\omega}[t]\end{aligned}$$

- Strategically choosing K allows us to have specific λ's for A + BK (Good!).
- This process is called coefficient matching.

Example

- Suppose we are given a controllable system defined by:

$$\vec{x}[t+1] = \begin{bmatrix} 2 & -1 \\ 0 & 1 \end{bmatrix} \vec{x}[t] + \begin{bmatrix} 2 \\ -1 \end{bmatrix} u[t]$$

- Is the system stable? No! $\lambda = 2, 1$
- What if we let

$$u[t] = \begin{bmatrix} f_1 & f_2 \end{bmatrix} \vec{x}[t]$$

Then we have:

$$\vec{x}(t+1) = \begin{bmatrix} -2 & -1 \\ 0 & 1 \end{bmatrix} \vec{x}[t] + \begin{bmatrix} 2 \\ -1 \end{bmatrix} \begin{bmatrix} f_1 & f_2 \end{bmatrix} \vec{x}[t]$$

- Solve for the values of f_1 and f_2 such that $\lambda_1 = -0.25$ and $\lambda_2 = 0$
- The answer is $f_1 = -1.50$ and $f_2 = 0.25$
- Although the process is very messy hopefully you see why eigenvalue placement is very important to stabilize systems. What about bigger matrices?

Example

- Suppose we are given a controllable system defined by:

$$\vec{x}[t+1] = \begin{bmatrix} 2 & -1 \\ 0 & 1 \end{bmatrix} \vec{x}[t] + \begin{bmatrix} 2 \\ -1 \end{bmatrix} u[t]$$

- Is the system stable? No! $\lambda = 2, 1$
- What if we let

$$u[t] = \begin{bmatrix} f_1 & f_2 \end{bmatrix} \vec{x}[t]$$

Then we have:

$$\vec{x}(t+1) = \begin{bmatrix} -2 & -1 \\ 0 & 1 \end{bmatrix} \vec{x}[t] + \begin{bmatrix} 2 \\ -1 \end{bmatrix} \begin{bmatrix} f_1 & f_2 \end{bmatrix} \vec{x}[t]$$

- Solve for the values of f_1 and f_2 such that $\lambda_1 = -0.25$ and $\lambda_2 = 0$
- The answer is $f_1 = -1.50$ and $f_2 = 0.25$
- Although the process is very messy hopefully you see why eigenvalue placement is very important to stabilize systems. What about bigger matrices?

Example

- Suppose we are given a controllable system defined by:

$$\vec{x}[t+1] = \begin{bmatrix} 2 & -1 \\ 0 & 1 \end{bmatrix} \vec{x}[t] + \begin{bmatrix} 2 \\ -1 \end{bmatrix} u[t]$$

- Is the system stable? No! $\lambda = 2, 1$
- What if we let

$$u[t] = \begin{bmatrix} f_1 & f_2 \end{bmatrix} \vec{x}[t]$$

Then we have:

$$\vec{x}(t+1) = \begin{bmatrix} -2 & -1 \\ 0 & 1 \end{bmatrix} \vec{x}[t] + \begin{bmatrix} 2 \\ -1 \end{bmatrix} \begin{bmatrix} f_1 & f_2 \end{bmatrix} \vec{x}[t]$$

- Solve for the values of f_1 and f_2 such that $\lambda_1 = -0.25$ and $\lambda_2 = 0$
- The answer is $f_1 = -1.50$ and $f_2 = 0.25$
- Although the process is very messy hopefully you see why eigenvalue placement is very important to stabilize systems. What about bigger matrices?

Example

- Suppose we are given a controllable system defined by:

$$\vec{x}[t+1] = \begin{bmatrix} 2 & -1 \\ 0 & 1 \end{bmatrix} \vec{x}[t] + \begin{bmatrix} 2 \\ -1 \end{bmatrix} u[t]$$

- Is the system stable? No! $\lambda = 2, 1$
- What if we let

$$u[t] = \begin{bmatrix} f_1 & f_2 \end{bmatrix} \vec{x}[t]$$

Then we have:

$$\vec{x}(t+1) = \begin{bmatrix} -2 & -1 \\ 0 & 1 \end{bmatrix} \vec{x}[t] + \begin{bmatrix} 2 \\ -1 \end{bmatrix} \begin{bmatrix} f_1 & f_2 \end{bmatrix} \vec{x}[t]$$

- Solve for the values of f_1 and f_2 such that $\lambda_1 = -0.25$ and $\lambda_2 = 0$
- The answer is $f_1 = -1.50$ and $f_2 = 0.25$
- Although the process is very messy hopefully you see why eigenvalue placement is very important to stabilize systems. What about bigger matrices?

Break

Please give us feedback @ `hkn.mu/feedback`!

Let us know of any issues in the slides @ `https://github.com/hkntutoring/ee16b-review/issues`!

Overview

Schur Decomposition

- Start with $M_n = A$
- Find an eigenvalue/eigenvector pair λ_i, v_i for M_{n-i+1}
- Find R_{n-i} such that $R_{n-i}^T v_i = \vec{0}$ and $R_{n-i}^T R_{n-i} = I$ by Gram Schmidt.
- Find the next M with $M_{n-i} = R_{n-i}^T M_{n-i+1} R_{n-i}$
- $$U = [v_1, R_{n-1} v_2, R_{n-1} R_{n-2} v_3 \ldots]$$
- $$A = UMU^T \implies M = U^T AU$$

Schur Decomposition

- Start with $M_n = A$
- Find an eigenvalue/eigenvector pair λ_i, v_i for M_{n-i+1}
- Find R_{n-i} such that $R_{n-i}^T v_i = \vec{0}$ and $R_{n-i}^T R_{n-i} = I$ by Gram Schmidt.
- Find the next M with $M_{n-i} = R_{n-i}^T M_{n-i+1} R_{n-i}$
- $$U = [v_1, R_{n-1} v_2, R_{n-1} R_{n-2} v_3 \ldots]$$
- $$A = UMU^T \implies M = U^T AU$$

Schur Decomposition

- Start with $M_n = A$
- Find an eigenvalue/eigenvector pair λ_i, v_i for M_{n-i+1}
- Find R_{n-i} such that $R_{n-i}^T v_i = \vec{0}$ and $R_{n-i}^T R_{n-i} = I$ by Gram Schmidt.
- Find the next M with $M_{n-i} = R_{n-i}^T M_{n-i+1} R_{n-i}$
- $$U = [v_1, R_{n-1} v_2, R_{n-1} R_{n-2} v_3 \ldots]$$
- $$A = UMU^T \implies M = U^T AU$$

Schur Decomposition

- Start with $M_n = A$
- Find an eigenvalue/eigenvector pair λ_i, v_i for M_{n-i+1}
- Find R_{n-i} such that $R_{n-i}^T v_i = \vec{0}$ and $R_{n-i}^T R_{n-i} = I$ by Gram Schmidt.
- Find the next M with $M_{n-i} = R_{n-i}^T M_{n-i+1} R_{n-i}$
- $$U = [v_1, R_{n-1} v_2, R_{n-1} R_{n-2} v_3 \ldots]$$
- $$A = UMU^T \implies M = U^T AU$$

Schur Decomposition

- Start with $M_n = A$
- Find an eigenvalue/eigenvector pair λ_i, v_i for M_{n-i+1}
- Find R_{n-i} such that $R_{n-i}^T v_i = \vec{0}$ and $R_{n-i}^T R_{n-i} = I$ by Gram Schmidt.
- Find the next M with $M_{n-i} = R_{n-i}^T M_{n-i+1} R_{n-i}$
- $$U = [v_1, R_{n-1} v_2, R_{n-1} R_{n-2} v_3 \ldots]$$
- $$A = UMU^T \implies M = U^T AU$$

Schur Decomposition

- Start with $M_n = A$
- Find an eigenvalue/eigenvector pair λ_i, v_i for M_{n-i+1}
- Find R_{n-i} such that $R_{n-i}^T v_i = \vec{0}$ and $R_{n-i}^T R_{n-i} = I$ by Gram Schmidt.
- Find the next M with $M_{n-i} = R_{n-i}^T M_{n-i+1} R_{n-i}$
- $$U = [v_1, R_{n-1} v_2, R_{n-1} R_{n-2} v_3 \ldots]$$
- $$A = UMU^T \implies M = U^T AU$$

Upper Triangularization

By Schur Decomposition, with $A = UMU^T$, you end up with a matrix of the form:

$$\begin{bmatrix} \lambda_1 & * & * & \cdots \\ 0 & \lambda_2 & * & \cdots \\ 0 & 0 & \lambda_2 & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

Spectral Theorem

If $A = A^T$ is a symmetric matrix,

- All eigenvalues are real
- Real eigenvectors exist
- Eigenvectors form an orthonormal basis for $\mathbb{R}^n$

Spectral Theorem

If $A = A^T$ is a symmetric matrix,

- All eigenvalues are real
- Real eigenvectors exist
- Eigenvectors form an orthonormal basis for $\mathbb{R}^n$

Spectral Theorem

If $A = A^T$ is a symmetric matrix,

- All eigenvalues are real
- Real eigenvectors exist
- Eigenvectors form an orthonormal basis for $\mathbb{R}^n$

Practice Question

- We know how to solve

$$A\vec{x} \approx \vec{y}$$

by viewing it as

$$\min_{\vec{x}} ||\vec{y} - A\vec{x}||^2 = \min_{\vec{x}} (\vec{y} - A\vec{x})^T (\vec{y} - A\vec{x})$$

- What if we wanted to minimize:

$$||C(\vec{y} - A\vec{x})||^2$$

- Simply expand into

$$||C\vec{y} - CA\vec{x}||$$

and solve as before with $\vec{y} = C\vec{y}$ and $A = CA$.

Practice Question

- We know how to solve

$$A\vec{x} \approx \vec{y}$$

by viewing it as

$$\min_{\vec{x}} ||\vec{y} - A\vec{x}||^2 = \min_{\vec{x}} (\vec{y} - A\vec{x})^T (\vec{y} - A\vec{x})$$

- What if we wanted to minimize:

$$||C(\vec{y} - A\vec{x})||^2$$

- Simply expand into

$$||C\vec{y} - CA\vec{x}||$$

and solve as before with $\vec{y} = C\vec{y}$ and $A = CA$.

Practice Question

- We know how to solve

$$A\vec{x} \approx \vec{y}$$

by viewing it as

$$\min_{\vec{x}} ||\vec{y} - A\vec{x}||^2 = \min_{\vec{x}} (\vec{y} - A\vec{x})^T (\vec{y} - A\vec{x})$$

- What if we wanted to minimize:

$$||C(\vec{y} - A\vec{x})||^2$$

- Simply expand into

$$||C\vec{y} - CA\vec{x}||$$

and solve as before with $\vec{y} = C\vec{y}$ and $A = CA$.

Practice Question

- We know how to minimize

$$||C(\vec{y} - A\vec{x})||^2$$

- What if we wanted to minimize:

$$(\vec{y} - A\vec{x})^T P(\vec{y} - A\vec{x})$$

  where P is symmetric.
- Recall by the spectral theorem that P can be decomposed into $U\Lambda U^T$.
- Let $\Lambda_{1/2}$ denote Λ with its diagonal square rooted.
- We know that $\Lambda_{1/2} = \Lambda_{1/2}^T$ because it is symmetrical.
- Simply solve with $C = \Lambda_{1/2}U$.

Overview

SVD Theorem

Any matrix $A \in \mathbb{R}^{m \times n}$ can be decomposed into the product of three matrices

$$A = U\Sigma V^T$$
$$U : m \times m$$
$$\Sigma : m \times n$$
$$V^T : n \times n$$

Such that U, V are unitary matrices and Σ only has nonnegative values along its main diagonal.

SVD: Compact Form

We can also express the SVD as

$$A = \mathcal{U} S \mathcal{V}^T$$
$$\mathcal{U} : m \times r$$
$$S : r \times r$$
$$\mathcal{V}^T : r \times n$$

where r is the rank of A. The compact form matrices maintain properties of the original matrices, but have entries removed whenever they correspond to zero singular values.

SVD: Outer Product Form

Lastly, we can express

$$A = \sum_{i=1}^{r} \sigma_i \vec{u}_i \vec{v}_i^T$$

where $\vec{u}_i, \vec{v}_i$ are the columns of U, V, respectively, and σ_i are corresponding diagonal entry of the matrix Σ

Computing SVD with $A^T A$

$$\begin{aligned} A^T A &= V \Sigma^T U^T U \Sigma U V^T \\ &= V \Sigma^2 V^T \end{aligned}$$

This is an eigen decomposition since Σ^2 is diagonal and $V^{-1} = V^T$. Thus solving for the eigenvalues and eigenvectors of $A^T A$ give $\lambda_i = \sigma_i^2$ with eigenvectors which correspond to the right singular vectors. We need to sort by decreasing σ_i.

Side note: $\Sigma^T \Sigma$ is not actually equal to Σ^2, but the former product yields a matrix with singular values squared on the diagonal entries, hence we call it Σ^2

Computing SVD with $A^T A$

Given a right singular vector $\vec{v}_i$ which we found from the previous part, we can apply it

$$\begin{aligned} A\vec{v}_i &= \left(\sum_{k=1}^{r} \sigma_k \vec{u}_k \vec{v}_k^T \right) \vec{v}_i \\ &= \sum_{k=1}^{r} \sigma_k \vec{u}_k \delta_k^i \\ &= \sigma_i \vec{u}_i \\ \vec{u}_i &= \frac{1}{\sigma_i} A\vec{v}_i \end{aligned}$$

Computing SVD with AA^T

Similar calculations yield $\sigma_i = \sqrt{\lambda_i}$ of AA^T with eigenvectors as left singular vectors, and $\vec{v}_i = \frac{1}{\sigma_i} A^T \vec{u}_i$

Intepretation of SVD

- Unitary matrices act as rotation in a given space. A diagonal matrix stretches in a given coordinate space.
- SVD visualization (open in browser)

Intepretation of SVD

For a product $A\vec{x}$, we can decompose every vector $\vec{x}$ into a linear combination of right singular vectors

$$\vec{x} = \sum_{i=1}^{n} \alpha_i \vec{v}_i$$

Thus, we can see exactly which parts of $\vec{x}$ affect the output.

Compression of Low-Rank Matrices

- Suppose I had a matrix $A \in \mathbb{R}^{m \times n}$ with $m, n >> rank(A)$. How could I more efficiently store A and compute products like $A\vec{x}$?

- With the SVD, we only have to save r set of two vectors and a scalar, which saves us a lot of space if the rank is small with respect to the matrix. Also, less computation is carried out if we represent the matrix as the outer product form.

Compression of Low-Rank Matrices

- Suppose I had a matrix $A \in \mathbb{R}^{m \times n}$ with $m, n >> rank(A)$. How could I more efficiently store A and compute products like $A\vec{x}$?

- With the SVD, we only have to save r set of two vectors and a scalar, which saves us a lot of space if the rank is small with respect to the matrix. Also, less computation is carried out if we represent the matrix as the outer product form.

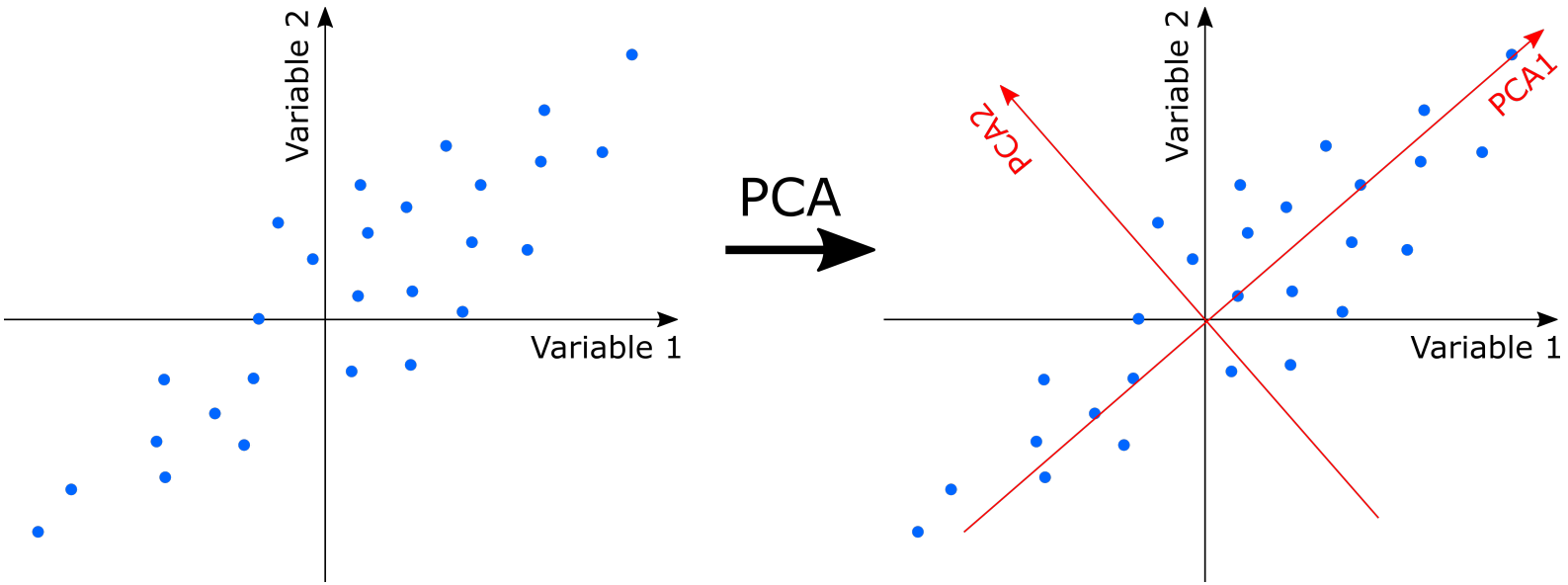
Overview

PCA

PCA is a linear dimensionality reduction tool. Given data $\vec{x}_i \in \mathbb{R}^d$, we can create a mapping $T : \mathbb{R}^d \to \mathbb{R}^{d'}, d' < d$ such that the variance in the dataset is still captured

PCA captures the direction along which the data varies the most. (Direction of Maximal Variance)

PCA Illustrated



PCA — Computation

1. Store data row-major in $A \in \mathbb{R}^{n \times d}$
2. De-mean A *Step is optional, only used for specific scenarios*
3. Take SVD: $A = U\Sigma V^T$
4. Create $V_{d'} \in \mathbb{R}^{n \times d'}$ from vectors of V corresponding to d' greatest signular values
5. To project data into the representative subspace: $T(x) := V_{d'}^T x$

PCA — Computation

1. Store data row-major in $A \in \mathbb{R}^{n \times d}$
2. De-mean A *Step is optional, only used for specific scenarios*
3. Take SVD: $A = U\Sigma V^T$
4. Create $V_{d'} \in \mathbb{R}^{n \times d'}$ from vectors of V corresponding to d' greatest signular values
5. To project data into the representative subspace: $T(x) := V_{d'}^T x$

PCA — Computation

1. Store data row-major in $A \in \mathbb{R}^{n \times d}$
2. De-mean A *Step is optional, only used for specific scenarios*
3. Take SVD: $A = U\Sigma V^T$
4. Create $V_{d'} \in \mathbb{R}^{n \times d'}$ from vectors of V corresponding to d' greatest signular values
5. To project data into the representative subspace: $T(x) := V_{d'}^T x$

PCA — Computation

1. Store data row-major in $A \in \mathbb{R}^{n \times d}$
2. De-mean A *Step is optional, only used for specific scenarios*
3. Take SVD: $A = U\Sigma V^T$
4. Create $V_{d'} \in \mathbb{R}^{n \times d'}$ from vectors of V corresponding to d' greatest signular values
5. To project data into the representative subspace: $T(x) := V_{d'}^T x$

PCA — Computation

1. Store data row-major in $A \in \mathbb{R}^{n \times d}$
2. De-mean A *Step is optional, only used for specific scenarios*
3. Take SVD: $A = U\Sigma V^T$
4. Create $V_{d'} \in \mathbb{R}^{n \times d'}$ from vectors of V corresponding to d' greatest signular values
5. To project data into the representative subspace: $T(x) := V_{d'}^T x$

PCA: computation

The mapping T can then be expressed as

$$T(\vec{x}) = V_k^T \vec{x}$$

If we apply this transformation onto the entire dataset (which has row vectors), we can say

$$T(A) = B = AV_k$$

where $B \in \mathbb{R}^{n \times k}$

PCA: computation

If we were to show the projected vectors in the original space, we can multiply back with the projection vectors

$$A' = BV_k^T$$

Overview

Discretization: Q1

Note: this section follows hw8 q1 almost exactly. Suppose we have a scalar system

$$\frac{d}{dt}x(t) = \alpha x + \vec{\beta}^T \vec{u}(t)$$

and we apply a constant input $\vec{u}_n$ for times $t \in [nT, (n+1)T)$ for some $T > 0$. Given $x(nT)$ solve the differential equation

Discretization: Q1 Sol

From $t = nT$ to $t = (n+1)T$, $\vec{\beta}^T \vec{u}$ is a constant scalar. Thus, we can solve this like a normal differential equation. Let $x = x' - \frac{\vec{\beta}^T \vec{u}}{\alpha}$. Then

$$\frac{d}{dt}x(t) = \alpha\left(x' - \frac{\vec{\beta}^T \vec{u}}{\alpha}\right) + \vec{\beta}^T \vec{u}(t)$$

$$\mathrm{d}t\, x = \mathrm{d}t\, x' = \alpha x'$$

$$x' = Ae^{\alpha(t-nT)}, \text{ for some integration constant } A$$

$$x + \frac{\vec{\beta}^T \vec{u}}{\alpha} = Ae^{\alpha(t-nT)}$$

$$x = Ae^{\alpha(t-nT)} - \frac{\vec{\beta}^T \vec{u}}{\alpha}$$

Discretization: Q1 Sol

At which point we can use our initial condition to get

$$x(nT) = A - \frac{\vec{\beta}^T \vec{u}}{\alpha}$$

$$A = x(nT) + \frac{\vec{\beta}^T \vec{u}}{\alpha}$$

$$x = \left(x(nT) + \frac{\vec{\beta}^T \vec{u}}{\alpha} \right) e^{\alpha(t-nT)} - \frac{\vec{\beta}^T \vec{u}}{\alpha}$$

Discretization: Q2

Using the differential equation derived from question 1, create a discrete-time system to model the continuous time. In other words, if $x[n] = x(nT), \vec{u}[n] = \vec{u}(nT)$, find a relation such that

$$x[n+1] = A_d x[n] + B_d \vec{u}[n]$$

Discretization: Q2 Sol

We can solve the previous solution for $x((n+1)T)$

$$x((n+1)T) = \left(x(nT) + \frac{\vec{\beta}^T \vec{u}(nT)}{\alpha}\right) e^{\alpha((n+1)T - nT)} - \frac{\vec{\beta}^T \vec{u}(nT)}{\alpha}$$

$$x[n+1] = e^{\alpha T} x[n] + \frac{e^{\alpha T} - 1}{\alpha} \vec{\beta}^T \vec{u}[n]$$

We see that $A_d = e^{\alpha T}, B_d = ((e^{\alpha T} - 1)/\alpha)\vec{\beta}^T$

Discretization: Q3

Instead of a scalar, we instead have a diagonal matrix A such that

$$\frac{d}{dt}\vec{x} = A\vec{x} + B\vec{u}$$

Discretize this system in the same was as Q2.

Discretiziation: Q3 Sol

Expanding the original system out line-by-line gives

$$\frac{d}{dt}x_i = a_i x_i + b_i \vec{u}_i$$

where x_i is the ith variable of $\vec{x}$, a_i is the diagonal entry of A, and b_i is the row of B.

Discretization: Generic Matrix

Math not shown, but we can perform a change of basis from our original space to our diagonal space, and then apply the results of the previous part.

Overview

Notes

These are some potential topics that would be good to look at

1. Proofs for all the theorems we mentioned
2. Outlier Rejection

Thank You!

Thank you everyone for coming and good luck on your midterm!
In addition, HKN volunteers will be giving out oreos at the end of your midterm so please take one if you want!