EECS 16B Final Review Session

Presented by <George Higgins Hutchinson, Bryan Ngo, Vincent Wang >(HKN)

May 4, 2021

Disclaimer

This is an unofficial review session and HKN is not affiliated with this course. Although some of the presenters may be course staff, the material covered in the review session may not be an accurate representation of the topics covered in and difficulty of the exam.

Slides are also posted at @2229 on Piazza.

HKN Drop-In Tutoring

- HKN has office hours Monday, Wednesday, and Friday from **1 PM - 3 PM** and **8 PM - 10 PM** on hkn.mu/ohqueue
- The schedule of tutors can be found at hkn.mu/tutor

Controls

Reviewing State Space

Discrete Time State Space Model:

$$\vec{x}[k+1] = A\vec{x}[k] + B\vec{u}[k]$$

Where $\vec{x}[\cdot]$ as the state vector, $u[\cdot]$ as the input vector.

Controllability

Goal: Modify $x(t)$ to be in any state we desire.

$$\vec{x}[t+1] = A\vec{x}[t] + B\vec{u}[t]$$

Expand out $x[t]$ in terms of the initial state and all inputs,

$$\vec{x}(t) = A^t\vec{x}(0) + A^{t-1}Bu(0) + A^{t-2}Bu(1) + \cdots + ABu(t-2) + Bu(t-1)$$

$$\vec{x}(t) - A^t\vec{x}(0) = \underbrace{\begin{bmatrix} A^{t-1}B & A^{t-2}B & \cdots & AB & B \end{bmatrix}}_{\triangleq R_t} \begin{bmatrix} u(0) \\ u(1) \\ \vdots \\ u(t-2) \\ u(t-1) \end{bmatrix}$$

Controllability

Goal: Modify $x(t)$ to be in any state we desire.

$$\vec{x}[t+1] = A\vec{x}[t] + B\vec{u}[t]$$

Expand out $x[t]$ in terms of the initial state and all inputs,

$$\vec{x}(t) = A^t\vec{x}(0) + A^{t-1}Bu(0) + A^{t-2}Bu(1) + \cdots + ABu(t-2) + Bu(t-1)$$

$$\vec{x}(t) - A^t\vec{x}(0) = \underbrace{\begin{bmatrix} A^{t-1}B & A^{t-2}B & \cdots & AB & B \end{bmatrix}}_{\triangleq R_t} \begin{bmatrix} u(0) \\ u(1) \\ \vdots \\ u(t-2) \\ u(t-1) \end{bmatrix}$$

Given the initial condition, $x(0)$ the output of the system can be expressed in terms of the solely our inputs!

What states can we change $x(t)$ to?

$$\vec{x}(t) - A^t\vec{x}(0) = \underbrace{\begin{bmatrix} A^{t-1}B & A^{t-2}B & \cdots & AB & B \end{bmatrix}}_{\triangleq R_t} \begin{bmatrix} u(0) \\ u(1) \\ \vdots \\ u(t-2) \\ u(t-1) \end{bmatrix}$$

The $\mathrm{Col}(R_t)$ determines the subspace $\vec{u}(t)$ can map to.

In order to control the state to any vector in $\mathbb{R}^n$, $\mathrm{Col}(R_t) = R^n$, or it must be full rank.

i.e. The system is Controllable if and only if

$$\operatorname{rank} R_n = \operatorname{rank} \begin{bmatrix} A^{n-1}B & A^{n-2}B & \cdots & AB & B \end{bmatrix} = n$$

Stability

Stability in Discrete Time

A discrete system is stable iff all eigenvalues have magnitude less than 1. If any eigenvalue has magnitude greater than 1, then any state vector with a nonzero corresponding eigenvector component will have that component repeatedly magnified.

For example: $x[t+1] = 2x[t]$

Stability in Discrete Time

A discrete system is stable iff

$$\forall x \in eig(A) : |x| < 1$$

The eigenvectors form a basis (called the eigenbasis) which spans the entire space if A is full rank. (can you prove this?)

If any eigenvalue has magnitude greater than 1, then any state vector with a nonzero corresponding eigenvector component will have that component repeatedly magnified.

Stability in Discrete Time

How do the eigenvalues govern system dynamics?

If initial state is x(0), and there's no control input, the *n*th state is

$$x(n) = A^n x(0)$$

If any eigenvalue of A is larger in magnitude than 1, it "blows up" through repeated exponentiation - the system destabilizes!

Stability in Discrete Time

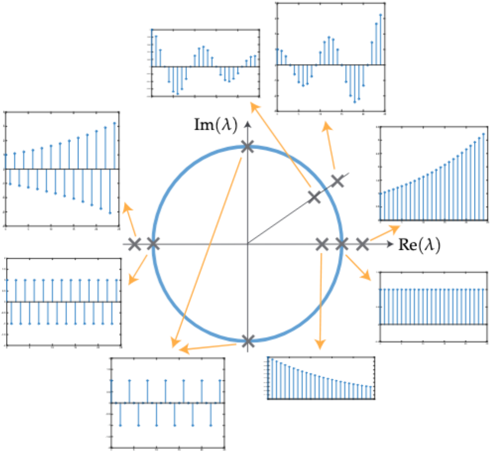


Figure 1: The real part of λ^t for various values of λ in the complex plane. It grows unbounded when $|\lambda| > 1$, decays to zero when $|\lambda| < 1$, and has constant amplitude when λ is on the unit circle ($|\lambda| = 1$).

Stability in Continuous Time

A continuous system is stable iff the real parts of all eigenvalues are negative. If any eigenvalue is positive, then any state vector with a nonzero corresponding eigenvector component will have that component grow exponentially to infinity.

For example: $\frac{\mathrm{d}}{\mathrm{d}t}x(t) = 2\,x(t)$

Stability in Continuous Time

$$\frac{\mathrm{d}}{\mathrm{d}t}x(t) = ax(t) + bu(t)$$

$$x(t) = e^{at}x(0) + b\int_0^t e^{a(t-s)}u(s)\,\mathrm{d}s$$

For scalar case, system is stable if $\mathrm{Re}\{a\} < 0$ and not stable if $\mathrm{Re}\{a\} > 0$.

By careful application of diagonalization, we get the same result for the eigenvalues of A in the matrix case.

Stability in Continuous Time

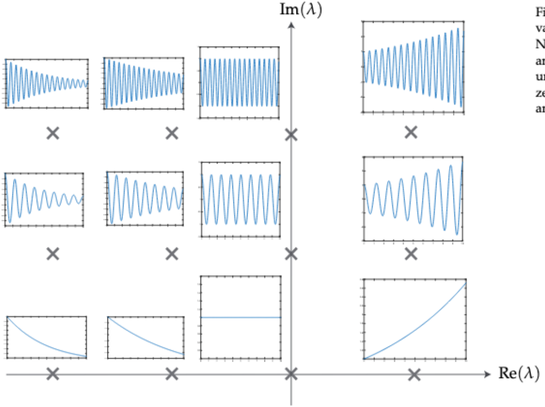


Figure 2: The real part of $e^{\lambda t}$ for various values of λ in the complex plane. Note that $e^{\lambda t}$ is oscillatory when λ has an imaginary component. It grows unbounded when $\text{Re}\{\lambda\} > 0$, decays to zero when $\text{Re}\{\lambda\} < 0$, and has constant amplitude when $\text{Re}\{\lambda\} = 0$.

Stability in Continuous Time

How do the eigenvalues govern system dynamics?

If initial state is $\vec{x}(0)$, and there's no control input, state at time t is

$$\vec{x}(t) = e^{At}\vec{x}(0)$$

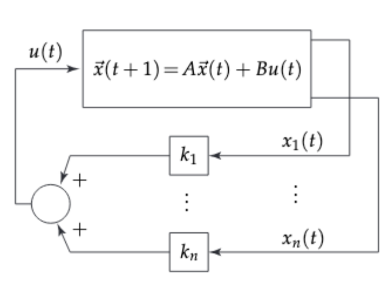
If any eigenvalue of A is larger in magnitude than 1, it "blows up" through repeated exponentiation — the system destabilizes!

Stability Through State Feedback

- If we add a feedback path (modifying the input values with the state) our state update equation changes

$$\vec{x}(t+1) = (A + BK)\vec{x}(t)$$

- What determines the stability of this new system?



State Feedback

- By designing K, we can give our system specific dynamic properties
  - Can analyze and design the way its state changes over time
- If our "open-loop" system is unstable, choosing the right values of K can make it stable!
- Is this always possible?

Example: Controllability and Stability

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

$$\vec{y}[t] = \begin{bmatrix} 1 & 1 \end{bmatrix} \vec{x}[t]$$

Controllable?

Stable for $u[t] = 0$?

Example: Controllability and Stability

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

$$\vec{y}[t] = \begin{bmatrix} 1 & 1 \end{bmatrix} \vec{x}[t]$$

Controllable? **Yes**

Stable for $u[t] = 0$?

Example: Controllability and Stability

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

$$\vec{y}[t] = \begin{bmatrix} 1 & 1 \end{bmatrix} \vec{x}[t]$$

Controllable? **Yes**

Stable for $u[t] = 0$? **No**

Upper Triangularization

Upper Triangularization

- Recall that not all square matrices are diagonalizable
  - An $n \times n$ matrix is diagonalizable iff has n linearly independent eigenvectors
- However, all square matrices can be brought into upper triangular form
- I'll walk through the proof from the notes
- (But I'm not sure how useful this will be / how they would ask questions about this on the test)
- (So if people want to I can instead start taking questions on SVD, time- and frequency-domain analysis of RLC circuits, and phasors)

Upper Triangularization Proof

- What are we trying to prove?
  - Remember that if M is diagonalizable, this means that there exists a matrix P such that PMP^{-1} was diagonal
  - In our case, we want to prove that for any square matrix A, there exists a matrix T such that TAT^{-1} is upper triangular
- We will proceed by induction
- First prove a base case (a 1×1 matrix must be upper triangular)
- Prove that if there exists such a matrix T_0 for a $k \times k$ matrix, then there exists the matrix T for a size $(k+1) \times (k+1)$ matrix

Upper Triangularization Proof

- Clearly a 1×1 matrix is upper triangular
- First we choose one arbitrary eigenvalue / eigenvector pair, choose an orthonormal basis for $\mathbb{R}^n$ (with Gram-Schmidt), then define V formed with those vectors.

We can upper triangularize $(k+1) \times (k+1)$ matrices if we assume that $k \times k$ matrices can be upper triangularized. To show this, let A be an arbitrary $(k+1) \times (k+1)$ matrix and let $\lambda, \vec{v}$ by an eigenvalue/vector pair: $A\vec{v} = \lambda\vec{v}$. Normalize $\vec{v}$ so that $\|\vec{v}\| = 1$ and choose k other vectors $\vec{v}_1, \ldots, \vec{v}_k \in \mathbb{R}^{k+1}$ such that $\{\vec{v}, \vec{v}_1, \ldots, \vec{v}_k\}$ is an orthonormal basis for $\mathbb{R}^{k+1}$. Then the $(k+1) \times (k+1)$ matrix $V = \begin{bmatrix} \vec{v} & \vec{v}_1 & \cdots & \vec{v}_k \end{bmatrix}$ is orthogonal, *i.e.* $V^{-1} = V^T$.

Discretization

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 $x = x' - \frac{\vec{\beta}^T \vec{u}}{\alpha}$.

$$\begin{aligned}
\frac{d}{dt} x(t) &= \alpha(x' - \frac{\vec{\beta}^T \vec{u}}{\alpha}) + \vec{\beta}^T \vec{u}(t) \\
&= \alpha x' \\
x' &= Ae^{\alpha(x - nT)} \\
x + \frac{\vec{\beta}^T \vec{u}}{\alpha} &= Ae^{\alpha(x - nT)} \\
x &= Ae^{\alpha(x - nT)} - \frac{\vec{\beta}^T \vec{u}}{\alpha}
\end{aligned}$$

Discretization: Q1 Sol Continued

At which point we can use our initial condition to get

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

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.

Linearization

Linearization

- Recall that if we have $\frac{dx}{dt} = \lambda x(t) + bu(t)$ we know a solution to this is:

$$x(t) = x(0)e^{\lambda t} + \int_0^t e^{\lambda(t-\tau)}u(\tau)\ d\tau$$

- What if we had $\frac{dx}{dt} = f(x(t)) + bu(t)$, where f is nonlinear (e.g *sin*)?

Linearization

- Recall that if we have $\frac{dx}{dt} = \lambda x(t) + bu(t)$ we know a solution to this is:

$$x(t) = x(0)e^{\lambda t} + \int_0^t e^{\lambda(t-\tau)} u(\tau)\, d\tau$$

- What if we had $\frac{dx}{dt} = f(x(t)) + bu(t)$, where f is nonlinear (e.g *sin*)?
- Big Picture: linearize f around an operating point and then treat it as a linear function in a small neighborhood of that point.
- Why linearization?

Linearization

- Recall that if we have $\frac{dx}{dt} = \lambda x(t) + bu(t)$ we know a solution to this is:

$$x(t) = x(0)e^{\lambda t} + \int_0^t e^{\lambda(t-\tau)}u(\tau)\ d\tau$$

- What if we had $\frac{dx}{dt} = f(x(t)) + bu(t)$, where f is nonlinear (e.g *sin*)?
- Big Picture: linearize f around an operating point and then treat it as a linear function in a small neighborhood of that point.
- Why linearization?
  It allows you to treat the system as a linear one, which is very helpful as linear ODE are (usually) much easier to solve!

Linearizing a Single-Variable Function

- Suppose we have $f(x)$ that is a non linear function. We can use a first order Taylor polynomial to linearize the function, this is equivalent to finding the slope of the tangent line of $f(x)$ at a particular point.
- From calculus: $f(x) \approx f(x^*) + f'(x^*)(x - x^*)$.
- As long as we are within some (very small) δ neighborhood of x^* the linearization is valid.
- Example: Linearize $f(x) = 3e^{x^2+2}$ around x^*

Linearizing a Single-Variable Function

- Suppose we have $f(x)$ that is a non linear function. We can use a first order Taylor polynomial to linearize the function, this is equivalent to finding the slope of the tangent line of $f(x)$ at a particular point.
- From calculus: $f(x) \approx f(x^*) + f'(x^*)(x - x^*)$.
- As long as we are within some (very small) δ neighborhood of x^* the linearization is valid.
- Example: Linearize $f(x) = 3e^{x^2+2}$ around x^*
- Solution:
  $f(x^*) = 3e^{(x^*)^2+2}$
  $f'(x) = 3e^{x^2+2}(2x) = 6xe^{x^2+2}$
  $f'(x^*) = 6x^*e^{(x^*)^2+2}$
  Therefore : $f(x) \approx 3e^{(x^*)^2+2} + 6x^*e^{(x^*)^2+2}(x - x^*)$

Linearizing Steps for $\frac{dx(t)}{dt} = f(x(t)) + bu(t)$

(i) Choose, or you may be given, a DC input point. That is, a point $u^* \equiv u(t)$ that is constant with time.

Linearizing Steps for $\frac{dx(t)}{dt} = f(x(t)) + bu(t)$

(i) Choose, or you may be given, a DC input point. That is, a point $u^* \equiv u(t)$ that is constant with time.

(ii) Find a DC operating point, $x^* \equiv x(t)$. That is, solve $\frac{dx^*}{dt} = f(x^*) + bu^*$. Notice that this boils down to finding an x^* such that $f(x^*) + bu^* = 0$.

Linearizing Steps for $\frac{dx(t)}{dt} = f(x(t)) + bu(t)$

(i) Choose, or you may be given, a DC input point. That is, a point $u^* \equiv u(t)$ that is constant with time.

(ii) Find a DC operating point, $x^* \equiv x(t)$. That is, solve $\frac{dx^*}{dt} = f(x^*) + bu^*$. Notice that this boils down to finding an x^* such that $f(x^*) + bu^* = 0$.

(iii) Define $x_l(t) = x(t) - x^*$ and $u_l(t) = u(t) - u^*$, and re-write the ODE in terms of $x_l(t)$ and $u_l(t)$. By plugging in you get: $\frac{dx_l(t)}{dt} = f(x_l(t) + x^*) + b(u_l(t) + u^*)$

Linearizing Steps for $\frac{dx(t)}{dt} = f(x(t)) + bu(t)$

(i) Choose, or you may be given, a DC input point. That is, a point $u^* \equiv u(t)$ that is constant with time.

(ii) Find a DC operating point, $x^* \equiv x(t)$. That is, solve $\frac{dx^*}{dt} = f(x^*) + bu^*$. Notice that this boils down to finding an x^* such that $f(x^*) + bu^* = 0$.

(iii) Define $x_l(t) = x(t) - x^*$ and $u_l(t) = u(t) - u^*$, and re-write the ODE in terms of $x_l(t)$ and $u_l(t)$. By plugging in you get: $\frac{dx_l(t)}{dt} = f(x_l(t) + x^*) + b(u_l(t) + u^*)$

(iv) It is ok to assume at this point that $u_l(t)$ is small, that means that the $u(t)$ in step 1 does not deviate too much from u^*.

Linearizing Steps for $\frac{dx(t)}{dt} = f(x(t)) + bu(t)$

(i) Choose, or you may be given, a DC input point. That is, a point $u^* \equiv u(t)$ that is constant with time.

(ii) Find a DC operating point, $x^* \equiv x(t)$. That is, solve $\frac{dx^*}{dt} = f(x^*) + bu^*$. Notice that this boils down to finding an x^* such that $f(x^*) + bu^* = 0$.

(iii) Define $x_l(t) = x(t) - x^*$ and $u_l(t) = u(t) - u^*$, and re-write the ODE in terms of $x_l(t)$ and $u_l(t)$. By plugging in you get: $\frac{dx_l(t)}{dt} = f(x_l(t) + x^*) + b(u_l(t) + u^*)$

(iv) It is ok to assume at this point that $u_l(t)$ is small, that means that the $u(t)$ in step 1 does not deviate too much from u^*.

(v) Linearize the ODE: $f(x_l(t) + x^*) \approx f(x^*) + f'(x^*)x_l(t)$. Here we assume that $x_l(t)$ is also small. This is something that we will need to verify in the next step!

Linearizing Steps for $\frac{dx(t)}{dt} = f(x(t)) + bu(t)$ (Continued)

(vi) Plug (vi) back into (iii) and we obtain :
$\frac{dx_l(t)}{dt} \approx f'(x^*)f(x_l(t)) + \cancel{f(x^*)} + bu_l(t) + \cancel{bu^*} = f'(x^*)f(x_l(t)) + bu_l(t)$

(vii) Verify the linearization!

How do we know if the linearization is valid?

Linearizing Steps for $\frac{dx(t)}{dt} = f(x(t)) + bu(t)$ (Continued)

(vi) Plug (vi) back into (iii) and we obtain :
$\frac{dx_l(t)}{dt} \approx f'(x^*)f(x_l(t)) + \cancel{f(x^*)} + bu_l(t) + \cancel{bu^*} =$
$f'(x^*)f(x_l(t)) + bu_l(t)$

(vii) Verify the linearization!

How do we know if the linearization is valid? Well, if we have $\frac{dx_l(t)}{dt} = \lambda x_l(t) + bu(t)$ we know the solution doesn't blow up if $\lambda < 0$ as we will have a term $e^{\lambda t}$.

This means that we want $m = f'(x^*) < 0$.

Linearizing Steps for $\frac{dx(t)}{dt} = f(x(t)) + bu(t)$ (Continued)

(vi) Plug (vi) back into (iii) and we obtain :
$\frac{dx_l(t)}{dt} \approx f'(x^*)f(x_l(t)) + \cancel{f(x^*)} + bu_l(t) + \cancel{bu^*} =$
$f'(x^*)f(x_l(t)) + bu_l(t)$

(vii) Verify the linearization!

How do we know if the linearization is valid? Well, if we have $\frac{dx_l(t)}{dt} = \lambda x_l(t) + bu(t)$ we know the solution doesn't blow up if $\lambda < 0$ as we will have a term $e^{\lambda t}$.

This means that we want $m = f'(x^*) < 0$.

So what do we do if $m > 0$?

Linearizing Steps for $\frac{dx(t)}{dt} = f(x(t)) + bu(t)$ (Continued)

(vi) Plug (vi) back into (iii) and we obtain :
$\frac{dx_l(t)}{dt} \approx f'(x^*)f(x_l(t)) + \cancel{f(x^*)} + bu_l(t) + \cancel{bu^*} =$
$f'(x^*)f(x_l(t)) + bu_l(t)$

(vii) Verify the linearization!
How do we know if the linearization is valid? Well, if we have $\frac{dx_l(t)}{dt} = \lambda x_l(t) + bu(t)$ we know the solution doesn't blow up if $\lambda < 0$ as we will have a term $e^{\lambda t}$.
This means that we want $m = f'(x^*) < 0$.

So what do we do if $m > 0$?

We need to go back and change our DC operating point x^*

Practice Problem

Linearize $\frac{dx(t)}{dt} = \cos(x(t)) + bu(t)$ about $u^* = 0$.

Practice Problem

Linearize $\frac{dx(t)}{dt} = \cos(x(t)) + bu(t)$ about $u^* = 0$.
Hint: $\cos(x^*) = 0$ has multiple solutions, which means that we can find numerous DC operating points, can you guess which one would result in a stable system ?

Practice Problem Solution

(i) We were given the DC input, $u^* = 0$

Practice Problem Solution

(i) We were given the DC input, $u^* = 0$

(ii) $\cos(x^*) = 0$, which means that $x^* = k\frac{\pi}{2}$ for $k \in \{... - 2, -1, 1, 2, ...\}$. We will choose $x^* = \frac{\pi}{2}$

Practice Problem Solution

(i) We were given the DC input, $u^* = 0$

(ii) $\cos(x^*) = 0$, which means that $x^* = k\frac{\pi}{2}$ for $k \in \{... - 2, -1, 1, 2, ...\}$. We will choose $x^* = \frac{\pi}{2}$

(iii) Let $x_l(t) = x(t) - \frac{\pi}{2}$ and $u_l(t) = u(t) - 0$. By plugging in we get: $\frac{dx_l(t)}{dt} = \cos(x_l(t) + \frac{\pi}{2}) + u_l(t)$

Practice Problem Solution

(i) We were given the DC input, $u^* = 0$

(ii) $\cos(x^*) = 0$, which means that $x^* = k\frac{\pi}{2}$ for $k \in \{... - 2, -1, 1, 2, ...\}$. We will choose $x^* = \frac{\pi}{2}$

(iii) Let $x_l(t) = x(t) - \frac{\pi}{2}$ and $u_l(t) = u(t) - 0$. By plugging in we get: $\frac{dx_l(t)}{dt} = \cos(x_l(t) + \frac{\pi}{2}) + u_l(t)$

(iv) We assume that $u_l(t)$ is small.

Practice Problem Solution

(i) We were given the DC input, $u^* = 0$

(ii) $\cos(x^*) = 0$, which means that $x^* = k\frac{\pi}{2}$ for $k \in \{... - 2, -1, 1, 2, ...\}$. We will choose $x^* = \frac{\pi}{2}$

(iii) Let $x_l(t) = x(t) - \frac{\pi}{2}$ and $u_l(t) = u(t) - 0$. By plugging in we get: $\frac{dx_l(t)}{dt} = \cos(x_l(t) + \frac{\pi}{2}) + u_l(t)$

(iv) We assume that $u_l(t)$ is small.

(v) Linearize the ODE: $\cos(x_l(t) + \frac{\pi}{2}) \approx \cos(\frac{\pi}{2}) - \sin(\frac{\pi}{2})x_l(t)$.

Practice Problem Solution

(i) We were given the DC input, $u^* = 0$

(ii) $\cos(x^*) = 0$, which means that $x^* = k\frac{\pi}{2}$ for $k \in \{... - 2, -1, 1, 2, ...\}$. We will choose $x^* = \frac{\pi}{2}$

(iii) Let $x_l(t) = x(t) - \frac{\pi}{2}$ and $u_l(t) = u(t) - 0$. By plugging in we get: $\frac{dx_l(t)}{dt} = \cos(x_l(t) + \frac{\pi}{2}) + u_l(t)$

(iv) We assume that $u_l(t)$ is small.

(v) Linearize the ODE: $\cos(x_l(t) + \frac{\pi}{2}) \approx \cos(\frac{\pi}{2}) - \sin(\frac{\pi}{2})x_l(t)$.

(vi) Plug (v) back into ODE: $\frac{dx_l(t)}{dt} = -x_l(t) + u_l(t)$

Practice Problem Solution

(i) We were given the DC input, $u^* = 0$

(ii) $\cos(x^*) = 0$, which means that $x^* = k\frac{\pi}{2}$ for $k \in \{... -2, -1, 1, 2, ...\}$. We will choose $x^* = \frac{\pi}{2}$

(iii) Let $x_l(t) = x(t) - \frac{\pi}{2}$ and $u_l(t) = u(t) - 0$. By plugging in we get: $\frac{dx_l(t)}{dt} = \cos(x_l(t) + \frac{\pi}{2}) + u_l(t)$

(iv) We assume that $u_l(t)$ is small.

(v) Linearize the ODE: $\cos(x_l(t) + \frac{\pi}{2}) \approx \cos(\frac{\pi}{2}) - \sin(\frac{\pi}{2})x_l(t)$.

(vi) Plug (v) back into ODE: $\frac{dx_l(t)}{dt} = -x_l(t) + u_l(t)$

We see that our assumption that $x_l(t)$ is small is indeed satisfied as we will have a e^{-t} term in the solution which means that $x_l(t)$ will decay.

Practice Problem Solution

(i) We were given the DC input, $u^* = 0$

(ii) $\cos(x^*) = 0$, which means that $x^* = k\frac{\pi}{2}$ for $k \in \{... - 2, -1, 1, 2, ...\}$. We will choose $x^* = \frac{\pi}{2}$

(iii) Let $x_l(t) = x(t) - \frac{\pi}{2}$ and $u_l(t) = u(t) - 0$. By plugging in we get: $\frac{dx_l(t)}{dt} = \cos(x_l(t) + \frac{\pi}{2}) + u_l(t)$

(iv) We assume that $u_l(t)$ is small.

(v) Linearize the ODE: $\cos(x_l(t) + \frac{\pi}{2}) \approx \cos(\frac{\pi}{2}) - \sin(\frac{\pi}{2})x_l(t)$.

(vi) Plug (v) back into ODE: $\frac{dx_l(t)}{dt} = -x_l(t) + u_l(t)$

We see that our assumption that $x_l(t)$ is small is indeed satisfied as we will have a e^{-t} term in the solution which means that $x_l(t)$ will decay.

What if we had chosen a different DC Operating point, say $-\frac{\pi}{2}$? When we linearize the system we see that the solution will "explode" around that particular DC operating point.

Linearization of Vector Functions

What if we had $\frac{d\vec{x}}{dt} = \vec{f}(\vec{x}, \vec{u})$? We will see that the process is very similar to the scalar case we just analyzed!

Linearization of Vector Functions

What if we had $\frac{d\vec{x}}{dt} = \vec{f}(\vec{x}, \vec{u})$? We will see that the process is very similar to the scalar case we just analyzed!
First, let's see how to linearize $\vec{f}(\vec{x})$ around a DC operating point $\vec{x}^*$. Where $\vec{f} \in \mathbb{R}^{n \times 1}$ is a vector of scalar functions.

Linearization of Vector Functions

What if we had $\frac{d\vec{x}}{dt} = \vec{f}(\vec{x}, \vec{u})$? We will see that the process is very similar to the scalar case we just analyzed!
First, let's see how to linearize $\vec{f}(\vec{x})$ around a DC operating point $\vec{x}^*$. Where $\vec{f} \in \mathbb{R}^{n\times 1}$ is a vector of scalar functions.
The idea is to linearize individually each one of the f_i around the DC operating point.

Linearization of Vector Functions

What if we had $\frac{d\vec{x}}{dt} = \vec{f}(\vec{x}, \vec{u})$? We will see that the process is very similar to the scalar case we just analyzed!

First, let's see how to linearize $\vec{f}(\vec{x})$ around a DC operating point $\vec{x}^*$. Where $\vec{f} \in \mathbb{R}^{n\times 1}$ is a vector of scalar functions.

The idea is to linearize individually each one of the f_i around the DC operating point.

For example:

$f_1 \approx f_1(\vec{x}^*) + \frac{\partial f_1}{\partial x_1}(\vec{x}^*)(x_1 - x_1^*) + ... + \frac{\partial f_n}{\partial x_1}(\vec{x}^*)(x_n - x_n^*)$

Linearization of Vector Functions

What if we had $\frac{d\vec{x}}{dt} = \vec{f}(\vec{x}, \vec{u})$? We will see that the process is very similar to the scalar case we just analyzed!

First, let's see how to linearize $\vec{f}(\vec{x})$ around a DC operating point $\vec{x}^*$. Where $\vec{f} \in \mathbb{R}^{n\times 1}$ is a vector of scalar functions.

The idea is to linearize individually each one of the f_i around the DC operating point.

For example:

$f_1 \approx f_1(\vec{x}^*) + \frac{\partial f_1}{\partial x_1}(\vec{x}^*)(x_1 - x_1^*) + ... + \frac{\partial f_n}{\partial x_1}(\vec{x}^*)(x_n - x_n^*)$

Repeating this for all n functions in $\vec{f}$ we see we get a system of scalar, linearized, multivariate functions which makes you think, wouldn't it be nice to express it in a shorthand matrix notation?

Jacobian Matrix

We can use the Jacobian to express everything nicely and neatly. The Jacobian is the name given to the matrix of partial derivatives of $\vec{f}$, and it is denoted by $J_{\vec{x}}$ or $\nabla_{\vec{x}}\vec{f}$.

Jacobian Matrix

We can use the Jacobian to express everything nicely and neatly.
The Jacobian is the name given to the matrix of partial derivatives of $\vec{f}$, and it is denoted by $J_{\vec{x}}$ or $\nabla_{\vec{x}}\vec{f}$.
Continuing from the previous slide we have:

$$\begin{bmatrix} f_1(\vec{x}) \\ \vdots \\ f_n(\vec{x}) \end{bmatrix} \approx \begin{bmatrix} f_1(\vec{x}^*) \\ \vdots \\ f_n(\vec{x}^*) \end{bmatrix} + \begin{bmatrix} \frac{\partial f_1}{\partial x_1}(\vec{x}^*) & \dots & \frac{\partial f_1}{\partial x_n}(\vec{x}^*) \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1}(\vec{x}^*) & \dots & \frac{\partial f_n}{\partial x_n}(\vec{x}^*) \end{bmatrix} (\vec{x} - \vec{x}^*)$$

Linearization with Jacobians Example

Linearize $\vec{f}(\vec{x}(t)) = \begin{bmatrix} \sin(x_1(t) \times x_2(t)) + 2x_1(t)x_3^2(t) \\ x_3(t)\cos(x_2(t)) + \frac{x_1(t)}{x_3(t)} \\ x_1(t) + 2x_3(t)x_2^3(t) \end{bmatrix}$ about $\vec{x}^* = \begin{bmatrix} 0 \\ 2\pi \\ \frac{2\pi}{3} \end{bmatrix}$

Solutions

Find the Jacobian:

$$\begin{bmatrix} x_2(t)\cos(x_1(t)\times x_2(t)) + 2x_3^2(t) & x_1(t)\cos(x_1(t)\times x_2(t)) & 4x_1(t)x_3(t) \\ \frac{1}{x_3(t)} & -x_3(t)\sin(x_2(t)) & \cos(x_2(t)) - \frac{x_1(t)}{x_3^2(t)} \\ 1 & 6x_3(t)x_2^2(t) & 2x_2^3(t) \end{bmatrix}$$

Evaluate the Jacobian about $\vec{x}^*$:

$$\begin{bmatrix} 5\pi & 0 & 0 \\ \frac{2\pi}{3} & 0 & 1 \\ 1 & 36\pi^3 & 16\pi^3 \end{bmatrix}$$

Linearize:

$$\vec{f}(\vec{x}(t)) \approx \begin{bmatrix} 0 \\ \frac{3\pi}{4} \\ 24\pi^4 \end{bmatrix} + \begin{bmatrix} 5\pi & 0 & 0 \\ \frac{2\pi}{3} & 0 & 1 \\ 1 & 36\pi^3 & 16\pi^3 \end{bmatrix} \begin{bmatrix} x_1(t) - 0 \\ x_2(t) - \frac{3\pi}{4} \\ x_3(t) - 24\pi^4 \end{bmatrix}$$

Steps to Linearize Vector ODE Systems

To linearize $\frac{d\vec{x}}{dt} = \vec{f}(\vec{x}(t), \vec{u}(t))$ we use a similar procedure as we did for the scalar case.

Steps to Linearize Vector ODE Systems

To linearize $\frac{d\vec{x}}{dt} = \vec{f}(\vec{x}(t), \vec{u}(t))$ we use a similar procedure as we did for the scalar case.

(i) Solve $\vec{f}(\vec{x}^*, \vec{u}^*) = \vec{0}$ to determine the equilibrium point.

Steps to Linearize Vector ODE Systems

To linearize $\frac{d\vec{x}}{dt} = \vec{f}(\vec{x}(t), \vec{u}(t))$ we use a similar procedure as we did for the scalar case.

(i) Solve $\vec{f}(\vec{x}^*, \vec{u}^*) = \vec{0}$ to determine the equilibrium point.

(ii) Let $\vec{x}_l(t) = \vec{x}(t) - \vec{x}^*$ and $\vec{u}_l(t) = \vec{u}(t) - \vec{u}^*$

Steps to Linearize Vector ODE Systems

To linearize $\frac{d\vec{x}}{dt} = \vec{f}(\vec{x}(t), \vec{u}(t))$ we use a similar procedure as we did for the scalar case.

(i) Solve $\vec{f}(\vec{x}^*, \vec{u}^*) = \vec{0}$ to determine the equilibrium point.

(ii) Let $\vec{x}_l(t) = \vec{x}(t) - \vec{x}^*$ and $\vec{u}_l(t) = \vec{u}(t) - \vec{u}^*$

(iii) Linearize $\vec{f}(\vec{x}, \vec{u})$ about $(\vec{x}^*, \vec{u}^*)$. That is:
$\vec{f}(\vec{x}(t), \vec{u}(t)) \approx \vec{f}(\vec{x}^*, \vec{u}^*) + J_{\vec{x}}\vec{x}_l(t) + J_{\vec{u}}\vec{u}_l(t)$

Steps to Linearize Vector ODE Systems

To linearize $\frac{d\vec{x}}{dt} = \vec{f}(\vec{x}(t), \vec{u}(t))$ we use a similar procedure as we did for the scalar case.

(i) Solve $\vec{f}(\vec{x}^*, \vec{u}^*) = \vec{0}$ to determine the equilibrium point.

(ii) Let $\vec{x}_l(t) = \vec{x}(t) - \vec{x}^*$ and $\vec{u}_l(t) = \vec{u}(t) - \vec{u}^*$

(iii) Linearize $\vec{f}(\vec{x}, \vec{u})$ about $(\vec{x}^*, \vec{u}^*)$. That is:
$\vec{f}(\vec{x}(t), \vec{u}(t)) \approx \vec{f}(\vec{x}^*, \vec{u}^*) + J_{\vec{x}}\vec{x}_l(t) + J_{\vec{u}}\vec{u}_l(t)$

(iv) Plug (iv) back into the ODE:
$\frac{d\vec{x}}{dt} \approx \cancel{\vec{f}(\vec{x}^*, \vec{u}^*)} + J_{\vec{x}}\vec{x}_l(t) + J_{\vec{u}}\vec{u}_l(t)$

Linearizing Vector ODE Systems Example

Given a DC input $u^* = 1$, linearize:

$$\frac{d\vec{x}(t)}{dt} = \begin{bmatrix} x_1^2(t) - x_2(t)u(t) \\ x_2^2(t)(1 + x_1(t)) + \sin(\pi x_1(t)u(t)) \end{bmatrix}$$

Solutions

Again, we will do this in steps:

(i) We are given $u^* = 1$

Solutions

Again, we will do this in steps:

(i) We are given $u^* = 1$

(ii) We need to find a DC operating point, this means solving the following system of equations:

$$x_1^{*2} - x_2^* u^* = 0 \tag{1}$$

$$x_2^{*2}(x_1^* + 1) + \sin(\pi x_1^* u^*) = 0 \tag{2}$$

The solution is $x_1^* = -1$ and $x_2^* = 1$.

Solutions

Again, we will do this in steps:

(i) We are given $u^* = 1$

(ii) We need to find a DC operating point, this means solving the following system of equations:

$$x_1^{*2} - x_2^* u^* = 0 \tag{1}$$

$$x_2^{*2}(x_1^* + 1) + \sin(\pi x_1^* u^*) = 0 \tag{2}$$

The solution is $x_1^* = -1$ and $x_2^* = 1$.

(iii) Let $\vec{x}_l(t) = \vec{x}(t) - \vec{x}^*$ and $\vec{u}_l(t) = \vec{u}(t) - \vec{u}^*$

Solutions

Again, we will do this in steps:

(i) We are given $u^* = 1$

(ii) We need to find a DC operating point, this means solving the following system of equations:

$$x_1^{*2} - x_2^* u^* = 0 \tag{1}$$

$$x_2^{*2}(x_1^* + 1) + \sin(\pi x_1^* u^*) = 0 \tag{2}$$

The solution is $x_1^* = -1$ and $x_2^* = 1$.

(iii) Let $\vec{x}_l(t) = \vec{x}(t) - \vec{x}^*$ and $\vec{u}_l(t) = \vec{u}(t) - \vec{u}^*$

(iv) Linearize,

$$\vec{f}(\vec{x}(t), u(t)) \approx \vec{f}(\vec{x}^*, 1) + \begin{bmatrix} -2 & -1 \\ 1 - \pi & 0 \end{bmatrix} \vec{x}_l(t) + \begin{bmatrix} -1 \\ \pi \end{bmatrix} u_l(t)$$

Solutions Continued

(v) Substitute linear approximation back into the system,

$$\frac{d\vec{x}(t)}{dt} \approx \cancel{\vec{f}(\vec{x}^*, 1)} + \begin{bmatrix} -2 & -1 \\ 1-\pi & 0 \end{bmatrix} \vec{x}_l(t) + \begin{bmatrix} -1 \\ \pi \end{bmatrix} u_l(t)$$

Singular Value Decomposition

SVD Theorem

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

$$
\begin{aligned}
A &= U \Sigma V^T \\
U &: m \times m \\
\Sigma &: m \times n \\
V^T &: n \times n
\end{aligned}
$$

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 $A^T A$

$$A^T A = U\Sigma V^T V\Sigma^T U^T$$
$$= U\Sigma^2 U^T$$

This is an eigen decomposition since Σ^2 is diagonal and $U^{-1} = U^T$. 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

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

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 $A\vec{x}$?

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.

Principle Component Analysis

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 — Computation

1. Store data row-major in $A \in \mathbb{R}^{n \times d}$

PCA — Computation

1. Store data row-major in $A \in \mathbb{R}^{n \times d}$
2. De-mean A

PCA — Computation

1. Store data row-major in $A \in \mathbb{R}^{n \times d}$
2. De-mean A
3. Take SVD: $A = U\Sigma V^T$

PCA — Computation

1. Store data row-major in $A \in \mathbb{R}^{n \times d}$
2. De-mean A
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

PCA — Computation

1. Store data row-major in $A \in \mathbb{R}^{n \times d}$
2. De-mean A
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

$$A' = BV_k^T$$