Course Outline - 1st Half

- Review Classical Feedback Control
- Review Vector/Matrix Theory
- State-Space Representations
- LTI Response, Matrix Exponential
- Transfer Functions & Eigenvalues (2)
- Frequency-Domain Analysis
- Harmonic & Impulse Responses
- Pole Placement
- Controllability

Dynamic System Models

Transfer Function Matrices

Eigenvalues of the State Matrix

Matlab Representation

Introduction

Last meeting we discussed the a variety of approaches to solving the state equations.

The Laplace transform approach gave us a relationship between the state matrix $\mathbf{A}$ and the state transition matrix $e^{\mathbf{A}t}$,

$$L^{-1}\left\{\left[s\mathbf{I}-\mathbf{A}\right]^{-1}\right\}=e^{\mathbf{A}t}$$

Aside from giving us a method to compute $e^{\mathbf{A}t}$, this relationship gives us the ability to relate the <u>properties of $\mathbf{A}$</u> to the <u>dynamics</u> of the system.

Transfer Functions - I

What about the output equations?

$$\mathbf{y}(t) = \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{u}(t)$$

Notice that the output equations are only a function of the states and the input, therefore, if we know the states and the input as functions of time we can always find the outputs from the expression for $\mathbf{y}(t)$.

Transfer Functions - II

Combining the Laplace representation of the state dynamics with the output equations will yield the input-output transfer functions of the system.

Remember that we discussed single-input-single-output (SISO) transfer functions in L1

$$\frac{y(s)}{u(s)} = H(s) = \left(\frac{b_m s^m + b_{m-1} s^{m-1} + \cdots + b_1 s + b_0}{a_n s^n + a_{n-1} s^{n-1} + \cdots + a_1 s + a_0} \right) \quad n \geq m$$

Transfer Functions - III

In this course we will deal with multi-input-multi-output (MIMO) systems.

The state-space formulation naturally leads to systems described by <u>transfer function matrices</u>, where each entry of the matrix is a <u>SISO</u> transfer function

$$\begin{bmatrix} y_1(s) \\ \vdots \\ y_P(s) \end{bmatrix} = \begin{bmatrix} H_{11}(s) & \cdots & H_{1M}(s) \\ \vdots & \ddots & \vdots \\ H_{P1}(s) & \cdots & H_{PM}(s) \end{bmatrix} \begin{bmatrix} u_1(s) \\ \vdots \\ u_M(s) \end{bmatrix}$$

Transfer Functions - IV

How do we find the matrix $\mathbf{H}(s)$?

Start by taking the Laplace transform of the state and output equations:

$$s\mathbf{x}(s)-\mathbf{x}(0)=\mathbf{A}\mathbf{x}(s)+\mathbf{B}\mathbf{u}(s)$$

$$\mathbf{y}(s)=\mathbf{C}\mathbf{x}(s)+\mathbf{D}\mathbf{u}(s)$$

Next, solve for $\mathbf{x}(s)$ in terms of $\mathbf{u}(s)$ assuming zero initial conditions

Transfer Functions - V

Substitute the expression for $\mathbf{x}(s)$ into the Laplace transformed output equation

$$\mathbf{y}(s)=\mathbf{C}\left[\left[s\mathbf{I}-\mathbf{A}\right]^{-1}\mathbf{B}\mathbf{u}(s)\right]+\mathbf{D}\mathbf{u}(s)$$

and factor the input term

The bracketed term above is the matrix of transfer functions

Transfer Functions - VI

Using our definition of the matrix inverse

$$[s\mathbf{I}-\mathbf{A}]^{-1} = \frac{adj[s\mathbf{I}-\mathbf{A}]}{|s\mathbf{I}-\mathbf{A}|} = \frac{\text{Adjoint}}{\text{Determinant}}$$

We can rewrite the transfer function matrix as

$$\underset{[P\times M]}{\mathbf{H}(s)} = \underset{[P\times N]}{\mathbf{C}} \left[\frac{\underset{[N\times N]}{adj[s\mathbf{I}-\mathbf{A}]}}{\underset{\substack{N^{th}\text{ order scalar} \\ \text{polynomial in } s}}{|s\mathbf{I}-\mathbf{A}|}} \right] \underset{[N\times M]}{\mathbf{B}} + \underset{[P\times M]}{\mathbf{D}}$$

Transfer Functions - VII

This analysis tells us that every transfer function element in $\mathbf{H}(s)$ has the same denominator.

This denominator is simply the determinant of $[s\mathbf{I} - \mathbf{A}]$.

The poles of the system are the solutions of the equation:

In-Class Assignment

Find the matrix of transfer functions $\mathbf{H}(s)$ for the system:

The image cannot be displayed. Your computer may not have enough memory to open the image, or the image may have been corrupted. Restart your computer, and then open the file again. If the red x still appears, you may have to delete the image and then insert it again.

$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{bmatrix} = \begin{bmatrix} 3 & 1 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} u_1(t) \\ u_2(t) \end{bmatrix}$$

$$\begin{bmatrix} y_1(t) \end{bmatrix} = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + \begin{bmatrix} 0 & 0 \end{bmatrix} \begin{bmatrix} u_1(t) \\ u_2(t) \end{bmatrix}$$

In-Class Assignment

Canonical Forms - I

Let's go back to our general SISO transfer function representation, but w.l.o.g., we can assume $a_n = 1$

$$H(s) = \frac{y(s)}{u(s)} = \left(\frac{b_m s^m + b_{m-1} s^{m-1} + \cdots + b_1 s + b_0}{s^n + a_{n-1} s^{n-1} + \cdots + a_1 s + a_0} \right) \quad n > m$$

We can rewrite this transfer function as

$$H(s) = \frac{y(s)}{z(s)} \frac{z(s)}{u(s)} = \left(\frac{b_m s^m + b_{m-1} s^{m-1} + \cdots + b_1 s + b_0}{s^n + a_{n-1} s^{n-1} + \cdots + a_1 s + a_0} \right) \quad n > m$$

Canonical Forms - II

Now define

$$\frac{z(s)}{u(s)} = \left(\frac{1}{s^n + a_{n-1}s^{n-1} + \cdots + a_1 s + a_0} \right)$$

We can rewrite this expression

$$\left\{ s^n + a_{n-1}s^{n-1} + \cdots + a_1 s + a_0 \right\} z(s) = u(s)$$

and transform back into the time domain

Canonical Forms - III

The following choice of state variables

$$\mathbf{x}(t)=\begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \\ \vdots \\ x_n(t) \end{bmatrix}=\begin{bmatrix} z(t) \\ \dot{z}(t) \\ \ddot{z}(t) \\ \vdots \\ \frac{d^{n-1}z(t)}{dt^{n-1}} \end{bmatrix}$$

Leads to the following state equations

Canonical Forms - IV

Based on the previous definition, we must have the following output

$$y(s) = \left(b_m s^m + b_{m-1} s^{m-1} + \cdots + b_1 s + b_0\right) z(s)$$

Applying the inverse Laplace Transform

$$y(t) = b_m \frac{d^m z(t)}{dt^m} + b_{m-1} \frac{d^{m-1} z(t)}{dt^{m-1}} + \cdots + b_1 \frac{dz(t)}{dt} + b_0 z(t)$$

Substituting the state variable definitions

Canonical Forms - V

Putting all these results into matrix form, the state-space representation is:

$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \\ \vdots \\ \dot{x}_{n-1}(t) \\ \dot{x}_n(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ -a_0 & -a_1 & -a_2 & \cdots & -a_{n-1} \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \\ \vdots \\ x_{n-1}(t) \\ x_n(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} u(t)$$

Denominator Coefficients

$$\begin{bmatrix} y(t) \end{bmatrix} = \begin{bmatrix} b_0 & b_1 & \cdots & b_m & 0 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \\ \vdots \\ x_{n-1}(t) \\ x_n(t) \end{bmatrix} + \begin{bmatrix} 0 \end{bmatrix} u(t)$$

Numerator Coefficients

Canonical Forms - VI

Notice that the last row of the $\mathbf{A}$ matrix contains the denominator coefficients and the $\mathbf{C}$ matrix contains the numerator coefficients

This transfer function representation is called a companion form, and only works when $n > m$

TF2SS and SS2TF Conversion - I

Start with an example transfer function

$$\frac{y(s)}{u(s)} = \left(\frac{6s+4}{s^2+2s+10} \right)$$

First write the state-space representation in companion form

$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -10 & -2 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t)$$

$$y(t) = \begin{bmatrix} 4 & 6 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix}$$

TF2SS and SS2TF Conversion - II

Now lets transform the state-space representation back to a transfer function

We know the answer will be the scalar transfer function:

$$H(s) = \mathbf{C}\left[s\mathbf{I} - \mathbf{A}\right]^{-1}\mathbf{B} + \mathbf{D}$$

TF2SS and SS2TF Conversion - III

Substituting the **A**, **B**, **C**, and **D** matrices into the matrix expression for H

$$H(s) = \begin{bmatrix} 4 & 6 \end{bmatrix} \left[\begin{bmatrix} s & 0 \\ 0 & s \end{bmatrix} - \begin{bmatrix} 0 & 1 \\ -10 & -2 \end{bmatrix} \right]^{-1} \begin{bmatrix} 0 \\ 1 \end{bmatrix} + \begin{bmatrix} 0 \end{bmatrix}$$

$$= \begin{bmatrix} 4 & 6 \end{bmatrix} \begin{bmatrix} s & -1 \\ 10 & s+2 \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

TF2SS and SS2TF Conversion - IV

The inverse matrix is

$$\begin{bmatrix} s & -1 \\ 10 & s+2 \end{bmatrix}^{-1} = \frac{\begin{bmatrix} s+2 & 1 \\ -10 & s \end{bmatrix}}{s(s+2)-(-1)(10)} = \frac{\begin{bmatrix} s+2 & 1 \\ -10 & s \end{bmatrix}}{s^2+2s+10}$$

Substituting this result back into the transfer function we get

$$H(s) = \frac{\begin{bmatrix} 4 & 6 \end{bmatrix}\begin{bmatrix} s+2 & 1 \\ -10 & s \end{bmatrix}\begin{bmatrix} 0 \\ 1 \end{bmatrix}}{s^2+2s+10} = \frac{\begin{bmatrix} 4 & 6 \end{bmatrix}\begin{bmatrix} 1 \\ s \end{bmatrix}}{s^2+2s+10}$$

TF2SS and SS2TF Conversion - V

The final expanded result is

$$H(s) = \left(\frac{6s + 4}{s^2 + 2s + 10} \right)$$

Which is identical to our original transfer function

TF2SS and SS2TF Conversion - VI

How do we perform these transformations in Matlab?

We first have to look at how transfer functions and state-space representations are stored in Matlab.

The most efficient method is to use the object-oriented tools in the Control System Toolbox

TF2SS and SS2TF Conversion - VII

Let's say we have the following continuous time transfer function

$$H(s) = \left(\frac{b_3 s^3 + b_2 s^2 + b_1 s + b_0}{a_4 s^4 + a_3 s^3 + a_2 s^2 + a_1 s + a_0} \right)$$

In Matlab, we first need to store the numerator and denominator coefficients in arrays starting with the highest power of s

```
num = [b3, b2, b1, b0];
den = [a4, a3, a2, a1, a0];
```

TF2SS and SS2TF Conversion - VIII

To store these polynomial coefficients as a transfer function object, use the `tf()` function

```
tf_obj = tf(num, den);
```

To extract the numerator and denominator coefficients from a transfer function object use the `tfdata()` function

```
[num, den] = tfdata(tf_obj,'v');
```

TF2SS and SS2TF Conversion - IX

Now let's say we have the following state-space model

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

$$\mathbf{y}(t) = \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{u}(t)$$

To store these matrices as a state-space object, use the `ss()` function

```
ss_obj = ss(A, B, C, D);
```

TF2SS and SS2TF Conversion - X

To extract the state-space matrices from a state-space object use the `ssdata()` function

```
[A, B, C, D] = ssdata(ss_obj);
```

TF2SS and SS2TF Conversion - XI

Finally, to convert between a transfer function representation and a state-space representation use either:

```
[A, B, C, D] = tf2ss(num,den);
```

or

```
[num, den] = ss2tf(A, B, C, D);
```

TF2SS and SS2TF Conversion - XII

Matlab/Control System Toolbox has a built-in “viewer” to analyze the time and frequency responses of LTI models:

```
ltiview(tf_obj);

ltiview(ss_obj);

ltiview(tf_obj,ss_obj);
```

Eigenvalues of the State Matrix

Recall our discussion of the eigenvalue problem in the first week of the course. The eigenvalues of the $\mathbf{A}$ matrix are the solutions of the equation

Comparing this with the equation from L6/S10 we see that this expression is equivalent to the expression for the poles of the system.

In-Class Assignment

Using the previous ICA example (L6/S11), show that the eigenvalues of $\mathbf{A}$ are equivalent to the poles of the transfer functions.

Eigenvalues of the State Matrix

This result says that the eigenvalues of the state matrix $\mathbf{A}$ are equivalent to the poles of the system.

Remember that the poles tell us a great deal about the system dynamics:

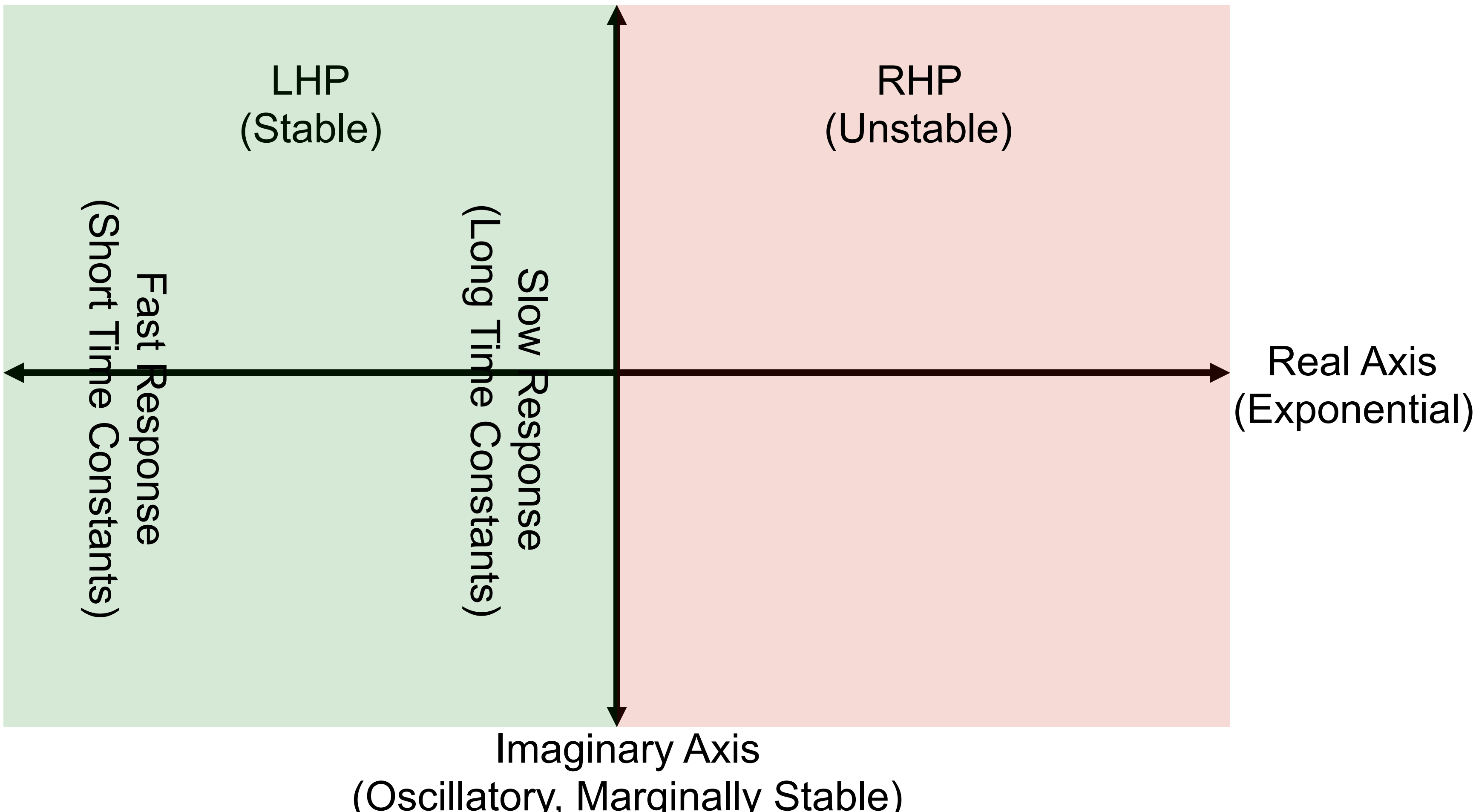
– Will the system be stable or unstable?

– Will the system oscillate?

– How long will it take to reach steady-state?

Eigenvalues of the State Matrix

Remember that the poles are located in the complex s-plane



Eigenvalues of the State Matrix

How hard is it to find the eigenvalues of a matrix? Again our friends at the MathWorks have simplified life for us.

The MATLAB funtion `eig` solves for the eigenvalues of a matrix.

```
lambda = eig(A)
```

Will return a column vector `lambda` of eigenvalues of `A`.

Remember, in general, the eigenvalues of the state matrix will be <u>complex</u>.

Eigenvalues of the State Matrix

The MATLAB funtion `eig` can also be used to solve for the eigenvectors.

```
[V,lambda] = eig(A)
```

Will return a matrix `V` of eigenvectors stored in columns, and a diagonal matrix `lambda` of eigenvalues of `A`.

In general, the eigenvalues and eigenvectors will be <u>complex</u>.

Eigenvalues of the State Matrix

Therefore we now know that:

1. The poles of a system can be determined directly from the eigenvalues of $\mathbf{A}$.
2. Stability is determined by the location of the poles. The response character is also strongly influenced by the location of the system poles.

Summary

- State-space models lead to a matrix of transfer functions. All of the transfer functions will have the same denominator but their numerators will generally differ.
- The eigenvalues of the state matrix A are equivalent to the poles of the system.