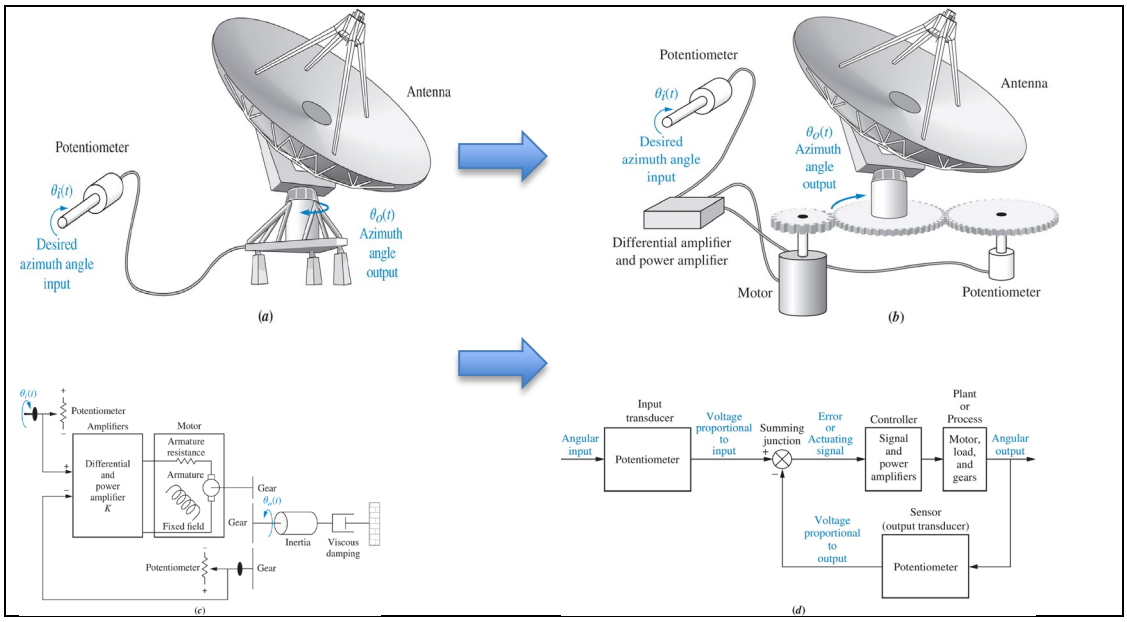
ENME 462: Studio 0

The first objective of this studio is to understand modeling of a physical control system into function block diagrams. Then, MATLAB tools will be introduced to computationally analyze Laplace Transforms, transfer functions, and their corresponding inverse. The use of MATLAB for computing partial fractions will also be studied.

Antenna Azimuth Position Control System

The diagram below illustrates the antenna azimuth position control system.



Constructing Transfer Functions and Partial Fractions using MATLAB (This material can be found in APPENDIX B of NISE, 7th Edition)

Note: The MATLAB files for running the code below has been uploaded on CANVAS. Please utilize the appropriate file where the file name is provided below in ' '. The TA will guide you through the use of each file and the corresponding content.

(ch2p2): Polynomials in s can be represented as row vectors containing the coefficients. Thus $P1 = s^3 + 7s^2 - 3s + 23$ can be represented by the vector shown below with elements separated by a space or comma. Bit strings can be used to identify each section of this tutorial.

```
'(ch2p2)'                % Display label.
P1=[1 7 -3 23]           % Store polynomial s^3 + 7s^2 -3s + 23 as P1 and display.
```

(ch2p3): Running the previous statements causes MATLAB to display the results. Ending

the command with a semicolon suppresses the display. Typing an expression without a left-hand assignment and without a semicolon causes the expression to be evaluated and the result displayed. Enter P2 in the MATLAB Command Window after execution.

```
'(ch2p3)'                % Display label.
P2=[3 5 7 8];            % Assign 3s^3 + 5s^2 +7s + 8 to P2 without displaying.
3*5                      % Evaluate 3*5 and display result.
```

(ch2p4): An F(s) in factored form can be represented in polynomial form. Thus $P3 = (s + 2)(s + 5)(s + 6)$ can be transformed into a polynomial using poly(V), where V is a row vector containing the roots of the polynomial and poly(V) forms the coefficients of the polynomial.

```
'(ch2p4)'                % Display label.
P3=poly([-2 -5 -6])      % Store polynomial (s+2)(s+5)(s+6) as P3 and display the coefficients
```

(ch2p5): We can find roots of polynomials using the roots(V) command. The roots are returned as a column vector. For example, find the roots of $5s^4 + 7s^3 + 9s^2 - 3s + 2 = 0$.

```
'(ch2p5)'                % Display label.
P4=[5 7 9 -3 2]          % Form 5s^4+7s^3+9s^2-3s+2 and display.
rootsP4=roots(P4)        % Find roots of 5s^4+7s^3+9s^2 -3s+2, assign to rootsP4, and display.
```

(ch2p6): Polynomials can be multiplied together using the conv(a,b) command (standing for convolve). Thus, $P5 = (s^3 + 7s^2 + 10s + 9)(s^4 - 3s^3 + 6s^2 + 2s + 1)$ is generated as follows:

```
'(ch2p6)'                % Display label.
P5=conv([1 7 10 9],[1 -3 6 2 1])% Form (s^3+7s^2+10s+9)(s^4-3s^3+6s^2+2s+1), assign to P5, and display.
```

(ch2p7): The partial-fraction expansion for F(s) = b(s)/a(s) can be found using the [K,p,k] = residue(b,a) command (K = residue; p = roots of denominator; k = direct quotient, which is found by dividing polynomials prior to performing a partial-fraction expansion). We expand $F(s) = (7s^2 + 9s + 12)/[s(s + 7)(s^2 + 10s + 100)]$ as an example. Using the results from MATLAB yields:
$F(s) = [(0.2554 - 0.3382i)/(s + 5.0000 - 8.6603i)] + [(0.2554 + 0.3382i)/(s + 5.0000 + 8.6603i)] - [0.5280/(s + 7)] + [0.0171/s]$.

```
'(ch2p7)'                % Display label.
numf=[7 9 12];           % Define numerator of F(s).
denf=conv(poly([0 -7]),[1 10 100]); % Define denominator of F(s).
[K,p,k]=residue(numf,denf)  % Find residues and assign to K; find roots of denominator and assign
to p; find constant and assign to k.
```

(ch2p8) Example 2.3: Let us do Example 2.3 in the book using MATLAB.

```
'(ch2p8) Example 2.3'          % Display label.
numy=32;                       % Define numerator.
deny=poly([0 -4 -8]);          % Define denominator.
[r,p,k] = residue(numy,deny)   % Calculate residues, poles, and direct quotient.
```

(ch2p9): Creating Transfer Functions

(1) Vector Method, Polynomial Form: A transfer function can be expressed as a numerator polynomial divided by a denominator polynomial, i.e. $F(s) = N(s)/D(s)$. The numerator, $N(s)$, is represented by a row vector, numf, that contains the coefficients of $N(s)$. Similarly, the denominator, $D(s)$, is represented by a row vector, denf, that contains the coefficients of D(s). We form $F(s)$ with the command, $F = \mathrm{tf}(\mathrm{numf}, \mathrm{denf})$. F is called a linear time-invariant (LTI) object. This object, or transfer function, can be used as an entity in other operations, such as addition or multiplication. We demonstrate with $F(s) = 150(s^2 + 2s + 7)/[s(s^2 + 5s + 4)]$. Notice after executing the tf command, MATLAB prints the transfer function.

(2) Vector Method, Factored Form: We also can create LTI transfer functions if the numerator and denominator are expressed in factored form. We do this by using row vectors containing the roots of the numerator and denominator. Thus $G(s) = K * N(s)/D(s)$ can be expressed as an LTI object using the command, $G = \mathrm{zpk}(\mathrm{numg}, \mathrm{deng}, K)$, where numg is a row vector containing the roots of N(s) and deng is a row vector containing the roots of D(s). The expression zpk stands for zeros (roots of the numerator), poles (roots of the denominator), and gain, K. We demonstrate with $G(s) = 20(s + 2)(s + 4)/[(s + 7)(s + 8)(s + 9)]$. Notice after executing the zpk command, MATLAB prints the transfer function.

(3) Rational Expression in s Method, Polynomial Form (Requires Control System Toolbox 8.0): This method allows you to type the transfer function as you normally would write it. The statement $s = \mathrm{tf}('s')$ must precede the transfer function if you wish to create an LTI transfer function in polynomial form equivalent to using $G = \mathrm{tf}(\mathrm{numg}, \mathrm{deng})$.

(4) Rational Expression in s Method, Factored Form (Requires Control System Toolbox 8.0): This method allows you to type the transfer function as you normally would write it. The statement $s = \mathrm{zpk}('s')$ must precede the transfer function if you wish to create an LTI transfer function in factored form equivalent to using $G = \mathrm{zpk}(\mathrm{numg}, \mathrm{deng}, K)$.

For both rational expression methods the transfer function can be typed in any form regardless of whether $s = \mathrm{tf}('s')$ or $s = \mathrm{zpk}('s')$ is used. The difference is in the created LTI transfer function. We use the same examples above to demonstrate the rational expression in s methods.

```
'(ch2p9)'                      % Display label.
'Vector Method, Polynomial Form'  % Display label.
```

```
numf=150*[1 2 7]                    % Store 150(s^2+2s+7) in numf and display.
denf=[1 5 4 0]                      % Store s(s+1)(s+4) in denf and display.
'F(s)'                              % Display label.
F=tf(numf,denf)                     % Form F(s) and display.
Clear                               % Clear previous variables from workspace.

'Vector Method, Factored Form'      % Display label.
numg=[-2 -4]                        % Store (s+2)(s+4) in numg and display.
deng=[-7 -8 -9]                     % Store (s+7)(s+8)(s+9) in deng and display.
K=20                                % Define K.
'G(s)'                              % Display label.
G=zpk(numg,deng,K)                  % Form G(s) and display.
Clear                               % Clear previous variables from workspace.

'Rational Expression Method, Polynomial Form'   % Display label.
s=tf('s')                           % Define 's' as an LTI object in polynomial form.
F=150*(s^2+2*s+7)/[s*(s^2+5*s+4)]   % Form F(s) as an LTI transfer function in polynomial form.
G=20*(s+2)*(s+4)/[(s+7)*(s+8)*(s+9)] % Form G(s) as an LTI transfer function in polynomial form.
Clear                               % Clear previous variables from workspace.

'Rational Expression Method, Factored Form'   Display label.
s=zpk('s')                          % Define 's' as an LTI object in factored form.
F=150*(s^2+2*s+7)/[s*(s^2+5*s+4)]   % Form F(s)as an LTI transfer function in factored form.
G=20*(s+2)*(s+4)/[(s+7)*(s+8)*(s+9)] % Form G(s) as an LTI transfer function in factored form.
```

(ch210): Transfer function numerator and denominator vectors can be converted between polynomial form containing the coefficients and factored form containing the roots. The MATLAB function, $\mathrm{tf2zp(numtf, dentf)}$, converts the numerator and denominator from coefficients to roots. The results are in the form of column vectors. We demonstrate this with $F(s) = (10s^2 + 40s + 60)/(s^3 + 4s^2 + 5s + 7)$. The MATLAB function, $\mathrm{zp2tf(numzp, denzp, K)}$, converts the numerator and denominator from roots to coefficients. The arguments numzp and denzp must be column vectors. In the demonstration below apostrophes signify transpose. We demonstrate the conversion from roots to coefficients with $G(s) = 10(s + 2)(s + 4)/[s(s + 3)(s + 5)]$.

```
'(ch2p10)'                          % Display label.
'Coefficients for F(s)'             % Display label.
numftf=[10 40 60]                   % Form numerator of F(s) = (10s^2+40s+60)/(s^3+4s^2+5s+7).
denftf=[1 4 5 7]                    % Form denominator of F(s) = (10s^2+40s+60)/(s^3+4s^2+5s+7).
'Roots for F(s)'                    % Display label.
[numfzp,denfzp]=tf2zp(numftf,denftf)   % Convert F(s) to factored form.
'Roots for G(s)'                    % Display label.
numgzp=[-2 -4]                      % Form numerator of G(s) = 10(s+2)(s+4)/[s(s+3)(s+5)].
K=10
dengzp=[0 -3 -5]                    % Form denominator of G(s) = 10(s+2)(s+4)/[s(s+3)(s+5)].
'Coefficients for G(s)'             % Display label.
[numgtf,dengtf]=zp2tf(numgzp',dengzp',K)   % Convert G(s) to polynomial form.
```

(ch2p11): LTI models can also be converted between polynomial and factored forms. MATLAB commands tf and zpk are also used for the conversion between LTI models. If a

transfer function, Fzpk(s), is expressed as factors in the numerator and denominator, then tf(Fzpk) converts Fzpk(s) to a transfer function expressed as coefficients in the numerator and denominator. Similarly, if a transfer function, Ftf(s) is expressed as coefficients in the numerator and denominator, then zpk(Ftf) converts Ftf(s) to a transfer function expressed as factors in the numerator and denominator. The following example demonstrates the concepts.

```
'(ch2p11)'                      % Display label.
'Fzpk1(s)'                      % Display label.
Fzpk1=zpk([-2 -4],[0 -3 -5],10) % Form Fzpk1(s)= 10(s+2)(s+4)/[s(s+3)(s+5)].
'Ftf1'                          % Display label.
Ftf1=tf(Fzpk1)                  % Convert Fzpk1(s) to coefficients form.
'Ftf2'                          % Display label.
Ftf2=tf([10 40 60],[1 4 5 7])   % Form Ftf2(s)= (10s^2+40s+60)/(s^3+4s^2+5s+7).
'Fzpk2'                         % Display label.
Fzpk2=zpk(Ftf2)                 % Convert Ftf2(s) to factored form.
```

(ch2p12): Functions of time, can be easily plotted using MATLAB's plot(X, Y, S), where X is the independent variable, Y is the dependent variable, and S is a character string describing the plot's color, marker, and line characteristic. Type HELP PLOT in the Command Window to see a list of choices for S. Multiple plots also can be obtained using plot(X1, Y1, S1, X2, Y2, S2, X3, Y3, S3, . . .). In the following example we plot on the same graph sin(5t) in red and cos(5t) in green for t = 0 to 10 seconds in 0.01 second increments. Time is specified as follows: t = start:increment:final.

```
'(ch2p12)'              % Display label.
t=0:0.01:10;            % Specify time range and increment.
f1=cos(5*t);            % Specify f1 to be cos(5t).
f2=sin(5*t);            % Specify f2 to be sin(5t)
plot(t,f1,'r',t,f2,'g') % Plot f1 in red and f2 in green.
pause
```

Constructing Transfer Functions, Laplace Transforms, and their Inverse using Symbolic Math Toolbox in MATLAB (APPENDIX F of NISE, 7th Edition)

(ch2sp1): MATLAB's calculating power is greatly enhanced using the Symbolic Math Toolbox. In this example we demonstrate its power by calculating inverse Laplace transforms of F(s). The beginning of any symbolic calculation requires defining the symbolic objects. For example, the Laplace transform variable, s, or the time variable, t, must be defined as a symbolic object. This definition is performed using the syms command. Thus, syms s defines s as a symbolic object; syms t defines t as a symbolic object; and syms s t defines both s and t as symbolic objects. We need to only define objects that we input to the program. Variables produced by the program need not be defined. Thus, if we are finding inverse Laplace transforms, we need to only define s as a symbolic object, since t results from the calculation. Once the object is defined, we can then type F as a function of s as we normally would write it. We do not have to use vectors to represent the numerator and denominator. The Laplace transforms or time

functions can also be printed in the MATLAB Command Window as we normally would write it. This form is called pretty printing. The command is $\text{pretty}(F)$, where F is the function we want to pretty print. In the code below you can see the difference between normal printing and pretty printing if you run the code without the semicolons at the steps where the functions, F or f, are defined. Once F is defined as F(s), we can find the inverse Laplace transform using the command $\text{ilaplace}(F)$. In the following example, we find the inverse Laplace transforms of the frequency functions in the examples used for Cases 2 and 3 in Section 2.2 in the text.

```
'(ch2sp1)'                      % Display label.
syms s                          % Construct symbolic object for Laplace variable 's'.
'Inverse Laplace transform'     % Display label.
F=2/[(s+1)*(s+2)^2];            % Define F(s) from Case 2 example.
'F(s) from Case 2'              % Display label.
pretty(F)                       % Pretty print F(s).
f=ilaplace(F);                  % Find inverse Laplace transform.
'f(t) for Case 2'               % Display label.
pretty(f)                       % Pretty print f(t) for Case 2.
F=3/[s*(s^2+2*s+5)];            % Define F(s) from Case 3 example.
'F(s) for Case 3'               % Display label.
pretty(F)                       % Pretty print F(s) for Case 3.
f=ilaplace(F);                  % Find inverse Laplace transform.
'f(t) for Case 3'               % Display label.
pretty(f)                       % Pretty print f(t) for Case 3.
```

(ch2sp2): In this example, we find Laplace transforms of time functions using the command, $\text{laplace}(f)$, where f is a time function, $f(t)$. As an example, we use the time functions that resulted from the calculations in Cases 2 and 3 in Section 2.2 in the text and work in reverse to obtain their Laplace transforms. We will see that the command, $\text{laplace}(f)$, yields $F(s)$ in partial fractions. In addition to pretty printing discussed in the previous example, the Symbolic Math Toolbox contains other commands that can change the look of the displayed result for readability and form. Some of these commands are: $\text{collect}(F)$—collect common coefficient terms of F; $\text{expand}(F)$—expands product of factors of F; $\text{factor}(F)$—factors F; $\text{simple}(F)$—finds simplest form of F with the least number of terms; $\text{simplify}(F)$—simplifies F; $\text{vpa}(\text{expression}, \text{places})$—standing for variable precision arithmetic, this command converts fractional symbolic terms into decimal terms with a specified number of decimal places. For example, the symbolic fraction, 3/16, would be converted to 0.1875 if the argument, places, were 4. In the example below, we find the Laplace transform of a time function. The result is displayed as partial fractions. To combine the partial fractions, we use the command, $\text{simplify}(F)$, where F is the Laplace transform of $f(t)$ found using $\text{laplace}(f)$. Finally, we use $F = \text{vpa}(F, 3)$ to convert the symbolic fractions to decimals in the displayed result.

```
'(ch2sp2)'                      % Display label.
syms t                          % Construct symbolic object for time variable 't'.
'Laplace transform'             % Display label.
'f(t) from Case 2'              % Display label.
```

```
f=2*exp(-t)-2*t*exp(-2*t)-2*exp(-2*t);      % Define f(t) from Case 2 example.
pretty(f)                                   % Pretty print f(t) from Case 2 example.
'F(s) for Case 2'                           % Display label.
F=laplace(f);                               % Find Laplace transform.
pretty(F)                                   % Pretty print partial fractions of F(s) for Case 2.
F=simplify(F);                              % Combine partial fractions.
pretty(F)                                   % Pretty print combined partial fractions.
'f(t) for Case 3'                           % Display label.
f=3/5-3/5*exp(-t)*[cos(2*t)+(1/2)*sin(2*t)];% Define f(t) from Case 3 example.
pretty(f)                                   % Pretty print f(t) for Case 3.
'F(s) for Case 3 - Symbolic fractions'      % Display label.
F=laplace(f);                               % Find Laplace transform.
pretty(F)                                   % Pretty print partial fractions of F(s) for Case 3.
'F(s) for Case 3 - Decimal representation'  % Display label.
F=vpa(F,3);       % Convert symbolic numerical fractions to 3-place decimal representation for F(s).
pretty(F)         % Pretty print decimal representation.
'F(s) for Case 3 - Simplified'              % Display label.
F=simplify(F);                              % Combine partial fractions.
pretty(F)                                   % Pretty print combined partial fractions.
```

(ch2sp3): MATLAB's Symbolic Math Toolbox may be used to simplify the input of complicated transfer functions as follows: Initially, input the transfer function $\mathrm{G(s)} = \mathrm{numg/deng}$ via symbolic math statements. Then convert G(s) to an LTI transfer function object. This conversion is done in two steps. The first step uses the command $[\mathrm{numg, deng}] = \mathrm{numden(G)}$ to extract the symbolic numerator and denominator of G. The second step converts, separately, the numerator and denominator to vectors using the command $\mathrm{sym2poly(S)}$, where S is a symbolic polynomial. The last step consists of forming the LTI transfer function object by using the vector representation of the transfer function's numerator and denominator. As an example, we form the LTI object $\mathrm{G(s)} = [54(\mathrm{s}+27)(\mathrm{s}^3+52\mathrm{s}^2+37\mathrm{s}+73)]/[\mathrm{s}(\mathrm{s}^4+872\mathrm{s}^3+437\mathrm{s}^2+89\mathrm{s}+65)(\mathrm{s}^2+79\mathrm{s}+36)]$, making use of MATLAB's Symbolic Math Toolbox for simplicity and readability.

```
'(ch2sp3)'                                  % Display label.
syms s                                      % Construct symbolic object for frequency variable 's'.
G=54*(s+27)*(s^3+52*s^2+37*s+73)...
/(s*(s^4+872*s^3+437*s^2+89*s+65)*(s^2+79*s+36));      % Form symbolic G(s).
'Symbolic G(s)'                             % Display label.
pretty(G)                                   % Pretty print symbolic G(s).
[numg,deng]=numden(G);                      % Extract symbolic numerator and denominator.
numg=sym2poly(numg);                        % Form vector for numerator of G(s).
deng=sym2poly(deng);                        % Form vector for denominator of G(s).
'LTI G(s) in Polynomial Form'               % Display label.
Gtf=tf(numg,deng)                           % Form and display LTI object for G(s) in polynomial form.
'LTI G(s) in Factored Form'                 % Display label.
Gzpk=zpk(Gtf)                               % Convert G(s) to factored form.
```

Exercises (To Be Completed In Class)

1. Use MATLAB to represent and generate the transfer function

$$G(s) = \frac{5(s+15)(s+26)(s+72)}{s(s+55)(s^2+5s+30)(s+56)(s^2+27s+52)}$$

2. Use MATLAB to generate partial fraction expansion of the following function

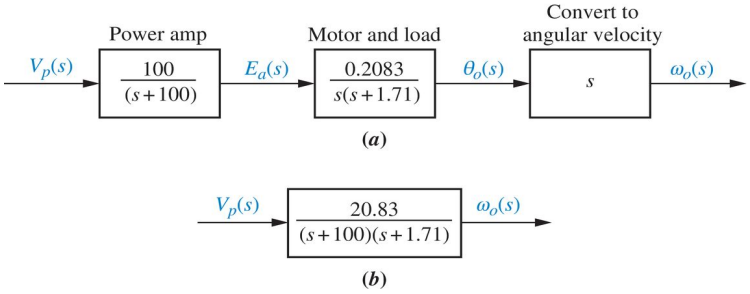
$$F(s) = \frac{10^4(s+5)(s+70)}{s(s+45)(s+55)(s^2+7s+110)(s^2+6s+95)}$$

3. Use MATLAB and the SYMBOLIC MATH TOOLBOX to find the Laplace Transform of the following functions

(1) $f(t) = 8t^2\cos(3t+45^\circ)$

(2) $f(t) = 3te^{-2t}\sin(4t+60^\circ)$

4. The forward transfer of the antenna azimuth position control system is given as



Using MATLAB find and plot the angular velocity response to a unit step input.