Drexel University

Office of the Dean of the College of Engineering

ENGR 232 – Dynamic Engineering Systems

Week 7 - Per Lab

Laboratory Primer:

1. **laplace, ilaplace**
2. **symbolic function vs. anonymous function (& its function handle)**
3. **matlabFunction availability issue**
4. **Solving a 2nd-order linear constant coefficient ODE – a MATLAB example**

1. laplace, ilaplace

The symbolic toolbox can be used to solve the Laplace transform and inverse Laplace transform of symbolic expressions. For example, find the Laplace transform of

$$y_1(t) = 3t^2 - 2\cos(t)\, e^{-4t}$$

And the inverse Laplace transform of

$$Y_2(s) = \frac{6}{s^3} - \frac{2(s+4)}{(s+4)^2+1}$$

```
syms s t           % Need to declare s and t as symbolic variables first.

% Ex 1: Find    L{y1(t)} = Y1(s) using "laplace" built-in function
Y1 = laplace(3*t^2 - 2*cos(t)*exp(-4*t))        % Note use of capital Y

% Ex 2: Find L^-1{Y2(s)} = y2(t) using "ilaplace" built-in function
y2 = ilaplace(6/s^3 - (2*(s+4))/((s+4)^2 + 1))  % Note use of lowercase y
```

Command Window Output:

```
Y1 =
6/s^3 - (2*(s + 4))/((s + 4)^2 + 1)


y2 =
3*t^2 - 2*exp(-4*t)*cos(t)
```

Note that in the examples above, $y_1(t) = L^{-1}\{Y_2(s)\}$. So, as expected, Y1 matches with the given $Y_2(s)$.

As a good practice,

- use uppercase for Laplace domain variable, lowercase for time domain variable;
- double check independent variable in the calculated expression. The variable Y1 is $Y_1(s)$, thus, it should only contain s, not t.

2. Symbolic function vs. anonymous function (& its function handle)

Recall that the variables Y1 and y2 above are of symbolic data type. They cannot be used in plot command directly. We need to use `matlabFunction` to convert them to "usual functions" first. Use `whos` command to check variable data type (class).

```
>> y2_func = matlabFunction(y2)
y2_func =
  function_handle with value:
    @(t)exp(t.*-4.0).*cos(t).*-2.0+t.^2.*3.0

>> whos y2 y2_func

  Name         Size            Bytes  Class             Attributes

  y2           1x1                 8  sym
  y2_func      1x1                32  function_handle
```

Note that y2 is of symbolic data type, whereas y2_func is of function handle data type.

In this class so far, each of the "functions" is defined in its own separate .m-file, like when we define the ODE function to be used by ode45. However, we can also quickly define a function without having to save it as a separate file. This is called "anonymous function" since it is not stored as a file. To refer to an anonymous function once it is created, we need to assign a variable to keep track of it. Such variable is called a "function handle". When we used matlabFunction to convert symbolic y2 into a "usual function", we assigned a new variable y2_func as its "handle". Since y2_func is a handle to a function, we can pass in some input values to calculate y2(t) at various time just like how we pass inputs to a function normally.

```
>> y2_func([0,1,2])

ans =
   -2.0000    2.9802   12.0003
```

The example above computes $[y_2(0), y_2(1), y_2(2)]$. Thus, to plot y2 vs. t, for example, do the following.

```
t = linspace(0,10,51); % equally spaced values between 0-10, 51 data points
plot(t, y2_func(t), 'r.-') % plot y2 vs. t. Note y2_func with (t) right after.
```

3. matlabFunction availability issue

If matlabFunction is not available on your machine, you may use a custom-made function "my_matlabFunction" posted on BB learn instead. my_matlabFunction imitates one of the behaviors of the actual built-in matlabFunction. It can convert a symbolic function expression into an anonymous function. You can call my_matlabFunction in the same manner like that of the actual one. For the sake of this lab, this should be sufficient.

- Limitation:
  - my_matlabFunction expects ONE independent variable with the name 't', or 's'.
- You are not required to understand how it works internally, or read the content of the function file.

4. Solving a 2nd-order linear constant coefficient ODE – a MATLAB example

Solve the following differential equation by using (1) `dsolve` and (2) using Laplace transforms (`laplace`, `ilaplace`):

$$y'' + y' - 12y = e^{-2t}, \ \ y(0) = 1, \qquad y'(0) = 0$$

```
clear; clc;
syms y t s

%% (1) Use dsolve
y_dsolve = dsolve('D2y+Dy-12*y= exp(-2*t)','y(0)=1','Dy(0)=0')

%% (2) Use Laplace transforms
% First, transforms both sides of the differential equation, and rearrange
% to get explicit solution for Y(s).
RHS = laplace(exp(-2*t)) % L{forcing function}
% LHS is suggested to be done manually, due to y'', y' and initial
% condition substitution. Refer to the lecture slide for guidance.

% The manually derived Y(s) equation is typed in MATLAB as shown below.
Y = ((1/(s+2)+s+1)/(s^2+s-12));

% Next, use MATLAB command "ilaplace" to inverse transform Y.
y_lap = ilaplace(Y)

%% Finally, check and compare if the results from (1) and (2) are the same.
```

<u>Command Window Output:</u>

```
y_dsolve =

(3*exp(3*t))/5 - exp(-2*t)/10 + exp(-4*t)/2



RHS =

1/(s + 2)



y_lap =

(3*exp(3*t))/5 - exp(-2*t)/10 + exp(-4*t)/2
```