MATLAB

Linearization, transfer functions and stuffs

2016 BRAZIL STUDY ABROAD PROGRAM

TEXAS A&M UNIVERSITY- UNIVERSITY OF SAO PAULO

KENNY ANDERSON QUEIROZ CALDAS

MAURÍCIO EIJI NAKAI

ELMER ALEXIS GAMBOA PEÑALOZA

RODOLPHO VILELA ALVES NEVES

RAFAEL FERNANDO QUIRINO MAGOSSI

MICHEL BESSANI

DEPARTAMENTO DE ENGENHARIA ELÉTRICA E COMPTAÇÃO, USP - SÃO CARLOS

SYMBOLIC OBJECTS AND SYMBOLIC EXPRESSIONS

Symbolic objects can be variables or numbers. They can be created with the **sym** and/or **syms** commands. A single symbolic object can be created with the sym command:

```
object_name = sym('string')
```

```
syms variable_name variable_name variable_name
```

Examples

```
a=sym('a')
a =
a
>> bb=sym('bb')
bb =
bb
>> x=sym('x');
```

```
>> syms y z d
>> y
y =
y
```

SOLVING ALGEBRAIC EQUATIONS

A single algebraic equation can be solved for one variable, and a system of equations can be solved for several variables with the solve function

```
h = solve(eq)
```

or

```
h = solve(eq,var)
```

Examples

$$ax^2 + bx + c = k$$

```
>> syms a b c k x
>> eq = a*x^2 + b*x + c-k;
>> pretty(eq)
>>  X = solve(eq,x);
>> pretty(X)
```

Hands on!

Let's consider the linear system

$$x - 2y + z = 12 \qquad eq1$$
$$3x + 4y + 5z = 20 \qquad eq2$$
$$-2x + y + 7z = 11 \qquad eq3$$

Find the solution using the Matlab command
[x1,x2,x3]=*solve(eq1,eq2,eq3)*

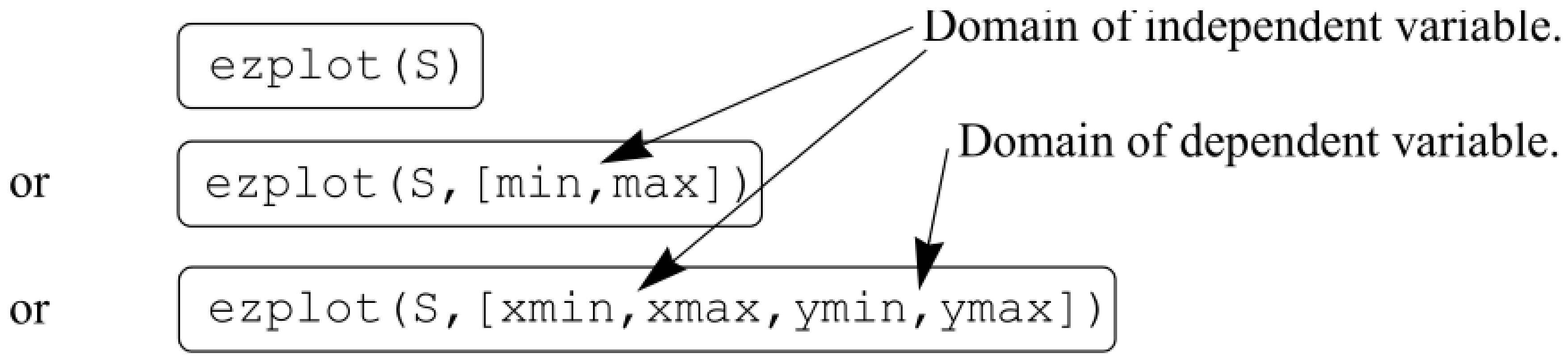
Hands on!

Solution:

```
>> syms x y z;
>> eq1 = x - 2*y+z-12;
>> eq2 = 3*x+4*y+5*z-20;
>> eq3 = -2*x+y+7*z-11;
>>[X,Y,Z] = solve(eq1,eq2,eq3)
```

PLOTTING SYMBOLIC E XPRESSIONS

In many cases, there is a need to plot a symbolic expression. This can easily be done with the ezplot command.



Example

```
>> syms x
>> S=(3*x+2)/(4*x-1)
>> ezplot(S)
```

Hands on!

Plot the following equations:

1) Circle

$$x^2 + y^2 = 1$$

2) Ellipse

$$4x^2 - 18x + 4y^2 + 12y - 11 = 0$$

Hands on!

Plot the following equations:

3) Plot x vs y

$$x = \cos(2 * t)$$
$$y = \sin(4 * t)$$

Hands on!

Solution:

1)

```
>> syms x y
>>S = x^2 + y^2-1
>>ezplot(S)
```

2)

```
>> syms x y
>> S=4*x^2-18*x+4*y^2+12*y-11
>> ezplot(S)
```

3)

```
>> syms t
 >> x = cos(2*t)
 >> y = sin(4*t)
 >> ezplot(x,y)
```

Laplace transform

Matlab has a command to compute the Laplace transform on time-domain equation. The syntax is:

```
laplace(F)
laplace(F, t)
```

Examples

```
>>syms t a;
>>f = exp(-a*t);
>>laplace(f)
ans =

    1/(a + s)
```

Hands on!

Calculate the Laplace transform:

1) Unit step $u(t)$ (tip: on matlab unit step is heaviside(t))

2) sin(w*t)

3) Unit impulse $\delta(t)$ (tip: on matlab unit impulse is dirac(t))

4) cos(w*t)

Laplace transform

Also, there is another command to compute the inverse of the Laplace transform. The syntax is:

```
F = ilaplace(L)
```

Examples

```
>> syms s a;
>>L = 1/(s+a);
>> ilaplace(L)
ans =
        exp(-a*t)
```

Hands on!

Calculate the inverse Laplace transform:

1) $1/s$

2) $w/(s^2+w^2)$

3) 1

4) $s/(s^2+w^2)$

5) $1/(s + a)^2$

Partial fraction

Whenever you have to work with fractions, it's always difficult to simplify them. Matlab can reduce this problem with some lines of code. The *residue()* command can give the partial fractions from a fraction.

Example

$$F(s) = \frac{b(s)}{a(s)} = \frac{5s^3 + 3s^2 - 2s + 7}{-4s^3 + 8s + 3}.$$

```
>>b = [ 5 3 -2 7];
>>a = [-4 0 8 3];
 >>[r,p,k] = residue(b,a)
```

```
r = -1.4167 -0.6653 1.3320
p = 1.5737 -1.1644 -0.4093
k = -1.2500
```

$$F(s) = \frac{b(s)}{a(s)} = \frac{-1.4167}{s - 1.5737} - \frac{0.6653}{s + 1.1644} + \frac{1.3320}{s + 0.4093} - 1.2500.$$

DIFFERENTIATION

Symbolic differentiation can be carried out by using the *diff()* command. The syntax of the command is:

`diff(S)` or `diff(S,var)`

Examples

```
>> syms x
>> S=exp(x^4);
>> diff(S)

4*x^3*exp(x^4)
```

```
>>syms x
>> S=exp(x^4);
>>diff(S,2)

12*x^2*exp(x^4)+16*x^6*exp(x^4)
```

Laplace transform

Examples

$$y''(t) + 7y'(t) + 12y(t) = 0$$

```
>> syms y(t) t;
>> laplace(diff(diff(y(t),t))+diff(y(t),t)*7+y(t)*12);
ans =
        7*s*laplace(y(t), t, s) - D(y)(0) - 7*y(0) - s*y(0) +
        s^2*laplace(y(t), t, s) + 12*laplace(y(t), t, s)
```

laplace(y(t), t, s) means Y(s)

Transfer function

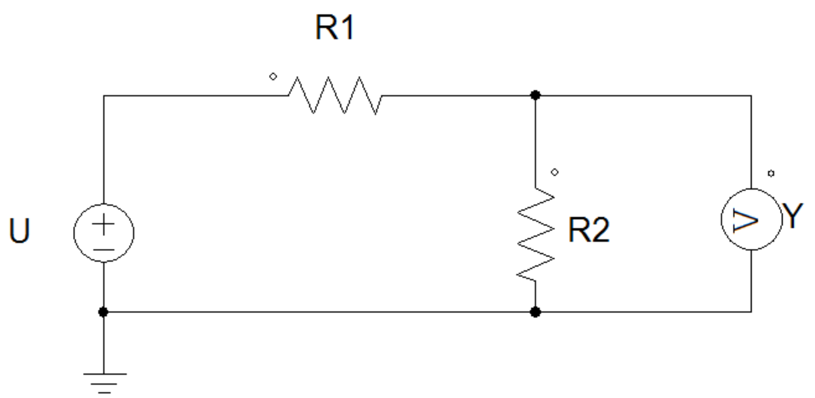
One way to show the input-output relation is using transfer functions. Matlab can compute and work with TF in many different ways.
Commands like tf(num,den) and tf('s') create a TF object that can be used on Matlab routines.

Example

$$G(s) = \frac{s+1}{s^2+2s+1}$$

```
>> G1 = tf([1 1],[1 2 1]);
>> s = tf('s');
>> G2 = (s+1)/(s^2+2*s+1);
```

Hands on!



We know that:

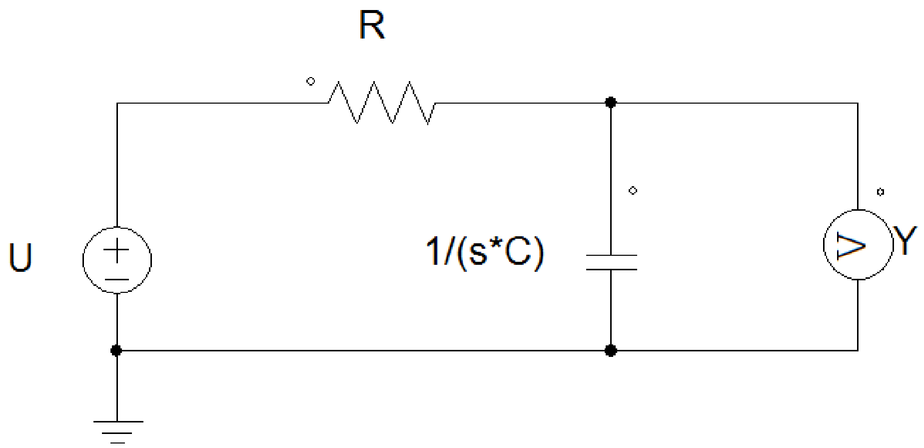
$$Y = \frac{R2}{R1 + R2} U$$

Then,

$$G(s) = \frac{Y}{U} = \frac{R2}{R1 + R2}$$

Hands on!

Get the transfer function of the RC circuit and check the charge and discharge curve of the capacitor. Consider R = 1kΩ and C = 1000uF.



TIP:

To check the charge curve use the step command

To check the discharge use the impulse command

State-space and transfer function

If you have the matrices A,B,C and D, it's possible to use Gss = ss(A,B,C,D) to create a state-space system. To get the transfer function you can use Gtf = tf(Gss).

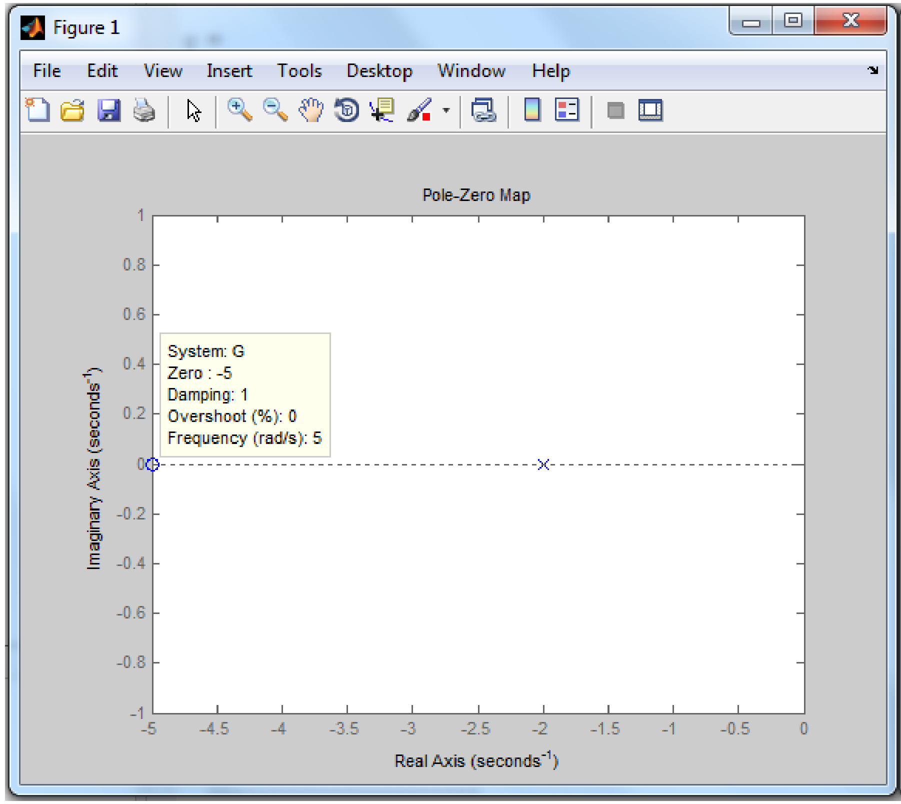
Example

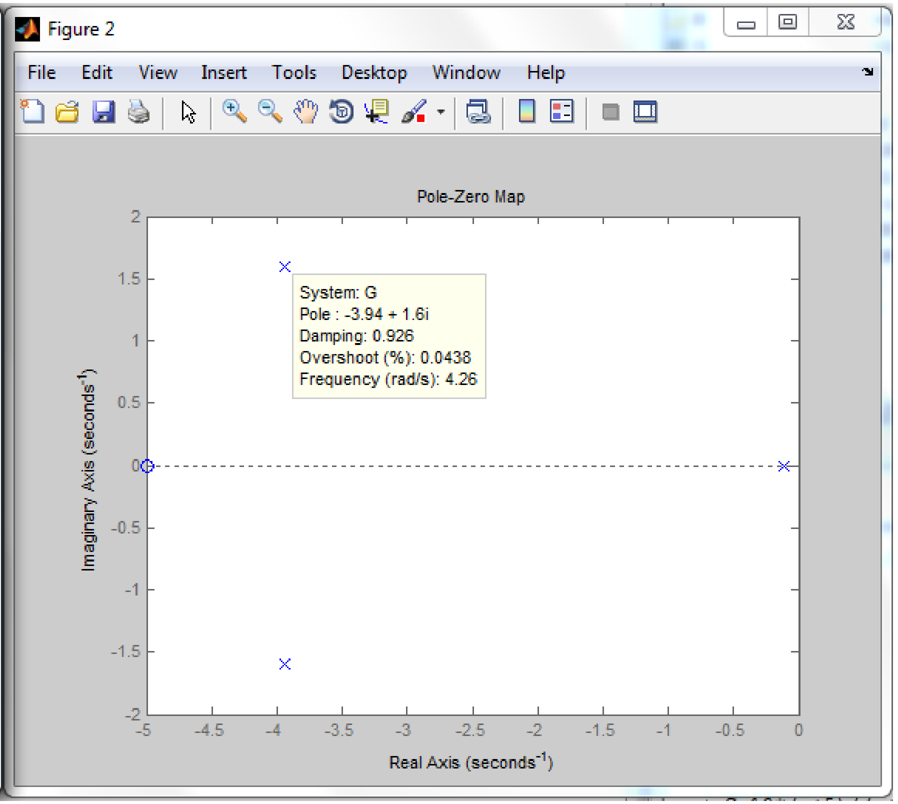
```
>>A = [-2 -1; 1 0];
>>B = [1;0];
>>C = [1 1];
>>D = 0;
>> Gss =ss(A,B,C,D);
>> G = tf(Gss)
```

```
>>G =

    s + 1
  -------------
  s^2 + 2 s + 1

Continuous-time transfer function.
```

Poles and zeros

```
figure(1)
s=tf('s');
G=10*(s+5)/(s+2);
pzmap(G)
pole(G)
figure(2)
G=(2*s+10)/(s^3+8*s^2+19*s+2)
pzmap(G)
pole(G)
```





Local linearization

```
% MAGLEV System
%xdot=f(x,v)
syms x1 x2 x3 v g L0 a m R L ka c1
BL=[0;0;ka/L];
CL=[-c1 0 0]; DL=0;
P=[g m a R L ka L0];
f=[x2;g-L0/(2*a*m)*(x3^2/(1+x1/a)^2);-R/L*x3+ka/L*v];
As=jacobian(f,[x1 x2 x3]);
```

```
% Parameters values  MAGLEV of teaching laboratory
g=9.8;m=22.6e-3;a=6.72e-3;R=19.9;L=0.52;ka=2.4;
L0=0.0249; c1=173.61e+1;
% equilibrium point: xdot=0
x1eq=4.5e-3; %calculate the value of  x3eq
x3eq=sqrt(g*2*a*m*(1+x1eq/a)^2/L0);
veq=R*x3eq/ka;

AL=simplify(subs(As,[x1 x2 x3 v],[x1eq 0 x3eq veq]));
pretty (AL)

AL = eval(AL);
BL = eval(BL); % Space state matrix
CL = eval(CL);
DL = DL;
```

Hands on!

Get the transfer function and poles/zeros localization of the MAGLEV system example.

Hands on!

Get the transfer function of the MAGLEV system example

```
% Example MAGLEV
>>Gmaglev_ss =ss(AL,BL,CL,DL);
>>Gmaglev = tf(Gmaglev_ss);
>>pzmap(Gmaglev)
>>pole(Gmaglev)
```

Response × poles localization

```
clear all; close all; clc;
s = tf('s');


%% Case 1 – Simple poles
 p1 = 1;
 G1 = 1/(s+p1);
figure(1)
impulse(G1)
```

```
%Case 2 – Real positive poles


p5 = 5;
G2 = 1/(s-p5);


figure(2)
impulse(G2)
```

Response × poles localization

```
%Case 3 – Complex poles

s = tf('s')

omegan = 100; % Natural frequency

zeta1 = [0 0.5 1 1.5]; % Damping values

 %Calculate different transfer functions

for n = 1:4

    zeta = zeta1(n);

    G3(n)=omegan^2  /(s^2+
2*zeta*omegan*s+omegan^2);

end
```

```
%Plotting typical responses to the
transfer functions

for k = 1:size(G3,2)

    figure(3)

    hold on

    step(G3(k),0:.0001:.2)

    hold off

end
```

Test#2

Using the transfer functions from Case 1, 2 and 3, do:

a) Find the transfer functions with complex and real poles, plot the step response and comment on the results.

b) Find the transfer functions that only have complex imaginary poles, plot the impulse response and comment on the results.

c) Find the transfer functions that only have real positive poles, plot the step response and comment on the results.

References

[1] Matlab Product Help.

[2]Matlab Demystified. A Self-Teaching Guide, David McMahon, McGraw Hill.