Universidade

Estadual de Londrina

Centro de Tecnologia e Urbanismo
Departamento de Engenharia Elétrica

Laboratório de 2ele044 T-1011 e T-1012 **Londrina, __ de _________de 2015.**

Nome:

Encontre a solução para os programas.

1. (1): Bit strings will be used to identify parts of this tutorial on the computer output. Bit strings are represented by the text enclosed in apostrophes, such as 'ab'. Comments begin with and are ignored by MATLAB. Numbers are entered without any other characters. Arithmetic can be performed using the proper arithmetic operator. Numbers can be assignedusing a left-hand argument and an equals sign.

```
'(1)'                                  % Display
label.
'How are you?'                         % Display
string.
-3.96                                  % Display
scalar number -3.96.
-4+7i                                  % Display
complex number -4+7i.
-5-6j                                  % Display
complex number -5-6j.
(-4+7i)+(-5-6i)                        % Add two
complex numbers and
                                     %   display
sum.
```

Nome:

```
(-4+7j)*(-5-6j)                                    % Multiply
two complex numbers and
                                                  %   display
product.
M=5                                                % Assign 5
to M and display.
N=6                                                % Assign 6
to N and display.
?????
```

Resultado 11.

2. (2): Transfer function numerator and denominator vectors can be converted between polynomial form containing the coefficients and factored form containing the roots. The MATLAB function, 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, 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

Universidade

Estadual de Londrina

Centro de Tecnologia e Urbanismo
Departamento de Engenharia Elétrica

Laboratório de 2ele044 T-1011 e T-1012 **Londrina, __ de _________de 2015.**

Nome:

conversion from roots to coefficients with G(s) = 10(s+2)(s+4)/[s(s+3)(s+5)].

```
'(2)'                              % 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
```

3. (3):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.

```
'(3)'                              % Display label.
'Fzpk1(s)'                                 % Display
label.
Fzpk1=zpk([-2 -4],[0 -3 -5],10)
```

4. (4): 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 = tf(numf,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 = zpk(numg,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 4.2): This method allows you to type the transfer function as you normally would write it. The statement s = tf('s') must precede the transfer function if you wish to create an LTI transfer function in polynomial form equivalent to using G=tf(numg,deng).

(4) Rational Expression in s Method, Factored Form (Requires Control System Toolbox 4.2): This method allows you to type the transfer function as you normally would write it. The statement s = zpk('s') must precede the

transfer function if you wish to create an LTI transfer function in factored form equivalent to using G = zpk(numg,deng,K).

For both rational expression methods the transfer function can be typed in any form regardless of whether s = tf('s') or s = 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.

```
'(4)'                                          % 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
form.
```

5. (5) 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).

Universidade

Estadual de Londrina

Centro de Tecnologia e Urbanismo
Departamento de Engenharia Elétrica

Laboratório de 2ele044 T-1011 e T-1012 **Londrina, __ de _________de 2015.**

Nome:

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 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 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 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(s) is defined as F, we can find the inverse Laplace transform using the command ilaplace(F). In the example below,

Universidade

Estadual de Londrina

Centro de Tecnologia e Urbanismo
Departamento de Engenharia Elétrica

Laboratório de 2ele044 T-1011 e T-1012 **Londrina, __ de _________de 2015.**

Nome:

```
we find the inverse Laplace transforms of the
frequency functions .
% in the text.

'(5)'                        % 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
```

6. Explique esse programa.

```
clear all
clc

%declarar variaveis
io=1
G=1
H=1

%F1_7b  Signal g(t) multiplied by a pulse
functions
t= -2:0.01:10;
```

Laboratório de 2ele044 T-1011 e T-1012 **Londrina, __ de _________de 2015.**

Nome:

```
q=size(t);
r=size(t);
f=zeros(q(1),q(2)); % set f = a vector of zeros
ff=zeros(r(1),r(2));
q=size(t(201:1201));
r=size(t(701:1201));
f(201:1201)=ones(q(1),q(2));% set final 1200
points of f to 1
ff(701:1201)=ones(r(1),r(2));% set final 1200
points of f to 1
rr=io*(exp(-(G/H)*t).*f-(exp(-(G/H)*(t-
5))).*ff);

plot(t,rr),title('Fig.1.7a Unit step
function');
axis([-2,10,-1,2]); % sets limits on axes
xlabel('time, t');
ylabel(' u(t)');
grid;
```

7. (7) MATLAB's Symbolic Math Toolbox may be used to simplify the input of complicated transfer functions as follows: Initially, input the transfer function G(s) = 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 [numg,deng]=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 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
% G(s) = [54(s+27)(s^3+52s^2+37s+73)]/
% [s(s^4+872s^3+437s^2+89s+65)(s^2+79s+36)],
making use of MATLAB's Symbolic
% Math Toolbox for simplicity and readability.

'(7)'                          % 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).
```