MATLAB Session 2 Results

The collection of MATLAB statements and screen display:

```
>> %Finding poles and partial fractions
>> p = [1 5 4]

p =

     1     5     4

>> poles = roots(p)

poles =

    -4
    -1

>> p2 = poly(poles)

p2 =

     1     5     4

>> q = 1;
>> residue(q,p)

ans =

   -0.3333
    0.3333

>> [a,b,k]=residue(q,p)

a =

   -0.3333
    0.3333


b =

    -4
    -1


k =

     []

>> %The next example
>> poles=[0 -1 -2 -3];
>> p=poly(poles);
>> q=1;
>> [a,b,k]=residue(q,p)

a =
```

```
   -0.1667
    0.5000
   -0.5000
    0.1667


b =

   -3.0000
   -2.0000
   -1.0000
         0


k =
     []
```

```
>> %Another Example
>> q=[1 4 3];
>> zeros=roots(q)

zeros =

    -3
    -1

>> p=[1 -7 11 7 -12];
>> poles=roots(p)

poles =

    4.0000
    3.0000
    1.0000
   -1.0000

>> [a,b,k]=residue(q,p)

a =

    2.3333
   -3.0000
    0.6667
         0


b =

    4.0000
    3.0000
    1.0000
   -1.0000


k =

     []
```

Warning

"`zeros`" is a MATLAB function. When we use it as a variable name, the function will no longer work. To revive the function if in case you need to use it, enter "`clear zeros`" to wipe out the variable. But using "`poles`" as a variable name is fine. It is "`pole`" in this case that it is a MATLABfunction.

```
>> %Use of pole-zero form and doing "conversions"
>> q=[6 0 -12];
>> p=[1 1 -4 -4];
>> [zeros,poles,k]=tf2zp(q,p)

zeros =

    1.4142
   -1.4142


poles =

   -2.0000
    2.0000
   -1.0000


k =

     6

>> [q,p]=zp2tf(zeros,poles,k)

q =

         0    6.0000   -0.0000  -12.0000


p =

    1.0000    1.0000   -4.0000   -4.0000

>> %Another simple example
>> zero= -2;
>> poles=[-4 -3 -1];
>> k=1;
>> [q,p]=zp2tf(zero,poles,k)

q =

     0     0     1     2


p =

     1     8    19    12

>> %Double check the result
>> [zero,poles,k]=tf2zp(q,p)

zero =

    -2


poles =
```

```
   -4.0000
   -3.0000
   -1.0000


k =

     1

>> %Also can check with:
>> roots(q)

ans =

    -2

>> roots(p)

ans =

   -4.0000
   -3.0000
   -1.0000

>> %Use of object-oriented transfer function
>> G1 = tf([1 0], [1 -5 4])

Transfer function:
      s
-------------
s^2 - 5 s + 4

>> G2 = tf([6 -12], [1 1 -4 -4])

Transfer function:
     6 s - 12
--------------------
s^3 + s^2 - 4 s - 4

>> G3 = zpk([],[0 -1 -2 -3], 4)

Zero/pole/gain:
         4
--------------------
s (s+1) (s+2) (s+3)

>> %Convert between different forms:
>> tf(G3)

Transfer function:
            4
-------------------------
s^4 + 6 s^3 + 11 s^2 + 6 s

>> zpk(G2)

Zero/pole/gain:
     6 (s-2)
```

```
-----------------
(s+2) (s+1) (s-2)

>> %Find the poles of a transfer function
>> pole(G1)

ans =

     4
     1

>> pole(G2)

ans =

   -2.0000
    2.0000
   -1.0000

>> %Extract the information back from a transfer function
>> [q,p]=tfdata(G1,'v')

q =

     0     1     0


p =

     1    -5     4

>> [z,p,k]=zpkdata(G3,'v')

z =

   Empty matrix: 0-by-1    %<--That's correct, G3 has no zeros


p =

     0
    -1
    -2
    -3


k =

     4

>> %MATLAB operators are overloaded for the transfer functions
>> G1=tf(1,[1 1])

Transfer function:
  1
-----
s + 1
```

```
>> G2=tf(2,[1 2])

Transfer function:
  2
-----
s + 2

>> G1+G2

Transfer function:
   3 s + 4
-------------
s^2 + 3 s + 2

>> G1*G2

Transfer function:
      2
-------------
s^2 + 3 s + 2

>> %Looking inside the transfer function object
>> get(G1)
             num: {[0 1]}
             den: {[1 1]}
        Variable: 's'
              Ts: 0
      InputDelay: 0
     OutputDelay: 0
   ioDelayMatrix: 0
       InputName: {''}
      OutputName: {''}
      InputGroup: {0x2 cell}
     OutputGroup: {0x2 cell}
           Notes: {}
        UserData: []
>> G1.InputName = 'Flow Rate';
>> G1.OutputName = 'Level';
>> G1.Notes = 'My first MATLAB function';
>> G1

Transfer function from input "Flow Rate" to output "Level":
  1
-----
s + 1

>> get(G1)
             num: {[0 1]}
             den: {[1 1]}
        Variable: 's'
              Ts: 0
      InputDelay: 0
     OutputDelay: 0
   ioDelayMatrix: 0
       InputName: {'Flow Rate'}
      OutputName: {'Level'}
      InputGroup: {0x2 cell}
     OutputGroup: {0x2 cell}
           Notes: {'My first MATLAB function'}
```

```
        UserData: []
```

```
>> %Symbolic algebra (reading optional -- it is based on the Web
Supplement notes)
>> p = 'a*x^2+b*x+c'

p =

a*x^2+b*x+c

>> diff(p)

ans =

2*a*x+b

>> p = sym('a*x^2+b*x+c')

p =

a*x^2+b*x+c

>> p=[1 2 0 3];
>> sp=poly2sym(p)

sp =

x^3+2*x^2+3

>> pretty(sp)

                                   3      2
                                  x  + 2 x  + 3
>> sym2poly(sp)

ans =

     1     2     0     3

>> %Illustration of doing Laplace transform
>> f=sym('exp(-a*t)')

f =

exp(-a*t)

>> F=laplace(f)

F =

1/(s+a)

>> fcheck=ilaplace(F)
```

```
fcheck =

exp(-a*t)

>> laplace(sym('sin(freq*t)'))

ans =

freq/(s^2+freq^2)

>> ilaplace(ans)

ans =

freq/(freq^2)^(1/2)*sin((freq^2)^(1/2)*t)  %<--very clumsy result
from the new MATLAB!

>> simplify(ans)    %<--add a step to simplify the answer

ans =

csgn(freq)*sin(csgn(freq)*freq*t)    %It's correct. csgn is just the
sign

>> f=sym('exp(-a*t)*sin(freq*t)')

f =

exp(-a*t)*sin(freq*t)

>> laplace(f)

ans =

freq/((s+a)^2+freq^2)

>> ilaplace(ans)

ans =

freq/(-4*freq^2)^(1/2)*(exp((-a+1/2*(-4*freq^2)^(1/2))*t)-exp((-a-
1/2*(-4*freq^2)^(1/2))*t))
%This is a big mess too, but we'll not dwell on this MATLAB
deficiency since we won't really use
% symbolic algebra to design controllers

>>
>> syms w t x;    %The new version gave me an error without the x
>> laplace(sin(w*x),t)

ans =

w/(t^2+w^2)

>> %Re-visiting Example 2.4 here. We first make the transfer
function
>> q=[6 0 -12];
```

```
>> sq=poly2sym(q);
>> sp=poly2sym([1 1 -4 -4]);
>> g=sq/sp

g =

(6*x^2-12)/(x^3+x^2-4*x-4)

>> pretty(g)

                                   2
                                6 x  - 12
                            -----------------
                             3    2
                            x  + x  - 4 x - 4

>> %Now, we can find the partial fraction and time domain function
>> diff(int(g))

ans =

1/(x-2)+3/(x+2)+2/(x+1)

>> subs(ans,'s','x')

ans =

1/((s)-2)+3/((s)+2)+2/((s)+1)

>> pretty(ans)

                              1       3       2
                            ----- + ----- + -----
                            s - 2   s + 2   s + 1

>> ilaplace(ans)

ans =

exp(2*t)+3*exp(-2*t)+2*exp(-t)

>> ilaplace(g)    %Maybe it is redundant, but we double the check
results

ans =

exp(2*t)+3*exp(-2*t)+2*exp(-t)

>> laplace(ans)

ans =

1/(s-2)+3/(s+2)+2/(s+1)

>> pretty(ans)
```

```
       1         3         2
     ----- + ----- + -----
     s - 2   s + 2   s + 1
```

```
>> %The last example in the Session
>> poles=[0 -1 -2 -3];
>> p=poly(poles);
>> sp=poly2sym(p)

sp =

x^4+6*x^3+11*x^2+6*x

>> factor(sp)

ans =

x*(x+3)*(x+2)*(x+1)

>> tf=symdiv('1',sp)

tf =

1/(x^4+6*x^3+11*x^2+6*x)

>> ilaplace(tf)

ans =

1/6-1/6*exp(-3*t)+1/2*exp(-2*t)-
1/2*exp(-t)

>> diff(int(tf))   %Again, checking with the redundant approach

ans =

1/6/x-1/6/(x+3)+1/2/(x+2)-1/2/(x+1)

>> ilaplace(ans)

ans =

1/6-1/6*exp(-3*t)+1/2*exp(-2*t)-1/2*exp(-t)
```

Warning

"`tf`" is a MATLAB function. When we use it as a variable name, the function will no longer work. To revive the function if in case you need to use it, enter "`clear tf`" to wipe out the variable.