Control System Engineering

Matlab Exmaples

Youngshik Kim, Ph.D.
Mechanical Engineering
youngshik@hanbat.ac.kr

국립 한밭대학교
HANBAT NATIONAL UNIVERSITY

Partial Fraction: residue

```
>> help residue
 RESIDUE Partial-fraction expansion (residues).
   [R,P,K] = RESIDUE(B,A) finds the residues, poles and direct term of
   a partial fraction expansion of the ratio of two polynomials B(s)/A(s).
   If there are no multiple roots,
     B(s)      R(1)      R(2)           R(n)
     ----  =  -------- + -------- + ... + -------- + K(s)
     A(s)     s - P(1)   s - P(2)       s - P(n)
   Vectors B and A specify the coefficients of the numerator and
   denominator polynomials in descending powers of s.  The residues
   are returned in the column vector R, the pole locations in column
   vector P, and the direct terms in row vector K.
```

Partial Fraction: residue

- Example:

$$Y(s) = \frac{(s+2)(s+4)}{(s^2+s)(s+3)} = -\frac{1/6}{s+3} - \frac{3/2}{s+1} + \frac{8/3}{s}$$

$$= \frac{-0.1667}{s+3} + \frac{-1.5000}{s+1} + \frac{2.6667}{s}$$

```
num = conv ([1 2],[1 4]);      % form numerator polynomial
den = conv ([1 1 0],[1 3]);    % form denominator polynomial
[r,p,k] = residue (num,den)    % compute the residues
```

[r : residues, p : poles, k : direct terms]

```
r = [-0.1667 -1.5000 2.6667]';   p = [-3  -1  0]';   k=[];
```

Partial Fraction: residue

- Example

$$Y(s) = \frac{2}{(s+2)(s+1)(s+4)} = \frac{1/3}{s+4} - \frac{1}{s+2} + \frac{2/3}{s+1}$$

```
num = 2;                    % form numerator
den = poly([-2;-1;-4]);     % form denominator polynomial from its roots
[r , p , k] = residue(num , den)   % compute the residues
```

$$r = \begin{bmatrix} 0.3333 & -1 & 0.6667 \end{bmatrix}' \quad p = \begin{bmatrix} -4 & -2 & 1 \end{bmatrix}' \quad k = \begin{bmatrix} \ \end{bmatrix}$$

Transfer Function: tf

```
>> help tf
 TF  Constructs transfer function or converts to transfer function.

   Construction:
     SYS = TF(NUM,DEN) creates a continuous-time transfer function SYS with
     numerator NUM and denominator DEN. SYS is an object of class @tf.

     SYS = TF(NUM,DEN,TS) creates a discrete-time transfer function with
     sampling time TS (set TS=-1 if the sampling time is undetermined).

>>  sys = tf([1 2],[1 0 10])

Transfer function:
 s + 2
--------
s^2 + 10
```

Symbolic Objects: syms

```
>> help syms
 SYMS   Short-cut for constructing symbolic objects.
   SYMS arg1 arg2 ...
   is short-hand notation for
     arg1 = sym('arg1');
     arg2 = sym('arg2'); ...

 Each input argument must begin with a letter and must contain only
   alphanumeric characters.

   By itself, SYMS lists the symbolic objects in the workspace.

   Examples:
>>   syms x beta real
```

Laplace Transforms: laplace

```
>> help laplace
 --- help for sym/laplace ---

 LAPLACE Laplace transform.
    L = LAPLACE(F) is the Laplace transform of the scalar sym F with
    default independent variable t.  The default return is a function
    of s.  If F = F(s), then LAPLACE returns a function of t:  L = L(t).
    By definition L(s) = int(F(t)*exp(-s*t),0,inf), where integration
    occurs with respect to t.

Examples:
      syms a s t w x
      laplace(t^5)         returns  120/s^6
      laplace(exp(a*s))     returns  1/(t-a)
      laplace(sin(w*t))    returns w/(s^2 + w^2)
      laplace(diff(sym('F(t)')))  returns  laplace(F(t),t,s)*s-F(0)
```

Inverse Laplace Transform: ilaplace

```
>> help ilaplace
 --- help for sym/ilaplace ---

 ILAPLACE Inverse Laplace transform.
   F = ILAPLACE(L) is the inverse Laplace transform of the scalar sym L
   with default independent variable s.  The default return is a
   function of t.  If L = L(t), then ILAPLACE returns a function of x:
   F = F(x).

 Examples:
     syms s t w x y
     ilaplace(1/(s-1))          returns   exp(t)
     ilaplace(1/(t^2+1))          returns   sin(x)
```

Laplace Transforms

- Example

```
>> syms t; f = t^4; F=laplace(f)

F =

24/s^5

>> ilaplace(F)

ans =

t^4
```

Zeros, Poles, Gains: tf2zp

```
>> help tf2zp
 TF2ZP  Transfer function to zero-pole conversion.
   [Z,P,K] = TF2ZP(NUM,DEN)  finds the zeros, poles, and gains:

                        (s-z1)(s-z2)...(s-zn)
      H(s) =  K ---------------------
                        (s-p1)(s-p2)...(s-pn)

   from a SIMO transfer function in polynomial form:

            NUM(s)
      H(s) = --------
            DEN(s)

   Vector DEN specifies the coefficients of the denominator in
   descending powers of s.  Matrix NUM indicates the numerator
   coefficients with as many rows as there are outputs.
```

Zeros, Poles, Gains: tf2zp

- Example: $\ddot{y} + 6\dot{y} + 25y = 9u + 3\dot{u}$ $\quad \frac{Y(s)}{U(s)} = \frac{3s+9}{s^2+6s+25}$

```
numG = [3   9];
denG = [1   6   25];
[z , p , k] = tf2zp(numG , denG)
[numG , denG] = zp2tf(z , p , k)
```

$$z = -3 \qquad p = [-3+4j \quad -3-4j]' \qquad k = 3$$

$$numG = [0 \quad 3 \quad 9] \qquad denG = [1 \quad 6 \quad 25]$$

Step Response: step

>> help step
STEP Step response of dynamic systems.

STEP(SYS) plots the step response of the dynamic system SYS. For multi-input models, independent step commands are applied to each input channel. The time range and number of points are chosen automatically.

STEP(SYS,TFINAL) simulates the step response from t=0 to the final time t=TFINAL. For discrete-time models with unspecified sampling time, TFINAL is interpreted as the number of samples.

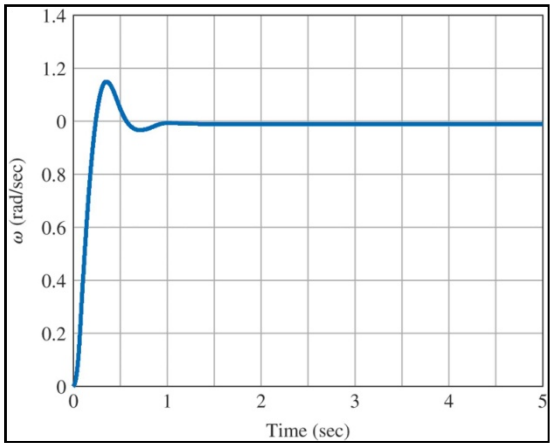
STEP(SYS,T) uses the user-supplied time vector T for simulation. For discrete-time models, T should be of the form Ti:Ts:Tf where Ts is the sample time. For continuous-time models, T should be of the form Ti:dt:Tf where dt will become the sample time for the discrete approximation to the continuous system. The step input is always assumed to start at t=0 (regardless of Ti).

Step Response: step

- DC motor:

$$\frac{s\Theta_m(s)}{V_a(s)} = G(s) = \frac{100s}{s^3 + 10.1s^2 + 101s} = \frac{100}{s^2 + 10.1s + 101}$$

```
numb = [0  0  100];
denb = [1  10.1  101];
sysb = tf(numb , denb);
t = 0 : 0.01 : 5;
y = step(sysb , t);
plot(t , y)
```



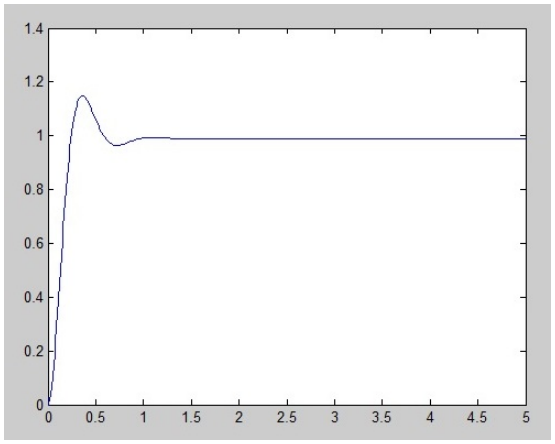


Figure 3.5 Transient response for DC motor