MATLAB 수업 자료 (기계항공시스템해석 2009.03.18)

1. Matlab의 소개

1.1 Matlab 이란?

Matlab이란 MATrix LABoratory를 뜻하는 말로서, 수치해석, 행렬연산, 신호처리 및 간편한 그래픽 기능 등을 통합하여 고성능의 수치계산 및 결과의 가시화 기능을 제공하는 프로그램이다.

Matlab은 행렬과 벡터를 기본 자료로 사용하여 기능을 수행하는 계산환경을 제공한다. Matlab은 기본적으로 행렬을 자료로 다루기 때문에 dimensioning이 필요하지 않으며 통상적인 프로그래밍 언어들을 사용하여 프로그램을 작성하지 않고도 쉽게 수치 계산을 수행할 수 있다.

Matlab의 가장 큰 특징은 M-file(Matlab code가 포함된 Text 파일, 확장자는 .m)을 사용함으로써 특정한 해를 구하는데 필요한 응용 프로그램을 손쉽게 작성할 수 있다는 점이다.

2. 행렬(Matrices) 다루기

2.1 행렬과 Matlab

Matlab에서 행렬은 직사각형의 숫자들의 배열이다. 1-by-1 행렬은 스칼라를 1개의 row또는 column으로 이루어진 행렬은 벡터를 나타낸다. Matlab은 다른 언어들이 한 번에 숫자들을 일일이 순차적으로 계산하는 것과 달리, 전체 행렬을 빠르고 쉽게 계산할 수 있게 해준다. Matlab은 행렬 외에도 다른 수치 저장 방법들을 가지지만, 먼저 행렬을 다루는 법을 잘 익히면 Matlab에 쉽게 접근할 수 있다.

(1) 행렬 입력하기

- 행렬 입력 방법

a. MATLAB의 명령 창 (Command Window)

b. 내부명령 또는 함수들을 사용하여 행렬을 생성

c. M-파일 내에서 행렬을 구성

d. 외부의 자료 파일로부터 행렬을 불러들임

※ MATLAB은 다른 프로그래밍 언어들과 달리 차원의 선언이나 변수 형태의 선언이 필요 없다. 컴퓨터가 사용 가능한 크기까지 자동적으로 저장 공간을 할당해 준다.

- 행렬입력의 규칙

a. 원소들은 빈칸 또는 쉼표를 사용하여 분리한다.

b. 전체 원소들은 대괄호([ ])로 감싼다.

c. 원소의 끝에 세미콜론(;)을 붙이면 한 행의 종료를 의미한다.

ex)
>>A = [16 3 2 13; 5 10 11 8; 9 6 7 12; 4 15 14 1] (엔터)
A =
16 3 2 13
5 10 11 8
9 6 7 12
4 15 14 1 (마방진) Workspace도 확인해본다.

(2) sum, transpose, and diag

- sum

ex)
>>sum(A)
ans = 34 34 34 34 (column 벡터들의 합이 row 벡터의 형태로 구해진다.)

- transpose

ex1)
>>A'
ans =
16 5 9 4
3 10 6 15
2 11 7 14
13 8 12 1

ex2)
>>sum(A')'
ans =
34
34
34
34 (row 벡터들의 합이 column 벡터의 형태로 구해진다.)

- diag

ex1)
>>diag(A)
ans =
16
10
7
1
(＼방향의 대각선의 원소들로 이루어진 column 벡터를 구한다.)

ex2)
>>sum(diag(A)) --> ans = 34

ex3) 반대 방향(/ 방향)의 대각선의 합 구하기
>>sum(diag(fliplr(A))) --> ans = 34
※ fliplr은 행렬의 좌우를 뒤바꾸는 함수이다.

(3) Subscripts

- A(i,j) = A의 row i 와 column j 에 해당하는 원소

ex) A(4,2) = 15. A(1,4) + A(2,4) + A(3,4) + A(4,4) = 34

- Single Subscript A(k) = A의 column vector들을 순서대로 이어서 하나의 column vector로 만들었을 때, k번째에 해당하는 원소

ex) A(8) = A(4,2) = 15.

- Subscripts를 이용하여, 행렬에 새 원소를 추가할 수 있음

```
ex)
>>X = A;
>>X(4,5) = 17
X =
16 3 2 13 0
5 10 11 8 0
9 6 7 12 0
4 15 14 1 17
```

(4) The Colon Operator " : " (Matlab의 가장 중요한 연산자)

- 등차수열을 만들어준다. 물론 벡터 형태로. 초항: 공차: 말항

```
ex) 1:10  -->  1 2 3 4 5 6 7 8 9 10 (row vector)
    100:-7:50  -->  100 93 86 79 72 65 58 51
    0:pi/4:pi  -->  0 0.7854 1.5708 2.3562 3.1416
```

- Subscript 표현식에 응용 가능

```
ex) sum(A(1:4,4)) = 4번째 column의 합.
    sum(A(:,end)) = 마지막 column의 합.
    (':'그 자체는 row 또는 column의 전체 원소를 가리키게 됨.
    end는 마지막 row 또는 column를 가리킴.)
    sum(1:16)/4  -->  34
```

2.2 Expressions

다른 프로그램 언어들과 마찬가지로 Matlab도 변수, 상수, 연산자, 함수와 같은 표현식을 제공한다. 그러나 다른 프로그래밍 언어와는 다르게 행렬과 연관되어 있다.

(1) Variables

- Matlab은 변수 형태의 선언이나 차원 선언을 요구하지 않는다.
- 명령 줄이나 M-files에서 새로운 변수이름을 발견하면 자동적으로 변수가 생성되고 저장 공간이 할당된다. 변수가 이미 존재하면 변수 내용이 바뀐다. 필요하면 새로운 저장 공간도 할당된다.
- 변수 이름은 문자, 숫자(문자 뒤에 와야 함), '_' 가능. 31글자까지 가능. 대문자와 소문자는 다르게 인식함.

(2) Numbers

- 10진법, 소수점 사용, 부호는 숫자 앞에 표시.
- 10의 지수는 e를 사용 ex) 6.02252e23
- 허수는 i or j를 접미사로 사용. ex) -3.14159j, 3e5i
- long 형식, 부동소수점으로 데이터를 저장, 총 16자리까지 표현 가능, 범위는 10^(-308)~10^(308) 까지.

(3) Operators: 다른 언어와 마찬가지로 산술연산자와 그 규칙을 그대로 이용.

+ Addition - Subtraction * Multiplication

/ Division \ (\) Left division

^ Power ' Complex conjugate transpose ( ) Specify evaluation order

(4) Functions

- Matlab은 많은 양의 수학 함수들(abs, sqrt, exp, sin…….)을 제공함.

help elfun, help specfun, help elmat에서 확인 가능.

- 몇몇 함수는 유용한 상수들의 값들을 계산해준다.

pi : 3.14159265……

i, j : Imaginary unit, $\sqrt{-1}$

eps : Floating-point relative precision, 2-52

realmin : Smallest floating-point number, 2-1022

realmax : Largest floating-point number, (2-e)21023

Inf : Infinity (0이 아닌 숫자가 0으로 나누어 졌거나, 값이 realmax 보다 클 때,)

NaN : Not-a-number (0/0 or Inf-Inf 등 값이 수학적으로 정의 되지 않을 때,)

- 함수 이름은 예약어가 아니므로, 사용자가 함수와 똑같은 이름을 가진 변수를 선언해서 사용해도 상관없다.

ex)

```
eps = 1.e-6,
clear eps (clear는 변수를 Workspace에서 제거한다. 즉 다시 원 상태로 되돌림.)
```

2.3 행렬의 생성과 수정

(1) 내부 함수로부터 - 특별한 행렬들

magic: 마방진 ex) magic(4)

zeros: 0만으로 구성된 행렬 ex) zeros(2)(2-by-2 행렬이 됨), zeros(2,4)

ones: 1만으로 구성된 행렬 ex) 5*ones(3,3)

eye: 단위행렬

rand: Uniformly distributed random elements ex) fix(10*rand(1,10))

randn: Normally distributed random elements ex) randn(4,4)

(2) load 명령어

load 명령어는 이전에 Matlab에서 생성된 행렬이 들어 있는 2진 파일들을 읽어 들이거나, 수치 데이터가 들어있는 텍스트 파일을 읽어 들인다. 텍스트 파일을 읽어 들이는 경우, 행렬로 변환할 수 있게 수치들이 직사각형의 표의 형태로 구성 되어있어야 한다. 수치들은 공백(blank)으로 구분되며, 1줄(line)에 1row씩 구성되어있어야 한다. 1줄에 같은 개수의 데이터가 들어가 있어야 한다.

ex)

a. 메모장 같은 프로그램을 이용하여. 다음과 같은 내용을 magik.dat로 저장한다.

16.0 3.0 2.0 13.0

5.0 10.0 11.0 8.0

9.0 6.0 7.0 12.0

4.0 15.0 14.0 1.0

b. Matlab 명령 창에서 load magik.dat --> 파일을 읽은 뒤, magik라는 변수를 생성함.※ load 명령어 대신에 Import Wizard(File -> Import Data)의 도움을 받을 수도 있다.

(3) M-Files

M-File를 이용해서 행렬을 만들 수도 있다.

ex)

a. FIle -> New -> M-Files, 다음과 같은 내용을 magik.m으로 저장

```
A = [ ...
16.0 3.0 2.0 13.0
5.0 10.0 11.0 8.0
9.0 6.0 7.0 12.0
4.0 15.0 14.0 1.0 ];
```

b. Matlab 명령 창에서 magik --> 파일을 읽은 뒤, 변수 A를 생성함.

(4) 행렬의 접합

ex) `B = [A A+32; A+48 A+16]`

```
B =
16 3 2 13 48 35 34 45
5 10 11 8 37 42 43 40
9 6 7 12 41 38 39 44
4 15 14 1 36 47 46 33
64 51 50 61 32 19 18 29
53 58 59 56 21 26 27 24
57 54 55 60 25 22 23 28
52 63 62 49 20 31 30 17
```

(5) 행렬의 원소를 지우는 방법

ex1) row 또는 Column을 지울 때,

```
X = A;
X(:,2) = []
X =
16 2 13
5 11 8
9 7 12
4 14 1   (두 번째 Column이 지워짐)
```

ex2) 원소 하나를 지울 때,

X(1,2) = [] -> error.

single subscript를 이용해서 single element를 지울 수 있다. 이 때, 결과는 row 벡터로 나타남.

X(2:2:10) = [] -> X = 16 9 2 7 13 12 1

(6) 행렬의 row 또는 column을 바꾸는 방법

ex)

```
>>B = magic(4)
B =
16 2 3 13
5 11 10 8
9 7 6 12
4 14 15 1
```

```
>>A = B(:,[1 3 2 4])
A =
16 3 2 13
5 10 11 8
9 6 7 12
4 15 14 1
```

(2번째 column과 3번째 column의 줄이 바뀌게 됨.)

(7) 행렬의 원소 수정

ex)

```
>> B(1,1) = 5
>>B(1:2,2:3) = 0
B =
5 0 0 13
5 0 0 8
9 7 6 12
4 14 15 1
```

2.4 More About Matrices

(1) Linear Algebra

- 행렬은 linear transformation을 표현할 수 있는 2차원 배열.
- 덧셈: A + B, 곱셈: A*B, 지수: A^k
- 나누기

좌측 나누기 : X=A\B -----> A*X=B의 해 (X = inv(A)*B)

우측 나누기 : X=A/B -----> X*B=A의 해 (X = A*inv(B))

- determinant: det(A)
- row echelon form: rref(A)
- inverse matrix: inv(A)
- eigenvalues: eig(A)
- characteristic polynomial: poly(A)

ex) poly(A) = 1 -34 -64 2176 0 -> $\det(A - \lambda I) = \lambda^4 - 34\lambda^3 - 64\lambda^2 + 2176\lambda$

(4) Scalar Expansion

- 행렬에 스칼라 뺀 것은 행렬에 모든 원소에 그 스칼라 값을 뺀 것과 같다.

ex)

```
>> B = A - 8.5 (A는 '(6) 행렬의 row 또는 column을 바꾸는 방법'에서 구한 A)
B =
7.5 -5.5 -6.5 4.5
-3.5 1.5 2.5 -0.5
0.5 -2.5 -1.5 3.5
-4.5 6.5 5.5 -7.5
```

2.5 Command Window의 입력과 출력 제어

(1) format 명령어: format 명령어는 Matlab에서 화면에 출력되는 값들의 표현 형식을 제어한다. 오직 출력되는 형식만 제어할 뿐, 계산하고 저장하는 방식에는 영향을 미치지 않는다. sprintf와 fprintf 함수를 이용하면 출력 형식을 좀 더 자세히 제어할 수 있다.

```
ex) x = [4/3 1.2345e-6]
  format short   -->  1.3333 0.0000
  format short e   -->  1.3333e+000 1.2345e-006
  format short g   -->  1.3333 1.2345e-006
  format long   -->  1.33333333333333 0.00000123450000
  format long e   -->  1.333333333333333e+000 1.234500000000000e-006
  format long g   -->  1.33333333333333 1.2345e-006
  format bank   -->  1.33 0.00
  format rat   -->  4/3 1/810045
  format hex   -->  3ff5555555555555 3eb4b6231abfd271
  format compact   -->  결과 출력 시, 빈 줄을 나타내지 않음.
```

(2) 출력 결과를 나타내고 싶지 않을 때: 명령문을 입력하고, 엔터를 치면 Matlab은 그 계산결과를 자동적으로 화면에 출력한다. 이를 원하지 않으면 문장의 끝에 ';'를 붙인다.

```
ex) A = magic(100);
```

(3) 명령문이 길어 2줄 이상으로 입력해야 할 때: '...'를 입력한 뒤, 엔터를 치면, 커서가 다음 줄로 넘어가 다음 줄에 이어서 입력할 수 있다.

```
ex) s = 1 -1/2 + 1/3 -1/4 + 1/5 - 1/6 + 1/7 ...(여기서 엔터)
    - 1/8 + 1/9 - 1/10 + 1/11 - 1/12;
```

3. MATLAB 그래픽

3.1.서론

MATLAB은 자체로서 수치계산 프로그램이기도 하지만 계산결과나 데이터를 그래픽으로 처리할 수 있는 훌륭한 후처리기(Post-Processing)이기도 하고 다른 프로그램 언어를 사용한 후처리보다 쉽고 강력한 그래픽 기능을 제공한다.

MATLAB에서는 데이터를 디스플레이하기 위해서 line plot, bar, histogram과 같은 그래프나 contour plot, mesh 나 surface plot 그리고 animation 과 같은 고(高)수준의 그래픽 관련 함수를 제공한다. 또한 자체의 MATLAB command 나 그래픽 창의 팝업메뉴를 사용하여 그래픽 객체의 color 나 shading, 그리고 축의 라벨링 등을 제어할 수 있도록 해준다.

MATLAB에서는 C언어에서와 같이 어떤 그래픽 객체를 표현하기 위해서 그래픽 장치를 설정해준다든지 그래픽 시스템을 초기화한다든지 하는 작업을 할 필요가 없다. 사용자는 단지 다른 MATLAB함수나 명령어를 사용하는 것과 마찬가지로 그래픽과 관련된 MATLAB 함수를 적절한 입력 값으로 사용하면 된다.

3.2. 2차원 그래픽

3.2.1. 2차원 그래픽을 그리는 절차

(1) plot 할 데이터를 준비한다.
(2) 데이터를 plot 할 그림 창을 선택하고 single graphic으로 할건지 Multi graphic으로 할 건지 선택한다.
(3) plot 함수를 도출한다.
(4) Line 이나 marker 의 property를 선택한다.
(5) 축의 한계값, Grid line 등을 설정한다.
(6) xlabel(축), legend(삽입문, 범례), text(제목 및 부가설명) 등으로 그래픽 객체에 라벨링을 한다.
(7) 그래픽 객체를 출력한다.

ex) 0≦x≦10 에 대해서 $y = \sin(x)\cos(x+\frac{\pi}{2})$ 를 2차원 그래프에 그려보자.

(1) plot 할 데이터를 준비한다.

```
>>x=0:0.01:10;
>>y=sin(x).*cos(x+pi/2);
>>size(x)
ans =
          1        1001
>>size(y)
ans =
           1        1001
```

(2) 데이터를 plot 할 Figure 창을 선택하고 Single graphic으로 할 건지 Multi graphic으로 할 건지를 선택한다.

```
>> figure(1)
```

(3) plot 함수를 호출한다.

```
>> plot(x,y);
```

(4) Line 이나 marker 의 property 를 선택한다.

```
>> set(plot(x,y),'LineStyle,'--')
```

(5) 축의 한계값, Grid line 등을 설정한다.

-MATLAB에서 축의 한계 값은 그래프에 그릴 데이터의 크기에 맞도록 자동으로 조정된다. 만약 자동으로 설정된 축의 한계 값을 바꾸고 싶을 때는 axis를 사용한다. 또한 그래픽 객체에 격자(grid)를 그리거나 지우고 싶으면 grid in/off 를 사용한다.

```
>> axis([1,5,-0.5,0])
>> grid on
```

(6) xlabel, legend, text 등으로 그래픽 객체에 라벨링을 한다.

```
>> xlabel('x');
>> ylabel('y');
>> title('plotting x and y');
```

(7) 그래픽 객체를 출력한다.

- 출력의 방법에는 2가지가 있다. 첫째는 비트맵과 같은 그래픽 파일 형식으로 그래픽 객체를 저장. 둘째는 프린터를 사용하여 그래픽 객체를 직접 인쇄.

```
>> print -dbitmap ex1.bmp        % bitmap format 을 사용하여
>> dir ex1.bmp                  % "ex1.bmp"라는 파일명으로 저장
ex1.bmp                         % "ex1.bmp"라는 파일이 생성되었음
```

3.2.2. 새로운 그림 창의 생성과 Multi-graphic 객체

(1) figure 함수의 용도

a. 새로운 그림 창을 생성.

```
>> figure         % Figure No.1 을 생성
>> figure         % Figure No.2 을 생성
>> figure(10)     % Figure No.10 을 생성
```

b. 이미 존재하는 그림 창을 활성화.

```
>>figure(2)        % Figure No.2 그림창이 활성화 되면서 앞으로 수행될 그래픽 관련된 작업은 다른 그림
                   창이 활성화되기 전까지는 모두 이 그림 창에서 이루어지게 된다.
```

(2) Multi-graphic 객체

하나의 그림 창안에 다수의 그래프를 그리고 싶을 때 subplot을 사용하면 된다. 즉, 하나의 그림 창을 하부의 영역으로 나누어 주는 작업을 수행하는 함수가 subplot이다.

`subplot(m,n,i)` 또는 `subplot(mni)`

는 하나의 그림 창을 m x n 의 하부 영역으로 나눈다. 여기에서 i 는 m x n 의 영역으로 나누어진 하부 영역의 순서를 나타내는 것으로 위에서 아래, 왼쪽에서 오른쪽으로 지정된다.

(3) 이미 존재하는 그래프에 새로운 그래프를 추가
hold 명령어를 사용하면 기존의 그래프에 추가로 입력한 수식의 그래프가 도시된다.

3.2.3. 기본적인 그래픽 함수들과 plot 의 제어

(1) 기본적인 그래픽 함수

함 수	설 명
plot	x와 y 축에 대해서 모두 선형 배율(linear scale)로 된 그래프를 그린다.
loglog	x와 y축에 대해서 모두 log 배율로 된 그래프를 그린다.
semilogx	x축에 대해서는 log배율로 y축에 대해서는 선형 배율로 된 그래프를 그린다.
semilogy	x축에 대해서는 선형 배율로 y축에 대해서는 log 배율로 된 그래프를 그린다.
plotyy	y축의 좌우에 값으로 라벨이 된 그래프를 그린다.

(2) plot 제어
위의 도표에서 plot을 포함한 이하 함수들도 마찬가지로 같은 방법으로 제어한다.

```
plot(x,y,'Color+LineStyle+Marker')
```

<가능한 선의 color>

MATLAB에서의 symbol	Color(RGB값)	MATLAB에서의 symbol	Color(RGB값)
c	Cyan(011)	g	green(010)
m	Magenta(101)	b	blue(001)
y	yellow(110)	w	white(111)
r	red(100)	k	black(000)

<가능한 선의 style>

MATLAB에서의 symbol	Style	MATLAB에서의 symbol	Style
-	Solid line	:	Dotted line
--	Dashed line	-.	Dash-dot line
none	No line		

<가능한 선의 marker>

MATLAB에서의 symbol	Marker Style	MATLAB에서의 symbol	Marker Style
+ (plus sign)	+	^ (Upward pointing triangle)	^
o (Circle)	o	v (Downward pointing triangle)	▽
* (asterisk)	*	>	▷
. (point)	●	<	◁
X (cross)	x	pentagram	☆
Square (square)	□	hexagram	*
Diamond (Diamond)	◇	none	No marker

```
ex) plot(x,y,'b-')
```

3.2.4. 복소수 데이터 처리

MATLAB에서 제공되는 복소수 관련 함수는 isreal, real, imag, conj, abs, angle 등이 있다.

- isreal은 주어진 입력 값이 복소수인지 실수인지를 결정해준다. 실수는 1, 복소수는 0으로 반환한다.
- real과 imag는 각각 주어진 복소수 입력 값에서 실수부만을 반환하고 허수부만을 반환한다.
- abs와 angle은 각각 복소수의 크기와 각도를 계산한다. 단, **각도의 단위는 라디안**이며 부호는 1, 2 사분면에서 존재할 경우 양의 각으로, 3, 4 분면에서는 음의 각으로 계산된다.

```
ex)
   >> a= 1+2i;
    >>isreal(a)
   ans =
       0
```

```
>> b= 2;
 >> isreal(b)
 ans =
     1
```

3.2.5. 축의 한계 값 조정

축의 최대 최소값은 기본적으로 데이터의 최대 최소값에 따른다. 이러한 축의 범위를 바꾸고 싶으면 axis 명령어를 사용하면 된다.

```
axis(v)
```

v=[x축의 최소값, x축의 최대값, y축의 최소값, y축의 최대값]의 행벡터이다.

```
ex) >>x=0:0.1:10;
    >>y=sin(x);
    >>plot(x,y);
    >>v=[0,6,-0.2,0.6];
    >>axis(v)
```

3.2.6. 그래픽 객체의 라벨링(annotation)

<그래픽 객체의 라벨링 관련 함수>

함 수 명	설 명
title	그래프의 제목을 첨가
xlabel	x축에 제목 첨가
ylabel	y축에 제목 첨가
zlabel	z축에 제목 첨가
legend	그래프에 범례 첨가
text	그래픽 객체의 임의의 위치에 문자열 첨가
gtext	마우스를 사용하여 첨가 문자열 위치지정
grid	그래픽 객체에 격자 첨가 혹은 제거

```
ex) >>t=0:0.1:50;
    >>x1=sin(t+pi/2);
    >>x2=exp(-0.1*t).*sin(t);
    >>plot(t,x1,'b-',t,x2,'r--')
    >>xlabel('time(sec)');
    >>ylabel('Displacement(m)');
    >>title('Free&Forced Vibration');
```

legend('문자열1',’문자열2',...,’문자열n',정수)

정수는 그래프 내에서 범례가 더해지는 위치를 결정하는 것으로 0은 그려진 데이터를 고려해서 MATLAB에서 결정한 최적의 위치를, 1은 오른쪽 위를, 2는 왼쪽 위를, 3은 왼쪽 아래를, 4는 오른쪽 아래를, -1은 그래프 영역 오른쪽 밖을, 없으면 default의 위치를 나타낸다.

text는 함수를 호출할 때 문자열과 그 문자열이 더해질 위치를 같이 입력하고, gtext는 문자열만 입력하고 그래프창이 활성화 되면서 그 문자열이 더해질 위치는 마우스를 사용해서 결정한다.

text(x좌표,y좌표,'문자열')

\leftarrow 와 \rightarrow 는 문자열에 화살표를 추가해주는 기능을 한다.

4. SYSTEM CONTROL ANALYSIS

4.1 시스템제어에 활용되는 함수

4.1.1 num, den, tf

라플라스 전달함수 h(s)에서 num과 den은 각각 분자(numerator) n(s)와 분모(denominator) d(s)의 s 변수에 대한 다항식의 계수를 대표하는 함수이다.

$$h(s) = \frac{n(s)}{d(s)}$$

그리고 tf는 num, den 변수들을 가지고 라플라스 전달함수 h(s)를 구성하는 함수이다.

ex) $h(s) = \frac{s}{s^2 + 2s + 10}$

```
>> num=[1 0];
>> den=[1 2 10];
>> h=tf(num,den)

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

4.1.2 residue

residue란 부분분수와 다항식의 계수전환을 가능케 하는 함수이다. 기본 표현은 다음과 같다.

```
[r,p,k]=residue(b,a)
[b,a]=residue(r,p,k)
```

b,a	s에 대한 내림차수 다항식의 분자, 분모계수
r	나머지의 계수
p	극점의 계수
k	분수가 아닌 다항식의 계수

$$\frac{b(s)}{a(s)} = \frac{b_1 s^m + b_2 s^{m-1} + b_3 s^{m-2} + \ldots + b_{m+1}}{a_1 s^n + a_2 s^{n-1} + a_3 s^{n-2} + \ldots + a_{n+1}}$$

$$\frac{b(s)}{a(s)} = \frac{r_1}{s - p_1} + \frac{r_2}{s - p_2} + \ldots + \frac{r_n}{s - p_n} + k(s)$$

ex) 다음과 같은 함수에 대하여 residue 함수를 적용해 보자.

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

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

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

한편 residue를 역으로 [a,b]에 대하여 [r,p,k]를 가지고 실행하면 [b,a]가 분모의 최고차항을 1로 하는 기준화를 시키기 때문에 계수가 바뀌게 된다. 하지만 결과는 당연히 동일한 식이다.

ex)

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

```
b =                                          a =
   -1.2500   -0.7500    0.5000   -1.7500          1.0000   -0.0000   -2.0000   -0.7500
```

4.1.3 다양한 함수들(step, impulse, lsim, ...)

응답함수에는 다양한 형태의 입력을 반영하는 함수들이 개별적으로 존재한다. 그 중 제어에서 가장 흔하게 사용하는 형태는 스텝 함수(step), 임펄스 함수(impulse)가 있다. 이것은 구성한 시스템에 대하여 설정한 응답에 대한 거동을 살피기 위한 함수로서 사용법은 다음 예제와 같이 사용한다. 그리고 설계한 시스템에 대한 거동을 시뮬레이션으로 살펴보기 위해 시뮬레이션 함수(lsim)를 사용한다.

ex1) step function

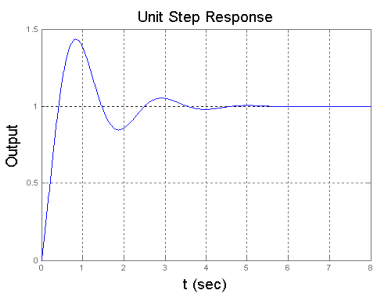
```
t=0:0.01:8;
num=[2 10];
den=[1 2 10];
sys=tf(num,den);
step(sys,t)
```

ex2) impulse function

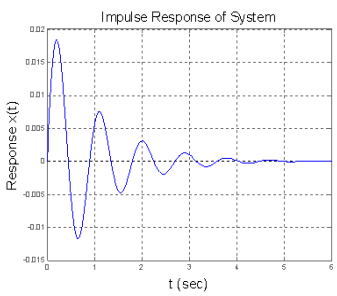
```
num=[7.9984];
den=[50.01 100 2500];
sys=tf(num,den);
impulse(sys)
```

ex3) lsim function

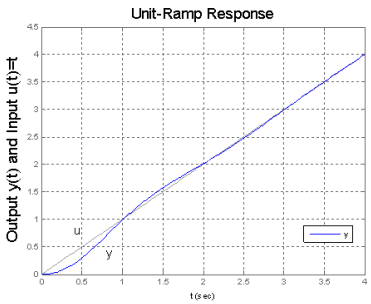
```
num=[2 10];
den=[1 2 10];
sys=tf(num,den);
t=0:0.01:4;
u=t;
lsim(sys,u,t)
```



ex1 step function



ex2 impulse function



ex3 simulation function