1장 MATLAB 입문

1. 기본 연산

가. 스칼라, 벡터, 행렬 입력

1. 스칼라: 예) a = 5
   - 상수: **pi**(원주율), **i ,j**(복소수), **inf**(무한대), **NaN**(Not a Number), **ans**(최근 계산값), **eps**(부동소수점의 상대적 정확도), **flops**(부동 소수점 연산 수행 횟수), **realmax**(가장 큰 부동 소수점), **realmin**(가장 작은 양의 부동 소수점), **clock**(현재 시간), **date**(날짜), **cputime**(matlab 동작 시간)

2. 행렬: 행구분 - 세미콜론(;), 열구분 – 콤마(,) 또는 공백
   - 특수 행렬: [] (빈행렬), **zeros**(m,n) (영행렬), **ones**(m,n) (1행렬), **eye**(m,n) (단위행렬), **rand**(m,n) (균등분포 난수행렬), **randn**(m,n) (정규분포 난수행렬), **pascal**(n) (파스칼 삼각행렬), **magic**(n) (마방진), **compan**(P) (동반행렬), **hadamard**(n) (Hadamard 행렬)

3. 벡터: [] 또는 콜론(:)을 사용하여 입력
   예) 행 벡터: a=[1,2,3] , 열 벡터: b=[1;2;3] , 콜론사용: c=시작:증가:끝 ex> c=1:1:10;

나. 스칼라, 벡터, 행렬 연산 및 조작

1. 스칼라: 덧셈(+), 뺄셈(-), 곱셈(*), 좌측나누기(₩), 우측나누기(/), 지수(^)

2. 행렬: - 덧셈과 뺄셈은 스칼라 연산과 같고, 곱셈과 나눗셈 연산에서는 행, 열의 개수를 고려해야함. 예> [3x4]*[4x3])
   [참고] 나눗셈 연산은 역행렬(**inv**())의 곱과 같음.
   - **rot90**(A,k) (행렬을 반시계 방향으로 90도 회전, k가 양수이면 반시계 방향), **flipud**(A) (행렬을 상하 교환), **filplr**(A) (행렬을 좌우 교환), **reshape**(A,m,n) (A 행렬의 요수로 m,n 행렬을 만듦)

3. 배열: 곱셈이나 나눗셈, 지수연산에서 점(.)을 이용하여 배열 연산
   [참고] 벡터의 내적, 외적 – 내적: sum(A.*B) 또는 dot(A,B) , 외적: cross(A,B)

Tip.1 기본적인 수학, 삼각, 쌍곡선 함수 정리

* 수학함수: abs(x), sqrt(x), round(x), fix(x), floor(x), ceil(x), sign(x), rem(x,y), exp(x)
* 삼각함수: sin(x), cos(x), tan(x), asin(x), acos(x), atan(x), atan2(x,y)
* 쌍곡선함수: sinh(x), cosh(x), tanh(x), asinh(x), acosh(x), atanh(x)

Tip.2 명령어

* clc(command prompt 내용 지우기), clear (workspace 지우기, 예> clear 변수, clear all)
  who(사용한 변수의 목록 출력), whos(사용한 변수의 목록 상세 출력)

2. 그래픽

가. 2차원 그래픽

1. 선 그리기

ㄱ. **plot**: x, y 축에 대한 선형 배율 그래프를 그림, 3차원의 경우 plot3(x,y,z)를 이용.

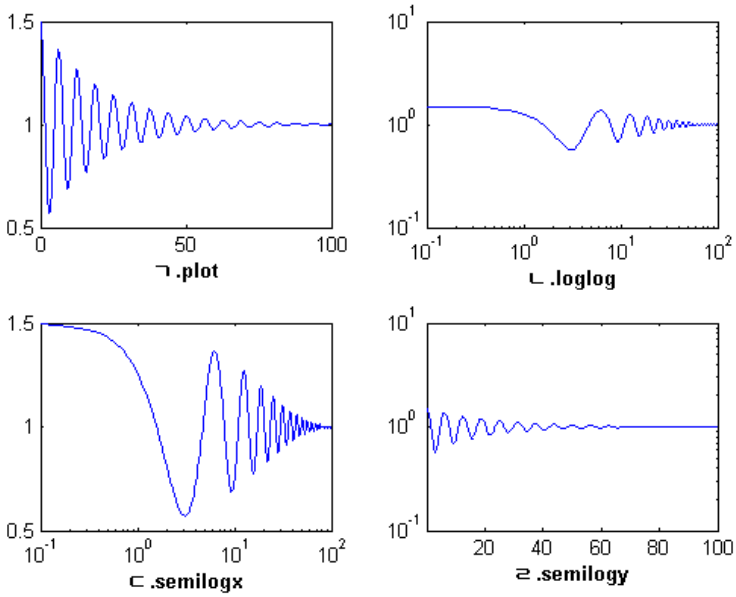
ㄴ. **loglog**: x, y 축에 대한 log 배율 그래프를 그림

ㄷ. **semilogx**: x 축에 대해서 log 배율, y축은 선형 배율 그래프를 그림

ㄹ. **semilogy**: y 축에 대해서 log 배율, x축은 선형 배율 그래프를 그림

ㅁ. **plotyy**: y축에 대해 양쪽으로 구분하여 2개의 그래프를 동시에 그림

ㅂ. **polar**: 극 좌표계 그래프 그리기, polar(각도,반지름,'색상+스타일+표시')



2. 선 꾸미기: **plot**(x,y, '색상+스타일+표시')

ㄱ. 색: c (Cyan), m (Magenta), y (Yellow), r (Red), g (Green), b (Blue), w (White), k (Black)

ㄴ. 스타일: - (Solid line), -- (Dashed line), : (Dotted line), -. (Dash-dot line), none (no line)

ㄷ. 표시: +, o, *, ., X, ^, v, >, <, Square, Diamond, pentagram, hexagram, none

예) **plot**(x,y,'b-+'), **plot**(a,b,'r--diamond'), **plot**(x,y1,'b-',x,y2,'g-.o')

3. 그래프 서식(축, 눈금, 제목 등)

ㄱ. 축값 지정: **axis**([x축 최소값, x축 최대값, y축 최소값, y축 최대값]),
axis on, **axis off**를 통해 보임 설정

ㄴ. 보조 눈금: **set**(gca,'xtick',v) 또는 **set**(gca,'ytick',v'), v 값은 표시하는 눈금 값 행 벡터

ㄷ. 제목 설정: **title**('그래프 제목'), **xlabel**('x축 제목'), **ylabel**('y축 제목'), **zlabel**('z축 제목')

ㄹ. 범례 설정: **legend**('범례1','범례2',...,정수) – 정수의 값은 위치를 나타냄 1(오른쪽 위), 2(왼쪽 위), 3(왼쪽 아래), 4(오른쪽 아래), -1(그래프 영역 밖), 0(최적위치)

ㅁ. 문자 추가: **text**('x좌표','y좌표','문자열') – 임의의 문자열 그래프에 추가
gtext('문자열') – 문자열의 위치를 마우스 선택으로 지정

ㅂ. 격자: **grid on**(격자 표시), **grid off**(격자 숨김)

4. 좌표계 변환
   [각도, 반지름] = **cart2pol**(x좌표, y좌표)
   [각도, 반지름, 높이] = **cart2pol**(x좌표, y좌표, z좌표)
   [x좌표, y좌표] = **pol2cart**(각도, 반지름)
   [x좌표, y좌표, z좌표] = **pol2cart**(각도, 반지름, 높이)
   [pi,theta,반지름] = **cart2sph**(x좌표, y좌표, z좌표)
   [x좌표, y좌표, z좌표] = **sph2cart**(pi, theta, 반지름)

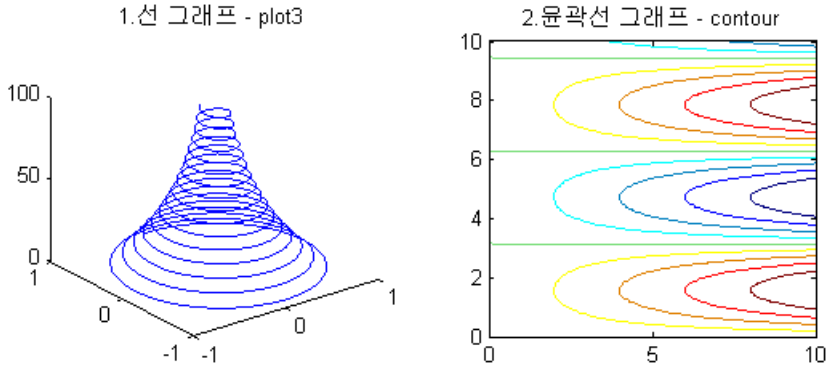
5. 마우스를 이용한 좌표 입력
   [x좌표, y좌표] = **ginput**(N) – N은 마우스로 선택할 좌표의 개수 지정

[참고] **subplot**(m,n,i) or **subplot**(mni) : 하나의 그림 창 안에 다수의 그래프를 그림
예) subplot(221) : 그림 창을 2행 2열로 나누고 그 중에 첫 번째 부분 활성화
[참고] **hold** : 기존의 그림을 지우지 않고 유지시킨 상태에서 그래프를 추가로 그림
hold on, hold off 명령을 사용

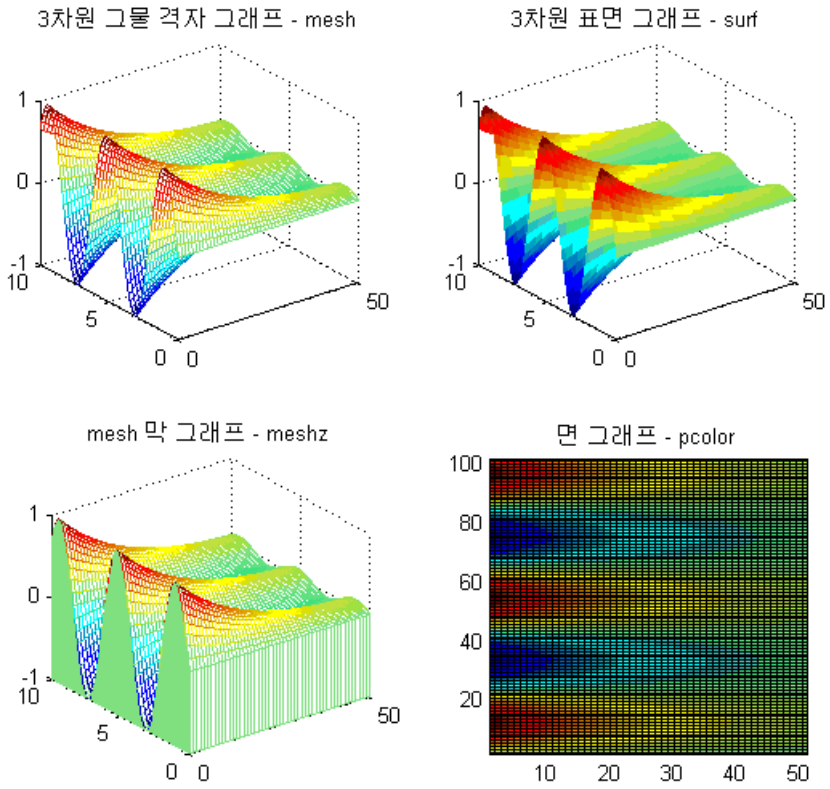
Tip. 데이터를 입력할 때 명령어의 마지막에 세미콜론(;)을 적어주면 입력한 데이터 값을 창에 보여주지 않고 바로 입력하게 됩니다. 그래서 매우 많은 양의 데이터를 넣을 경우 prompt 창이 갑자기 위로 올라가버리는 일이 생기지 않습니다.

나. 3차원 그래픽

1. 선 그래프: **plot3**(x,y,z,'색상+스타일+표시')
2. 윤곽선 그래프: 2차원-**contour**(x,y,z,N), 3차원-**contour3**(x,y,z,v)



3. 면 그래프:(1) **mesh**(x,y,z,C) – 3차원 그물 격자 그래프
   (2) **surf**(x,y,z,C) – 3차원 표면 그래프, 표면의 색은 shading 옵션을 이용.
   (3) **meshc** – mesh와 contour 그래프를 동시에 그림
   (4) **surfc** – surf와 contour 그래프를 동시에 그림
   (5) **meshz** – mesh에 막이 포함된 그래프
   (6) **pcolor**(z) – 값의 크기를 색으로 표현하는 면 그래프
   (7) **surfl**(z) – 임의의 광원이 존재하는 표면 그래프



4. 특별한 그래프

(1) 막대 및 면적 그래프: **bar**(x,y,'선색'), **barh**(x,y,'선색'), **bar3**(x,y,'선색'), **bar3h**(x,y,'선색')
area(x,y) – 면적 그래프, (막대 그래프에서 사용할 수 있는 변수 'group', 'stack')

(2) 원형 그래프: **pie**(y,explode), **pie3**(y), explode는 원의 일부를 분리시키기 위한 벡터 값

(3) 히스토그램: **hist**(y) – 직각 좌표계, **rose**(theta) – 각 좌표계

(4) 이산 데이터 그래프: **stem**(x,y,'선색+스타일+표시'), **stem3**(x,y,z,'선색+스타일+표시')
staris(x,y) – 계단 그래프, (선의 끝에 원을 그리려면 'fill' 변수 추가)

(5) 방향 및 속도 벡터 그래프:

- **comp**(x,y) 또는 **comp**(z): 극 좌표계의 원점에서 발산하는 그래프
- **feather**(Z): 수평선을 따라 등 간격으로 분포된 점들로부터 발산되는 벡터 표시
- **quiver**(x,y,u,v), **quiver3**(x,y,z,u,v,w,s) : 주어진 점들에서 시작하는 벡터 그래프

5. 애니메이션

(1) movie 이용

```
axis equal
M = moviein(n);
for j=1:n
 plot_command
 M(:,j) = getframe;
end
Movie(M)
```

(2) redraw, erase 이용

p=erase_mode가 사용된 plot_command 예)

```
hold on
axis([축값]) %축 고정
for j=1:n
 set(p,'XData',x,'YData',y...)
 drawnow
end
```

[참고] **colorbar**('vert') 또는 **colorbar**('horiz') – 그래프에 색의 크기를 표시하는 막대를 표시함

colormap(A) – 그래프에 사용된 색을 선택, 3개의 열을 갖는 행렬로 입력하거나 변수 지정. (변수: hsv, hot, cool, pink, copper, flag, gray, bone 등)

[참고] **light**('Color',[색상 행렬]), **light**('Position',[x좌표, y좌표, z좌표]) – 광원 색과 위치 지정

hidden on/off – 숨은선 표시 또는 표시 안 함

view(AZ,EL) – 방위각(AZ), 고도(EL)에서 그래프를 관찰. 예) view(45,45), view(2), view(3)

Tip. 두 점을 잇는 선 그리려면 **line**(Ax,Ay) 또는 **line3**(Ax,Ay,Az) 명령을 사용해서 그릴 수 있다.

3. Matlab 프로그래밍

가. 연산자

1. 산술 연산자: +(덧셈), -(뺄셈), *(곱셈), .*(요소 곱셈), /(우측 나눗셈) , ./(요소 나눗셈), ₩(좌측 나눗셈), .₩(좌측 요소 나눗셈), ^(거듭제곱), .^(요소 거듭제곱)
2. 관계 연산자: <(작다), <=(작거나 같다), >(크다), >=(크거나 같다), ==(같다), ~=(같지 않다)
3. 논리 연산자: &(논리곱), |(논리합), ~(부정), xor, any, all, exist
4. 비트 연산자: **bitand**(A,B) (비트 논리곱), **bitcmp**(A,N) (비트 보수), **bitor**(A,B) (비트 논리합), **bitxor**(A,B) (비트 xor), **bitset**(A,BIT) (비트 설정), **bitget**(A, BIT) (비트값 얻기), **bitshift**(A, N) (비트 이동), **bitmax**(최대 부동정수)

   Tip.1 십진수와 이진수 사이의 변환: **bin2dec**('이진수'), **dec2bin**(십진수)
5. 기타 연산자: **infinite**(A) (무한인지 판단, 유한 1), **isinf**(A) (무한인지 판단, 무한 1), **isnan**(NaN인지 판단), **isequal**(A,B) (서로 같은지 판단), **isnumeric**(수인지 아닌지 판단), **isreal**, **~isreal**(실수인지 판단)

나. 제어문

1. 택일문: if, elseif, else, switch, case
2. 반복문: while, for
3. 분기문: break, return

다. 함수

1. 문자열 관련: abs(S), double(S), isstr(S), strcmp(S1,S2), upper(S), lower(S), [S1,S2], setstr(A), char(A), ischar(S), num2str(A), int2str(A), str2double(S). (문자열은 작은 따옴표)

   Tip.2 문자열로 이루어진 수식을 연산하려면 eval('문자열')를 이용할 수 있음.
   M-file 함수 이름이 문자열로 되어 있을 때는 feval('answkduf',x1,x2...,xn)을 사용함.
2. 메뉴와 dialog: **menu**('제목','메뉴1','메뉴2',...), **msgbox**('메시지','제목','아이콘'), **inputdlg**('prompt','제목'), **questdlg**('메시지','제목','버튼1','버튼2',...'기본값'), **helpdlg**('메시지','제목'), **errordlg**('메시지','제목'), **warndlg**('메시지','제목'), **printdlg**(그림번호), **printdlg**('-setup',그림번호), **pagedlg** - 출력 용지 설정, **uigetfile**('*.확장자','제목'), **uiputfile**('*.확장자','제목'), **uisetfont**(문자열 변수,'제목'), **uisetcolor**('제목')
3. 파일 처리: **fopen**('파일명','권한') – (권한: r, w, a, r+, w+, a+, W, A), **fclose**(FileID), **fscanf**(FileID,'format',size), **fprintf**(FileID,'format',인자1, 인자2...), **ferror**(FileID), **fseek**(FileID,Offset,Origin), **ftell**(FileID), **fgets**(FileID), **feof**(FileID), **frewind**(FileID),
4. 기타: **error**('메시지') (메시지 출력 후 실행 종료), **warning**('메시지') (메시지 출력 후 진행) **disp**(A) (변수명은 출력하지 않고 값만 출력), **input**('메시지') (키보드로 입력을 받음)

[참고] m-file의 내용을 공개하지 않으려면 "**pcode** 파일명" 명령을 이용해서 코드를 바이너리로 변환하여 보호할 수 있다.

2장 MATLAB 수치해석 및 수학계산

1. Symbolic 계산

가. 함수

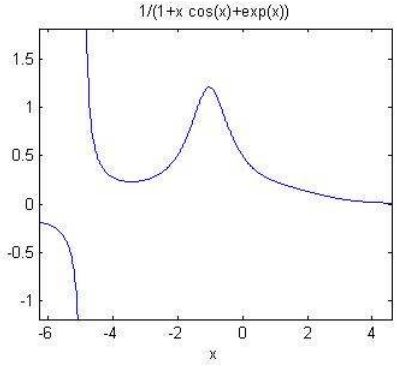
1. 계산: diff(미분), int(적분), limit(극한값), taylor(Taylor급수), jacobian(Jacobian 행렬), symsum(합)
2. 선형 대수: inv(역행렬), det(행렬식), rank(행렬의 계수), null(null space를 위한 basis를 계산), rref(행렬을 reduced row echelon form으로 만듦), eig(고유치와 고유벡터)
3. 간략화: simplify(수식의 간략화), expand(수식의 전개), simple(Symbolic으로 된 수식의 가장 간단한 형태를 찾음), subs(Symbolic으로 된 수식의 변수 갱신)
4. 방정식의 해: solve(대수 방정식), dsolve(미분 방정식), finverse(역함수)
5. 적분변환: fourier, laplace, ztrans, ifourier, ilaplace, iztrans
6. 데이터형 변환: double(Symbolic 데이터를 수치 데이터로 변환), char(Symbolic 데이터를 문자열로 변환)
7. 기초 연산: sym(symbolic 객체 생성), syms(Symbolic 객체 생성 단축형), findsym(Symbolic 객체 인지를 조사), pretty(Symbolic 수식을 보기 좋은 형태로 출력), latex(Symbolic 수식을 LaTex로 표현), ccode(C언어로 표현), fortran(Fortran언어로 표현)

나. 생성과 연산: a=**sym**('x') 또는 **syms** a 명령을 통해 symbolic a을 생성하고, 위 함수를 포함한 사칙연산을 수행할 수 있음. [참고] 수식 전개 함수: **expand**(f)

다. 그래프: ezplot(f) 또는 **ezplot**(f,[a b])

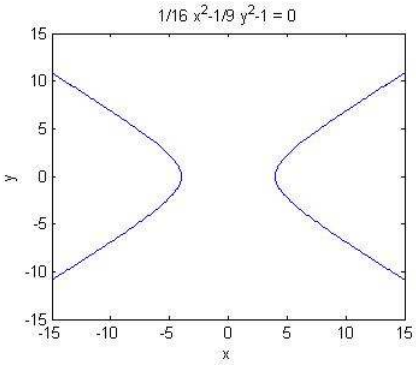
예제 1)

```
>> syms x
>> f=1/(1+x*cos(x)+exp(x));
>> ezplot(f)
```



예제 2)

```
>> syms x y
>> f=x^2/16 – y^2/9-1;
>> ezplot(f,[-15 15])
```



라. Symbolic를 수치 데이터로 변환

1. subs: symbolic 변수 대신에 수치를 대입하여 수치 데이터로 변환
   예) syms x; A=cos(x); x=30*pi/180; subs(A);
2. double: symbolic 변수가 없는 symbolic 데이터를 수치 데이터로 변환
   예) syms x; f=cos(x)-0.5; solve(f,x); double(ans);

2. 방정식의 해 구하기

가. fzero – 1변수 방정식

1. 함수: **fzero**('함수',x0) 또는 **fzero**('함수',x0,오차). 방정식을 m-file로 작성 후 명령 실행
2. 예제

<m-file 내용>

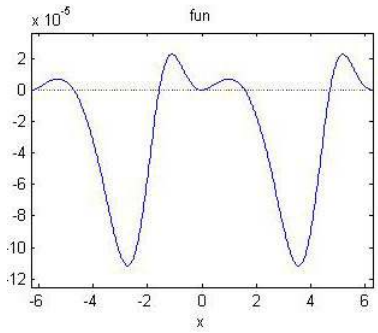
```
function f=fun(x)
 f=(sin(x)^2+cos(x)-1)/(12+2*sin(x))^4;
```

>> fzero('fun',2) %초기값 2부터 해를 구함

결과값: x = 1.5708

>> fzero('fun',4) %초기값 4부터 해를 구함

결과값: x= 4.7124



나. fsolve – 비선형 연립방정식

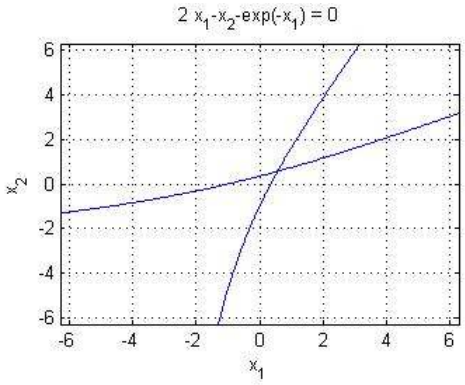
1. 함수: **fsolve**('함수',x0) 또는 **fsolve**('함수',x0,options). 방정식을 m-file로 작성 후 명령 실행
2. 예제

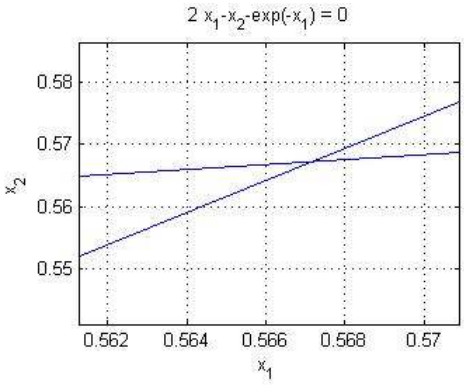
<m-file 내용>

```
function f=funs(x)
 f=[2*x(1)-x(2)-exp(-x(1));-x(1)+2*x(2)-exp(-x(2))];
```

>> x=fsolve('funs',[-5;-5]) %초기값 x(1) -5, x(2) -5부터 처음 0이 되는 점의 값을 구함

결과값: x = [0.5671; 0.5671]





다. solve – 비선형 방정식

1. 변수 비선형 방정식 – x=**solve**(s)

예) >> syms x a b c

```
>> f=a*x^2+b*x+c;
>> x=solve(f)
x = -1/2*(b-(b^2-4*a*c)^(1/2))/a
     -1/2*(b+(b^2-4*a*c)^(1/2))/a
>> pretty(x)
```

2. 비선형 연립 방정식 – [x1,x2,...,xn]=**fsolve**(f1,f2,...fn)

예) >> syms x y

>> f1 = x^2+y^2-4;

>> f2 = exp(x)+y-1;

>> [x,y]=solve('f1','f2');

>> double(x);

ans = -1.8163

>> double(y);

ans = 0.8374

그래프 그리기)

>> ezplot(f1,[-3,3])

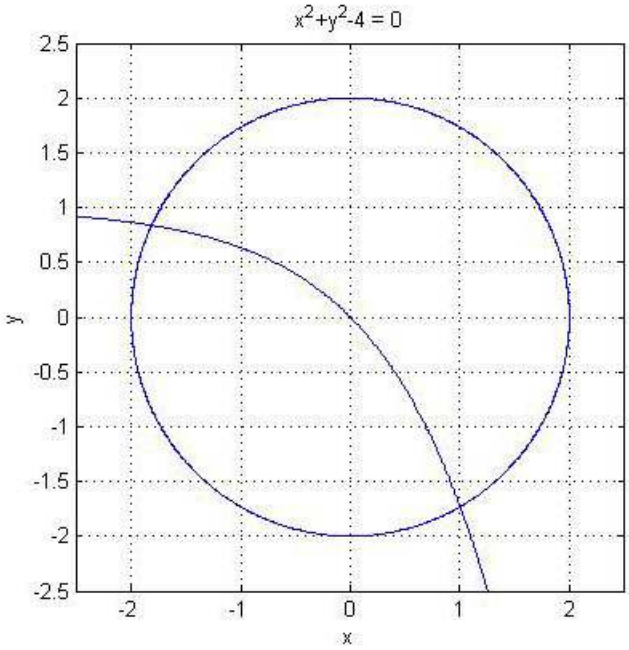
>> hold on

>> ezplot(f2,[-3,3])

>> axis equal

>> axis([-2,5 2.5 -2,5 2.5])

>> grid on



3. 행렬 연산

가. 전치 행렬 – **transpose**(A) 또는 A', (**ctranspose**: 켤레 복소수의 전치 행렬)

예) >> A=[1,2,3;4,5,6;7,8,9];

>> transpose(A)

ans = [1,4,7;2,5,8;3,6,9]

나. 역행렬 – **inv**(A)

다. 행렬식 – **det**(A)

예) >> syms k1, k2, k3

>> k = [(k1+k2+k3),-(k2+k3);-k3,(k2+k3)]

k =

[k1+k2+k3, -k2-k3]

[-k3, k2+k3]

>> det(k)

ans = k1*k2+k1*k3+k2^2+k2*k3

라. 계수 – **rank**(A) : 원래 행렬의 행이나 열의 수에서 reduced row echelon form 행렬에서 행의 요소가 모두 0인 행의 수를 뺀 값. (**rref** : reduced row echelon form 행렬 생성)

예) >> a=[1,-2,3;2,-5,1;1,-4,-7]

>> rank(a)

ans = 2

>> rref(a)

ans = [1,0,13;0,1,5;0,0,0] %rref에서 행의 요소가 모두 0인 행은 3행 뿐임.

마. trace – **trace**(A) : 행렬의 대각 요소의 합

바. 대각 행렬 – **diag**(A) : 대각 성분을 제외한 모든 요소가 0인 행렬

사. 삼각 인수 분해

1. [L,U,P] = **lu**(A). (L: 하삼각 행렬, U: 상삼각 행렬, P:순열 행렬), PA = LU
2. [Lp U] = **lu**(A). (Lp: permuted 하삼각 행렬, U: 상삼각 행렬), Lp = inv(P)*L

아. 직교 인수 분해 – [Q,R,E] = **qr**(A) (Q: unitary 행렬, R: 상삼각 행렬, E: 순열행렬), AE = QR

자. Cholesky 인수 분해 – [R,p,S] = **chol**(A) (R: 상삼각 행렬)

차. 고유치와 고유 벡터 – [X,lamda] = **eig**(A), (X: 고유벡터, lamda: 고유치)

- A*X = lamda*X, X가 0이 아닌 해를 갖기 위한 lamda 값을 고유치, X를 고유벡터라 함.
- (A-lamda*I)X = 0, det(A-lamda*I) = 0, (**poly**: 행렬의 특성 방정식)

카. 놈(norm)

1. 벡터 norm – norm(x,p): sum(abs(x).^p)^(1/p)와 같음. norm(x,inf) = max(abs(x))
2. 행렬 norm – norm(A,1): 열의 절대값의 합 중 가장 큰 값, max(sum(abs(A)))와 같음.
   norm(A,inf): 행의 절대값의 합 중 가장 큰 값, max(sum(abs(A')))
   norm(A,'fro'): 모든 요소의 제곱의 합에 대한 제곱근, sqrt(sum(diag(A'*A)))
   norm(A) 또는 norm(A,2): 행렬 A의 가장 큰 singular value, max(svd(A))

4. 선형 연립 방정식의 해

가. 역행렬을 이용한 선형 연립 방정식의 해

예) 3a+2b-c=10, -a+3b+3c=5, a-b-c=-1 의 해를 구하기

```
>>A=[3,2,-1;-1,3,3;1,-1,-1];
>>B=[10;5;-1];
>>X=inv(A)*B
X=1.000; 3.000; -1.000 % a, b, c
```

나. 행렬 나누기를 이용한 선형 연립 방정식의 해

예)
```
>>A=[3,2,-1;-1,3,3;1,-1,-1];
>>B=[10;5;-1];
>>X=A₩B
X=1.000; 3.000; -1.000 % a, b, c
```

다. 행렬의 삼각 분해를 이용한 선형 연립 방정식의 해

예)
```
>>[Lp,U]=lu(A);
>>Y=inv(Lp)*B;
>>X=inv(U)*Y;
X=1.000; 3.000; -1.000 % a, b, c
```

5. 보간법 및 회귀 분석

가. 선형 보간법 - **interp1**(x,y,x0), **interp2**(x,y,z,x0,y0)

예)
```
>>x=[40 48 56 64 72 80];
>>y=[1.33 1.67, 2.08 2.36 2.71 3.19];
>>plot(x,y,'o-')
>>interp1(x,y,52)      %x값이 52일 때 y에 해당하는 값을 구함
ans = 1.8750
```

나. Cubic spline 보간법 - **spline**(x,y,x0)

예)
```
>>x=[40 48 56 64 72 80];
>>y=[1.33 1.67, 2.08 2.36 2.71 3.19];
>>x1=40:0.5:80;
>>y1=spline(x,y,x1);
>>plot(x,y,'o',x1,y1)
>>spline(x,y,52)      %x값이 52일 때 y에 해당하는 값을 구함
ans = 1.8860
```

다. 최소 자승법에 의한 곡선의 근사 - **polyfit**(x,y,1)

예) >>x=[40 48 56 64 72 80];

>>y=[1.33 1.67, 2.08 2.36 2.71 3.19];

>>P=polyfit(x,y,1);

>>plot(x,y,'o',x,P(1).*x+P(2))

라. 다항식을 사용한 회귀 곡선 - **polyfit**(x,y,N), **polyval**(P,x)

예) >>x=[40 48 56 64 72 80];

>>y=[1.33 1.67, 2.08 2.36 2.71 3.19];

>>x1=40:0.5:80;

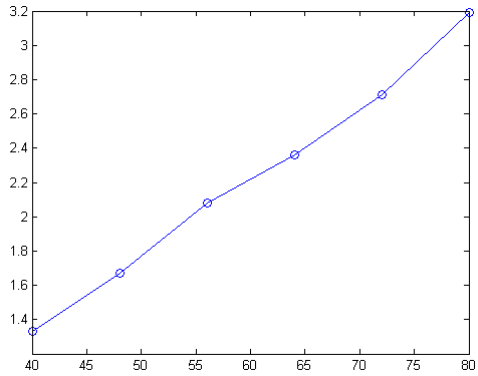
>>P1=polyfit(x,y,2); %2차 다항식을 이용

>>y1=polyval(P1,x1);

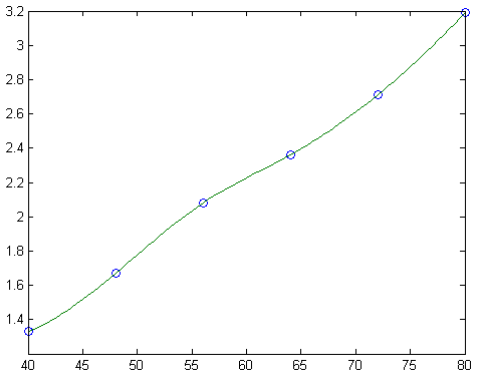
>>P2=polyfit(x,y,5); %5차 다항식을 이용

>>y2=polyval(P2,x1);

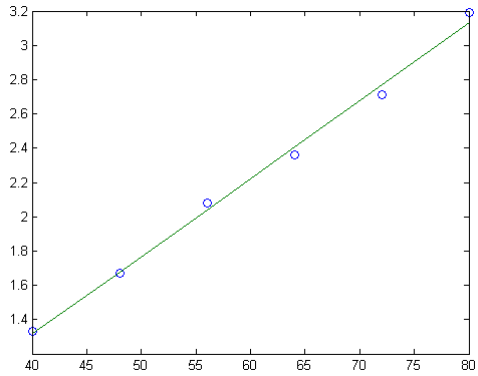
>>plot(x,y,'o',x1,y1,x1,y2)



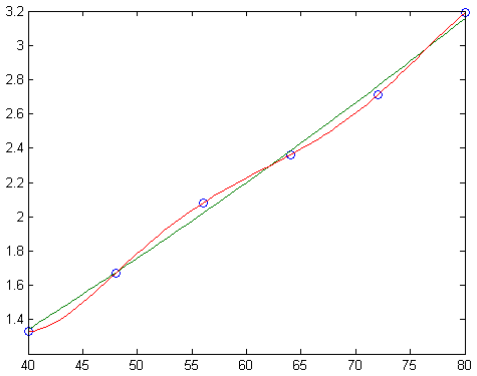
<interp1을 이용한 그래프>



<spline을 이용한 그래프>



<polyfit을 이용한 그래프>



<polyfit, polyeval을 이용한 그래프>

6. 다항식

가. 다항식 값 구하기 – **polyval**(배열연산), **polyvalm**(행렬연산)

예) >>func=[1,-3,2,1];
>>x=10;
>>y=polyval(func,x) %x^3-3*x^2+2*x+1
y= 721
>>x2=[5,6,7];
>>y2=polyval(func,x2)
y2= 61, 121, 211
>>x3=[1,2,3;4,5,6;7,8,9];
>>y3=polyval(func,x3)
y3= 1 1 7
25 61 121
211 337 505
>>y4=polyvalm(func,x3) %Matrix must be square
y4= 381 472 564
872 1073 1272
1364 1672 1981

나. 다항식의 산술 연산 -> + : 덧셈, - : 뺄셈, **conv**(a,b) : 곱셈, **deconv**(a,b) : 나눗셈,

[R,P,K] = **residue**(f,g): 부분 분수 전개(R:나머지, P:극, K항)

예) >>a=[1,2,3];
>>b=[4,5,6];
>>c= conv(a,b)
c= 4 13 28 27 18
>>deconv(c,a)
ans= 4, 5, 6

예) >>f=[1,5,9,7]; % x^3+5*x^2+9*x+7
>>g=[1,3,2]; % x^2+3*x+2
>>[R,P,K]=residue(f,g)
R= -1 ; 2
P= -2 ; -1
K= 1 2 % -1/(x-(-2)) + 2/(x-(-1)) + x + 2

다. 다항식의 해 구하기– **roots**(coeff)

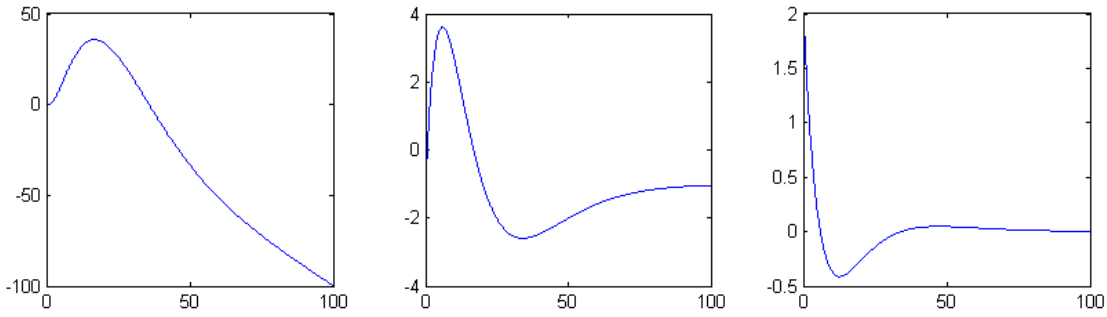
예) >>func=[1,-3,-1,3];
>>p=roots(func)
p= 3.0000 ; 1.0000 ; -1.0000
예) >>func=[1,-2,-3,4];
>>s=roots(func)
s= -1.6506
-0.1747 + 1.5469i
-0.1747 - 1.5469i

7. 수치 미분과 적분

가. 미분 – **diff**(x)

예) >>x=0:0.1:100;

```
>>y=x.^2.*exp(-0.1*x)-x;
>>subplot(1,3,1)
>>plot(x,y)                  % x^2 그래프
>>dy=diff(y)./diff(x);       % 차분법에 의한 미분
>>dx=x(2:length(x));         % 차분법에 의해 x의 개수가 1개 줄어듦
>>subplot(1,3,2)
>>plot(dx,dy)                % 1차 미분 그래프
>>ddy=diff(dy)./diff(dx);    % 2차 미분
>>ddx=dx(2:length(dx));      % dx보다 개수가 1개 줄어듦
>>subplot(1,3,3)
>>plot(ddx,ddy)              % 2차 미분 그래프
```



[참고] 차분법에 의한 미분은 데이터 사이 간격이 충분히 조밀해야 오차가 작으므로, 데이터 간격이 클 경우 보간법을 이용한 후 미분법 사용.

나. 적분 – **quad**('M-file',a,b,tol): simpson's rule, **quadl**('M-file',a,b,tol): Lobatto quadrature

int(f,'x'): Symbolic 객체를 이용한 수식 적분, f를 x로 적분

예제 1) <m-file 내용>

```
function y=func(x)
y=1./(1+x).^2;
```

```
>>quad('func',0,10)
ans = 0.909091053721440
>>quadl('func',0,10)
ans = 0.909090909256147
```

예제 2)

```
>>syms x                  % x를 symbolic로 지정
>>f=1/((1+x)^2);          % 함수식
>>int(f,'x')              % symbolic 함수 적분
ans = -1/(1+x)
>>double(int(f,'x',0,10))  % 구간 0~10 에서 적분하고 값을 실수 변환
ans = 0.909090909090909
```

8. 라플라스 변환

가. 라플라스 변환 – **laplace**(F)

예) >>syms t a b

>>F=t^2+sin(a*t)-t*cos(b*t);

>>L=laplace(F)

L= 2/s^3+a/(s^2+a^2)-(s^2-b^2)/(s^2+b^2)^2

>>pretty(L)

```
                                    2      2
          2          a          s   - b
        ---- + ------- - ----------
          3       2     2      2     2 2
        s       s   + a     (s   + b )
```

나. 역라플라스 변환 – **ilaplace**(F)

예) >>syms s a b

>>L=2/s^3+a/(s^2+a^2)-(s^2-b^2)/(s^2+b^2)^2

>>I=ilaplace(L)

I= t^2+sin(a*t)-t*cos(b*t)

9. 상미분 방정식

가. 함수

- **ode23**: 2차 또는 3차의 Runge-Kutta method
  (ODE23 Solve non-stiff differential equations, low order method)
- **ode45**: 4차 또는 5차의 Runge-Kutta method
  (ODE45 Solve non-stiff differential equations, medium order method)
- **사용법**: ode23('M-file',time,Initial_Condition,Options), ode45('M-file',time,Initial_Condition,Options)

나. 예제 - y''=y'(1-y^2)-y의 해 구하기

<m-file 내용>

```
function U_prime=U_func5(x,u)
U_prime=zeros(2,1);                  % 변수를 0으로 초기화
U_prime(1)=u(1)*(1-u(2)^2)-u(2); % 상태 변수를 사용한 1차 미분 방정식
U_prime(2)=u(1);
```

```
>>time=[0,30];                        % 시작과 끝점
>>Initial_Condition=[0,0.1];          % 초기조건 y(0)=0.1, y'(0)=0
>>[x,Y]=ode45('U_func5',time,Initial_Condition);   % Y의 1열은 y', 2열은 y를 의미
>>dy=Y(:,1); y=Y(:,2);
>>subplot(211); plot(x,dy); xlabel('x'); ylabel('dy')
>>subplot(212); plot(x,y); xlabel('x'); ylabel('y')
```

10. 데이터 분석

가. 데이터 분석 함수

- max(x): 최대값, min(x): 최소값, mean(x): 평균값, median(x): 중간값
- sum(x): 합, prod(x): 누적곱, cumsum(x): 누적합, cumprod(x): 누적곱
- sort(x): 정렬, var(x): 분산, std(x): 표준편차

나. 퓨리에 변환 – **fft**(x)

- Matlab 내의 doc fft 내용

```
>>Fs = 1000;                    % Sampling frequency
>>T = 1/Fs;                     % Sample time
>>L = 1000;                     % Length of signal
>>t = (0:L-1)*T;                % Time vector
% Sum of a 50 Hz sinusoid and a 120 Hz sinusoid
>>x = 0.7*sin(2*pi*50*t) + sin(2*pi*120*t);
>>y = x + 2*randn(size(t));     % Sinusoids plus noise
>>subplot(121)
>>plot(Fs*t(1:50),y(1:50))
>>title('Signal Corrupted with Zero-Mean Random Noise')
>>xlabel('time (milliseconds)')

>>NFFT = 2^nextpow2(L); % Next power of 2 from length of y
>>Y = fft(y,NFFT)/L;
>>f = Fs/2*linspace(0,1,NFFT/2);

% Plot single-sided amplitude spectrum.
>>subplot(122)
>>plot(f,2*abs(Y(1:NFFT/2)))
>>title('Single-Sided Amplitude Spectrum of y(t)')
>>xlabel('Frequency (Hz)')
>>ylabel('|Y(f)|')
```