MATLAB과 과학계산

고려대학교 수학과

2013-08-05

차 례

1 장

MATLAB 계산

제 1 절 입력과 계산

수 입력은 Command Window와 Editor를 이용할 수 있다. MATLAB을 이용하여 계산할 것이 복잡하거나 알고리듬이 긴 경우 Editor를 사용한다. 다음은 Command Window로 계산한 간단한 예이다.

```
>> a = 1; b=2;c=a+b
c= 3
```

이 출력된다.
곱하기와 나눗셈, n제곱, n제곱근은 다음과 같이 계산할 수 있다.

```
>> a = 1; b=2;n=a*b,d=1/b,e=b^n,f=sqrt(b),g=b^(1/n)


n = 2
d = 0.5000
e = 4
f = 1.414
g = 1.414
```

명령문 뒤 ;은 계산된 값을 화면에 출력하지 않고 MATLAB안의 메모리(Workspace)에 저장만 할 때 쓴다. 또한 한 줄에 여러가지 출력 명령어를 쓸 때 ,를 이용한다.

로그 함수는 밑이 자연상수 e로 두고 계산한다.

```
>> a=log(10);b=log(20);
>> c=a+b

c  =  6.9078
```

밑을 2나 10으로 할 때는 log(2)나 log(10)으로 나누어 주거나 MATLAB 명령문 log2와 log10을 쓴다.

```
>> a =log(10)/log(10);b=log(100)/log(10);
>> a1=log10(10);b1=log10(100);c=log2(16);
>> c,d=a+b,d1=a1+b1

c  =  3
d  =  3
d1 =  3
```

자연상수는 exp이며 π는 pi 그리고 어떤 수 (x)에 대한 수학 함수는 cos(x), sin(x), tan(x), cot(x), asin(x), acos(x), atan(x), acot(x), gamma(n+1)=$n \times (n-1) \cdots 2 \times 1$이 있다.

그 외 명령문

clear는 MATLAB에 저장된 변수를 삭제할 때 쓰인다.

변수를 선택적으로 삭제할 수 있는데, 이경우 **clearvars**를 쓴다. 예를 들면

```
>>clearvars a* -except ab
>>clearvars -global -except x*
```

라고 입력하면 ab를 제외한 a로 시작하는 모든 변수는 삭제되게 된다. 두 번째 명령문의 경우는 X로 시작하는 변수를 제외하고 모든 변수는 삭제된다.

clc를 하면 출력 화면이 지워진다.

whos는 MATLAB 메모리에 저장된 변수의 차원과 크기를 보여준다.

주석으로 쓰기 위한 문장 앞에 % 두면 MATLAB은 실행 시 % 뒤에서 그 줄 끝부분까지 읽지 않게 된다.

제 2 절 벡터 계산

MATLAB을 이용하여 1에서 10까지 1만큼 증가하는 벡터 $v1$을 만들어 보자.

```
>> n=10;v1=1:n

v1=  1     2     3     4     5     6     7     8     9    10
```

증분을 2로 한다면 $v1 = 1 : 2 : n$이라고 쓰자.

```
>> n=10;v1=1:2:n

v1=  1     3     5     7     9
```

이 된다. 만약 10에서 1까지 -2만큼 감소하게 만들고 싶다면 $n = 10; v1 = n : -2 : 1$라고 쓰면 된다. **logspace**를 쓰면 로그 크기 만큼 벡터를 만든다.

```
>> format rat
>> logv1=logspace(-1,4,6)

logv1 = 1/10     1      10      100     1000     10000
```

벡터 곱은 **dot** 명령어를 쓰거나 간단히 벡터간의 곱으로 나타낼 수 있다.

```
>> v1 = [1; 2; 3];
>> v2 = [4  5  6];
>> v3=v1*v2,v4=dot(v1,v2)

v3 = 4     5     6
     8    10    12
    12    15    18

v4 = 32
```

$v3$는 차원 $v1(3 \times 1)$와 $v2(1 \times 3)$의 곱이므로 3×3인 행렬이 출력되었고 $v4$는 벡터곱의 명령어를 썼기 때문에 스칼라 값이 출력되었다. v 벡터의 크기는

$$\|v\| = \sqrt{(v,v)} = \sqrt{\sum_{k=1}^{n} v_k^2} \tag{1.1}$$

norm(v)으로 구한다.

```
>> v1 = [1; 2; 3];n1=norm(v1)

n1 = 3.7417
```

주어진 수의 평균을 구할 는 **mean**을 쓴다.

```
>> e1 = [1 2 3 4 5 6 7 8 9 10];s1=mean(e1)

s1 = 5.5000
```

또한 행렬에서 서로 다른 원소를 찾기 위해서는

```
>> A = [1 1 7;0 2 0;0 5 3];
>> uA = unique(A)

uA = 0
     1
     2
     3
     5
     7
```

unique는 원소 크기대로 재배열해 준다.

벡터곱은 **cross** 명령어로 구할 수 있다.

```
>> v1=[1 2 3];v2=[3 2 1];v3=cross(v1,v2)

v3 = -4 8 -4
```

제 3 절 행렬 계산

행렬 A, B의 계산해 보자.

```
>> A = [1 0 0;0 2 0;0 0 3];
>> B = [2 1 0;0 3 1;0 0 4];
>> A+B

ans = 3 1 0
      0 5 1
      0 0 7
```

행렬 A를 입력 시 다음 행을 입력할 때는 ;을 입력한다. $A + B$의 결과값을 입력할 변수를 정의하지 않았을 때 MATLAB은 그 값을 ans에 입력하게 된다. 곱하기도 위와 같이 입력한 후 실행하면 된다.

```
>> C=A*B

C  = 2 1 0
     0 6 2
     0 0 12
```

행렬 안의 원소끼리 더하거나 곱할 때는 행과 열로 지정하여 계산한다.

```
>> A = [1 1 0;0 2 0;0 0 3];
>> B = [2 1 4;0 3 1;0 0 4];
>> c1 =A(1,2)+B(1,3)

c1 =  5
```

콜론 :을 쓰면 행이나 열 전체를 가르킬 수 있다.

```
>> A = [1 1 0;0 2 0;0 0 3];
>> B = [2 1 4;0 3 1;0 0 4];
>> v1 =A(2,:),v2=B(:,3)'
```

```
>> v3 =v1+v2

v1 = 0  2  0
v2 = 4  1  4
v3 = 0  2  4
```

A에서는 두 번째 행이 B에서는 세 번째 열이 각각 v1과 v2에 입력되어 계산되었다. MATLAB에서는 행렬의 Block끼리의 모을 수 있다.

```
>> A = [1 0 0;0 2 0;0 0 3];
>> B = [2 1 0;0 3 1;0 0 4];
>> C = [1 2 3;4 5 6;7 8 9];
>> D = [1 0 0;0 1 0;0 0 1];
>> E = [A B; C D]

E =  1     0     0     2     1     0
     0     2     0     0     3     1
     0     0     3     0     0     4
     1     2     3     1     0     0
     4     5     6     0     1     0
     7     8     9     0     0     1
```

그러면 E행렬의 (1,1)은 A행렬, (1,2)는 B행렬, (2,1)은 C 그리고 (2,2)의 행렬은 D이다.

이제 벡터와 행렬을 곱해 보자.

```
>> v1=[1 2 3];A=[1 0 0;0 2 0;0 0 3];v2=A*v1'

v2 = 1
     4
     9
```

행렬과 벡터의 차원을 맞추기 위해 v1에 '을 하여 전치(transpose)를 하였다. 계산하고자 하는 행렬이 대각행렬 (diagonal matrix)이면 **diag**을 이용하여 행렬을

쉽게 만들 수 있다. 예를 들어 x1=[1 2 3]이라고 하면

```
>> A = diga(x1)

A = 1 0 0
    0 2 0
    0 0 3
```

이 출력된다. 또한 대각을 중심으로 그 아래와 위의 값도 입력할 수 있다. 즉,

```
>> A=[1 0 0;0 2 0;0 0 3];

A = 0 0 0 0
    1 0 0 0
    0 2 0 0
    0 0 3 0
>> A = diga(x1,1)

A = 0 1 0 0
    0 0 2 0
    0 0 0 3
    0 0 0 0
```

이 출력된다. 벡터나 행렬곱이 아니라 행렬 A와 B의 각각의 원소에 곱을 할 때는 점 .을 써서 계산한다.

```
>> A=[1 2 3;0 2 0;0 0 3];B=[1 2 3;2 2 2;3 3 3];C=A.*B

A = 1 4 9
    0 4 0
    0 0 9
```

이 출력된다.

sum을 이용하면 행렬의 열방향의 원소를 다 더하게 된다.

```
>> A  = [1 1 2;0 2 0;0 0 3];
>> sA = sum(A)

sA= 1 3 5
```

행렬의 크기가 $m \times n$인 원소가 모두 영의 행렬 A는 **zeros**(m, n),

```
>> m=3;n=2;A=zeros(m,n)

A =  0     0
     0     0
     0     0
```

원소가 모두 1인 행렬은 **ones**(m, n)를 쓴다.

```
>> m=3;n=2;A=ones(m,n)

A =  1     1     1
     1     1     1
```

대각(diagonal)이 1인 I 단위행렬은 **eye**(m, n)

```
>> m=3;n=2;A=eye(m,n)

A =  1     0     0
     0     1     0
```

명령어로 만든다. 행렬 A가 정방행렬일 경우 역행렬과 행렬식은 **inv**(A)과 **det**(A)로 구한다.

```
>> A=[1 2;3 4];
>> inA=inv(A),d=det(A);
inA = -2.0000    1.0000
       1.5000   -0.5000
d   = -2
```

행렬의 rank를 확인하고자 할 때는 **rank**를 쓴다.

```
>> A = [3 2 -2; -3 -1 3; 1 2 0];
>> rank_A = rank(A)

rank_A = 3
```

MATLAB에서는 eigenvalue를 구할 수도 있는데 **eig**(**A**)를 사용하면 쉽게 구할 수 있다.

```
>> A=[1 2;3 4];
>> lambda=eig(A);

lambda =
   -0.3723
    5.3723
```

또한 $det(A - \lambda I)$는

```
>>  A = [3 2 -2; -3 -1 3; 1 2 0];
>>  lambda = roots(poly(A))

lambda =
   -1.0000
    2.0000
    1.0000
```

roots(poly(A))를 이용하여 구한다. 그리고 주어진 λ에 대해 **rref**를 쓰면 eigenvectors가 계산된다.

```
>>  A = [3 2 -2; -3 -1 3; 1 2 0];
>>  eignv1 = rref(A-2*eye(3))

eignv1 =
```

```
      1       0       0
      0       1      -1
      0       0       0
```

임의의 수를 가지는 난수 행렬은 **rand**$(\mathbf{m}, \mathbf{n})$ 균등분포 (uniform distribution, open interval (0,1))를 사용한다.

```
>>  m=3;n=2;R1=rand(3,2)


R1 =
    0.8147    0.9134
    0.9058    0.6324
    0.1270    0.0975
```

randn$(\mathbf{m}, \mathbf{n})$을 사용하면 평균이 0 표준편차가 1인 정규분포의 행렬이 만들어 진다.

제 4 절 실수에서의 계산

분수 형식으로 표현하기 **format rat**.

```
>> format rat
>> a= 1/2,b=0.5


a    = 1/2
b    = 1/2
```

소수 넷째자리까지 표현하기 **format short**.

```
>> format short
>> a= 1/3


a = 0.3333
```

소수 열다섯째자리까지 표현하기 **format long**.

```
>> format long
>> a= 1/3

a    =  0.333333333333333
```

MATLAB에서의 실수 값 중 가장 큰 값인 $MAX = 1.797693134862315 \times 10^{+308}$이 넘을 경우 **Inf**으로 표시된다. 또한 0/0이나 inf / inf인 경우를 계산할 때 결과 값은 not a number **NaN**이 출력된다. 계산이나 데이터를 다루다 보면 NaN이 있는지 확인할 때가 있다. 그 때에는 **isnan**을 쓴다. 그러면 그 해당하는 행렬의 값이 NaN이면 1을 출력하게 된다. 수가 존재하지 않는 경우를 확인할 시에는 **isempty**으로 확인하면 된다. 예를 들어 집합 A에 아무런 수가 들어가지 않았다면 isempty(A)는 1을 출력하게 된다.

제 5 절 허수에서의 계산

```
>>z1= 1+4*j+2+9*i
 3.0000 +13.0000i
```

MATLAB에서는 알파벳 i나 j에 대해 이미 정의되지 않았다면 자동으로 복소수로 인식하게 된다.

위 식을 i, j를 사용하지 않고 쓸 수도 있다.

```
>>x1= 1;y1=4;x2=2;y2=9;
>>z1=complex(x1,y1);z2=complex(x2,y2);
>>z3=z1+z2
z3 = 3.0000 +13.0000i
```

복소수의 크기와 각은 **abs**와 **angle** 그리고 **compass**를 이용하면 구할 수 있다.

```
>>z1=complex(sqrt(2)/2,sqrt(2)/2);
>>ab=abs(z1),an=angle(z1)*180/pi,compass(z1)
```

```
ab = 1
an = 0.7071
```

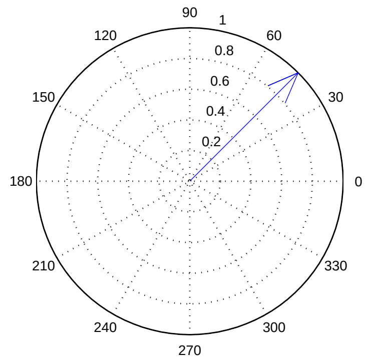


그림 1.1: compass 실행 시 출력되는 복소수의 크기와 각

복소수 값의 켤레(conjugate)는 **conj**($z1$), 실수는 **real**($z1$), 허수는 **imag**($z1$) 입력하면 나온다.

```
>>z1=complex(sqrt(2)/2,sqrt(2)/2);
>>z1c=conj(z1),x1=real(z1),y1=imag(z1)

z1c = 0.7071 - 0.7071i
x1  = 0.7071
y1  = 0.7071
```

제 6 절 함수 계산

MATLAB에서 함수의 근을 구할 수 있다. 다음과 같은 함수가 있다고 하자.

$$f(x) = x^2 - x - 6$$

그러면 근은 $x = -2$와 $x = 3$으로 구할 수 있다. MATLAB의 fzero을 이용하여 근을 구해보자.

```
>> func=@(x)[x^2-x-6]; x0=-3;
>> z1=fzero(func,x0)
```

그러면 MATLAB은 func에 $x^2 - x - 6$의 함수를 x에 대해 입력하게 되고 fzero은 x0의 근처에 가까이 있는 근 하나를 출력하게 된다. x0의 상수값 대신에 범위값 x0=[0 4]을 넣으면 그 범위 안에 근이 하나일 때 그 근을 출력하게 된다.
이번에는 함수의 값을 구해 보자. MATLAB에서 주어진 inline함수를 사용하여 그 값을 구할 수도 있고 간단히 @로 그 함수 값을 구할 수도 있다. 다음을 보자.

```
>> f0=(@(x)(cos(x)));f1=inline('cos(x)');
>> f0(pi),f1(pi)
ans =
     -1
ans =
     -1
```

그러면 그 함수 f0와 f1에 대해서 맞는 값이 나오는 것을 확인할 수 있다.

2장

MATLAB graph

제 1 절 2차 그래프 그리기

간단히 $y = x^2$의 그래프를 그려보자. 컴퓨터는 연속적인 곡선을 인식할 수 없으므로 곡선을 n등분하여 곡선을 그리게 된다. Editor에 아래와 같이 명령문을 써보자.

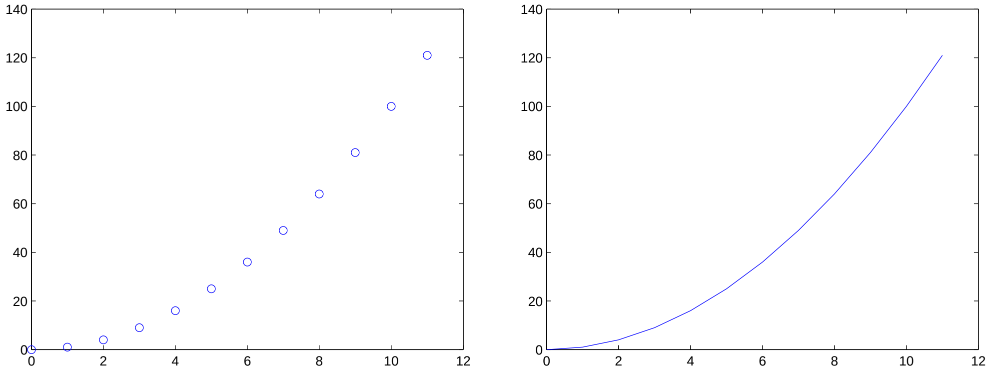


그림 2.1: plot

```
clc;clear;
n=12;
```

```
for i=1:n
x(i)=i-1;
end
y=x.^2
figure(1),
plot(x,y,'o')
figure(2),
plot(x,y)
```

위 코드에서 for 문을 쓰지 않고 x=linspace(0,11,n)로 써도 동일한 결과를 얻는다. **linspace**는 시작점을 1 그리고 끝점을 11으로 하여 n구간만큼 나누어서 벡터값으로 반환한다. 실제로는 그림1같이 점들로 그려지지만 MATLAB에서 자동으로 곡선으로 그려준다.
이번에는 닫힌 곡선(closed curve)의 그림을 그려보자. 이번에도 plot을 이용하면 그릴 수 있다. 다만 안과 밖을 구별하려고 할 때는 fill을 쓰도록 한다.

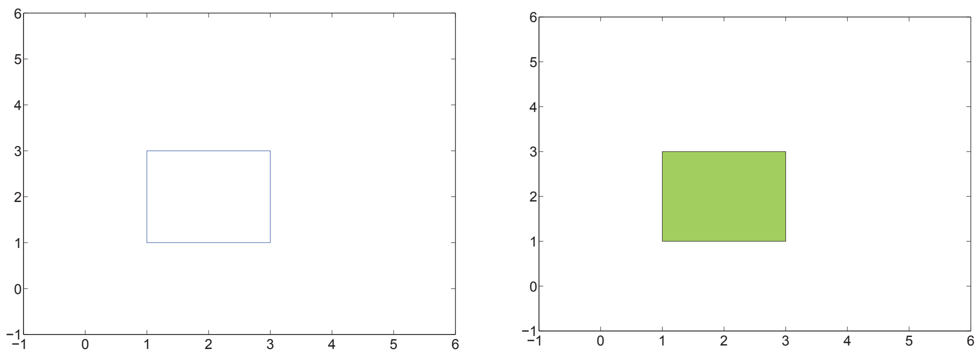


그림 2.2: plot과 fill로 닫힌 곡선 그리기

```
clc;clear;
x=[1 2 3 3 3 2 1 1 1];
y=[1 1 1 2 3 3 3 2 1];
figure(1),
plot(x,y)
```

```
axis ([-1 6 -1 6])
figure(2),
fill(x,y,1)
axis ([-1 6 -1 6])
```

clc와 clear은 스크립트(script)를 초기화하기 위함이고 위와 같이 x와 y의 닫힌 좌표가 있다고 있다고 할 때 plot을 사용하면 직선을 그려주고 **fill**을 사용하면 닫힌 곡선의 색을 채워준다. 여기에서는 axis ([-1 6 -1 6])을 사용하여 출력하는 그림의 범위를 지정해 주었다.

직교좌교가 아니라 극좌표의 식을 그릴 때에는 **polar**를 사용하면 쉽게 그릴 수 있다. 다음 그림은 $r = \theta$의 식을 $0 \le \theta \le 4\pi$까지 그린 것이다.

```
theta = 0 : 0.05 : 4 * pi;
        r = 2*sin(theta);
        polar ( theta, r )
```

배경이 원이 아닌 사각형을 원한다면 **pol2cart**와 plot을 이용하면 된다.

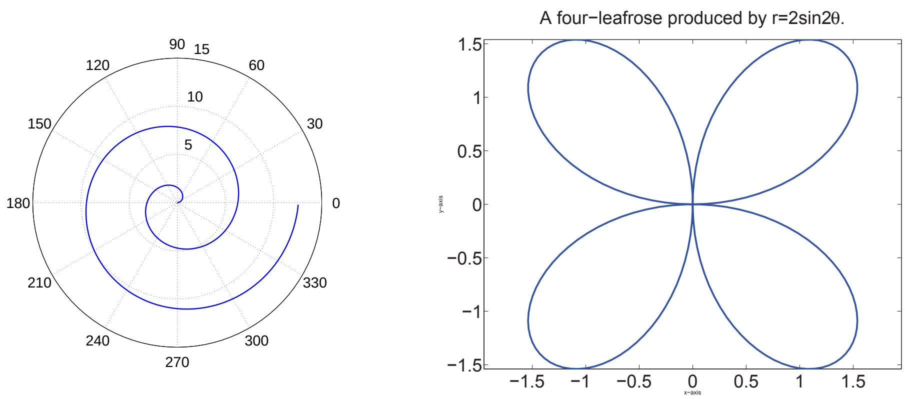


그림 2.3: polar와 pol2cart를 이용한 그리기

```
theta=linspace(0,2*pi,200);
r=2*sin(2*theta);
[x,y]=pol2cart(theta,r);
```

```
plot(x,y)
axis equal
xlabel('x-axis'),ylabel('y-axis')
title('A four-leafrose produced by r=2sin 2 theta.')
```

또한 그래프를 그리기 위해 식을 정리할 때 한 변수(x나 y)에 대해 정리하기가 어려울 경우가 있다. MATLAB을 통해 그릴 때도 이런 문제가 있을 수 있다. 이런 경우 **ezplot**를 쓰면 편리하게 그릴 수 있다. 아래와 같은 식이 있다고 하자.

$$(x^2 + y^2)^2 - (x^2 - y^2) = 0$$

위와 같은 식을 그리기 위해 ezplot을 실행해 보자.

```
>> ezplot('(2*x^2+y^2)^2-(x^2-y^2)',[-1,1,-0.5,0.5])
```

그러면 그림이 출력된다.

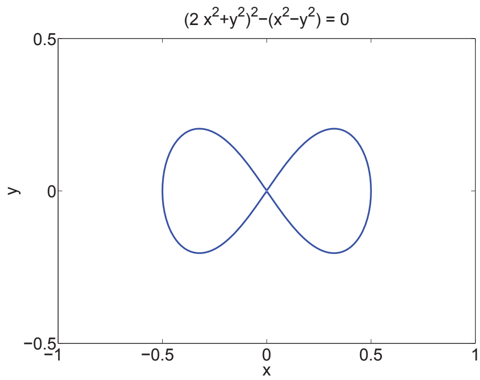


그림 2.4: ezplot를 이용한 그리기 x와 y의 범위는 $[-1\ 1] \times [-0.5\ 0.5]$이다.

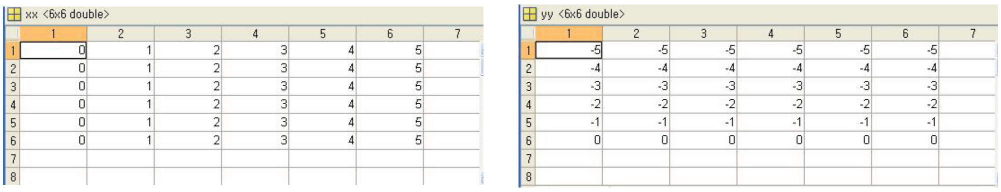
제 2 절 3차 그래프 그리기

2.1 3차 surface 그리기

meshgird는 x축과 y축의 벡터 $x1$와 $y1$를 묶어 3차원 그래프를 그리도록 행렬 정의역을 만들어 준다.

- 3차원 그래프를 만들기 위해 정의역 xx, yy 만들자.

```
>> x1 = [0 1 2 3 4 5]; y1=[-5 -4 -3 -2 -1 0];
>> [xx yy]=meshgrid(x1,y1);
```



xx <6x6 double>

	1	2	3	4	5	6	7
1	0	1	2	3	4	5	
2	0	1	2	3	4	5	
3	0	1	2	3	4	5	
4	0	1	2	3	4	5	
5	0	1	2	3	4	5	
6	0	1	2	3	4	5	
7							
8							

yy <6x6 double>

	1	2	3	4	5	6	7
1	-5	-5	-5	-5	-5	-5	
2	-4	-4	-4	-4	-4	-4	
3	-3	-3	-3	-3	-3	-3	
4	-2	-2	-2	-2	-2	-2	
5	-1	-1	-1	-1	-1	-1	
6	0	0	0	0	0	0	
7							
8							

그림 2.5: meshgrid

xx는 행렬의 가로로 가면서 벡터 $x1$의 값을 가지게 되고 yy는 세로로 내려가면서 벡터 $y1$의 값을 가지게 된다. 만약 $x1$을 세로로 $y1$을 바꾸고 싶다면 **ndgird**를 사용한다.

```
>> x1 = [0 1 2 3 4 5]; y1=[-5 -4 -3 -2 -1 0];
>> [xx yy]=ndgrid(x1,y1);
```

위의 meshgrid를 사용하여 정의역 $\Omega = [0\ 5] \times [-5\ 0]$에서의 곡면

$$z = 12 - x^2 + y^2$$

를 그려 보자.

```
>> x1 = [-4 -3 -2 -1 0 1 2 3 4]; y1=[-4 -3 -2 -1 0 1 2 3 4];
>> [xx yy]=meshgrid(x1,y1);
>> zz=12-xx.^2+yy.^2;
>> mesh(xx,yy,zz);
```

2차원 x, y에 정의된 3차원 그래프를 그릴 때에는 **mesh**를 쓴다.

trisurf와 quiver3

```
clc;clear;
p=[0 0 0;1 1 1;-1 1 1;-1 -1 1;1 -1 1];
```

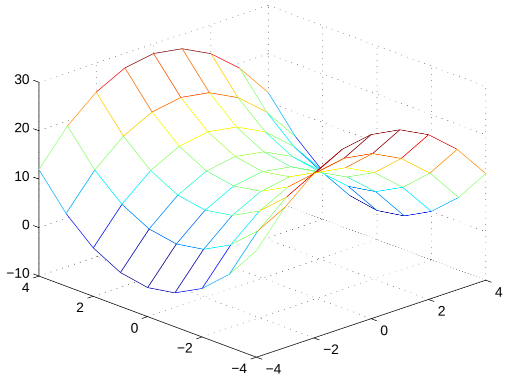


그림 2.6: mesh를 이용한 그래프

```
t=[1 2 3;  1 3 4;  1 4 5;  1 5 2];
len=length(t);
figure(1), trisurf(t, p(:,1), p(:,2),p(:,3))
 for i=1:len;
     v1=p(t(i,2),:)-p(t(i,1),:);
     v2=p(t(i,3),:)-p(t(i,1),:);
     n(i,:)=cross(v1,v2);
     nrm=norm(n(i,:));
     n(i,:)=n(i,:)/nrm;
 end
 figure(2),
quiver3(zeros(len,1),zeros(len,1),zeros(len,1),n(:,1),n(:,2),n(:,3))
```

trisurf의 첫 번째에는 연결할 세 점의 번호를 쓰고 다음에는 점들의 x, y, z 좌표를 쓰면 삼각화(triangulation)된 그림이 나온다. 다음 그림으로 quiver3는 벡터가 시작할 x, y, z좌표를 쓰고 각각의 벡터의 x, y, z를 시작 점 좌표 다음으로 입력하면 된다.

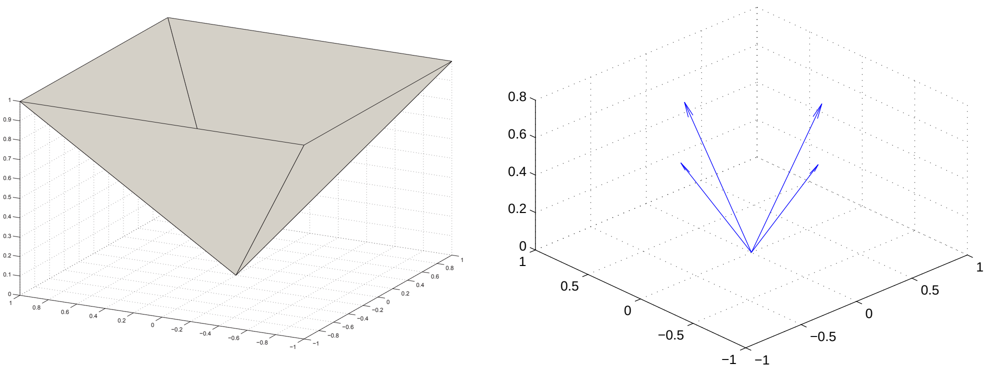


그림 2.7: trisurf와 quiver3를 이용한 그래프

2.2 3차 points 그리기

scatter3를 사용하면 다양한 크기와 색을 가지는 점을 3차원 그래프 안에 그릴 수 있다.

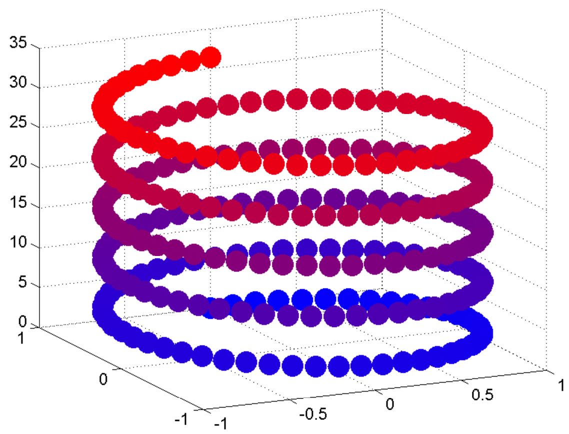


그림 2.8: scatter3 이용한 그래프

```
clc;clear;
t=linspace(0,10*pi,255); t=t(:);
red=[1 0 0]; blue=[0 01];
```

```
x=sin(t); y=cos(t);
frac=t/(10*pi); color=frac*red+(1-frac)*blue;
scatter3(x,y,t,200,color,'filled')
```

2.3 3차 그래프에서 contour 데이터 다루기

데이터를 등고선으로 나타내 보자. 아래와 같은 데이터A가 있다고 하자. 이 데

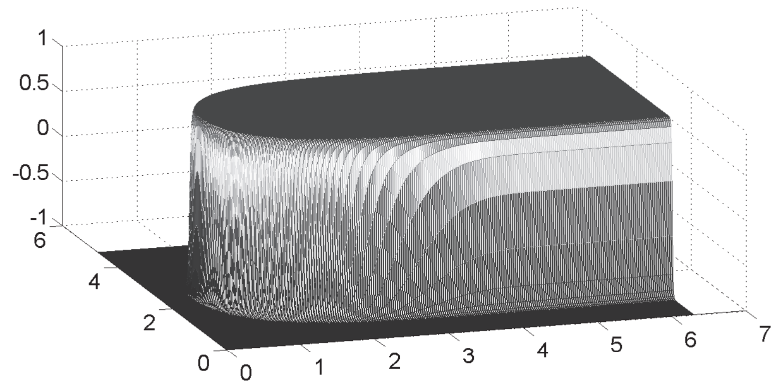


그림 2.9: contour 명령문 쓰기.

이터에서 0의 값만 추출하려면 아래와 같이 입력한다.

```
[c h]=contour(A,[0 0]);
```

그러면 등고선이 0인 데이터을 모은 contour 행렬 c가 출력된다. c의 첫번째 행은 x의 좌표이며 두 번째 행은 y값을 의미한다.

제 3 절 그래프에 주석 넣기

title(′**text**′)는 그래프의 제목을 쓴다.
xlabel(′**text**′)와 **ylabel**(′**text**′)는 x축과 y축의 변수를 표시해 준다.
grid on은 주어진 그래프에 격자선을 생성한다.
axis image으로는 생성하고자 하는 그래프의 범위를 정해준다.
set(**gca**,′**xtick**′, [])는 x축의 tick 즉, 좌표숫자를 없애준다.

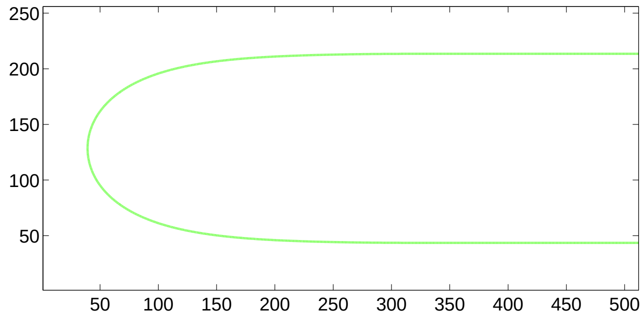


그림 2.10: contour로 zero-level set의 그림

여러 개의 그래프를 순서대로 출력 후 자동으로 jpg나 bmp로 저장하기 위해 saveas 명령문을 사용하면 쉽게 그 결과 그래프를 다른 이름으로 저장할 수 있게 된다.

```
for it=1:100
fnam=sprintf('figure0 %2.2d',it);
saveas(gcf,fnam,'bmp');
axis image, getframe(gcf);
end
```

위와 같이 입력하면 파일 이름이 figure1.bmp에서 figure100.bmp까지 100개의 그림 파일이 다른 이름으로 저장된다.

제 4 절 Vector Fields

다음의 상미분 방정식 Ordinary Difference equation(ODE)을 풀어보자.

$$\frac{dy}{dx} = \frac{3x^2 + 4x + 2}{2(y-1)} \tag{2.1}$$

```
clc;clear;
[x,y]=meshgrid(-3:.4:3,-3:.4:3);
dy=3*x.^2+4*x+2;
dx=2*(y-1);
```

```
dyu=dy./sqrt(dy.^2+dx.^2);
dxu=dx./sqrt(dy.^2+dx.^2);
quiver(x,y,dxu,dyu)
grid off
```

quiver는 x축와 y축으로 묶인 좌표(meshgrid)의 한 점에서 벡터의 성분을 표시한다.

미분 방정식 (2.1)을 아래와 같이 바꾸어 초기값 문제를 푼다.

$$2(y-1)dy = (3x^2+4x+2)dx$$
$$y^2-2y = x^3+2x^2+2x+c$$

초기값 $y(0)=-1$을 가진다고 하면 근은 다음과 같다.

$$y = 1-\sqrt{x^3+2x^2+2x+4}$$

```
clc;clear;
hold on
fplot('1-(x.^3+2*x.^2+2*x-1)^(1/2)',[0.35,2])
fplot('1+(x.^3+2*x.^2+2*x-1)^(1/2)',[0.35,2])
fplot('1-(x.^3+2*x.^2+2*x+1)^(1/2)',[-1.0,2])
fplot('1+(x.^3+2*x.^2+2*x+1)^(1/2)',[-1.0,2])
fplot('1-(x.^3+2*x.^2+2*x+4)^(1/2)',[-2,1.5])
fplot('1+(x.^3+2*x.^2+2*x+4)^(1/2)',[-2,1.5])
fplot('1-(x.^3+2*x.^2+2*x+6)^(1/2)',[-2.28,1.5])
fplot('1+(x.^3+2*x.^2+2*x+6)^(1/2)',[-2.28,1.5])
axis ([-3 3 -2.5 3])
```

이번에는 **ode45**을 사용하여 그려보자. ode45은 Runge-Kutta 방법을 사용하여 수치해석적으로 미분방정식을 풀게 된다. 다음과 같은 미분방정식이 있다

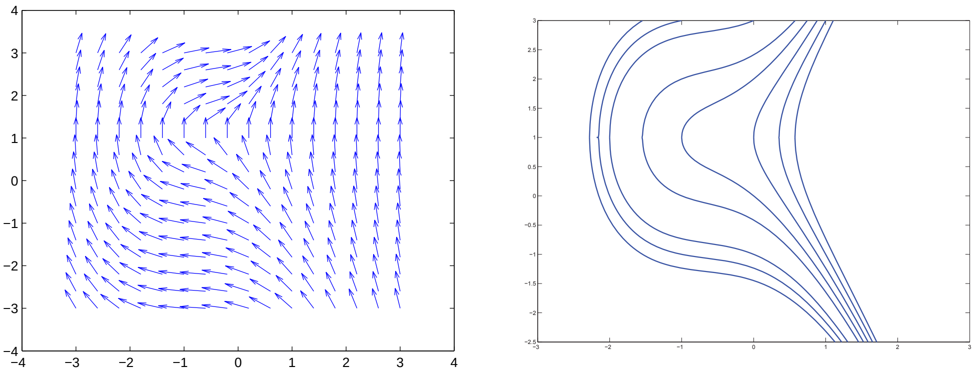


그림 2.11: quiver와 fplot을 이용한 그래프

고 하자.

$$y'' + 2y' + y = e^{-t} \tag{2.2}$$

초기조건은

$$y(0) = 1, \qquad y'(0) = 1$$

이다.

그러면 $y1 = y$, $y2 = y'$이라고 하고 $y1$과 $y2$에 대한 함수를 정의해 보자.

```
function dy = ode_examp1(t,y)
    y1=y(1);
    y2=y(2);
dy1 = y2;
dy2 = -2*y2 -y1 +2*exp(-t);
dy = [dy1;dy2];
```

현재 directory에 함수 파일을 저장하고 Command Window에 다음과 같이 입력한다.

```
>> [t,y]=ode45(@ode_examp1,[0 10],[1;0]);
```

명령문 괄호 중 @는 함수를 부르기 위한 기호이며 t의 범위는 [0 10]까지 이며 초기조건 $y1(0) = 1,\ y2(0) = 0$이므로 [1;0]을 입력하였다. 이제 근을 그리기 위해 plot명령문을 실행해 본다.

```
>> plot(t,y(:,1))
```

ode45에 의해 생성된 y의 첫 번째 열이 근이므로 y(:,1)만 출력하였다. 그러면 해

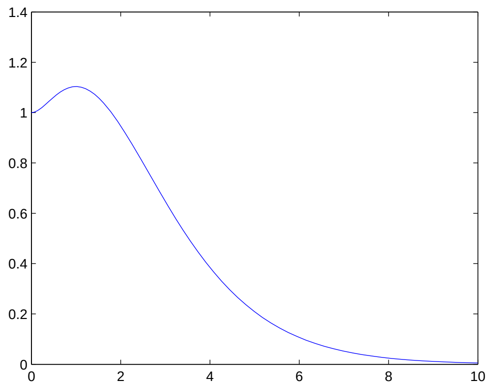


그림 2.12: ode45를 이용한 plot

석해와 비교해 보자.
처음으로 식(2.2)의 일반근(general solution)을 구하기 위해

$$y'' + 2y' + y = 0 \tag{2.3}$$

위의 식의 근을 상수 r을 가지고 $y_1(t) = e^{rt}$이라고 하자. 그리고 식(2.3)에 이 근을 넣으면 다음과 같은 식을 얻는다.

$$0 = y_1'' + 2y_1' + y_1 = (r^2 + 2r + 1)e^{rt} = (r+1)e^{rt}$$

따라서 $r = -1$이므로 $y_1(t) = e^{-t}$이다. 다음으로 d'Alembert's Method을 적용한다. 모든 근$y(t)$을 어떤 t에 대한 함수 $\nu(t)$과 $y_1(t)$의 곱으로 나타낼 수 있다고 하

자. 그러면 다음과 같은 미분식을 얻는다.

$$
\begin{aligned}
y &= \nu e^{-t} \\
y' &= (\nu' - \nu)e^{-t} \\
y'' &= (\nu'' - 2\nu' + \nu)e^{t}
\end{aligned}
$$

위 식들을 다시 식(2.3)에 넣으면

$$
\begin{aligned}
0 &= y'' + 2y' + y \\
&= \nu'' e^{-t}
\end{aligned}
$$

이다. 그러므로 어떤 상수 c_1와 c_2에 대하여 $\nu(t) = c_1 + c_2 t$이므로 일반근은

$$
y(t) = \nu(t)y_1(t) = c_1 e^{-t} + c_2 t e^{-t}
$$

이다. 끝으로 nonhomogeneous equation에 대한 particular solution을 $Y = Y(t)$라고 하고 미정 계수 방법(Method of Undetermined Coefficients)으로 근을 구해 본다. 그러면 근은 $Y'' + 2Y' + 2Y = 2e^{-t}$ 만족하므로 근을 다음과 같이 가정하게 되면

$$
Y(t) = e^{\alpha t}(A_0 t^2 + A_1 t + A_2)
$$

이 식의 상수 $\alpha,\ A_0,\ A_1,\ A_2$는 다음을 통해 결정된다.

$$
\begin{aligned}
2e^{-t} &= Y'' + 2Y' + 2Y \\
&= \left[(\alpha+1)^2 A_1\right] t^2 e^{\alpha t} + \left[4(\alpha+1)A_0 + (\alpha+1)^2 A_1\right] t e^{\alpha t} \\
&+ \left[2\alpha A_0 + 2(\alpha+1)A_1 + (\alpha+1)^2 A_2\right] e^{\alpha t}
\end{aligned}
$$

α가 -1일 때 $2A_0 e^{-t}$가 되어 $A_0 = 1$이고 $A_1 = A_2 = 0$이 된다. 따라서 particular solution은 $Y(t) = t^2 e^{-t}$이고 초기 조건을 적용하면 근은

$$
y(t) = e^{-t} + t e^{-t} + t^2 e^{-t}
$$

이다. 해석해는 아래와 같이 MATLAB으로 그릴 수 있다.

```
clc;clear;
n=100;
t=linspace(0,10,n);
c1=1;c2=1;
y=c1*exp(-t)+c2*t.*exp(-t)+t.^2.*exp(-t);
figure(2),
plot(t,y)
```

제 5 절 복소수에서 그래프 그리기

복소수를 가지는 정의역 $\Omega \subset \mathbf{C}$에서

$$\Omega = \{\eta | -\infty \le Re(\eta) \le \infty,\ 0 \le Im(\eta) \le \pi\},$$

다음과 같은 함수(exponential mapping, meromorphic function)가 있다고 하자.

$$z(\eta) = \sqrt{1 + e^{\eta}}.$$

그러면 이 함수는 좌우로 긴 띠(The domain of parallelism)를 (1,0)가 빈(pole) 원의 일부분으로 보내게 된다.

```
clc;clear;
nu=50; nv=50;
uu =linspace(-2,3,nu);
vv =linspace(0,pi,nu);
figure(1),
[u,v]=meshgrid(uu, vv);
plot(v,u,'b-')
for ii=1:nu
for jj=1:nv
h(ii,jj) = uu(ii) + vv(jj)*i;
```

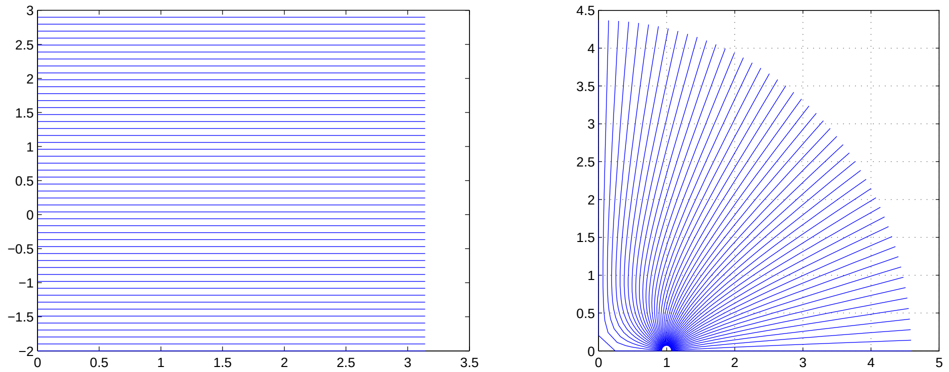


그림 2.13: η에서 z으로의 함수, $(\eta) = \sqrt{1+e^{\eta}}$을 이용한 복소수 그래프

```
end
end
figure(2),
plot(sqrt(1+exp(h)),'b')
axis square; grid on
```

3 장

MATLAB 입출력

제 1 절 출력

계산하는 중간이든 끝이 나서 결과값을 출력할 필요가 있다. 그런 경우 **disp** 명령문을 쓴다.

```
clc;clear;
n=9;str='안녕';
disp(str)
disp(n)
```

위와 같이 입력하고 실행하면

```
안녕
     9
```

이 나온다. 문장이나 단어 중간에 숫자를 삽입하고 싶으면 **sprintf**를 써도 된다.

```
>> n = 9;
>> sprintf('안녕 %d',n)
```

```
안녕 9
```

출력을 잠시 멈추고 진행하려면 **pause**은 쓴다. pause(t)에서 t초 만큼의 멈추었다가 실행을 진행하게 된다.

제 2 절 입력

load를 쓰면 아주 큰 데이터도 쉽게 입력할 수 있다. 현재 directory에 data.dat라는 파일에 데이터가 저장되어있다고 하자.

```
%%%%% data.dat %%%%%
1 2 3
2 3 4
3 3 3
```

그러면 load data.dat라고 입력하면 workspace에 data가 입력된다.

제 3 절 데이터 내보내기

MATLAB의 workspace에 저장된 데이터를 파일로 출력할 수 있다. 가령 points라는 데이터가 MATLAB에 저장되어 있다고 하자.

```
fid=fopen('points.m','w');
fprintf(fid,'%f %f %f \n',points);
fclose(fid);
```

위와 같이 입력하면 points.m이라는 파일을 생성하고 points의 x, y, z의 3개의 행을 파일로 쓰게 된다.

제 4 절 여러 개의 연속된 그림을 동영상으로 내보내기

우선 여러 개의 연속된 그림을 만든다. 그리고 그 그림의 이름을 증가하는 정수로 표현하여 각 그림을 프레임으로 지정한다.

```
filename = sprintf('figure%d', i);
saveas(gcf, filename, 'jpg');
```

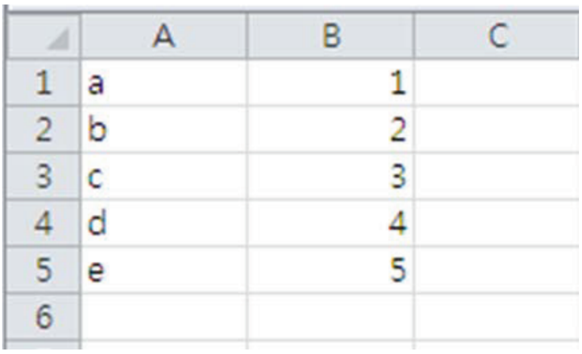
```
pause(0.1)
M(i) = getframe;
```

여기서 filename은 연속된 정수의 파일 이름을 받아 그 그림을 saveas로 하여 jpg파일 형식으로 저장하고 i번째 프레임을 M(i)으로 저장하게 된다. 그리고 아래와 같이 입력하면 movies.avi라는 영상파일이 만들어 진다.

```
>>  movie2avi(M, 'movies.avi');
```

제 5 절 Excel 데이터 읽고 내보내기

아래와 같이 문자와 숫자로 되어 있는 엑셀파일 abc.xls가 있다고 하자. MATLAB 명령어 **xlsread** 데이터를 입력 받을 수 있다.



	A	B	C
1	a	1	
2	b	2	
3	c	3	
4	d	4	
5	e	5	
6			

그림 3.1: 엑셀파일 읽기

```
[number, text] =xlsread('abc');
```

이라고 입력하면 수는 number에 문자는 text에 들어가게 된다. 이제 엑셀 파일을 내보내기 위해 **xlswrite** 명령어를 써보자. 아래와 같이 문자와 숫자가 함께 셀(cell)에 있다.

```
 M = 'a', 'b'; 1 2; 3 4; 5 5;
```

그러면 엑셀의 B2에서 행렬을 써보자. 아래와 같이 입력을 한다.

```
xlswrite('abcd.xls', d, 'aaa', 'B2')
```

그러면 abcd.xls이라는 파일을 만들어 aaa라는 sheet의 B2를 시작으로 행렬을 입력하게 된다.

4 장

MATLAB과 미분방정식

제 1 절 Euler 방법

한 점 (t_i, y_i)에서 1차 미분함수$f(t_i, y_i)$는 이 점에서의 기울기ϕ를 의미한다.

$$\phi = f(t_i, y_i)$$

그러면 이 미분방정식을 이용하여 t축의 t_i점에서 h만큼 이동한 점 $t_{i+1} = t_i + h$에서의 y_{i+1}값을 추정할 수 있다.

$$y_{i+1} = y_i + f(t_i, y_i)h \tag{4.1}$$

위 식(4.1)을 이용하여 근사값을 구하는 것을 Euler 방법이라고 한다. 다음의 초기값 문제를 보자.

$$\frac{dy}{dt} = \frac{3t^2 - e^t}{2y - 5}, \qquad y(0) = 1 \tag{4.2}$$

우선 식 (4.2)을 해석적으로 구하여 그 값과 Euler 방법으로 근사한 값을 비교해 보자. 해석해를 구하기 위해 주어진 식에 integrating factor $e^{1/2t}$를 곱하고 초기값을 넣어 근을 구하면

$$y = -13e^{-t/2} - 4t + 14 \tag{4.3}$$

이다. 이제 Euler 방법으로 각 $t_1 = 0.2, t_2 = 0.4, \cdots, t_5 = 1.0$에서의 $y_i,\ i = 1 \cdots 5$의 값을 추정해 보자. t의 크기 변화(step size)를 일정하게 $h = t_{i+1} - t_i = 0.2$이라고 하면 $t_1 = 0.2$에서의 첫 번째 추정값 $Y(1) \approx y(0.2)$은 다음과 같이 쓸 수 있다.

$$Y(1st) = y(t_0) + f(t_0, y_0) \cdot (h)$$
$$y(0.2) \approx Y(1) = y(0) + f(0, 1) \cdot (0.2)$$

마찬가지로 두 번째 추정값 $Y(2)$는 첫 번째 추정값 $Y(1)$을 이용하면

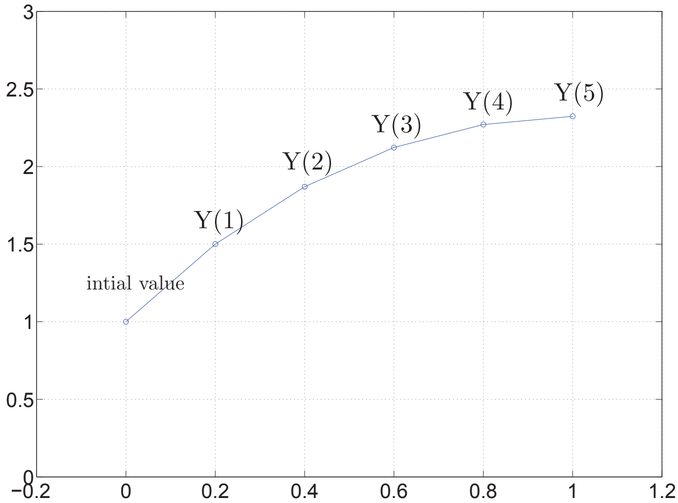


그림 4.1: 도메인 x와 $y(i)$의 관계

$$Y(2nd) = y(t_1) + f(t_1, y_1) \cdot (h)$$
$$y(0.4) \approx Y(2) = Y(1) + f(0.2, Y(1)) \cdot (0.2)$$

이다. 그림(4.1)은 위 식의 $y(t_i)\ 1 \le i \le 5$을 나타낸 것이다.

즉, 그래프에서의 $Y(i)$는 벡터값이며 $y(t_i)$의 값을 나타내게 된다. 예를 들어 $y(3)$은 y벡터의 세 번째 값으로서 x 도메인의 $x = 0.6$에 해당하는 값을 의미한다.

$$Y = [y(t_0) + f(t_0, y_0)h,\ y(t_1) + f(t_1, y_1)h,\ y(t_2) + f(t_2, y_2)h,\ \cdots\ y(t_5) + f(t_5, y_5)h]^T$$

다음으로 Euler 방법을 사용하여 $t = 1$일 때 $y(1.0)$을 값을 추정하여 보자. t사이의 간격(h)를 0.2로 두었기 문에 $y(1.0)$의 값은 Y벡터의 다섯 번째 값 $Y(5)$이 된다. 각 t_i 점에서 근의 상대 오차는 다음과 같이 정의된다.

t	y함수값	Y벡터	해석해	근사해	상대 오차
0.2	y(0.2)	Y(1)	1.5000	1.4371	4.1924
0.4	y(0.4)	Y(2)	1.8700	1.7565	6.0695
0.6	y(0.6)	Y(3)	2.1230	1.9694	7.2368
0.8	y(0.8)	Y(4)	2.2707	2.0858	8.1411
1.0	y(1.0)	Y(5)	2.3236	2.1151	8.9743

표 4.1: Table ($h = 0.2$일 때 해석해와 근사해의 상대 오차(relative error))

$$100 \times \frac{(\text{근사해-해석해})}{\text{근사해}}.$$

```
clc;clear;
previous_value=1;
h=0.2;
t=0:h:1;
tend=length(t);

dydt=@(t,y) (3-2*t-0.5*y);
y=zeros(1,tend);err=zeros(tend,1);
y(1)=previous_value;
for i=1:tend-1
y(i+1)=previous_value+dydt(t(i),previous_value)*h;
previous_value=y(i+1);
```

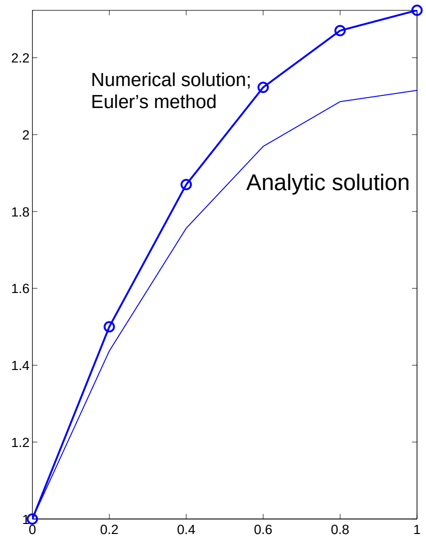


그림 4.2: 해석해와 근사해의 차이

```
end
figure(2),
plot(t,y,'o-')
hold on
yanly=-13.*exp(-t./2)-4.*t+14;
plot(t,yanly)
axis image
% relative error
err=100*(y(:)-yanly(:))./(y(:));
fprintf('          t      true      estimate     error\n')
disp([t' y' yanly' err])
```

5장

MATLAB과 복소해석학

MATLAB에서 별다른 지정이 없으면 i, j가 복소수를 나타내는 수가 된다. 만일 $(1+i)^4$를 구한다면 아래와 같이 입력한다.

```
>> (1+i)^(4)
ans =

   -4
```

또한 $p(x) = x^2 + 1$의 근(roots)을 찾기 위해서는 함수 p의 계수를 차례대로 입력한 후 **roots** 명령어를 사용한다.

```
>> p=[1 0 1];r=roots(p)
r =
        0 + 1.0000i
        0 - 1.0000i
```

근(roots)은 $i, -i$임을 알 수 있다.

함수 f가 자신을 뺀 한 점 a 주변(neighborhood)에서 해석적(analytic)이고 만일 $\lim_{z\to a} f(z) = \infty$이면 함수 f의 한 점을 pole이라고 한다. 또한 함수 f가 양의 정수n을 갖고 함수 g가 점 a에서 다음 식(5.1)을 만족하면서 없어지지(vanishing) 않을 때 그 n을 order라고 하고 이와 같이 discrete poles를 제외하고 analytic한 함

수를 meromorphic이라고 한다.

$$f(z) = \frac{g(z)}{(z-a)^n}. \tag{5.1}$$

제 1 절 The Residue Theorem

복소함수에서 residue theorem은 complex line integrals을 계산하는데 중요하다. 함수 $f(x)$의 z_0에서의 residue은 Laurent expansion $f(z) = \sum_{n=-\infty}^{\infty} a_n(z-z_0)^m$, $0 < |z-z_0| < \rho$에서의 계수 a_{-1}를 의미한다. 즉, 다음과 같다.

$$Res[f(z), z_0] = a_{-1} = \frac{1}{2\pi i}\oint_{|z=z_0|=r} f(z)dz \tag{5.2}$$

$$\text{where } r \text{ is any fixed radius satisfying} \quad 0 < r < \rho. \tag{5.3}$$

예를 들어 정의와 같이 complex line integrals을 한다.

$$Res[\frac{1}{z}, 0] = 1, \quad Res[\frac{1}{z-z_0)^2}, z_0] = 0 \tag{5.4}$$

MATLAB으로 아래의 식의 poles과 residue을 계산해 보자.

$$f(z) = \frac{1}{z^2+2iz+3} \tag{5.5}$$

residue을 사용하면 r은 residue와 그에 해당하는 p의 pole 그리고 direct term k를 출력한다.

```
>> p=[1];q=[1 2*i 3];[r,a,k]=residue(p,q)
r =
  -0.0000 + 0.2500i
   0.0000 - 0.2500i


a =
        0 - 3.0000i
   0.0000 + 1.0000i


k =
     []
```

제 2 절 Complex Line Integral

Complex Line Integral을 계산해 보자. 선구간(straight line segment) 0에서 $1+i$에서 $\int_0^{1+i} z^2$를 구한다고 하자. 그러면 z에 대하여 t를 사용하여 x와 y로 매개변수화 한다. 즉, $z(t) = t + ti,\ 0 \le t \le 1$. 그러면 $x(t) = t, \quad y(t) = t$가 되어 $dz = dx + idy = (1+i)dt$을 얻는다. 이제 적분을 하면

$$\int_0^{1+i} z^2 dz = \int_0^1 [(1+i)t]^2 (1+i)dt = (1+i)^3 \int_0^1 t^2 dt = \frac{(1+i)^3}{3} \tag{5.6}$$

을 얻는다. 이 계산을 MATLAB으로 해보자.

```
clc;clear;
syms t real;
y=t; x=t; z=simple(x+i*y);
f=z^2;
Integrand=f*diff(z,t);
F=int(Integrand,'t',0,1);
F=double(F)


F =


  -0.6667 + 0.6667i
```

위 계산을 매개변수를 사용하지 않고 간단하게 할 수 있는 방법도 있다. 내장함수 quad를 이용하여 위의 식을 풀어 보자.

```
clc;clear;
f1='z.^2';
c1=1+i;
cli=quad(f1,0,c1);


cli =


  -0.6667 + 0.6667i
```

그러면 매개변수를 사용한 값과 동일한 값을 얻을 수 있다.

6장

MATLAB과 기하학

제 1 절 곡률(Curvature)

곡면조각 $\overrightarrow{X}: \mathbf{U} \subset \mathbb{R}^2 \to \mathbb{R}^3$에 모양 연산자(shape operator)라는 것은 곡면 상의 한 접벡터를 공간 상의 벡터로 보내주는 사상 S를 말한다.

곡면 조각$\overrightarrow{X}$의 제 1기본 형식(First Fundamental Form)

$$E =:< \overrightarrow{X}_u, \overrightarrow{X}_u >, F =:< \overrightarrow{X}_u, \overrightarrow{X}_v >, G =:< \overrightarrow{X}_v, \overrightarrow{X}_v >$$

```
function f1 = EFG(r)
syms u v real;
ru = diff(r, u);
rv = diff(r, v);
E = ru*ru';
F = ru*rv';
G = rv*rv';
f1 = simplify([E, F, G]);
end
```

제 2기본 형식(Second Fundamental Form)

$e =:< \overrightarrow{X}_{uu}, N >$, $f =:< \overrightarrow{X}_{uv}, V >$, $g =:< \overrightarrow{X}_{vv}, N >$

```
function f2 = LMN(X)
syms u v real;
ru = diff(r, u);
rv = diff(r, v);
ruu = diff(ru, u);
ruv = diff(ru, v);
rvv = diff(rv, v);
n = cross(ru, rv);
UN = n/simple(sqrt(n*n'));
L =UN*ruu';
M = UN*ruv';
N = UN*rvv';
f2 = simplify([L, M, N]);
end
```

평균 곡률 H와 가우스 곡률 K는 다음과 같이 정의된다.

$$H = \frac{eG - 2Ff + Gg}{EG - F^2} \quad K = \frac{eg - f^2}{EG - F^2} \tag{6.1}$$

평균 곡률 계산하기

```
f = simplify((G*L + E*N - 2*F*M)/(2*E*G - 2*F^2))
```

가우스 곡률 계산하기

```
function f = GK(r)
syms u v real;
S = EFG(r);
T = LMN(r);
E = S(1);
F = S(2);
G = S(3);
```

```
L = T(1);
M = T(2); N = T(3);
f = simplify((L*N - M^2)/(E*G - F^2));
end
```

예)

```
clear;clc; syms u v R real; helicoid = [u*cos(v), u*sin(v), v];
aa=GK(helicoid)
simplify(aa)
%Answer: -1/cosh4 u
```

참고 문헌

[1] 김강수, 이기황, MIKA, 김지운, 샘처럼 옮김, The Not So Short Introduction to LaTeX2e, 2005.

[2] Day RA. How to write and publish a scientific paper. 3rd ed. Phoenix, AZ: Oryx Press, 1988.

[3] Chapra, Applied Numerical Methods with MATLAB. 2nd ed, McGraw-Hill, 2008.

[4] KTUG 한글 TeX 사용자 그룹 http://www.ktug.or.kr