

7장 최적화

7.1 소개와 배경

7.2 1차원 최적화

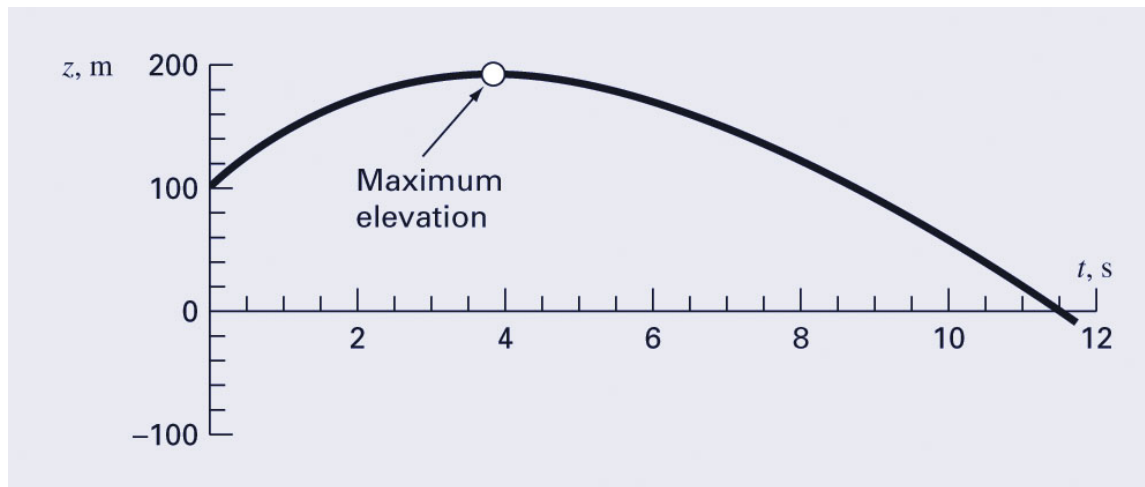
7.3 다차원 최적화



7장 최적화

- 선형 항력을 받는 물체를 일정 속도로 쏘아 올린 경우, 물체의 시간에 따른 높이는 다음과 같다.

$$z(t) = z_0 + \frac{m}{c} \left(v_0 + \frac{mg}{c} \right) \left(1 - e^{-(c/m)t} \right) - \frac{mg}{c} t$$



매개변수:

$$g = 9.81 \text{ m/s}^2$$

$$z_0 = 100 \text{ m}$$

$$v_0 = 55 \text{ m/s}$$

$$m = 80 \text{ kg}$$

$$c = 15 \text{ kg/s}$$



최고 높이와 같은 극점을 구하는 방법 → 최적화

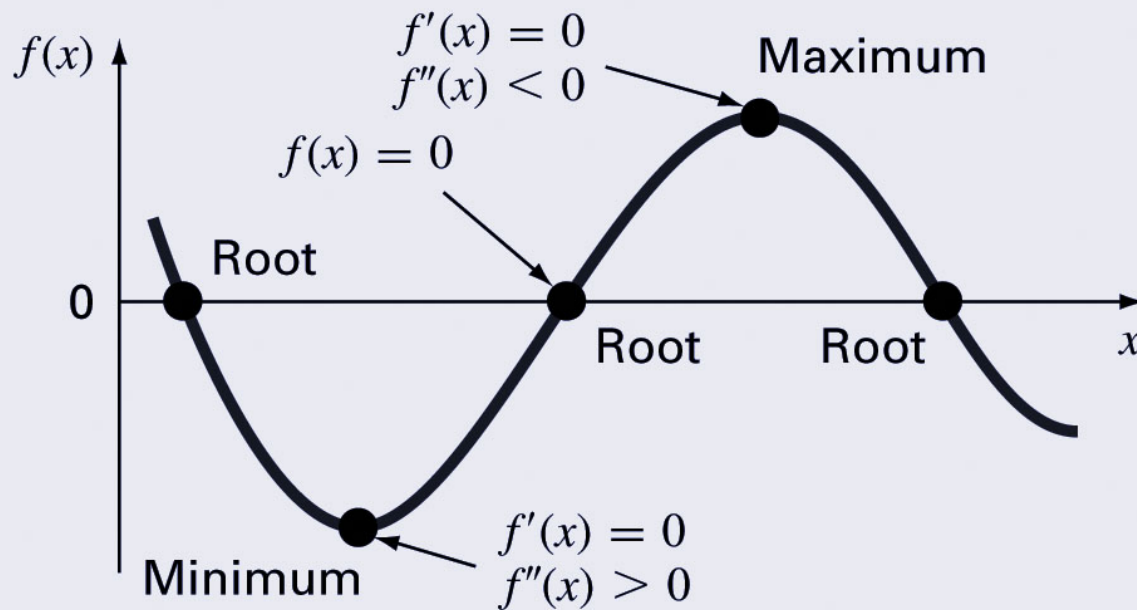


7.1 소개 및 배경 (1/3)

- 공학자는 최소경비로 효율적인 방식을 통하여 작동하는 장치나 제품을 계속해서 설계해야 하므로, 항상 성능과 제약조건 사이에 이해득실을 다루는 최적화 문제에 부딪치게 된다.
- 수학적으로 최적화는 한 개 이상의 변수에 의존하는 함수의 최대값과 최소값을 구하는 것임.
- 근 구하기는 함수 $f(x)=0$ 이 되는 위치를 찾지만, 최적화는 함수의 극점을 찾음.
- 해석적으로 구할 수 있는 경우는 제한적이며, 대부분의 경우 수치적으로 구함.



7.1 소개 및 배경 (2/3)



최적값은 곡선 상의 평탄한 점이 된다. 즉 도함수 $f'(x)=0$ 인 x 값에 해당한다. 또한 2차 도함수인 $f''(x)$ 는 최적값이 최소값인지 최대값인지를 나타낸다. 만일 $f''(x)<0$ 이면 최대값, $f''(x)>0$ 이면 최소값을 나타낸다.



예제 7.1 [1/2]

Q. 식 (7.1)를 사용하여 최고 높이에 도달하는 시간과 크기를 구하라. 단 $g = 9.81 \text{ m/s}^2$, $z_0 = 100 \text{ m}$, $v_0 = 55 \text{ m/s}$, $m = 80 \text{ kg}$, $c = 15 \text{ kg/s}$ 이다.

풀이) 식을 미분하면 다음과 같다.

$$\frac{dz}{dt} = v_0 e^{-(c/m)t} - \frac{mg}{c} (1 - e^{-(c/m)t})$$

최대 높이는 이 식이 0이 되는 값에서 발생한다. 따라서 이 식의 근을 해석적으로 풀면 다음과 같다.

$$t = \frac{m}{c} \ln \left(1 + \frac{cv_0}{mg} \right) = \frac{80}{15} \ln \left(1 + \frac{15(55)}{80(9.81)} \right) = 3.83166 \text{ s}$$



예제 7.1 [2/2]

식 (7.1)로부터 최대값을 구하면,

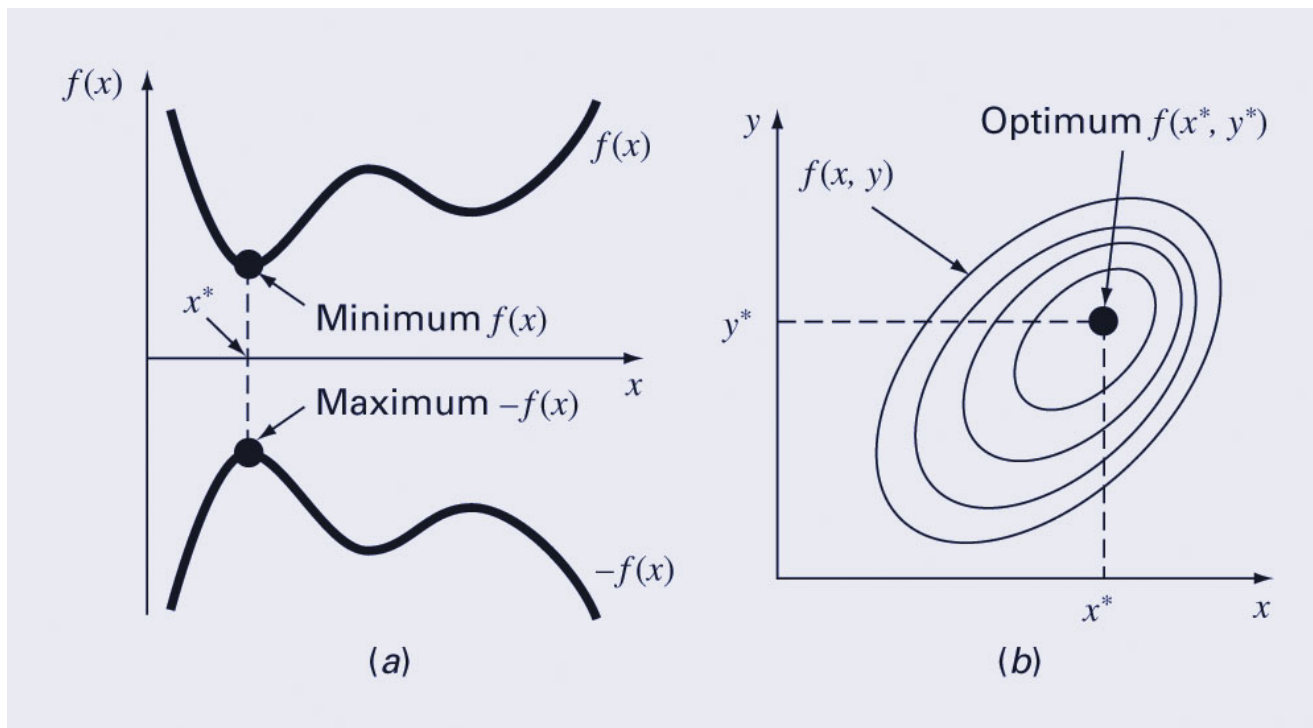
$$z = 100 + \frac{80}{15} \left(50 + \frac{80(9.81)}{15} \right) \left(1 - e^{-(15/80)3.83166} \right) - \frac{80(9.81)}{15} 3.83166 = 192.8609 \text{ m}$$

2차도함수를 구하여 결과가 최대값이 되는지를 확인한다.

$$\frac{d^2 z}{dt^2} = -\frac{m}{c} v_0 e^{-(c/m)t} - g e^{-(c/m)t} = -9.81 \frac{m}{s^2} < 0 \implies \text{최대값}$$



7.1 소개 및 배경 (3/3)



(a) 1차원 최적화. 이 그림은 $f(x)$ 의 최소화가 $-f(x)$ 의 최대화와 같음을 보여준다. (b) 2차원 최적화. 이 그림은 최대값(함수의 등고선 값이 산처럼 정상에서 최대값) 혹은 최소값(함수의 등고선 값이 계곡에서 최소값)을 나타낸다.

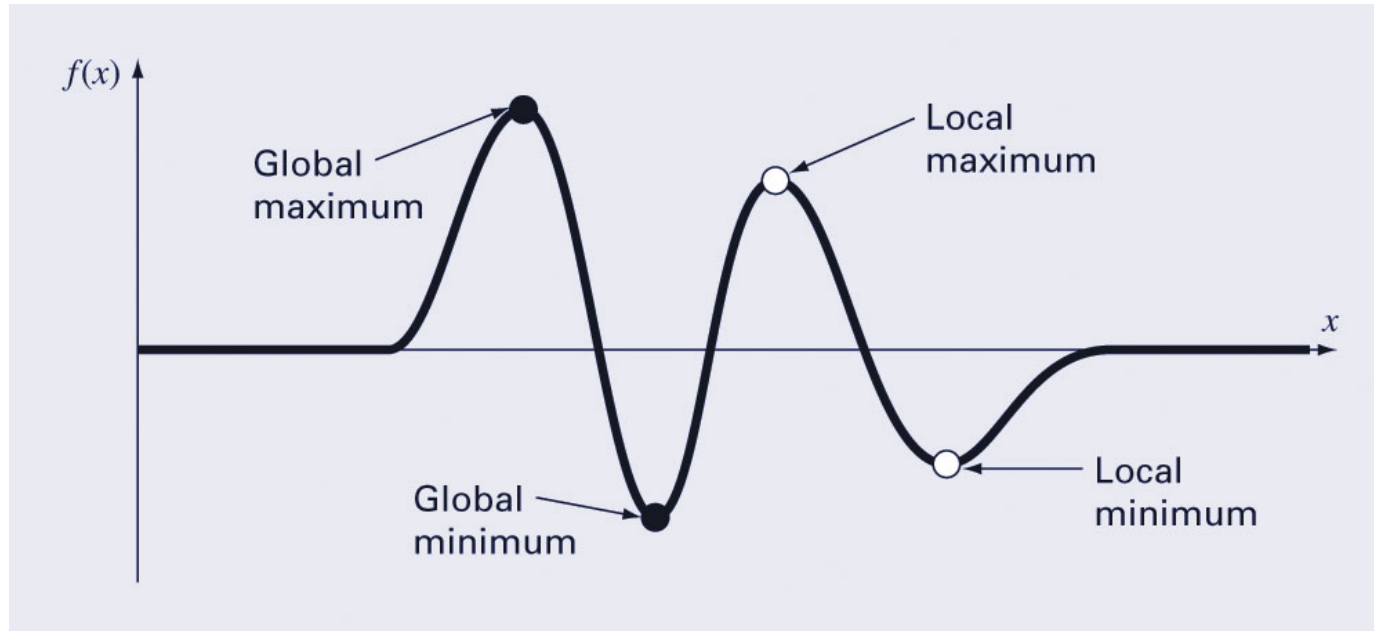


7.2 1차원 최적화 (1/15)

- $f(x)$ 의 최대값과 최소값을 찾는 방법에 대해 설명한다. 최적화 문제에서는 국소 최적값과 전체 최적값이 모두 나타날 수 있다.
- 전체 최적값(global optimum)은 가장 좋은 해에 해당된다. 반면 국소 최적값(local optimum)은 가장 좋은 것은 아니지만 그것에 인접한 값보다는 우수하다. 국소 최적값을 포함하는 경우를 다모드(multimodal) 문제라고 한다.
- 일반적으로 전체 최적값을 찾는 것에 거의 언제나 관심이 있다.



7.2 1차원 최적화 (2/15)



원점 근처에서 두 개의 최대값과 2개의 최소값을 갖는 함수가 있다. 오른쪽에 있는 두 개의 점은 국소적인 최적값에 해당하고, 왼쪽에 있는 두 개의 점은 전체 최적값이 된다.



7.2 1차원 최적화 (3/15)

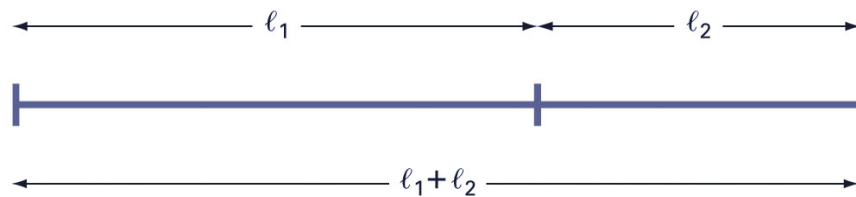
■ 황금분할탐색법

- Euclid의 황금비, ϕ 정의 : “전체 선분에 대한 내부 긴 선분의 비가 내부의 긴 선분에 대한 짧은 선분의 비와 같다면, 이 직선은 외중비(extreme and mean ratio, 황금비)로 나누어진다고 말한다.”

$$\frac{\ell_1 + \ell_2}{\ell_1} = \frac{\ell_1}{\ell_2}$$

$$\text{let } \phi = \frac{\ell_1}{\ell_2} \rightarrow \phi^2 - \phi - 1 = 0$$

$$\phi = \frac{1 + \sqrt{5}}{2} = 1.61803398874989\dots$$



7.2 1차원 최적화 (4/15)

- 한 개의 최소값을 포함하고 있는 구간[단모드 (unimodal) 구간]에서 최소값을 탐색하는 방법.
- 이분법에서 한 개의 중간값을 사용하는 것(부호의 변화를 찾아내어 근을 찾음)과는 다르게, 황금분할 탐색법은 최소값의 발생 여부를 알기 위해 두 개의 중간 함수값이 필요로 한다.
- 이러한 방법이 효율적이기 위해서는 중간점들을 현명하게 선택해야 하며, 이분법에서처럼 이전 값을 새로운 값으로 치환함으로써 함수 계산을 최소화시킨다.



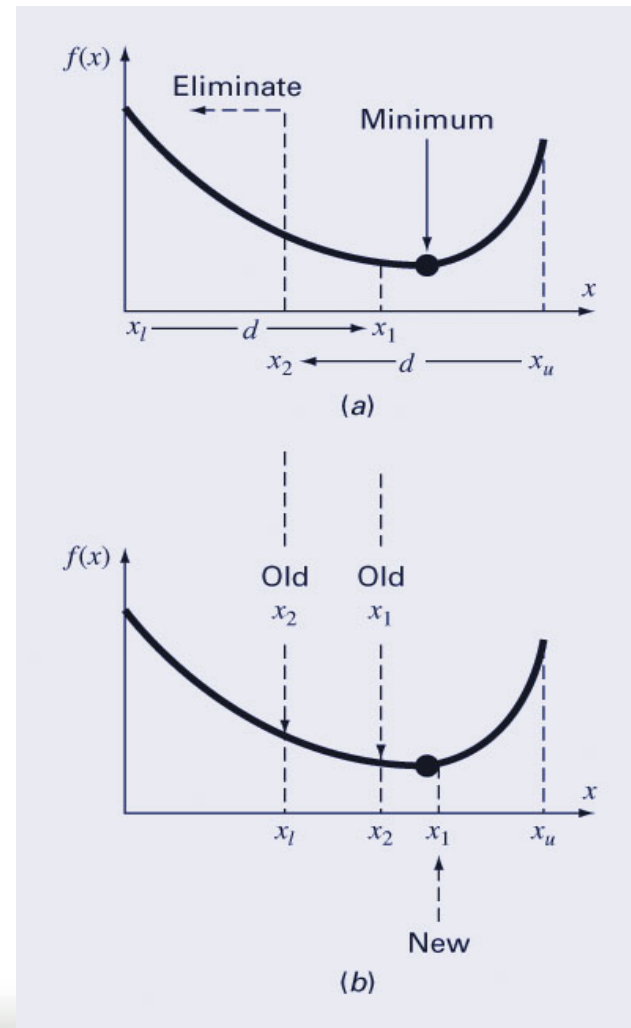
7.2 1차원 최적화 (5/15)

$$d = (\phi - 1)(x_u - x_l)$$

$$x_1 = x_l + d$$

$$x_2 = x_u - d$$

- $f(x_1) < f(x_2)$ 이면 $f(x_1)$ 은 최소이고, x_1 부터 x_2 까지는 최소값을 포함하지 않으므로 제거한다. 이 경우에 x_2 는 새로운 x_l 이 된다.
- $f(x_2) < f(x_1)$ 이면 $f(x_2)$ 가 최소이고, x_1 부터 x_u 까지는 최소값을 포함하지 않으므로 제거한다. 이 경우에 x_1 는 새로운 x_u 가 된다.



7.2 1차원 최적화 [6/15]

- 황금비를 사용하는 장점은 이전 값 x_1 은 새로운 값 x_2 가 된다. 이는 새로운 함수값 $f(x_2)$ 는 다시 계산할 필요가 없음을 의미한다.
- 알고리즘을 완성하기 위하여 단지 새로운 x_1 을 결정하는 것이 필요하며, 이는 새로운 값 x_1 과 x_u 를 기초로 황금비를 이용하여 계산한다.



예제 7.2 [황금분할탐색법] [1/2]

Q. 황금분할탐색법을 사용하여 다음 함수의 최소값을 구간 $x_l = 0$ 에서 $x_u = 4$ 까지의 범위에서 구하라.

$$f(x) = \frac{x^2}{10} - 2 \sin x$$

풀이) 황금비로 두 개의 내부점을 구하면 다음과 같다.

$$d = 0.61803(4 - 0) = 2.4721$$

$$x_1 = 0 + 2.4721 = 2.4721$$

$$x_2 = 4 - 2.4721 = 1.5279$$

내부점에서의
함수값 :

$$f(x_2) = \frac{1.5279^2}{10} - 2 \sin(1.5279) = -1.7647$$

$$f(x_1) = \frac{2.4721^2}{10} - 2 \sin(2.4721) = -0.6300$$



예제 7.2 [황금분할탐색법] (2/2)

$f(x_2) < f(x_1) \rightarrow$ 선택 구간 $[x_1, \cancel{x_2}, \cancel{x_l}]$

d 와 새로운 x_2 계산

\downarrow \downarrow
 x_1 x_u

i	x_l	$f(x_l)$	x_2	$f(x_2)$	x_1	$f(x_1)$	x_u	$f(x_u)$	d
1	0	0	1.5279	-1.7647	2.4721	-0.6300	4.0000	3.1136	2.4721
2	0	0	0.9443	-1.5310	1.5279	-1.7647	2.4721	-0.6300	1.5279
3	0.9443	-1.5310	1.5279	-1.7647	1.8885	-1.5432	2.4721	-0.6300	0.9443
4	0.9443	-1.5310	1.3050	-1.7595	1.5279	-1.7647	1.8885	-1.5432	0.5836
5	1.3050	-1.7595	1.5279	-1.7647	1.6656	-1.7136	1.8885	-1.5432	0.3607
6	1.3050	-1.7595	1.4427	-1.7755	1.5279	-1.7647	1.6656	-1.7136	0.2229
7	1.3050	-1.7595	1.3901	-1.7742	1.4427	-1.7755	1.5279	-1.7647	0.1378
8	1.3901	-1.7742	1.4427	-1.7755	1.4752	-1.7732	1.5279	-1.7647	0.0851



7.2 1차원 최적화 (7/15)

- 황금분할 탐색법에 대한 오차 : 첫 번째 반복에서 최적값은 하부구간(x_l, x_2, x_1) 또는 상부구간(x_2, x_1, x_u) 중 하나에 있고, 내부 점들은 대칭이므로 어느 경우든지 오차를 정의할 수 있다.
- 상부구간 (x_2, x_1, x_u)을 보면,

- 참값이 왼쪽 끝에 위치하는 경우, 추정값으로부터 최대 거리

$$\begin{aligned}\Delta x_a &= x_1 - x_2 \\ &= x_l + (\phi - 1)(x_u - x_l) - x_u + (\phi - 1)(x_u - x_l) \\ &= (x_u - x_l) + 2(\phi - 1)(x_u - x_l) \\ &= (2\phi - 3)(x_u - x_l) = 0.2361(x_u - x_l)\end{aligned}$$



7.2 1차원 최적화 (8/15)

- 참값이 오른 끝에 위치하는 경우, 추정값으로부터 최대 거리

$$\begin{aligned}\Delta x_b &= x_u - x_l \\ &= x_u - x_l - (\phi - 1)(x_u - x_l) \\ &= (x_u - x_l) - (\phi - 1)(x_u - x_l) \\ &= (2 - \phi)(x_u - x_l) = 0.3820(x_u - x_l) \rightarrow \text{최대오차}\end{aligned}$$

- 정규화하면,

$$\varepsilon_a = (2 - \phi) \frac{|x_u - x_l|}{x_{opt}} \times 100\% \rightarrow \text{종료판정 기준}$$



7.2 1차원 최적화 (9/15)

[최소화를 위한 황금분할탐색법의 M-파일 함수]

```
Untitled
File Edit View Text Debug Breakpoints Web Window Help
function [x,fx,ea,iter]=goldmin(f,xl,xu,es,maxit,varargin)
% goldmin: minimization golden section search
% [xopt,fopt,ea,iter]=goldmin(f,xl,xu,es,maxit,p1,p2,...):
%   uses golden section search to find the minimum of f
% input:
%   f = name of function
%   xl, xu = lower and upper guesses
%   es = desired relative error (default = 0.0001%)
%   maxit = maximum allowable iterations (default = 50)
%   p1,p2,... = additional parameters used by f
% output:
%   x = location of minimum
```



7.2 1차원 최적화 (9/15)

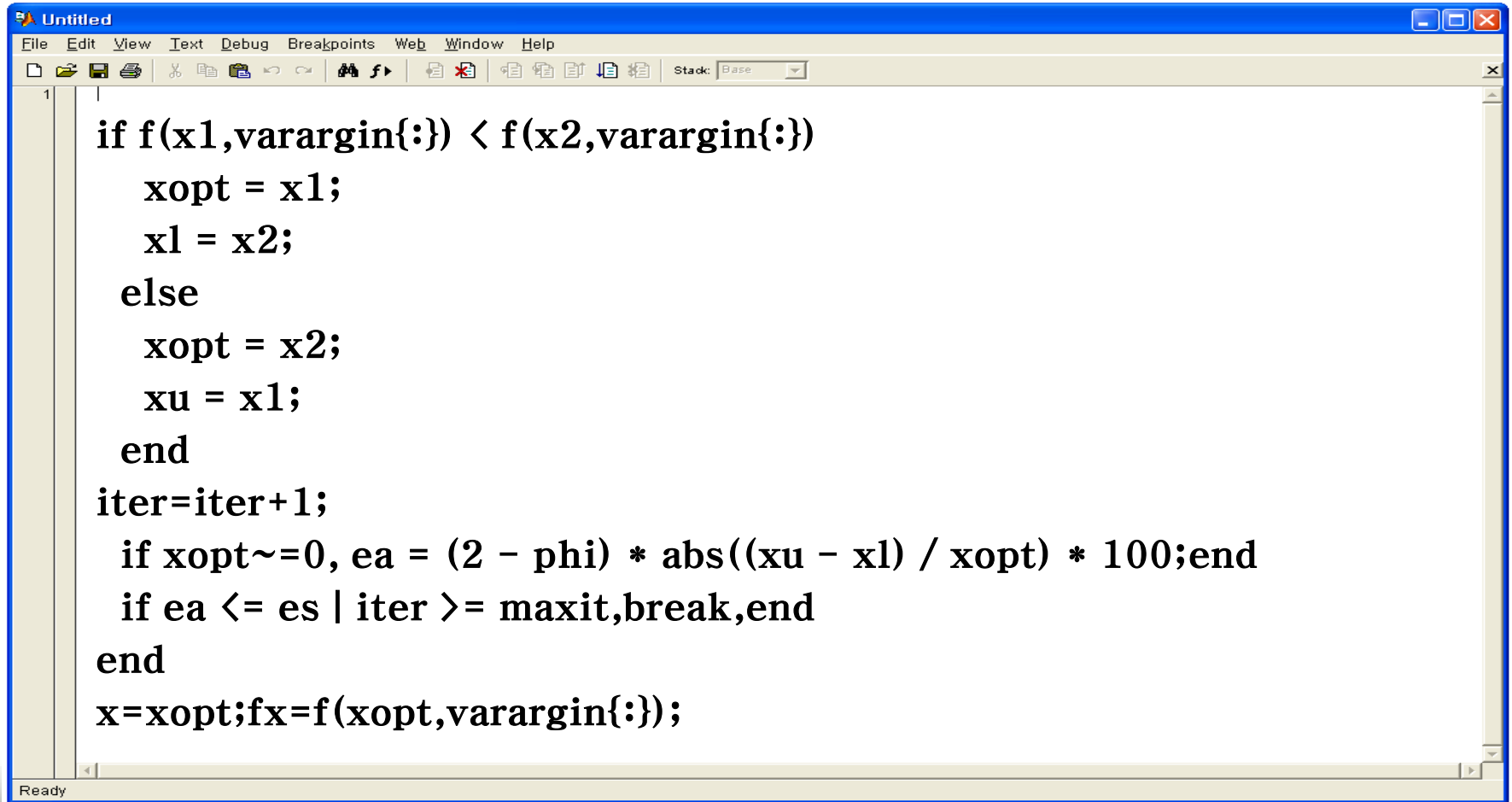
[최소화를 위한 황금분할탐색법의 M-파일 함수]

```
Untitled
File Edit View Text Debug Breakpoints Web Window Help
[Icons] Stack: Base
1 |
  | % fx = minimum function value
  | % ea = approximate relative error (%)
  | % iter = number of iterations
  | if nargin<3,error('at least 3 input arguments required'),end
  | if nargin<4||isempty(es), es=0.0001;end
  | if nargin<5||isempty(maxit), maxit=50;end
  | phi=(1+sqrt(5))/2;
  | iter=0;
  | while(1)
  |     d = (phi-1)*(xu - x1);
  |     x1 = x1 + d;
  |     x2 = xu - d;
```



7.2 1차원 최적화 (9/15)

[최소화를 위한 황금분할탐색법의 M-파일 함수]

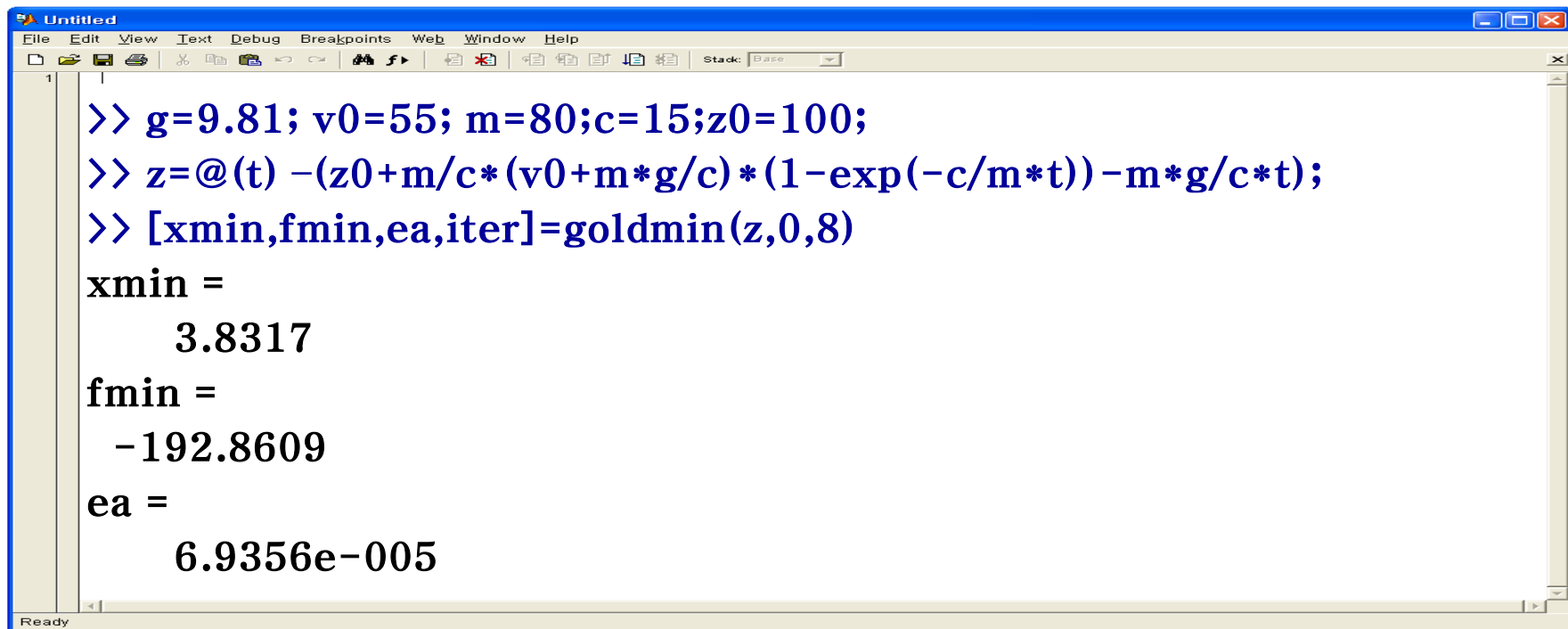


```
1
if f(x1,varargin{:}) < f(x2,varargin{:})
    xopt = x1;
    x1 = x2;
else
    xopt = x2;
    xu = x1;
end
iter=iter+1;
if xopt~=0, ea = (2 - phi) * abs((xu - x1) / xopt) * 100;end
if ea <= es | iter >= maxit,break,end
end
x=xopt;fx=f(xopt,varargin{:});
```



7.2 1차원 최적화 (10/15)

[예제 7.1에 대한 M-파일]



```
Untitled
File Edit View Text Debug Breakpoints Web Window Help
>> g=9.81; v0=55; m=80;c=15;z0=100;
>> z=@(t) -(z0+m/c*(v0+m*g/c)*(1-exp(-c/m*t))-m*g/c*t);
>> [xmin,fmin,ea,iter]=goldmin(z,0,8)
xmin =
    3.8317
fmin =
   -192.8609
ea =
    6.9356e-005
Ready
```

Note : 최대화 문제이므로 식 (7.1)에 음 부호를 추가한 것에 주목한다.
따라서 fmin는 최대 높이 192.8609에 해당된다.



7.2 1차원 최적화 (11/15)

- 황금분할탐색법에서 함수계산의 수를 최소화하고자 하는 두 가지 이유 :

(a) 많은 계산량.

황금분할탐색 알고리즘이 대형 계산의 일부인 경우가 있다. 이러한 경우 여러 번 황금분할탐색 부프로그램을 부르게 된다. 따라서 함수 계산의 수를 최소화 하는 것이 매우 유익하다.

(b) 시간이 많이 드는 계산.

어떤 함수는 매우 복잡하여 함수값을 구하는 시간이 많이 걸릴 수도 있다. 이러한 경우, 함수 계산의 수를 줄이는 방법이 유리하다.



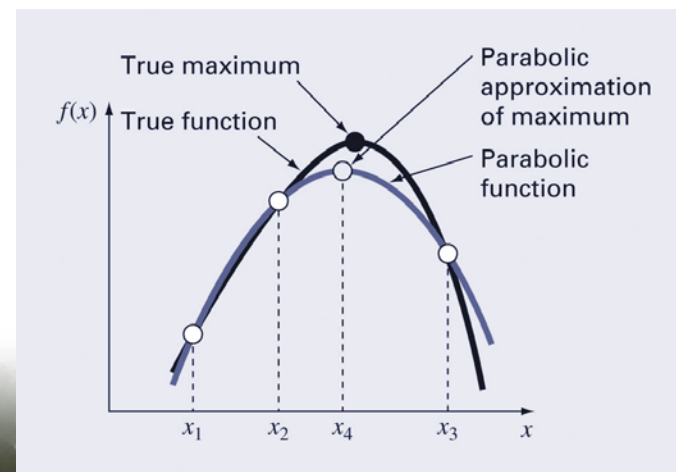
7.2 1차원 최적화 (12/15)

■ 2차 보간법

- 2차 다항식이 종종 최적값 근처에서 $f(x)$ 의 형상을 잘 근사한다는 사실을 이용한다.
- 세 점 (x_1, x_2, x_3) 을 보간하는 포물선의 최대값/최소값의 위치

$$x_4 = x_2 - \frac{1}{2} \frac{(x_2 - x_1)^2 [f(x_2) - f(x_3)] - (x_2 - x_3)^2 [f(x_2) - f(x_1)]}{(x_2 - x_1)[f(x_2) - f(x_3)] - (x_2 - x_3)[f(x_2) - f(x_1)]} \quad \text{식 (7.10)}$$

- 새로운 점 x_4 와 이 점을 둘러싸는 두 점(x_1 과 x_2 또는 x_2 와 x_3) 이 다음 반복에 사용된다.



예제 7.3 [2차 보간법] [1/3]

Q. 2차 보간법을 사용하여 다음 함수의 최소값을 구하라. 단 초기 가정값은 $x_1=0$, $x_2=1$, $x_3=4$ 이다.

$$f(x) = \frac{x^2}{10} - 2 \sin x$$

풀이) 세 개의 가정값에서 함수값은 다음과 같다.

$$x_1 = 0 \quad f(x_1) = 0$$

$$x_2 = 1 \quad f(x_2) = -1.5829$$

$$x_3 = 4 \quad f(x_3) = 3.1136$$

식 (7.10)에 대입하면,

$$x_4 = 1 - \frac{\frac{1}{2} (1-0)^2 [-1.5829 - 3.1136] - (1-4)^2 [-1.5829 - 0]}{(1-0)[-1.5829 - 3.1136] - (1-4)[-1.5829 - 0]} = 1.5055$$

$$f(1.5055) = -1.7691$$



예제 7.3 [2차 보간법] (2/3)

다음으로 황금분할탐색법과 비슷한 전략으로 버릴 점을 결정한다. 새로운 점에서의 함수값은 중간점 x_2 에서의 함수값 보다 작으며, 새로운 x 값이 중간점의 오른쪽에 있으므로 작은 가정값 x_1 을 버린다. 따라서 다음 반복은 아래와 같다.

$$x_1 = 1 \qquad f(x_1) = -1.5829$$

$$x_2 = 1.5055 \qquad f(x_2) = -1.7691$$

$$x_3 = 4 \qquad f(x_3) = 3.1136$$

다시 식 (7.10)에 대입하면,

$$\begin{aligned} x_4 &= 1.5055 - \frac{1}{2} \frac{(1.5055-1)^2 [-1.7691-3.1136] - (1.5055-4)^2 [-1.7691-(-1.5829)]}{(1.5055-1)[-1.7691-3.1136] - (1.5055-4)[-1.7691-(-1.5829)]} \\ &= 1.4903 \longrightarrow \boxed{f(1.4903) = -1.7714} \end{aligned}$$



예제 7.3 [2차 보간법] (3/3)

i	x_1	$f(x_1)$	x_2	$f(x_2)$	x_3	$f(x_3)$	x_4	$f(x_4)$
1	0.0000	0.0000	1.0000	-1.5829	4.0000	3.1136	1.5055	-1.7691
2	1.0000	-1.5829	1.5055	-1.7691	4.0000	3.1136	1.4903	-1.7714
3	1.0000	-1.5829	1.4903	-1.7714	1.5055	-1.7691	1.4256	-1.7757
4	1.0000	-1.5829	1.4256	-1.7757	1.4903	-1.7714	1.4266	-1.7757
5	1.4256	-1.7757	1.4266	-1.7757	1.4903	-1.7714	1.4275	-1.7757



7.2 1차원 최적화 (13/15)

■ MATLAB 함수 : fminbnd

- ◆ 내장함수 fminbnd는 느리지만 신뢰도가 높은 황금분할탐색법과 신뢰도는 낮으나 계산속도가 빠른 2차보간법을 함께 사용한다.

[xmin, fval] = fminbnd(function, x1, x2)

여기서 x = 최소값의 위치

fval = 최소값

function = 함수의 이름

x1, x2 = 구간의 경계



7.2 1차원 최적화 (14/15)

[예제 7.1에 대한 M-파일 : fminbnd]

```
Untitled
File Edit View Text Debug Breakpoints Web Window Help
>> g=9.81; v0=55; m=80;c=15;z0=100;
>> z=@(t) -(z0+m/c*(v0+m*g/c)*(1-exp(-c/m*t))-m*g/c*t);
>> [x,f]=fminbnd(z,0,8)

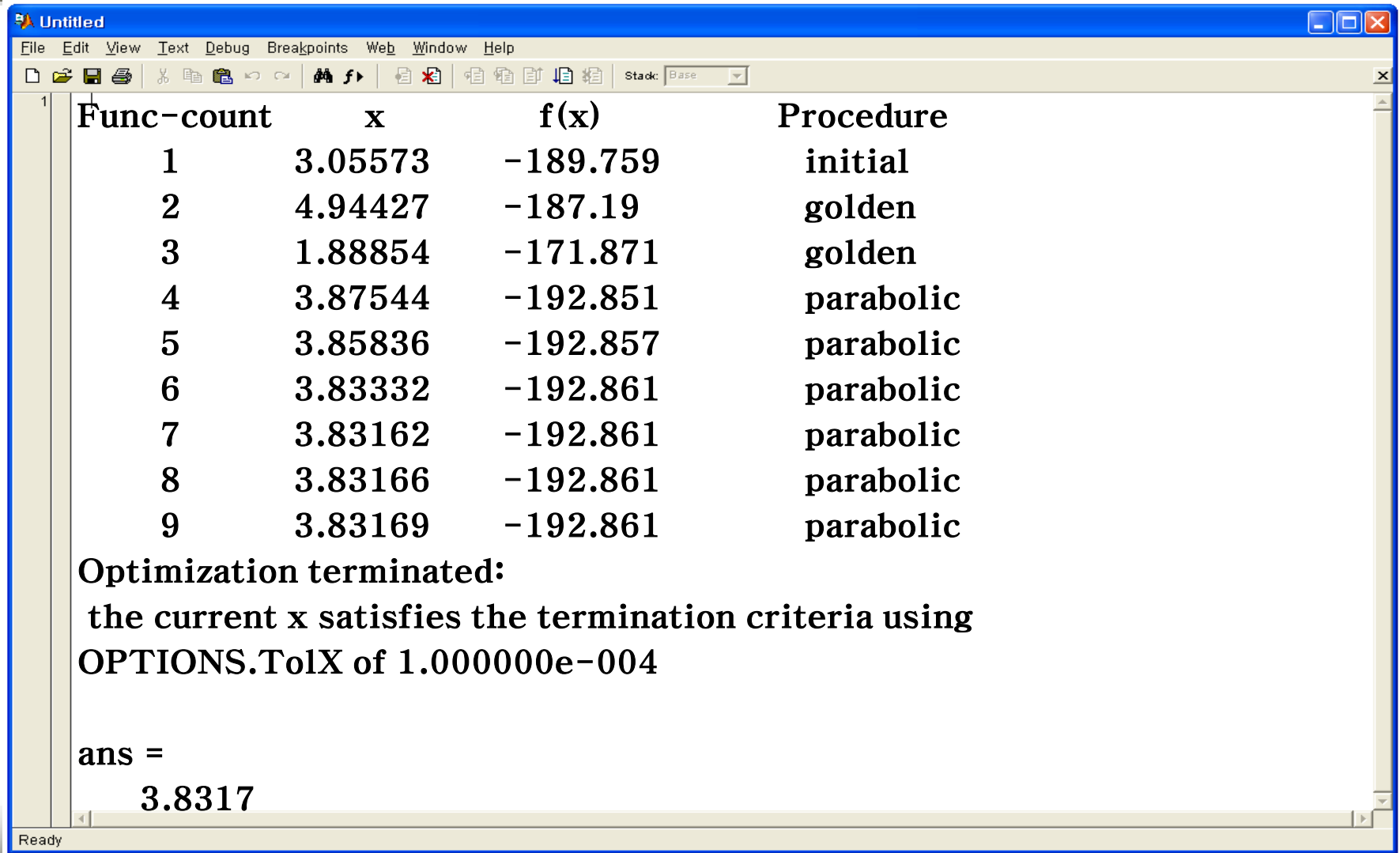
x =
    3.8317
f =
   -192.8609
Ready
```

옵션 매개변수를 optimset을 사용하여 지정할 수 있다. 예를 들어 위 계산에 대한 자세한 과정을 나타낼 수 있다.

```
Untitled
File Edit View Text Debug Breakpoints Web Window Help
>> options = optimset ('display', 'iter') ;
>> fminbnd(z, 0, 8, options)
```



7.2 1차원 최적화 (15/15)



Func-count	x	f(x)	Procedure
1	3.05573	-189.759	initial
2	4.94427	-187.19	golden
3	1.88854	-171.871	golden
4	3.87544	-192.851	parabolic
5	3.85836	-192.857	parabolic
6	3.83332	-192.861	parabolic
7	3.83162	-192.861	parabolic
8	3.83166	-192.861	parabolic
9	3.83169	-192.861	parabolic

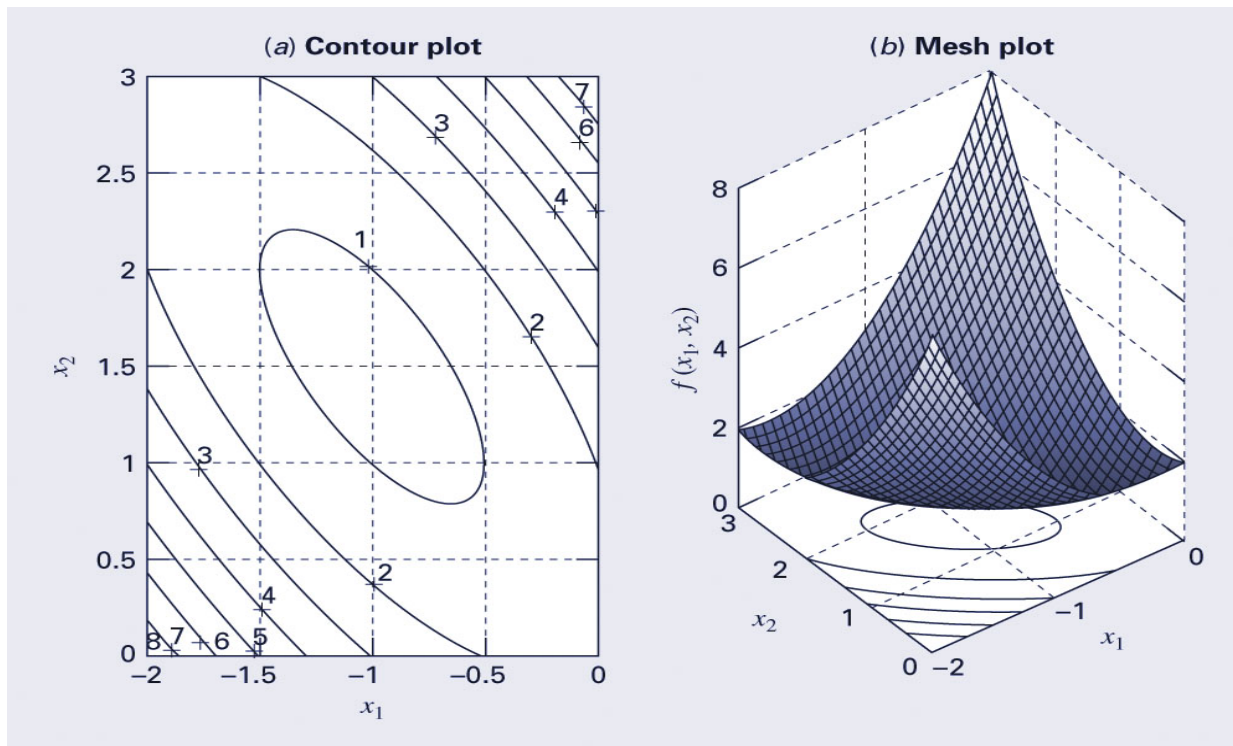
Optimization terminated:
the current x satisfies the termination criteria using
OPTIONS.TolX of 1.000000e-004

ans =
3.8317



7.3 다차원 최적화 (1/3)

2차원 함수의 시각적 형상은 MATLAB의 그래픽 기능을 이용하여 손쉽게 시각화할 수 있다.



예제 7.4 [2차원 함수의 시각화] [1/2]

Q. MATLAB의 그래픽 기능을 사용하여, $-2 \leq x_1 \leq 0$, $0 \leq x_2 \leq 3$ 의 범위 내에서 다음의 함수와 그 최소값을 시각적으로 나타내라.

$$f(x_1, x_2) = 2 + x_1 - x_2 + 2x_1^2 + 2x_1x_2 + x_2^2$$



예제 7.4 [2차원 함수의 시각화] [2/2]

풀이)

```
Untitled
File Edit View Text Debug Breakpoints Web Window Help
x y z Cut Copy Paste Undo Redo Find Replace Stack: Base
1 x=linspace(-2,0,40);y=linspace(0,3,40);
[X,Y] = meshgrid(x,y);
Z=2+X-Y+2*X.^2+2*X.*Y+Y.^2;
subplot (1,2,1);
cs=contour(X,Y,Z); clabel(cs);
xlabel('x_1'); ylabel('x_2');
title('(a) Contour plot') ;grid;
subplot(1,2,2);
cs=surfc(X,Y,Z);
zmin=floor(min(Z));
zmax=ceil(max(Z));
xlabel('x_1'); ylabel ('x_2'); zlabel('f(x_1,x_2)');
title ('(b) Mesh plot') ;
Ready
```



7.3 다차원 최적화 (2/3)

■ MATLAB 함수 : fminsearch

- MATLAB은 다차원 함수의 최소값을 구하는 fminsearch라는 함수를 가진다.

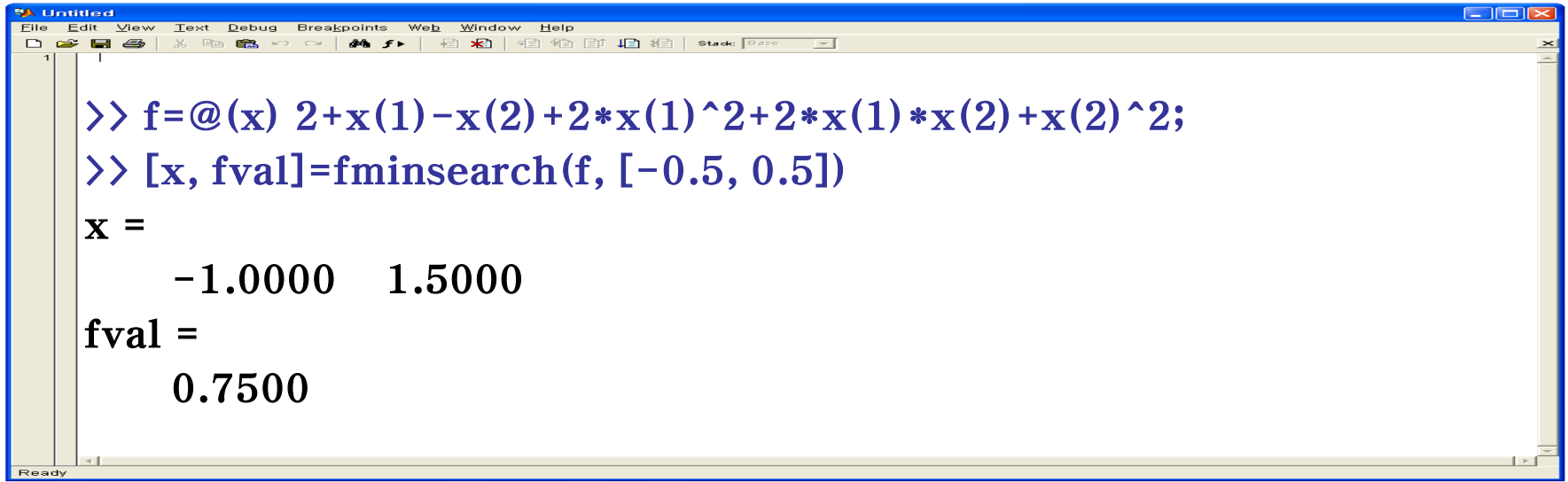
[xmin, fval] = fminsearch(function, x1, x2)

여기서 x = 최소값의 위치
fval = 최소값
function = 함수의 이름
 $x1, x2$ = 구간의 경계



7.3 다차원 최적화 (3/3)

[예제 7.4에 대한 M-파일 : fminsearch]



```
Untitled
File Edit View Text Debug Breakpoints Web Window Help
>> f=@(x) 2+x(1)-x(2)+2*x(1)^2+2*x(1)*x(2)+x(2)^2;
>> [x, fval]=fminsearch(f, [-0.5, 0.5])
x =
    -1.0000    1.5000
fval =
    0.7500
Ready
```

