

# 6장 방정식의 근: 개방법

6.1 단순 고정점 반복법

6.2 Newton-Raphson법

6.3 할선법

6.4 MATLAB 함수:fzero

6.5 다항식



# 6장 방정식의 근: 개방법

## ■ 구간법과 개방법의 비교

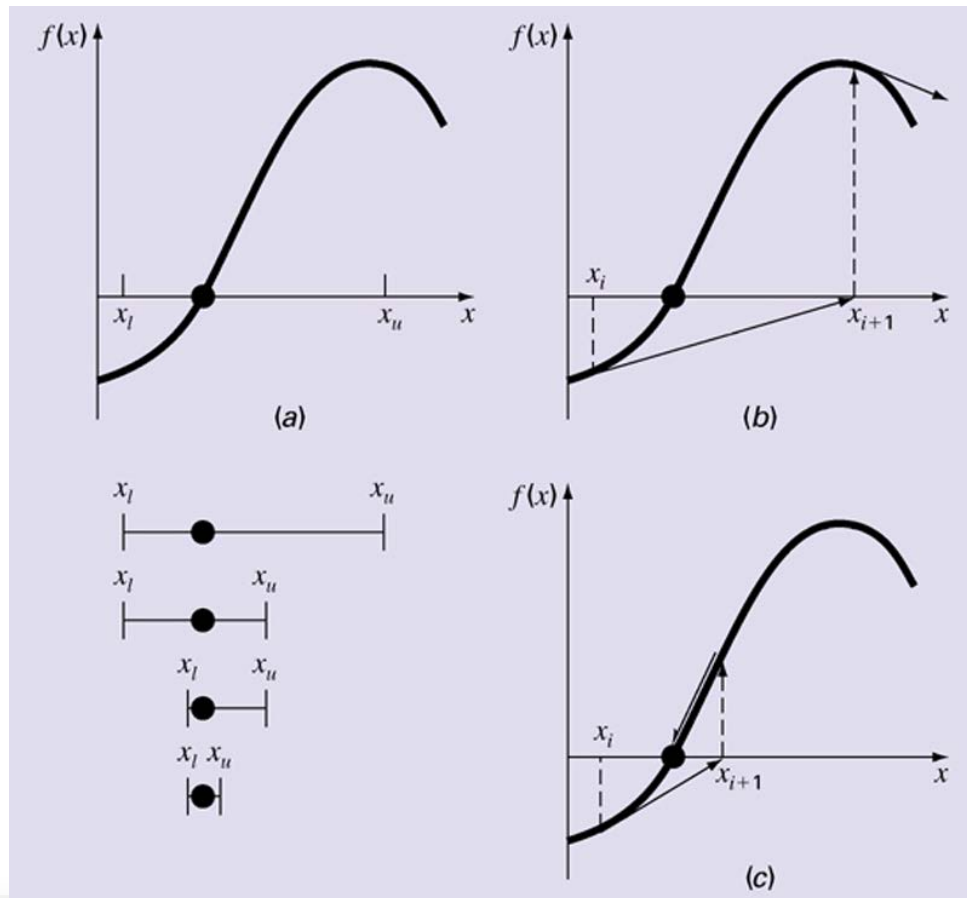


그림 6.1 도식적 비교:

(a)구간법 (b), (c) 개방법.

이분법인 (a)에서는 근이 반드시  $x_l$ 과  $x_u$ 을 포함하는 구간에 존재하나, Newton-Raphson법과 같은 개방법에서는 함수의 형태와 초기값의 설정에 따라 (b)와 같이 발산하거나 (c)와 같이 빠르게 수렴한다.



## 6.1 단순 고정점 반복법 (1/4)

- 단일점 반복법 또는 연속 대입법이라고도 한다.
- $f(x) = 0$  를 정리하여 좌변에  $x$  가 나타나도록 한다.

$$x = g(x)$$

- 대수적 조작이나 양변에  $x$  를 더하면 됨
- 이전 계산단계의  $x$  값을 사용하여 새로운  $x$ 를 예측하는 공식:

$$x_{i+1} = g(x_i)$$

- 근사오차 
$$\varepsilon_a = \left| \frac{x_{i+1} - x_i}{x_{i+1}} \right| 100\%$$



## 예제 6.1 [1/2]

Q. 고정점 반복법을 이용하여  $f(x) = e^{-x} - x$  의 근을 구하라.

풀이)

함수를 다음과 같이 정리할 수 있다.

$$x_{i+1} = e^{-x_i}$$

초기 가정값으로  $x_0 = 0$ 을 사용하여 반복 계산을 한다.



## 예제 6.1 [2/2]

- 참고로 근의 참값은 0.56714329이다.

$i$	$x_i$	$ \varepsilon_a $ (%)	$ \varepsilon_t $ (%)	$ \varepsilon_t _i /  \varepsilon_t _{i-1}$
0	0.0000		100.000	
1	1.0000	100.000	76.322	0.763
2	0.3679	171.828	35.135	0.460
3	0.6922	46.854	22.050	0.628
4	0.5005	38.309	11.755	0.533
5	0.6062	17.447	6.894	0.586
6	0.5454	11.157	3.835	0.556
7	0.5796	5.903	2.199	0.573
8	0.5601	3.481	1.239	0.564
9	0.5711	1.931	0.705	0.569
10	0.5649	1.109	0.399	0.566

- 주목할 사항은 참 백분율 상대오차가 이전 단계의 오차에 거의 비례한다는 것이다.

(대략 0.5 ~ 0.6배)  $\rightarrow$  선형 수렴 (고정점 반복법의 특징)  $\varepsilon_{t,i+1} \propto \varepsilon_{t,i}$



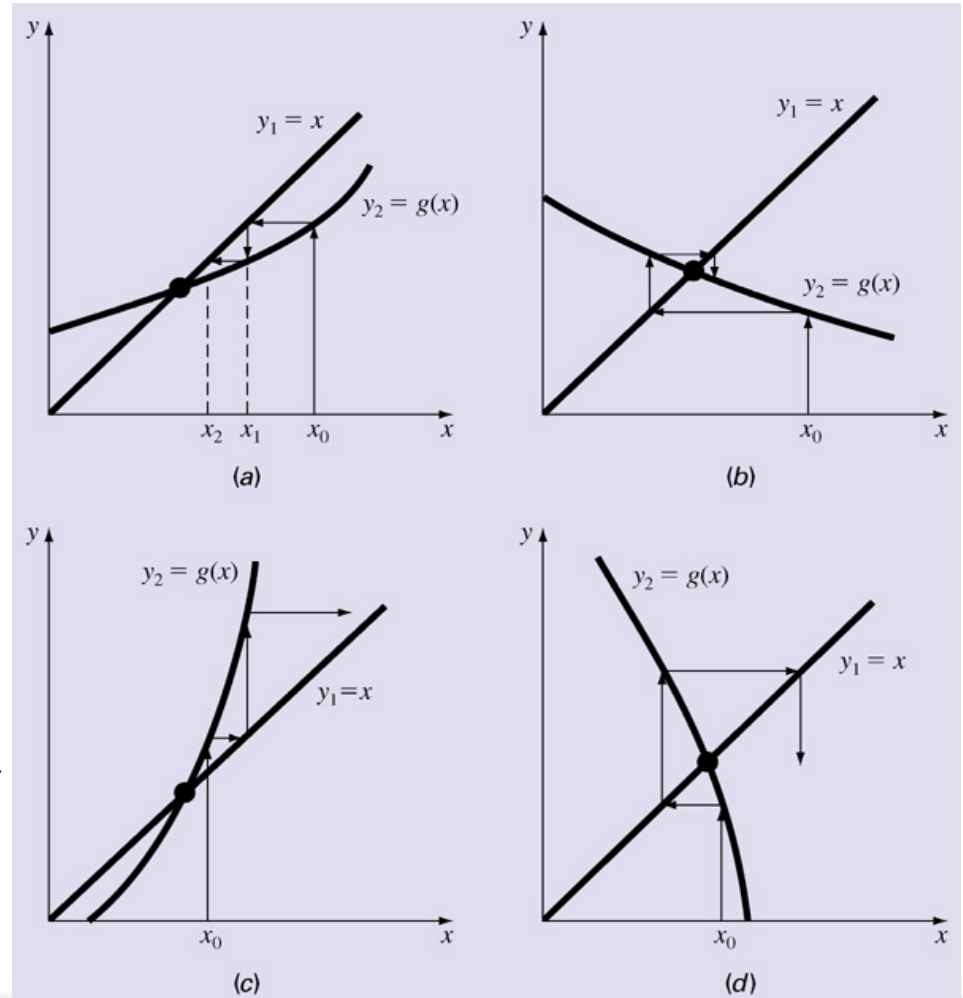
# 6.1 단순 고정점 반복법 (2/4)

## ■ 수렴 가능성

- 수렴
- 발산

그림 6.3 (a)와 (b)는 수렴하는 경우, (c)와 (d)는 발산하는 경우를 나타낸다. (a)와 (c)는 단조형태, (b)와 (d)는 진동 또는 나선형태라고 한다.

$|g'(x)| < 1$ 인 경우에 수렴한다.





# 6.1 단순 고정점 반복법 (3/4)

고정점 반복법에서는  $|g'(x)| < 1$  일 때,

즉  $g(x)$  의 기울기의 절대값이  $f(x) = x$ 의 기울기보다 작을 때 수렴

고정점 반복법 공식은  $x_{i+1} = g(x_i)$

정해를  $x_r$  라 하면  $x_r = g(x_r)$

위의 두 식에서  $x_r - x_{i+1} = g(x_r) - g(x_i)$

도함수의 평균값정리로부터 (  $a = x_i$ 와  $b = x_r$ 로 놓았음)

$$g(x_r) - g(x_i) = (x_r - x_i)g'(\xi)$$

$$x_r - x_{i+1} = (x_r - x_i)g'(\xi)$$

반복  $i$ 에서 참오차를  $E_{t,i} = x_r - x_i$  라고 정의하면,

$$E_{t,i+1} = g'(\xi)E_{t,i}$$



# 6.1 단순 고정점 반복법 (4/4)

## ■ 결론적으로

- $|g'(x)| < 1$  이면  $\rightarrow$  오차  $\downarrow$ ;  $|g'(x)| > 1$  이면  $\rightarrow$  오차  $\uparrow$
- $g'(x) > 0$  이면  $\rightarrow$  오차는 양  $\rightarrow$  단조형태
- $g'(x) < 0$  이면  $\rightarrow$  오차는 진동형태





# 잠깐 휴식 (1/2)

## ■ 평균값 정리

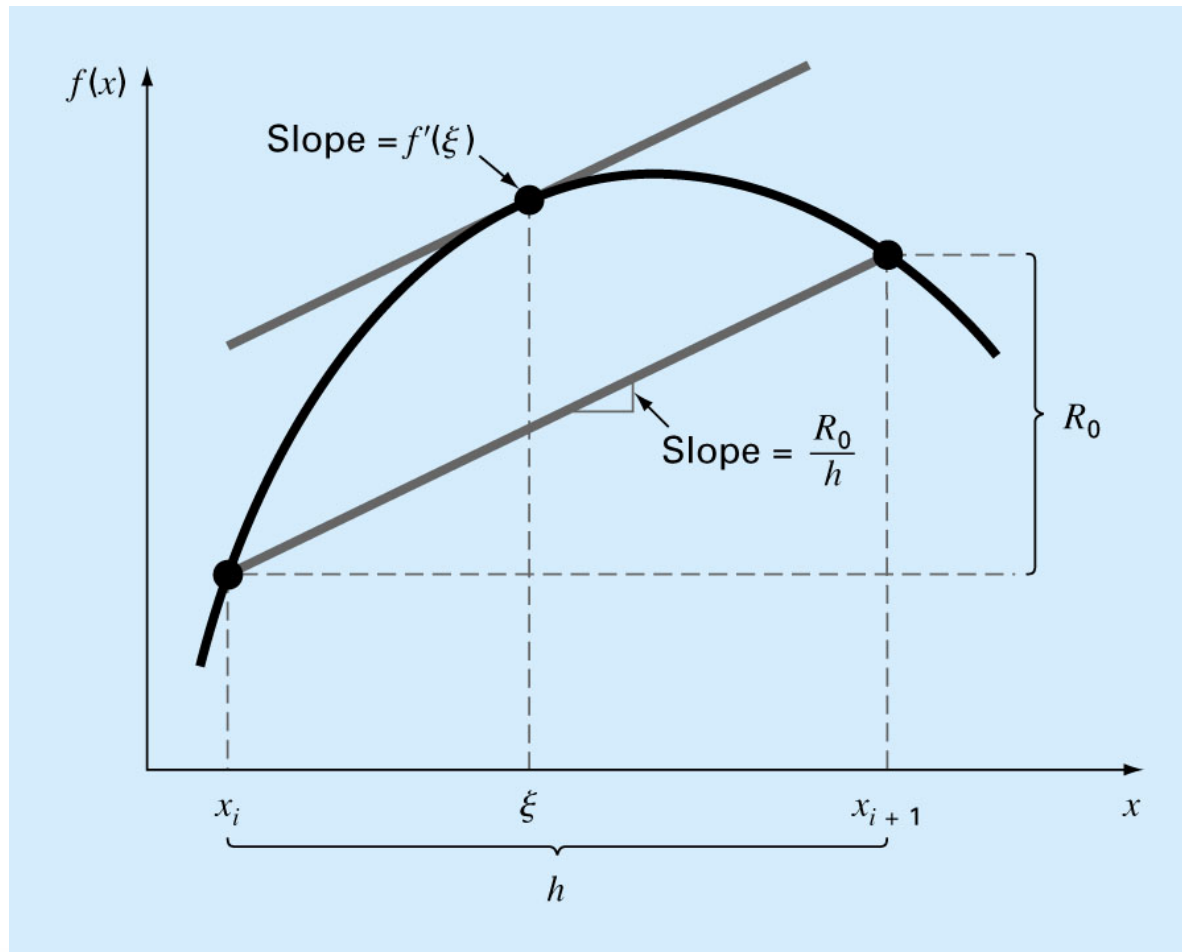
- 함수  $g(x)$ 와 그 일차 도함수가 구간  $a \leq x \leq b$ 에서 연속이면, 그 구간 내에 다음의 관계를 만족하는 점( $x = \xi$ )이 적어도 하나 이상 존재한다.

$$g'(\xi) = \frac{g(b) - g(a)}{b - a}$$

- 두 점  $(a, g(a))$ 와  $(b, g(b))$ 를 잇는 평균기울기와 같은  $g'(\xi)$ 값을 적어도 한 점 이상에서 갖는다.



## 잠깐 휴식 (2/2)



## 6.2 Newton-Raphson법 (1/3)

### ■ 가장 폭넓게 사용되는 공식

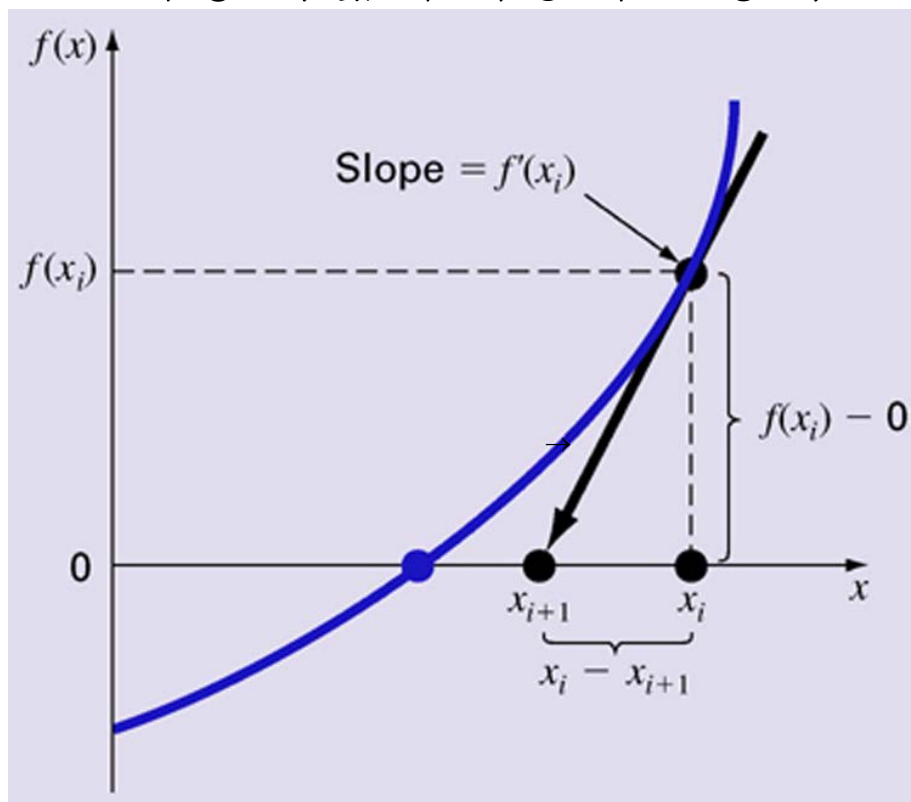


그림 6.4 Newton-Raphson법의 원리.  
 $x_i$ 에서의 함수의 접선  $f'(x)$ 이 근의 근사값  $x_{i+1}$ 을 추정하기 위하여  $x$  축까지 연장된다.

근의 초기 가정 값 =  $x_i$

- 점  $[x_i, f(x_i)]$ 에서의 접선을 연장
- 보다 개선된 근으로  $x$ 축과 만나는 점을 선택

$$\bullet \quad f'(x_i) = \frac{f(x_i) - 0}{x_i - x_{i+1}}$$

$$\rightarrow x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

Newton-Raphson 공식



## 예제 6.2 [1/2]

Q. Newton-Raphson 법을 사용해서  $f(x) = e^{-x} - x$  의 근을 추정하라. 초기 가정은  $x_0 = 0$ 이다.

풀이)

$f'(x) = -e^{-x} - 1$  이므로 공식은

$$x_{i+1} = x_i - \frac{e^{-x_i} - x_i}{-e^{-x_i} - 1}$$



## 예제 6.2 [2/2]

$i$	$x_i$	$ \varepsilon_t $ (%)
0	0	100
1	0.500000000	11.8
2	0.566311003	0.147
3	0.567143165	0.0000220
4	0.567143290	$< 10^{-8}$

- 오차는 이전 단계에서의 오차의 제곱에 비례한다.  
2차적 수렴;  $E_{t,i+1} \propto E_{t,i}^2$
- 중근을 갖는 경우에는 효율이 떨어진다.



# 잠깐 휴식 (1/2)

## ■ 2차적 수렴

Taylor 급수 전개

$$f(x_{i+1}) = f(x_i) + f'(x_i)(x_{i+1} - x_i) + \frac{f''(\xi)}{2!}(x_{i+1} - x_i)^2$$

일차 도함수 이후의 항을 무시하면

$$f(x_{i+1}) \cong f(x_i) + f'(x_i)(x_{i+1} - x_i)$$

$x$  축과 만나는 점을 구하기 위해

$$0 = f(x_i) + f'(x_i)(x_{i+1} - x_i) \quad \text{또는} \quad x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$



## 잠깐 휴식 (2/2)

Taylor 급수 전개에서  $x_{i+1} = x_r$ 을 정해로 가정하면  $f(x_r) = 0$ ,

$$0 = f(x_i) + f'(x_i)(x_r - x_i) + \frac{f''(\xi)}{2!}(x_r - x_i)^2$$

오차는  $E_{t,i+1} = x_r - x_{i+1}$ 이므로 최종적으로

$$0 = f'(x_i)E_{t,i+1} + \frac{f''(\xi)}{2!}E_{t,i}^2$$

수렴하는 경우에는,

$x_i$ 와  $\xi$ 를 정해  $x_r$ 과 같다고 근사할 수 있으므로

$$E_{t,i+1} = \frac{-f''(x_r)}{2f'(x_r)}E_{t,i}^2$$





## 예제 6.3 [1/2]

Q. Newton-Raphson 법을 사용해서  $f(x) = x^{10} - 1$  의  
양의 근을 구하라. 단,  $x_0 = 0.5$

풀이)

$$x_{i+1} = x_i - \frac{x_i^{10} - 1}{10x_i^9}$$

$i$	$x_i$	$ \varepsilon_a $ (%)
0	0.5	
1	51.65	99.032
2	46.485	11.111
3	41.8365	11.111
4	37.65285	11.111
⋮		
40	1.002316	2.130
41	1.000024	0.229
42	1	0.002



## 예제 6.3 [2/2]

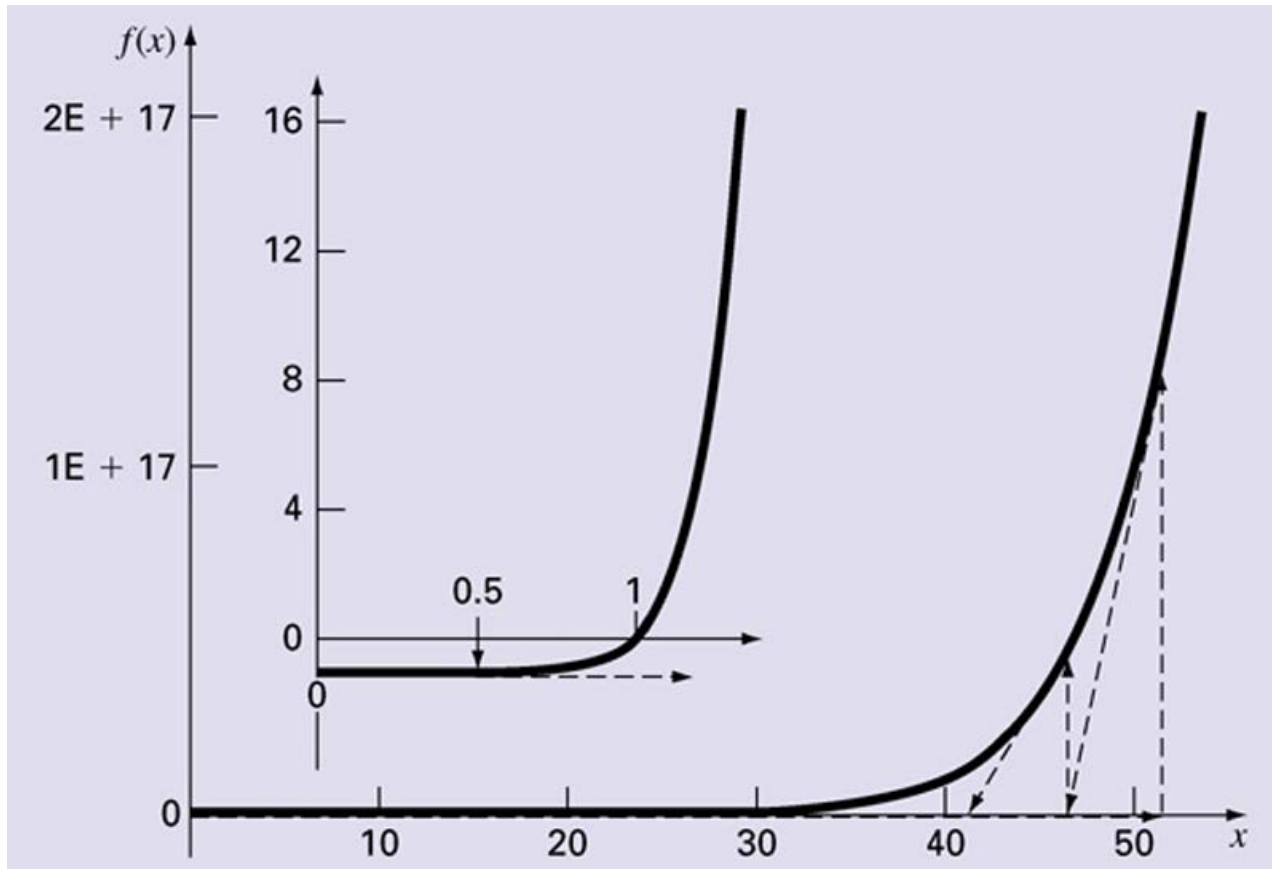
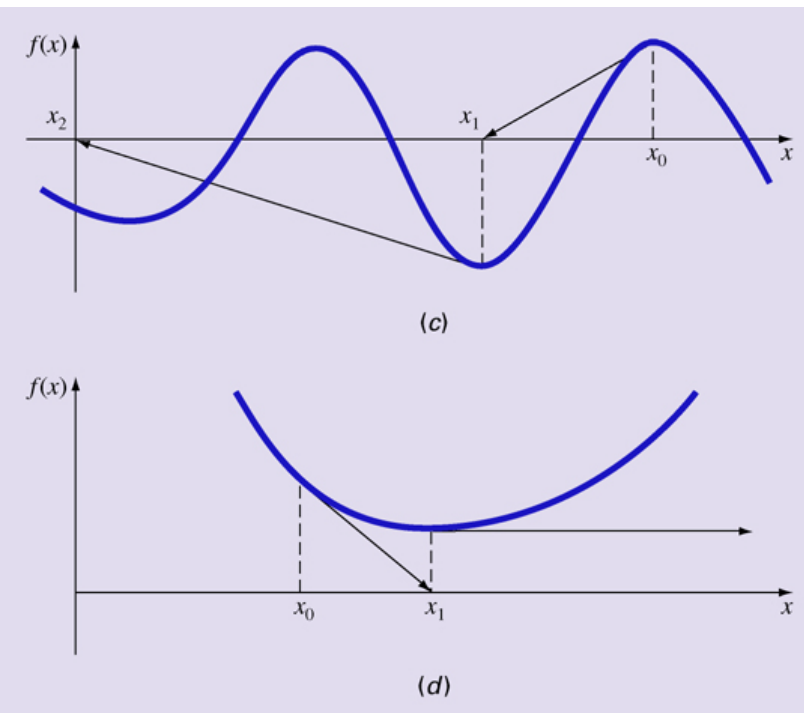
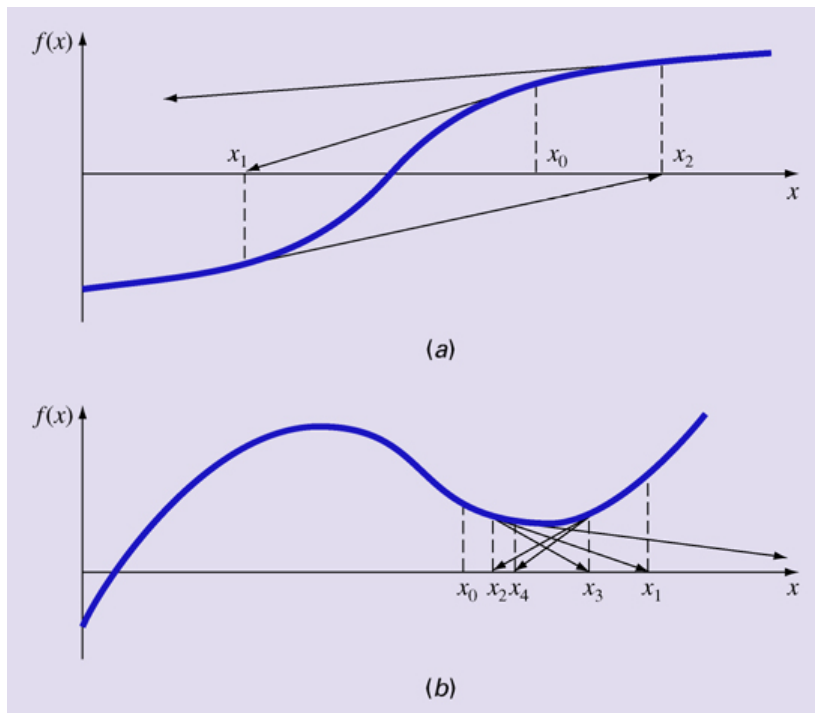


그림 6.5 느리게 수렴하는 Newton-Raphson법의 도식적 묘사.  
삽입된 그림은 초기에 0에 가까운 기울기가 어떻게 해를 근으로부터  
멀리 보내고 있는가를 보여준다. 그러므로 해는 매우 느리게 근에 수렴한다.



## 6.2 Newton-Raphson법 (2/3)

- Newton-Raphson 법이 느리게 수렴되지 않는 네 가지 경우

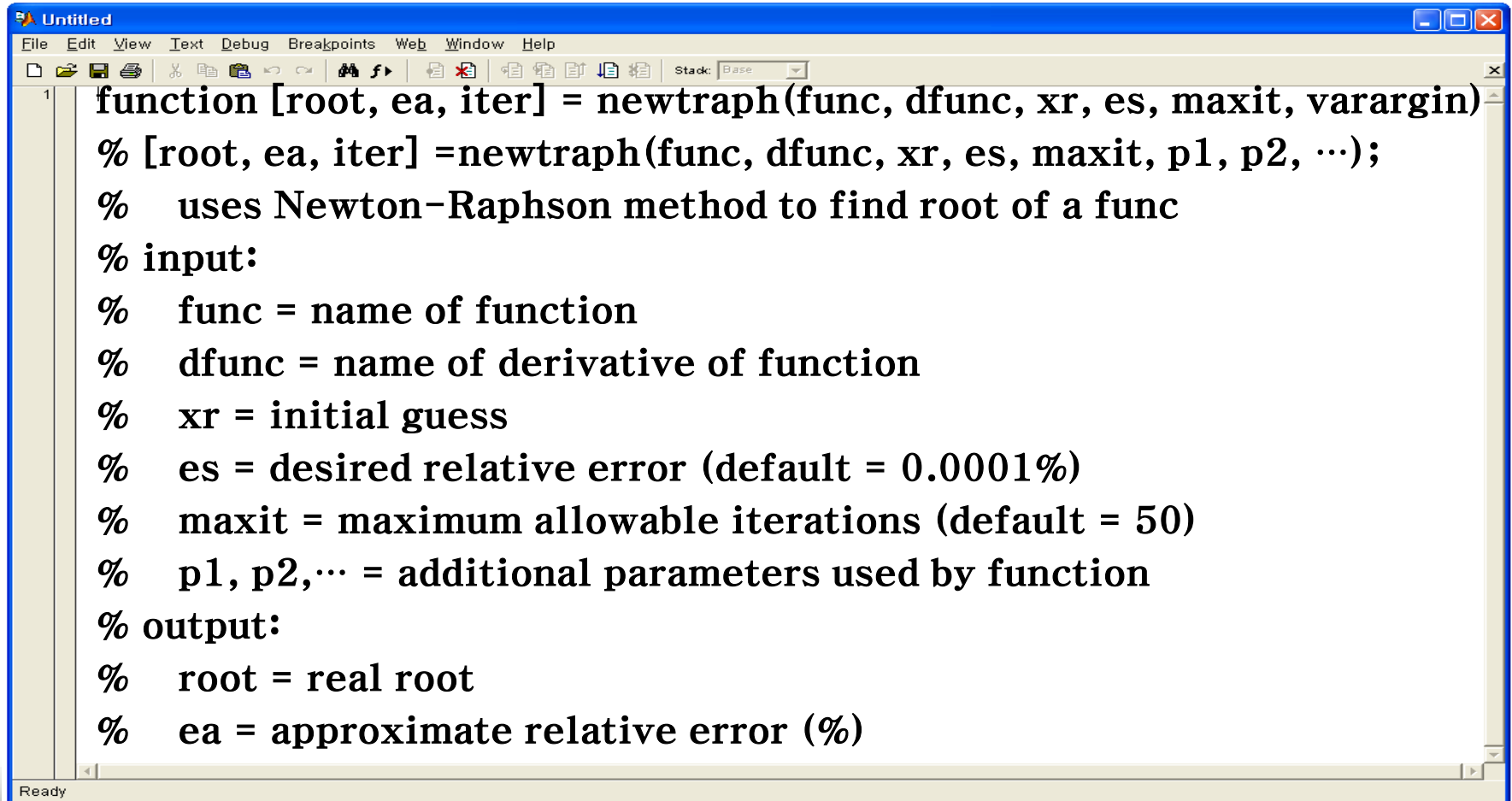


- 기울기가 0 [ $f'(x) = 0$ ] 이면 N-R 공식에서 0으로 나누는 경우가 발생
- N-R 법의 수렴 ~ ① 함수의 성질 ② 초기가정의 정확도



## 6.2 Newton-Raphson법 (3/3)

[An M-file to implement the Newton-Raphson method]



```
function [root, ea, iter] = newtraph(func, dfunc, xr, es, maxit, varargin)
% [root, ea, iter] = newtraph(func, dfunc, xr, es, maxit, p1, p2, ...);
% uses Newton-Raphson method to find root of a func
% input:
% func = name of function
% dfunc = name of derivative of function
% xr = initial guess
% es = desired relative error (default = 0.0001%)
% maxit = maximum allowable iterations (default = 50)
% p1, p2, ... = additional parameters used by function
% output:
% root = real root
% ea = approximate relative error (%)
```



## 6.2 Newton-Raphson법 (3/3)

[An M-file to implement the Newton-Raphson method]

```
Untitled
File Edit View Text Debug Breakpoints Web Window Help
% 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
iter = 0;
while (1)
    xrold = xr;
    xr = xr - feval(func,xr)/feval(dfunc,xr);
    iter = iter +1;
    if xr ~= 0, ea = abs((xr - xrold)/xr) * 100; end
    if ea <= es | iter >= maxit, break, end
end
root = xr;
```



## 6.3 할선법 (1/2)

### ■ N-R 법에서 도함수의 표현을 없앤 방법

N-R 법에서 나타나는 도함수를 후향제차분으로 근사시키면

$$f'(x_i) \cong \frac{f(x_{i-1}) - f(x_i)}{x_{i-1} - x_i} \quad \rightarrow \quad x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

$$x_{i+1} = x_i - \frac{f(x_i)(x_{i-1} - x_i)}{f(x_{i-1}) - f(x_i)}$$

- $x$ 에 대해 두 개의 초기값이 필요
- 초기값 사이에서  $f(x)$ 의 부호가 바뀔 필요는 없음  
→ 개방법



## 6.3 할선법 (2/2)

- 또 다른 방법으로 독립변수에 약간의 변동을 주면,

$$f'(x_i) \cong \frac{f(x_i + \delta x_i) - f(x_i)}{\delta x_i}$$

반복계산식

$$x_{i+1} = x_i - \frac{\delta x_i f(x_i)}{f(x_i + \delta x_i) - f(x_i)} \quad \text{수정된 할선법}$$





## 예제 6.5 [1/2]

Q. 수정된 할선법으로 항력계수가  $0.25 \text{ kg/m}$ 일 때 자유낙하 4초 후의 속도가  $36 \text{ m/s}$ 가 되도록 변지점프하는 사람의 질량을 구하라. 중력가속도는  $9.81 \text{ m/s}^2$ 이다. 질량의 초기가정으로  $50 \text{ kg}$ 으로 놓고, 변동량을  $10^{-6}$ 으로 잡아라.

풀이)

$$\begin{aligned} \text{첫 번째 반복에 대해서 } x_0 &= 50 & f(x_0) &= -4.57938708 \\ x_0 + \delta x_0 &= 50.00005 & f(x_0 + \delta x_0) &= -4.579381118 \\ x_1 &= 50 - \frac{10^{-6}(50)(-4.57938708)}{-4.579381118 - (-4.57938708)} \\ &= 88.39931 (|\varepsilon_t| = 38.1\%; |\varepsilon_a| = 43.4\%) \end{aligned}$$



## 예제 6.5 [2/2]

두 번째 반복에 대해서

$$x_1 = 88.39931$$

$$f(x_1) = -1.69220771$$

$$x_1 + \delta x_1 = 88.39940$$

$$f(x_1 + \delta x_1) = -1.692203516$$

$$\begin{aligned} x_2 &= 88.39931 - \frac{10^{-6}(88.39931)(-1.69220771)}{-1.692203516 - (-1.69220771)} \\ &= 124.08970 (|\varepsilon_t| = 13.1\%; |\varepsilon_a| = 28.76\%) \end{aligned}$$

$i$	$x_i$	$ \varepsilon_a $ (%)	$ \varepsilon_t $ (%)
0	50.0000	64.971	
1	88.3993	38.069	43.438
2	124.0897	13.064	28.762
3	140.5417	1.538	11.706
4	142.7072	0.021	1.517
5	142.7376	$4.1 \times 10^{-6}$	0.021
6	142.7376	$3.4 \times 10^{-12}$	$4.1 \times 10^{-6}$



## 6.4 MATLAB 함수: fzero (1/3)

- fzero는 단일 방정식의 실근을 구할 때 구간법과 개방법의 장점을 만족하도록 설계되었다.

[fzero 구문]

fzero(function, x0)

여기서 function = 함수의 이름  
 $x_0$  = 초기 가정값



## 6.4 MATLAB 함수: fzero (2/3)

- $f(x) = x^2 - 9$  의 근을 MATLAB으로 구해보자.

```
Command Window
File Edit View Web Window Help

>> x=fzero(@(x) x^2-9, -4) %음의 근
x =
    -3

>> x=fzero(@(x) x^2-9, 4) % 양의 근
x =
     3

>> x=fzero(@(x) x^2-9, 0)
x =
    -3

>> x=fzero(@(x) x^2-9, [0 4]) % 확실히 양의 근
x =
     3

Ready
```



## 6.4 MATLAB 함수: fzero (3/3)

- fzero는 단일 방정식의 실근을 구할 때 구간법과 개방법의 장점을 만족하도록 설계되었다.

[fzero 보다 복잡한 구문]

```
[x, fx] = fzero(function, x0, options, p1, p2, ...)  
options = optimset('par1', val1, 'par2', val2, ...)
```

여기서 display = 모든 반복에 대해 자세한 레코드를 표시하기 위한 매개변수로 'iter'로 지정.

tolx = x 에 대한 종료 허용치를 지정하는 매개변수로 양의 스칼라 값을 사용



## 예제 6.6 (fzero와 optimset) [1/2]

Q. 예제 6.3의 의  $f(x) = x^{10} - 1$  근을 구하는 문제를 optimset과 fzero로 풀어라.

```
Command Window
File Edit View Web Window Help

>> options = optimset('display','iter','tolx',0.00001);
>> [x,fx] = fzero(@(x) x^10-1, 0.5, options)

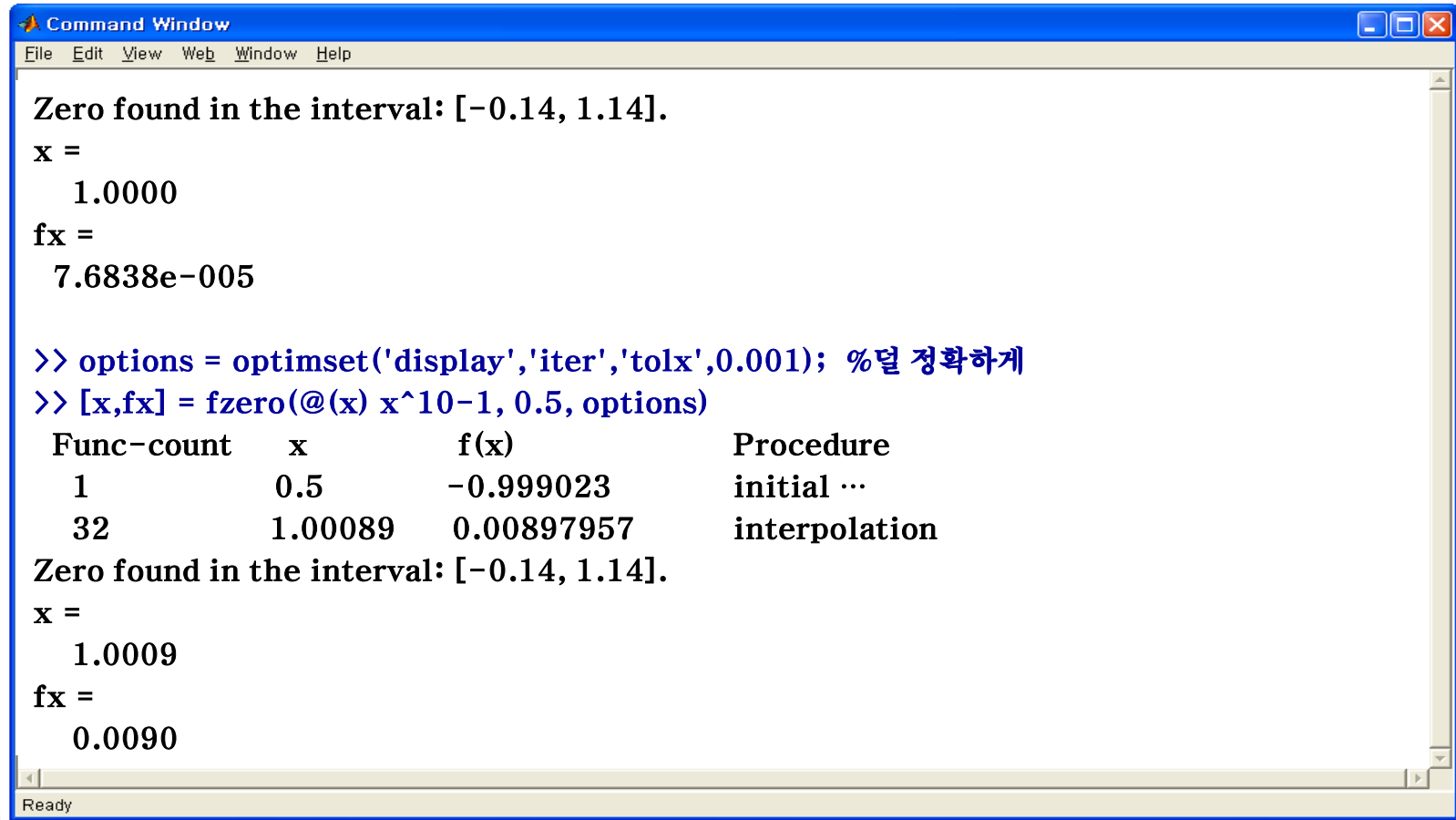
Func-count    x          f(x)          Procedure
    1         0.5        -0.999023      initial
    2        0.485858    -0.999267      search
    ...         ...         ...         ...
   25         1.14         2.70722       search
Looking for a zero in the interval [-0.14, 1.14]
   26        0.205272     -1            interpolation
   27        0.672636    -0.981042      bisection
   28        0.906318    -0.626056      bisection
   29        1.02316     0.257278      bisection
   30        0.989128    -0.103551      interpolation
   31        0.998894    -0.0110017     interpolation
   32        1.00001     7.68385e-005   interpolation
   33        0.999988    -0.000123159   interpolation

Ready
```



## 예제 6.6 (fzero와 optimset) [2/2]

Q. 예제 6.3의 의  $f(x) = x^{10} - 1$  근을 구하는 문제를 optimset과 fzero로 풀어라.



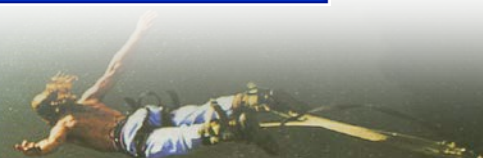
```
Command Window
File Edit View Web Window Help

Zero found in the interval: [-0.14, 1.14].
x =
    1.0000
fx =
    7.6838e-005

>> options = optimset('display','iter','tolx',0.001); %덜 정확하게
>> [x,fx] = fzero(@(x) x^10-1, 0.5, options)

Func-count    x          f(x)          Procedure
    1         0.5        -0.999023      initial ...
   32        1.00089    0.00897957      interpolation
Zero found in the interval: [-0.14, 1.14].
x =
    1.0009
fx =
    0.0090

Ready
```





## 6.5 다항식

- `roots`는 고차다항식의 모든 근을 구할 때 사용하는 내장함수이다.

[root 구문]  
`x = roots(c)`

여기서 `x` = 근을 나타내는 열벡터  
`c` = 다항식의 계수를 나타내는 열벡터

`roots`의 역함수는 `poly`이며 근의 값이 들어가면 다항식의 계수가 나온다.

`c = poly(r)`

여기서 `r` = 근을 나타내는 열벡터  
`c` = 다항식의 계수를 나타내는 열벡터



## 예제 6.7 [1/3]

Q. MATLAB에서 다항식  $f(x) = x^5 - 3.5x^4 + 2.75x^3 + 2.125x^2 - 3.875x + 1.25$ 을 조작해 보자. (note: 실근 0.5, -1.0, 2, 허근  $1 \pm 0.5i$ )

```
Command Window
File Edit View Web Window Help

>> a=[1 -3.5 2.75 2.125 -3.875 1.25]; % 다항식의 계수 행렬
>> polyval(a,1) % 다항식에 1을 대입한 결과
ans =
    -0.2500

>> b=[1 .5 -.5] % (x-0.5)(x+1)을 전개한 다항식의 계수
b =
    1.0000    0.5000   -0.5000

>> b=poly([0.5 -1]) % 두 근 0.5와 -1을 갖는 2차식의 계수
b =
    1.0000    0.5000   -0.5000

Ready
```



## 예제 6.7 [2/3]

```
Command Window
File Edit View Web Window Help

>> [q,r] = deconv(a,b)           % 다항식 a를 다항식 b로 나눈 결과
q =
    1.0000   -4.0000    5.2500   -2.5000   % 몫
r =
    0     0     0     0     0     0       % 나머지
>> x=roots(q)                   % 다항식 q의 모든 근
x =
    2.0000
    1.0000 + 0.5000i
    1.0000 - 0.5000i
>> a=conv(q,b)                  % 다항식 q와 b의 곱
a =
    1.0000   -3.5000    2.7500    2.1250   -3.8750    1.2500

Ready
```



## 예제 6.7 [3/3]

```
Command Window
File Edit View Web Window Help

>> x=roots(a)                                % 다항식 a의 모든 근
x =
    2.0000
   -1.0000
    1.0000 + 0.5000i
    1.0000 - 0.5000i
    0.5000

>> c=poly(x)                                % 모든 근 x를 갖는 다항식의 계수
c =
    1.0000   -3.5000    2.7500    2.1250   -3.8750    1.2500

Ready
```



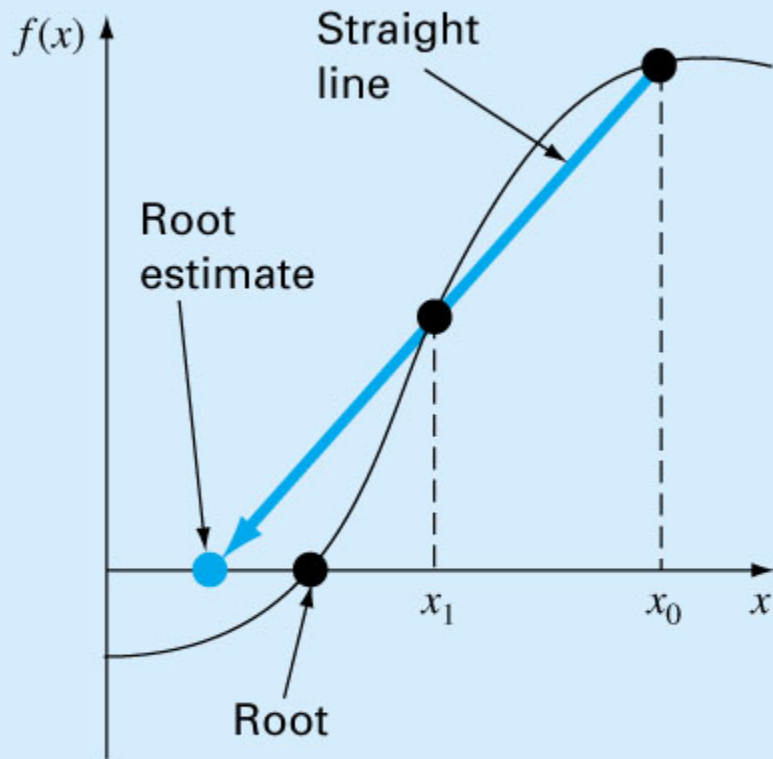
# 잠깐 휴식 (1/6)

## ■ Müller법

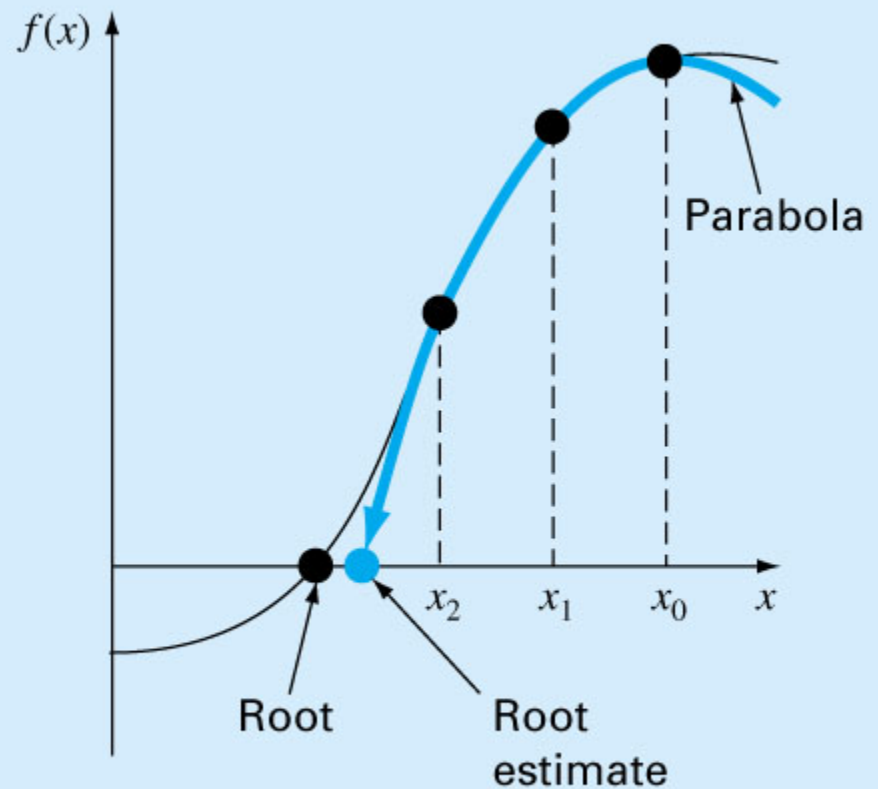
- 두 점을 지나는 직선 대신에 세 점을 지나는 포물선이  $x$  축과 만나는 점을 구함
- 2차 식을 이용하기 때문에 실근뿐만 아니라 복소근도 구할 수 있음
  - Müller법의 주요 장점



## 잠깐 휴식 (2/6)



(a)



(b)



## 잠깐 휴식 (3/6)

편의상 2차 식을 다음과 같이 표시한다.

$$f(x) = a(x - x_2)^2 + b(x - x_2) + c$$

세 점  $[x_0, f(x_0)]$ ,  $[x_1, f(x_1)]$ ,  $[x_2, f(x_2)]$ 을 각각 대입하면

$$f(x_0) = a(x_0 - x_2)^2 + b(x_0 - x_2) + c$$

$$f(x_1) = a(x_1 - x_2)^2 + b(x_1 - x_2) + c$$

$$f(x_2) = a(x_2 - x_2)^2 + b(x_2 - x_2) + c = c$$

여기서  $x_0$ ,  $x_1$ , 그리고  $x_2$ 는 초기 가정값이다.





## 잠깐 휴식 [4/6]

위 식에서 뺄셈을 수행하면 다음과 같다.

$$f(x_0) - f(x_2) = a(x_0 - x_2)^2 + b(x_0 - x_2)$$

$$f(x_1) - f(x_2) = a(x_1 - x_2)^2 + b(x_1 - x_2)$$

다음과 같이 정의하면

$$h_0 = x_1 - x_0 \quad h_1 = x_2 - x_1 \quad \delta_0 = \frac{f(x_1) - f(x_0)}{x_1 - x_0} \quad \delta_1 = \frac{f(x_2) - f(x_1)}{x_2 - x_1}$$

이렇게 해서 얻는 수식은 아래와 같다.

$$(h_0 + h_1)b - (h_0 + h_1)^2 a = h_0 \delta_0 + h_1 \delta_1$$

$$h_1 b - h_1^2 a = h_1 \delta_1$$



# 잠깐 휴식 (5/6)

a와 b에 대해 풀면

$$a = \frac{\delta_1 - \delta_0}{h_1 + h_0} \qquad b = ah_1 + \delta_1 \qquad c = f(x_2)$$

반올림오차를 줄이기 위해 근의 공식을 변형하여 적용하면

$$x = \frac{-2c}{b \pm \sqrt{b^2 - 4ac}} \quad \leftarrow \quad x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

따라서 현재 추정 근( $x_3$ )와 이전 추정 근( $x_2$ )사이의 관계는

$$x_3 - x_2 = \frac{-2c}{b \pm \sqrt{b^2 - 4ac}} \quad \text{또는} \quad x_3 = x_2 + \frac{-2c}{b \pm \sqrt{b^2 - 4ac}}$$



## 잠깐 휴식 [6/6]

부호는  $b$ 의 부호와 일치하게 선택하여  
분모가 가장 큰 값을 갖도록 하였다.  
그로 인해  $x_2$ 에 가까운 근을 추정한다.

오차는 다음과 같이 계산할 수 있다.

$$\varepsilon_a = \left| \frac{x_3 - x_2}{x_3} \right| 100\%$$



## 예제 6.6 [1/2]

Q. Müller법을 이용하여 초기 가정값을  $x_0, x_1, x_2$ 를 각각 4.5, 5.5, 5로 놓고, 방정식의 근을 구하라.  
참고로 이 방정식의 정해는 -3, -1, 4이다.

풀이)

$$f(4.5) = 20.625 \quad f(5.5) = 82.875 \quad f(5) = 48$$

$$h_0 = x_1 - x_0 = 5.5 - 4.5 = 1 \quad \delta_0 = \frac{f(x_1) - f(x_0)}{x_1 - x_0} = \frac{82.875 - 20.625}{5.5 - 4.5} = 62.25$$

$$h_1 = x_2 - x_1 = 5 - 5.5 = -0.5 \quad \delta_1 = \frac{f(x_2) - f(x_1)}{x_2 - x_1} = \frac{48 - 82.875}{5 - 5.5} = 69.75$$

$$a = \frac{\delta_1 - \delta_0}{h_1 + h_0} = \frac{69.75 - 62.25}{-0.5 + 1} = 15 \quad b = ah_1 + \delta_1 = 15(-0.5) + 69.75 = 62.25$$

$$c = f(x_2) = 48$$



## 예제 6.6 [2/2]

$$\sqrt{b^2 - 4ac} = \sqrt{62.25^2 - 4(15)48} = 31.54461$$

$$|62.25 + 31.54461| > |62.25 - 31.54461| \rightarrow \text{양의 부호를 취한다!}$$

$$x_3 = x_2 + \frac{-2c}{b \pm \sqrt{b^2 - 4ac}} = 5 + \frac{-2(48)}{62.25 + 31.54451} = 3.976487$$

$$\varepsilon_a = \left| \frac{x_3 - x_2}{x_3} \right| 100\% = \left| \frac{-1.023513}{3.976487} \right| 100\% = 25.74\%$$

$i$	$x_r$	$ \varepsilon_a $ (%)
0	5	
1	3.976487	25.74
2	4.00105	0.6139
3	4	0.0262
4	4	0.0000119

