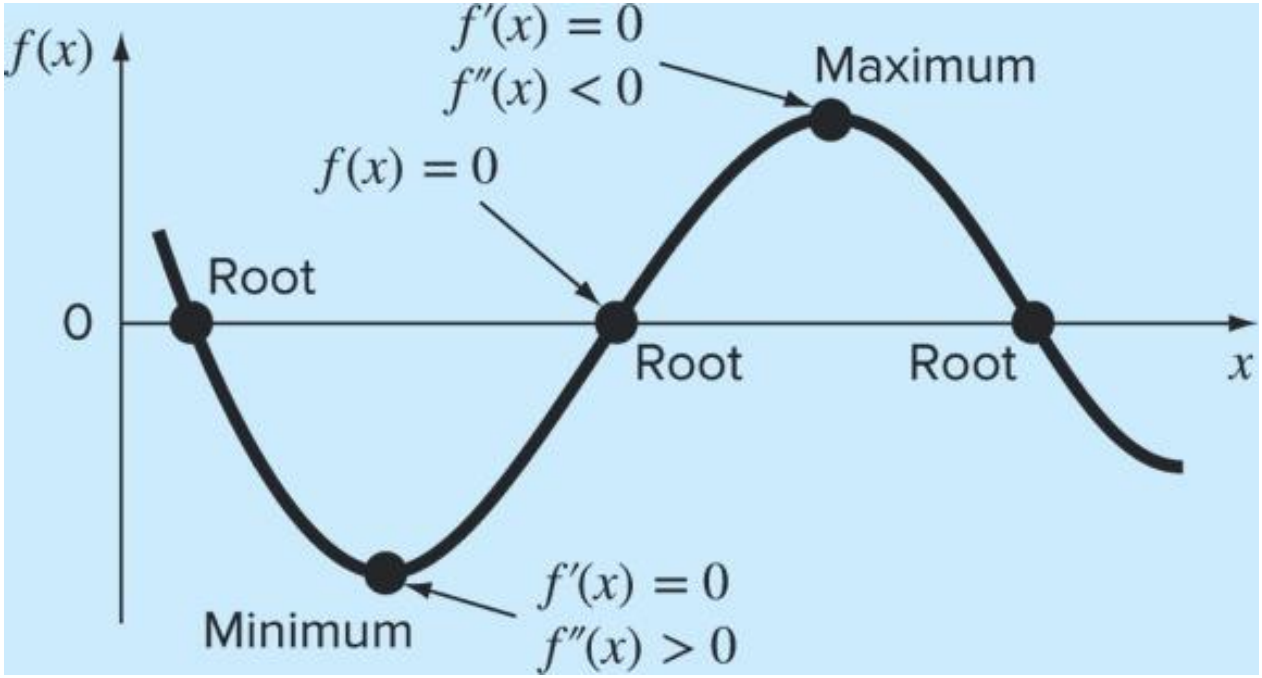
Chapter 7

Optimization

Numerical Methods
Fall 2019

Optimization

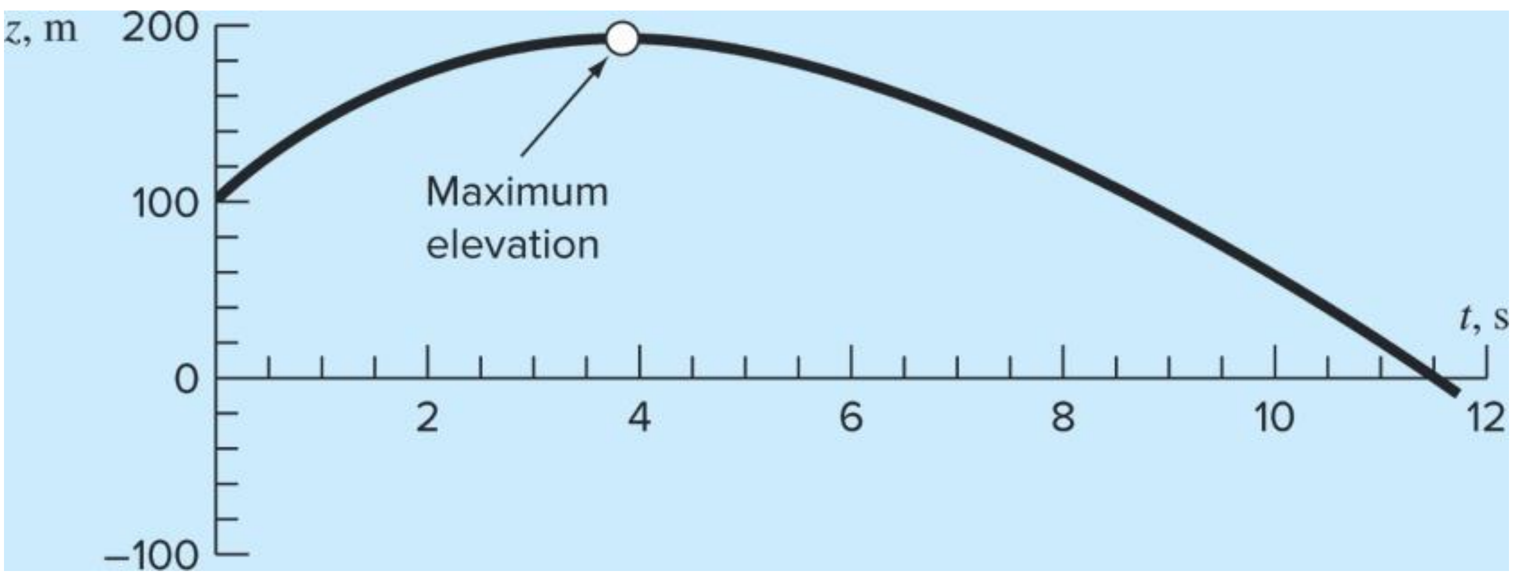
- Optimization (최적화 ?)
- From a mathematical perspective, optimization deals with finding the maxima and minima of a function that depends on one or more variables.



Optimization

- An object can be projected upward at a specified velocity. If it is subject to linear drag, its altitude as a function of time can be computed as

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



Optimization

- Example
  - g = 9.81 m/s^2, z_0 = 100 m, v_0 = 55 m/s, m = 80 kg, and c = 15 kg/s.

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

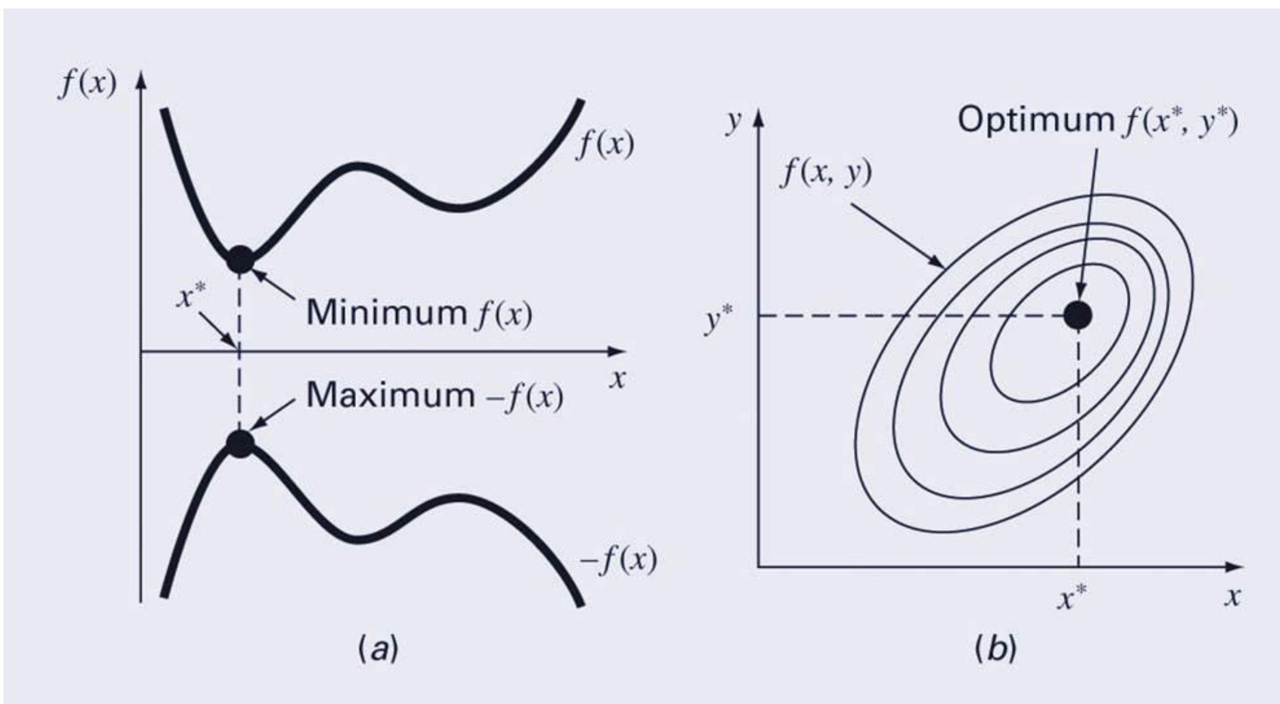
$$t = \frac{m}{c}\ln\left(1 + \frac{cv_0}{mg}\right)$$

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

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

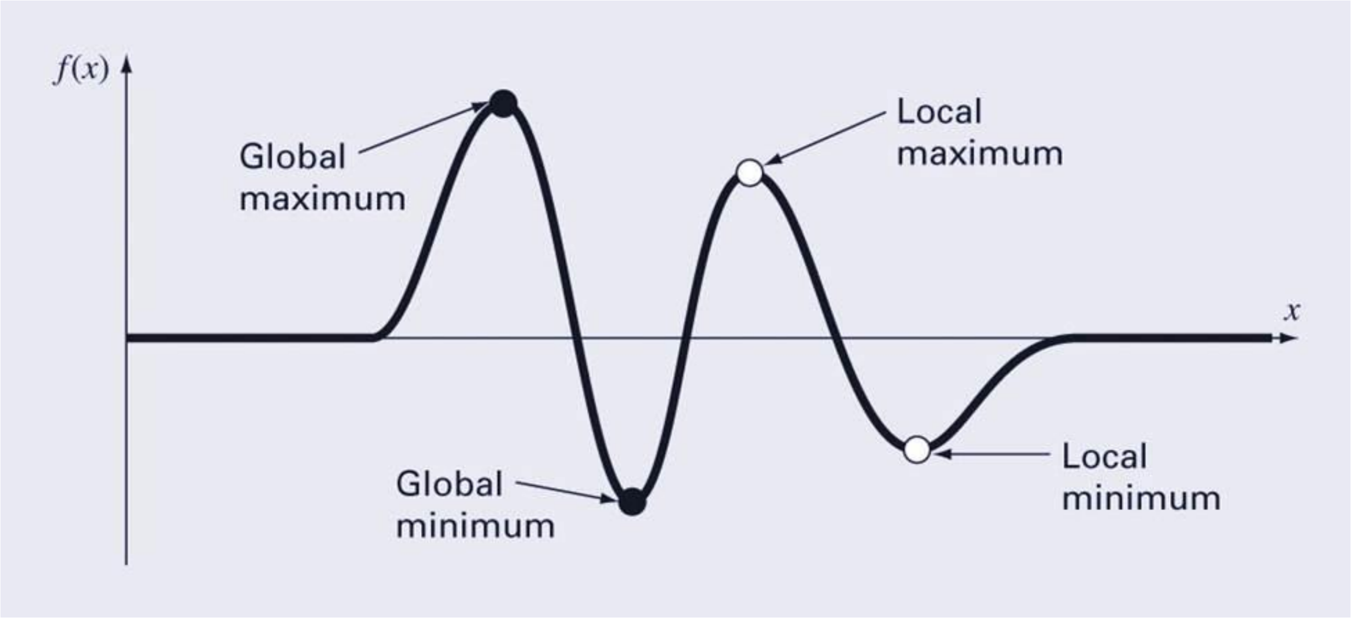
Multidimensional Optimization

- One-dimensional problems involve functions that depend on a single dependent variable—for example, $f(x)$.
- Multidimensional problems involve functions that depend on two or more dependent variables—for example, $f(x,y)$



Global vs. Local

- A *global optimum* represents the very best solution while a *local optimum* is better than its immediate neighbors. Cases that include local optima are called *multimodal*.
- Generally desire to find the global optimum.

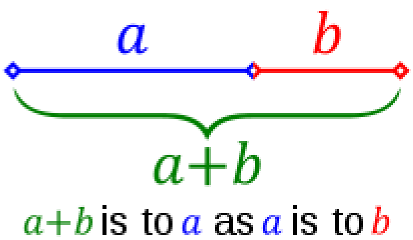


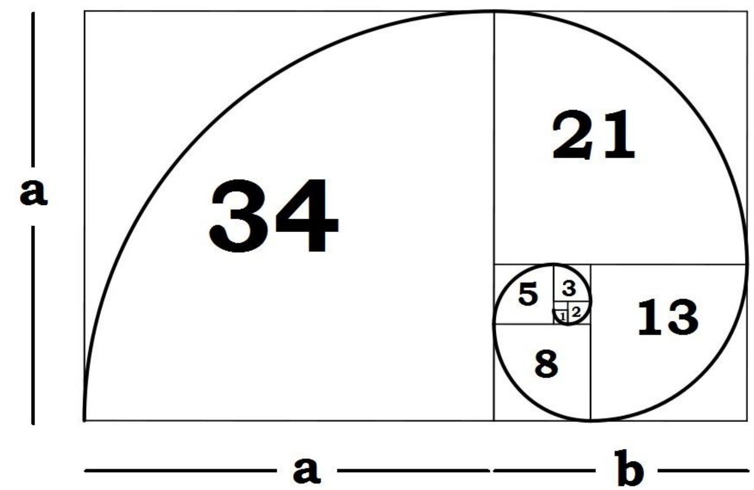
Golden-Section Search

- Search algorithm for finding a minimum on an interval $[x_l, x_u]$ with a *single* minimum (*unimodal* interval)
- Uses the *golden ratio* $\phi=1.6180\ldots$ to determine location of two interior points x_1 and x_2; by using the golden ratio, one of the interior points can be re-used in the next iteration.

Golden ratio

- In mathematics, two quantities are in the **golden ratio** if their ratio is the same as the ratio of their sum to the larger of the two quantities.





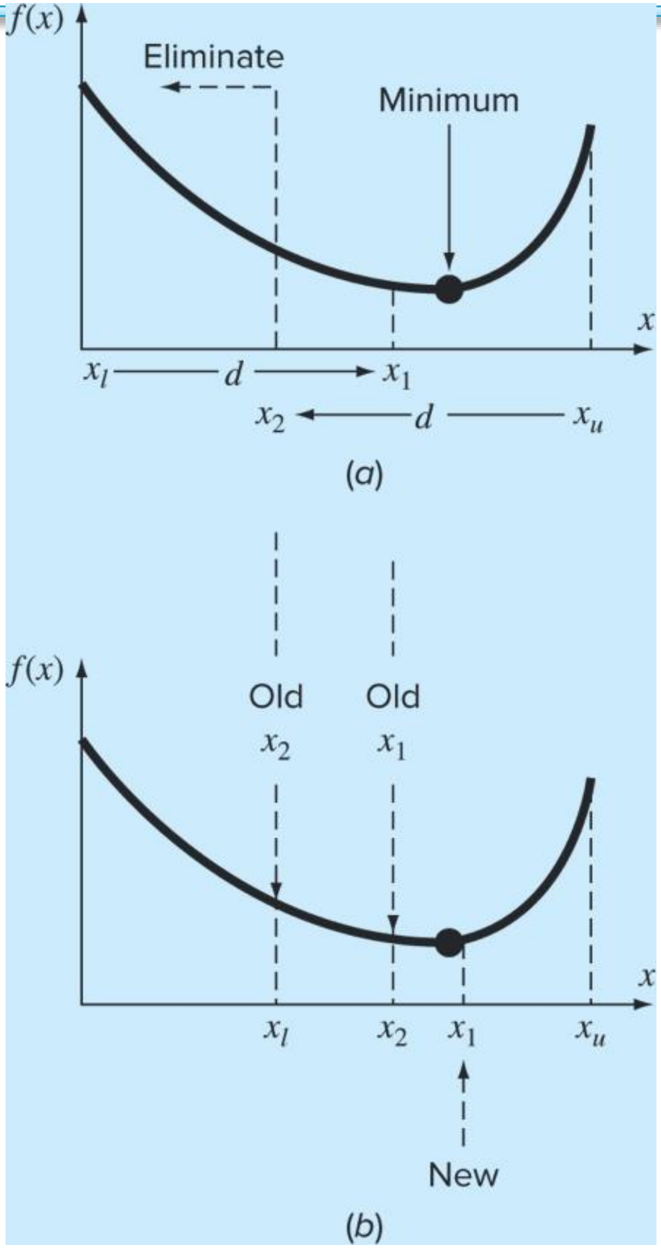
$$\frac{a+b}{a} = \frac{a}{b} \stackrel{\text{def}}{=} \varphi$$

$$\varphi = \frac{1+\sqrt{5}}{2} = 1.6180339887\ldots.$$

Golden-Section Search

$$d = (\varphi - 1)(x_u - xl)$$
$$x_1 = xl + d$$
$$x_2 = xu - d$$

- If $f(x_1) < f(x_2)$, x_2 becomes the new lower limit and x_1 becomes the new x_2 (as in figure).
- If $f(x_2) < f(x_1)$, x_1 becomes the new upper limit and x_2 becomes the new x_1.
- In either case, only one new interior point is needed and the function is only evaluated one more time.



Golden-Section Search

- Example

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

- initial interval: x_l=0, x_u=4

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

$$x_1 = 0 + 2.4721 = 2.4721$$

$$x_2 = 4 - 2.4721 = 1.5279$$

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

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

- New upper bound: x_u = 2.4721
- new x_1 = 1.5279 => f (1.5279) = -1.7647

$$d = 0.61803(2.4721 - 0) = 1.5279$$

$$x_2 = 2.4721 - 1.5279 = 0.9443$$

Golden-Section Search

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

- After the eighth iteration, the minimum occurs at $x = 1.4427$ with a function value of -1.7755.
- The result is converging on the true value of -1.7757 at $x = 1.4276$.
- Error measure

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

Code for Golden-Section Search

```
function [x,fx,ea,iter]=goldmin(f,xl,xu,es,maxit,varargin)
% goldmin: minimization golden section search
%   [x,fx,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
%   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; d = (phi-1)*(xu - xl);
iter = 0; x1 = xl + d; x2 = xu - d;
f1 = f(x1,varargin{:}); f2 = f(x2,varargin{:});
while(1)
  xint= xu - xl;
  if f1 < f2
    xopt = x1; xl = x2; x2 = x1; f2 = f1;
    x1 = xl + (phi-1)*(xu-xl); f1 = f(x1,varargin{:});
  else
    xopt = x2; xu = x1; x1 = x2; f1 = f2;
    x2 = xu - (phi-1)*(xu-xl); f2 = f(x2,varargin{:});
  end
  iter=iter+1;
  if xopt~=0, ea = (2 - phi) * abs(xint / xopt) * 100;end
  if ea <= es | iter >= maxit,break,end
end
x=xopt;fx=f(xopt,varargin{:});
```

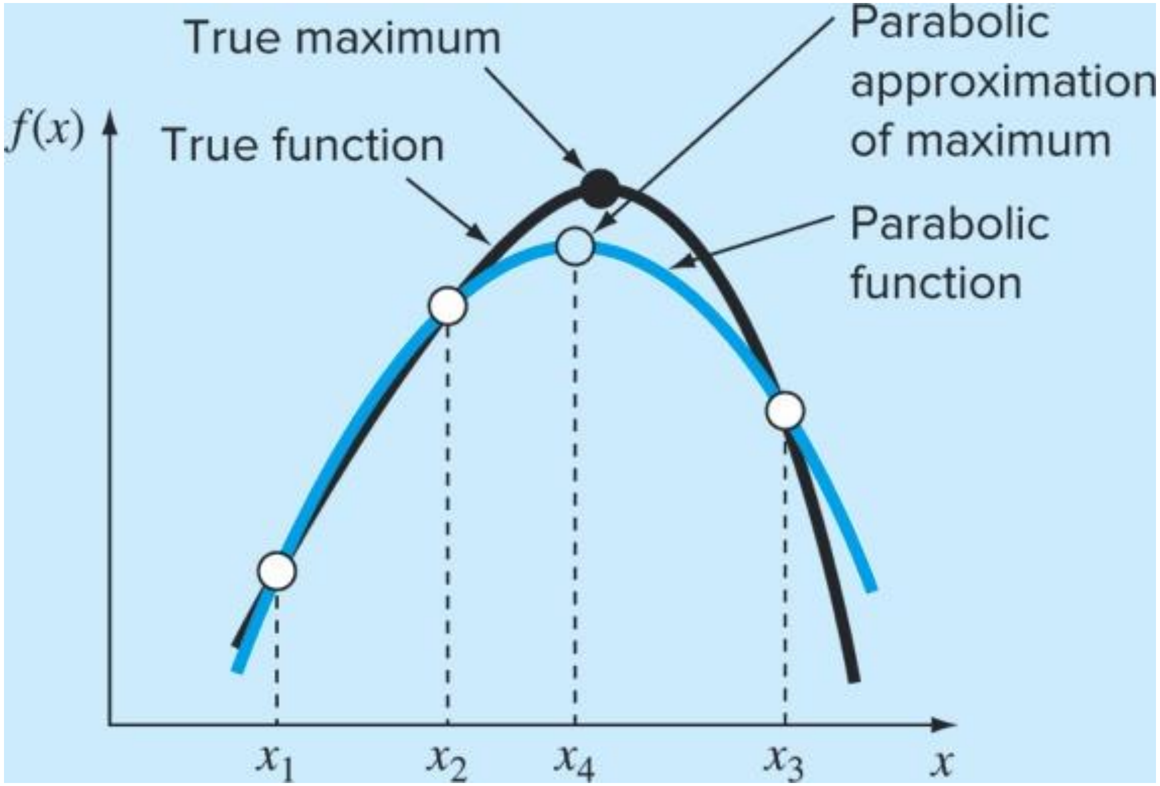
Parabolic Interpolation

- Another algorithm uses parabolic interpolation of three points to estimate optimum location.
- The location of the maximum/minimum of a parabola defined as the interpolation of three points (x_1, x_2, and x_3) is:

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

- The new point x_4 and the two surrounding it (either x_1 and x_2 or x_2 and x_3) are used for the next iteration of the algorithm.

Parabolic Interpolation



Parabolic Interpolation

- Example

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

- initial guesses of $x_1 = 0$, $x_2 = 1$, and $x_3 = 4$.

$$\begin{aligned} x_1 &= 0 & f(x_1) &= 0 \\ x_2 &= 1 & f(x_2) &= -1.5829 \\ x_3 &= 4 & f(x_3) &= 3.1136 \end{aligned}$$

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

- $f(1.5055) = -1.7691$

$$\begin{aligned} x_1 &= 1 & f(x_1) &= -1.5829 \\ x_2 &= 1.5055 & f(x_2) &= -1.7691 \\ x_3 &= 4 & f(x_3) &= 3.1136 \end{aligned}$$

$$x_4 = 1.5055 - \frac{1}{2}\frac{(1.5055-1)^2[-1.7691 - 3.1136] - (1.5055-4)^2[-1.7691 - (-1.5829)]}{(1.5055-1)[-1.7691 - 3.1136] - (1.5055-4)[-1.7691 - (-1.5829)]}$$

$$= 1.4903$$

Parabolic Interpolation

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

- The result is converging rapidly on the true value of -1.7757 at $x = 1.4276$.

Newton–Rapson Method

- In the one-dimensional problem, Newton's (or Newton–Rapson) method attempts to find the roots of f' by constructing a sequence x_n from an initial guess x_0 that converges towards some value x^* satisfying $f'(x^*) = 0$.

Newton–Rapson Method

Suppose that $f(x)$ is approximated by a *quadratic function* at a point x^k (*Taylor series*)

$f(x) = ax^2+bx+c$

$$f\left(x^{k+1}\right) \cong f\left(x^k\right) + f'\left(x^k\right)\left(x^{k+1} - x^k\right) + \frac{1}{2!} f''\left(x^k\right)\left(x^{k+1} - x^k\right)^2$$

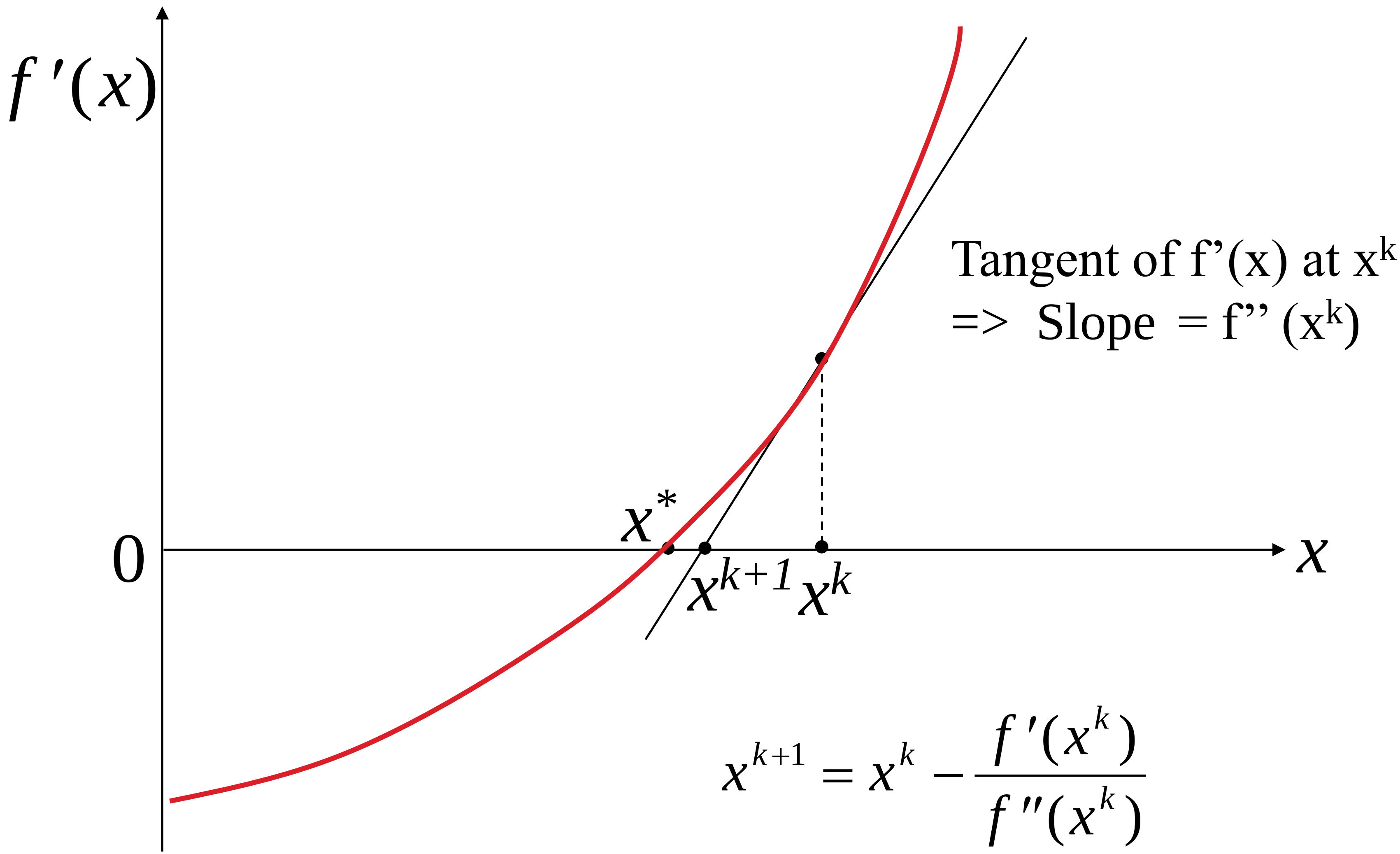
Then the stationary point, $df(x)/dx = 0$, is given as:

$$f'\left(x^k\right) + f''\left(x^k\right)\left(x^{k+1} - x^k\right) = 0$$

yielding the next approximation x^{k+1} as:

$$x^{k+1} = x^k - \frac{f'(x^k)}{f''(x^k)}$$

Newton-Rapson Method



$$x^{k+1} = x^k - \frac{f'(x^k)}{f''(x^k)}$$

`fminbnd` Function

- MATLAB has a built-in function, `fminbnd`, which combines the golden-section search and the parabolic interpolation.

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

- Options may be passed through a fourth argument using `optimset`, similar to `fzero`.

fminbnd Function

```
>> 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
```

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

`fminsearch` Function

- MATLAB has a built-in function, `fminsearch`, that can be used to determine the minimum of a multidimensional function.

  `[`*`xmin`*`, `*`fval`*`] = fminsearch(`*`function`*`, `*`x0`*`)`

  - *`xmin`* in this case will be a row vector containing the location of the minimum, while *`x0`* is an initial guess. Note that *`x0`* must contain as many entries as the function expects of it.
- The `function` must be written in terms of a single variable, where different dimensions are represented by different indices of that variable.

`fminsearch` Function

- To minimize
  $f(x,y)=2+x-y+2x^2+2xy+y^2$
  rewrite as
  $f(x_1, x_2)=2+x_1-x_2+2(x_1)^2+2x_1x_2+(x_2)^2$

```
>> 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
```

- Note that $x0$ has two entries, f is expecting it to contain two values.