Optimization #1

Dep. EE, KPU

2020년 2학기

Book & Homepage

- Convex Optimization
  - Stephen Boyd & Lieven Vandenberghe
    - Cambridge University Press (2009)
  - Authors' homepage
    - https://web.stanford.edu/~boyd/cvxbook/
  - Boyd's Lecture Videos:
    - https://web.stanford.edu/class/ee364a/videos.html
  - 교재의 pdf 파일
    - http://www.stanford.edu/~boyd/cvxbook/ 에서 다운로드 가능

Basic Concept
&
Definition

Optimization Problems

Objective function

$$\min_{\mathbf{x}\in R^n} f(\mathbf{x})$$

subject to $g_i(\mathbf{x}) \le 0, \quad i = 1, \dots, r$ inequality constraints

$h_k(\mathbf{x}) = 0, \quad k = 1, \dots, m$ equality constraints

n x 1 vector

- Feasible Solution
  - Constraints를 만족시키는 solution
- Feasible Set
  - Feasible solution 들의 모든 집합
- Optimal Solution
  - $\mathbf{x}^*$는 모든 constraints를 만족시킴
  - constraints를 만족시키는 모든 $\bar{\mathbf{x}}$에 대해서 $f(\mathbf{x}^*) \le f(\bar{\mathbf{x}})$

Optimization Problems

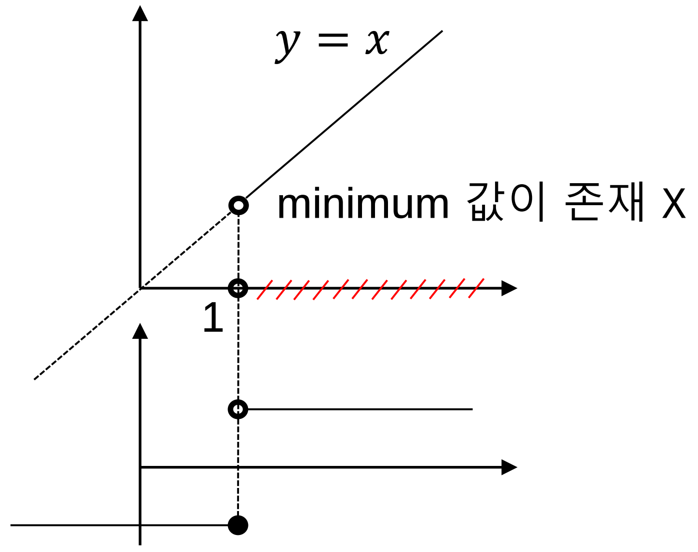
- 최적화 문제는 항상 optimal solution을 가지는가?

$$\min x$$

s. t. $g(x) \geq 0$

where

$$g(x) = \begin{cases} 1, & for\ \ x > 1 \\ -1, & for\ x \leq 1 \end{cases}$$



This problem does not have an optimal solution

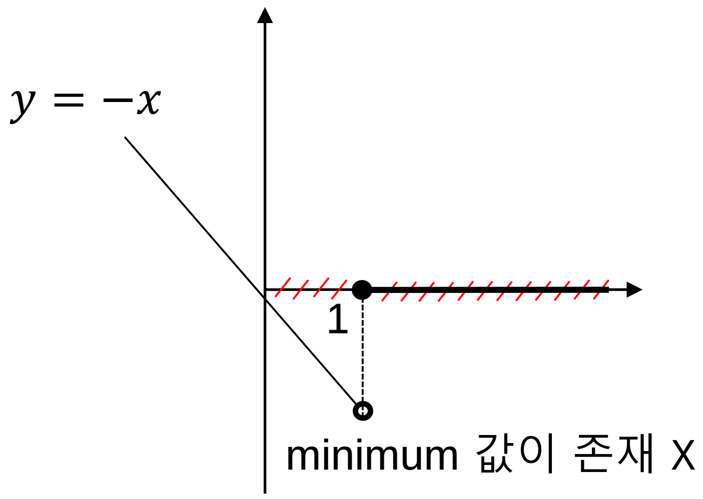
Optimization Problems

$$\min f(x)$$

s. t. $x \geq 0$

where

$$f(x) = \begin{cases} -x, & for\ \ x < 1 \\ 0, & \text{otherwise} \end{cases}$$



This problem does not have an optimal solution

Infeasible Problem

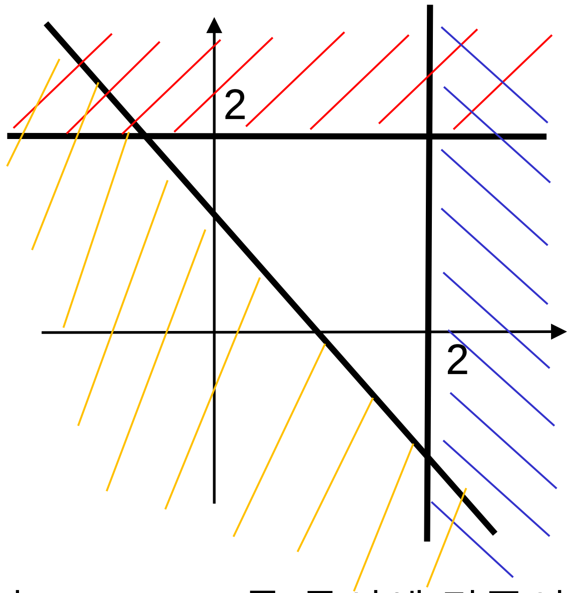
$$\min x_1 - x_2$$

s. t.

$$x_1 + x_2 \leq 1$$

$$x_1 \geq 2$$

$$x_2 \geq 2$$



3개의 constraints를 동시에 만족시키는 해가 없음

→ Infeasible problem

Unbounded Problem

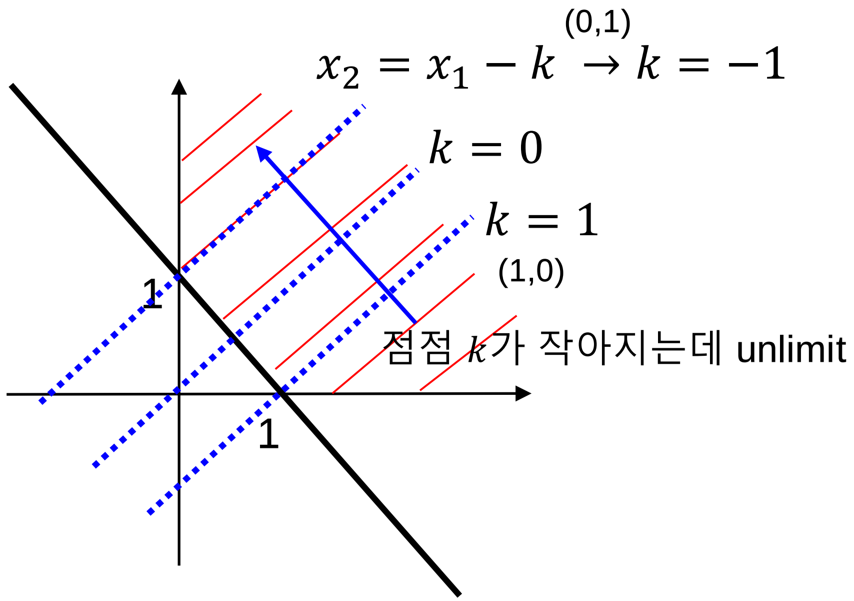
$$\min x_1 - x_2$$

s. t.

$$x_1 + x_2 \geq 1$$

$$x_1, x_2 \geq 0$$

$x_1 - x_2 = k$


(0,1)
$x_2 = x_1 - k \rightarrow k = -1$
$k = 0$
$k = 1$
(1,0)
점점 k가 작아지는데 unlimit
1
1

Objective function decreases w/o any bound if the solution moves along vector (1,0) from point (0,1)

Feasible problem이나 unbounded로 solution이 없음

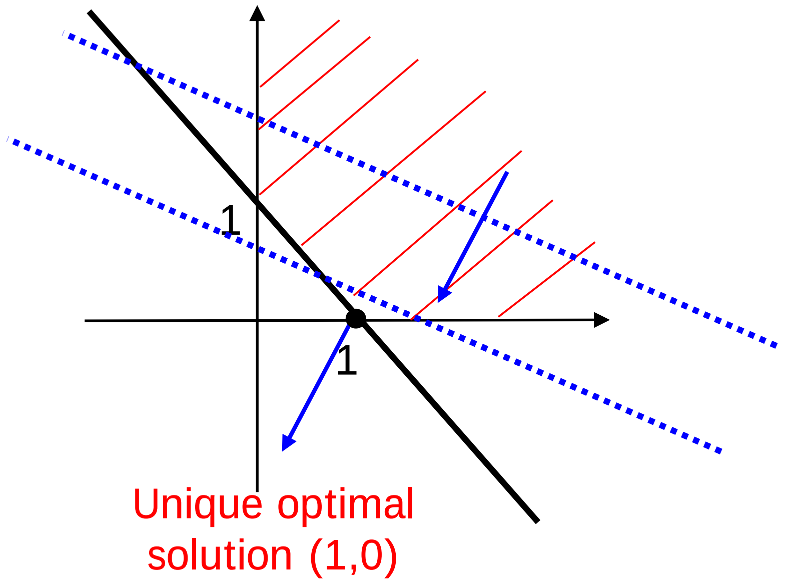
Unique optimal solution

$$\min x_1 + 2x_2$$

s. t.

$$x_1 + x_2 \geq 1$$

$$x_1, x_2 \geq 0$$



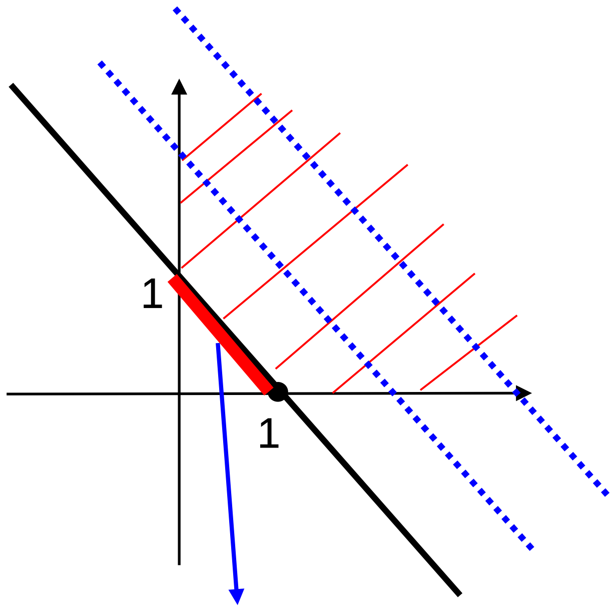
Multiple optimal solutions

$$\min x_1 + x_2$$

s. t.

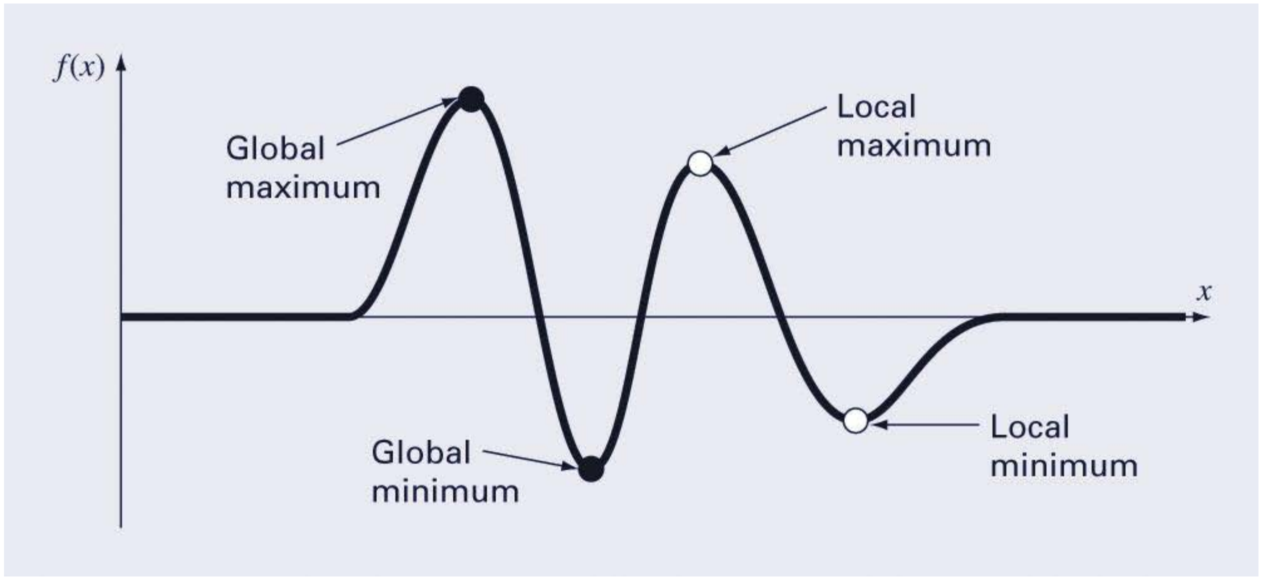
$$x_1 + x_2 \geq 1$$

$$x_1, x_2 \geq 0$$



Multiple optimal solutions

Global vs Local Optimum



Unconstrained Optimization Problem
- 1D Optimization Problem

Unconstrained Optimization Problem

$$\min_{x \in R} f(x)$$

- (0th order info.) only objective value f is available
- (1st order info.) f' is available, but not f''
- (2nd order info.) both f' and f'' are available

- Higher-order information tends to give more powerful algorithms.

- These methods are also used in multi-dimensional optimization as line search for determine how far to move along a given direction

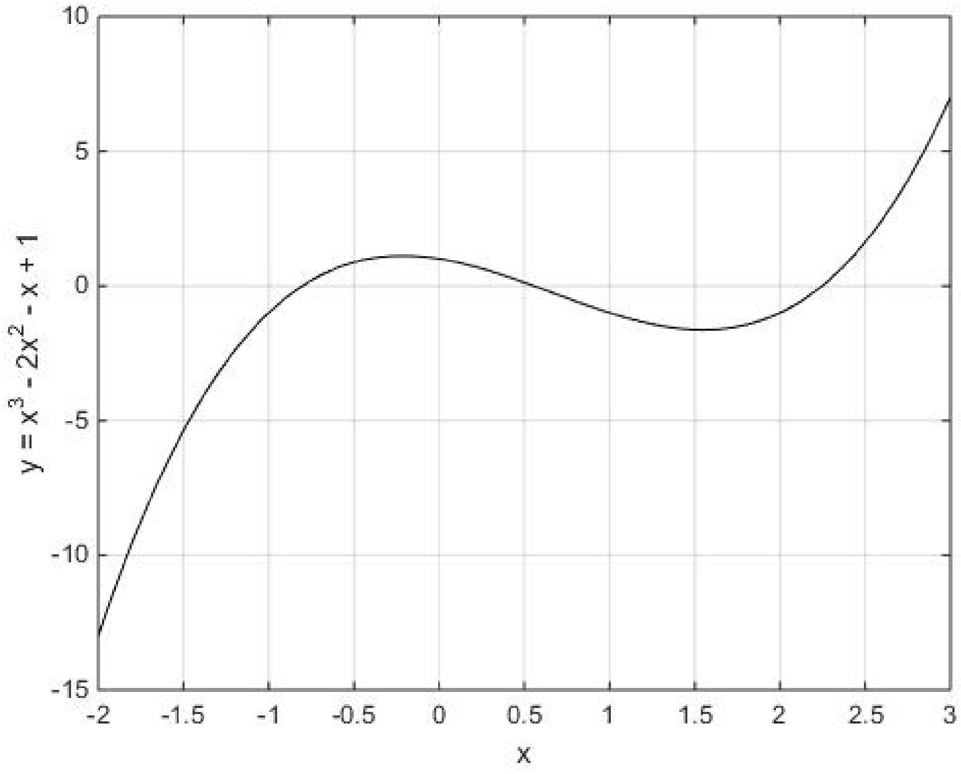
Inequality constraints just limit the searching range! (Omit)

Graphical Solution

- Procedures
  - 1. Plot each constraint as an equation and then decide which side of the line is feasible
  - 2. Find the feasible region
  - 3. Find the coordinations(좌표) of the corner points of the feasible region
  - 4. Substitute the corner point coordinates in the objective function
  - 5. Choose the optimal solution

Graphical Solution

$$\min_{x \in R} x^3 - 2x^2 - x + 1$$



고차식으로 가면? 그림을 그리기 용이하지 않음!!

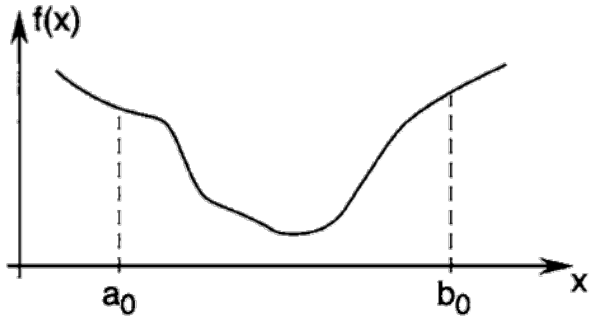
Iterative Algorithm

- Most optimization problems cannot be solved in a closed form (a single step).
- For them, we develop iterative algorithms:
    - Start from an initial candidate solution : $x^{(0)}$
    - Generate a sequence of candidate solutions(iterates): $x^{(1)}$, $x^{(2)}$, ….
    - Stop when a certain condition is met; return the candidate solution
- In a large number of algorithms, $x^{(k+1)}$ is generate from $x^{(k)}$, that is, using the information of f at $x^{(k)}$.
- In some algorithms, $x^{(k+1)}$ is generate from $x^{(k)}$, $x^{(k-1)}$, … But, for time and memory consideration, most history iterates are not kept in memory.

Iterative Algorithm

Golden Section Search

Given a closed interval $[a_0, b_0]$, a unimodal function (that is, having one and only one local minimizer in $[a_0, b_0]$), and only objective value information, find a point that is no more than ϵ away from that local minimizer.



- Mid-point : evaluate the mid-point, that is compute $f\left(\frac{a_0+b_0}{2}\right)$.
- But, it cannot determine which half contains the local minimizer. Does not work!!!

- Then? Two-points Approach!

Golden Section Search

- (Step 0) Let $a_0 = a$ and $b_0 = b$, Let $k = 0$
  - b_k는 upperlimit, a_k는 lowerlimit
- (Step 1) If $b_k - a_k < \epsilon$, Stop.

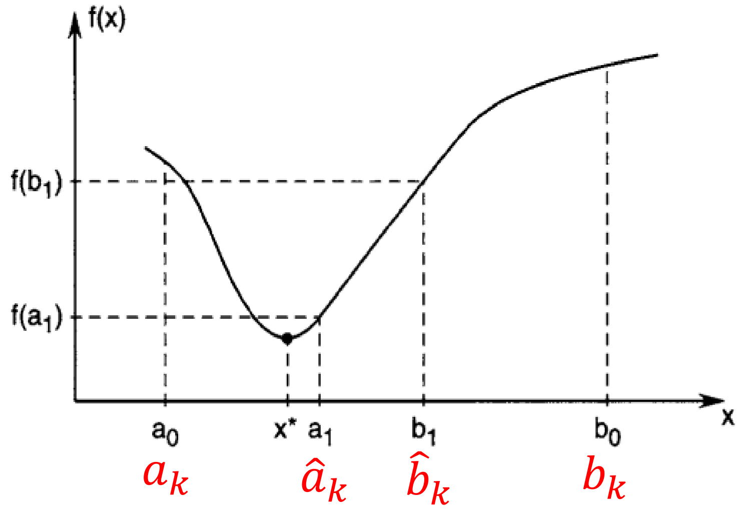
  Otherwise, go to Step 2.
- (Step 2) Let $\hat{a}_k = a_k + \tau(b_k - a_k)$

  $$\hat{b}_k = b_k - \tau(b_k - a_k)$$

  where $\tau = \frac{3-\sqrt{5}}{2} \approx 0.382$ (Golden Ratio)

  1. When $f(\hat{a}_k) < f(\hat{b}_k)$,
     (a) If $f(a_k) > f(\hat{a}_k)$, then let $a_{k+1} = a_k$, $b_{k+1} = \hat{b}_k$
     (b) If $f(a_k) \leq f(\hat{a}_k)$, then let $a_{k+1} = a_k$, $b_{k+1} = \hat{a}_k$
  2. When $f(\hat{a}_k) > f(\hat{b}_k)$,
     (a) If $f(\hat{b}_k) < f(b_k)$, then let $a_{k+1} = \hat{a}_k$, $b_{k+1} = b_k$
     (b) If $f(\hat{b}_k) \geq f(b_k)$, then let $a_{k+1} = \hat{b}_k$, $b_{k+1} = b_k$
  3. When $f(\hat{a}_k) = f(\hat{b}_k)$, let $a_{k+1} = \hat{a}_k$, $b_{k+1} = \hat{b}_k$

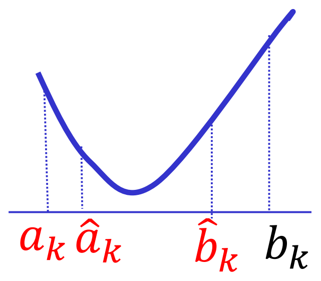
뒷장에 상세히

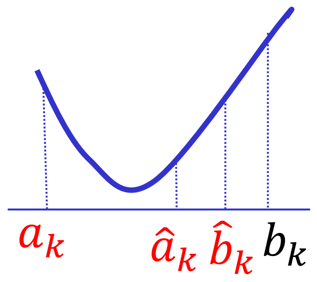


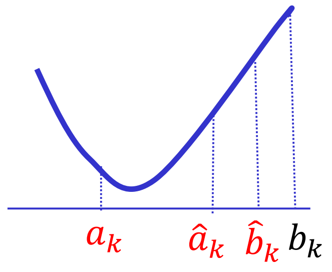
- Let $k = k + 1$. Go to Step 1.

Golden Section Search

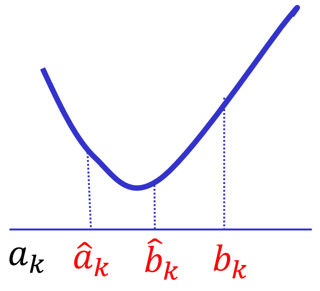
1. When $f(\hat{a}_k) < f(\hat{b}_k)$,
   (a) If $f(a_k) > f(\hat{a}_k)$, then let $a_{k+1} = a_k$, $b_{k+1} = \hat{b}_k$
   (b) If $f(a_k) \leq f(\hat{a}_k)$, then let $a_{k+1} = a_k$, $b_{k+1} = \hat{a}_k$

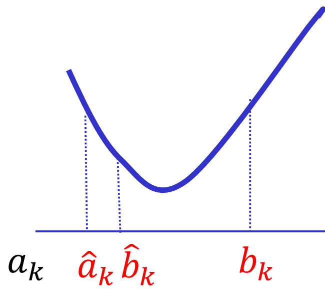


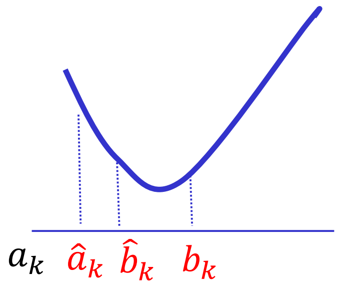




2. When $f(\hat{a}_k) > f(\hat{b}_k)$,
   (a) If $f(\hat{b}_k) < f(b_k)$, then let $a_{k+1} = \hat{a}_k$, $b_{k+1} = b_k$
   (b) If $f(\hat{b}_k) \geq f(b_k)$, then let $a_{k+1} = \hat{b}_k$, $b_{k+1} = b_k$







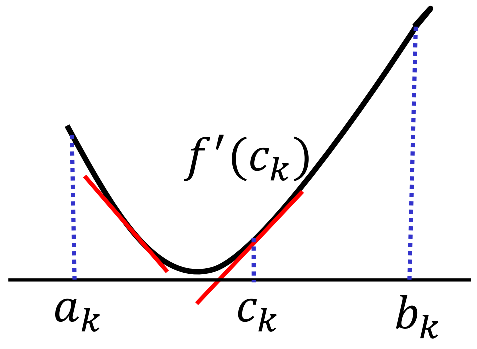
3. When $f(\hat{a}_k) = f(\hat{b}_k)$, let $a_{k+1} = \hat{a}_k$, $b_{k+1} = \hat{b}_k$

Bisection Search

Bisection Search

- Given a closed interval $[a_0, b_0]$, a continuously differentiable, a unimodal function, and derivative information, find a point that is no more than ϵ away from that local minimizer.
- Mid-point works with derivative!

- (Step 0) Let $a_0 = a$ and $b_0 = b$, Let $k = 0$
  - b_k는 upperlimit, a_k는 lowerlimit
- (Step 1) If $b_k - a_k < \epsilon$, Stop.
  Otherwise, go to Step 2.
- (Step 2) Let $c_k = \frac{1}{2}(a_k + b_k)$
  1. If $f'(c_k) = 0$, then c_k is the local minimizer.
  2. If $f'(c_k) > 0$, then $a_{k+1} = a_k$, $b_{k+1} = c_k$
  3. If $f'(c_k) < 0$, then $a_{k+1} = c_k$, $b_{k+1} = b_k$
- Let $k = k + 1$. Go to Step 1.



Newton's Method

Newton's Method

- Given a twice continuously differentiable function and objective, derivative, and 2nd derivative information, find an approximate minimizer.
- Newton's method does not need intervals but must start sufficiently close to x^* !!!
- Iteration: minimize the quadratic approximation

$$x_{k+1} \leftarrow \arg\min q(x) \coloneqq f(x_k) + f'(x_k)(x - x_k) + \frac{1}{2} f''(x_k)(x - x_k)^2$$

k번째 iteration point인 x_k에서 2nd order approximation 최소화하는 x를 찾음
(해당 함수 미분해서 0되는 지점 찾음)

Taylor series expansion

$$f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n = f(a) + f'(a)(x-a) + \frac{1}{2} f''(a)\,(x-a)^2 + \cdots$$

2nd order Taylor series expansion at x_k

$$f(x) \approx f(x_k) + f'(x_k)(x - x_k) + \frac{1}{2} f''(x_k)(x - x_k)^2$$

Newton's Method

$$f(x) \approx f(x_k) + f'(x_k)(x - x_k) + \frac{1}{2} f''(x_k)(x - x_k)^2$$

$$= f(x_k) + f'(x_k)\Delta x + \frac{1}{2} f''(x_k)(\Delta x)^2$$

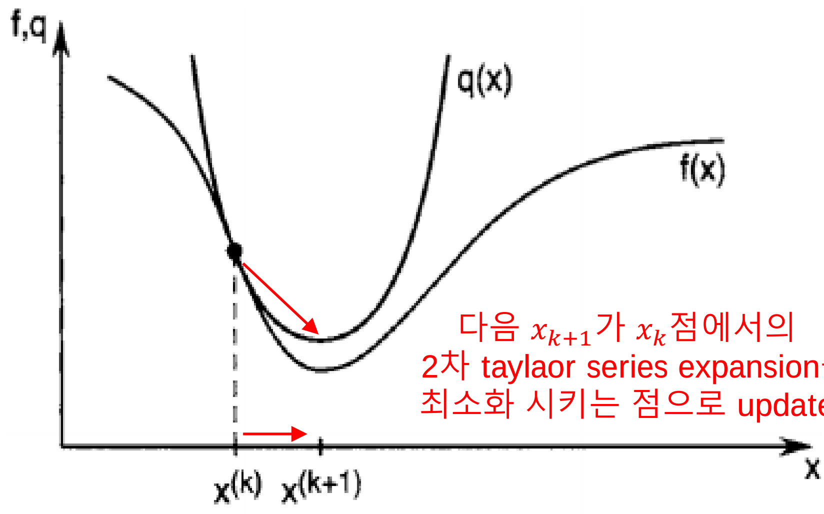
- Derivative with respect to Δx

$$0 = \frac{d}{d\Delta x}\left(f(x_k) + f'(x_k)\Delta x + \frac{1}{2} f''(x_k)(\Delta x)^2 \right) = f'(x_k) + f''(x_k)\Delta x$$

$$\Delta x = -\frac{f'(x_k)}{f''(x_k)}$$

$$x_{k+1} \leftarrow x_k + \Delta x = x_k - \frac{f'(x_k)}{f''(x_k)}$$

Second derivative → 계산량 많은 문제



다음 x_{k+1}가 x_k점에서의
2차 taylaor series expansion을
최소화 시키는 점으로 update

Quadratic approximation with f''(x)>0

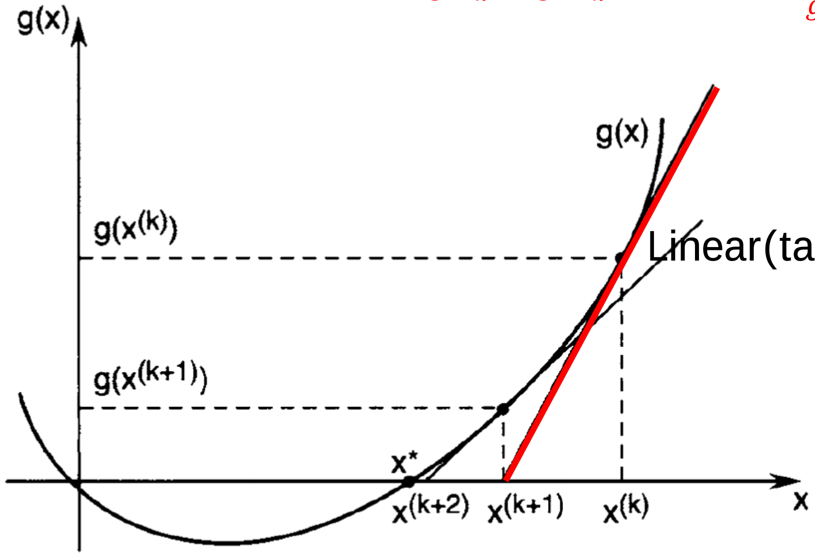
$$\arg\min q(x) \coloneqq f(x_k) + f'(x_k)(x - x_k) + \frac{1}{2} f''(x_k)(x - x_k)^2$$

Newton's Method

- Let us define $g(x) = f'(x) \rightarrow g'(x) = f''(x)$
- Newton's method is a way to solve $g(x) = f'(x) = 0$

$$x_{k+1} \leftarrow x_k - \frac{f'(x_k)}{f''(x_k)} = x_k - \frac{g(x_k)}{g'(x_k)}$$

$$0 = g(x_k) + g'(x_k)\Delta x \ \rightarrow \ \Delta x = -\frac{g(x_k)}{g'(x_k)}$$

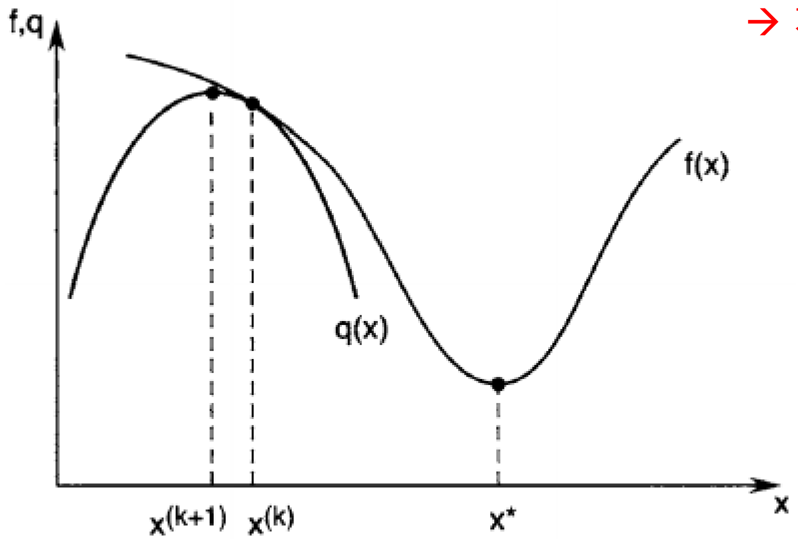


Linear(tangent) approximation to g(x)=f'(x)

Newton's Method

- What if $f''(x_k) < 0$?

q(x) (2차 approximation)가 위로 볼록 함수 → 미분해서 0 되는 지점을 찾는데
이 지점은 최소값 찾는 방향에서
더욱 멀어짐
→ 처음에 초기값을 솔루션 근처로
잘 잡아야함



즉, Newton's iteration 은 발산함

Secant Method

- Recall Newton's method uses $f''(x_k)$ for minimizing $f(x)$:

$$x_{k+1} := x_k - \frac{f'(x_k)}{f''(x_k)}$$

- If f'' is not available or is expensive to compute, we can approximate

$$f''(x_k) \approx \frac{f'(x_k) - f'(x_{k-1})}{x_k - x_{k-1}}$$

- This gives the iteration of the secant method:

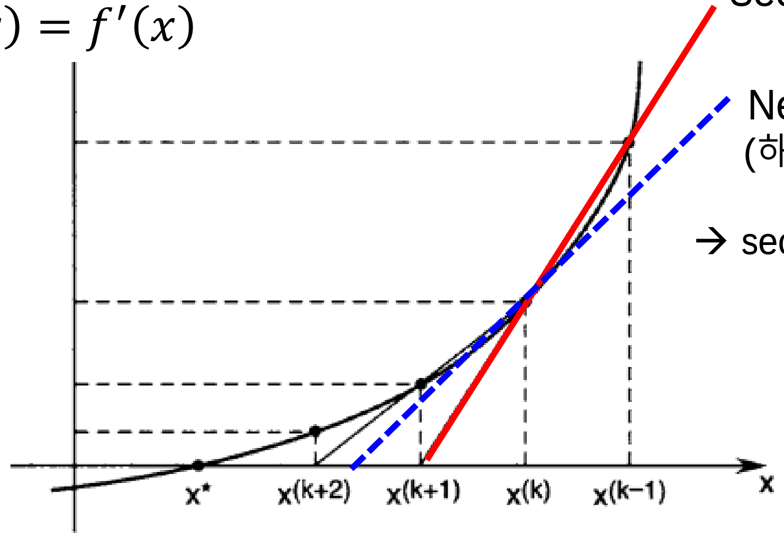
$$x_{k+1} := x_k - \frac{(x_k - x_{k-1})f'(x_k)}{f'(x_k) - f'(x_{k-1})} = \frac{x_{k-1}f'(x_k) - x_k f'(x_{k-1})}{f'(x_k) - f'(x_{k-1})}$$

Note: the method needs two initial points. (미분 해야 하므로)

Secant Method

$$f''(x_k) = \frac{f'(x_k) - f'(x_{k-1})}{x_k - x_{k-1}}$$

$g(x) = f'(x)$



Secant method

Newton's method
(해당 지점의 미분값
구할 수 있음
→ second derivative 가능)

The secant method is slightly slower than
Newton's method but is cheaper!

Comparisons

- Comparisons of different 1D search methods
  - Golden section search (and Fibonacci search):
    - One objective evaluation at each iteration
    - Narrows search interval by less than half each time
  - Bisection search:
    - One derivative evaluation at each iteration
    - Narrows search interval to exactly half each time
  - Secant method:
    - Two points to start with; then one derivative evaluation at each iteration
    - Must start near x^*, has superlinear convergence
  - Newton’s method:
    - One derivative and one second derivative evaluations at each iteration
    - Must start near x^*, has quadratic convergence, fastest among the four

Comparisons

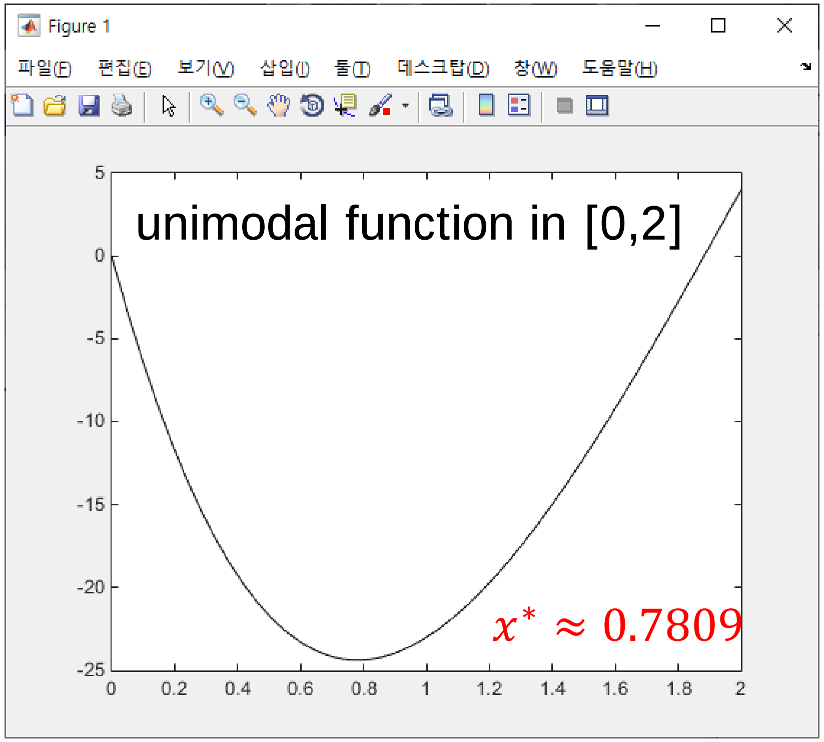
- 이외에도 많은 알고리즘
  - Gradient descent 알고리즘 등 (다음 주 강의)

실습

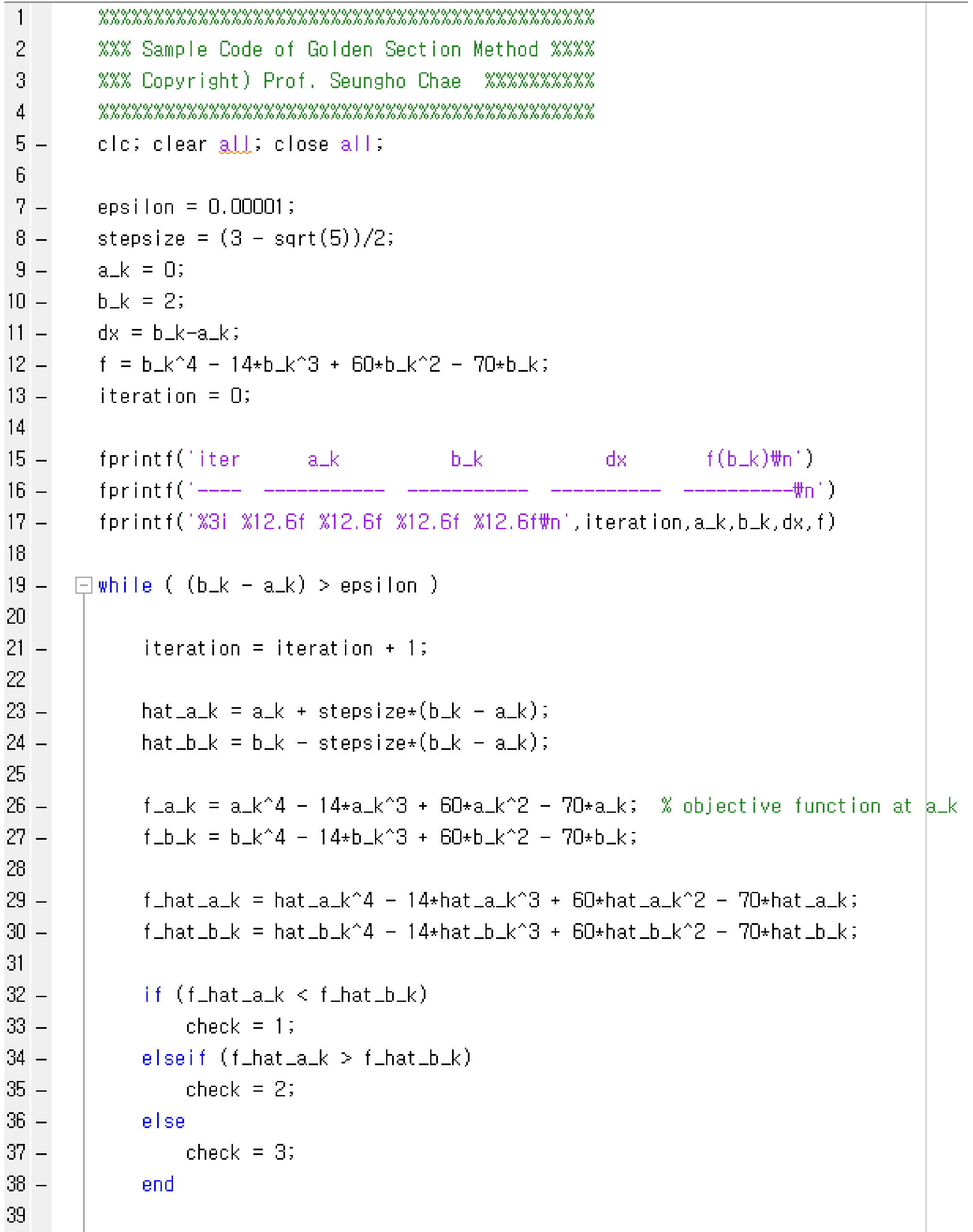
- MATLAB으로 Golden Section Search 알고리즘을 구현하여 해를 구하시오.

$$\min f(x) = x^4 - 14x^3 + 60x^2 - 70x$$

- Initial Interval: [0,2]
- Required Accuracy $\varepsilon = 0.00001$
- 각 iteration마다 a_k값을 확인하시오.

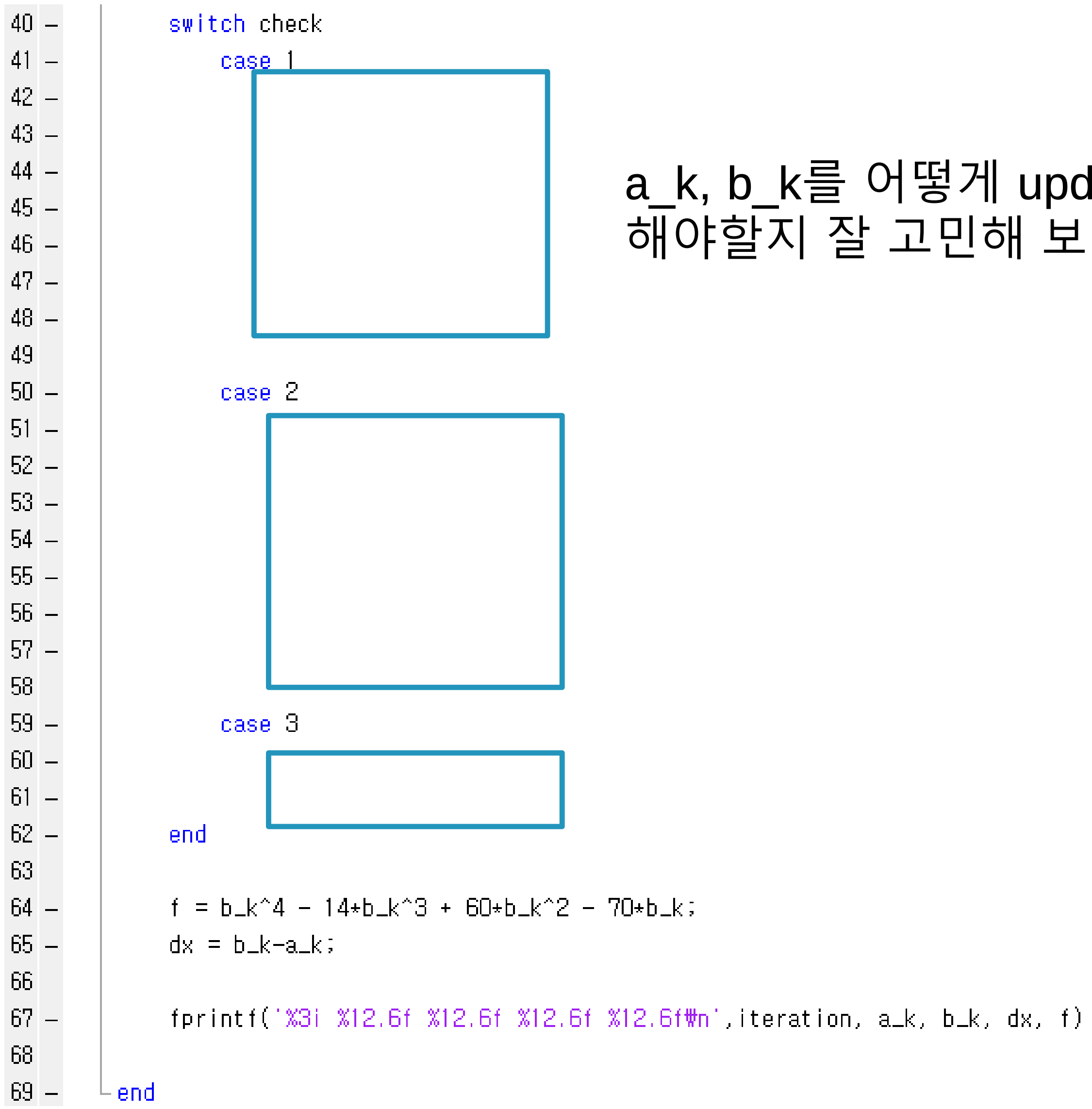


실습



```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Sample Code of Golden Section Method %%%%
%%% Copyright) Prof. Seungho Chae  %%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clc; clear all; close all;

epsilon = 0.00001;
stepsize = (3 - sqrt(5))/2;
a_k = 0;
b_k = 2;
dx = b_k-a_k;
f = b_k^4 - 14*b_k^3 + 60*b_k^2 - 70*b_k;
iteration = 0;

fprintf('iter      a_k          b_k           dx        f(b_k)₩n')
fprintf('----  -----------  -----------  ----------  ----------₩n')
fprintf('%3i %12.6f %12.6f %12.6f %12.6f₩n',iteration,a_k,b_k,dx,f)

while ( (b_k - a_k) > epsilon )

    iteration = iteration + 1;

    hat_a_k = a_k + stepsize*(b_k - a_k);
    hat_b_k = b_k - stepsize*(b_k - a_k);

    f_a_k = a_k^4 - 14*a_k^3 + 60*a_k^2 - 70*a_k;  % objective function at a_k
    f_b_k = b_k^4 - 14*b_k^3 + 60*b_k^2 - 70*b_k;

    f_hat_a_k = hat_a_k^4 - 14*hat_a_k^3 + 60*hat_a_k^2 - 70*hat_a_k;
    f_hat_b_k = hat_b_k^4 - 14*hat_b_k^3 + 60*hat_b_k^2 - 70*hat_b_k;

    if (f_hat_a_k < f_hat_b_k)
        check = 1;
    elseif (f_hat_a_k > f_hat_b_k)
        check = 2;
    else
        check = 3;
    end

```

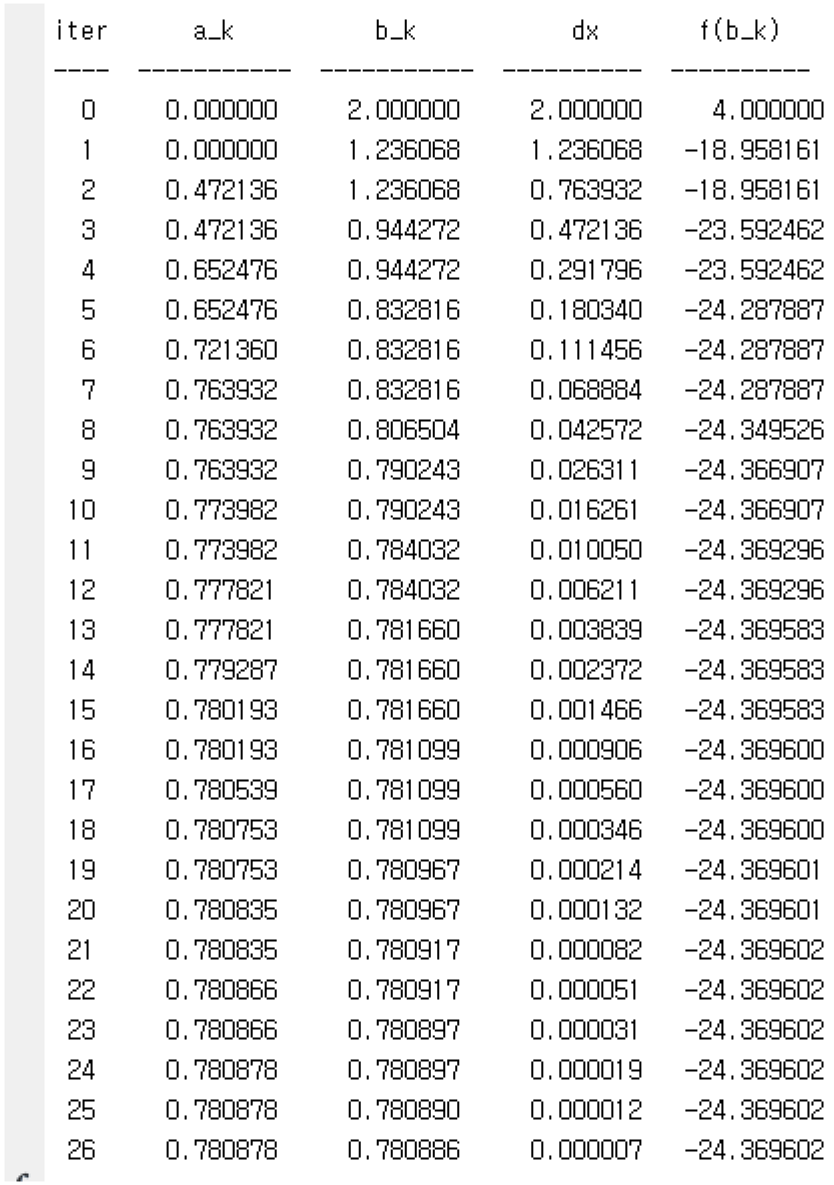
실습



```
    switch check
        case 1









        case 2









        case 3



    end

    f = b_k^4 - 14*b_k^3 + 60*b_k^2 - 70*b_k;
    dx = b_k-a_k;

    fprintf('%3i %12.6f %12.6f %12.6f %12.6f₩n',iteration, a_k, b_k, dx, f)

end
```

a_k, b_k를 어떻게 update를 해야할지 잘 고민해 보기

실습



```
iter      a_k          b_k          dx         f(b_k)
----  -----------  -----------  ----------  ----------
   0    0.000000     2.000000    2.000000     4.000000
   1    0.000000     1.236068    1.236068   -18.958161
   2    0.472136     1.236068    0.763932   -18.958161
   3    0.472136     0.944272    0.472136   -23.592462
   4    0.652476     0.944272    0.291796   -23.592462
   5    0.652476     0.832816    0.180340   -24.287887
   6    0.721360     0.832816    0.111456   -24.287887
   7    0.763932     0.832816    0.068884   -24.287887
   8    0.763932     0.806504    0.042572   -24.349526
   9    0.763932     0.790243    0.026311   -24.366907
  10    0.773982     0.790243    0.016261   -24.366907
  11    0.773982     0.784032    0.010050   -24.369296
  12    0.777821     0.784032    0.006211   -24.369296
  13    0.777821     0.781660    0.003839   -24.369583
  14    0.779287     0.781660    0.002372   -24.369583
  15    0.780193     0.781660    0.001466   -24.369583
  16    0.780193     0.781099    0.000906   -24.369600
  17    0.780539     0.781099    0.000560   -24.369600
  18    0.780753     0.781099    0.000346   -24.369600
  19    0.780753     0.780967    0.000214   -24.369601
  20    0.780835     0.780967    0.000132   -24.369601
  21    0.780835     0.780917    0.000082   -24.369602
  22    0.780866     0.780917    0.000051   -24.369602
  23    0.780866     0.780897    0.000031   -24.369602
  24    0.780878     0.780897    0.000019   -24.369602
  25    0.780878     0.780890    0.000012   -24.369602
  26    0.780878     0.780886    0.000007   -24.369602
```

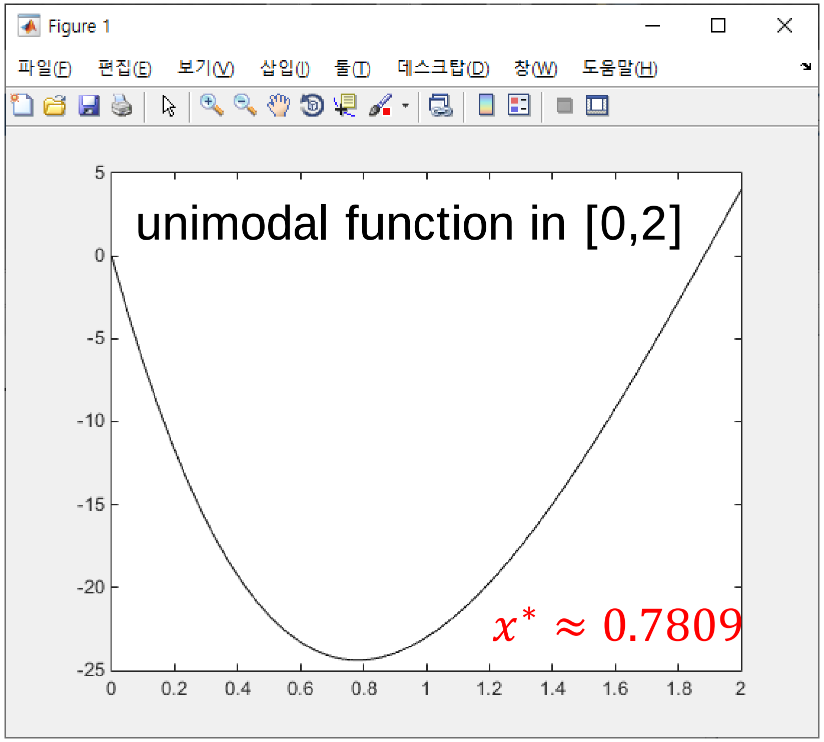
$x^* \approx 0.7809$

Homework

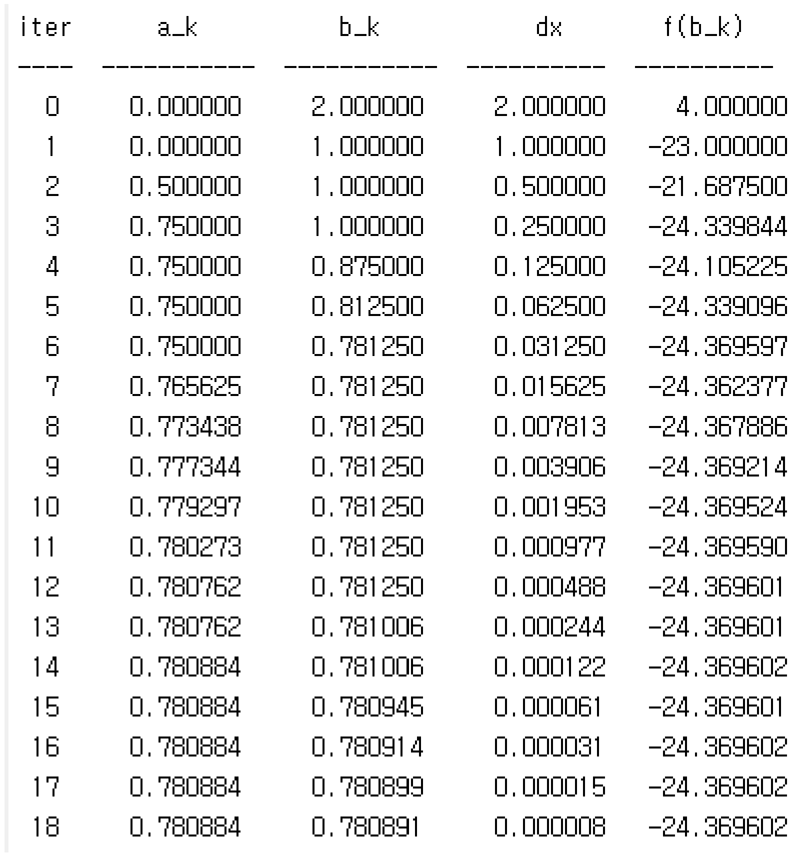
- MATLAB으로 bisection Search 알고리즘을 구현하여 해를 구하시오.

$$\min f(x) = x^4 - 14x^3 + 60x^2 - 70x$$

- Initial Interval: [0,2]
- Required Accuracy $\varepsilon = 0.00001$
- 각 iteration마다 c_k값을 확인하시오.



Homework



```
iter       a_k          b_k          dx          f(b_k)
----   -----------  -----------  ----------  ----------
  0     0.000000     2.000000     2.000000     4.000000
  1     0.000000     1.000000     1.000000   -23.000000
  2     0.500000     1.000000     0.500000   -21.687500
  3     0.750000     1.000000     0.250000   -24.339844
  4     0.750000     0.875000     0.125000   -24.105225
  5     0.750000     0.812500     0.062500   -24.339096
  6     0.750000     0.781250     0.031250   -24.369597
  7     0.765625     0.781250     0.015625   -24.362377
  8     0.773438     0.781250     0.007813   -24.367886
  9     0.777344     0.781250     0.003906   -24.369214
 10     0.779297     0.781250     0.001953   -24.369524
 11     0.780273     0.781250     0.000977   -24.369590
 12     0.780762     0.781250     0.000488   -24.369601
 13     0.780762     0.781006     0.000244   -24.369601
 14     0.780884     0.781006     0.000122   -24.369602
 15     0.780884     0.780945     0.000061   -24.369601
 16     0.780884     0.780914     0.000031   -24.369602
 17     0.780884     0.780899     0.000015   -24.369602
 18     0.780884     0.780891     0.000008   -24.369602
```

$$x^* \approx 0.7809$$

Homework

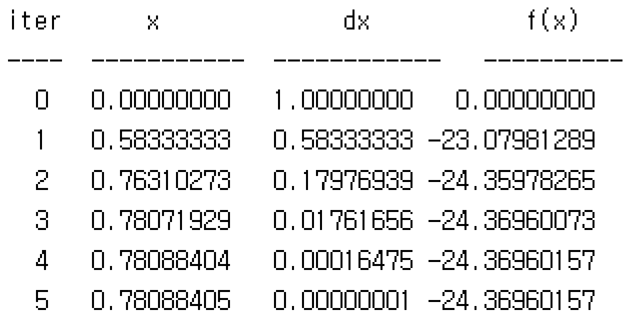
- 다음 문제를 MATLAB으로 Newtons' method를 구현하여 해를 구하시오.

$$\min f(x) = x^4 - 14x^3 + 60x^2 - 70x \qquad x^* \approx 0.7809$$

  - Initial point: $x_0 = 0$
  - $x_1 = 0.5833$
  - $x_2 = 0.7631$
  - $x_3 = 0.7807$
  - $x_4 = 0.7809$

$\rightarrow$ Produce highly accurate solutions in
$\rightarrow$ Just a few steps.

Homework



```
iter      x             dx             f(x)
----  -----------  ------------   ----------
  0   0.00000000   1.00000000    0.00000000
  1   0.58333333   0.58333333  -23.07981289
  2   0.76310273   0.17976939  -24.35978265
  3   0.78071929   0.01761656  -24.36960073
  4   0.78088404   0.00016475  -24.36960157
  5   0.78088405   0.00000001  -24.36960157
```

$x^* \approx 0.7809$

Only 5 steps!