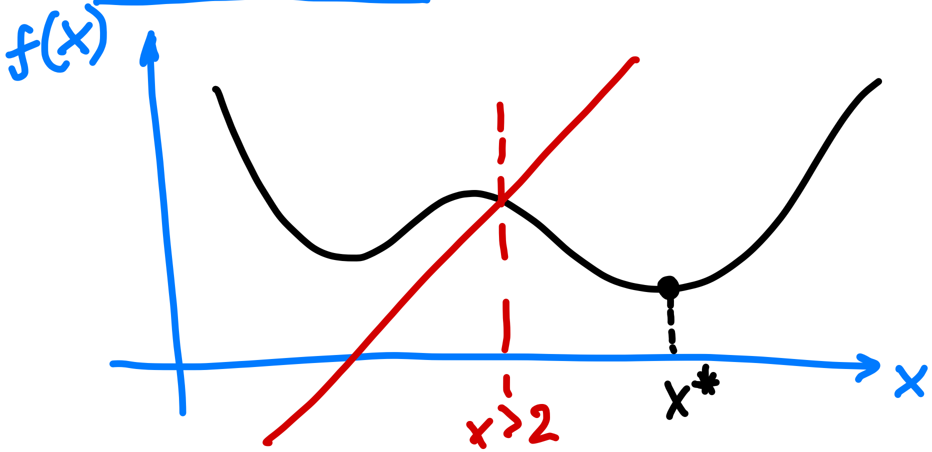
Optimization (Introduction)

Optimization

1D $f(x): \mathbb{R} \to \mathbb{R}$

ND $f(\underset{\sim}{x}): \mathbb{R}^n \to \mathbb{R}$

Goal: Find the **minimizer** $\boldsymbol{x}^*$ that minimizes the **objective (cost) function** $f(\boldsymbol{x}): \mathcal{R}^n \to \mathcal{R}$



Unconstrained Optimization

$$\boxed{f(x^*) = \min_{x} f(x)}$$

or

$$x^* = \arg\min_{\boxed{x}} f(x)$$

Optimization

Goal: Find the **minimizer** $\boldsymbol{x}^*$ that minimizes the **objective (cost) function** $f(\boldsymbol{x}): \mathcal{R}^n \to \mathcal{R}$

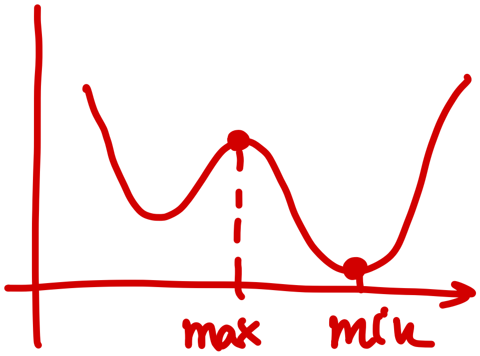
Constrained Optimization

$$\begin{cases} f(x^*) = \min\limits_{x} f(x) \\ \text{s.t.} \quad h_i(x) = 0 \rightarrow \text{equality} \\ \qquad g_j(x) \leq 0 \rightarrow \text{inequality} \end{cases}$$

$$i = 1, n$$
$$j = 1, m$$

Unconstrained Optimization

- **What if we are looking for a maximizer $\boldsymbol{x}^*$?**



$$f(\boldsymbol{x}^*) = \max_{\boldsymbol{x}} f(\boldsymbol{x})$$

$$f(x^*) = \min_{x} (-f(x))$$

Calculus problem: maximize the rectangle area subject to perimeter constraint

area

$$\max_{\boldsymbol{d}\in\mathcal{R}^2} \qquad f(d_1,d_2) = d_1 \times d_2$$

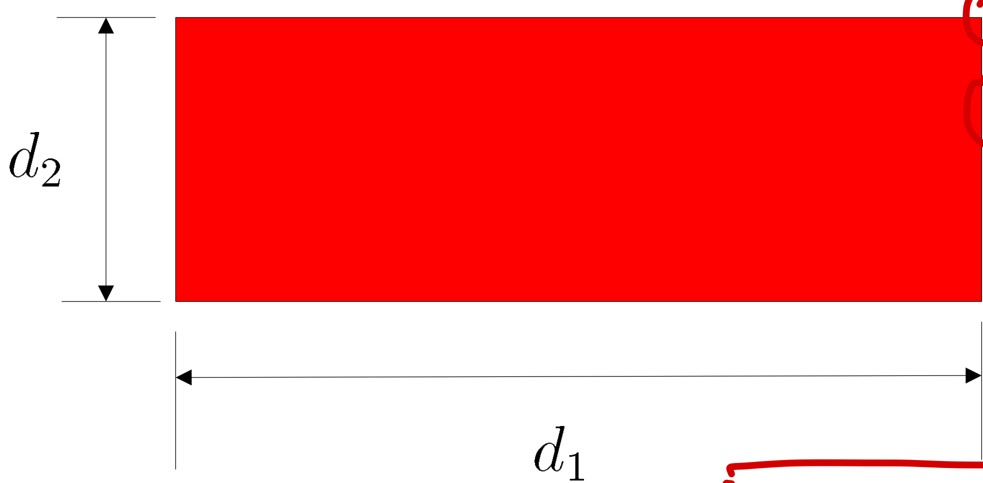
max Area

such that (1) $g(d_1,d_2) = \underbrace{2\,(d_1+d_2)}_{\text{perimeter}} - 20 \le 0$

perimeter constraint

(2) $d_1 \le 10$

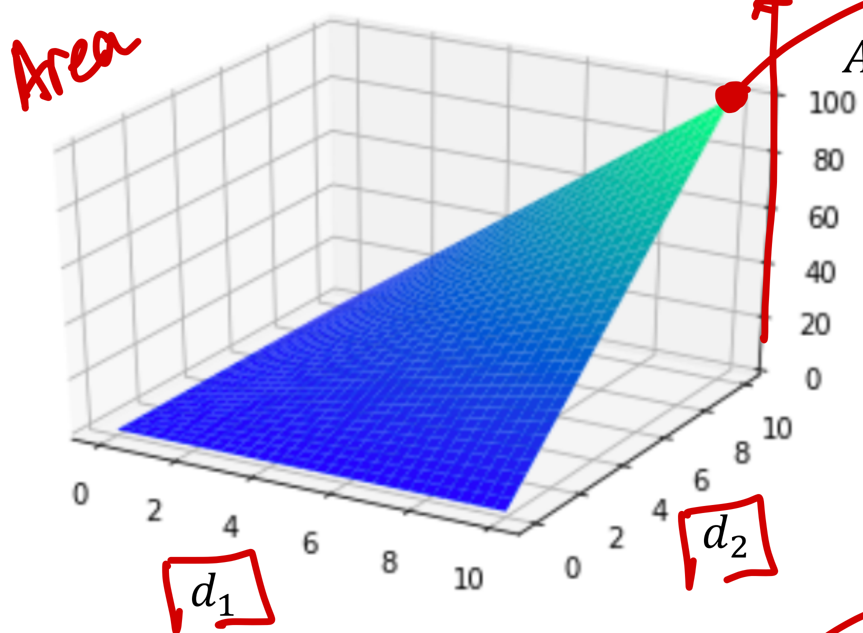
(3) $d_2 \le 10$



d_1^*, d_2^* (without peri const) $\Longrightarrow$ $\boxed{d_1 = d_2 = 10 \rightarrow A = 100}$

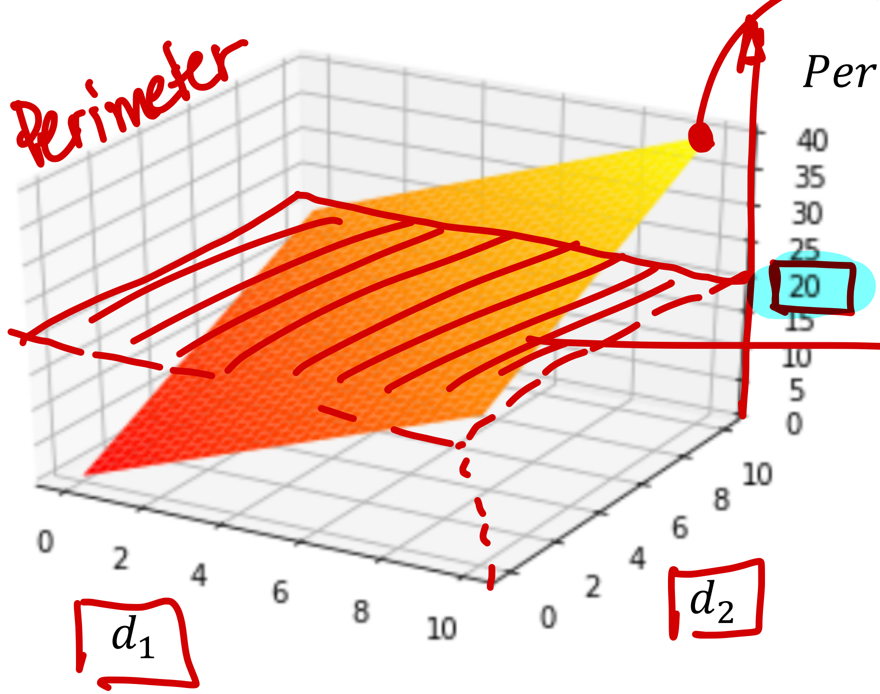
max A = 100

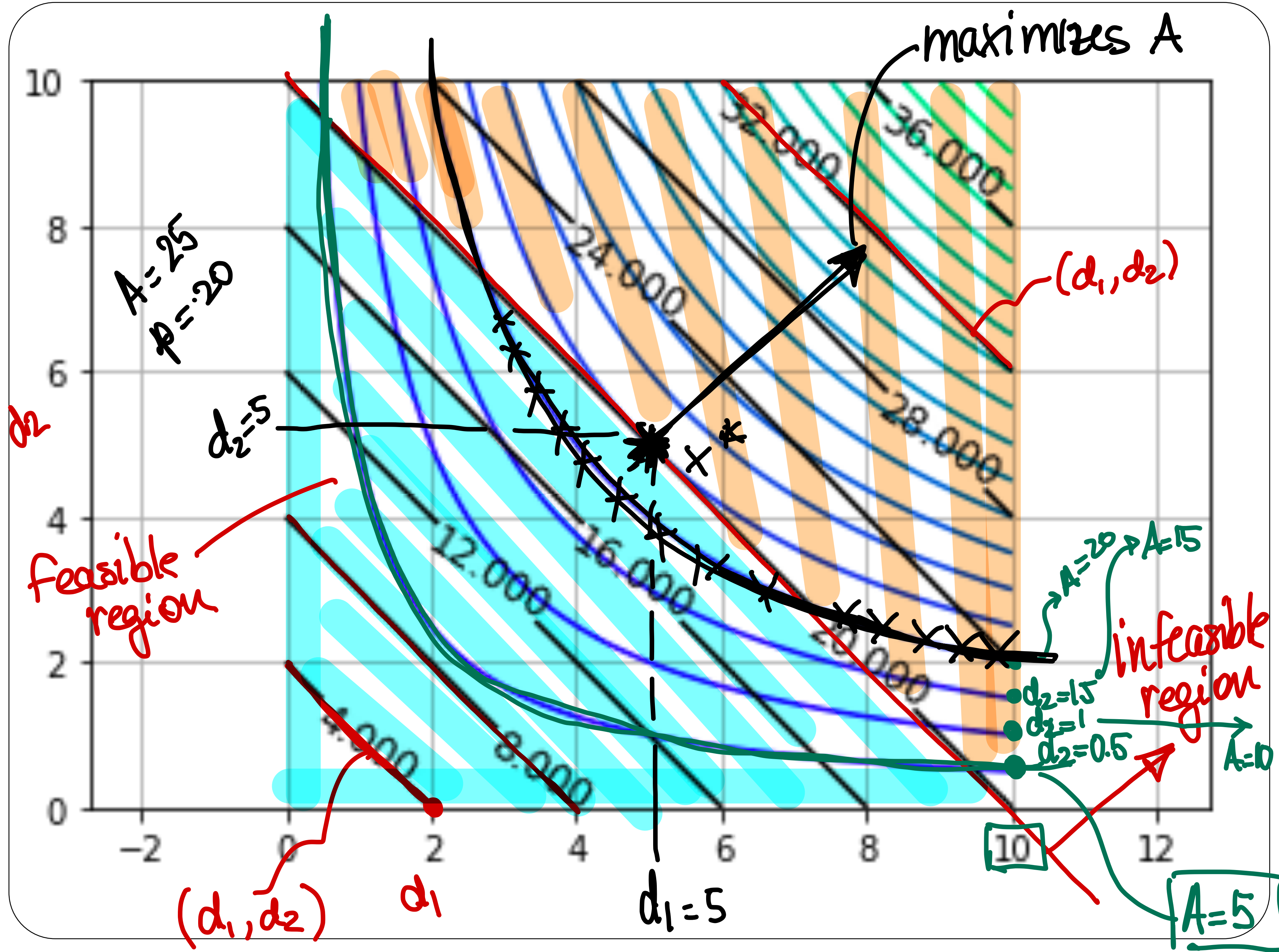
$$Area = d_1 d_2$$



P= 40 (violates!)

$$Perimeter = 2(d_1 + d_2)$$





maximizes A
A=25
p=.20
d_2
$d_2=5$
$d_1=5$
d_1
feasible region
infeasible region
(d_1,d_2)
(d_1,d_2)
A=20
A=15
A=10
A=5
$d_2=1.5$
$d_2=1$
$d_2=0.5$
4.000
8.000
12.000
16.000
20.000
24.000
28.000
32.000
36.000

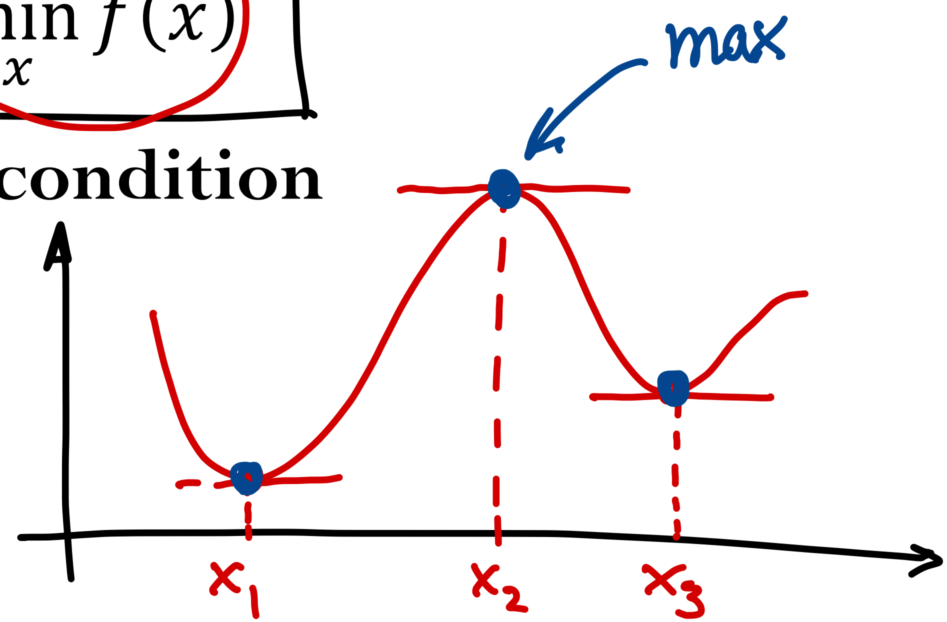
What is the optimal solution? (1D)

$$\boxed{f(x^*) = \min_x f(x)}$$

(First-order) Necessary condition

$$\boxed{f'(x^*) = 0}$$

gives stationary points



(Second-order) Sufficient condition

$f''(x^*) > 0 \longrightarrow x^*$ is minimum

$f''(x^*) < 0 \longrightarrow x^*$ is maximum

Types of optimization problems

$$f(x^*) = \min_x f(x)$$

f: nonlinear, continuous and smooth

Gradient-free methods

Evaluate $f(x)$

Gradient (first-derivative) methods

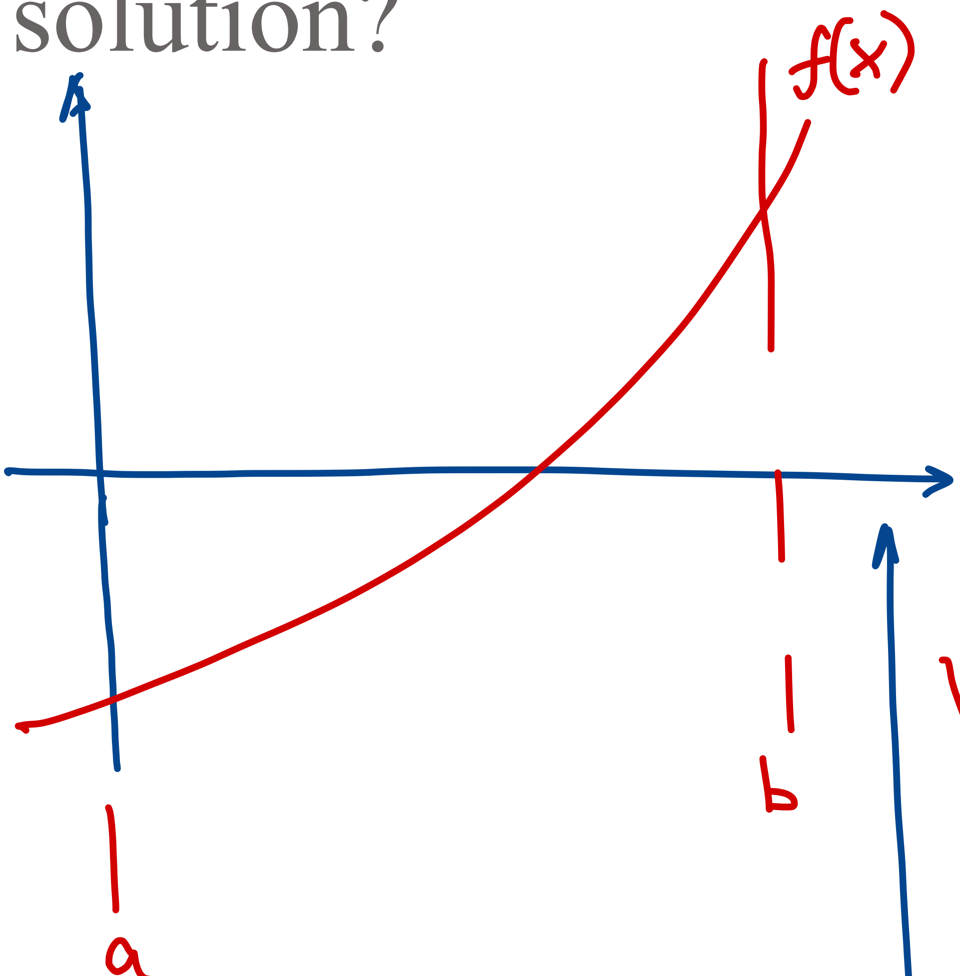
Evaluate $f(x), f'(x)$

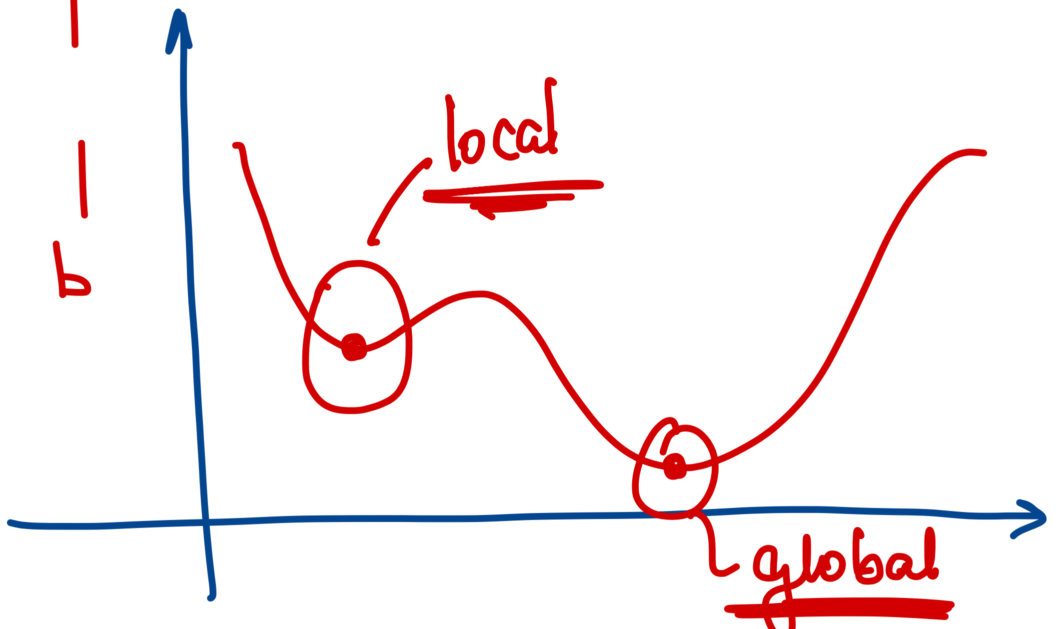
Second-derivative methods

Evaluate $f(x), f'(x), f''(x)$

Does the solution exists? Local or global solution?

no stationary



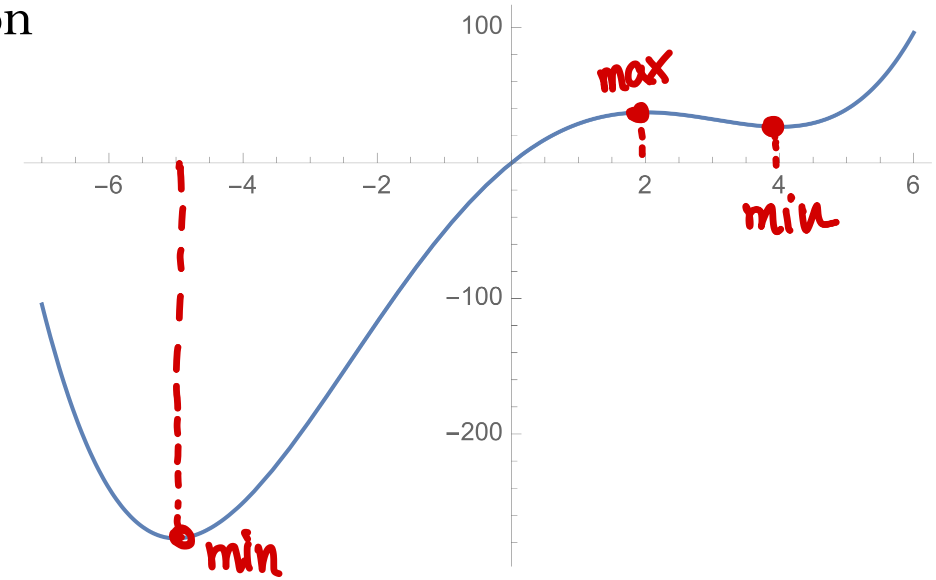


Example (1D)

min f(x)

×

Consider the function $f(x) = \frac{x^4}{4} - \frac{x^3}{3} - 11x^2 + 40x$. Find the stationary point and check the sufficient condition



* 1st order necessary condition

$f'(x) = \frac{4x^3}{4} - \frac{3x^2}{3} - 22x + 40$

$f'(x) = 0 \Rightarrow x^3 - x^2 - 22x + 40 = 0$

solutions $\Rightarrow x = \begin{cases} -5 \\ 2 \\ 4 \end{cases}$

* 2nd order condition :

$f''(x) = 3x^2 - 2x - 22$

$f''(-5) = 3(25) + 10 - 22 > 0$
(MIN)

$f''(2) = 12 - 4 - 22 < 0 \longrightarrow$ (MAX)

$f''(4) = 3(16) - 8 - 22 > 0 \rightarrow$ (MIN)

What is the optimal solution? (ND)

$$f(\boldsymbol{x}^*) = \min_{\boldsymbol{x}} f(\boldsymbol{x})$$

$f(x)$

$\underset{=}{f}(\underset{\sim}{x})$

(First-order) Necessary condition

1D: $f'(x) = 0$

$\underline{\underline{\text{ND}}}$: $\underset{\sim}{\nabla} f(\underset{\sim}{x}^*) = \underset{\sim}{0} \longrightarrow$ gives stationary solution x^*

(Second-order) Sufficient condition

1D: $f''(x) > 0$

$\underline{\underline{\text{ND}}}$: $\underset{=}{H}(\underset{\sim}{x}^*)$ is positive definite $\rightarrow$ x^* is minimizer

Taking derivatives…

$f: \mathbb{R}^n \longrightarrow \mathbb{R}$

$f(\underset{\sim}{x}) = f(x_1, x_2, \ldots, x_n)$

$\dfrac{\partial f}{\partial x_1} \quad \dfrac{\partial f}{\partial x_2} \quad \cdots \quad \dfrac{\partial f}{\partial x_n} \Longrightarrow \underline{\underline{\underset{\sim}{\nabla} f(\underset{\sim}{x})}} = \begin{bmatrix} \dfrac{\partial f}{\partial x_1} \\ \dfrac{\partial f}{\partial x_2} \\ \vdots \\ \dfrac{\partial f}{\partial x_n} \end{bmatrix} (n \times 1)$

gradient of f

$\dfrac{d}{\partial x_i} \underset{\sim}{\nabla} f$

$$\underset{\sim}{H}(\underset{\sim}{x}) = \begin{bmatrix} \dfrac{\partial^2 f}{\partial x_1^2} & \dfrac{\partial^2 f}{\partial x_1 \partial x_2} & \dfrac{\partial^2 f}{\partial x_1 \partial x_3} & \cdots & \dfrac{\partial^2 f}{\partial x_1 \partial x_n} \\ \dfrac{\partial^2 f}{\partial x_2 \partial x_1} & \dfrac{\partial^2 f}{\partial x_2^2} & \dfrac{\partial^2 f}{\partial x_2 \partial x_3} & \cdots & \dfrac{\partial^2 f}{\partial x_2 \partial x_n} \\ \dfrac{\partial^2 f}{\partial x_n \partial x_1} & \dfrac{\partial^2 f}{\partial x_n \partial x_2} & & \cdots & \dfrac{\partial^2 f}{\partial x_n^2} \end{bmatrix}_{n \times n}$$

$\Downarrow$

symm!

$(\underset{\sim}{\nabla} f)_i = \dfrac{\partial f}{\partial x_i}$

$(\underset{\sim}{H})_{ij} = \dfrac{\partial^2 f}{\partial x_i \partial x_j}$

From linear algebra:

$\boldsymbol{y}^T \underline{\underline{H}} \boldsymbol{y} \rightarrow$ scalar

vector

$\boldsymbol{y} \cdot \underline{\underline{H}} \boldsymbol{y}$

A symmetric $n \times n$ matrix $\boldsymbol{H}$ is **positive definite** if $\boldsymbol{y}^T \boldsymbol{H}\, \boldsymbol{y} > \boldsymbol{0}$ for any $\boldsymbol{y} \neq \boldsymbol{0}$

A symmetric $n \times n$ matrix $\boldsymbol{H}$ is **positive semi-definite** if $\boldsymbol{y}^T \boldsymbol{H}\, \boldsymbol{y} \geq \boldsymbol{0}$ for any $\boldsymbol{y} \neq \boldsymbol{0}$

A symmetric $n \times n$ matrix $\boldsymbol{H}$ is **negative definite** if $\boldsymbol{y}^T \boldsymbol{H}\, \boldsymbol{y} < \boldsymbol{0}$ for any $\boldsymbol{y} \neq \boldsymbol{0}$

A symmetric $n \times n$ matrix $\boldsymbol{H}$ is **negative semi-definite** if $\boldsymbol{y}^T \boldsymbol{H}\, \boldsymbol{y} \leq \boldsymbol{0}$ for any $\boldsymbol{y} \neq \boldsymbol{0}$

A symmetric $n \times n$ matrix $\boldsymbol{H}$ that is not negative semi-definite and not positive semi-definite is called **indefinite**

$y \cdot H y$:

$> 0 \longrightarrow$ PD

$\geq 0 \longrightarrow$ PSD

$< 0 \longrightarrow$ ND

$\leq 0 \longrightarrow$ NSD

$\left.\begin{array}{l} y \longrightarrow > 0 \\ y \longrightarrow < 0 \end{array}\right\}$ I la.eig(H)

$$f(\boldsymbol{x}^*) = \min_{\boldsymbol{x}} f(\boldsymbol{x})$$

First order necessary condition: $\boldsymbol{\nabla} f(\boldsymbol{x}) = \mathbf{0}$

Second order sufficient condition: $\boldsymbol{H}(\boldsymbol{x})$ **is positive definite**

How can we find out if the Hessian is positive definite?

$\boxed{\underline{\underline{H}}\, y = \lambda y} \longrightarrow (\lambda, y) \longrightarrow$ are eigenpairs of H

$y^T H y = \lambda y^T y = \lambda \|y\|_2^2 \Longrightarrow \boxed{\lambda = \dfrac{y^T H y}{\|y\|_2^2}}$ → always positive

$\star$ $\lambda_i > 0 \;\; \forall i \Rightarrow y^T H y > 0 \;\; \forall y \Longrightarrow$ H is pos. def $\Rightarrow x^*$ is minimizer

$\star$ $\lambda_i < 0 \;\; \forall i \Rightarrow y^T H y < 0 \;\; \forall y \Rightarrow$ H is neg def $\Rightarrow x^*$ is maximizer

$\star$ $\left.\begin{matrix}\lambda_i > 0 \\ \lambda_i < 0\end{matrix}\right\} \rightarrow$ H is indefinite $\rightarrow x^*$ is saddle point

Types of optimization problems

$$f(\boldsymbol{x}^*) = \min_{\boldsymbol{x}} f(\boldsymbol{x})$$

f: nonlinear, continuous and smooth

Gradient-free methods

Evaluate $f(\boldsymbol{x})$

Gradient (first-derivative) methods

Evaluate $f(\boldsymbol{x}), \boldsymbol{\nabla} f(\boldsymbol{x})$

Second-derivative methods

Evaluate $f(\boldsymbol{x}), \boldsymbol{\nabla} f(\boldsymbol{x}), \boldsymbol{\nabla}^2 f(\boldsymbol{x}) = \underline{\underline{H}}(\underline{x})$

Example (ND)

Consider the function $f(x_1, x_2) = 2x_1^3 + 4x_2^2 + 2x_2 - 24x_1$

Find the stationary point and check the sufficient condition

$$\nabla f = \begin{bmatrix} 6x_1^2 - 24 \\ 8x_2 + 2 \end{bmatrix}; \quad H = \begin{bmatrix} 12x_1 & 0 \\ 0 & 8 \end{bmatrix}$$

1) $\nabla f = 0 \Longrightarrow \begin{bmatrix} 6x_1^2 - 24 \\ 8x_2 + 2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Longrightarrow$ $6x_1^2 = 24 \rightarrow x_1^2 = 4 \rightarrow x_1 = \pm 2$

$8x_2 = -2 \Longrightarrow x_2 = -0.25$

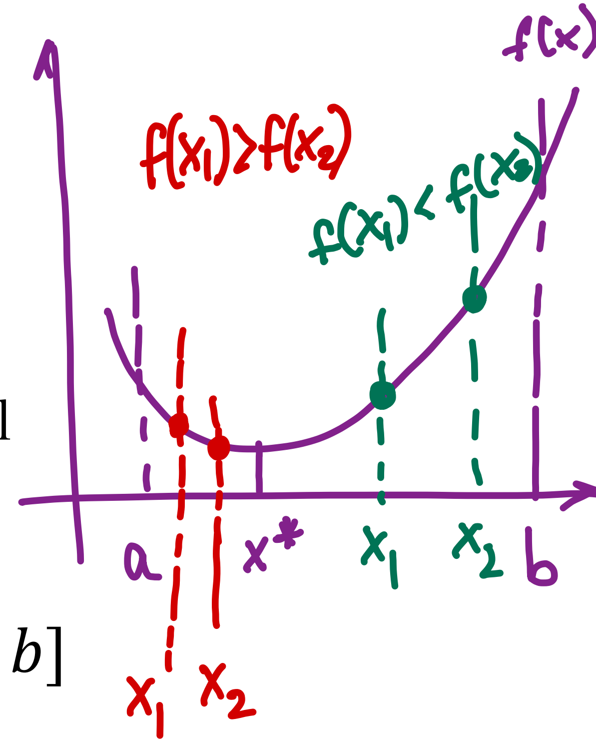
stationary points: $x^* = \begin{bmatrix} +2 \\ -0.25 \end{bmatrix}$ $x^* = \begin{bmatrix} -2 \\ -0.25 \end{bmatrix}$

2) $H\begin{pmatrix} -2 \\ -0.25 \end{pmatrix} = \begin{bmatrix} -24 & 0 \\ 0 & 8 \end{bmatrix} \Rightarrow$ indefinite $\Downarrow$ saddle point

$H\begin{pmatrix} 2 \\ -0.25 \end{pmatrix} = \begin{bmatrix} 24 & 0 \\ 0 & 8 \end{bmatrix}$ pos. def. $\Downarrow$ Minimizer!

Optimization (1D Methods)

Optimization in 1D: Golden Section Search



- Similar idea of bisection method for root finding
- Needs to bracket the minimum inside an interval
- Required the function to be unimodal

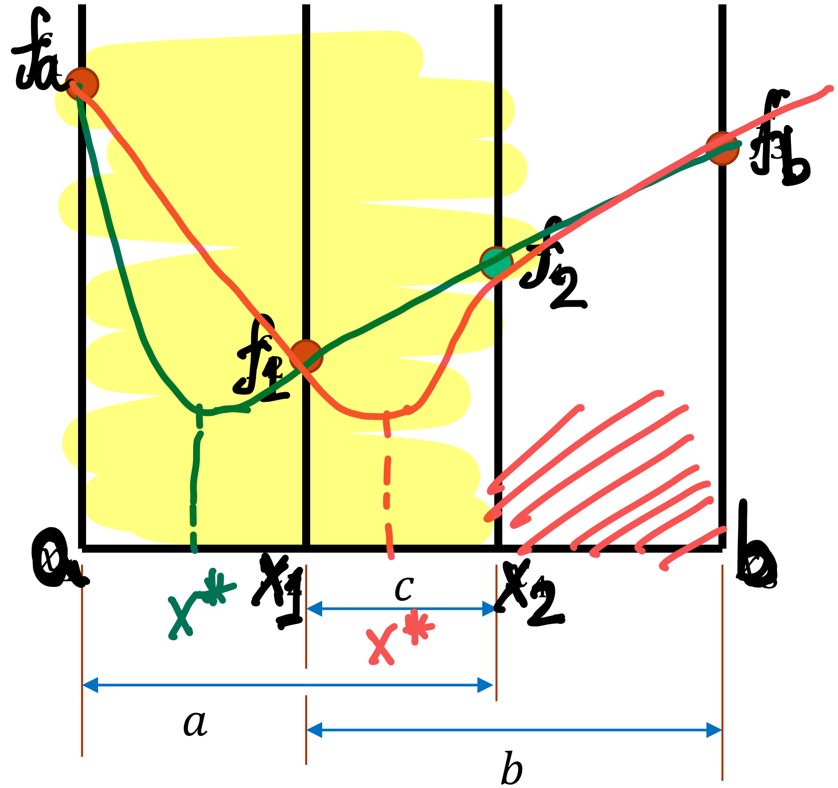
A function $f: \mathcal{R} \rightarrow \mathcal{R}$ is unimodal on an interval $[a, b]$

✓ There is a unique $\boldsymbol{x}^* \in [a, b]$ such that $f(\boldsymbol{x}^*)$ is the minimum in $[a, b]$

✓ For any $x_1, x_2 \in [a, b]$ with $\boxed{x_1 < x_2}$

- $x_2 < \boldsymbol{x}^* \Longrightarrow f(x_1) > f(x_2)$
- $x_1 > \boldsymbol{x}^* \Longrightarrow f(x_1) < f(x_2)$

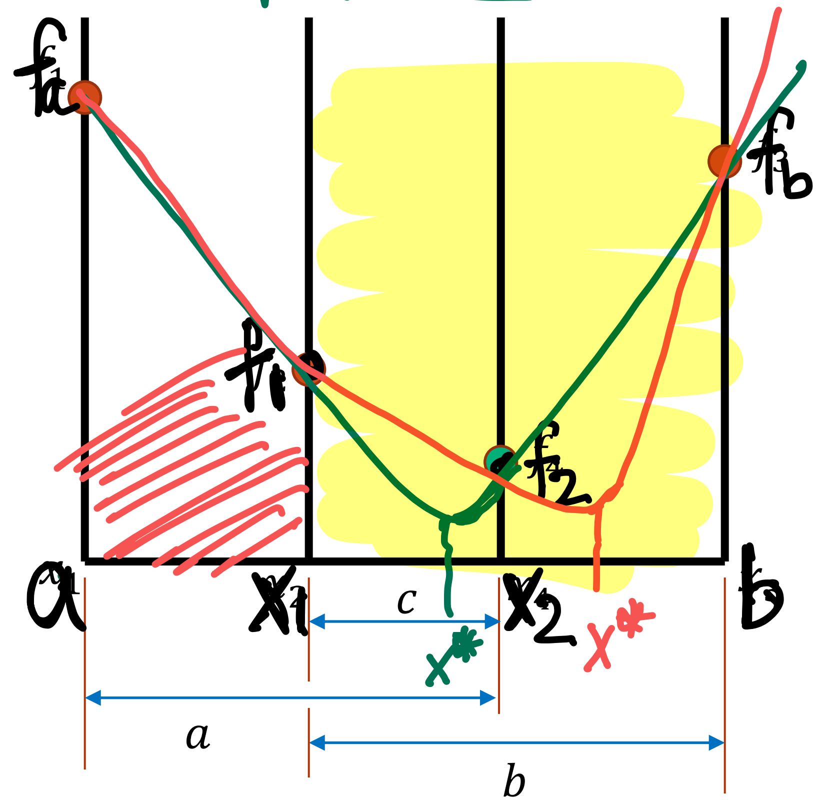
$\boxed{f_1 < f_2}$



$\boxed{f_1 < f_2}$ $\boxed{x_1 < x_2}$

$x^* \in [a, x_2]$

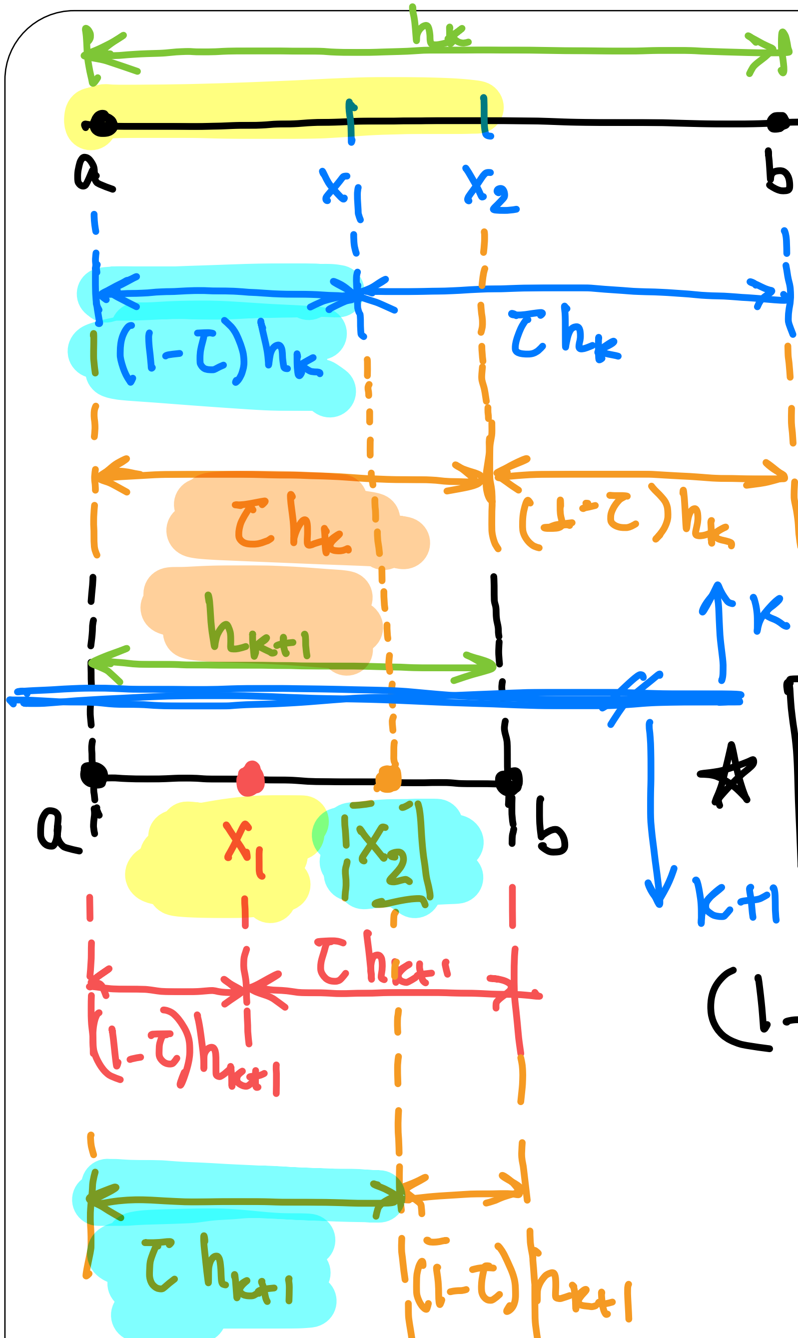
$\boxed{f_1 > f_2}$



$\boxed{f_1 > f_2}$ $\boxed{x_1 < x_2}$

$x^* \in [x_1, b]$

$\boxed{x_1, x_2 = ?}$



Propose the point x_1, x_2 s.t.

$$x_1 = a + (1-\tau) h_k$$

$$x_2 = a + \tau h_k$$

at the start $h_k = (b-a)$

$f_1 > f_2$ or $f_1 < f_2$

$[x_1, b]$ $[a, x_2]$

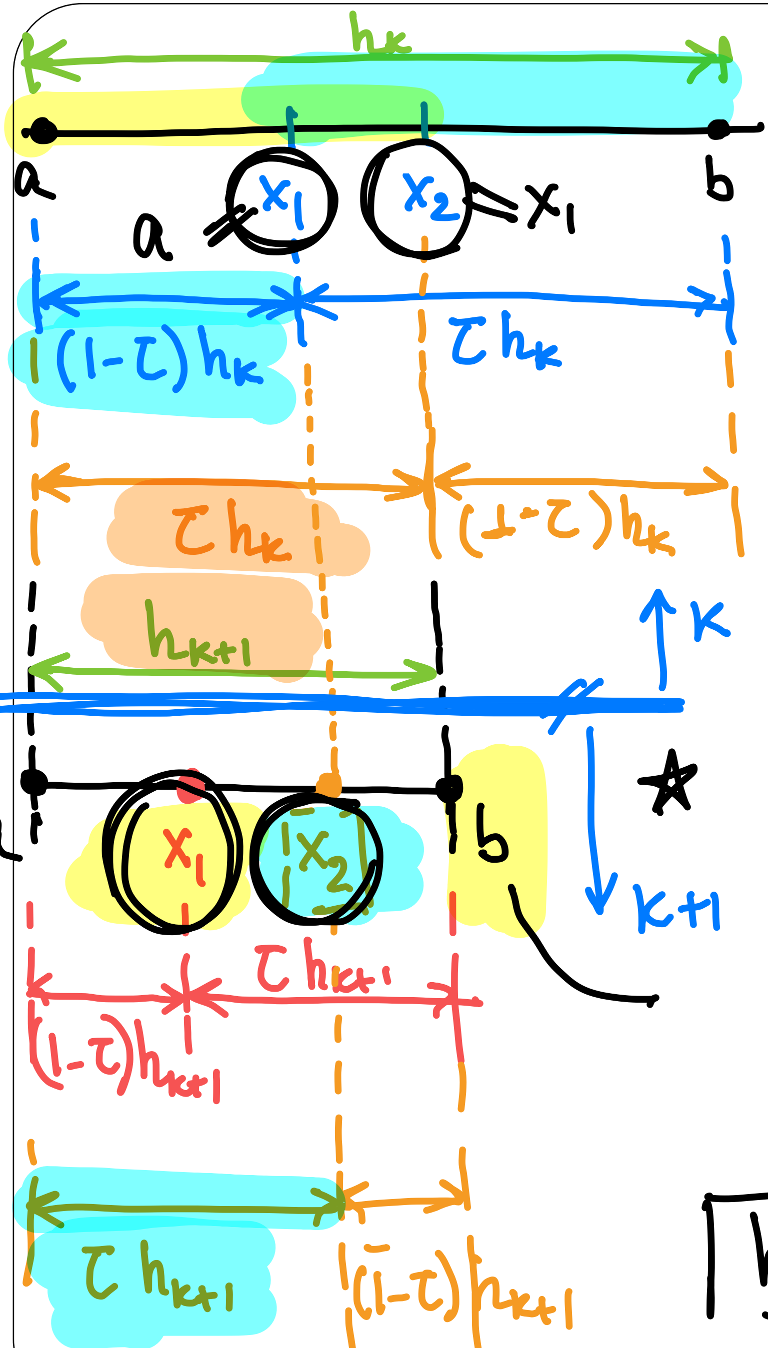
★ $$\boxed{h_{k+1} = \tau h_k}$$

Every iteration interval gets reduced by τ

$$(1-\tau) h_k = \tau h_{k+1} = \tau(\tau h_k)$$

$$(1-\tau) h_k = \tau^2 h_k$$

$$(1-\tau) = \tau^2 \rightarrow \boxed{\tau = 0.618}$$



interval (a,b) $\boxed{\tau = 0.618}$

$h_0 = (b-a)$

$x_1 = a + (1-\tau)h_0$

$x_2 = a + \tau h_0$

$f_1 = f(x_1)$ $f_2 = f(x_2)$

if $f_1 < f_2$: $\longrightarrow x^* \in [a, x_2]$

$b = x_2$

$x_2 = x_1 \longrightarrow f_2 = f_1$

$h_{k+1} = \tau h_k$

$x_1 = a + (1-\tau)h_{k+1}$

$f_1 = f(x_1)$

if $f_1 > f_2$: $\longrightarrow x^* \in [x_1, b]$

$a = x_1$

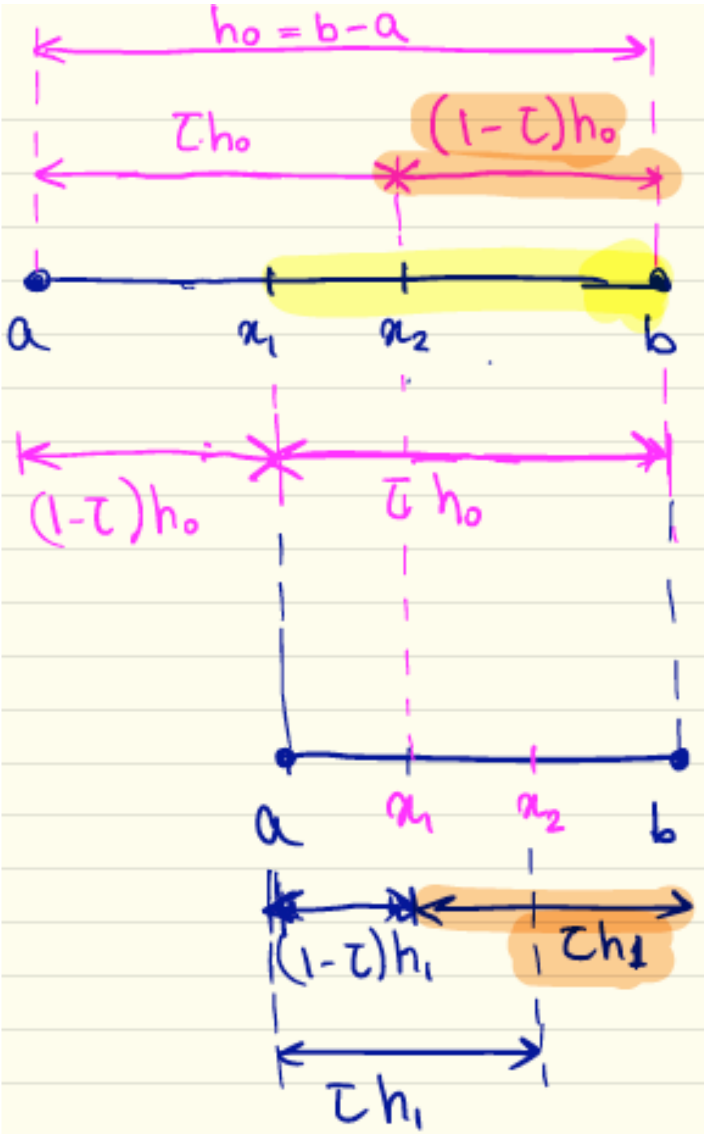
$x_1 = x_2 \longrightarrow f_1 = f_2$

$h_{k+1} = \tau h_k$

$x_2 = a + \tau h_k$ $f_2 = f(x_2)$

$\boxed{h_{k+1} < tol}$

Golden Section Search



Propose:

$x_1 = a + (1-\tau)h_0$

$x_2 = a + \tau h_0$

Evaluate $f_1 = f(x_1)$

$f_2 = f(x_2)$

if $(f_1 > f_2)$:

$a = x_1$

$x_1 = x_2 \rightarrow$ already have func. value!

$h_1 = b - a$

$x_2 = a + \tau h_1$

$f_2 = f(x_2) \rightarrow$ only one

if $(f_1 < f_2)$:

$b = x_2$

$x_2 = x_1$

$x_1 = a + (1-\tau)h_1$

$f_1 = f(x_1)$

Golden Section Search

What happens with the length of the interval after one iteration?

$$h_1 = \tau\, h_o$$

Or in general: $h_{k+1} = \tau\, h_k$

Hence the interval gets reduced by $\boldsymbol{\tau}$

(for bisection method to solve nonlinear equations, τ=0.5)

For recursion:

$$\tau\, h_1 = (1 - \tau)\, h_o$$
$$\tau\, \tau\, h_o \; = (1 - \tau)\, h_o$$
$$\tau^2 = (1 - \tau)$$
$$\boldsymbol{\tau = 0.618}$$

Golden Section Search

$\underline{\underline{x^*}} \longrightarrow \underline{\underline{h_k}} < tol$

$x^* \in h_k$

evaluate $f(x)$

- Derivative free method!

$e_k = h_k$

$\frac{e_{k+1}}{e_k^r} = \frac{h_{k+1}}{h_k^r} = \frac{\tau h_k}{h_k^r}$

$r = 1 \longrightarrow \tau$

- Slow convergence:

$$\lim_{k\to\infty} \frac{|e_{k+1}|}{|e_k|} = 0.618 \quad r = 1 \; (linear\ convergence)$$

- Only one function evaluation per iteration

x_1 , x_2 cheap ,

Example

Consider running golden section search on a function that is unimodal. If golden section search is started with an initial brakcet of $[-10, 10]$, what is the length of the new bracket after 1 iteration?

A) 20
B) 10
$\boxed{\text{C) } 12.36}$
D) 7.64

$a = -10$
$b = 10$ $\Longrightarrow$ $h_0 = 20$

$h_1 = ?$

$h_1 = \tau h_0 \Longrightarrow 0.618 \times 20 = 12.36$

Newton's Method

$x_{k+1} = x_k + h$

Using Taylor Expansion, we can approximate the function f with a quadratic function about x_0

$$\underbrace{f(x)}_{\text{nonlinear}} \approx \overbrace{f(x_0) + f'(x_0)\,(x - x_0) + \frac{1}{2} f''(x_0)\,(x - x_0)^2}^{\text{quadratic approximation}} = \hat{f}$$

And we want to find the minimum of the quadratic function using the first-order necessary condition

$$f'(x) = 0 \longrightarrow \hat{f}' = 0$$

$$f'(x_0) + \cancel{\tfrac{1}{2}} f''(x_0)(x - x_0)\cancel{2} = 0$$

$$f'(x_0) + f''(x_0)(x - x_0) = 0$$

$$x - x_0 = -\frac{f'(x_0)}{f''(x_0)} \Longrightarrow \boxed{x = x_0 - \frac{f'(x_0)}{f''(x_0)}}$$

stationary point

Newton step

Newton's Method

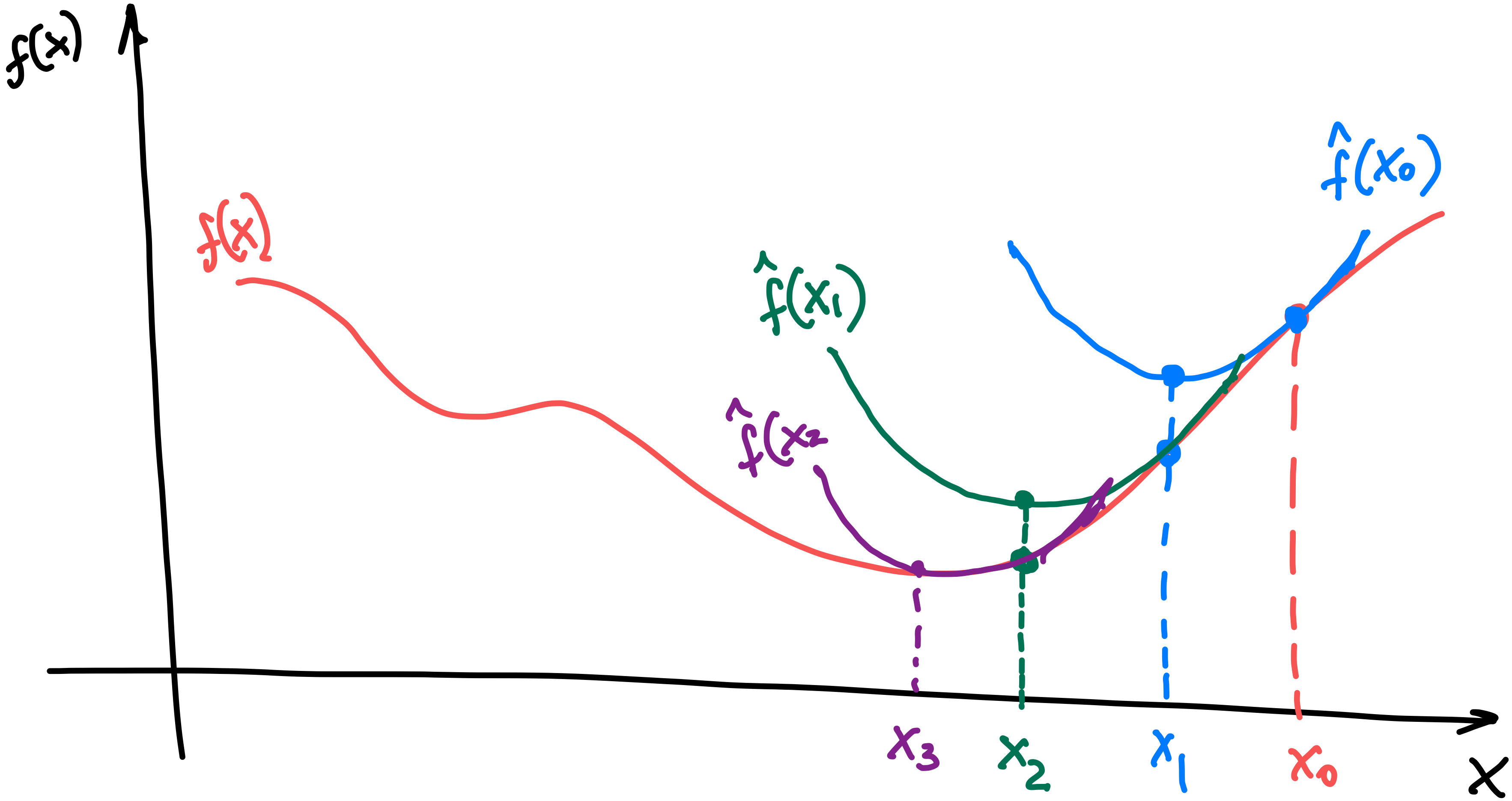
- **Algorithm:**

$x_0 = \text{starting guess}$

$x_{k+1} = x_k - f'(x_k)/f''(x_k)$

- **Convergence:**
  - Typical quadratic convergence
  - Local convergence (start guess close to solution)
  - May fail to converge, or converge to a maximum or point of inflection

Newton's Method (Graphical Representation)



sequence of opt.
using quad. approx $\hat{f}$

Example

Consider the function $f(x) = 4\,x^3 + 2\,x^2 + 5\,x + 40$

If we use the initial guess $x_0 = 2$, what would be the value of x after one iteration of the Newton's method?

$x_1 = ?$

$f'(x) = 12x^2 + 4x + 5$

$f''(x) = 24x + 4$

$$h = -\frac{f'(x)}{f''(x)} = -\frac{(12(4) + 4(2) + 5)}{24(2) + 4} = -\frac{61}{52}$$

$$x_1 = x_0 + h \Longrightarrow x_1 = 2 - \frac{61}{52} \longrightarrow \boxed{x_1 = 0.8269}$$

Optimization (ND Methods)

Optimization in ND: Steepest Descent Method

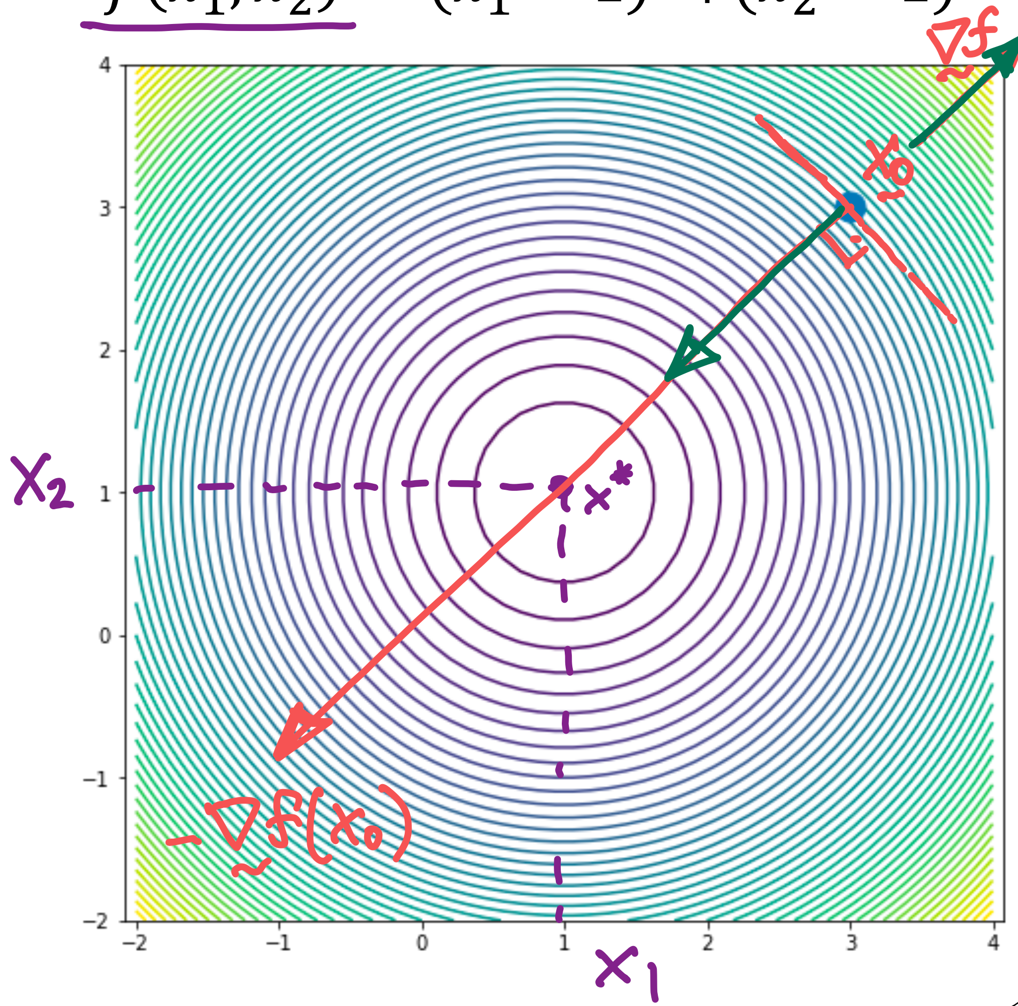
$\min_{x} f(x)$

$\boxed{-\nabla f}$

Given a function $f(\boldsymbol{x})\colon \mathcal{R}^n \to \mathcal{R}$ at a point $\boldsymbol{x}$, the function will decrease its value in the direction of steepest descent: $-\boldsymbol{\nabla} f(\boldsymbol{x})$

What is the steepest descent direction?

$$\underline{f(x_1, x_2)} = (x_1 - 1)^2 + (x_2 - 1)^2$$



Steepest Descent Method

$\underset{\sim}{x}_2 = \underset{\sim}{x}_1 - \nabla f(\underset{\sim}{x}_1)$

Start with initial guess:

$$\boldsymbol{x}_0 = \begin{bmatrix} 3 \\ 3 \end{bmatrix}$$

Check the update:

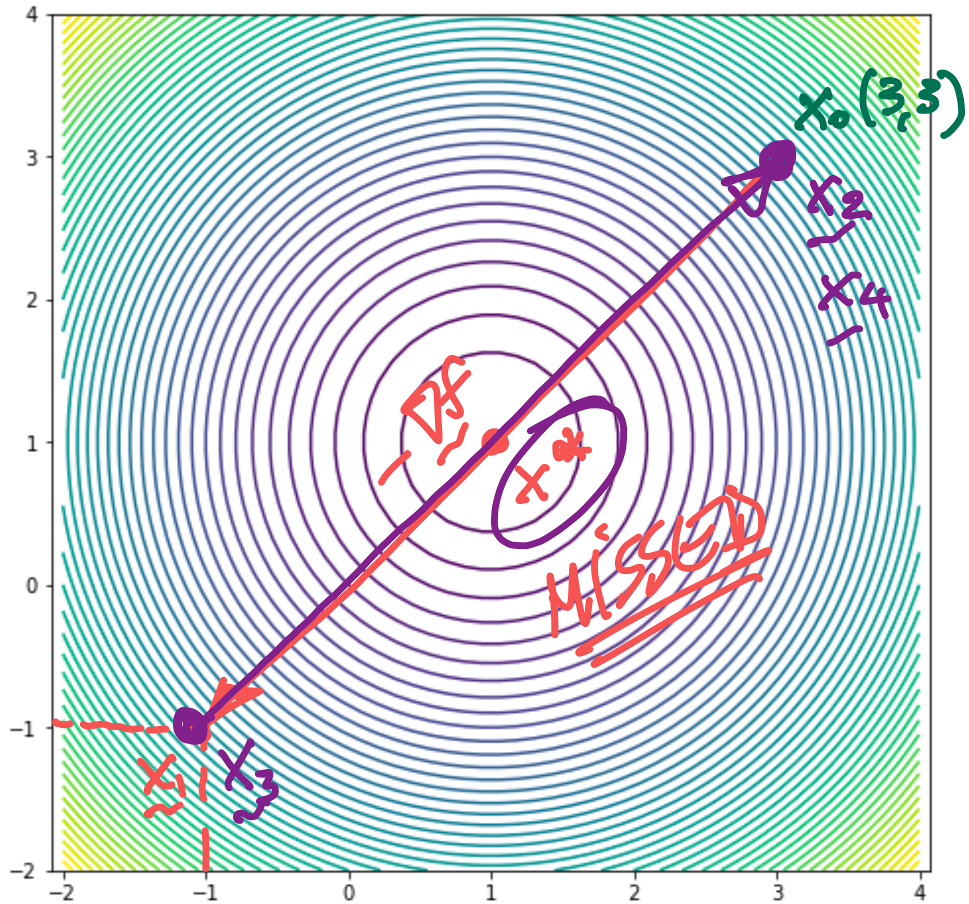
$$\boxed{\underset{\sim}{x}_1 = \underset{\sim}{x}_0 - \underset{\sim}{\nabla} f(\underset{\sim}{x}_0)}$$

$$\underset{\sim}{\nabla} f = \begin{bmatrix} 2(x_1 - 1) \\ 2(x_2 - 1) \end{bmatrix}$$

$$\underset{\sim}{\nabla} f(\underset{\sim}{x}_0) = \begin{bmatrix} 4 \\ 4 \end{bmatrix}$$

$$\underset{\sim}{x}_1 = \begin{bmatrix} 3 \\ 3 \end{bmatrix} - \begin{bmatrix} 4 \\ 4 \end{bmatrix} = \begin{bmatrix} -1 \\ -1 \end{bmatrix}$$

$$f(x_1, x_2) = (x_1 - 1)^2 + (x_2 - 1)^2$$



Steepest Descent Method

Update the variable with:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha_k \nabla f(\boldsymbol{x}_k)$$

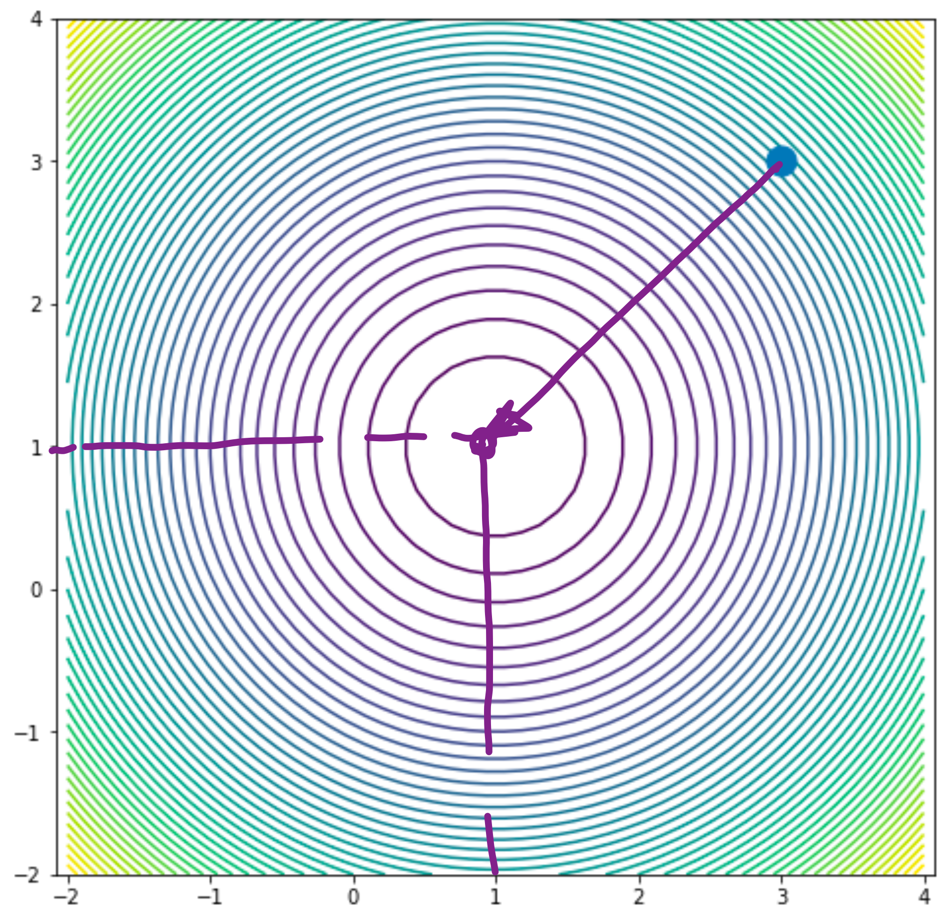
How far along the gradient should we go? What is the "best size" for α_k?

$$x_1 = x_0 - \underline{0.5}\, \nabla f(x_0)$$

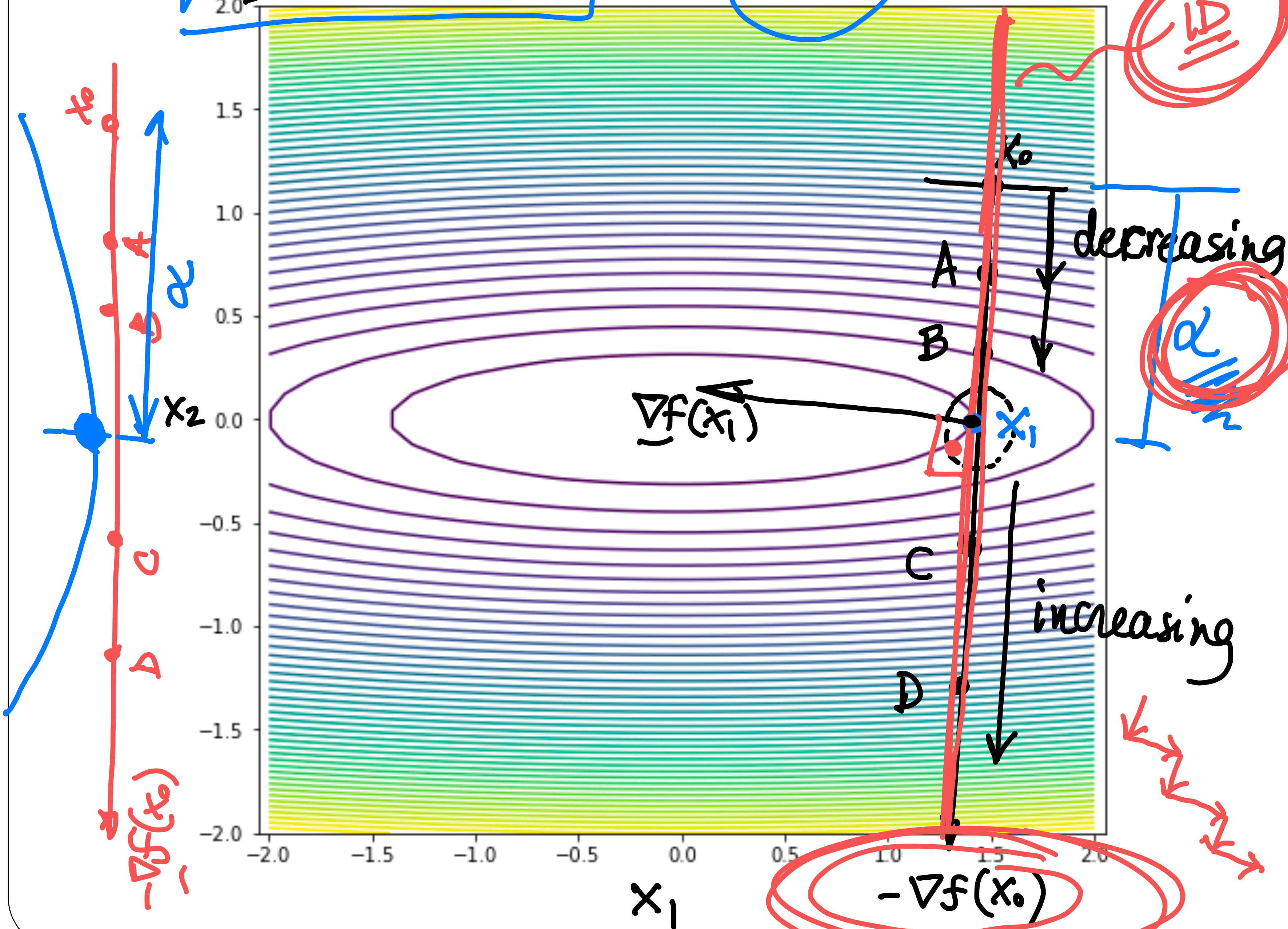
$$\boxed{\alpha = 0.5}$$

How can we get α?

$$f(x_1, x_2) = (x_1 - 1)^2 + (x_2 - 1)^2$$



Find $\boxed{x_1 = x_0 - \alpha \nabla f(x_0)}$ s.t $f(x_1)$ is minimized



Steepest Descent Method

Algorithm:

1D optimization problem

Initial guess: $\boldsymbol{x}_0$

several fc eval.

Evaluate: $\boldsymbol{s}_k = -\nabla f(\boldsymbol{x}_k)$

Perform a line search to obtain α_k (for example, Golden Section Search)

$$\boxed{\alpha_k} = \underset{\alpha}{\operatorname{argmin}}\, f(\underbrace{\boldsymbol{x}_k + \alpha\, \boldsymbol{s}_k}_{x_{k+1} = x_k + \alpha s_k})$$

Update: $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k\, \boldsymbol{s}_k$

Line Search

$x_{k+1} = x_k - \alpha_k \nabla f(x_k)$

we want to find α_k s.t.

$f(x_{k+1})$

$$\min_{\alpha} f(\underbrace{x_k - \alpha_k \nabla f(x_k)}_{x_{k+1}})$$

1st order condition $\frac{df}{d\alpha} = 0 \rightarrow$ gives α

$$\frac{df}{d\alpha} = \frac{\partial f}{\partial x_{k+1}} \frac{\partial x_{k+1}}{\partial \alpha} = \nabla f(x_{k+1}) \cdot \nabla f(x_k) = 0$$

$$\nabla f(x_{k+1}) \cdot \nabla f(x_k) = 0$$

$\nabla f(x_{k+1})$ is orthogonal to $\nabla f(x_k)$

zig-zag pattern convergence.

Example

$$\min_{x_1, x_2} f(x_1, x_2)$$

Consider minimizing the function

$$f(x_1, x_2) = 10(x_1)^3 - (x_2)^2 + x_1 - 1$$

Given the initial guess

$$x_1 = 2, x_2 = 2$$

$$\underset{\sim}{x_0} = \begin{bmatrix} 2 \\ 2 \end{bmatrix}$$

what is the direction of the first step of gradient descent?

$$\nabla f = \begin{bmatrix} 30x_1^2 + 1 \\ -2x_2 \end{bmatrix} \qquad \underline{\nabla} f(\underset{\sim}{x_0}) = \begin{bmatrix} 121 \\ -4 \end{bmatrix}$$

$$\boxed{\text{steepest descent direction} \Longrightarrow \begin{bmatrix} -121 \\ +4 \end{bmatrix}}$$

Newton's Method

Using Taylor Expansion, we build the approximation:

$$\underbrace{f(\underset{\sim}{x}+\underset{\sim}{s})}_{\text{nonlinear}} = \underbrace{f(\underset{\sim}{x}) + \nabla f(\underset{\sim}{x})^T \underset{\sim}{s} + \frac{1}{2}\underset{\sim}{s}^T \underset{=}{H} \underset{\sim}{s}} = \hat{f}(\underset{-}{s})$$

quadratic approx of f

1st order condition: $\nabla_s \hat{f} = 0$

$$\boxed{\underset{\sim}{\nabla} f(\underset{\sim}{x}) + \underset{=}{H}\underset{\sim}{s} = 0} \rightarrow \underset{=}{H} \text{ is symmetric}$$

$$\underset{=}{H} = \underset{=}{H}^T$$

$$\boxed{\underset{=}{H}(\underset{\sim}{x})\,\underset{\sim}{s} = -\nabla f(\underset{\sim}{x})} \rightarrow \text{solve linsys to find Newton step } \underset{\sim}{s}$$

Newton's Method

$H_{ij} = \frac{\partial^2 f}{\partial x_i \partial x_j}$ → $O(n^2)$

Algorithm:

Initial guess: $\boldsymbol{x}_0$

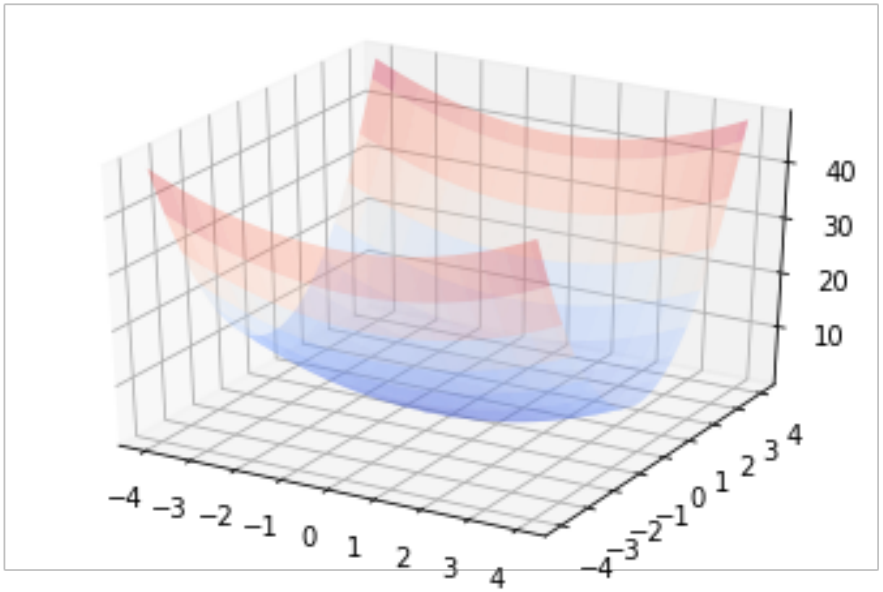
Solve: $\boxed{\boldsymbol{H}_f(\boldsymbol{x}_k)}\, \boldsymbol{s}_k = -\nabla f(\boldsymbol{x}_k)$ → solve $O(n^3)$ $\boldsymbol{s}$

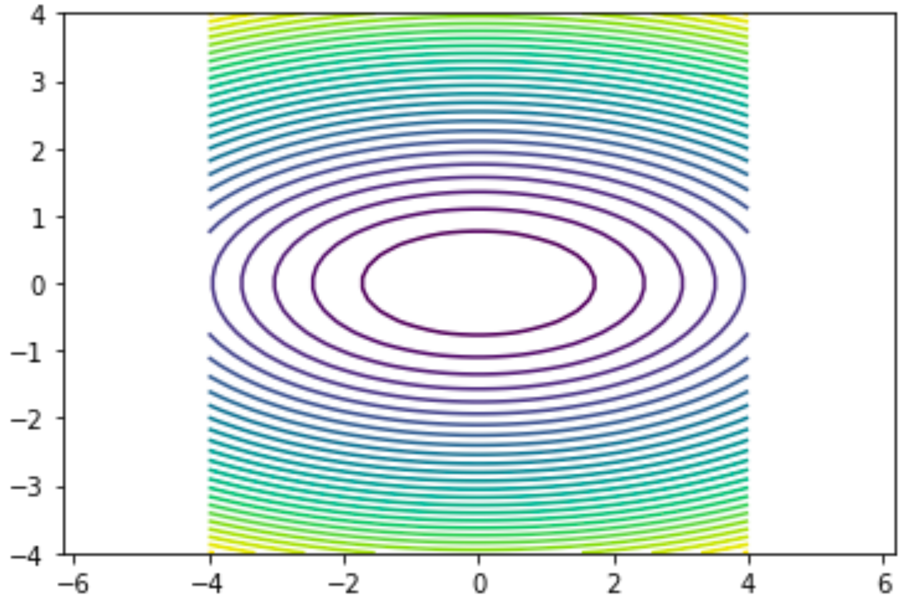
Update: $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \boldsymbol{s}_k$

Note that the Hessian is related to the curvature and therefore contains the information about how large the step should be.

Try this out!

$$f(x, y) = 0.5x^2 + 2.5y^2$$





When using the Newton's Method to find the minimizer of this function, estimate the number of iterations it would take for convergence?

A) 1 B) 2-5 C) 5-10 D) More than 10 E) Depends on the initial guess

Newton's Method Summary

Algorithm:

Initial guess: $\boldsymbol{x}_0$

Solve: $\boldsymbol{H_f}(\boldsymbol{x}_k)\,\boldsymbol{s}_k = -\boldsymbol{\nabla} f(\boldsymbol{x}_k)$

Update: $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \boldsymbol{s}_k$

About the method…

- Typical quadratic convergence ☺
- Need second derivatives ☹
- Local convergence (start guess close to solution)
- Works poorly when Hessian is nearly indefinite
- Cost per iteration: $O(n^3)$