# Solving Nonlinear Equations

- first only 1 equation $f(x) = 0$ ... $S = \frac{1}{2} r^2 (\theta - \sin\theta)$
  ↳ no analytic solution for $\theta$

## Estimating errors

$x_T$ ... true solution → $f(x_T) = 0$

$x_N$ ... numerical solution → $f(x_N) = \varepsilon$

- True error $= x_T - x_N$ ... but we dont know $x_T$
- Tolerance $= |f(x_T) - f(x_N)| = |\varepsilon|$
- True relative error $= \left| \frac{x_T - x_N}{x_N} \right|$
- Estimated relative error $= \left| \frac{x_N^{(n)} - x_N^{(n-1)}}{x_N^{(n)}} \right|$ ← tohle se reálně počítá

  $x_N$ v kroku $N-1$

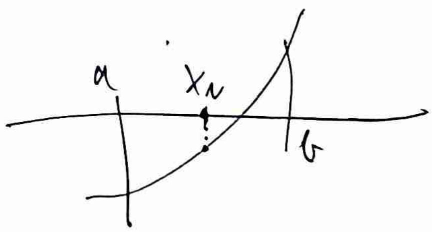
  ↳ stop when this $< 0.001$ for example

## Methods

- **Bracketing methods**
  → assume that $f$ is continuous & $\exists!$ solution
  ↳ iteratively reduce interval size – půlení intervalů
  → chose the first interval by finding points $a, b$ s.t. $f(a) \cdot f(b) < 0$
  ↳ since $f$ is continuous, the solution $\in [a, b]$

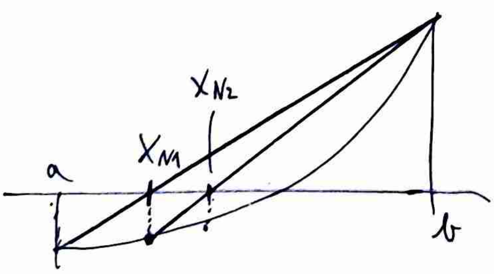
### ① The Bisection Method

1. $x_N \leftarrow \frac{a+b}{2}$
2. if $f(x_N) \cdot f(b) < 0$:
   $a \leftarrow x_N$
   else:
   $b \leftarrow x_N$
3. goto 1




### ② The Regula Falsi Method

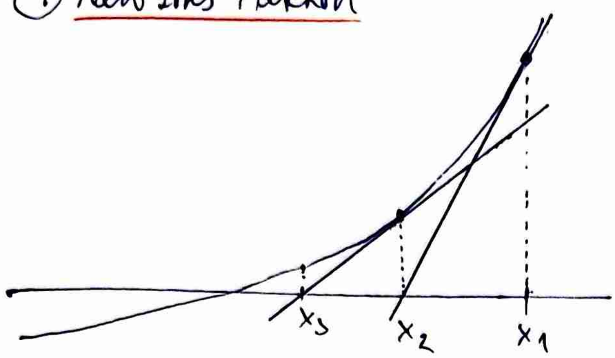
→ stejná myšlenka, ale počítá jinak $x_N$




$$x_N = \frac{a \cdot f(b) - b \cdot f(a)}{f(b) - f(a)}$$

- Open methods
  - initial guess and then improve it
  - more efficient but may not yield a solution

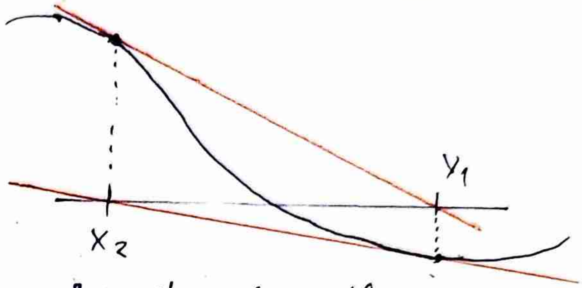
① Newtons Method




$$\frac{dy}{dx} = f'(x) \Rightarrow \frac{f(x_1)}{x_1 - x_2} = f'(x_1) \Rightarrow \frac{f(x_1)}{f'(x_1)} = x_1 - x_2$$

$$\Rightarrow \underline{\underline{x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}}}$$

↳ might not converge:



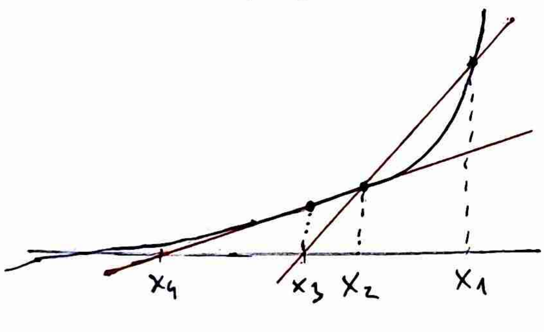

Fact: If $f'$ and $f''$ are continuous & $f'(x_T) \neq 0$ then

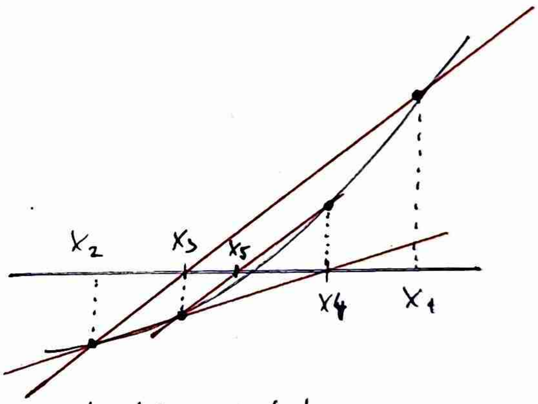
$\exists \varepsilon > 0$ s.t. $|x_1 - x_T| < \varepsilon \Rightarrow$ Newtons method will converge

② The Secant Method

→ same idea as newton but instead of a tangent uses a secant

⇒ needs 2 initial guesses




1. Chose $(x_1, f(x_1))$, $(x_2, f(x_2))$ as starting points
2. Calculate $(x_3, 0)$ on the same line to get $(x_3, f(x_3))$

$$\frac{f(x_2) - f(x_1)}{x_2 - x_1} = \frac{dy}{dx} = \frac{0 - f(x_2)}{x_3 - x_2} \Rightarrow x_3 - x_2 = -f(x_2)\frac{x_2 - x_1}{f(x_2) - f(x_1)}$$

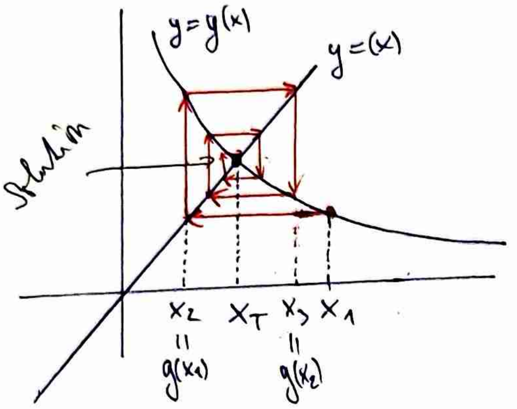
$$\Rightarrow \underline{\underline{x_{n+1} = x_n - f(x_n)\frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})}}}$$

→ basically approximates $\frac{dx}{dy} = \frac{1}{f'(x)}$

## ③ The Fixed-Point Iteration Method

rewrite $f(x)=0$ as $x=g(x)$

↳ $xe^x + 2x - 5 = 0 \Rightarrow x(e^x+2) = 5 \Rightarrow x = \frac{5}{e^x+2}$



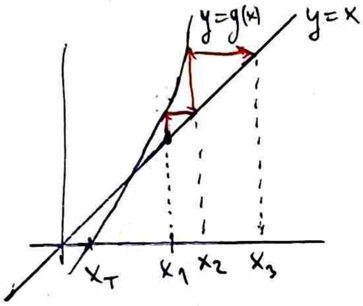

Solution = intersection of $y=x$ and $y=g(x)$

$x_1 \Rightarrow$ get $g(x_1)$

$x_2 \leftarrow g(x_1)$ ... point on $y=x$ s.t. $y=g(x_1)$

$\Rightarrow \underline{\underline{x_{m+1} \leftarrow g(x_m)}}$ ... recall $x_T = g(x_T)$

→ this method can diverge easily




$\Rightarrow$ if $\underline{|g'(x)| < 1}$ in a neighborhood of $x_T$ and $x_1 \in$ this neighborhood $\Rightarrow$ it will converge

❗ there are often multiple ways of expressing $x=g(x)$ and not all will converge

$$xe^x + 2x - 5 = 0 \Rightarrow x = \frac{5-xe^x}{2} \quad \& \quad x = \frac{5}{e^x+2} \quad \& \quad x = \frac{5-2x}{e^x}$$

↳ only the second option will converge

- **Systems of Nonlinear Equations**

$f_1(x,y) = y - \sinh\left(\frac{x}{2}\right) = 0$

$f_2(x,y) = 9x^2 + 25y^2 - 225 = 0$

$\tilde{a} = \tilde{x}_N$

### ① Newtons Method

$f_1(x_1, \dots x_m) = 0$

$\vdots$

$f_m(x_1, \dots x_m) = 0$

↓

get $\Delta\tilde{x}$

↓

$\tilde{x}_{m+1} = \tilde{x}_m + \Delta\tilde{x}$

1. Guess $(x_1^N, x_2^N, \dots x_m^N) =: \tilde{x}_N$
2. estimate $f_1, \dots f_m$ using their total differencial
$$f(\tilde{x}) \approx f(\tilde{a}) + \nabla f(\tilde{a}) \cdot (\tilde{x} - \tilde{a}) = f(\tilde{a}) + \sum_{k=1}^{m} \frac{\partial f(\tilde{a})}{\partial x_k}(x_k - a_k)$$
3. plug in the true solution
$$0 = f(\tilde{x}_T) \approx f(\tilde{x}_N) + \nabla f(\tilde{x}_N) \cdot \overbrace{(\tilde{x}_T - \tilde{x}_N)}^{\Delta\tilde{x}}$$

$\Rightarrow \underline{\underline{\nabla f(\tilde{x}_N) \cdot \Delta\tilde{x} = -f(\tilde{x}_N)}}$

↳ remember for 1 variable $\Delta x = -\frac{f(x_N)}{f'(x_N)} \Rightarrow f'(x_N) \cdot \Delta x = -f(x_N)$

→ finding the $\Delta\tilde{X}$

$$f_1 \Rightarrow \nabla f_1(\tilde{X}_N)\cdot\Delta\tilde{X} = -f_1(\tilde{X}_N)$$
$$\vdots \qquad\qquad \vdots$$
$$f_m \Rightarrow \nabla f_m(\tilde{X}_N)\,\Delta\tilde{X} = -f_m(\tilde{X}_N)$$

$$\Rightarrow \begin{pmatrix} \text{---}\nabla f_1(\tilde{X}_N)\text{---} \\ \text{---}\nabla f_2(\tilde{X}_N)\text{---} \\ \vdots \\ \text{---}\nabla f_m(\tilde{X}_m)\text{---} \end{pmatrix} \cdot \begin{pmatrix} \Delta X_1 \\ \vdots \\ \Delta X_m \end{pmatrix} = \begin{pmatrix} -f_1(\tilde{X}_N) \\ -f_2(\tilde{X}_N) \\ \vdots \\ -f_m(\tilde{X}_N) \end{pmatrix}$$

EX: 2 equations, solve for $\Delta x$ with Cramers rule ↳ Jacobian of $f_1, \dots f_m$

$$f(x,y) = x^2 + 2x + 2y^2 - 26 = 0$$
$$g(x,y) = x^3 - y^2 + 4y - 19 = 0$$
, initial guess $x_0 = 1$, $y_0 = 1$

$$J(f,g)\cdot\begin{pmatrix}\Delta x \\ \Delta y\end{pmatrix} = \begin{pmatrix}-f_1 \\ -f_2\end{pmatrix} \Rightarrow \begin{bmatrix} \frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} \\ \frac{\partial g}{\partial x} & \frac{\partial g}{\partial y} \end{bmatrix} \cdot \begin{bmatrix}\Delta x \\ \Delta y\end{bmatrix} = \begin{bmatrix}-f_1 \\ -f_2\end{bmatrix}$$

Cramer:

$$\Delta x = \frac{1}{|J|}\cdot\left(-f_1\cdot\frac{\partial g}{\partial y} + f_2\frac{\partial f}{\partial y}\right) \quad \& \quad \Delta y = \frac{1}{|J|}\cdot\left(-f_2\cdot\frac{\partial f}{\partial x} + f_1\frac{\partial g}{\partial x}\right)$$

Evaluation

1. Evaluate the partial derivatives at $x_n, y_n$
2. Calculate the Jacobi determinant using ↗
3. Evaluate $f_1$ and $f_2$ at $x_n, y_n$
4. Calculate $\Delta x$ and $\Delta y$ → improve $x_n, y_n$

goto 1

## ② The Fixed-Point Iteration Method

$$\begin{matrix} f_1(x_1, x_2, \dots x_m) = 0 \\ \vdots \\ f_m(x_1, x_2, \dots x_m) = 0 \end{matrix} \leadsto \begin{matrix} x_1 = g_1(x_1, \dots x_m) \\ x_2 = g_2(x_1 \dots x_m) \\ \vdots \\ x_m = g_m(x_1, \dots x_m) \end{matrix} \leadsto \tilde{X} = G(\tilde{X}) \quad , G: \mathbb{R}^n \to \mathbb{R}^n$$

⇒ choose starting point $\tilde{x}_0 \in \mathbb{R}^n$ → $\underline{\tilde{X}_{m+1} \leftarrow G(\tilde{X}_m)}$

The algorithm will converge under the following sufficient but not necessary conditions:

1) $g_i$ and $\frac{\partial g_i}{\partial x_k}$ are all continuous within a neighborhood of $X_T$
2) $\forall i: \sum_k \left|\frac{\partial g_i}{\partial x_k}\right| \leq 1$ in a neighborhood of $X_T$
3) $X_0$ starts in this neighborhood

# • Solving Systems of Linear Equations

## • Direct methods

(1)
- Gauss elimination ⇝ upper triangular matrix + back-substitution
- Gauss-Jordan ⇝ pivots will be divided to be 1
  ↳ eliminate off-diagonal terms in ALL equations
  $O(n^3)$
  ⇒ result = diagonal matrix

(2) LU Decomposition

$Ax = b$ → find $L \cdot U = A$ s.t. $L$ = lower-$\Delta$ and $U$ = upper-$\Delta$

$L\underbrace{Ux}_{y} = b$ ⇒ $Ly = b$ … find $y$ – forward substitution
$Ux = y$ … find $x$ – backward substitution

→ this is better than Gauss if we will be solving $Ax = b$ for different $b$-s a lot
→ there is some precomputing for finding the LU-decomp $O(n^3)$
→ but evaluating is only $O(n^2)$

a) Using Gauss

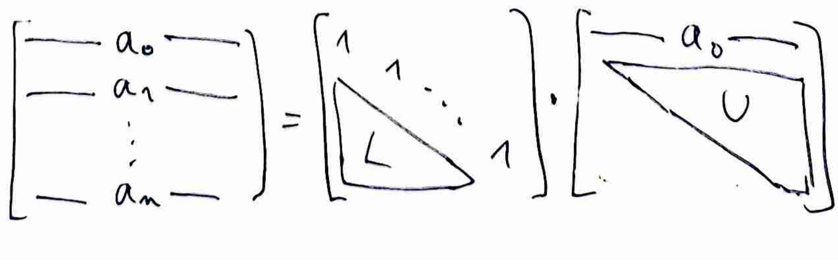
$$\begin{bmatrix} 1 & 2 & 3 \\ -1 & -6 & -1 \\ 2 & -4 & 9 \end{bmatrix} \overset{(-1)}{\sim} \begin{bmatrix} 1 & 2 & 3 \\ 0 & -4 & 2 \\ 2 & -4 & 9 \end{bmatrix} \overset{(2)}{\sim} \begin{bmatrix} 1 & 2 & 3 \\ 0 & -4 & 2 \\ 0 & -8 & 3 \end{bmatrix} \overset{(2)}{\sim} \begin{bmatrix} 1 & 2 & 3 \\ 0 & -4 & 2 \\ 0 & 0 & -1 \end{bmatrix}$$

in the third row is the second row included $\frac{-8}{-4} = 2$ times

$$\begin{bmatrix} 1 & 2 & 3 \\ 0 & -4 & 2 \\ 0 & 0 & -1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 2 & 2 & 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 3 \\ -1 & -6 & -1 \\ 2 & -4 & 9 \end{bmatrix} \approx \begin{array}{l} (1,2,3) = 1\cdot(1,2,3) \\ (-1,-6,-1) = -1\cdot(1,2,3) + 1\cdot(0,-4,2) \\ (2,-4,9) = 2(1,2,3) + 2(0,-4,2) + 1\cdot(0,0,-1) \end{array}$$

→ can not use scalar multiplication of lines during the procedure
→ there will allways be ones on the diagonal of $L$

⇒ result 

b) Crout's Method

Has the form
$$\begin{bmatrix} | & | & & | \\ a_0 & a_1 & \cdots & a_n \\ | & | & & | \end{bmatrix} = \begin{bmatrix} | & & \\ a_0 & L & \\ | & & \end{bmatrix} \cdot \begin{bmatrix} 1 & & U \\ & 1 & \\ & & \ddots & \\ & & & 1 \end{bmatrix}$$

→ first fill in the first column of $L$ and diagonal of $U$

→ then calculate the first row of $U$

→ then work the way from the Top-left corner

$4 \cdot x = -2$

$$\begin{array}{cccc|cccc}
 & & & & 1 & -\frac{1}{2} & \frac{1}{2} & 0 \\
 & & & & 0 & 1 & 0 & 0 \\
 & & & & 0 & 0 & 1 & -\frac{1}{2} \\
 & & & & 0 & 0 & 0 & 1 \\
\hline
4 & 0 & 0 & 0 & 4 & -2 & 2 & 0 \\
-2 & 1 & 0 & 0 & -2 & 2 & -1 & 0 \\
2 & 0 & 4 & 0 & 2 & -1 & 5 & -2 \\
0 & 0 & -2 & 1 & 0 & 0 & -2 & 2
\end{array}$$

③ Using inverse matrix → good for a lot of calculations $Ax=b$ with different $b$-s

$Ax = b \Rightarrow$ calculate $A^{-1} \Rightarrow x = A^{-1}b$

→ usable only for invertible $A$

→ calculating $A^{-1}$ ... $O(n^3)$

→ evaluating $\tilde{x} = A^{-1}b$ ... $O(n^2)$

• Iterative methods

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 &= b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 &= b_2 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 &= b_3 \end{aligned} \Rightarrow \begin{aligned} x_1 &= (b_1 - a_{12}x_2 - a_{13}x_3)/a_{11} \\ x_2 &= (b_2 - a_{21}x_1 - a_{23}x_3)/a_{22} \\ x_3 &= (b_3 - a_{31}x_1 - a_{32}x_2)/a_{33} \end{aligned}$$

① Jacobi iterative method – choose initial guess $\tilde{x}_0 \in \mathbb{R}^n$

→ in each step update all $x_i$ (using the equations above) at the same time

$$\tilde{x}_{n+1} \leftarrow F(\tilde{x}_n)$$

② Gauss-Seidel iterative method

→ in each step update only one $x_i$ at a time ← converges faster but no vectorization

$$x_k \leftarrow f(x_1, \dots x_{k-1}, x_{k+1}, \dots x_n)$$

Fact: A sufficient but not necessary condition for convergence of both of these is when $A$ is diagonally dominant, meaning

$$\forall i: |a_{ii}| > \sum_{j \neq i} |a_{ij}|$$

- Tridiagonal Systems of Equations

$$\begin{bmatrix} A_{11} & A_{12} & & & & \\ A_{21} & A_{22} & A_{23} & & & \\ & A_{32} & A_{33} & A_{34} & & \\ & & & \ddots & & \\ & & & A_{n-1,n-2} & A_{n-1,n-1} & A_{n-1,n} \\ & & & & A_{n,n-1} & A_{n,n} \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ \vdots \\ X_n \end{bmatrix} = \begin{bmatrix} B_1 \\ B_2 \\ \vdots \\ \vdots \\ B_n \end{bmatrix}$$

→ solving this with Gauss would be very memory and time inefficient

$$\underset{O(n^2)}{\text{memory}} \qquad \underset{O(n^3)}{\text{time}}$$

- The Thomas Algorithm
  - → instead store diagonals as vectors
    - $v_u = (A_{12}, A_{23}, A_{34}, \dots A_{n-1,n}) \in \mathbb{R}^{n-1}$
    - $v_d = (A_{11}, A_{22}, A_{33}, \dots A_{n,n}) \in \mathbb{R}^n$
    - $v_l = (A_{21}, A_{32}, A_{43}, \dots A_{n,n-1}) \in \mathbb{R}^{n-1}$
  - → and essentially perform Gauss
    1. eliminate the lower diagonal elements using the main diagonal
    2. last row will give us $X_n$
    3. then allways main + upper diagonal ~ $X_{n-1}$ and $X_n$
       ⟹ back substitution is very straightforward

- Error of Iterative Matrix Algorithms

Consider $Ax = b$

- True error $= \tilde{X}_T - \tilde{X}_N =: \tilde{e}$
- Residual error $= A(\tilde{X}_T - \tilde{X}_N) = A\tilde{X}_T - A\tilde{X}_N = b - A\tilde{X}_N =: \tilde{r}$

True Error $= A^{-1} \cdot$ Residual Error ... $\tilde{e} = A^{-1}\tilde{r}$

→ we want $\tilde{r} \to \tilde{0}$

→ norms will allow us to covert $\tilde{r}$ into a number

**Def:** A norm on a vector space $V$ over $\mathbb{R}$ is $\|\cdot\|: V \to \mathbb{R}_0^+$ which satisfies

1) $\forall u \in V: \quad \|u\| \geq 0 \quad \& \quad \|u\| = 0 \iff u = 0$
2) $\|\alpha u\| = |\alpha| \cdot \|u\|$
3) $\|u + v\| \leq \|u\| + \|v\|$

• Vector norms

a) maxim $\sim \ell^\infty$ $\qquad \|u\|_\infty = \max_i |u_i|$

b) manhattan $\sim \ell^1$ $\qquad \|u\|_1 = \sum_i |u_i|$

c) euclidean $\sim \ell^2$ $\qquad \|u\|_2 = \left(\sum_i u_i^2\right)^{\frac{1}{2}}$

• Matrix norms

a) max row $\qquad \|A\|_\infty = \max_i \left\{ \sum_j |a_{ij}| \right\}$

b) max col $\qquad \|A\|_1 = \max_j \left\{ \sum_i |a_{ij}| \right\}$

c) euclidean $\qquad \|A\|_E = \left( \sum_{i,j} a_{ij}^2 \right)^{\frac{1}{2}}$

Fact: It can be shown that

$$\frac{1}{\|A\| \cdot \|A^{-1}\|} \cdot \frac{\|r\|}{\|b\|} \leq \frac{\|e\|}{\|x_T\|} \leq \|A^{-1}\| \cdot \|A\| \cdot \frac{\|r\|}{\|b\|}$$

$\Rightarrow$ this gives us an upper an lower bound on $\dfrac{\|x_T - x_u\|}{\|x_T\|}$ = error relative to $x_T$

• Stability of a System of Equations

Def: The conditional number of a matrix $A$ is $\operatorname{cond}(A) := \|A\| \cdot \|A^{-1}\|$

if $\operatorname{cond}(A)$ is large, then the bounds on $\frac{\|e\|}{\|x_T\|}$ arent very helpful

• large $\operatorname{cond}(A) \Rightarrow A$ is ill-conditioned $\rightarrow$ greatly larger than 1

$\Rightarrow$ small change in $A$ will greatly affect the solution

• small $\operatorname{cond}(A) \Rightarrow A$ is well-conditioned $\rightarrow$ smaller than 1

$\Rightarrow$ small change in $A$ will not affect the solution much

Ex:

$$\begin{bmatrix} 6 & -2 \\ 11.5 & -3.85 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 10 \\ 17 \end{bmatrix} \Rightarrow \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 45 \\ 130 \end{bmatrix} \quad \& \quad \begin{bmatrix} 6 & -2 \\ 11.5 & -3.84 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 10 \\ 17 \end{bmatrix} \Rightarrow \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 110 \\ 325 \end{bmatrix}$$

$\Rightarrow A$ is ill-conditioned $\rightarrow \operatorname{cond}(A) > 1500$ regardless of what norm is used

- # Eigenvalues and Eigenvectors

**Def:** $\lambda \in \mathbb{R}$ is an eigenvalue of $A \in \mathbb{R}^{n \times n}$ $\equiv \exists v \in \mathbb{R}^n: Av = \lambda v$

↳ $v$ is an eigenvector corresponding to $\lambda$

## ① The Power Method - finding the largest eigenvalue

$A \in \mathbb{R}^{n \times n}$ with $n$ eigenvalues $|\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq \dots \geq |\lambda_n|$

$n$ eigenvectors $v_1 \quad v_2 \quad v_3 \quad \quad v_n$

$$\Rightarrow \forall x \in \mathbb{R}^n: \quad x = \alpha_1 v_1 + \alpha_2 v_2 + \dots + \alpha_n v_n$$

$$x^1 := Ax = A \sum_i \alpha_i v_i = \sum_i \alpha_i A v_i = \sum_i \alpha_i \lambda_i v_i$$

$$x^2 := Ax^1 = \sum_i \alpha_i \lambda_i^2 v_i$$

$$\Rightarrow x^m = Ax^{m-1} = \sum_i \alpha_i \lambda_i^m v_i = \lambda_1^m \left[ \alpha_1 v_1 + \sum_{i=2}^{m} \alpha_i \left(\frac{\lambda_i}{\lambda_1}\right)^m v_i \right]$$

$$\left(\frac{\lambda_i}{\lambda_1}\right)^m \to 0 \text{ as } m \to \infty \Rightarrow A^m x \to \lambda_1^m \underbrace{(\alpha_1 v_1)}_{\text{also an eigenvector}} \text{ as } m \to \infty$$

**Conclusion:** We have the largest eigenvalue $\lambda$ and want eigenvector

1. pick $x \in \mathbb{R}^n$.
2. calculate $\lim_{m \to \infty} \frac{A^m x}{\lambda^m} = v$

↳ this converges to an eigenvector with asymptotic error constant $\left|\frac{\lambda_2}{\lambda_1}\right|$

## How to get the largest eigenvalue?

→ let $i$ be an index where $x_i^m \neq 0$

$$m \to \infty: \quad \frac{x_i^{m+1}}{x_i^m} \to \frac{\lambda^{m+1} \cdot v_i}{\lambda^m \cdot v_i} = \lambda \quad \Rightarrow \text{just divide two successive approximations}$$

**Algorithm:**

1. Choose $x \in \mathbb{R}^n$, $x \neq 0$
2. $x^{n+1} \leftarrow Ax^n$
3. normalize $x^{n+1} = \lambda \cdot v$
4. Our estimate now is
5. Go to 2. with $x^n = v$

→ pick a norm and fix $\|v\| = 1$

↳ split $x^{n+1}$ to $\lambda \cdot v$ so that this is true

→ easy for maximum norm: $\begin{bmatrix} 3 \\ 4 \\ 2 \end{bmatrix} = \underset{\lambda}{4} \cdot \underset{v}{\begin{bmatrix} 0.75 \\ 1 \\ 0.5 \end{bmatrix}}$

→ recall $\|v\| = 1 \Rightarrow \|Av\| = \|\lambda v\| = |\lambda| \cdot \|v\| = |\lambda|$

↳ again split to

## ② The Inverse Power Method – finding the smallest eigenvalue

$Ax = \lambda x \Rightarrow x = A^{-1}(\lambda x) \Rightarrow A^{-1} x = \frac{1}{\lambda} x$

↳ the eigenvalues of $A^{-1}$ are the reciprocals of the eigenvalues of $A$

⇒ find largest of $A^{-1}$ ⟺ find smallest of $A$

→ do powermethod with $A^{-1}$

↳ when multiplying $x^{n+1} \leftarrow A^{-1} x^n$

it is possible to solve $A x^{n+1} = x^n$ for $x^{n+1}$ instead

might be more efficient

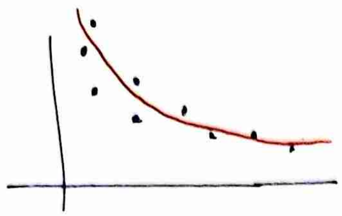
## ③ QR-Factorization

- an iterative algorithm used for calculating ALL of the eigenvalues

↳ uses the fact that the eigenvalues of an upper-Δ matrix are on its diagonal

• Curve Fitting and Interpolation

→ discrete experimental data



• curve fitting = minimizing difference error

• interpolation = curve which passes through all of the points

① Linear Regression

Data: $(x_1, y_1), (x_2, y_2), \dots (x_m, y_m)$

↳ want to find $f(x) = a_1 x + a_0$ to fit it best

$$\Rightarrow \text{minimize} \quad E = \sum_{i=1}^{m} (y_i - f(x_i))^2 = \sum_{i=1}^{m} (y_i - a_1 x_i - a_0)^2 = \sum_{i=1}^{m} (a_1 x_i + a_0 - y_i)^2$$

$$\frac{\partial E}{\partial a_0} = 2 \sum_{i=1}^{m} (a_1 x_i + a_0 - y_i) = 0 \Rightarrow a_1 \sum x_i + m \cdot a_0 - \sum y_i = 0$$

$$\frac{\partial E}{\partial a_1} = 2 \sum_{i=1}^{m} (a_1 x_i + a_0 - y_i) x_i = 0 \Rightarrow a_1 \sum x_i^2 + a_0 \sum x_i - \sum x_i y_i = 0$$

$\Rightarrow$ Denote $S_x = \sum x_i$, $S_y = \sum y_i$, $S_{xx} = \sum x_i^2$, $S_{xy} = \sum x_i y_i$

$$\begin{aligned} a_1 S_x + a_0 \cdot m &= S_y \\ a_1 S_{xx} + a_0 S_x &= S_{xy} \end{aligned} \Rightarrow \quad a_1 = \frac{m S_{xy} - S_x S_y}{m S_{xx} - S_x^2}, \quad a_0 = \frac{S_{xx} S_y - S_{xy} S_x}{m S_{xx} - S_x^2}$$

② Linearizing Nonlinear Data

a) we want to fit $y = b \cdot x^m$ to the data

$$y = b \cdot x^m \Rightarrow \ln(y) = \ln(b) + m \cdot \ln(x)$$

$\Rightarrow (x_1, \dots x_m) \rightsquigarrow (X_1, \dots X_m) \rightarrow X_i = \ln(x_i)$

$(y_1, \dots y_m) \rightsquigarrow (Y_1, \dots Y_m) \rightarrow Y_i = \ln(y_i)$

$\Big\} \; Y = m \cdot X + \ln(b)$

↳ fit X and Y using linear linear regression to get

$a_1 = m$ and $a_0 = \ln(b) \Rightarrow b = e^{a_0}$

b) $y = b \cdot e^{mx} \Rightarrow \ln(y) = \ln(b) + m \cdot x$

$X = x$, $Y = \ln(y) \Rightarrow Y = m \cdot X + \ln(b) \rightsquigarrow m = a_1$, $a_0 = \ln(b)$

c) $y = \frac{1}{mx + b} \Rightarrow \frac{1}{y} = mx + b \Rightarrow Y = \frac{1}{y}$, $X = x \Rightarrow Y = mX + b$

d) $y = \frac{mx}{x + b} \Rightarrow \frac{1}{y} = \frac{x + b}{mx} = \frac{b}{m} \cdot \frac{1}{x} + \frac{1}{m} \Rightarrow \begin{aligned} Y &= \tfrac{1}{y} \\ X &= \tfrac{1}{x} \end{aligned} \Rightarrow Y = \underbrace{\frac{b}{m}}_{a_1} \cdot X + \underbrace{\frac{1}{m}}_{a_0}$

Ex: $y = v \cdot e^{-\frac{x}{R \cdot C}}$ ; R is known $\rightarrow C = ?$

$m = -\frac{1}{RC} \Rightarrow y = v \cdot e^{m x} \Rightarrow \ln(y) = \ln(v) + m \cdot x$

$$\left.\begin{array}{l} Y = \ln(y) \\ X = x \end{array}\right\} Y = m \cdot X + \ln(v)$$ $\leadsto$ lin. reg gives value of $m$

$\Rightarrow C = -\frac{1}{Rm}$

## (3) Polynomial Regression

$f(x) = \sum_{k=0}^{m} a_k x^k$ $\rightarrow$ $n$ data points can be curve fit with polynomials up to order $n-1$

higher order ~ it will pass through more data points and the commulative square error will be lower

↳ but it may deviate significantly from the overall trend between points, making the approximation inaccurate

→ Quadratic regression

$$E = \sum (f(x_i) - y_i)^2 = \sum (a_2 x_i^2 + a_1 x_i + a_0 - y_i)^2$$

$$\frac{\partial E}{\partial a_0} = 2 \sum (a_2 x_i^2 + a_1 x_i + a_0 - y_i) = 0 \Rightarrow a_2 \sum x_i^2 + a_1 \sum x_i + n a_0 = \sum y_i$$

$$\frac{\partial E}{\partial a_1} = 2 \sum (a_2 x_i^2 + a_1 x_i + a_0 - y_i) \cdot x_i = 0 \Rightarrow a_2 \sum x_i^3 + a_1 \sum x_i^2 + a_0 \sum x_i = \sum y_i x_i$$

$$\frac{\partial E}{\partial a_2} = 2 \sum (a_2 x_i^2 + a_1 x_i + a_0 - y_i) x_i^2 = 0 \Rightarrow a_2 \sum x_i^4 + a_1 \sum x_i^3 + a_0 \sum x_i^2 = \sum y_i x_i^2$$

$\Rightarrow$ this system of equations can again be solved for $\begin{pmatrix} a_2 \\ a_1 \\ a_0 \end{pmatrix}$

## (4) Curve Fitting using multiple functions

$$F(x) = C_1 f_1(x) + \dots + C_m f_m(x) = \sum_{i=1}^{m} C_i f_i(x) \Rightarrow E = \sum (F(x_i) - y_i)^2$$

$$\frac{\partial E}{\partial C_k} = 2 \sum (F(x_i) - y_i) \cdot \frac{\partial}{\partial C_k} (F(x_i) - y_i) = 0$$ $\rightarrow \frac{\partial F}{\partial C_k}(x) = f_k(x)$

$$\sum (F(x_i) - y_i) \cdot f_k(x_i) = \sum_i \left[ \sum_j C_j f_j(x_i) - y_i \right] \cdot f_k(x_i) = 0$$

$$\Rightarrow \forall k: \sum_j C_j \cdot \left( \sum_i f_j(x_i) f_k(x_i) \right) = \sum_i y_i f_k(x_i)$$

↳ system of $m$ linear equations in $m$ unknowns $C_1, \dots C_m$

- Interpolation
  - gives exact value at the data points an estimated value in between

→ $n$ points ⇒ exactly one $(n-1)$-degree polynomial

① Lagrange interpolation

Data $(x_1, y_1), \dots (x_n, y_n)$

$$f(x) = \sum_{i=1}^{n} y_i L_i(x) \quad , \quad L_i(x) = \prod_{\substack{j=1 \\ j \neq i}}^{n} \frac{(x-x_j)}{(x_i - x_j)} \longrightarrow \text{this is a degree } (n-1) \text{ polynomial}$$

$$L_i(x_k) = \begin{cases} 0, & k \neq i \\ 1, & k = i \end{cases} \Rightarrow f(x_k) = y_k \checkmark$$

⇓ $f$ is a degree $(n-1)$ polynomial

② Newton interpolation

Def: Given datapoints $(x_1, y_1), \dots (x_n, y_n)$ where $x_k$ are pairwise distinct, the forward divided differences are defined as

$$[y_k] := y_k \quad , \quad [y_k, \dots y_{k+j}] := \frac{[y_{k+1}, \dots y_{k+j}] - [y_k, \dots y_{k+j-1}]}{x_{k+j} - x_k}$$

$$\begin{array}{lllll}
x_1 \to y_1 = [y_1] & & & & \\
 & [y_1, y_2] & & & \\
x_2 \to y_2 = [y_2] & & [y_1, y_2, y_3] & & \\
 & [y_2, y_3] & & [y_1, y_2, y_3, y_4] & \\
x_3 \to y_3 = [y_3] & & [y_2, y_3, y_4] & & \\
 & [y_3, y_4] & & & \\
x_4 \to y_4 = [y_4] & & & &
\end{array}$$

Def: The $(n-1)$-degree Newton polynomial for $(x_1, y_1), \dots (x_n, y_n)$ is defined as

$$f(x) := y_1 + [y_1, y_2](x - x_1) + [y_1, y_2, y_3](x-x_1)(x-x_2) + \dots + [y_1, \dots y_n](x-x_1) \cdots (x-x_{n-1})$$

$$= \sum_{k=1}^{n} [y_1, \dots y_k] \cdot \prod_{j=1}^{k-1} (x - x_j)$$

Fact: $\forall i : f(x_i) = y_i$

→ data points can be subsequently added and only one coefficient per extra data point needs to be calculated

→ also the data points don't need to be in any particular order

⇒ Newton polynomials are a good option

Ex: Calculate the Newton polynomial for the following data points

| $x_i$ | $y_i$ ($a_0$) | $a_1$ | $a_2$ | $a_3$ | $a_4$ |
|---|---|---|---|---|---|
| 1 | 52 | $\frac{5-52}{2-1} = -47$ | $\frac{-5+47}{4-1} = 14$ | $\frac{-10-14}{5-1} = -6$ | $\frac{6+6}{7-1} = 2$ |
| 2 | 5 | $\frac{-5-5}{4-2} = -5$ | $\frac{-35+5}{5-2} = -10$ | $\frac{20+10}{7-2} = 6$ | |
| 4 | -5 | $\frac{-40+5}{5-4} = -35$ | $\frac{25+35}{7-4} = 20$ | | |
| 5 | -40 | $\frac{10+40}{7-5} = 25$ | | | |
| 7 | 10 | | | | |

$$f(x) = a_0 + a_1(x-x_1) + a_2(x-x_1)(x-x_2) + a_3(x-x_1)(x-x_2)(x-x_3) + a_4(x-x_1)(x-x_2)(x-x_3)(x-x_4)$$
$$= 52 - 47(x-1) + 14(x-1)(x-2) - 6(x-1)(x-2)(x-4) + 2(x-1)(x-2)(x-4)(x-5)$$

$$\Rightarrow f(3) = 52 - 47\cdot 2 + 14\cdot 2 - 6(-2) + 2\cdot 4 = 52 - 94 + 28 + 12 + 8 = 100 - 94 = 6$$

## 3) Spline / Piecewise Interpolation → Data points $(x_0, y_0), \dots (x_n, y_n)$

→ connect each two consecutive points by a lower degree polynomial

### a) Linear splines

→ just makes line segments between points

line $(x_i, y_i) — (x_{i+1}, y_{i+1}) \Rightarrow f_i(x) = \frac{(x - x_{i+1})}{(x_i - x_{i+1})} y_i + \frac{(x - x_i)}{(x_{i+1} - x_i)} y_{i+1}$

### b) Quadratic splines

function between $(x_i, y_i) — (x_{i+1}, y_{i+1})$ is $f_i(x) = a_i x^2 + b_i x + c_i$

→ $n+1$ data points $\Rightarrow$ $n$ intervals & 3 coefficients in $f_i$

$\Rightarrow$ $3n$ coefficients in total

Conditions:

1) $f_i(x_i) = y_i$, $f_i(x_{i+1}) = y_{i+1}$ $\Rightarrow$ $2n$ equations

2) $f_i'(x_{i+1}) = f_{i+1}'(x_{i+1})$ $\Rightarrow$ $n-1$ equations

→ we need one more equation

3) $f_0''(x_0) = 0 \iff 2a_1 = 0 \Rightarrow a_1 = 0$

$\Rightarrow$ in total $3n$ linear equations in $3n$ unknowns - coefficients

## (c) Cubic / Natural splines

$f_i(x) = a_i x^3 + b_i x^2 + c_i x + d_i$

→ $n$ functions & 4 coefficients ⇒ $4n$ unknowns

Conditions:

1, $f_i(x_i) = y_i$ , $f_i(x_{i+1}) = y_{i+1}$ ... $2n$ equations
2, $f_i'(x_{i+1}) = f_{i+1}'(x_{i+1})$ ... $n-1$ equations
3, $f_i''(x_{i+1}) = f_{i+1}''(x_{i+1})$ ... $n-1$ equations
4, $f_0''(x_0) = 0$ & $f_{n-1}''(x_n) = 0$ ... 2 equations

} $4n$ equations and $4n$ unknowns

## (d) Natural splines using Lagrange polynomials

→ we will do some smart shit and end up with only $n$ equations and $n$ unknowns

$f_i''(x) = 6a_i x + 2b_i$ = linear function.

$$\Rightarrow f_i''(x) = \frac{x - x_{i+1}}{x_i - x_{i+1}} f''(x_i) + \frac{x - x_i}{x_{i+1} - x_i} f''(x_{i+1}) \quad \rightarrow f''(x_i),\ f''(x_{i+1}) \text{ unknowns}$$

$$\Rightarrow f_i'(x) = \int f_i''\,dx = \frac{f''(x_i)}{x_i - x_{i+1}} \cdot \frac{(x - x_{i+1})^2}{2} + \frac{f''(x_{i+1})}{x_{i+1} - x_i} \cdot \frac{(x - x_i)^2}{2} + C$$

$$f_i(x) = \int f_i'\,dx = \frac{f''(x_i)}{6(x_i - x_{i+1})} \cdot (x - x_{i+1})^3 + \frac{f''(x_{i+1})}{6(x_{i+1} - x_i)} \cdot (x - x_i)^3 + C \cdot x + D$$

- $f_i(x_i) = y_i = \frac{f''(x_i)}{6} \cdot (x_i - x_{i+1})^2 + Cx_i + D$
- $f_i(x_{i+1}) = y_{i+1} = \frac{f''(x_{i+1})}{6} \cdot (x_{i+1} - x_i)^2 + Cx_{i+1} + D$

⇒ denote $h_i := x_{i+1} - x_i$
$a_i := \frac{1}{6} f''(x_i)$

$$\Rightarrow \left.\begin{aligned} Cx_i + D &= y_i - a_i h_i^2 \\ Cx_{i+1} + D &= y_{i+1} - a_{i+1} h_i^2 \end{aligned}\right\} C\overbrace{(x_{i+1} - x_i)}^{h_i} = y_{i+1} - y_i - h_i^2(a_{i+1} - a_i)$$

$$\Rightarrow C = \frac{y_{i+1} - y_i}{h_i} - h_i(a_{i+1} - a_i) = \left(\frac{y_{i+1}}{h_i} - h_i a_{i+1}\right) - \left(\frac{y_i}{h_i} - h_i a_i\right)$$

$$\Rightarrow D = \underbrace{y_i - a_i h_i^2}_{h_i\left(\frac{y_i}{h_i} - a_i h_i\right)} - x_i\left(\frac{y_{i+1}}{h_i} - h_i a_{i+1}\right) + x_i\left(\frac{y_i}{h_i} - h_i a_i\right)$$

$x_{i+1}$

$$= \left(x_i + h_i\right)\left(\frac{y_i}{h_i} - a_i h_i\right) - x_i\left(\frac{y_{i+1}}{h_i} - h_i a_{i+1}\right)$$

$$\Rightarrow f_i(x) = \frac{a_i}{-h_i}(x - x_{i+1})^3 + \frac{a_{i+1}}{h_i}(x - x_i)^3 + Cx + D$$

$$= -\frac{a_i}{h_i}(x - x_{i+1})^3 + \frac{a_{i+1}}{h_i}(x - x_i)^3 + (x - x_i)\left(\frac{y_{i+1}}{h_i} - h_i a_{i+1}\right) - (x - x_{i+1})\left(\frac{y_i}{h_i} - a_i h_i\right)$$

$$\Rightarrow \underline{f_i(x) = \frac{a_{i+1}}{h_i}(x-x_i)^3 + \alpha_i(x-x_i) - \frac{a_i}{h_i}(x-x_{i+1})^3 - \beta_i(x-x_{i+1})}\quad , \; h_i = x_{i+1} - x_i$$

$\hookrightarrow$ where $\underline{\alpha_i = \frac{y_{i+1}}{h_i} - h_i a_{i+1}}$ ; $\underline{\beta_i = \frac{y_i}{h_i} - h_i a_i}$

$\rightarrow$ just need to find $a_i = \frac{1}{6} f''(x_i)$ for $i = 0, \dots n$

$\Rightarrow$ enforce $f_i'(x_{i+1}) = f_{i+1}'(x_{i+1})$

$$f_i'(x) = \frac{3a_{i+1}}{h_i}(x-x_i)^2 - \frac{3a_i}{h_i}(x-x_{i+1})^2 + \overbrace{\frac{y_{i+1}-y_i}{h_i} - h_i(a_{i+1}-a_i)}^{\alpha_i - \beta_i}$$

$\Rightarrow f_i'(x_{i+1}) = f_{i+1}'(x_{i+1}) \iff$

$$3a_{i+1}h_i + \frac{y_{i+1}-y_i}{h_i} - h_i a_{i+1} + h_i a_i = -3a_{i+1}h_{i+1} + \frac{y_{i+2}-y_{i+1}}{h_{i+1}} - h_{i+1}a_{i+2} + h_{i+1}a_{i+1}$$

$$\underline{h_i a_i + 2(h_i + h_{i+1})a_{i+1} + h_{i+1}a_{i+2} = \frac{y_{i+2}-y_{i+1}}{h_{i+1}} - \frac{y_{i+1}-y_i}{h_i}}$$

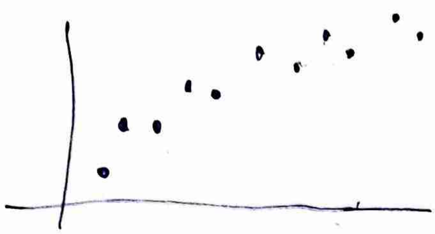
$\hookrightarrow$ $n-1$ equations with $n+1$ unknowns $a_0, \dots a_n$.

$\Rightarrow$ enforce $f_0''(x_0) = f_{n-1}''(x_n) = 0 \Rightarrow \underline{a_0 = a_n = 0}$

$\Rightarrow$ in total $n+1$ equations with $n+1$ unknowns

- Numerical Differentiation

We have experimental data $(x_1, y_1), \dots (x_n, y_n)$



1, fit a curve and differentiate it analytically
2, do some numerical shit

Def : The derivative of $f$ at $x=a$ is $f'(a) := \lim_{x \to a} \frac{f(x)-f(a)}{x-a}$

Problem : $f$ is sampled at points $x_1, \dots x_n$. Aproximate $f'(x_i)$

1, Forward difference formula : $f'(x_i) \approx \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i}$

2, Backward difference formula ; $f'(x_i) \approx \frac{f(x_i) - f(x_{i-1})}{x_i - x_{i-1}}$

3) Central difference formula : $f'(x_i) \approx \frac{f(x_{i+1}) - f(x_{i-1})}{x_{i+1} - x_{i-1}}$ ← The best

→ We will show using Taylor series that the error of the Central difference formula is an order of a magnitude smaller.

$$f(x) = f(x_i) + f'(x_i)\cdot(x-x_i) + \frac{f''(x_i)}{2}(x-x_i)^2 + \dots$$

⊛ $f(x_{i+1}) = f(x_i) + f'(x_i)\cdot h + O(h^2)$ assuming $h = x_{i+1} - x_i < 1$

$$\Rightarrow f'(x_i) = \frac{f(x_{i+1}) - f(x_i)}{h} + O(h) \quad \dots \text{2-point forward diff}$$

⊠ $f(x_{i-1}) = f(x_i) - f'(x_i)\cdot h + O(h^2)$ assuming $h = x_i - x_{i-1} < 1$

$$\Rightarrow f'(x_i) = \frac{f(x_i) - f(x_{i-1})}{h} + O(h) \quad \dots \text{2-point backward diff}$$

! Assuming the points are evenly spaced: $x_{i+1} - x_i = x_i - x_{i-1} = h$ :

→ expand ⊛ and ⊠ and subtract them:

$$f(x_{i+1}) = f(x_i) + f'(x_i)h + \frac{f''(x_i)}{2}h^2 + O(h^3)$$
$$f(x_{i-1}) = f(x_i) - f'(x_i)h + \frac{f''(x_i)}{2}h^2 - O(h^3)$$
⊖

$$\Rightarrow f(x_{i+1}) - f(x_{i-1}) = 2f'(x_i)h + O(h^3)$$

$$\Rightarrow f'(x_i) = \frac{f(x_{i+1}) - f(x_{i-1})}{2h} + O(h^2) \quad \dots \text{2-point central diff}$$

**Theorem:** Given $f$ evaluated at evenly spaced points $x_0, \dots x_n$, with $x_{i+1} - x_i = h$, one can approximate the derivatives like:

1) 2-point central difference formula: $f'(x_i) = \frac{f(x_{i+1}) - f(x_{i-1})}{2h} + O(h^2)$

↳ use at midpoints

2) 3-point forward difference formula: $f'(x_i) = \frac{-3f(x_i) + 4f(x_{i+1}) - f(x_{i+2})}{2h} + O(h^2)$

↳ use at $x_0$ endpoint

3) 3-point backword difference formula: $f'(x_i) = \frac{f(x_{i-2}) - 4f(x_{i-1}) + 3f(x_i)}{2h} + O(h^2)$

↳ use at $x_n$ endpoint

**Proof:** We have already shown 1). To show 2) do

$$(*)\quad \begin{aligned} f(x_{i+1}) &= f(x_i) + f'(x_i)h + \frac{f''(x_i)}{2}h^2 + O(h^3) \quad \cdot 4 \\ f(x_{i+2}) &= f(x_i) + f'(x_i)(2h) + \frac{f''(x_i)}{2}(2h)^2 + O(h^3) \end{aligned} \quad \ominus$$

$$\Rightarrow 4f(x_{i+1}) - f(x_{i+2}) = 3f(x_i) + 2f'(x_i)h + O(h^3)$$

$$\Rightarrow f'(x_i) = \frac{-3f(x_i) + 4f(x_{i+1}) - f(x_{i+2})}{2h} + O(h^2) \qquad \Rightarrow 3) \text{ is similar} \quad \blacksquare$$

**Theorem:** Similarly, we can derive 3-point difference formulas for the 2nd derivative

1) 3-point central diff: $f''(x_i) = \frac{f(x_{i-1}) - 2f(x_i) + f(x_{i+1})}{h^2} + O(h^2)$ ... midpoints

2) 3-point forward diff: $f''(x_i) = \frac{f(x_i) - 2f(x_{i+1}) + f(x_{i+2})}{h^2} + O(h)$ ... endpoint $x_0$

3) 3-point backward diff: $f''(x_i) = \frac{f(x_{i-2}) - 2f(x_{i-1}) + f(x_i)}{h^2} + O(h)$ ... endpoint $x_n$

**Proof:**

1) $$\begin{aligned} f(x_{i+1}) &= f(x_i) + f'(x_i)h + \frac{f''(x_i)}{2}h^2 + \frac{f'''(x_i)}{6}h^3 + O(h^4) \\ f(x_{i-1}) &= f(x_i) - f'(x_i)h + \frac{f''(x_i)}{2}h^2 - \frac{f'''(x_i)}{6}h^3 + O(h^4) \end{aligned} \quad \oplus$$

$$\Rightarrow f(x_{i+1}) + f(x_{i-1}) = 2f(x_i) + f''(x_i)h^2 + O(h^4) \quad \Rightarrow 1)$$

2) using the equations (*) to eliminate $f'(x_i)$:

$$f(x_{i+2}) - 2f(x_{i+1}) = -f(x_i) + f''(x_i)h^2 + O(h^3)$$

$$\Rightarrow f''(x_i) = \frac{f(x_i) - 2f(x_{i+1}) + f(x_{i+2})}{h^2} + O(h)$$

$\blacksquare$

3) similar

- Differentiation using Lagrange Polynomials

⊕ the points don't have to be evenly spaced

⊕ we can ask for the derivative anywhere

⊖ there is no estimate for the error

Theorem: Given 3 points $(x_1, y_1), (x_2, y_2), (x_3, y_3)$ construct the L. polynomial:

$$f(x) = \frac{(x-x_2)(x-x_3)}{(x_1-x_2)(x_1-x_3)} y_1 + \frac{(x-x_1)(x-x_3)}{(x_2-x_1)(x_2-x_3)} y_2 + \frac{(x-x_1)(x-x_2)}{(x_3-x_1)(x_3-x_2)} y_3$$

→ find the derivative of this polynomial and use it as an approximation

$$f'(x) = \frac{2x - x_2 - x_3}{(x_1-x_2)(x_1-x_3)} y_1 + \frac{2x - x_1 - x_3}{(x_2-x_1)(x_2-x_3)} y_2 + \frac{2x - x_1 - x_2}{(x_3-x_1)(x_3-x_2)} y_3$$

- Differentiation using Curve Fitting

→ if there is a lot of scatter in the data

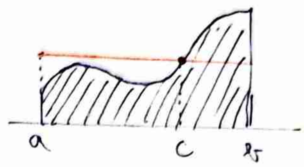
⇒ fit a curve and differentiate it analytically

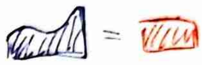
→ there is again no estimate for error

## • Numerical Integration

**Theorem (Integral Mean Value Theorem):** If $f$ is continuous on $[a,b]$ then

$$\exists c \in [a,b] : \int_a^b f(x)\,dx = (b-a)\cdot f(c)$$



### ① Rectangle method

→ divide $(a,b)$ into $n$ rectangles $(x_1,x_2) \dots (x_n, x_{n+1})$ of width $h$.

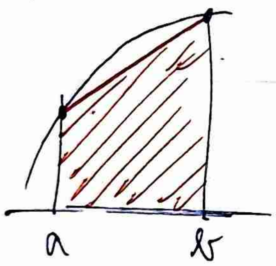
$$\int_a^b f(x)\,dx \approx \sum_{i=1}^{n} (x_{i+1}-x_i)\, f(x_i) = h \sum_{i=1} f(x_i)$$

### ② Midpoint method

$$\int_a^b f(x)\,dx \approx \sum_{i=1}^{n} (x_{i+1}-x_i)\, f\left(\frac{x_{i+1}-x_i}{2}\right) = h \sum_{i=1}^{n} f\left(\frac{x_{i+1}-x_i}{2}\right)$$

### ③ Trapezoid method

→ use a linear function to approximate the integrand




line using Newton polynomial

$$f(x) \approx y_1 + [y_1, y_2]\cdot(x-x_1) = f(a) + \frac{f(b)-f(a)}{b-a}(x-a)$$

$$\Rightarrow \int_a^b f(x)\,dx \approx \left[x f(a) + \frac{f(b)-f(a)}{b-a}\cdot\frac{(x-a)^2}{2}\right]_a^b$$

$$\Rightarrow \int_a^b f(x)\,dx \approx (b-a) f(a) + \frac{1}{2}\big(f(b)-f(a)\big)(b-a) = \frac{f(a)+f(b)}{2}(b-a)$$

$$\Rightarrow \int_a^b f(x)\,dx \approx \sum_{i=1}^{n} \frac{f(x_i)+f(x_{i+1})}{2}(x_{i+1}-x_i) = \frac{h}{2}\sum_{i=1}\big(f(x_i)+f(x_{i+1})\big)$$

$$\Rightarrow \int_a^b f(x)\,dx \approx \frac{h}{2}\big(f(a)+f(b)\big) + h\sum_{i=2}^{n} f(x_i)$$

## ④ Simpson's 1/3 method — use quadratics

→ approximate $\int_a^b f(x)\,dx$ with a quadratic passing through $x_1 = a$, $x_2 = (a+b)/2$, $x_3 = b$.

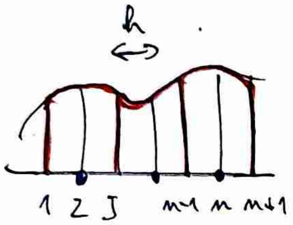
$$p(x) = \alpha + \beta(x-x_1) + \gamma(x-x_1)(x-x_2) \quad \dots \text{ Newton polynomial}$$

• $\alpha = f(x_1)$, $\beta = \frac{f(x_2)-f(x_1)}{x_2-x_1}$, $\gamma = \frac{f(x_3)-2f(x_2)+f(x_1)}{2h^2}$

→ $h = x_3 - x_2 = x_2 - x_1$

$$\Rightarrow \int_a^b f(x)\,dx \approx \int_{x_1}^{x_3} p(x)\,dx = \dots = \frac{h}{3}\left[f(x_1) + 4f(x_2) + f(x_3)\right]$$

→ now divide $[a,b]$ into $n = 2k$ equally spaced intervals $(x_1, x_2), \dots (x_n, x_{n+1})$




$$\int_a^b f(x)\,dx \approx \sum_{i=2}^{n} \frac{h}{3}\left[f(x_{i-1}) + 4f(x_i) + f(x_{i+1})\right]$$

↳ interval sidepoints are counted 2x
↳ $[a,b]$ endpoints only once.

$$\Rightarrow \int_a^b f(x)\,dx \approx \frac{h}{3}\left[f(a) + 4\cdot\sum_{i=2,4,6}^{n} f(x_i) + 2\cdot\sum_{j=3,5,7}^{n-1} f(x_j) + f(b)\right], \quad h = \frac{b-a}{n}$$

## ⑤ Simpson's 3/8 method — use cubics

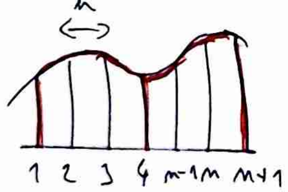
→ divide $[a,b]$ into 3 equal intervals ⇒ 4 points $x_1, x_2, x_3, x_4$

→ interpolate a cubic polynomial

$$p(x) = c_0 + c_1 x + c_2 x^2 + c_3 x^3$$

$$\Rightarrow \int_a^b f(x)\,dx \approx \int_a^b p(x)\,dx = \frac{3}{8}h\left[f(x_1) + 3f(x_2) + 3f(x_3) + f(x_4)\right]$$

→ in general divide $[a,b]$ into $n = 3k$ equally sized intervals




→ midpoints 2, 3, 5, 6, 8, 9, ... are counted 3x
⇒ sidepoints 4, 7, 10 ... are counted 2x

$$\Rightarrow \int_a^b f(x)\,dx \approx \frac{3h}{8}\left[f(a) + 3\sum_{i=2,5,8}^{n-1}\left(f(x_i) + f(x_{i+1})\right) + 2\cdot\sum_{i=4,7,10}^{n-2} f(x_i) + f(b)\right]$$

# ⑥ Gauss Quadrature

→ idea: approximate integral as a weighted sum

$$\int_{-1}^{1} f(x)\,dx \approx \sum_{i=1}^{n} c_i f(x_i)$$

→ $\tilde{c} \in \mathbb{R}^n$ and $\tilde{x} \in [-1,1]^n$ are $2n$ parameters
↳ weights ↳ Gauss points

→ we assume that $f$ can be reasonably well be approximated using polynomials

⇒ we want the relation to hold exactly for polynomials $1, x, x^2, \dots x^{2n-1}$

Ex: $n=2$:

$$\int_{-1}^{1} f(x)\,dx \approx c_1 f(x_1) + c_2 f(x_2)$$

- $f(x)=1 \Rightarrow \int_{-1}^{1} 1\,dx = 2 = c_1 + c_2$
- $f(x)=x \Rightarrow \int_{-1}^{1} x\,dx = 0 = c_1 x_1 + c_2 x_2$
- $f(x)=x^2 \Rightarrow \int_{-1}^{1} x^2\,dx = \frac{2}{3} = c_1 x_1^2 + c_2 x_2^2$
- $f(x)=x^3 \Rightarrow \int_{-1}^{1} x^3\,dx = 0 = c_1 x_1^3 + c_2 x_2^3$

4 equations (nonlinear)
4 unknowns
→ multiple solutions can exist

⇒ impose conditions that $x_1, x_2$ are symmetrically located about the origin

↳ $x_1 = -x_2$. ⇒ $c_1 = c_2$

⇒ solving gives: $c_1 = c_2 = 1$ & $x_1 = -\frac{1}{\sqrt{3}}$, $x_2 = \frac{1}{\sqrt{3}}$

$$\Rightarrow \underline{\underline{\int_{-1}^{1} f(x)\,dx \approx f\left(-\tfrac{1}{\sqrt{3}}\right) + f\left(\tfrac{1}{\sqrt{3}}\right)}}$$

← exact for $f(x) = 1, x, x^2, x^3$

Ex: $f(x) = \cos(x) \Rightarrow \int_{-1}^{1} \cos(x)\,dx = \sin(x)\Big|_{-1}^{1} \approx 1.6829$ → approximation $= 1.676$

⇒ more parameters = higher precision: $\cos(x)$ with 3 points $\approx 1.6828$

<u>In general</u>: Enforce that $x_1, x_2, \dots x_n$ are <u>symmetrical about the origin</u>

⇒ this ensures that $\underline{c_i = c_{n-i}}$

❗ What if the domain of integration is not $[-1,1]$?

$$\int_a^b f(x)\,dx = \int_{-1}^{1} g(t)\,dt$$

→ substitution $x = \frac{1}{2}\left[t(b-a) + a + b\right]$
$dx = \frac{1}{2}(b-a)\,dt$

- <u>What if the integrand is not well behaved?</u>

→ if there is a singularity $c \in [a,b]$, just split the integral

$$\int_a^b f(x)\,dx = \int_a^c f(x)\,dx + \int_c^b f(x)\,dx$$

→ there could also be a problem at an endpoint:

<u>Ex</u>: $\int_0^1 \frac{1}{\sqrt{x}}\,dx = 2$

↳ use any method which doesn't use the endpoint $x=0$

↳ midpoint method or Gauss quadrature

↳ or approximate something like $\int_{0.001}^{1} \frac{1}{\sqrt{x}}\,dx \approx 2$

- <u>What if the integral has unbound limits?</u>

$$\int_a^\infty f(x)\,dx$$

1) either it diverges ... rip

2) or it converges ⇒ $\lim_{x\to\infty} f(x) = 0$

⇒ approximate $\int_a^n f(x)\,dx$ for $n = 2^k$, $k \in \mathbb{N}$

until successive iterations yield only a small change in the result