Orthogonal Iteration

Search for p-dim subspace at the same time

When might this be useful? eg. low rank approximation

Input: A, Z_0 - orthogonal $n \times p$ matrix

Output: $\lambda_1 \dots \lambda_p$

$i=0$ $A_0 = Z_0^T A Z_0$ L_0 - lower triang. part of A_0

while $\frac{\|L_i\|_F}{\|A_i\|_F} > tol$

$Y_{i+1} = A Z_i$

$Q_{i+1} R_{i+1} = Y_{i+1}$

$Z_{i+1} = Q_{i+1}$

$A_{i+1} = Z_{i+1}^T A Z_{i+1}$, L_{i+1} : lower triang. part of A_{i+1}

diag of A_i are $\lambda_1 \dots \lambda_p$

Claim: Let $|\lambda_1| \geq \dots \geq |\lambda_n|$

$\Rightarrow \text{Span}(Z_i) = \text{Span}(v_1 \dots v_p)$

QR iteration - w/ a shift to speed up convergence

$i=0$ $A_0 = Z_0^T A Z_0$ L_0 - lower triangular part of A_0

while $\frac{\|L_i\|_F}{\|A_i\|_F} > tol$

$A_i = Q_i R_i$

$A_{i+1} = R_i Q_i$ L_{i+1}: lower triang. part of A_{i+1}

$i = i+1$

① A_{i+1} is orthogonally similar to A_i

② A_i is the same as $Z_i^T A Z_i$ w/ $Z_0 = I_0$

Key idea : equivalence

$$AQ_i = Q_{i+1} R_{i+1} \quad / Q_i^T$$

$$Q_i^T A_i Q_i = Q_i^T Q_{i+1} R_{i+1}$$

Define : $\hat{Q}_i = Q_i^T Q_{i+1}$

Claim $\hat{Q}_i$ is orthogonal:

$$(Q_i^T Q_{i+1})^T (Q_i^T Q_{i+1}) = Q_{i+1}^T \underbrace{Q_i Q_i^T}_{I_n} Q_{i+1}$$

$$= Q_{i+1}^T Q_{i+1} = I_n$$

Start at again

$$AQ_i = Q_{i+1} R_{i+1}$$

$/ \cdot Q_{i+1}^T \Rightarrow Q_{i+1}^T A Q_i = R_{i+1}$

$/ \cdot Q_i^T Q_{i+1} \Rightarrow \underbrace{Q_{i+1}^T A Q_{i+1}}_{A_{i+1}} = R_{i+1} Q_i^T Q_{i+1}$

$A_{i+1} = R_{i+1} \hat{Q}_i$

w/ one shift

for $i = 0, 1, \ldots$

- Choose $\sigma_i \in \mathbb{R}$ close to an eigenvalue of A
- $Q_i R_i = A_i - \sigma_i I_n$
- $A_{i+1} = R_i Q_i + \sigma_i I_n$
- $i = i+1$

A_{i+1} is orthogonally similar to A_i

So has the same eigenvalues

Pf: $A_{i+1} = R_{i+1} Q_{i+1} + \sigma_i I_n$

$= Q_{i+1}^T (A_i - \sigma_i I_n) Q_{i+1} + \sigma I_n$

$= Q_i^T A_i Q_i$

When to stop?

$\|A_i(n, 1:n-1)\| \leq (|A_i(n,n)| + |A_i(n-1, n-1)|) \cdot tol$

Speed of convergence

$$\frac{|\lambda_k - \sigma_i|}{\min_{j \neq k} |\lambda_j - \sigma_i|} \rightsquigarrow \min_j |\lambda_j - \sigma_i|$$

Cost: ① QR per step $O(n^3)$

② Since 1 eigenvalue per iteration: $O(n^4)$

Speed up: Observation: the form of the QR iteration stays the same

Def

H is Hessenberg if it is non-zero at most 1 below the diagonal

$$\begin{pmatrix} x & \cdots & \cdots & x \\ x & x & & \vdots \\ 0 & x & \ddots & \vdots \\ \vdots & \ddots & \ddots & x \\ 0 & \cdots & 0 & x & x \end{pmatrix}$$

We can transform A into H (Hessenberg) using Householder reflections

$$A_1 = \begin{pmatrix} 1 & 0 \\ 0 & H_1 \end{pmatrix}^T A \begin{pmatrix} 1 & 0 \\ 0 & H_1 \end{pmatrix}^T$$

$$A_2 = \begin{pmatrix} I_2 & 0 \\ 0 & H_2 \end{pmatrix}^T A_1 \begin{pmatrix} I_2 & 0 \\ 0 & H_2 \end{pmatrix}$$

... continue to $n-2$

$$A = \begin{pmatrix} x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \end{pmatrix} \to A_1 = \begin{pmatrix} x & x & x & x & x \\ x & x & x & x & x \\ 0 & x & x & x & x \\ 0 & x & x & x & x \\ 0 & x & x & x & x \end{pmatrix}$$

$$\to A_2 = \begin{pmatrix} x & x & x & x & x \\ x & x & x & x & x \\ 0 & x & x & x & x \\ 0 & 0 & x & x & x \\ 0 & 0 & x & x & x \end{pmatrix} \to A_3 = \begin{pmatrix} x & x & x & x & x \\ x & x & x & x & x \\ 0 & x & x & x & x \\ 0 & 0 & x & x & x \\ 0 & 0 & 0 & x & x \end{pmatrix}.$$

Cost: $\frac{10}{3}n^3 + O(n^2)$ $\frac{10}{3}n^3 + O(n^2)$

Claim: all A_i have the same eigenvalues

① during the QR iteration

② A_i keeps the Hessenberg form

ie if A_i is Hess. so is A_{i+1}

Computing QR form for Hess. matrices

Givens rotation

Idea: Find a rotation in $\mathbb{R}^2$

$$v = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \Rightarrow \begin{pmatrix} \sqrt{x_1^2 + x_2^2} \\ 0 \end{pmatrix}$$

Claim: $R = \begin{pmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{pmatrix}$

$$\cos\varphi = \frac{x_1^2}{\sqrt{x_1^2 + x_2^2}} \qquad \sin\varphi = -\frac{x_2}{\sqrt{x_1^2 + x_2^2}}$$

Fast QR decomp for Hess. matrices

① Find Givens rotation $\tilde{R}_{21}$

$$A(1{:}2, 1) \to \begin{pmatrix} \alpha \\ 0 \end{pmatrix}$$

$$A_1 = \begin{pmatrix} \tilde{R}_{21} & 0 \\ 0 & I_{n-2} \end{pmatrix} \cdot A = R_{21} A$$

② Find Givens rotation $\tilde{R}_{32}$

$$A_1(2{:}3, 2) \to \begin{pmatrix} \alpha_2 \\ 0 \end{pmatrix}$$

$$A_2 = \begin{pmatrix} I & 0 & 0 \\ 0 & \tilde{R}_{32} & 0 \\ 0 & 0 & I_{n-3} \end{pmatrix} A_1 = R_{32} A_1$$

..

$$A = \begin{pmatrix} x & x & x & x & x \\ x & x & x & x & x \\ 0 & x & x & x & x \\ 0 & 0 & x & x & x \\ 0 & 0 & 0 & x & x \end{pmatrix} \underbrace{\to}_{R_{21}} A_1 = \begin{pmatrix} x & x & x & x & x \\ 0 & x & x & x & x \\ 0 & x & x & x & x \\ 0 & 0 & x & x & x \\ 0 & 0 & 0 & x & x \end{pmatrix}$$

$$\underbrace{\to}_{R_{23}} A_2 = \begin{pmatrix} x & x & x & x & x \\ 0 & x & x & x & x \\ 0 & 0 & x & x & x \\ 0 & 0 & x & x & x \\ 0 & 0 & 0 & x & x \end{pmatrix} \underbrace{\to}_{R_{43}} A_3 = \begin{pmatrix} x & x & x & x & x \\ 0 & x & x & x & x \\ 0 & 0 & x & x & x \\ 0 & 0 & 0 & x & x \\ 0 & 0 & 0 & x & x \end{pmatrix}$$

$$\underbrace{\to}_{R_{54}} A_4 = \begin{pmatrix} x & x & x & x & x \\ 0 & x & x & x & x \\ 0 & 0 & x & x & x \\ 0 & 0 & 0 & x & x \\ 0 & 0 & 0 & 0 & x \end{pmatrix}.$$

Cost $O(n^2)$ R_{ij} : cost $O(n)$

QR decomp. cost goes from $O(n^4)$ to $O(n^3)$

Summary

Put into a good form first (Hessenberg)

Later iterations are cheap

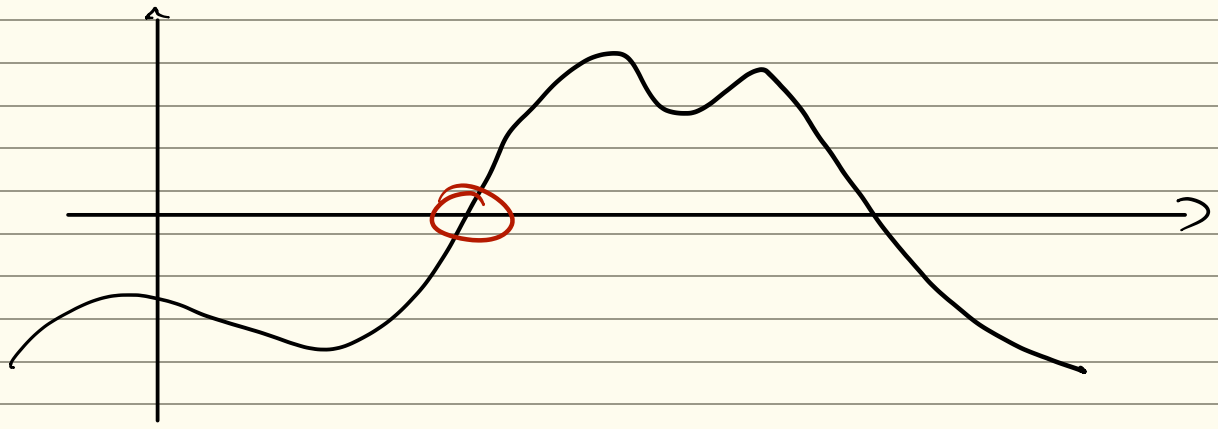
Solving non-linear equations & optimization

① $f(x) = 0$

② $f_i(x_1, \dots x_n) = 0$

③ $\min \{ f(x) : x \in K \subset \mathbb{R}^n \}$

Prototype problem: finding zero/roots



MUCH HARDER than systems of linear eq.

eg. roots of polynomials of degree 5 & up have no analytical formula

roots of systems of polynomials ⇒ algorithm is doubly exponential (2^{2^n})

⇒ so we use iterative methods

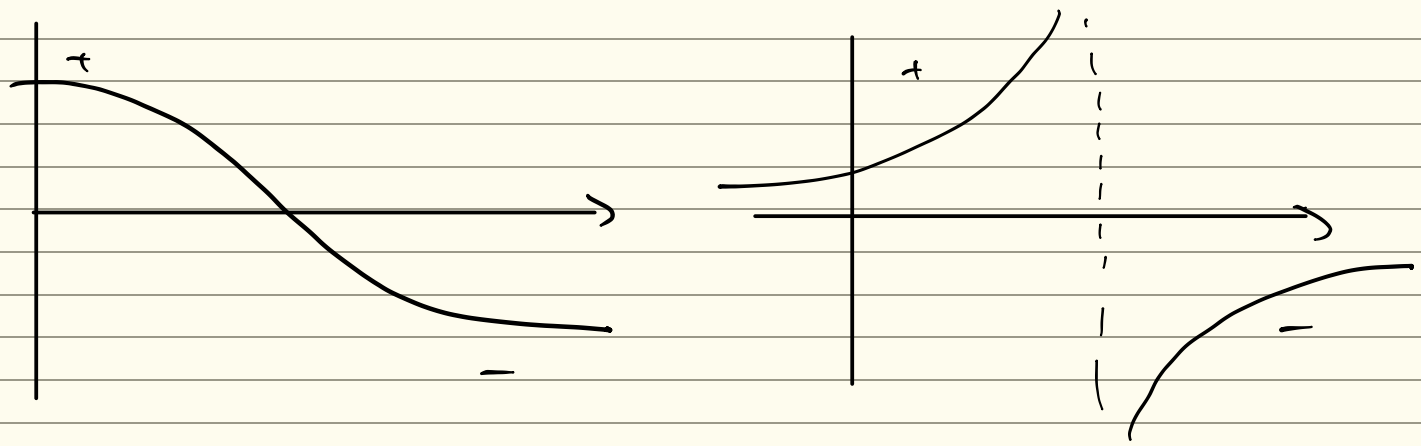
Basic strategy

① Set basic hypothesis - what is a zero

② At least 1 zero must exist in $[a,b]$ if $f(a)$ & $f(b)$ have different signs (in 1 dimension)

& if f is continuous in (a,b)

=> if f has a singularity in (a,b) no zero needs to exist

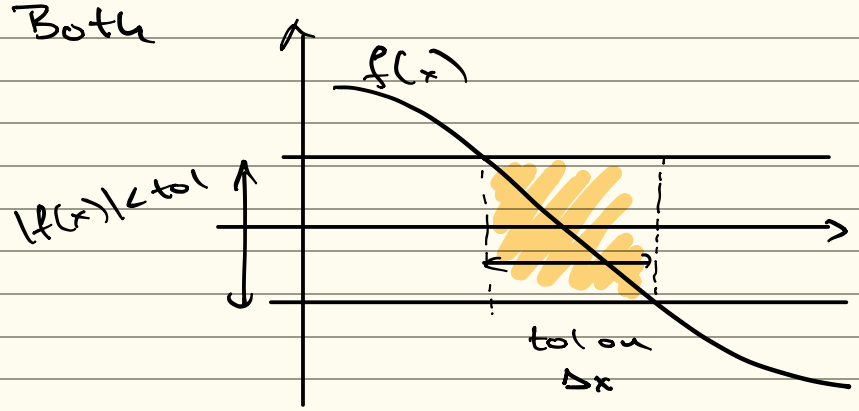


Convergence criteria for x

① $|x_k - x_{k-1}| < tol$

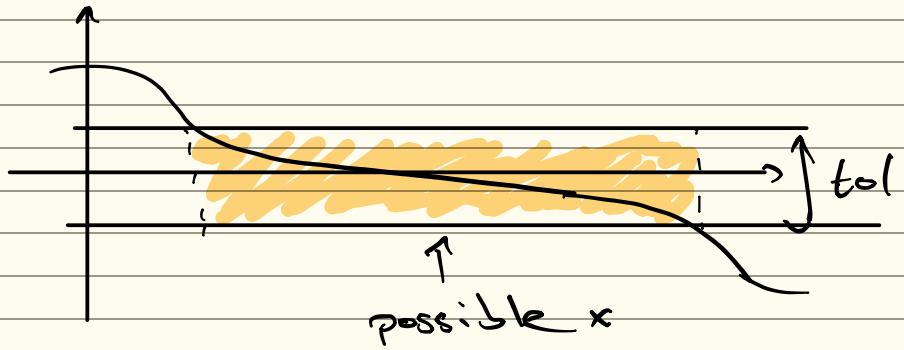
② $|f(x_k)| < tol$

③ Both

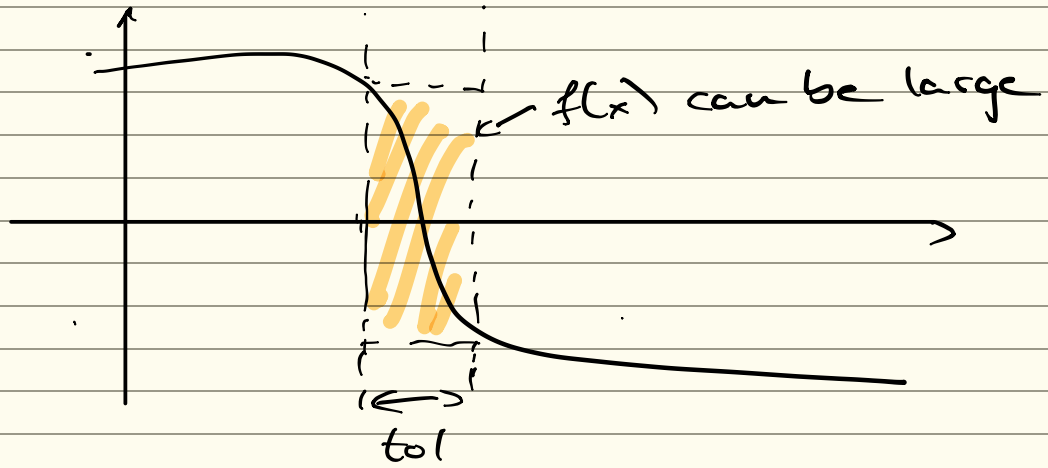


Comparison

if $|f'(x)|$ is small around zero $\Rightarrow$ $|f(x)| < tol$ easier to satisfy



if $|f'(x)|$ is large $\Rightarrow$ $f(x)$ can be large even if Δx is small



Connection between criteria

if x_a & x_b converge to x^*

$$\frac{f(x_b) - f(x_a)}{x_b - x_a} \to f'(x^*)$$

$$\Rightarrow |f(x_b) - f(x_a)| \approx |f'(x^*)| \cdot |x_b - x_a|$$

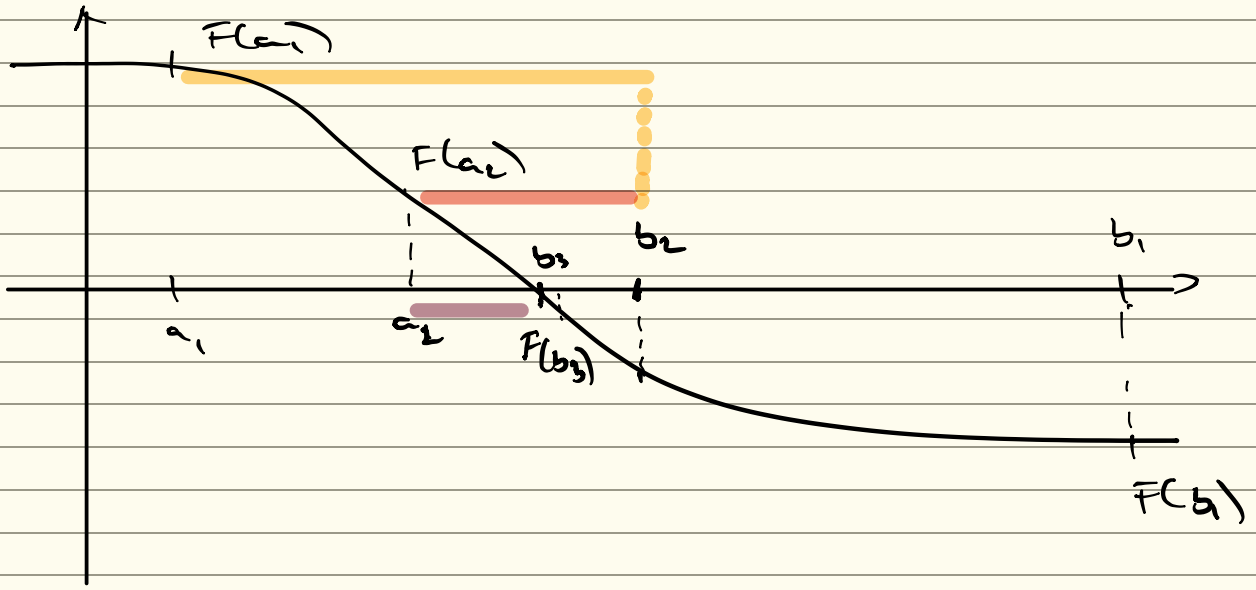
So $|f'(x^*)|$ determines the relationship (which we saw graphically)

Method 1 : Bisection

Define halfway point : $x_m = \frac{1}{2}(a+b)$

Approach

- find halfway point
- choose interval where endpoints have different signs
- repeat until interval is short enough



Alg

```
Vhod: funkcija f, krajisci a, b, toleranca tol
Izhod: priblizek x* za f(x) = 0 z |x − x*| < tol

for k = 1, 2, ...
   x_m = a + (b − a)/2
   if sign(f(x_m)) = sign(f(a))
      a = x_m
   else
      b = x_m
   end
   if |b-a|<tol, stop
end
```

Speed of convergence

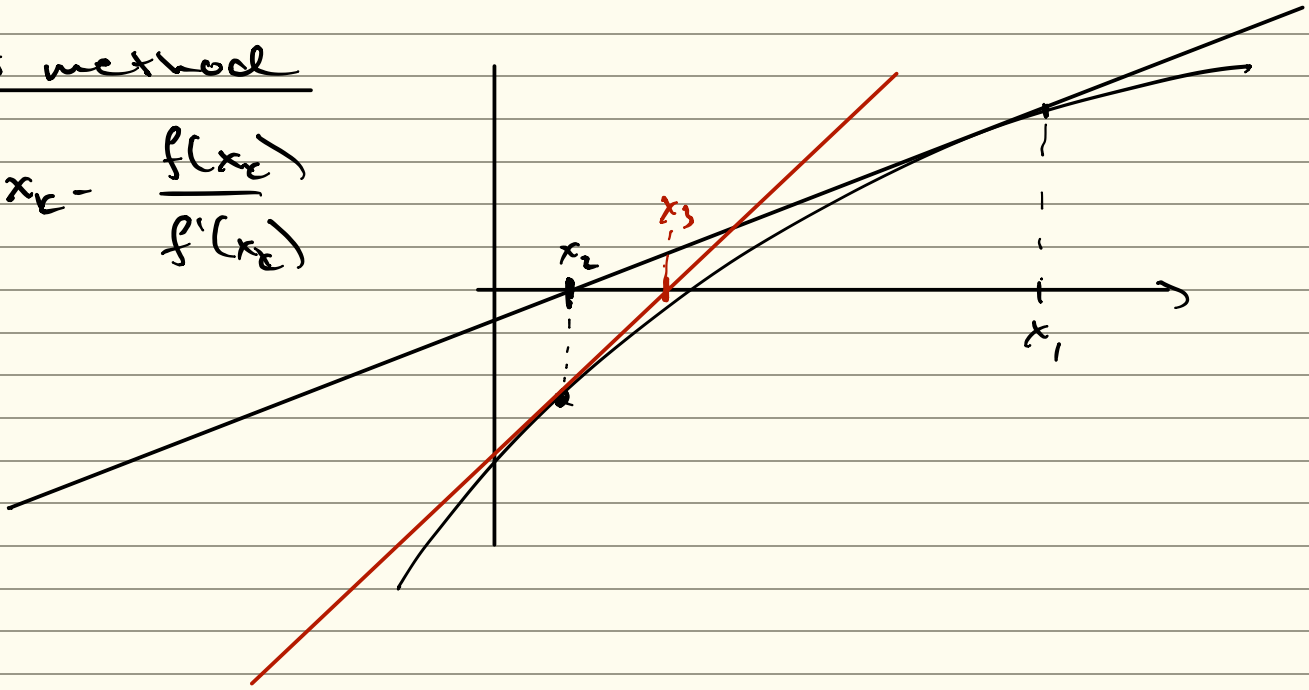
$\delta_0 = b - a \quad \delta_1 = \frac{1}{2}\delta_0 \quad \dots \quad \delta_n = \left(\frac{1}{2}\right)^n \delta_0$

$\frac{\delta_n}{\delta_0} = \left(\frac{1}{2}\right)^n = 2^{-n} \qquad n = \log_2\left(\frac{\delta_n}{\delta_0}\right)$

n	$\frac{\delta_n}{\delta_0}$	število izračunov funkcijskih vrednosti
5	3.1×10^{-2}	7
10	9.8×10^{-4}	12
20	9.5×10^{-7}	22
30	9.3×10^{-10}	32
40	9.1×10^{-13}	42
50	8.9×10^{-16}	52

Tangent method

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



Find the approximate zero using the derivative $f'(x_k)$

$y = f(x_k) - (x - x_k) f'(x_k)$

$\Rightarrow y = 0 \qquad 0 = f(x_k) - (x_{k+1} - x_k) f'(x_k)$

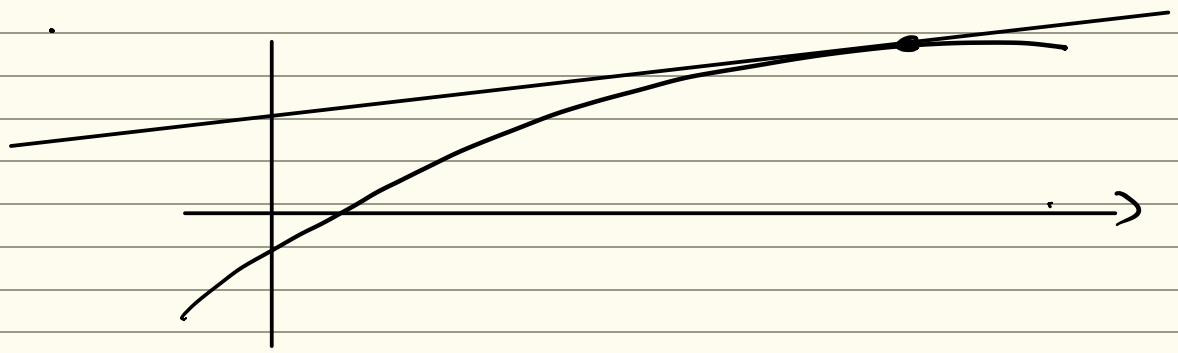
Alg:

- iterate until tol on x_k

Properties: ① Faster convergence than bisection
(order 2) ⇒ # of exact decimals doubles each step

② Requires $f'(x)$ (analytically)
(otherwise use next: secant method)

③ Need not converge if approximation "escapes"



Speed of convergence

We say convergence is of order r if there exists a $C>0$ such that

$$\lim_{k\to\infty} \frac{|e_{k+1}|}{|e_k|^r} = C$$

Thm: if $|f'(x^*)| \neq 0$, then tangent method has a speed of convergence of order 2

Pf Define $e_k = x_k - x^*$

Expand $f(x)$ around x^* (Taylor series)

$$f(x_k) = \underbrace{f(x^*)}_{0} + f'(x^*)(x_k - x^*) + \frac{1}{2} f''(x^*)(x_k - x^*)^2 + \dots$$

$$f(x_k) = f'(x^*)\, e_k + \frac{1}{2} f''(x^*) e_k^2 + o(e^2)$$

Expand $f'(x_k)$ around x^*

$$f'(x_k) = f'(x^*) + f''(x^*) e_k$$

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

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

$$e_{k+1} = e_k - \frac{f'(x^*)\, e_k + \frac{1}{2} f''(x^*) e_k^2}{f'(x^*) + f''(x^*) e_k}$$

take quotient

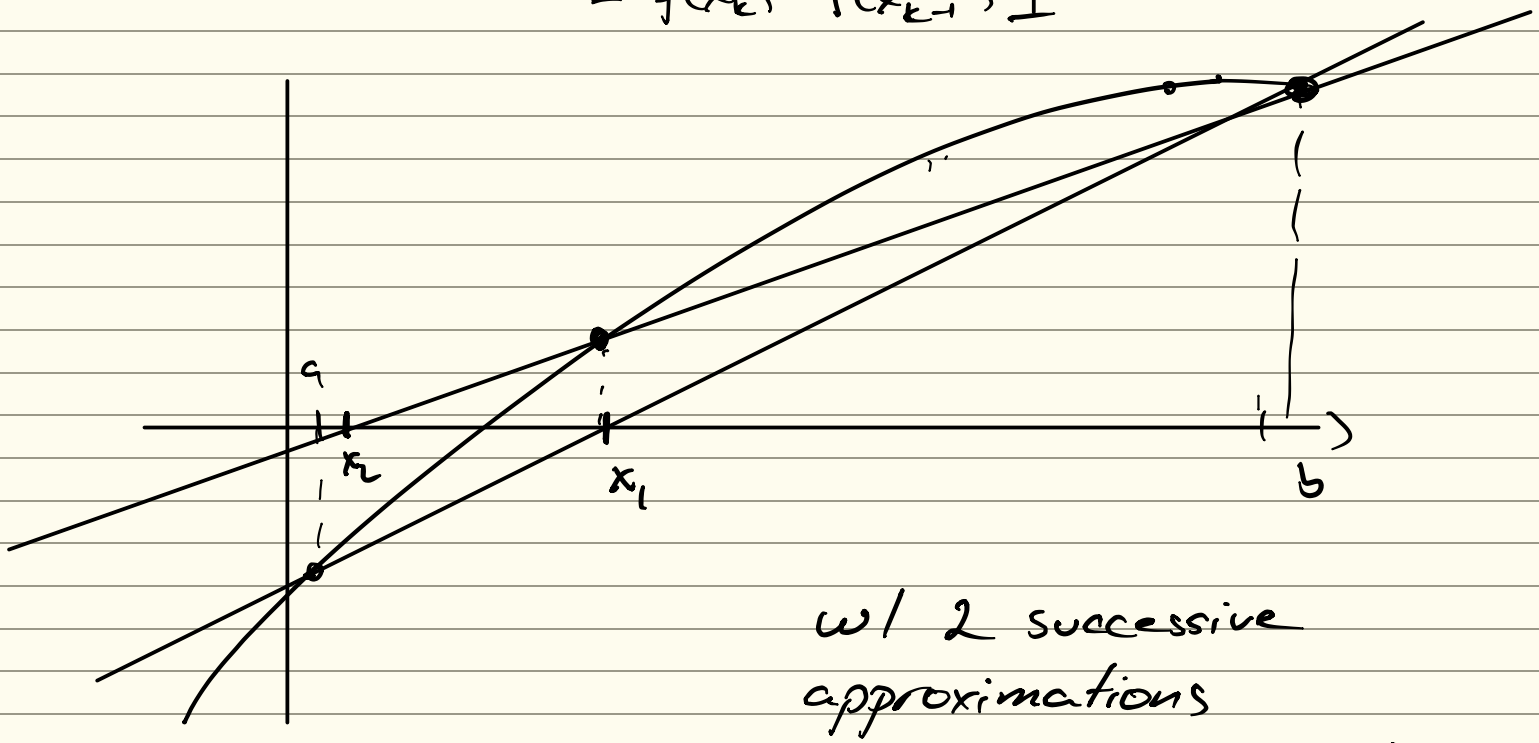
$$e_{k+1} = e_k - e_k \left(1 - \frac{f''(x^*)}{2 f'(x^*)} e_k\right)$$

$$e_{k+1} = e_k^2 \frac{f''(x^*)}{2 f'(x^*)}$$

so $\left|\dfrac{e_{k+1}}{e_k^2}\right| \to \dfrac{f''(x^*)}{2 f'(x^*)}$

Secant method

$$x_{k+1} = x_k - f(x_2)\left[\frac{x_k - x_{k-1}}{f(x_k) - f(x_{k-1})}\right]$$



w/ 2 successive approximations we take x-coord of the secant through $(x_k, f(x_k))$, $(x_{k-1}, f(x_{k-1}))$

x_k : curr est.

x_{k-1} : previous est.

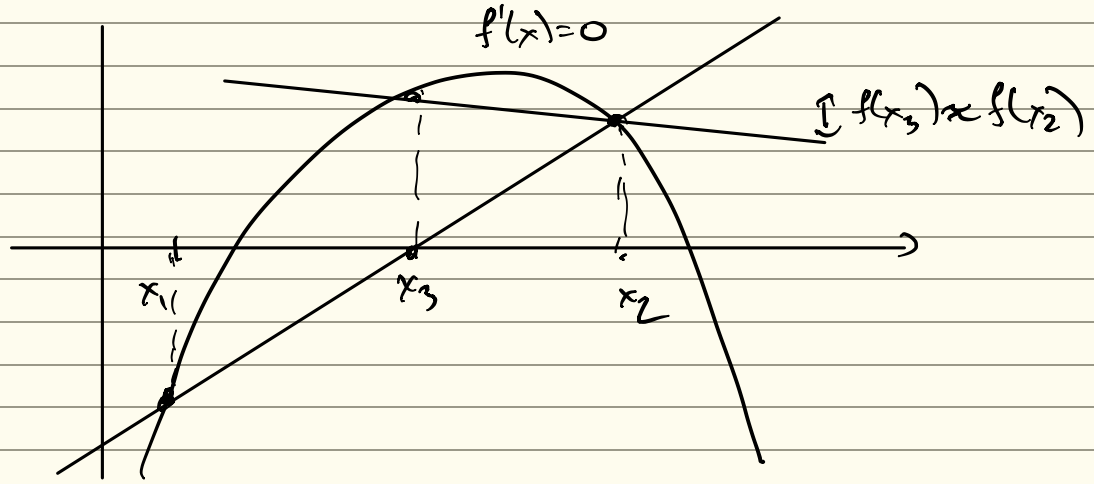
$$f'(x_k) \approx \frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}}$$

Alg

```
1  zacetni podatki:  f, x_1, x_2
2  for k = 2,3...
3    x_{k+1} = x_k − f(x_k)(x_k − x_{k−1})/(f(x_k) − f(x_{k−1}))
4    if je izpolnjen tolerancni pogoj, stop
5  end
```

Properties

- order of convergence ≈ 1.62
- do not need $f'(x)$
- approx. can leave starting interval



Next estimate for x_k if $f(x_k) \approx f(x_{k-1})$

Method of false position (Regula Falsi)

Hybrid of secant & biscection

```
zacetni podatki:  a, b
for k = 2, 3...
  c = b − f(b)(b − a)/(f(b) − f(a))
  if f(a)f(c) < 0
      b = c
  else
      a = c
  if je izpolnjen tolerancni pogoj, stop
end
```

Properties: ① slower convergence than secant
② stays within starting interval