ETH Lecture 401-0663-00L Numerical Methods for CSE

Numerical Methods for Computational Science and Engineering

Prof. R. Hiptmair, SAM, ETH Zurich

(with contributions from Prof. P. Arbenz and Dr. V. Gradinaru)

Autumn Term 2016

URL: http://www.sam.math.ethz.ch/~hiptmair/tmp/NumCSE/NumCSE16.pdf

VIII. Iterative Methods for non-linear Systems of Equations

$F: D \subset \mathbb{R}^n \longrightarrow \mathbb{R}^n$: $F(\underline{x}) = \underline{0}$

↳ vector of unknowns

same number of unknowns and equation

! No general theory

2.1. Iterative Methods

An iterative method for (approximately) solving the non-linear equation $F(\mathbf{x}) = 0$ is an algorithm generating an arbitrarily long sequence $\left(\mathbf{x}^{(k)}\right)_k$ of approximate solutions.

$\mathbf{x}^{(k)} \hat{=}$ k-th iterate

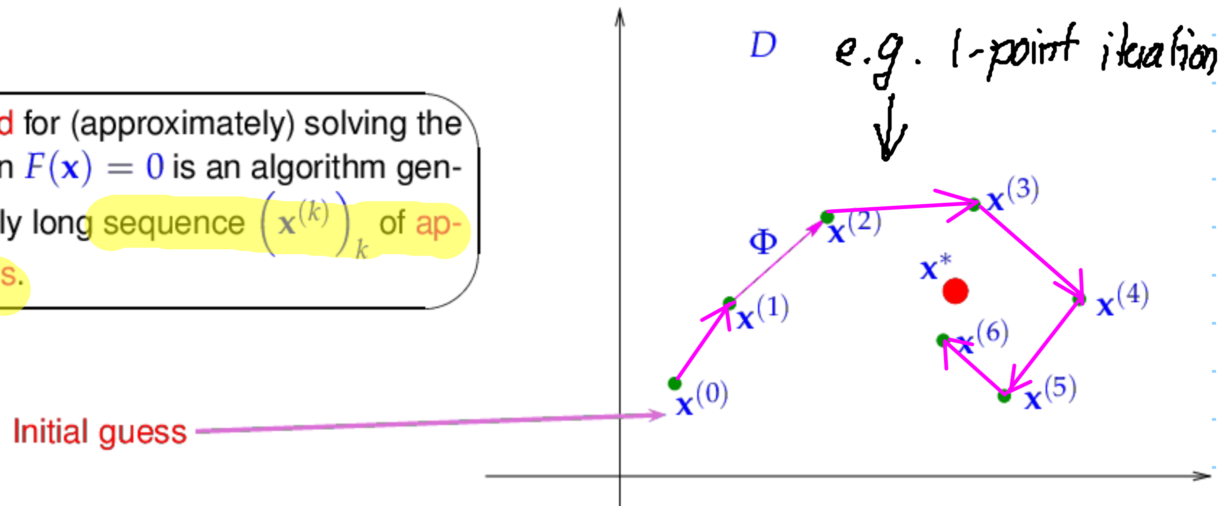


Fig. 282

Iteration error : $e^{(k)} := x^{(k)} - x^*$, $x^* \hat{=}$ solution

More concrete :

m-point iteration : $\underline{x}^{(k)} = \Phi_F(\underline{x}^{(k-1)}, \dots, x^{(k-m)})$

↳ iteration function

Needed initial guesses $x^{(0)}, \dots, \underline{x}^{(m-1)}$

Issues : • Convergence : $\underline{x}^{(k)} \longrightarrow \underline{x}^*$

• Consistency : $F(x^*) = 0 \overset{?}{\Longleftrightarrow} \Phi(\underline{x}^*, \dots, \underline{x}^*) = \underline{x}^*$

[Φ cont. & $x^{(k)}$ cvg. & consistent $\Longrightarrow$ limit is solution :

$\lim_{k\to\infty}$: | $\underline{x}^{(k)} = \Phi_F(\underline{x}^{(k-1)}, \dots, x^{(k-m)})$

2

$$\underline{x}^* = \lim_{k\to\infty} \phi(\underline{x}^{(k-1)}, \dots, \underline{x}^{(k-m)}) \overset{\text{cont.}}{=} \phi_F(\underline{x}^*, \dots, \underline{x}^*)$$

- Speed of cvg.: How fast $\|\underline{x}^{(k)} - \underline{x}^*\| \to 0$ for some norm on $\mathbb{R}^n$

Much depends on initial guess:

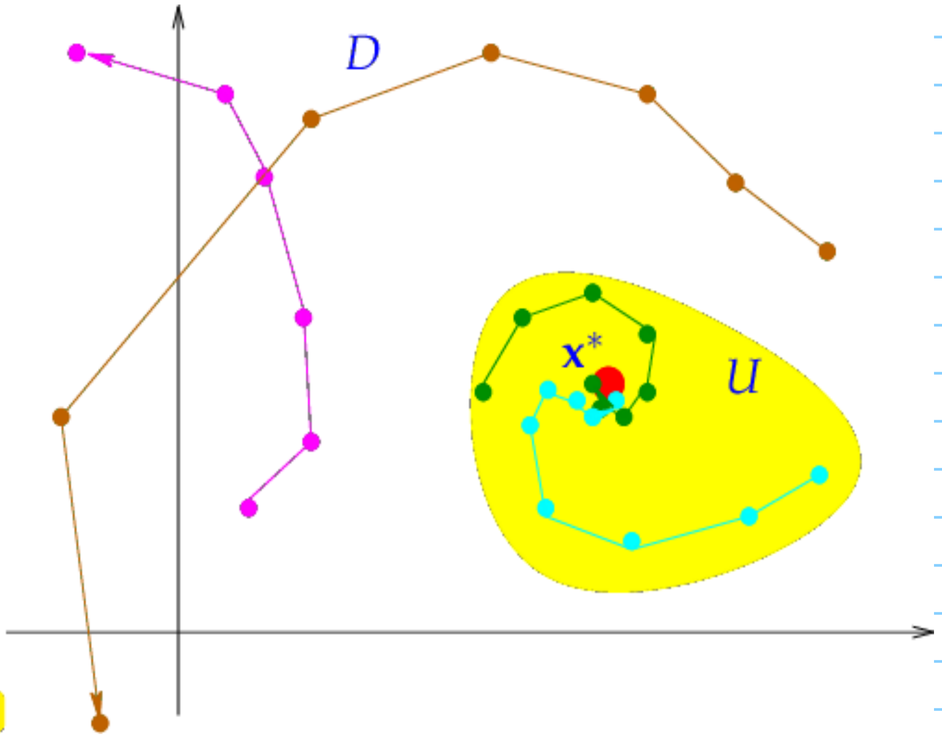


Fig. 283

Definition 8.1.8. Local and global convergence → [?, Def. 17.1]

As stationary m-point iterative method converges locally to $\mathbf{x}^* \in \mathbb{R}^n$, if there is a neighborhood $U \subset D$ of $\mathbf{x}^*$, such that

$$\mathbf{x}^{(0)}, \dots, \mathbf{x}^{(m-1)} \in U \quad \Rightarrow \quad \mathbf{x}^{(k)} \text{ well defined} \quad \wedge \quad \lim_{k\to\infty} \mathbf{x}^{(k)} = \mathbf{x}^*$$

where $(\mathbf{x}^{(k)})_{k\in\mathbb{N}_0}$ is the (infinite) sequence of iterates.
If $U = D$, the iterative method is globally convergent.

2.1.1. Speed of Convergence

Definition 8.1.9. Linear convergence

A sequence $\mathbf{x}^{(k)}$, $k = 0, 1, 2, \dots$, in $\mathbb{R}^n$ converges linearly to $\mathbf{x}^* \in \mathbb{R}^n$,

$$L \geq 0, \quad \exists L < 1: \quad \left\|\mathbf{x}^{(k+1)} - \mathbf{x}^*\right\| \leq L \left\|\mathbf{x}^{(k)} - \mathbf{x}^*\right\| \quad \forall k \in \mathbb{N}_0 .$$

smallest L : rate

How to tell linear cvg. in an experiment?

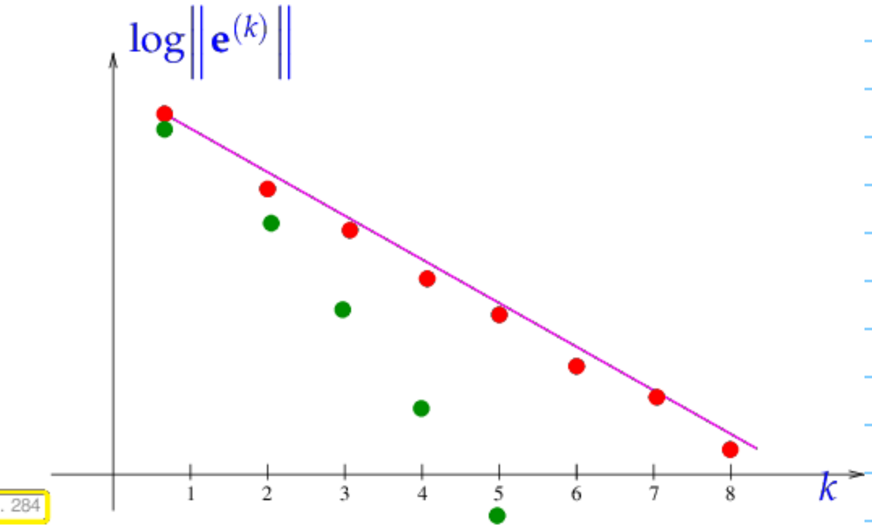
Assume sharpness $\varepsilon_k := \|\underline{x}^{(k)} - \underline{x}^*\| \approx L \|\underline{x}^{(k-1)} - \underline{x}^*\| = \varepsilon_{k-1}$

Visual evidence of linear cvg:

$$\varepsilon_k \approx L\, \varepsilon_{k-1}$$

$$\log \cdot \,| \quad \varepsilon_k \approx L^k \varepsilon_0 \Rightarrow \log \varepsilon_k \approx k \log L + \log \varepsilon_0$$

log | $\epsilon_k \approx L^k \epsilon_0 \Rightarrow \underbrace{\log \epsilon_k \approx k \log L + \log \epsilon_0}$



Data points on straight line* in k vs. $\log \epsilon_k$ plot.

* slope ~ rate

! Different notions & and terminology for

Approximation error
quadrature errors
↓
alg. cvg. / exp. cvg.

iteration errors
↓
linear cvg.
cvg. of order p

"Faster" than linear cvg:

Definition 8.1.17. Order of convergence → [?, Sect. 17.2], [?, Def. 5.14], [?, Def. 6.1]

A **convergent** sequence $\mathbf{x}^{(k)}$, $k = 0, 1, 2, \ldots$, in $\mathbb{R}^n$ with limit $\mathbf{x}^* \in \mathbb{R}^n$ converges with order p, if

$$\exists C > 0: \quad \left\| \mathbf{x}^{(k+1)} - \mathbf{x}^* \right\| \leq C \left\| \mathbf{x}^{(k)} - \mathbf{x}^* \right\|^p \quad \forall k \in \mathbb{N}_0 , \tag{8.1.18}$$

and, in addition, $C < 1$ in the case $p = 1$ (linear convergence → Def. 8.1.9).

How to tell cvg of order $p > 1$?
[assuming sharpness of estimate]

log | $\epsilon_k \approx C \epsilon_{k-1}^p$

$$\log \epsilon_k \approx \log C + p \log \epsilon_{k-1}$$
$$- \quad \log \epsilon_{k-1} \approx \log C + p \log \epsilon_{k-2}$$

$$p \approx \underbrace{\frac{\log \epsilon_k - \log \epsilon_{k-1}}{\log \epsilon_{k-1} - \log \epsilon_{k-2}}}_{\text{study these quotients}}$$

NLSE : $F(\underline{x}) = 0$, $F : \mathbb{R}^n \to \mathbb{R}^n$

Iteration produces sequence $(x^{(k)})_{k \in \mathbb{N}_0}$

Linear cvg : $\|x^{(k+1)} - x^*\| \leq L \|x^{(k)} - x^*\|$ for $0 \leq L < 1$

order-p cvg. : $\|x^{(k+1)} - x^*\| \leq C \|x^{(k)} - x^*\|^p$, $C > 0$
$(p > 1)$

Example : $\sqrt{}$-iteration $(n = 1)$

$$\underbrace{x^{(k+1)} = \frac{1}{2}\left(x^{(k)} + \frac{a}{x^{(k)}}\right)} \Rightarrow |x^{(k+1)} - \sqrt{a}| = \frac{1}{2x^{(k)}}|x^{(k)} - \sqrt{a}|^2 . \qquad (8.1.21)$$

$a > 0$

1-point iteration : $\Phi(t) = \frac{1}{2}\left(t + \frac{a}{t}\right)$

Convergence to $\sqrt{a}$ with order 2

$\|\underline{x}^{(k+1)} - x^*\| \leq C \|\underline{x}^{(k)} - \underline{x}^*\|^2$

Note: Convergence of order $p > 1$ guarantees local convergence

$\|\underline{x}^{(k+1)} - x^*\| \leq C \|x^{(k)} - x^*\|^{p-1} \|x^{(k)} - x^*\|$

If $x^{(0)} \in \mathcal{U} := \{ \underline{z} : C\|z - x^*\|^{p-1} < 1 \}$

$\Rightarrow x^{(k)} \to x^*$ for $k \to \infty$

k	$x^{(k)}$	$e^{(k)} := x^{(k)} - \sqrt{2}$	$\log \frac{\lvert e^{(k)} \rvert}{\lvert e^{(k-1)} \rvert} : \log \frac{\lvert e^{(k-1)} \rvert}{\lvert e^{(k-2)} \rvert}$
0	2.00000000000000000	0.58578643762690485	
1	1.50000000000000000	0.08578643762690485	
2	1.41666666666666652	0.00245310429357137	1.850
3	1.41421568627450966	0.00000212390141452	1.984
4	1.41421356237468987	0.00000000000159472	2.000
5	1.41421356237309492	0.00000000000000022	$\boxed{0.630}$

Note the doubling of the number of significant digits in each step ! [impact of roundoff !]

Relative errors : $x^{(k)} = x^*(1 + \delta_k)$

order-2 cvg : $\|x^{(k+1)} - x^*\| \leq C \|x^{(k)} - x^*\|^2$

$\delta_{k+1} \|x^*\| \leq C \delta_k^2 \|x^*\|^2$

"sharpness" $\delta_{k+1} \approx C\|x^*\| \delta_k^2$

If $C\|x^*\| \approx 1$, $\delta_k \approx 10^{-\ell} \Rightarrow \delta_{k+1} \approx 10^{-2\ell}$

2.1.2 Termination

↳ Stopping rules

Ideal:

STOP, if $\|x^{(k)} - x^*\| \leq \tau_{rel} \|x^*\|$: relative error

$\|x^{(k)} - x^*\| \leq \tau_{abs}$: absolute error

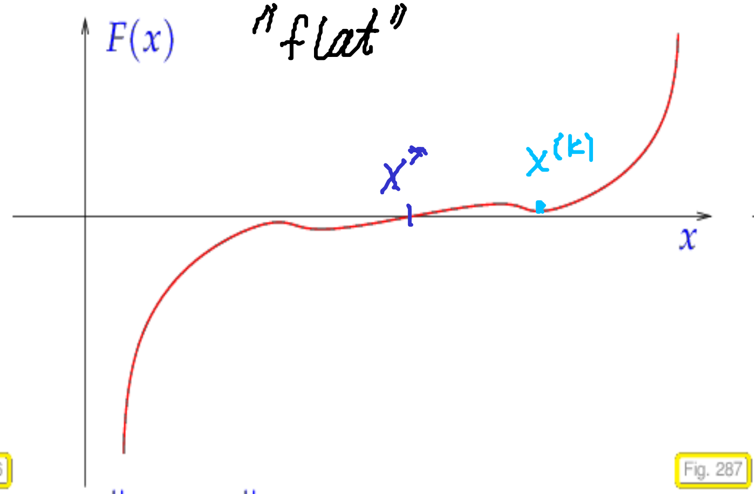
↑ not accessible

Practical:

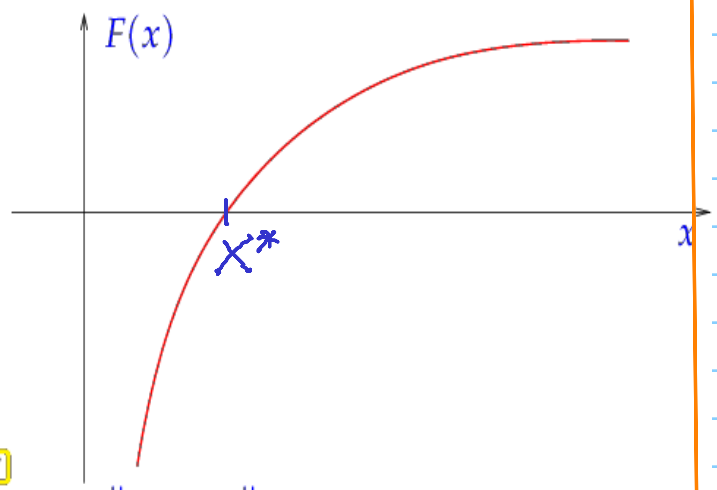
① Residual based termination

STOP, if $\underbrace{\|F(x^{(k)})\| \leq \tau}_{\text{tells little about error}}$

"flat"



$\|F(\mathbf{x}^{(k)})\|$ small $\not\Rightarrow |x - x^*|$ small



$\|F(\mathbf{x}^{(k)})\|$ small $\Rightarrow |x - x^*|$ small

② Correction based termination:

STOP, if $\|x^{(k+1)} - x^{(k)}\| \leq \begin{cases} \tau_{rel} \|x^{(k+1)}\| \\ \tau_{abs} \end{cases}$

→ Still no guarantee, except for linearly cvg. iterations with known rate L

$\|x^{(k+1)} - x^*\| \leq L \|x^{(k)} - x^*\|$

$\|x^{(k)} - x^*\| = \|x^{(k)} - x^{(k+1)} + x^{(k+1)} - x^*\|$

$$\left\|\mathbf{x}^{(k)} - \mathbf{x}^*\right\| \overset{\triangle\text{-inequ.}}{\leq} \left\|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\right\| + \left\|\mathbf{x}^{(k+1)} - \mathbf{x}^*\right\| \leq \left\|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\right\| + L\left\|\mathbf{x}^{(k)} - \mathbf{x}^*\right\| .$$

$\frac{1-L}{L} \|x^{(k+1)} - x^*\| \leq (1-L) \|x^{(k)} - x^*\| \leq \|x^{(k+1)} - x^{(k)}\|$

▶ Iterates satisfy: $$\boxed{\left\|\mathbf{x}^{(k+1)} - \mathbf{x}^*\right\| \leq \frac{L}{1-L}\left\|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\right\|} . \qquad (8.1.29)$$

! Upper bound $\tilde{L} < 1$ for L sufficient to obtain reliable bound!

$\|x^{(k)} - x^*\| \leq L^{k-1} \|x^{(1)} - x^*\|$

6

$$\left\|\mathbf{x}^*-\mathbf{x}^{(k)}\right\| \leq \frac{L^{k-l}}{1-L}\left\|\mathbf{x}^{(l+1)}-\mathbf{x}^{(l)}\right\| . \qquad (8.2.21)$$

Set $l=0$ in (8.2.21)	Set $l=k-1$ in (8.2.21)
a priori termination criterion	a posteriori termination criterion
$\boxed{\left\|\mathbf{x}^*-\mathbf{x}^{(k)}\right\| \leq \frac{L^{k}}{1-L}\left\|\mathbf{x}^{(1)}-\mathbf{x}^{(0)}\right\|}$ (8.2.22)	$\left\|\mathbf{x}^*-\mathbf{x}^{(k)}\right\| \leq \frac{L}{1-L}\left\|\mathbf{x}^{(k)}-\mathbf{x}^{(k-1)}\right\|$ (8.2.23)

- $\Phi_2(x) = x \Leftrightarrow 1+x = (1+e^x)x \Leftrightarrow 1 = xe^x$
  $\Leftrightarrow F(x) = 0$
-

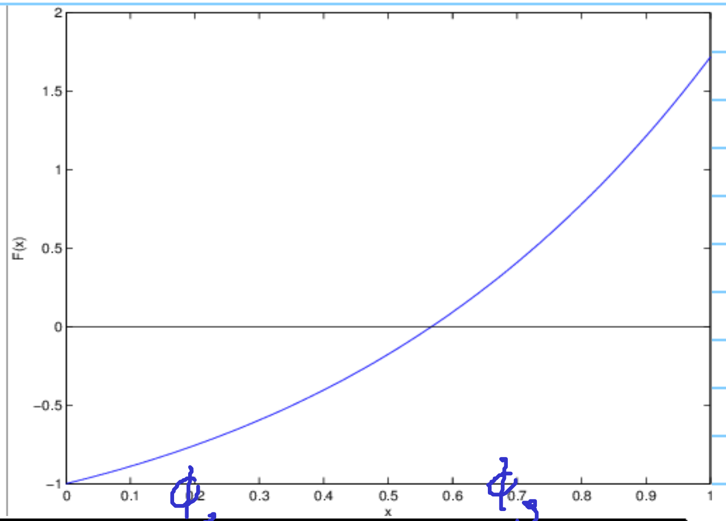
2.2. Fixed Point Iterations

$$x^{(k+1)} = \Phi_F(x^{(k)})$$

$\Phi : U \subset \mathbb{R}^n \to \mathbb{R}^n$ iteration function

consistent : $\Phi_F(x^*) = x^* \Leftrightarrow F(x^*) = 0$
with $F(x)=0$

$$F(x) = xe^x - 1\,, \quad x \in [0,1]\,.$$



Different fixed point forms:

$$\Phi_1(x) = e^{-x}\,,$$
$$\Phi_2(x) = \frac{1+x}{1+e^x}\,,$$
$$\Phi_3(x) = x + 1 - xe^x\,.$$

[all consistent]

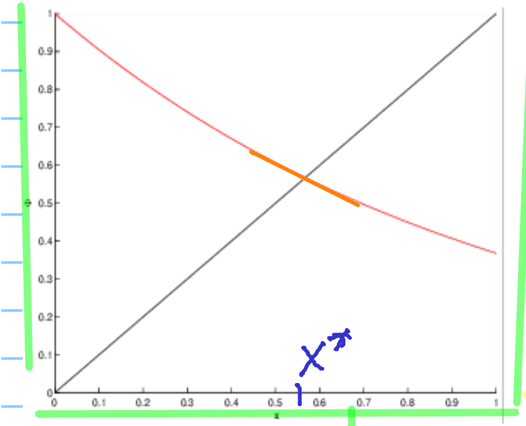
$x^{(0)} = 0.5$:

Φ_2 : order 2-cvg

Φ_1 : lin cvg.

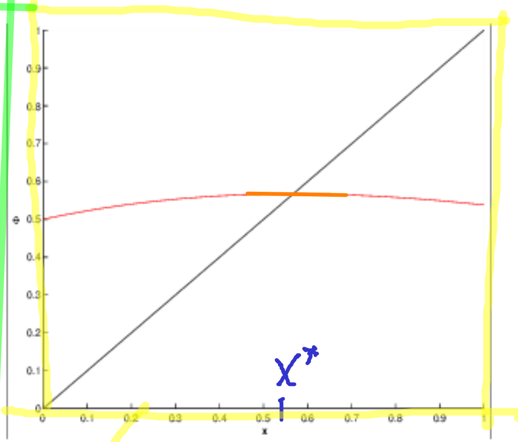
Φ_3 : no cvg.

k	Φ_1: $\lvert x_1^{(k+1)} - x^*\rvert$	Φ_2: $\lvert x_2^{(k+1)} - x^*\rvert$	Φ_3: $\lvert x_3^{(k+1)} - x^*\rvert$
0	0.067143290409784	0.067143290409784	0.067143290409784
1	0.039387369302849	0.000832287212566	0.108496074240152
2	0.021904078517179	0.000000125374922	0.219330611898582
3	0.012559804468284	0.000000000000003	0.288178118764323
4	0.007078662470882	0.000000000000000	0.723649245792953
5	0.004028858567431	0.000000000000000	0.410183132337935
6	0.002280343429460	0.000000000000000	1.186907542305364
7	0.001294757160282	0.000000000000000	0.146569797006362
8	0.000733837662863	0.000000000000000	0.310516641279937
9	0.000416343852458	0.000000000000000	0.357777386500765
10	0.000236077474313	0.000000000000000	0.974565695952037



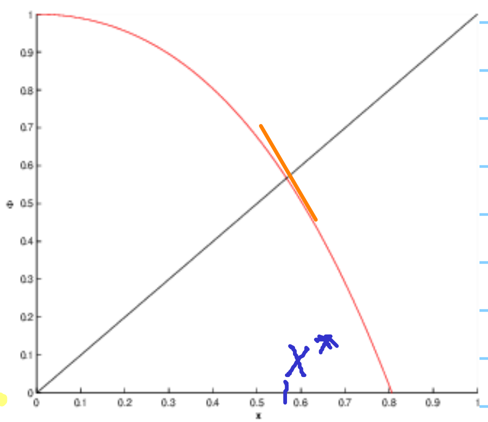
function Φ_1

$|\Phi_1'(x^*)| < 1$



function Φ_2

$\Phi_2'(x^*) = 0$



function Φ_3

$|\Phi_3'(x^*)| > 1$

▷ Slope of Φ at x^* matters! Taylor argument

$$x^{(k+1)} - x^* = \Phi(x^{(k)}) - \Phi(x^*)$$

$$= \Phi'(x^*)(x^{(k)}-x^*) + \tfrac{1}{2}\Phi''(x^*)(x^{(k)}-x^*)^2 + \underbrace{\tfrac{1}{6}\Phi'''(\xi)(x^{(k)}-x^*)^3}$$

If $|x^{(k)}-x^*| \ll 1$ ⟶ neglect

• and $\Phi'(x^*)=0$ (Φ "flat") $\Rightarrow$ $|x^{(k+1)}-x^*| \approx |\tfrac{1}{2}\Phi''(x^*)|\,|x^{(k)}-x^*|^2$

$\hat{=}$ order-2 cvg.

• and $|\Phi'(x^*)| < 1$ $\Rightarrow$ $|x^{(k+1)}-x^*| \leq |\Phi'(x^*)|\,|x^{(k)}-x^*|$ + small terms

$\hat{=}$ Linear convergence with rate $\approx |\Phi'(x^*)|$

Generalization to n-dim.:

Lemma 8.2.10. Sufficient condition for local linear convergence of fixed point iteration → **[?, Thm. 17.2], [?, Cor. 5.12]**

If $\Phi : U \subset \mathbb{R}^n \mapsto \mathbb{R}^n$, $\Phi(\mathbf{x}^*) = \mathbf{x}^*$, Φ differentiable in $\mathbf{x}^*$, and $\|D\,\Phi(\mathbf{x}^*)\| < 1$, then the fixed point iteration

$$\mathbf{x}^{(k+1)} := \Phi(\mathbf{x}^{(k)}) , \tag{8.2.2}$$

converges locally and at least linearly.

matrix norm, Def. 1.5.76 !

predictor for rate of lin. cvg.

If $D\Phi(x^*) = 0$ $\Rightarrow$ local order-2 cvg.

2.3. Zero Finding

Find $x^* \in I \subset \mathbb{R}$: $F(x^*) = 0$ for $F: I \to \mathbb{R}$

[F continuous]

2.3.1. Bisection

Input: $a, b \in I$ such that $F(a)F(b) < 0$ (different signs !)

$\Rightarrow$ $\exists\, x^* \in]\min\{a,b\}, \max\{a,b\}[$: $F(x^*) = 0$,

as we conclude from the intermediate value theorem.

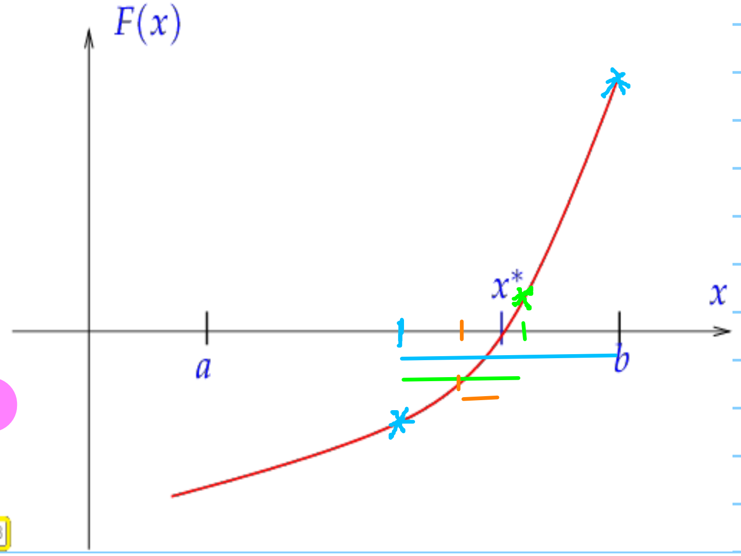


Fig. 288

C++11 code 8.3.2: Bisection method for solving $F(x) = 0$ on $[a, b]$

```
// Searching zero of F in [a,b] by bisection
template <typename Func>
double bisect(Func&& F, double a, double b, double tol)
{
  if (a > b) std::swap(a,b); // sort interval bounds
  double fa = F(a), fb = F(b);
  if (fa*fb > 0) throw "f(a) and f(b) have same sign";
  int v=1; if (fa > 0) v=-1;
  double x = 0.5*(b+a); // determine midpoint
  // termination, relies on machine arithmetic if tol = 0
  while (b-a > tol && ((a<x) && (x<b))) //
  {
    // sgn(f(x)) = sgn(f(b)), then use x as next right boundary
    if (v*F(x) > 0) b=x;
    // sgn(f(x)) = sgn(f(a)), then use x as next left boundary
    else a=x;
    x = 0.5*(a+b); // determine next midpoint
  }
  return x;
}
```

tol = 0 : Loop will stop, when

$a \mp \frac{1}{2}(b-a) = a$

$\hookrightarrow$ happens, if $|\frac{1}{2}(b-a)| \leq EPS \cdot |a|$

Convergence: "linear type"

Intervals shrink by factor 2 in each step

$|a^{(k)} - b^{(k)}| \leq 2^{-k} |a^{(0)} - b^{(0)}|$

$\Rightarrow$ $|x^{(k)} - x^*| \leq 2^{-k} |a^{(0)} - b^{(0)}|$

$\Rightarrow$ robust method

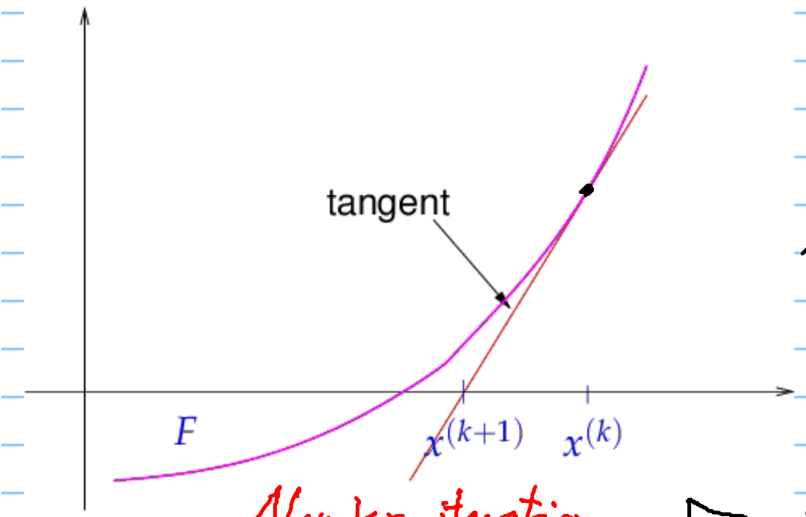
2.3.2 Model Function Methods

Idea: Replace F with $\tilde{F}: I \to \mathbb{R}$ (model function)

- depending on point values of F / F'
- whose zeros are easy to compute

↳ do this in every step of an iteration

2.3.2.1. Newtons method



Model function $\hat{F}$

≙ tangent at graph of F in $x^{(k)}$

$$\hat{F}(x) = F(x^{(k)}) + F'(x^{(k)})(x - x^{(k)})$$

$$x^{(k+1)}: \quad \hat{F}(x^{(k+1)}) = 0$$

Newton iteration ▷ $x^{(k+1)} = x^{(k)} - F(x^{(k)}) / F'(x^{(k)})$

[if $F'(x^{(k)}) \neq 0$]

↳ ≙ fixed point iteration with $\phi(x) = x - \frac{F(x)}{F'(x)}$

(Local) cvg. of Newton's method:

$$\phi'(x) = 1 - \frac{[F'(x)]^2 - F''(x)F(x)}{[F'(x)]^2} = \frac{F''(x)}{[F'(x)]^2} F(x)$$

▷ $F(x^*) = 0 \Rightarrow \phi'(x^*) = 0$, if $F'(x^*) \neq 0$

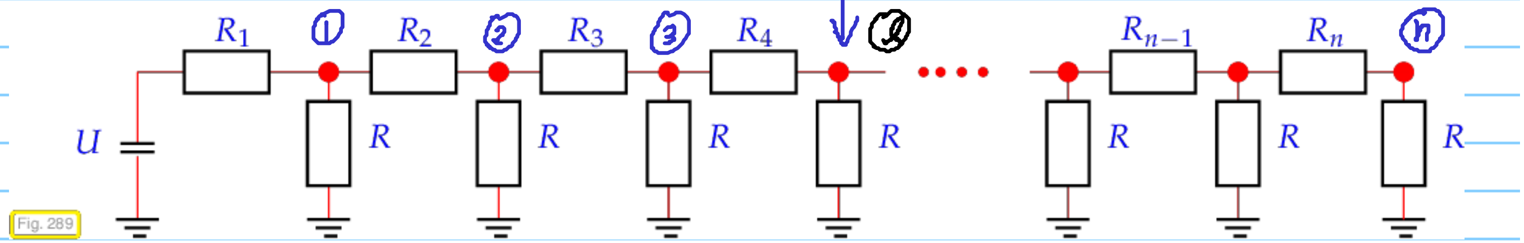
$\Rightarrow$ local 2nd-order cvg.

Convergence of Newton's method in 1D

Newton's method locally quadratically converges (→ Def. 8.1.17) to a zero x^* of F, if $F'(x^*) \neq 0$

Example: F' by implicit differentiation

achieve prescribed voltage = 1 here



[Nodal analysis] ⇒ LSE for nodal voltages u_i

$$(A + xI)\,u(x) = b \in \mathbb{R}^n$$

s.p.d. tridiagonal matrix $\in \mathbb{R}^{n,n}$

$= 1/R$ [→ unknown]

n.v.

Task: $x \in \mathbb{R}$: $\mathbf{w}^T \mathbf{u}(x) = 1$, $\mathbf{w} = \mathbf{e}_\ell$

$\Updownarrow$

Find zero of $F(x) = \mathbf{w}^T\mathbf{u}(x) - 1 = \mathbf{w}^T(A+xI)^{-1}\mathbf{b} - 1$

$F'(x) = \mathbf{w}^T \mathbf{u}'(x)$

$\mathbf{u}(x)$: $(A+xI)\mathbf{u}(x) = \mathbf{b} \quad \Big| \frac{d}{dx}$

$I\cdot\mathbf{u}(x) + (A+xI)\cdot\mathbf{u}'(x) = 0$

with product rule: $(fg)' = f'\cdot g + f\cdot g'$

$\Rightarrow \quad \mathbf{u}'(x) = -(A+xI)^{-1}\mathbf{u}(x)$

▷ Newton iteration:

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

$$= x^{(k)} - \frac{\mathbf{w}^T\mathbf{u}(x^{(k)}) - 1}{\mathbf{w}^T\mathbf{z}}$$

where: $(A + x^{(k)}I)\,\mathbf{u}(x^{(k)}) = \mathbf{b}$

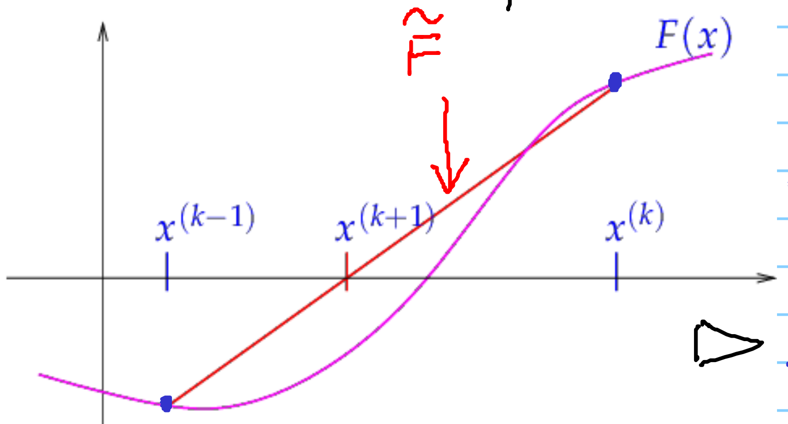
$$(A + x^{(k)}I)\,\underbrace{\mathbf{z}}_{:=\,\mathbf{u}'(x^{(k)})} = -\mathbf{u}(x^{(k)})$$

2.3.2.2. Multi-point methods

m-point method: $x^{(k+1)} = \phi(x^{(k)}, \dots, x^{(k-m+1)})$

↳ m initial $x^{(j)}$ required

Secant method: 2-point method



$\tilde{F}(x) = F(x^{(k)}) + \Delta s \cdot (x - x^{(k)})$

$$\Delta s = \frac{F(x^{(k)}) - F(x^{(k-1)})}{x^{(k)} - x^{(k-1)}}$$

▷ $$x^{(k+1)} = x^{(k)} - \underbrace{\frac{F(x^{(k)})}{\Delta s}}_{=:\,s}$$

1 F-evaluation per step

C++11 code 8.3.25: Secant method for 1D non-linear equaton

```
// Secant method for solving F(x) = 0 for F : D ⊂ ℝ → ℝ,
// initial guesses x0, x1,
// tolerances atol (absolute), rtol (relative)
template<typename Func>
double secant(double x0, double x1, Func&& F,
              double rtol, double atol, unsigned int maxIt)
{
  double fo = F(x0);
  for (unsigned int i=0; i<maxIt; ++i) {
      double fn = F(x1);
      double s = fn*(x1-x0)/(fn-fo); // secant correction
      x0 = x1; x1 = x1-s;
      // correction based termination (relative and absolute)
      if (abs(s) < max(atol,rtol*min(abs(x0),abs(x1))))
     return x1;
      fo = fn;
  }
  return x1;
}
```

(Empiric) convergence of secant method :

$F(x) = xe^x - 1$, $x^{(0)} = 0$, $x^{(1)} = 5$

estimate for order of cvg.

k	$x^{(k)}$	$F(x^{(k)})$	$e^{(k)} := x^{(k)} - x^*$	$\dfrac{\log\lvert e^{(k+1)}\rvert - \log\lvert e^{(k)}\rvert}{\log\lvert e^{(k)}\rvert - \log\lvert e^{(k-1)}\rvert}$
2	0.00673794699909	-0.99321649977589	-0.56040534341070	
3	0.01342122983571	-0.98639742654892	-0.55372206057408	24.43308649757745
4	0.98017620833821	1.61209684919288	0.41303291792843	2.70802321457994
5	0.38040476787948	-0.44351476841567	-0.18673852253030	1.48753625853887
6	0.50981028847430	-0.15117846201565	-0.05733300193548	1.51452723840131
7	0.57673091089295	0.02670169957932	0.00958762048317	1.70075240166256
8	0.56668541543431	-0.00126473620459	-0.00045787497547	1.59458505614449
9	0.56713970649585	-0.00000990312376	-0.00000358391394	1.62641838319117
10	0.56714329175406	0.00000000371452	0.00000000134427	
11	0.56714329040978	-0.00000000000001	-0.00000000000000	

$\Rightarrow$ order 1.6 convergence

Generalization: Inverse interpolation

Assume: $F : I \subset \mathbb{R} \mapsto \mathbb{R}$ one-to-one (monotone)

$$F(x^*) = 0 \quad \Rightarrow \quad F^{-1}(0) = x^* .$$

- Interpolate F^{-1} by polynomial p of degree $m-1$ determined by

$$p(F(x^{(k-j)})) = x^{(k-j)} , \quad j = 0, \dots, m-1 .$$

- New approximate zero $x^{(k+1)} := p(0)$

The graph of F^{-1} can be obtained by reflecting the graph of F at the angular bisector. ▷

$$F(x^*) = 0 \quad \Leftrightarrow \quad F^{-1}(0) = x^*$$

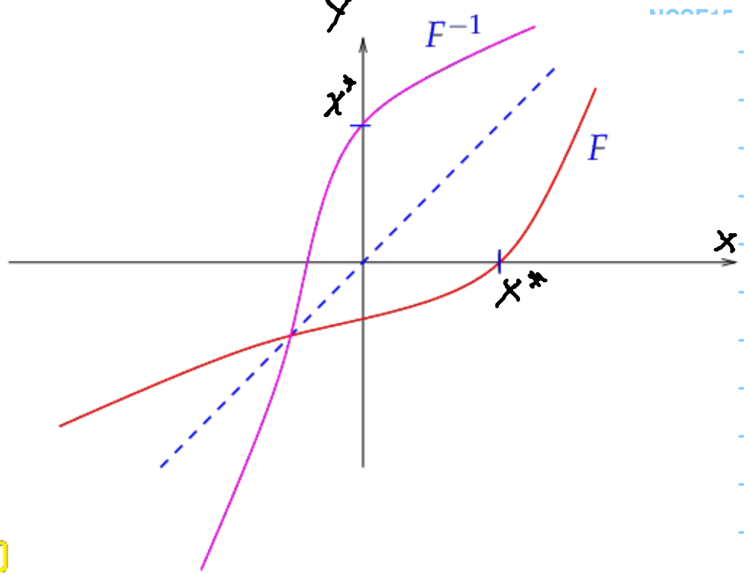


Fig. 292

Case $m = 2$ (2-point method)
➢ secant method

The interpolation polynomial is a line. In this case we do not get a new method, because the inverse function of a linear function (polynomial of degree 1) is again a polynomial of degree 1.

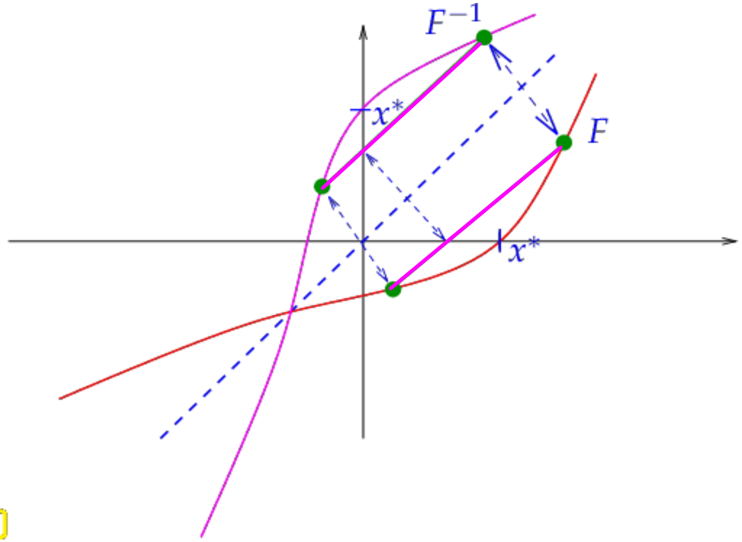


Fig. 293

m = 3 :

$$x^{(k+1)} = \frac{F_0^2(F_1x_2 - F_2x_1) + F_1^2(F_2x_0 - F_0x_2) + F_2^2(F_0x_1 - F_1x_0)}{F_0^2(F_1 - F_2) + F_1^2(F_2 - F_0) + F_2^2(F_0 - F_1)} .$$

$$\left(F_0 := F(x^{(k-2)}), F_1 := F(x^{(k-1)}), F_2 := F(x^{(k)}), x_0 := x^{(k-2)}, x_1 := x^{(k-1)}, x_2 := x^{(k)} \right)$$

$F(x) = xe^x - 1$, $x^{(0)} = 0$, $x^{(1)} = 2.5$, $x^{(2)} = 5$.

k	$x^{(k)}$	$F(x^{(k)})$	$e^{(k)} := x^{(k)} - x^*$	$\frac{\log\lvert e^{(k+1)}\rvert - \log\lvert e^{(k)}\rvert}{\log\lvert e^{(k)}\rvert - \log\lvert e^{(k-1)}\rvert}$
3	0.08520390058175	-0.90721814294134	-0.48193938982803	
4	0.16009252622586	-0.81211229637354	-0.40705076418392	3.33791154378839
5	0.79879381816390	0.77560534067946	0.23165052775411	2.28740488912208
6	0.63094636752843	0.18579323999999	0.06380307711864	1.82494667289715
7	0.56107750991028	-0.01667806436181	-0.00606578049951	1.87323264214217
8	0.56706941033107	-0.00020413476766	-0.00007388007872	1.79832936980454
9	0.56714331707092	0.00000007367067	0.00000002666114	1.84841261527097
10	0.56714329040980	0.00000000000003	0.00000000000001	

fractional order

2.3.3. Asymptotic efficiency

$\hookrightarrow \frac{\text{gain}}{\text{work}}$ ← [no. of digits] prescribed error reduction ← #F-eval + #F'-eval

gain: $\boxed{\|e^{(k)}\| \le \rho \|e^{(0)}\|}$ with $\rho \ll 1$ (*)

$K(\rho)$: minimal no. of iterations required for (*)

▷ Efficience : $\frac{|\log \rho|}{K(\rho) \cdot W}$, $W \hat{=}$ work/iteration

• Linearly cvgt. iteration: $\|e^{(k)}\| \le L^k \|e^{(0)}\|$, $L<1$

$\Rightarrow K(\rho) \ge \frac{\log \rho}{\log L} \Rightarrow \text{Eff.} = \frac{|\log L|}{W}$

• order $p>1$: $\|e^{(k)}\| \le C\|e^{(k-1)}\|^p$

$\|e^{(k)}\| \le \underbrace{C^{1+p+p^2+\dots+p^{k-1}}}_{= C^{\frac{p^k-1}{p-1}}} \|e^{(0)}\|^{p^k-1} \cdot \|e^{(0)}\|$

▷ $K(\rho)$: $\Big(\underbrace{C^{\frac{1}{p-1}} \|e^{(0)}\|}_{=: L_0}\Big)^{p^k-1} \le \rho$

$p^k \ge \frac{\log \rho}{\log L_0} + 1 \qquad | \cdot \log$

$k \ge \log\left(\frac{\log \rho}{\log L_0} \cancel{+1}\right) / \log p$

$\rho \ll 1$: $k \cdot \log p \ge \log\log\rho - \cancel{\log\log L_0}$

$K(\rho) = \frac{\log\log\rho}{\log p}$

$\text{Efficiency} = \frac{\log p}{W} \frac{\log \rho}{\log\log\rho}$

$\hookrightarrow$ different for different methods

$W_{\text{Newton}} = 2W_{\text{secant}}$, $p_{\text{Newton}} = 2$, $p_{\text{secant}} = 1.62$ $\Rightarrow \frac{\log p_{\text{Newton}}}{W_{\text{Newton}}} : \frac{\log p_{\text{secant}}}{W_{\text{secant}}} = 0.71$.

13

NCSE15

▷ "Secant method is 30% more efficient than Newton"

2.4. Newton's Method (for $n > 1$)

$\underline{x}^* \in \mathbb{R}^n$: $\quad F(\underline{x}^*) = 0$, $F : D \subset \mathbb{R}^n \to \mathbb{R}^n$

differentiable

"Tangent plane" at $x^{(k)}$ $\hat{=}$ model function by linearization

$\underline{x}^{(k+1)}$: $\quad \hat{F}(x) = F(\underline{x}^{(k)}) + DF(x^{(k)})(\underline{x} - \underline{x}^{(k)}) \overset{!}{=} 0$

Jacobian $\in \mathbb{R}^{n,n}$

Newton iteration $\quad \boxed{\underline{x}^{(k+1)} = \underline{x}^{(k)} - \underbrace{DF(\underline{x}^{(k)})^{-1} F(\underline{x}^{(k)})}}$

Newton correction $\Delta x^{(k)}$: by solving a LSE

Affine invariance :

Newton iterations for $\to F(x) = \underline{0}$; $\to A F(x) = \underline{0}$, $A \in \mathbb{R}^{n,n}$ regular

produce the same iterates, if started with the same value

$[\ G(x) := AF(x) \Rightarrow DG(x) = A\,DF(x)\ ]$

N.I. for $G(x) = 0$:

$$x^{(k+1)} = x^{(k)} - [\cancel{A} DF(x^{(k)})]^{-1} [\cancel{A} F(x^{(k)})]$$

2.4.2. Convergence of Newton's method

→ = Fixed point iteration with iteration function

$$\Phi(\underline{x}) := \underline{x} - DF(\underline{x})^{-1} \cdot F(\underline{x}) \ , \ x \in D$$

[product rule]

$$D\Phi(x^*) = \cancel{I} - ? \cdot \overbrace{F(x^*)}^{=0} - \cancel{DF(x)^{-1} DF(x)} = 0$$

$\Rightarrow$ $\boxed{\text{local quadratic cvg.}}$

Example : Quasi-linear system of equation

$$A(\underline{x})\underline{x} = \underline{b} \ , \quad A : \mathbb{R}^n \to \mathbb{R}^{n,n}$$

$$\boxed{\mathbf{A}(\mathbf{x})\mathbf{x} = \mathbf{b}}\ , \quad \mathbf{A}(\mathbf{x}) = \begin{pmatrix} \gamma(\mathbf{x}) & 1 & & & \\ 1 & \gamma(\mathbf{x}) & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \ddots \\ & & 1 & \gamma(\mathbf{x}) & 1 \\ & & & 1 & \gamma(\mathbf{x}) \end{pmatrix} \in \mathbb{R}^{n\times n}\ , \tag{8.4.20}$$

$\gamma(x) = 3 + \|x\|$

$$\Longleftrightarrow \quad F(\underline{x}) = \underline{b} - A(\underline{x})\underline{x} = \underline{0}$$

$$\mathbf{A}(\mathbf{x})\mathbf{x} = \mathbf{T}\mathbf{x} + \mathbf{x}\|\mathbf{x}\|_2\ , \quad \mathbf{T} := \begin{pmatrix} 3 & 1 & & & \\ 1 & 3 & 1 & & \\ & \ddots & 3 & \ddots & \\ & & \ddots & \ddots & \ddots \\ & & & 1 & 3 & 1 \\ & & & & 1 & 3 \end{pmatrix}.$$

$$\Rightarrow \quad F(\underline{x}) = \underline{b} - T\underline{x} - \|x\|\cdot\underline{x}$$

$$\Rightarrow \quad DF(\underline{x}) = -T - D\{\underbrace{x \to \|x\|\cdot x}_{=:\, g(\underline{x})}\}$$

$DF(x)$: (i) product rule

(ii) direct computation

$$(g(\underline{x}))_i = \sqrt{x_1^2 + \dots + x_n^2}\cdot x_i$$

$$(Dg(\underline{x}))_{i,j} = \frac{\partial g_i(x)}{\partial x_j} = \begin{cases} * & j = i \\ \dfrac{x_j}{\sqrt{x_1^2 + \dots + x_n^2}}\, x_i\ , & j \neq i \end{cases}$$

$$\frac{\partial}{\partial x_i}(\mathbf{x}\|\mathbf{x}\|)_i = \underbrace{\sqrt{x_1^2 + \cdots + x_n^2}}_{=\|x\|} + x_i \frac{x_i}{\sqrt{x_1^2 + \cdots + x_n^2}}\ ,$$

$$\frac{\partial}{\partial x_j}(\mathbf{x}\|\mathbf{x}\|)_i = x_i \frac{x_j}{\sqrt{x_1^2 + \cdots + x_n^2}}\ , \quad j \neq i\ .$$

$$\boxed{DF(\mathbf{x})} = \mathbf{T} + \|\mathbf{x}\|_2 + \mathbf{x}\frac{\mathbf{x}^\top}{\|\mathbf{x}\|_2} = \boxed{\left(\mathbf{A}(\mathbf{x}) + \frac{\mathbf{x}\mathbf{x}^\top}{\|\mathbf{x}\|_2}\right)}\ .$$

rank-1-modification of tridiagonal matrix A

$\Rightarrow$ Use Sherman-Morrison-Woodbury formula for computing Newton correction : cost $O(n)$

15

Example: Matrix inversion via Newton

$A \in \mathbb{R}^{n,n}$ regular: $F(X) = A - X^{-1} \overset{!}{=} 0$

$F : \mathbb{R}^{n,n} \to \mathbb{R}^{n,n}$

Newton iteration for $F(X) = 0$?

Trick: implicit differentiation $\quad F(X)\cdot X = AX - I$

[product rule]

[" General derivative of $\phi : V \to W$

$$D\phi(v)h = \lim_{\tau \to 0} \frac{\phi(v+\tau h) - \phi(v)}{\tau}$$]

h ↑ direction of differentiation

Product rule: $F : D \subset V \mapsto W$, $G : D \subset V \mapsto U$ sufficiently smooth, $b : W \times U \mapsto Z$ bilinear, *ie.*, linear in each argument:

$$T(\mathbf{x}) = b(F(\mathbf{x}), G(\mathbf{x})) \quad \Rightarrow \quad \mathrm{D}\,T(\mathbf{x})\mathbf{h} = b(\mathrm{D}\,F(\mathbf{x})\mathbf{h}, G(\mathbf{x})) + b(F(\mathbf{x}), \mathrm{D}\,G(\mathbf{x})\mathbf{h}) , \qquad (8.4.9)$$

$\mathbf{h} \in V, \mathbf{x} \in D$.

$$(f\cdot g)' = f'\cdot g + f\cdot g'$$

"$\frac{d}{dX}$" $F(X)\cdot X = AX - I$

$\Rightarrow \quad D(F(X)\cdot X)H = D(AX - I)H$

- General product rule with $b \Leftrightarrow$ matrix multiplication.
- $D\{X \to X\}H = H$ $\qquad H \in \mathbb{R}^{n,n}$

$$DF(X)(H)\cdot X + F(X)\cdot H = A\cdot H$$

$$\triangleright \quad \underbrace{DF(X)(H)}_{\in \mathbb{R}^{n,n}} = (A\cdot H - F(X)\cdot H)X^{-1} = X^{-1}HX^{-1}$$

Newton iteration: $DF(X)$ is not a matrix!

$$X^{(k+1)} = X^{(k)} - S, \quad DF(X^{(k)})(S) = F(X^{(k)})$$

$$\Updownarrow$$

$$(X^{(k)})^{-1}S(X^{(k)})^{-1} = A - (X^{(k)})^{-1}$$

$$\Leftrightarrow \quad S = X^{(k)}AX^{(k)} - X^{(k)}$$

$$\triangleright \quad X^{(k+1)} = 2X^{(k)} - X^{(k)}AX^{(k)}$$

General notion of derivative and Newton's method

For $f : \mathbb{R} \to \mathbb{R}$ smooth: local linear approximation

$$f(x+h) \approx f(x) + f'(x)h \quad , h \in \mathbb{R}$$

General: $F : V \to W$, $V, W \hat{=}$ vector space

$$F(\underline{x}+\underline{h}) \approx F(\underline{x}) + DF(\underline{x})(\underline{h}) \quad , \forall \underline{h} \in V$$

$\uparrow$ = linear mapping $V \longrightarrow W$

Note:
- $c \in \mathbb{R} \iff$ linear mapping $\mathbb{R} \to \mathbb{R}$, $t \to c \cdot t$
- $V = W = \mathbb{R}^n$: Jacobi matrix describes linear mapping $DF(x) : \mathbb{R}^n \to \mathbb{R}^n$

Abstract Newton iteration for solving $F(x) = 0$ for $F : D \subset V \to W$

$$\underline{x}^{(k+1)} = \underline{x}^{(k)} + \underline{s} \quad , \quad DF(x^{(k)})\underline{s} = -F(x^{(k)})$$

$\to$ To state Newton iteration we have to find expressions for $DF(x^{(k)})\underline{h}$, $\underline{h} \in V$

Application: Derivative of matrix inversion

Tool: Product rule for general derivatives

- Vector spaces: V, W, U, Z
- $F : V \to W$, $G : V \to U$ differentiable mappings
- $b : W \times U \to Z$ bilinear (linear in each argument)
- $T(\underline{x}) := b(F(x), G(x))$, $T : V \to Z$

Simplest setting: $V = W = U = Z = \mathbb{R}$, $b(\xi, \eta) = \xi \cdot \eta$

$$T(x) = F(x) \cdot G(x) \quad , x \in \mathbb{R}$$

std. prod. rule $\Rightarrow$ $T'(x) = F'(x) \cdot G(x) + F(x) \cdot G'(x) \in \mathbb{R}$

Regarded as linear mappings: $T'(x)h = F'(x)h \cdot G(x) + F(x) \cdot (G'(x)h) \quad \forall h \in \mathbb{R}$

$\uparrow$ linear mapping $\mathbb{R} \to \mathbb{R}$

Product rule: $F : D \subset V \mapsto W$, $G : D \subset V \mapsto U$ sufficiently smooth, $b : W \times U \mapsto Z$ bilinear, *ie.*, linear in each arguent:

linear map $V \to W$ linear map $V \to U$

$$T(\mathbf{x}) = b(F(\mathbf{x}), G(\mathbf{x})) \quad \Rightarrow \quad DT(\mathbf{x})\mathbf{h} = b(\underbrace{DF(\mathbf{x})\mathbf{h}}_{\in W}, G(\mathbf{x})) + b(F(\mathbf{x}), \underbrace{DG(\mathbf{x})\mathbf{h}}_{\in U}) \,, \tag{8.4.9}$$

$\forall \mathbf{h} \in V, \mathbf{x} \in D$.

Matrix inversion :

invertible $n\times n$ matrices

$$\mathrm{inv} := \begin{cases} \mathbb{R}^{n,n}_{reg} \longrightarrow \mathbb{R}^{n,n} \\ X \longrightarrow X^{-1} \end{cases}$$

$\Rightarrow$ $D\mathrm{inv}(X) \hat{=}$ linear mapping $\mathbb{R}^{n,n} \to \mathbb{R}^{n,n}$

Here : Basis free approach

Implicit differentiation & product rule

[Here, $I \hat{=}$ identity matrix is a const. mapping $\mathbb{R}^{n,n} \to I$]

diff.

$$T(X) := \mathrm{inv}(X)\cdot X = I$$

$$DT(X)H = D\mathrm{inv}(X)(H)\cdot X + \mathrm{inv}(X)\cdot H = O$$

$$\boxed{D\,\mathrm{inv}(X)H = -X^{-1}HX^{-1}}\ , H\in\mathbb{R}^{n,n}$$

Product rule: $F : D \subset V \mapsto W$, $G : D \subset V \mapsto U$ sufficiently smooth, $b : W \times U \mapsto Z$ bilinear, *ie.*, linear in each argument:

$$T(\mathbf{x}) = b(F(\mathbf{x}), G(\mathbf{x})) \quad\Rightarrow\quad \mathrm{D}\,T(\mathbf{x})\mathbf{h} = b(\mathrm{D}\,F(\mathbf{x})\mathbf{h}, G(\mathbf{x})) + b(F(\mathbf{x}), \mathrm{D}\,G(\mathbf{x})\mathbf{h})\ , \qquad (8.4.9)$$
$$\mathbf{h}\in V, \mathbf{x}\in D\,.$$

applied with : $F \hat{=} \mathrm{inv}$, $G \hat{=} \mathrm{Id}$ ($\Rightarrow DG(X)H = H$)

$b \Leftrightarrow$ matrix multiply

$V = U = W = Z = \mathbb{R}^{n,n}$

! We never use that matrices can represent linear maps

Note : $D\,\mathrm{id} = \mathrm{id}$:

$$\mathrm{id}(x+h) = x + h = \mathrm{id}(x) + \mathrm{id}(h)$$
$$\Updownarrow \qquad\qquad \Updownarrow \qquad \Updownarrow$$
$$F(x+h) \approx F(x) + DF(x)(h)$$

Newton iteration for matrix inversion

$$F : \mathbb{R}^{n,n}_{reg} \to \mathbb{R}^{n,n}\ , \quad F(X) = A - X^{-1}$$

$[\ A \in \mathbb{R}^{n,n}_{reg}\ ]$

$$F(X^*) = O \iff X^* = A^{-1}$$

Derivative of F : $F(X) = A - \mathrm{inv}(X)$

$$DF(X)\cdot H = O + X^{-1}\cdot H\cdot X^{-1}$$

Newton correction :

$$DF(X^{(k)})S = (X^{(k)})^{-1}S\,(X^{(k)})^{-1} = -F(X^{(k)}) = -A + (X^{(k)})^{-1}$$

$$\Rightarrow \quad S = -X^{(k)}AX^{(k)} + X^{(k)}$$

$\Rightarrow$ Newton it. :

$$X^{(k+1)} = X^{(k)} - X^{(k)}AX^{(k)} + X^{(k)}$$
$$= X^{(k)}(2I - A\cdot X^{(k)})$$

2.4.3. Termination of Newton Iteration

Asymptotic quadratic cvg:

$\Rightarrow \quad \|x^{(k+1)} - x^*\| \ll \|x^{(k)} - x^*\|$

$\Rightarrow \quad \|x^{(k)} - x^*\| \approx \|x^{(k)} - x^{(k+1)}\|$

$\Rightarrow$ correction based termination

↳ One redundant Newton step → significant additional cost.

→ often very few steps only

Cheaper: use simplified Newton correction
(for $n \gg 1$)

$$\Delta\bar{x}^{(k)} := DF(x^{(k-1)})^{-1} F(x^{(k)})$$

LU-dec. available

▶ Economical correction based termination criterion for Newton's method:

STOP, as soon as $\left\|\Delta\bar{\mathbf{x}}^{(k)}\right\| \le \tau_{\mathrm{rel}}\left\|\mathbf{x}^{(k)}\right\|$ **or** $\left\|\Delta\bar{\mathbf{x}}^{(k)}\right\| \le \tau_{\mathrm{abs}}$, (8.4.42)

with simplfied Newton correction $\Delta\bar{\mathbf{x}}^{(k)} := \mathrm{D}F(\mathbf{x}^{(k-1)})^{-1}F(\mathbf{x}^{(k)})$.

2.4.4. Damped Newton Method

Problem: Annoyingly local convergence

Causes:

1

$$F(x) = xe^x - 1 \quad \Rightarrow \quad F'(-1) = 0$$

$x^{(0)} < -1 \quad \Rightarrow \quad x^{(k)} \to -\infty$,
$x^{(0)} > -1 \quad \Rightarrow \quad x^{(k)} \to x^*$,

because all Newton corrections for $x^{(k)} < -1$ make the iterates decrease even further.

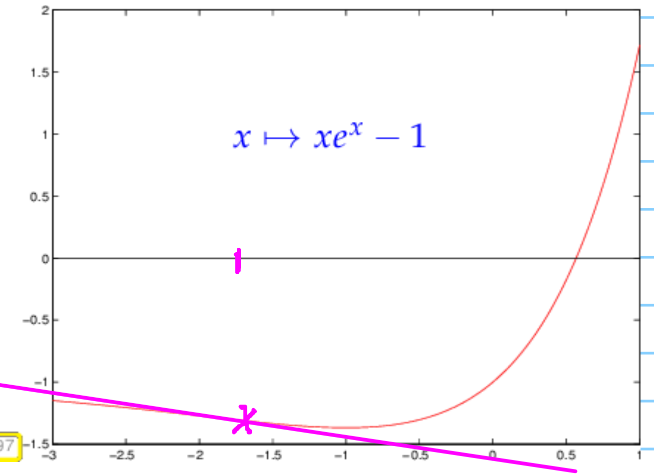


Fig. 297

→ Newton correction in the wrong direction

→ Beyond repair

2

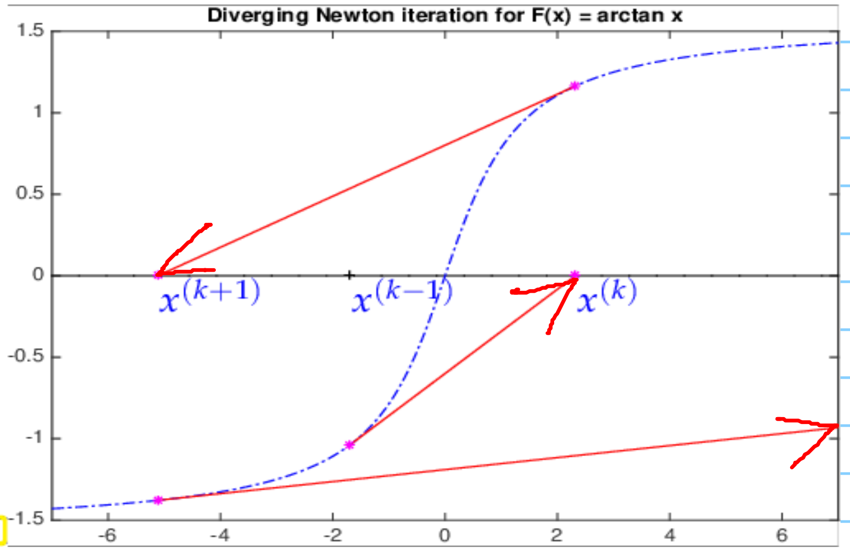


Fig. 299

Overshooting of Newton correction

Remedy: damping

▶ we observe "overshooting" of Newton correction

Idea: **damping** of Newton correction:

With $\lambda^{(k)} > 0$: $\mathbf{x}^{(k+1)} := \mathbf{x}^{(k)} - \lambda^{(k)} \, D F(\mathbf{x}^{(k)})^{-1} F(\mathbf{x}^{(k)})$. (8.4.47)

Terminology: $\lambda^{(k)}$ = damping factor , $\lambda^{(k)} \in]0,1]$

Affine invariant* damping strategy

Choice of damping factor: affine invariant **natural monotonicity test** [?, Ch. 3]: (NMT)

choose "maximal" $0 < \lambda^{(k)} \leq 1$: $\left\| \Delta\overline{\mathbf{x}}(\lambda^{(k)}) \right\| \leq \left(1 - \frac{\lambda^{(k)}}{2}\right) \left\| \Delta\mathbf{x}^{(k)} \right\|_2$ (8.4.49)

where
$\Delta\mathbf{x}^{(k)} := D F(\mathbf{x}^{(k)})^{-1} F(\mathbf{x}^{(k)})$ → current Newton correction ,
$\Delta\overline{\mathbf{x}}(\lambda^{(k)}) := D F(\mathbf{x}^{(k)})^{-1} F(\underbrace{\mathbf{x}^{(k)} + \lambda^{(k)} \Delta\mathbf{x}^{(k)}}_{})$ → tentative simplified Newton correction .

↳ tentative next iterate

* The same damping factor for $AF(x) = 0$ $\forall A \in \mathbb{R}^{n,n}_{reg}$

"Theory": If quad. cvg $\Rightarrow \|\Delta\overline{x}(1)\| \leq \frac{1}{2} \|\Delta x^{(k)}\|$

If NMT passed $\Rightarrow$ reduce damping : $\lambda \leftarrow 2 \cdot \lambda$
failed $\Rightarrow$ increase damping : $\lambda \leftarrow \lambda/2$

C++11 code 8.4.50: Generic damped Newton method based on natural monotonicity test

```cpp
template <typename FuncType, typename JacType, typename VecType>
void dampnewton(const FuncType &F, const JacType &DF,
                VecType &x, double rtol, double atol)
{
  using index_t = typename VecType::Index;
  using scalar_t = typename VecType::Scalar;
  const index_t n = x.size();
  const scalar_t lmin = 1E-3; // Minimal damping factor
  scalar_t lambda = 1.0; // Initial and actual damping factor
  VecType s(n), st(n); // Newton corrections
  VecType xn(n);       // Tentative new iterate
  scalar_t sn, stn;    // Norms of Newton corrections

  do {
    auto jacfac = DF(x).lu(); // LU-factorize Jacobian
    s = jacfac.solve(F(x));   // Newton correction
    sn = s.norm();            // Norm of Newton correction
    lambda *= 2.0;
    do {
      lambda /= 2;
      if (lambda < lmin) throw "No convergence: lambda -> 0";
      xn = x-lambda*s;          // Tentative next iterate
      st = jacfac.solve(F(xn)); // Simplified Newton correction
      stn = st.norm();
    }
    while (stn > (1-lambda/2)*sn); // Natural monotonicity test
    x = xn; // Now: xn accepted as new iterate
    lambda = std::min(2.0*lambda,1.0); // Try to mitigate damping
  }
  // Termination based on simplified Newton correction
  while ((stn > rtol*x.norm()) && (stn > atol));
}
```

tentatively reduce damping