1 Iterative Methods for Non-Linear Systems of Equations

A non-linear system of equations is a concept almost *too abstract to be useful*, because it covers an extremely wide variety of problems . Nevertheless in this chapter we will mainly look at "generic" methods for such systems. This means that every method discussed may take a good deal of fine-tuning before it will really perform satisfactorily for a given non-linear system of equations.

Given: function $F : D \subset \mathbb{R}^n \mapsto \mathbb{R}^n$, $n \in \mathbb{N}$

$$\Updownarrow$$

Possible meaning: Availability of a procedure `function y=F(x)` evaluating F

Sought: solution of non-linear equation $F(\mathbf{x}) = 0$

Note: $F : D \subset \mathbb{R}^n \mapsto \mathbb{R}^n$ $\leftrightarrow$ "same number of equations and unknowns"

$$\boxed{\text{In general no existence \& uniqueness of solutions}}$$

Gradinaru
D-MATH

1.1 Iterative methods

Remark 1.1.1 (Necessity of iterative approximation).

Gaussian elimination provides an algorithm that, if carried out in exact arithmetic, computes the solution of a linear system of equations with a *finite* number of elementary operations. However, linear systems of equations represent an exceptional case, because it is hardly ever possible to solve general systems of non-linear equations using only finitely many elementary operations. Certainly this is the case whenever irrational numbers are involved.

△

> An iterative method for (approximately) solving the non-linear equation $F(\mathbf{x}) = 0$ is an algorithm generating a sequence $(\mathbf{x}^{(k)})_{k\in\mathbb{N}_0}$ of approximate solutions.

Initial guess

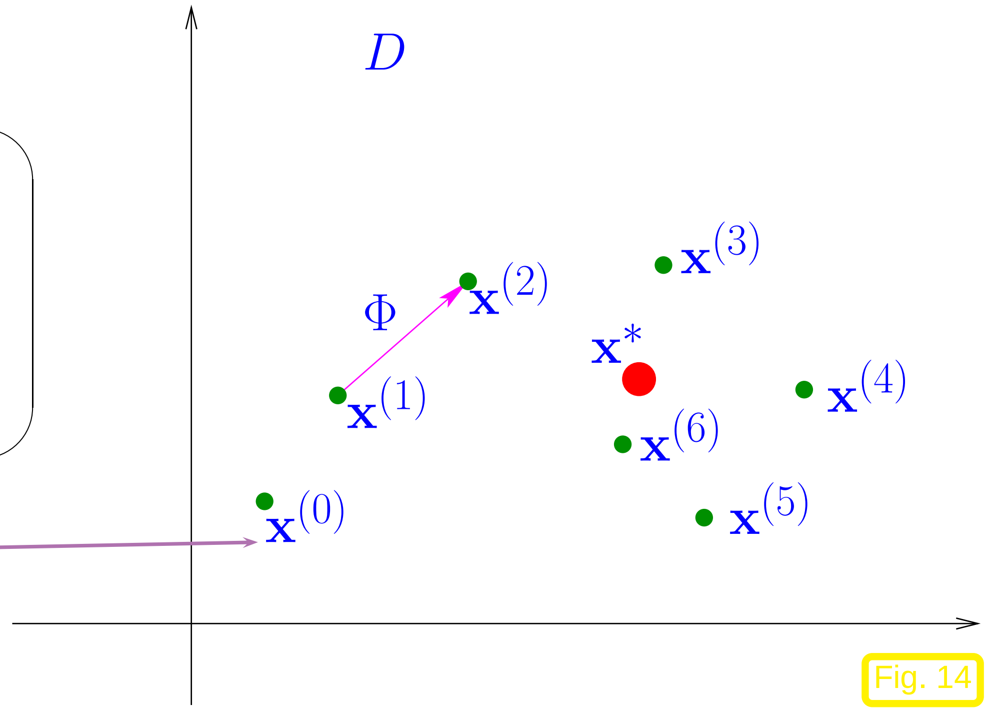


Fig. 14

Fundamental concepts: convergence ➡ speed of convergence

consistency

- iterate $\mathbf{x}^{(k)}$ depends on F and (one or several) $\mathbf{x}^{(n)}$, $n < k$, e.g.,

$$\mathbf{x}^{(k)} = \underbrace{\Phi_F(\mathbf{x}^{(k-1)}, \dots, \mathbf{x}^{(k-m)})}_{\text{iteration function for } m\text{-point method}} \tag{1.1.1}$$

- $\mathbf{x}^{(0)}, \dots, \mathbf{x}^{(m-1)}$ = initial guess(es) (*ger.:* Anfangsnäherung)

Definition 1.1.1 (Convergence of iterative methods)**.**
An iterative method converges (for fixed initial guess(es)) $:\Leftrightarrow$ $\mathbf{x}^{(k)} \to \mathbf{x}^*$ *and* $F(\mathbf{x}^*) = 0$.

Definition 1.1.2 (Consistency of iterative methods)**.**
An iterative method is consistent with $F(\mathbf{x}) = 0$

$$:\Leftrightarrow \quad \Phi_F(\mathbf{x}^*, \dots, \mathbf{x}^*) = \mathbf{x}^* \quad \Leftrightarrow \quad F(\mathbf{x}^*) = 0$$

Terminology: error of iterates $\mathbf{x}^{(k)}$ is defined as: $\mathbf{e}^{(k)} := \mathbf{x}^{(k)} - \mathbf{x}^*$

Definition 1.1.3 (Local and global convergence)**.**
An iterative method converges locally to $\mathbf{x}^* \in \mathbb{R}^n$, if there is a neighborhood $U \subset D$ of $\mathbf{x}^*$, such that

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

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

local convergence ▷

(Only initial guesses “sufficiently close” to $\mathbf{x}^*$ guarantee convergence.)

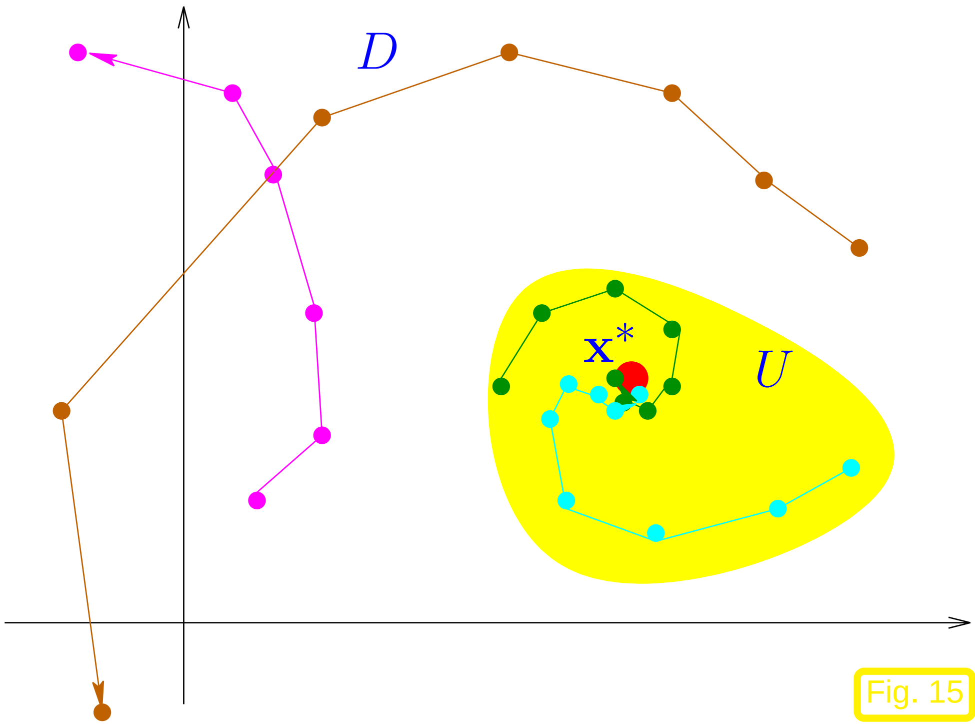


Fig. 15

Goal: Find iterative methods that converge (locally) to a solution of $F(\mathbf{x}) = 0$.

Two general questions: How to measure the speed of convergence?
When to terminate the iteration?

1.1.1 Speed of convergence

Here and in the sequel, $\|\cdot\|$ designates a generic vector norm, see Def. 1.1.9. Any occurring matrix norm is indiuced by this vector norm, see Def. 1.1.12.

It is important to be aware which statements depend on the choice of norm and which do not!

"*Speed of convergence*" $\leftrightarrow$ decrease of norm (see Def. 1.1.9) of iteration error

Definition 1.1.4 (Linear convergence)**.**
A sequence $\mathbf{x}^{(k)}$, $k = 0, 1, 2, \ldots$, *in* $\mathbb{R}^n$ *converges linearly to* $\mathbf{x}^* \in \mathbb{R}^n$, *if*

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

Terminology: least upper bound for L gives the rate of convergence

Remark 1.1.2 (Impact of choice of norm)*.*

Fact of convergence of iteration is	independent	of choice of norm
Fact of linear convergence	depends	on choice of norm
Rate of linear convergence	depends	on choice of norm

]

Norms provide tools for measuring errors. Recall from linear algebra and calculus:

Definition 1.1.9 (Norm).
X = vector space over field $\mathbb{K}$, $\mathbb{K} = \mathbb{C}, \mathbb{R}$. A map $\|\cdot\| : X \mapsto \mathbb{R}_0^+$ is a norm on X, if it satisfies

(i) $\forall \mathbf{x} \in X$: $\mathbf{x} \neq 0 \quad \Leftrightarrow \quad \|\mathbf{x}\| > 0$ *(definite),*

(ii) $\|\lambda \mathbf{x}\| = |\lambda| \|\mathbf{x}\| \quad \forall \mathbf{x} \in X, \lambda \in \mathbb{K}$ *(homogeneous),*

(iii) $\|\mathbf{x} + \mathbf{y}\| \leq \|\mathbf{x}\| + \|\mathbf{y}\| \quad \forall \mathbf{x}, \mathbf{y} \in X$ *(triangle inequality).*

Examples: (for vector space $\mathbb{K}^n$, vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)^T \in \mathbb{K}^n$)

name	:	definition	`numpy.linalg` function
Euclidean norm	:	$\|\mathbf{x}\|_2 := \sqrt{|x_1|^2 + \cdots + |x_n|^2}$	`norm(x)`
1-norm	:	$\|\mathbf{x}\|_1 := |x_1| + \cdots + |x_n|$	`norm(x,1)`
∞-norm, max norm	:	$\|\mathbf{x}\|_\infty := \max\{|x_1|, \ldots, |x_n|\}$	`norm(x,inf)`

Recall: equivalence of all norms on finite dimensional vector space $\mathbb{K}^n$:

Definition 1.1.10 (Equivalence of norms).
Two norms $\|\cdot\|_1$ and $\|\cdot\|_2$ on a vector space V are equivalent if

$$\exists \underline{C}, \overline{C} > 0: \quad \underline{C} \|v\|_1 \leq \|v\|_2 \leq \overline{C} \|v\|_2 \quad \forall v \in V\,.$$

Theorem 1.1.11 (Equivalence of all norms on finite dimensional vector spaces).

If $\dim V < \infty$ all norms (→ Def. 1.1.9) on V are equivalent (→ Def. 1.1.10).

$\triangle$

Simple explicit norm equivalences: for all $\mathbf{x} \in \mathbb{K}^n$

$$\|\mathbf{x}\|_2 \leq \|\mathbf{x}\|_1 \leq \sqrt{n}\,\|\mathbf{x}\|_2 \;, \tag{1.1.7}$$

$$\|\mathbf{x}\|_\infty \leq \|\mathbf{x}\|_2 \leq \sqrt{n}\,\|\mathbf{x}\|_\infty \;, \tag{1.1.8}$$

$$\|\mathbf{x}\|_\infty \leq \|\mathbf{x}\|_1 \leq n\,\|\mathbf{x}\|_\infty \;. \tag{1.1.9}$$

Definition 1.1.12 (Matrix norm)**.**

Given a vector norm $\|\cdot\|$ on $\mathbb{R}^n$, the associated *matrix norm* *is defined by*

$$\mathbf{M} \in \mathbb{R}^{m,n}: \quad \|\mathbf{M}\| := \sup_{\mathbf{x}\in\mathbb{R}^n\setminus\{0\}} \frac{\|\mathbf{M}\mathbf{x}\|}{\|\mathbf{x}\|} \;.$$

Gradinaru
D-MATH

▶ sub-multiplicative: $\mathbf{A} \in \mathbb{K}^{n,m}, \mathbf{B} \in \mathbb{K}^{m,k}$: $\|\mathbf{AB}\| \leq \|\mathbf{A}\| \, \|\mathbf{B}\|$

✎ notation: $\|\mathbf{x}\|_2 \to \|\mathbf{M}\|_2$, $\|\mathbf{x}\|_1 \to \|\mathbf{M}\|_1$, $\|\mathbf{x}\|_\infty \to \|\mathbf{M}\|_\infty$

Example 1.1.4 (Matrix norm associated with ∞-norm and 1-norm).

e.g. for $\mathbf{M} \in \mathbb{K}^{2,2}$:

$$\begin{aligned}
\|\mathbf{Mx}\|_\infty &= \max\{|m_{11}x_1 + m_{12}x_2|, |m_{21}x_1 + m_{22}x_2|\} \\
&\leq \max\{|m_{11}| + |m_{12}|, |m_{21}| + |m_{22}|\} \; \|x\|_\infty \;, \\
\|\mathbf{Mx}\|_1 &= |m_{11}x_1 + m_{12}x_2| + |m_{21}x_1 + m_{22}x_2| \\
&\leq \max\{|m_{11}| + |m_{21}|, |m_{12}| + |m_{22}|\}(|x_1| + |x_2|) \;.
\end{aligned}$$

For general $\mathbf{M} \in \mathbb{K}^{m,n}$

➢ matrix norm $\leftrightarrow$ $\|\cdot\|_\infty$ = row sum norm
$$\|\mathbf{M}\|_\infty := \max_{i=1,\dots,m} \sum_{j=1}^{n} |m_{ij}| \;, \tag{1.1.10}$$

➢ matrix norm $\leftrightarrow$ $\|\cdot\|_1$ = column sum norm
$$\|\mathbf{M}\|_1 := \max_{j=1,\dots,n} \sum_{i=1}^{m} |m_{ij}| \;. \tag{1.1.11}$$

Remark 1.1.5 (Seeing linear convergence).

norms of iteration errors

$\updownarrow$

$\sim$ straight line in lin-log plot

$$\left\|\mathbf{e}^{(k)}\right\| \leq L^k \left\|\mathbf{e}^{(0)}\right\| \; ,$$

$$\log(\left\|\mathbf{e}^{(k)}\right\|) \leq k \log(L) + \log(\left\|\mathbf{e}^{(0)}\right\|) \; .$$

(●: Any "faster" convergence also qualifies as linear !)

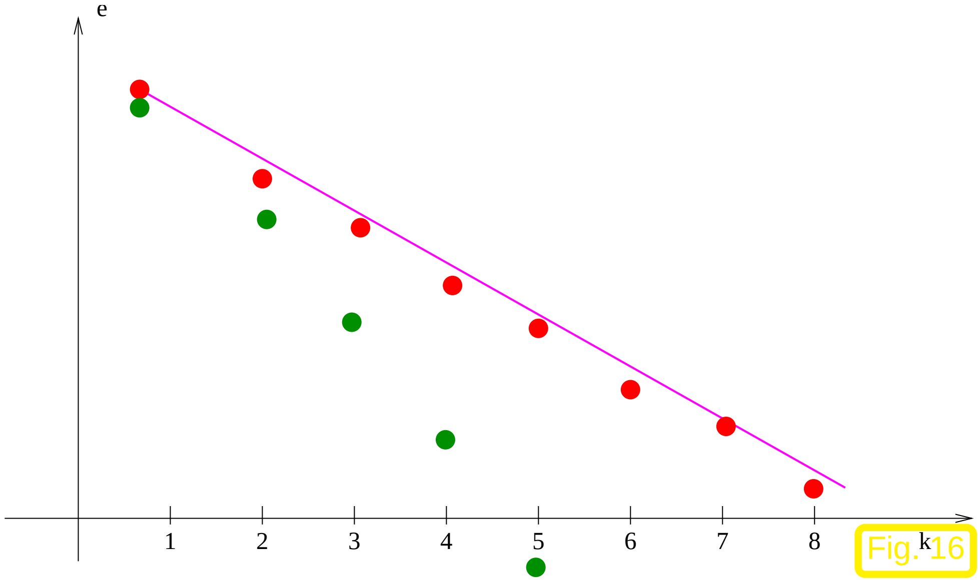


Fig. 16

Let us abbreviate the error norm in step k by $\epsilon_k := \left\|\mathbf{x}^{(k)} - \mathbf{x}^*\right\|$. In the case of linear convergence (see Def. 1.1.4) assume (with $0 < L < 1$)

$$\epsilon_{k+1} \approx L\epsilon_k \quad \Rightarrow \quad \log \epsilon_{k+1} \approx \log L + \log \epsilon_k \quad \Rightarrow \quad \log \epsilon_{k+1} \approx k \log L + \log \epsilon_0 \; . \tag{1.1.12}$$

We conclude that $\log L < 0$ describes slope of graph in lin-log error chart.

$\triangle$

Example 1.1.6 (Linearly convergent iteration).

Iteration (dimension $n = 1$):

$$x^{(k+1)} = x^{(k)} + \frac{\cos x^{(k)} + 1}{\sin x^{(k)}} .$$

Code 1.1.7: simple fixed point iteration

```
def lincvg(x):
    y = []
    for k in xrange(15):
        x = x +(cos(x)+1)/sin(x)
        y += [x]
    err = array(y) - x
    rate = err[1:]/err[:-1]
    return err, rate
```

Note: $x^{(15)}$ replaces the exact solution x^* in the computation of the rate of convergence.

k	$x^{(0)} = 0.4$		$x^{(0)} = 0.6$		$x^{(0)} = 1$	
	$x^{(k)}$	$\frac{\lvert x^{(k)}-x^{(15)}\rvert}{\lvert x^{(k-1)}-x^{(15)}\rvert}$	$x^{(k)}$	$\frac{\lvert x^{(k)}-x^{(15)}\rvert}{\lvert x^{(k-1)}-x^{(15)}\rvert}$	$x^{(k)}$	$\frac{\lvert x^{(k)}-x^{(15)}\rvert}{\lvert x^{(k-1)}-x^{(15)}\rvert}$
2	3.3887	0.1128	3.4727	0.4791	2.9873	0.4959
3	3.2645	0.4974	3.3056	0.4953	3.0646	0.4989
4	3.2030	0.4992	3.2234	0.4988	3.1031	0.4996
5	3.1723	0.4996	3.1825	0.4995	3.1224	0.4997
6	3.1569	0.4995	3.1620	0.4994	3.1320	0.4995
7	3.1493	0.4990	3.1518	0.4990	3.1368	0.4990
8	3.1454	0.4980	3.1467	0.4980	3.1392	0.4980

Rate of convergence ≈ 0.5

Linear convergence as in Def. 1.1.4

$\Updownarrow$

error graphs $=$ straight lines in lin-log scale

$\longrightarrow$ Rem. 1.1.5

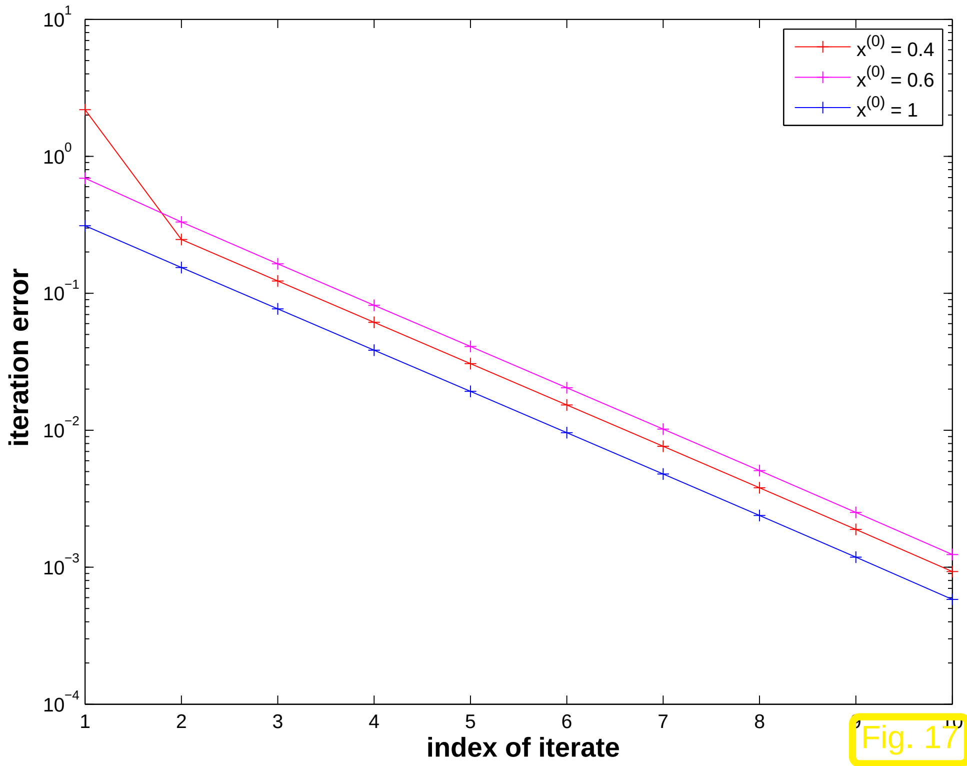


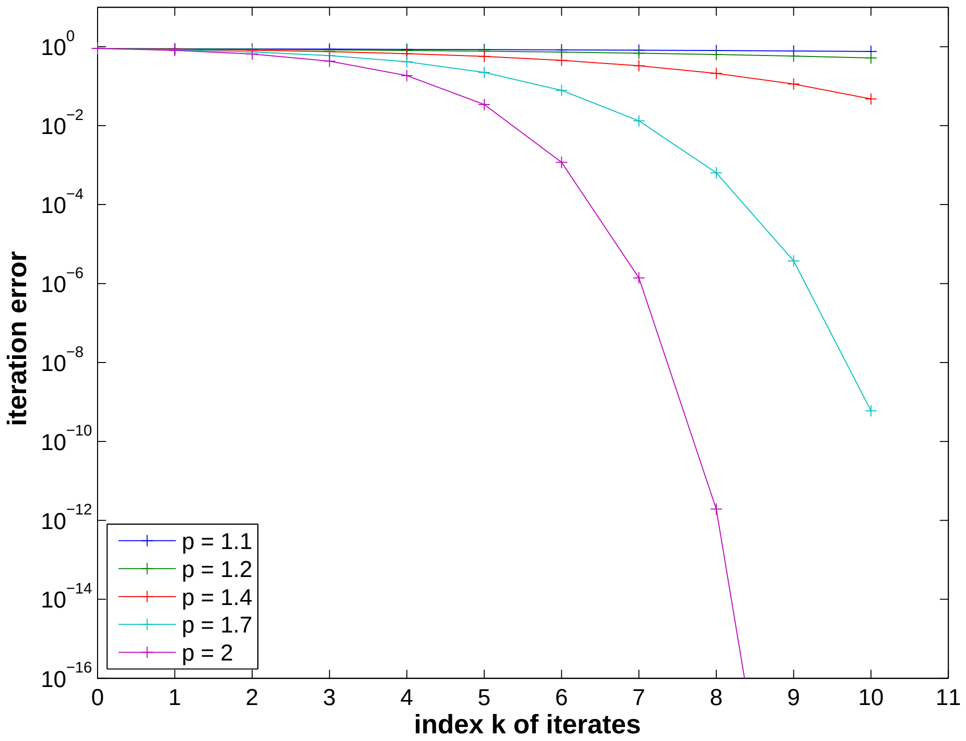
Fig. 17

$\Diamond$

Definition 1.1.13 (Order of convergence)**.**
*A **convergent** sequence $\mathbf{x}^{(k)}$, $k = 0, 1, 2, \ldots$, in $\mathbb{R}^n$ converges with order p to $\mathbf{x}^* \in \mathbb{R}^n$, if*

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

with $C < 1$ for $p = 1$ (linear convergence $\rightarrow$ Def. 1.1.4)



◁ Qualitative error graphs for convergence of order p
(lin-log scale)

In the case of convergence of order p ($p > 1$) (see Def. 1.1.13):

$$\epsilon_{k+1} \approx C\epsilon_k^p \quad \Rightarrow \quad \log\epsilon_{k+1} = \log C + p\log\epsilon_k \quad \Rightarrow \quad \log\epsilon_{k+1} = \log C\sum_{l=0}^{k} p^l + p^{k+1}\log\epsilon_0$$

$$\Rightarrow \quad \log\epsilon_{k+1} = -\frac{\log C}{p-1} + \left(\frac{\log C}{p-1} + \log\epsilon_0\right) p^{k+1} \;.$$

In this case, the error graph is a concave power curve (for sufficiently small ϵ_0 !)

Example 1.1.8 (quadratic convergence). (= convergence of order 2)

Iteration for computing $\sqrt{a}$, $a > 0$:

$$x^{(k+1)} = \frac{1}{2}(x^{(k)} + \frac{a}{x^{(k)}}) \quad \Rightarrow \quad |x^{(k+1)} - \sqrt{a}| = \frac{1}{2x^{(k)}}|x^{(k)} - \sqrt{a}|^2 \,. \tag{1.1.13}$$

By the arithmetic-geometric mean inequality (AGM) $\sqrt{ab} \leq \frac{1}{2}(a+b)$ we conclude: $x^{(k)} > \sqrt{a}$ for $k \geq 1$.

$\Rightarrow$ sequence from (1.1.13) converges with order 2 to $\sqrt{a}$

Note: $x^{(k+1)} < x^{(k)}$ for all $k \geq 2$ ➢ $(x^{(k)})_{k\in\mathbb{N}_0}$ converges as a decreasing sequence that is bounded from below ($\longrightarrow$ analysis course)

How to guess the order of convergence in a numerical experiment?
Abbreviate $\epsilon_k := \left\| \mathbf{x}^{(k)} - \mathbf{x}^* \right\|$ and then

$$\epsilon_{k+1} \approx C\epsilon_k^p \quad \Rightarrow \quad \log \epsilon_{k+1} \approx \log C + p \log \epsilon_k \quad \Rightarrow \quad \frac{\log \epsilon_{k+1} - \log \epsilon_k}{\log \epsilon_k - \log \epsilon_{k-1}} \approx p \,.$$

Numerical experiment: iterates for $a = 2$:

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 !]

The doubling of the number of significant digits for the iterates holds true for any convergent second-order iteration:

Indeed, denoting the relative error in step k by δ_k, we have:

$$x^{(k)} = x^*(1+\delta_k) \quad \Rightarrow \quad x^{(k)} - x^* = \delta_k x^* \ .$$
$$\Rightarrow |x^*\delta_{k+1}| = |x^{(k+1)} - x^*| \le C|x^{(k)} - x^*|^2 = C|x^*\delta_k|^2$$
$$\Rightarrow \qquad |\delta_{k+1}| \le C|x^*|\delta_k^2 \ . \tag{1.1.14}$$

Note: $\delta_k \approx 10^{-\ell}$ means that $\mathbf{x}^{(k)}$ has ℓ significant digits.

Also note that if $C \approx 1$, then $\delta_k = 10^{-\ell}$ and (1.1.8) implies $\delta_{k+1} \approx 10^{-2\ell}$.

1.1.2 Termination criteria

Usually (even without roundoff errors) the iteration will never arrive at an/the exact solution $\mathbf{x}^*$ after finitely many steps. Thus, we can only hope to compute an *approximate* solution by accepting $\mathbf{x}^{(K)}$ as result for some $K \in \mathbb{N}_0$. Termination criteria (*ger.:* Abbruchbedingungen) are used to determine a suitable value for K.

For the sake of efficiency: $\triangleright$ stop iteration when iteration error is just "small enough"

"small enough" depends on concrete setting:

Usual goal: $\left\|\mathbf{x}^{(K)} - \mathbf{x}^*\right\| \leq \tau$, $\tau \hat{=}$ prescribed tolerance.

$$\text{Ideal:}\quad K = \operatorname{argmin}\{k \in \mathbb{N}_0 \colon \left\|\mathbf{x}^{(k)} - \mathbf{x}^*\right\| < \tau\} \,. \tag{1.1.15}$$

① A priori termination: stop iteration after fixed number of steps (possibly depending on $\mathbf{x}^{(0)}$).

Drawback: hardly ever possible !

Alternative: A posteriori termination criteria

use already computed iterates to decide when to stop

②

Reliable termination: stop iteration $\{\mathbf{x}^{(k)}\}_{k\in\mathbb{N}_0}$ with limit $\mathbf{x}^*$, when

$$\left\| \mathbf{x}^{(k)} - \mathbf{x}^* \right\| \leq \tau \,, \qquad \tau \hat{=} \text{ prescribed tolerance } > 0 \,. \tag{1.1.16}$$

$\mathbf{x}^*$ not known !

Invoking additional properties of either the non-linear system of equations $F(\mathbf{x}) = 0$ or the iteration it is sometimes possible to tell that for sure $\left\| \mathbf{x}^{(k)} - \mathbf{x}^* \right\| \leq \tau$ for all $k \geq K$, though this K may be (significantly) larger than the optimal termination index from (1.1.15), see Rem. 1.1.10.

Code 1.1.9: stationary iteration

```
from numpy import sqrt, array
def sqrtit(a,x):
    exact = sqrt(a)
    e = [x]
    x_old = -1.
    while x_old != x:
        x_old = x
        x = 0.5*(x+a/x)
        e += [x]
    e = array(e)
    e = abs(e-exact)
    return e

e = sqrtit(2.,1.)
print e
```

③ use that $\mathbb{M}$ is finite! ($\longrightarrow$ Sect. **??**)

➢ possible to wait until (convergent) iteration becomes stationary

possibly grossly inefficient ! (always computes "up to machine precision")

④ Residual based termination: stop convergent iteration $\{\mathbf{x}^{(k)}\}_{k\in\mathbb{N}_0}$, when

$$\left\|F(\mathbf{x}^{(k)})\right\| \leq \tau\ , \qquad \tau \hat{=} \text{ prescribed tolerance } > 0\ .$$

no guaranteed accuracy

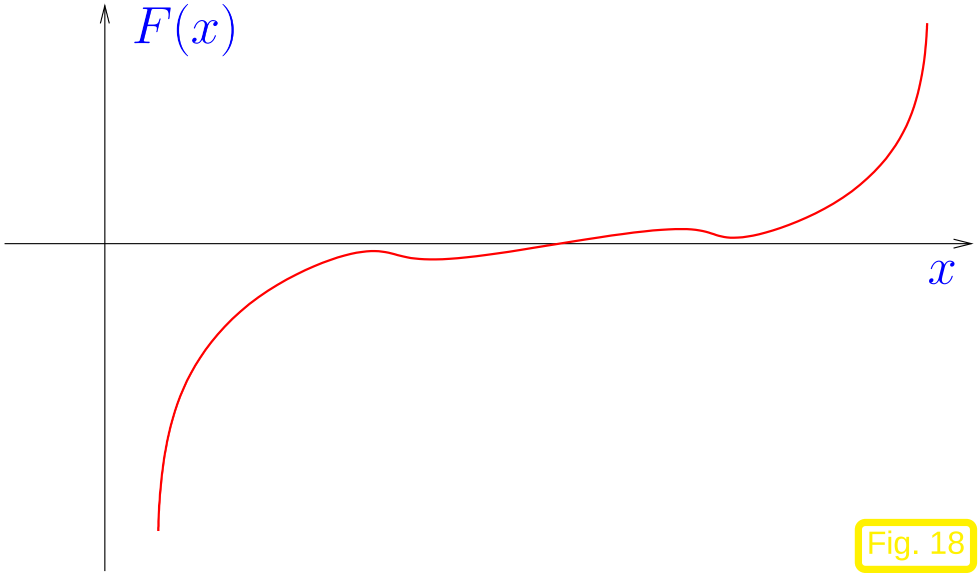


Fig. 18

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

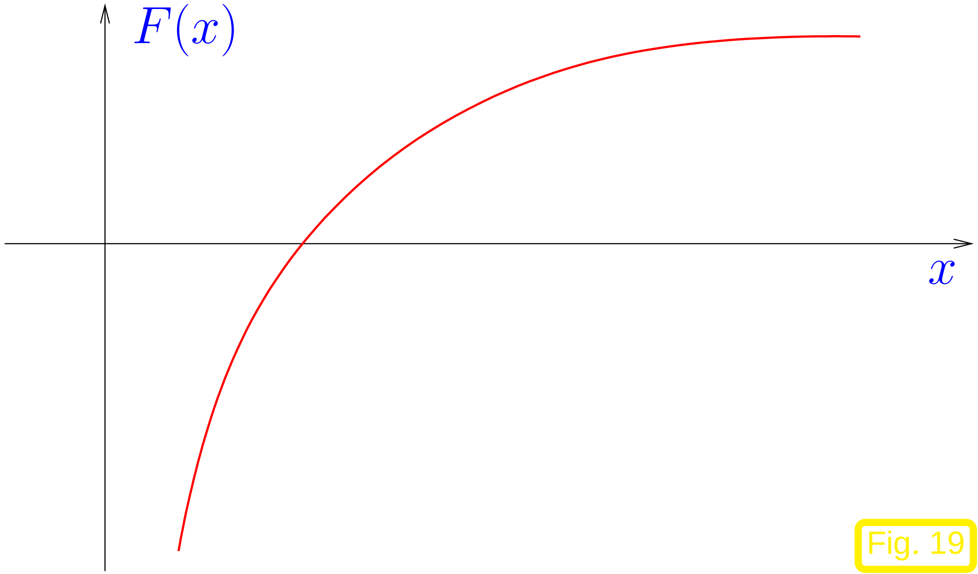


Fig. 19

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

Sometimes extra knowledge about the type/speed of convergence allows to achieve reliable termination in the sense that (1.1.16) can be guaranteed though the number of iterations might be (slightly) too large.

Remark 1.1.10 (A posteriori termination criterion for linearly convergent iterations).

Known: iteration linearly convergent with rate of convergence $0 < L < 1$:

Derivation of a posteriori termination criterion for linearly convergent iterations with rate of convergence $0 < L < 1$:

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

Iterates satisfy: $$\boxed{\left\|\mathbf{x}^{(k+1)} - \mathbf{x}^*\right\| \leq \frac{L}{1-L}\left\|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\right\|} . \qquad (1.1.17)$$

This suggests that we take the right hand side of (1.1.17) as a posteriori error bound.

△

Example 1.1.11 (A posteriori error bound for linearly convergent iteration).

Iteration of Example 1.1.6:

$$x^{(k+1)} = x^{(k)} + \frac{\cos x^{(k)} + 1}{\sin x^{(k)}} \quad \Rightarrow x^{(k)} \to \pi \quad \text{for } x^{(0)} \text{ close to } \pi .$$

Observed rate of convergence: $L = 1/2$

Error and error bound for $x^{(0)} = 0.4$:

k	$\lvert x^{(k)} - \pi \rvert$	$\frac{L}{1-L}\lvert x^{(k)} - x^{(k-1)} \rvert$	slack of bound
1	2.191562221997101	4.933154875586894	2.741592653589793
2	0.247139097781070	1.944423124216031	1.697284026434961
3	0.122936737876834	0.124202359904236	0.001265622027401
4	0.061390835206217	0.061545902670618	0.000155067464401
5	0.030685773472263	0.030705061733954	0.000019288261691
6	0.015341682696235	0.015344090776028	0.000002408079792
7	0.007670690889185	0.007670991807050	0.000000300917864
8	0.003835326638666	0.003835364250520	0.000000037611854
9	0.001917660968637	0.001917665670029	0.000000004701392
10	0.000958830190489	0.000958830778147	0.000000000587658
11	0.000479415058549	0.000479415131941	0.000000000073392
12	0.000239707524646	0.000239707533903	0.000000000009257
13	0.000119853761949	0.000119853762696	0.000000000000747
14	0.000059926881308	0.000059926880641	0.000000000000667
15	0.000029963440745	0.000029963440563	0.000000000000181

Hence: the a posteriori error bound is highly accurate in this case! ◇

Note: If L not known then using $\widetilde{L} > L$ in error bound is playing safe.

1.2 Fixed Point Iterations

Non-linear system of equations $F(\mathbf{x}) = 0$, $F : D \subset \mathbb{R}^n \mapsto \mathbb{R}^n$,

A fixed point iteration is defined by iteration function $\Phi : U \subset \mathbb{R}^n \mapsto \mathbb{R}^n$:

iteration function $\Phi : U \subset \mathbb{R}^n \mapsto \mathbb{R}^n$
initial guess $\mathbf{x}^{(0)} \in U$

$$\succ \underbrace{\boxed{\text{iterates } (\mathbf{x}^{(k)})_{k\in\mathbb{N}_0}:\quad \mathbf{x}^{(k+1)} := \Phi(\mathbf{x}^{(k)})}}_{\to \text{ 1-point method, } cf. \text{ (1.1.1)}} .$$

Sequence of iterates need not be well defined: $\mathbf{x}^{(k)} \notin U$ possible !

1.2.1 Consistent fixed point iterations

Definition 1.2.1 (Consistency of fixed point iterations, *c.f.* Def. 1.1.2)**.**
A fixed point iteration $\mathbf{x}^{(k+1)} = \Phi(\mathbf{x}^{(k)})$ *is consistent with* $F(\mathbf{x}) = 0$, *if*

$$F(\mathbf{x}) = 0 \quad \textit{and} \quad x \in U \cap D \quad \Leftrightarrow \quad \Phi(\mathbf{x}) = \mathbf{x} \ .$$

Note: Φ continuous **&** fixed point iteration (locally) convergent to $\mathbf{x}^*$ **then** $\mathbf{x}^*$ is fixed point of iteration function Φ.

General construction of fixed point iterations that is consistent with $F(\mathbf{x}) = 0$:
rewrite $F(\mathbf{x}) = 0 \quad \Leftrightarrow \quad \Phi(\mathbf{x}) = \mathbf{x}$ and then

$$\text{use the fixed point iteration} \quad \mathbf{x}^{(k+1)} := \Phi(\mathbf{x}^{(k)}) \ . \tag{1.2.1}$$

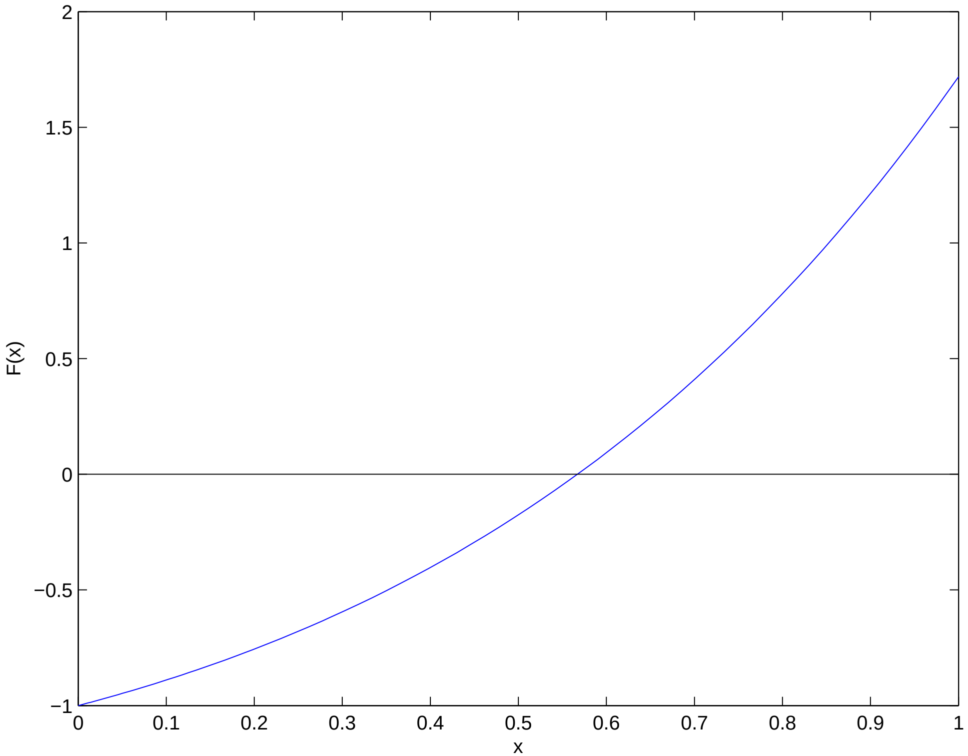
Note: there are *many ways* to transform $F(\mathbf{x}) = 0$ into a fixed point form **!**

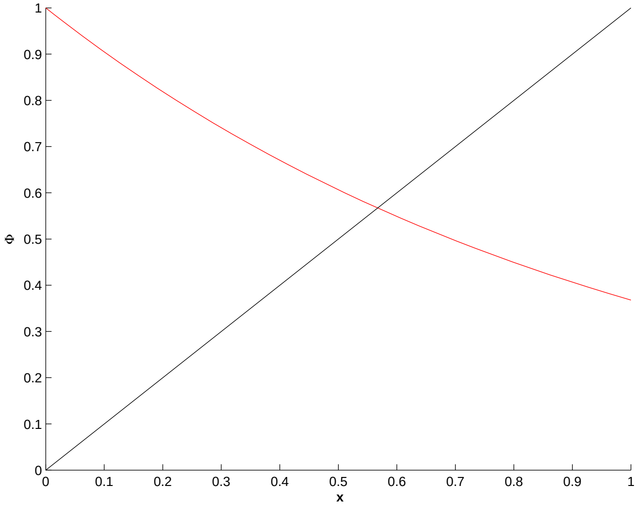
Example 1.2.1 (Options for fixed point iterations).

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

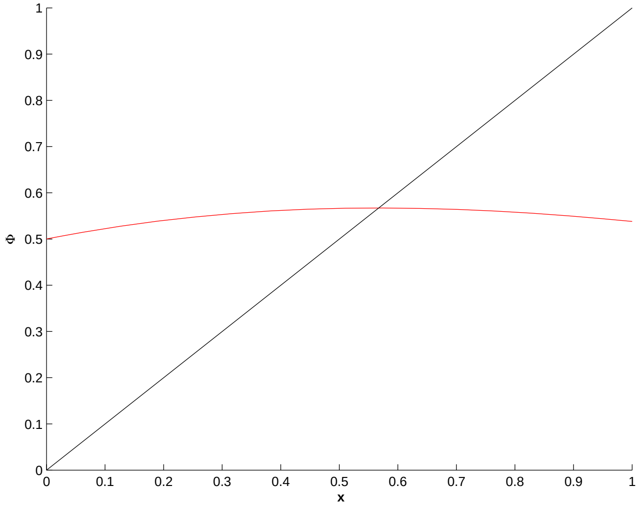
Different fixed point forms:

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

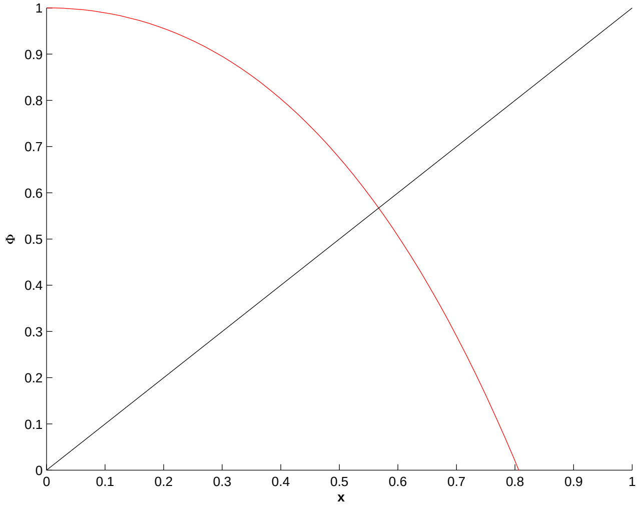




function Φ_1



function Φ_2



function Φ_3

k	$x^{(k+1)} := \Phi_1(x^{(k)})$	$x^{(k+1)} := \Phi_2(x^{(k)})$	$x^{(k+1)} := \Phi_3(x^{(k)})$
0	0.500000000000000	0.500000000000000	0.500000000000000
1	0.606530659712633	0.566311003197218	0.675639364649936
2	0.545239211892605	0.567143165034862	0.347812678511202
3	0.579703094878068	0.567143290409781	0.855321409174107
4	0.560064627938902	0.567143290409784	-0.156505955383169
5	0.571172148977215	0.567143290409784	0.977326422747719
6	0.564862946980323	0.567143290409784	-0.619764251895580
7	0.568438047570066	0.567143290409784	0.713713087416146
8	0.566409452746921	0.567143290409784	0.256626649129847
9	0.567559634262242	0.567143290409784	0.924920676910549
10	0.566907212935471	0.567143290409784	-0.407422405542253

k	$\lvert x_1^{(k+1)} - x^*\rvert$	$\lvert x_2^{(k+1)} - x^*\rvert$	$\lvert x_3^{(k+1)} - x^*\rvert$
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

Observed: linear convergence of $x_1^{(k)}$, quadratic convergence of $x_2^{(k)}$,
no convergence (erratic behavior) of $x_3^{(k)}$), $x_i^{(0)} = 0.5$. ◇

Question: can we explain/forecast the behaviour of the iteration?

1.2.2 Convergence of fixed point iterations

In this section we will try to find easily verifiable conditions that ensure convergence (of a certain order) of fixed point iterations. It will turn out that these conditions are surprisingly simple and general.

Definition 1.2.2 (Contractive mapping)**.**
$\Phi : U \subset \mathbb{R}^n \mapsto \mathbb{R}^n$ *is contractive (w.r.t. norm* $\|\cdot\|$ *on* $\mathbb{R}^n$*), if*

$$\exists L < 1 : \quad \|\Phi(\mathbf{x}) - \Phi(\mathbf{y})\| \leq L \|\mathbf{x} - \mathbf{y}\| \quad \forall \mathbf{x}, \mathbf{y} \in U \; . \tag{1.2.2}$$

A simple consideration: if $\Phi(\mathbf{x}^*) = \mathbf{x}^*$ (fixed point), then a fixed point iteration induced by a contractive mapping Φ satisfies

$$\left\| \mathbf{x}^{(k+1)} - \mathbf{x}^* \right\| = \left\| \Phi(\mathbf{x}^{(k)}) - \Phi(\mathbf{x}^*) \right\| \overset{(1.2.3)}{\leq} L \left\| \mathbf{x}^{(k)} - \mathbf{x}^* \right\| ,$$

that is, the iteration converges (at least) linearly ($\rightarrow$ Def. 1.1.4).

Remark 1.2.2 (Banach's fixed point theorem)*.* $\rightarrow$ [**?**, Satz 6.5.2]

A key theorem in calculus (also functional analysis):

Theorem 1.2.3 (Banach's fixed point theorem).
If $D \subset \mathbb{K}^n$ ($\mathbb{K} = \mathbb{R}, \mathbb{C}$) closed and $\Phi : D \mapsto D$ satisfies

$$\exists L < 1: \quad \|\Phi(\mathbf{x}) - \Phi(\mathbf{y})\| \leq L \|\mathbf{x} - \mathbf{y}\| \quad \forall \mathbf{x}, \mathbf{y} \in D ,$$

then there is a unique fixed point $\mathbf{x}^* \in D$, $\Phi(\mathbf{x}^*) = \mathbf{x}^*$, which is the limit of the sequence of iterates $\mathbf{x}^{(k+1)} := \Phi(x^{(k)})$ for any $\mathbf{x}^{(0)} \in D$.

Proof. Proof based on 1-point iteration $\mathbf{x}^{(k)} = \Phi(\mathbf{x}^{(k-1)})$, $\mathbf{x}^{(0)} \in D$:

$$\begin{aligned} \left\|\mathbf{x}^{(k+N)} - \mathbf{x}^{(k)}\right\| &\leq \sum_{j=k}^{k+N-1} \left\|\mathbf{x}^{(j+1)} - \mathbf{x}^{(j)}\right\| \leq \sum_{j=k}^{k+N-1} L^j \left\|\mathbf{x}^{(1)} - \mathbf{x}^{(0)}\right\| \\ &\leq \frac{L^k}{1-L} \left\|\mathbf{x}^{(1)} - \mathbf{x}^{(0)}\right\| \xrightarrow{k\to\infty} 0 . \end{aligned}$$

$(\mathbf{x}^{(k)})_{k\in\mathbb{N}_0}$ Cauchy sequence ➤ convergent $\mathbf{x}^{(k)} \xrightarrow{k\to\infty} \mathbf{x}^*$.
Continuity of Φ ➤ $\Phi(\mathbf{x}^*) = \mathbf{x}^*$. Uniqueness of fixed point is evident. □

△

A simple criterion for a differentiable Φ to be contractive:

Gradinaru
D-MATH

Lemma 1.2.4 (Sufficient condition for local linear convergence of fixed point iteration)**.**
If $\Phi : U \subset \mathbb{R}^n \mapsto \mathbb{R}^n$, $\Phi(\mathbf{x}^*) = \mathbf{x}^*$,$\Phi$ *differentiable in* $\mathbf{x}^*$, *and* $\|D\Phi(\mathbf{x}^*)\| < 1$, *then the fixed point iteration* (1.2.1) *converges locally and at least linearly.*

matrix norm, Def. 1.1.12 !

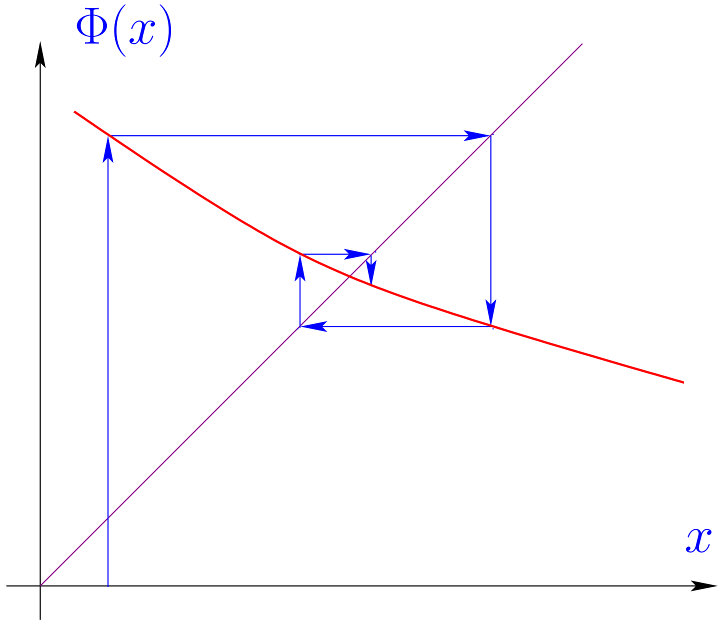
✎ notation: $D\Phi(\mathbf{x}) \hat{=}$ Jacobian (*ger.:* Jacobi-Matrix) of Φ at $\mathbf{x} \in D$

Example 1.2.3 (Fixed point iteration in 1D)*.*

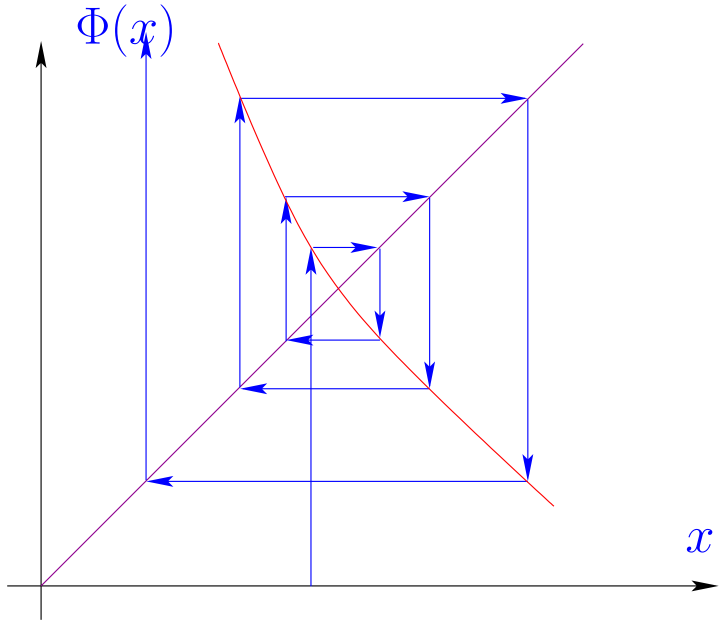
1D setting ($n = 1$): $\Phi : \mathbb{R} \mapsto \mathbb{R}$ continuously differentiable, $\Phi(x^*) = x^*$

fixed point iteration: $x^{(k+1)} = \Phi(x^{(k)})$

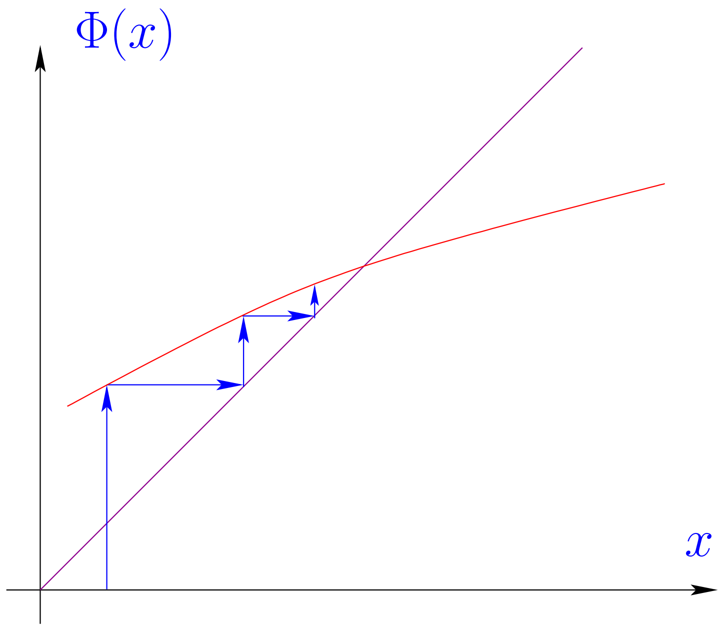
"Visualization" of the statement of Lemma 1.2.4: The iteration converges *locally*, if Φ is flat in a neighborhood of x^*, it will diverge, if Φ is steep there.



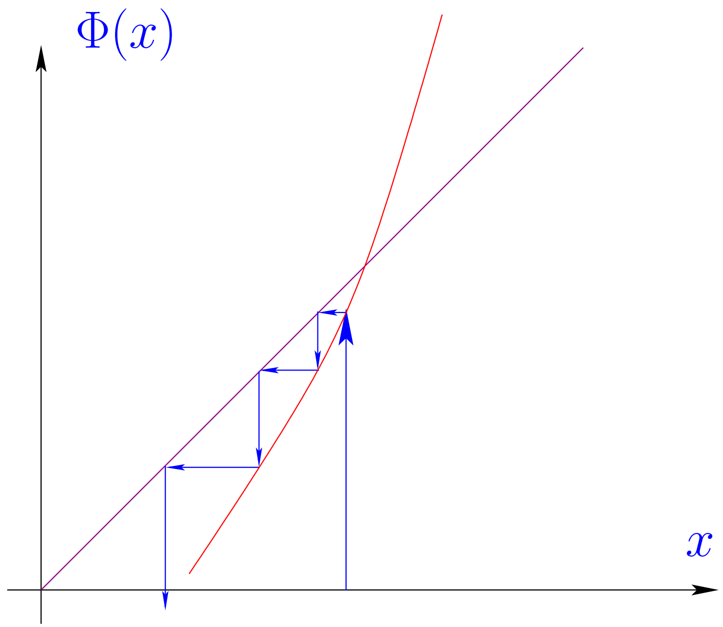
$-1 < \Phi'(x^*) \leq 0$ ➢ convergence



$\Phi'(x^*) < -1$ ➢ divergence



$0 \leq \Phi'(x^*) < 1$ ➢ convergence



$1 < \Phi'(x^*)$ ➢ divergence

$\diamond$

Proof. (of Lemma 1.2.4) By definition of derivative

$$\|\Phi(\mathbf{y}) - \Phi(\mathbf{x}^*) - D\Phi(\mathbf{x}^*)(\mathbf{y} - \mathbf{x}^*)\| \leq \psi(\|\mathbf{y} - \mathbf{x}^*\|)\, \|\mathbf{y} - \mathbf{x}^*\| \ ,$$

with $\psi : \mathbb{R}_0^+ \mapsto \mathbb{R}_0^+$ satisfying $\lim\limits_{t\to 0} \psi(t) = 0$.

Choose $\delta > 0$ such that

$$L := \psi(t) + \|D\Phi(\mathbf{x}^*)\| \leq \tfrac{1}{2}(1 + \|D\Phi(\mathbf{x}^*)\|) < 1 \quad \forall 0 \leq t < \delta \ .$$

By inverse triangle inequality we obtain for fixed point iteration

$$\|\Phi(\mathbf{x}) - \mathbf{x}^*\| - \|D\Phi(\mathbf{x}^*)(\mathbf{x} - \mathbf{x}^*)\| \leq \psi(\|\mathbf{x} - \mathbf{x}^*\|)\, \|\mathbf{x} - \mathbf{x}^*\|$$

$$\blacktriangleright \quad \left\|\mathbf{x}^{(k+1)} - \mathbf{x}^*\right\| \leq (\psi(t) + \|D\Phi(\mathbf{x}^*)\|) \left\|\mathbf{x}^{(k)} - \mathbf{x}^*\right\| \leq L \left\|\mathbf{x}^{(k)} - \mathbf{x}^*\right\| \ ,$$

if $\left\|\mathbf{x}^{(k)} - \mathbf{x}^*\right\| < \delta$. $\square$

Contractivity also guarantees the *uniqueness* of a fixed point, see the next Lemma.

Recalling the Banach fixed point theorem Thm. 1.2.3 we see that under some additional (usually mild) assumptions, it also ensures the *existence* of a fixed point.

Lemma 1.2.5 (Sufficient condition for local linear convergence of fixed point iteration (II))**.**
Let U *be convex and* $\Phi : U \subset \mathbb{R}^n \mapsto \mathbb{R}^n$ *continuously differentiable with* $L := \sup_{\mathbf{x}\in U} \|D\Phi(\mathbf{x})\| < 1$*. If* $\Phi(\mathbf{x}^*) = \mathbf{x}^*$ *for some interior point* $\mathbf{x}^* \in U$*, then the fixed point iteration* $\mathbf{x}^{(k+1)} = \Phi(\mathbf{x}^{(k)})$ *converges to* $\mathbf{x}^*$ *locally at least linearly.*

Recall: $U \subset \mathbb{R}^n$ convex $:\Leftrightarrow$ $(t\mathbf{x} + (1-t)\mathbf{y}) \in U$ for all $\mathbf{x}, \mathbf{y} \in U$, $0 \le t \le 1$

Proof. (of Lemma 1.2.5) By the mean value theorem

$$\Phi(\mathbf{x}) - \Phi(\mathbf{y}) = \int_0^1 D\Phi(\mathbf{x} + \tau(\mathbf{y}-\mathbf{x}))(\mathbf{y}-\mathbf{x})\,\mathrm{d}\tau \quad \forall \mathbf{x}, \mathbf{y} \in \mathrm{dom}(\Phi) .$$

$$\blacktriangleright \qquad \|\Phi(\mathbf{x}) - \Phi(\mathbf{y})\| \le L\,\|\mathbf{y}-\mathbf{x}\| .$$

There is $\delta > 0$: $B := \{\mathbf{x}\colon \|\mathbf{x}-\mathbf{x}^*\| \le \delta\} \subset \mathrm{dom}(\Phi)$. Φ is contractive on B with unique fixed point $\mathbf{x}^*$.

Remark 1.2.4.
If $0 < \|D\Phi(\mathbf{x}^*)\| < 1$, $\mathbf{x}^{(k)} \approx \mathbf{x}^*$ then the asymptotic rate of linear convergence is $L = \|D\Phi(x^*)\|$

$\triangle$

Example 1.2.5 (Multidimensional fixed point iteration).

System of equations in fixed point form:

$$\left\{\begin{aligned} x_1 - c(\cos x_1 - \sin x_2) &= 0 \\ (x_1 - x_2) - c \sin x_2 &= 0 \end{aligned}\right. \quad \Rightarrow \quad \left\{\begin{aligned} c(\cos x_1 - \sin x_2) &= x_1 \\ c(\cos x_1 - 2\sin x_2) &= x_2 \end{aligned}\right. .$$

Define: $\Phi\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = c\begin{pmatrix} \cos x_1 - \sin x_2 \\ \cos x_1 - 2\sin x_2 \end{pmatrix} \quad \Rightarrow \quad D\Phi\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = -c\begin{pmatrix} \sin x_1 & \cos x_2 \\ \sin x_1 & 2\cos x_2 \end{pmatrix}$.

Choose *appropriate* norm: $\|\cdot\|$ = ∞-norm $\|\cdot\|_\infty$ ($\rightarrow$ Example 1.1.4) ;

$$\text{if } c < \frac{1}{3} \quad \Rightarrow \quad \|D\Phi(\mathbf{x})\|_\infty < 1 \quad \forall \mathbf{x} \in \mathbb{R}^2 ,$$

➢ (at least) linear convergence of the fixed point iteration.

The existence of a fixed point is also guaranteed, because Φ maps into the closed set $[-3,3]^2$. Thus, the Banach fixed point theorem, Thm. 1.2.3, can be applied.

◇

What about higher order convergence ($\rightarrow$ Def. 1.1.13) ?

Refined convergence analysis for $n = 1$ (scalar case, $\Phi : \mathrm{dom}(\Phi) \subset \mathbb{R} \mapsto \mathbb{R}$):

Theorem 1.2.6 (Taylor's formula). → *[?, Sect. 5.5]*
If $\Phi : U \subset \mathbb{R} \mapsto \mathbb{R}$, U *interval, is* $m+1$ *times continuously differentiable,* $x \in U$

$$\Phi(y) - \Phi(x) = \sum_{k=1}^{m} \frac{1}{k!} \Phi^{(k)}(x)(y-x)^k + O(|y-x|^{m+1}) \quad \forall y \in U \; . \tag{1.2.3}$$

Apply Taylor expansion (1.2.3) to iteration function Φ:

If $\Phi(x^*) = x^*$ and $\Phi : \mathrm{dom}(\Phi) \subset \mathbb{R} \mapsto \mathbb{R}$ is "sufficiently smooth"

$$x^{(k+1)} - x^* = \Phi(x^{(k)}) - \Phi(x^*) = \sum_{l=1}^{m} \frac{1}{l!} \Phi^{(l)}(x^*)(x^{(k)} - x^*)^l + O(|x^{(k)} - x^*|^{m+1}) \; . \tag{1.2.4}$$

Lemma 1.2.7 (Higher order local convergence of fixed point iterations).
If $\Phi : U \subset \mathbb{R} \mapsto \mathbb{R}$ *is* $m+1$ *times continuously differentiable,* $\Phi(x^*) = x^*$ *for some* x^* *in the interior of* U, *and* $\Phi^{(l)}(x^*) = 0$ *for* $l = 1, \ldots, m$, $m \geq 1$, *then the fixed point iteration* (1.2.1) *converges locally to* x^* *with order* $\geq m+1$ *(→ Def. 1.1.13).*

Proof. For neighborhood $\mathcal{U}$ of x^*

$$(1.2.4) \quad \Rightarrow \quad \exists C > 0\colon \quad |\Phi(y) - \Phi(x^*)| \leq C\,|y - x^*|^{m+1} \quad \forall y \in \mathcal{U}\ .$$

$$\delta^m C < 1/2: \quad |x^{(0)} - x^*| < \delta \quad \Rightarrow \quad |x^{(k)} - x^*| < 2^{-k}\delta \quad \succ \quad \text{local convergence}\ .$$

Then appeal to (1.2.4) □

Example 1.2.1 continued:

$$\Phi_2'(x) = \frac{1 - xe^x}{(1 + e^x)^2} = 0 \quad \text{, if} \quad xe^x - 1 = 0 \quad \text{hence quadratic convergence ! .}$$

Example 1.2.1 continued: Since $x^* e^{x^*} - 1 = 0$

$$\Phi_1'(x) = -e^{-x} \quad \Rightarrow \quad \Phi_1'(x^*) = -x^* \approx -0.56 \quad \text{hence local linear convergence .}$$

$$\Phi_3'(x) = 1 - xe^x - e^x \quad \Rightarrow \quad \Phi_3'(x^*) = -\frac{1}{x^*} \approx -1.79 \quad \text{hence no convergence .}$$

Remark 1.2.6 (Termination criterion for contractive fixed point iteration).

Recap of Rem. 1.1.10:

Termination criterion for contractive fixed point iteration, *c.f.* (1.2.3), with contraction factor $0 \leq L < 1$:

$$\left\|\mathbf{x}^{(k+m)} - \mathbf{x}^{(k)}\right\| \overset{\triangle\text{-ineq.}}{\leq} \sum_{j=k}^{k+m-1} \left\|\mathbf{x}^{(j+1)} - \mathbf{x}^{(j)}\right\| \leq \sum_{j=k}^{k+m-1} L^{j-k} \left\|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\right\|$$

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

hence for $m \to \infty$, with $\mathbf{x}^* := \lim_{k\to\infty} \mathbf{x}^{(k)}$:

$$\left\|\mathbf{x}^* - \mathbf{x}^{(k)}\right\| \leq \frac{L^{k-l}}{1-L} \left\|\mathbf{x}^{(l+1)} - \mathbf{x}^{(l)}\right\| . \tag{1.2.5}$$

Set $l = 0$ in (1.2.5)	Set $l = k-1$ in (1.2.5)
▼	▼
a priori termination criterion	a posteriori termination criterion
$\left\Vert \mathbf{x}^* - \mathbf{x}^{(k)} \right\Vert \leq \frac{L^k}{1-L} \left\Vert \mathbf{x}^{(1)} - \mathbf{x}^{(0)} \right\Vert$ (1.2.6)	$\left\Vert \mathbf{x}^* - \mathbf{x}^{(k)} \right\Vert \leq \frac{L}{1-L} \left\Vert \mathbf{x}^{(k)} - \mathbf{x}^{(k-1)} \right\Vert$ (1.2.7)

△

1.3 Zero Finding

Now, focus on scalar case $n = 1$: $F : I \subset \mathbb{R} \mapsto \mathbb{R}$ **continuous**, I interval

Sought: $x^* \in I$: $F(x^*) = 0$

1.3.1 Bisection

Idea: use ordering of real numbers & intermediate value theorem

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

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

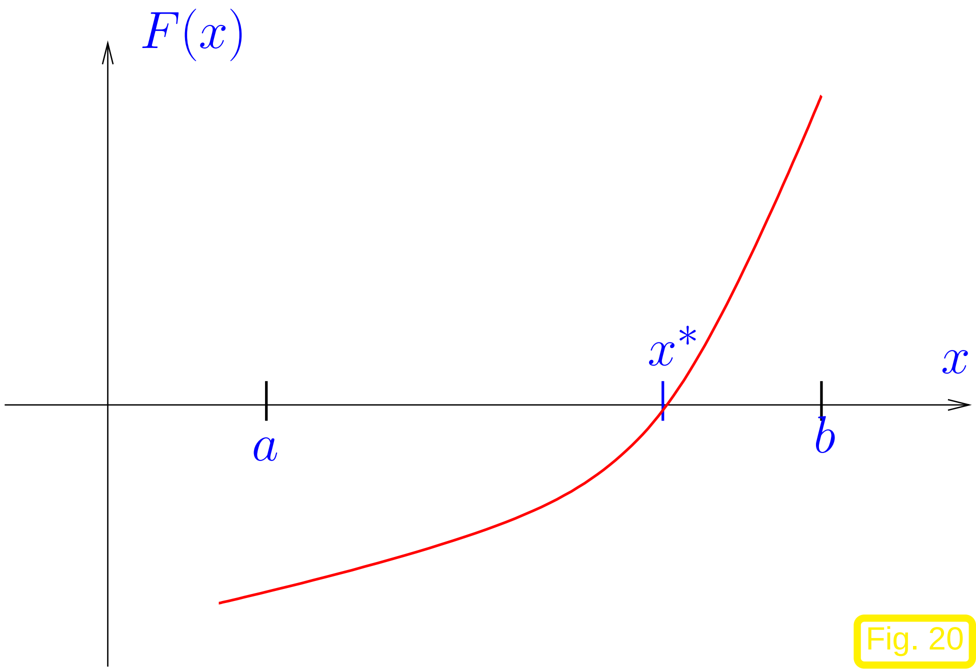


Fig. 20

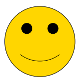
Algorithm 1.3.1 (Bisection method).

Code 1.3.2: Bisection method

```
def mybisect(f,a,b,tol=1e-12):
    if a>b:
        t = b; a = b; b = t
    fa = f(a); fb = f(b)
    if fa*fb > 0: raise ValueError
```

```python
    v = 1
    if fa>0: v = -1
    x = 0.5*(a+b)
    k = 1
    while (b-a>tol) and (a<x) and (x<b):
        if v*f(x)>0: b = x
        else: a = x
        x = 0.5*(a+b)
        k += 1
    return x, k

if __name__=='__main__':
    f = lambda x: x**3 - 2*x**2 + 4.*x/3. - 8./27.
    x, k = mybisect(f,0,1)
    print 'x_bisect = ', x, ' after k=', k, 'iterations'
    from scipy.optimize import fsolve, bisection
    x = fsolve(f,0,full_output=True)
    print x
    print bisection(f,0,1)
```



Advantages:

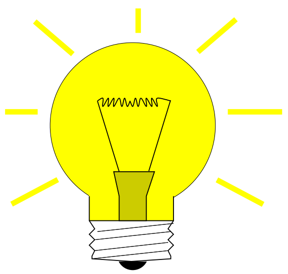
- “foolproof”
- requires only F evaluations

Drawbacks: Merely linear convergence: $|x^{(k)} - x^*| \leq 2^{-k}|b-a|$

▶ $\log_2\left(\frac{|b-a|}{\mathtt{tol}}\right)$ steps necessary

1.3.2 Model function methods

$\hat{=}$ class of iterative methods for finding zeroes of F:



Idea: Given: approximate zeroes $x^{(k)}, x^{(k-1)}, \dots, x^{(k-m)}$

❶ replace F with model function $\widetilde{F}$
(using function values/derivative values in $x^{(k)}, x^{(k-1)}, \dots, x^{(k-m)}$)

❷ $x^{(k+1)} :=$ zero of $\widetilde{F}$
(has to be readily available $\leftrightarrow$ analytic formula)

Distinguish (see (1.1.1)):

one-point methods : $x^{(k+1)} = \Phi_F(x^{(k)}), k \in \mathbb{N}$ (e.g., fixed point iteration $\rightarrow$ Sect. 1.2)
multi-point methods : $x^{(k+1)} = \Phi_F(x^{(k)}, x^{(k-1)}, \dots, x^{(k-m)})$, $k \in \mathbb{N}, m = 2, 3, \dots$.

Gradinaru
D-MATH

1.3.2.1 Newton method in scalar case

Assume: $F : I \mapsto \mathbb{R}$ continuously differentiable

model function:= tangent at F in $x^{(k)}$:

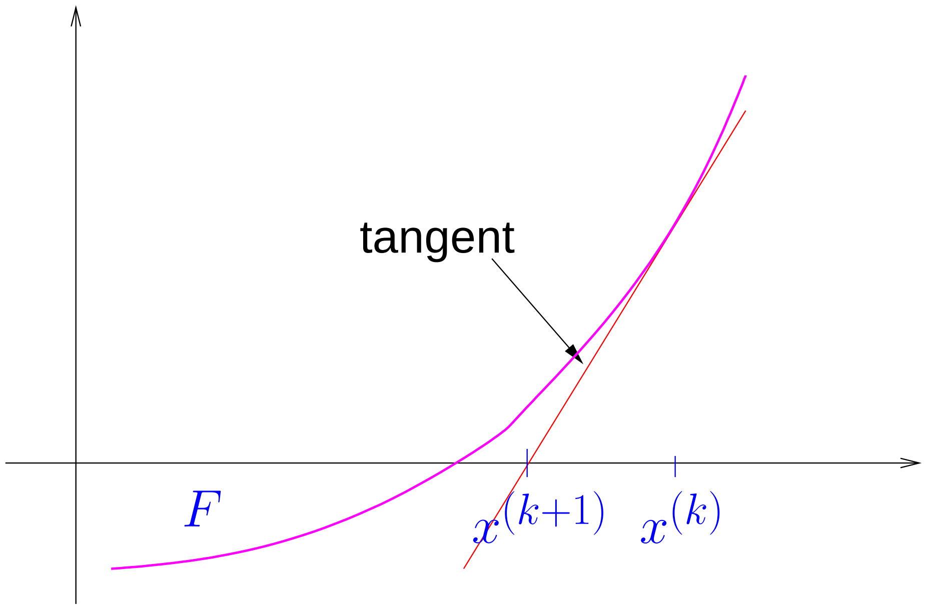
$$\widetilde{F}(x) := F(x^{(k)}) + F'(x^{(k)})(x - x^{(k)})$$

take $x^{(k+1)}$:= zero of tangent

We obtain Newton iteration

$$\boxed{x^{(k+1)} := x^{(k)} - \frac{F(x^{(k)})}{F'(x^{(k)})}} , \qquad (1.3.1)$$

that requires $F'(x^{(k)}) \neq 0$.



Example 1.3.3 (Newton method in 1D). ($\rightarrow$ Ex. 1.2.1)

Newton iterations for two different scalar non-linear equation with the same solution sets:

$$F(x) = xe^x - 1 \Rightarrow \quad F'(x) = e^x(1+x) \quad \Rightarrow \quad x^{(k+1)} = x^{(k)} - \frac{x^{(k)}e^{x^{(k)}} - 1}{e^{x^{(k)}}(1 + x^{(k)})} = \frac{(x^{(k)})^2 + e^{-x^{(k)}}}{1 + x^{(k)}}$$

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

1.2.1 1.1.13

Newton iteration (1.3.1) $\hat{=}$ fixed point iteration ($\rightarrow$ Sect.1.2) with iteration function

$$\Phi(x) = x - \frac{F(x)}{F'(x)} \quad \Rightarrow \quad \Phi'(x) = \frac{F(x)F''(x)}{(F'(x))^2} \quad \Rightarrow \quad \Phi'(x^*) = 0 \,,$$, if $F(x^*) = 0$, $F'(x^*) \neq 0$.

From Lemma 1.2.7:

Newton method locally quadratically convergent ($\rightarrow$ Def. 1.1.13) to zero x^*, if $F'(x^*) \neq 0$

1.3.2.2 Special one-point methods

Idea underlying other one-point methods: non-linear local approximation

Useful, if *a priori knowledge* about the structure of F (e.g. about F being a rational function, see below) is available. This is often the case, because many problems of 1D zero finding are posed for functions given in analytic form with a few parameters.

Prerequisite: Smoothness of F: $F \in C^m(I)$ for some $m > 1$

Example 1.3.4 (Halley's iteration).

Given $x^{(k)} \in I$, next iterate := zero of model function: $h(x^{(k+1)}) = 0$, where

$$h(x) := \frac{a}{x+b} + c \quad \text{(rational function) such that } F^{(j)}(x^{(k)}) = h^{(j)}(x^{(k)}) \;, \quad j = 0,1,2 \;.$$

$$\Downarrow$$

$$\frac{a}{x^{(k)}+b} + c = F(x^{(k)}) \;, \; -\frac{a}{(x^{(k)}+b)^2} = F'(x^{(k)}) \;, \; \frac{2a}{(x^{(k)}+b)^3} = F''(x^{(k)}) \;.$$

$$\Downarrow$$

$$x^{(k+1)} = x^{(k)} - \frac{F(x^{(k)})}{F'(x^{(k)})} \cdot \frac{1}{1 - \frac{1}{2}\frac{F(x^{(k)})F''(x^{(k)})}{F'(x^{(k)})^2}} \;.$$

Halley's iteration for $F(x) = \dfrac{1}{(x+1)^2} + \dfrac{1}{(x+0.1)^2} - 1 \;, \quad x > 0 :$ and $x^{(0)} = 0$

k	$x^{(k)}$	$F(x^{(k)})$	$x^{(k)} - x^{(k-1)}$	$x^{(k)} - x^*$
1	0.19865959351191	10.90706835180178	-0.19865959351191	-0.84754290138257
2	0.69096314049024	0.94813655914799	-0.49230354697833	-0.35523935440424
3	1.02335017694603	0.03670912956750	-0.33238703645579	-0.02285231794846
4	1.04604398836483	0.00024757037430	-0.02269381141880	-0.00015850652965
5	1.04620248685303	0.00000001255745	-0.00015849848821	-0.00000000804145

Compare with Newton method (1.3.1) for the same problem:

k	$x^{(k)}$	$F(x^{(k)})$	$x^{(k)} - x^{(k-1)}$	$x^{(k)} - x^*$
1	0.04995004995005	44.38117504792020	-0.04995004995005	-0.99625244494443
2	0.12455117953073	19.62288236082625	-0.07460112958068	-0.92165131536375
3	0.23476467495811	8.57909346342925	-0.11021349542738	-0.81143781993637
4	0.39254785728080	3.63763326452917	-0.15778318232269	-0.65365463761368
5	0.60067545233191	1.42717892023773	-0.20812759505112	-0.44552704256257
6	0.82714994286833	0.46286007749125	-0.22647449053641	-0.21905255202615
7	0.99028203077844	0.09369191826377	-0.16313208791011	-0.05592046411604
8	1.04242438221432	0.00592723560279	-0.05214235143588	-0.00377811268016
9	1.04618505691071	0.00002723158211	-0.00376067469639	-0.00001743798377
10	1.04620249452271	0.00000000058056	-0.00001743761199	-0.00000000037178

Note that Halley's iteration is superior in this case, since F is a rational function.

! Newton method converges more slowly, but also needs less effort per step (⟶ Sect. **??**) ◇

In the previous example Newton's method performed rather poorly. Often its convergence can be boosted by converting the non-linear equation to an equivalent one (that is, one with the same solutions) for another function g, which is "closer to a linear function":

Assume $F \approx \widehat{F}$, where $\widehat{F}$ is invertible with an inverse $\widehat{F}^{-1}$ that can be evaluated with little effort.

$$\blacktriangleright \qquad g(x) := \widehat{F}^{-1}(F(x)) \approx x \ .$$

Then apply Newton's method to $g(x)$, using the formula for the derivative of the inverse of a function

$$\frac{d}{dy}(\widehat{F}^{-1})(y) = \frac{1}{\widehat{F}'(\widehat{F}^{-1}(y))} \quad \Rightarrow \quad g'(x) = \frac{1}{\widehat{F}'(g(x))} \cdot F'(x) \ .$$

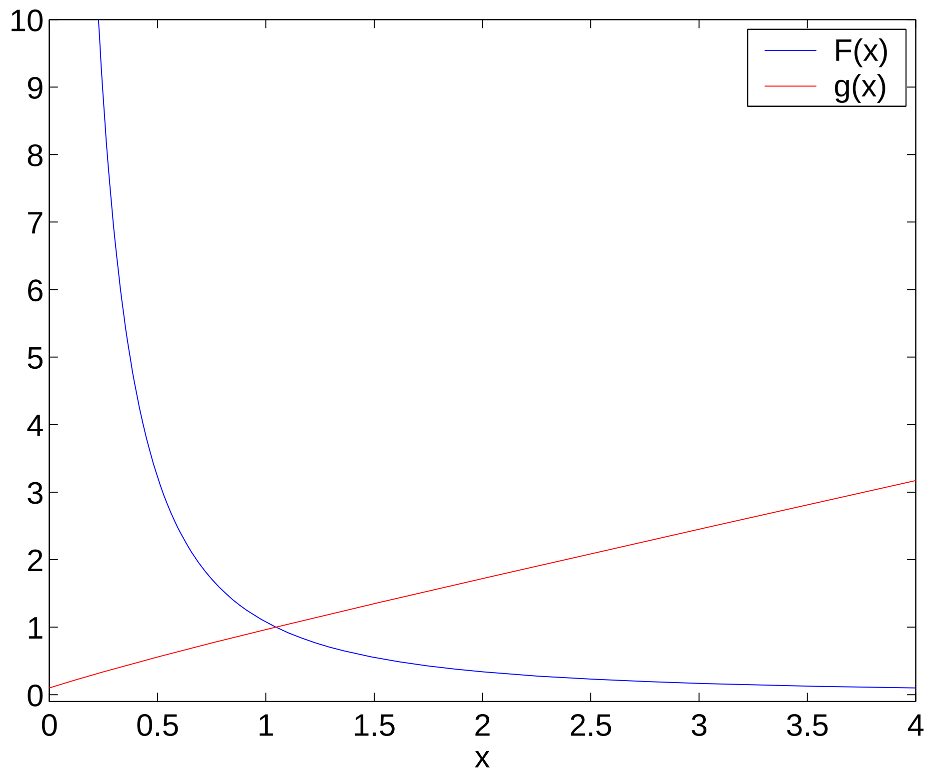
Example 1.3.5 (Adapted Newton method).

$$F(x) = \frac{1}{(x+1)^2} + \frac{1}{(x+0.1)^2} - 1 \ , \quad x > 0 :$$

Observation:

$$F(x) + 1 \approx 2x^{-2} \text{ for } x \gg 1$$

and so $g(x) := \dfrac{1}{\sqrt{F(x)+1}}$ "almost" linear for $x \gg 1$



Idea: instead of $F(x) \stackrel{!}{=} 0$ tackle $g(x) \stackrel{!}{=} 1$ with Newton's method (1.3.1).

$$x^{(k+1)} = x^{(k)} - \frac{g(x^{(k)}) - 1}{g'(x^{(k)})} = x^{(k)} + \left(\frac{1}{\sqrt{F(x^{(k)})+1}} - 1\right) \frac{2(F(x^{(k)})+1)^{3/2}}{F'(x^{(k)})}$$

$$= x^{(k)} + \frac{2(F(x^{(k)})+1)(1 - \sqrt{F(x^{(k)})+1})}{F'(x^{(k)})} .$$

Convergence recorded for $x^{(0)} = 0$:

k	$x^{(k)}$	$F(x^{(k)})$	$x^{(k)} - x^{(k-1)}$	$x^{(k)} - x^*$
1	0.91312431341979	0.24747993091128	0.91312431341979	-0.13307818147469
2	1.04517022155323	0.00161402574513	0.13204590813344	-0.00103227334125
3	1.04620244004116	0.00000008565847	0.00103221848793	-0.00000005485332
4	1.04620249489448	0.00000000000000	0.00000005485332	-0.00000000000000

◇

For zero finding there is wealth of iterative methods that offer higher order of convergence.

One idea: consistent modification of the Newton-Iteration:

▶ fixed point iteration : $\Phi(x) = x - \dfrac{F(x)}{F'(x)} H(x)$ with "proper" $H : I \mapsto \mathbb{R}$.

Aim: find H such that the method is of p-th order; tool: Lemma 1.2.7.

Assume: F smooth "enough" and $\exists\, x^* \in I$: $F(x^*) = 0$, $F'(x^*) \neq 0$.

$$\Phi = x - uH \quad , \quad \Phi' = 1 - u'H - uH' \quad , \quad \Phi'' = -u''H - 2u'H - uH'' \; ,$$

$$\text{with } u = \frac{F}{F'} \quad \Rightarrow \quad u' = 1 - \frac{FF''}{(F')^2} \quad , \quad u'' = -\frac{F''}{F'} + 2\frac{F(F'')^2}{(F')^3} - \frac{FF'''}{(F')^2} \; .$$

$F(x^*) = 0$ ➤ $u(x^*) = 0$, $u'(x^*) = 1$, $u''(x^*) = -\frac{F''(x^*)}{F'(x^*)}$.

$$\blacktriangleright \quad \Phi'(x^*) = 1 - H(x^*) \quad , \quad \Phi''(x^*) = \frac{F''(x^*)}{F'(x^*)} H(x^*) - 2H'(x^*) \; . \tag{1.3.2}$$

Lemma 1.2.7 ➢ **Necessary** conditions for local convergencd of order p:

$p = 2$ (quadratical convergence): $H(x^*) = 1$,

$p = 3$ (cubic convergence): $H(x^*) = 1 \quad \wedge \quad H'(x^*) = \frac{1}{2} \frac{F''(x^*)}{F'(x^*)}$.

In particular: $H(x) = G(1 - u'(x))$ with ”proper” G

$$\blacktriangleright \quad \text{fixed point iteration} \quad x^{(k+1)} = x^{(k)} - \frac{F(x^{(k)})}{F'(x^{(k)})} G\left(\frac{F(x^{(k)}) F''(x^{(k)})}{(F'(x^{(k)}))^2} \right) \; . \tag{1.3.3}$$

Lemma 1.3.1. *If* $F \in C^2(I)$, $F(x^*) = 0$, $F'(x^*) \neq 0$, $G \in C^2(U)$ *in a neighbourhood* U *of* 0, $G(0) = 1$, $G'(0) = \frac{1}{2}$, *then the fixed point iteration* (1.3.3) *converge locally cubically to* x^*.

Proof: Lemma 1.2.7, (1.3.2) and

$$H(x^*) = G(0) \quad , \quad H'(x^*) = -G'(0) u''(x^*) = G'(0) \frac{F''(x^*)}{F'(x^*)} \; .$$

Gradinaru
D-MATH

Example 1.3.6 (Example of modified Newton method).

- $G(t) = \dfrac{1}{1 - \frac{1}{2}t}$ ➡ Halley's iteration ($\longrightarrow$ Ex. 1.3.4)
- $G(t) = \dfrac{2}{1 + \sqrt{1 - 2t}}$ ➡ Euler's iteration
- $G(t) = 1 + \frac{1}{2}t$ ➡ quadratic inverse interpolation

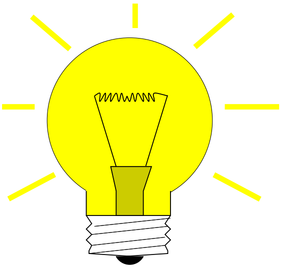
Numerical experiment:

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

k	$e^{(k)} := x^{(k)} - x^*$		
	Halley	Euler	Quad. Inv.
1	2.81548211105635	3.57571385244736	2.03843730027891
2	1.37597082614957	2.76924150041340	1.02137913293045
3	0.34002908011728	1.95675490333756	0.28835890388161
4	0.00951600547085	1.25252187565405	0.01497518178983
5	0.00000024995484	0.51609312477451	0.00000315361454
6		0.14709716035310	
7		0.00109463314926	
8		0.00000000107549	

◇

1.3.2.3 Multi-point methods



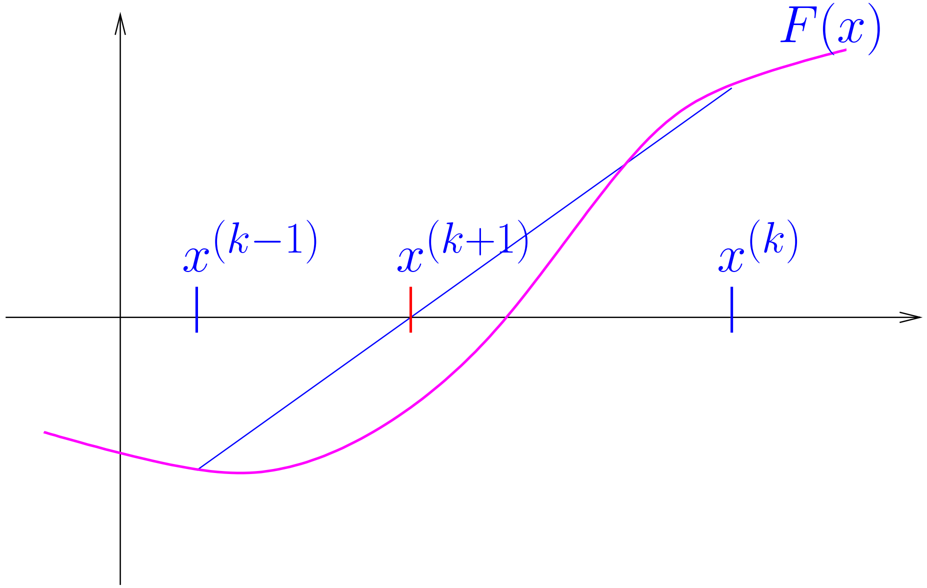
Idea: Replace F with interpolating polynomial

producing interpolatory model function methods

Simplest representative: secant method

$x^{(k+1)}$ = zero of secant

$$s(x) = F(x^{(k)}) + \frac{F(x^{(k)}) - F(x^{(k-1)})}{x^{(k)} - x^{(k-1)}}(x - x^{(k)}) \;, \tag{1.3.4}$$



$$\blacktriangleright \quad x^{(k+1)} = x^{(k)} - \frac{F(x^{(k)})(x^{(k)} - x^{(k-1)})}{F(x^{(k)}) - F(x^{(k-1)})} \;. \tag{1.3.5}$$

secant method
(python implementation)

- Only one function evaluation per step
- no derivatives required !

Code 1.3.7: secant method

```
def secant(f,x0,x1,maxit=50,tol=1e-12):
    fo = f(x0)
    for k in xrange(maxit):
        fn = f(x1)
        print 'x1=',x1, 'f(x1)=', fn
        s = fn*(x1-x0)/(fn-fo)
        x0 = x1; x1 -= s
        if abs(s)<tol:
            x = x1
            return x, k
        fo = fn
    x = NaN
    return x, maxit
```

Remember: $F(x)$ may only be available as output of a (complicated) procedure. In this case it is difficult to find a procedure that evaluates $F'(x)$. Thus the significance of methods that do not involve evaluations of derivatives.

Example 1.3.8 (secant method). $F(x) = xe^x - 1\,,\quad x^{(0)} = 0\,, x^{(1)} = 5\,.$

k	$x^{(k)}$	$F(x^{(k)})$	$e^{(k)} := x^{(k)} - x^*$	$\frac{\log\lvert e^{(k+1)}\rvert - \log\lvert e^{(k)}\rvert}{\log\lvert e^{(k)}\rvert - \log\lvert e^{(k-1)}\rvert}$
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	

A startling observation: the method seems to have a *fractional* (!) order of convergence, see Def. 1.1.13.

◇

Remark 1.3.9 (Fractional order of convergence of secant method).

Indeed, a fractional order of convergence can be proved for the secant method, see[**?**, Sect. 18.2]. Here is a crude outline of the reasoning:

Asymptotic convergence of secant method: error $e^{(k)} := x^{(k)} - x^*$

$$e^{(k+1)} = \Phi(x^* + e^{(k)}, x^* + e^{(k-1)}) - x^* \quad \text{, with} \quad \Phi(x,y) := x - \frac{F(x)(x-y)}{F(x)-F(y)} \,. \tag{1.3.6}$$

Use MAPLE to find Taylor expansion (assuming F sufficiently smooth):

```
> Phi := (x,y) -> x-F(x)*(x-y)/(F(x)-F(y));
> F(s) := 0;
> e2 = normal(mtaylor(Phi(s+e1,s+e0)-s,[e0,e1],4));
```

➣ linearized error propagation formula:

$$e^{(k+1)} \doteq \frac{1}{2}\frac{F''(x^*)}{F'(x^*)} e^{(k)} e^{(k-1)} = C e^{(k)} e^{(k-1)} \,. \tag{1.3.7}$$

Try $e^{(k)} = K(e^{(k-1)})^p$ to determine the order of convergence ($\rightarrow$ Def. 1.1.13):

$$\Rightarrow \qquad e^{(k+1)} = K^{p+1}(e^{(k-1)})^{p^2}$$

$$\Rightarrow \qquad (e^{(k-1)})^{p^2-p-1} = K^{-p}C \quad \Rightarrow \quad p^2 - p - 1 = 0 \quad \Rightarrow \quad p = \tfrac{1}{2}(1 \pm \sqrt{5}) \,.$$

As $e^{(k)} \rightarrow 0$ for $k \rightarrow \infty$ we get the rate of convergence $p = \frac{1}{2}(1 + \sqrt{5}) \approx 1.62$ (see Ex. 1.3.8 !) △

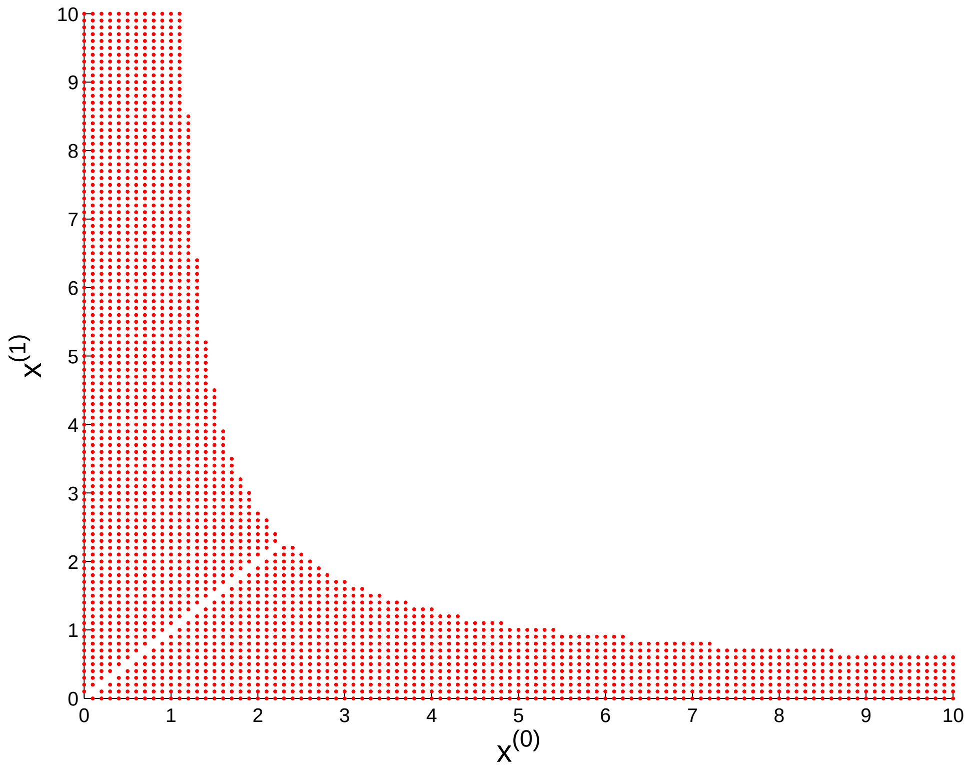
Example 1.3.10 (local convergence of secant method).

$$F(x) = \arctan(x)$$

$\cdot \hat{=}$ secant method converges for a pair $(x^{(0)}, x^{(1)})$ of initial guesses.

= local convergence → Def. 1.1.3

▷



◇

Another class of multi-point methods: *inverse interpolation*

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

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

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

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

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

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

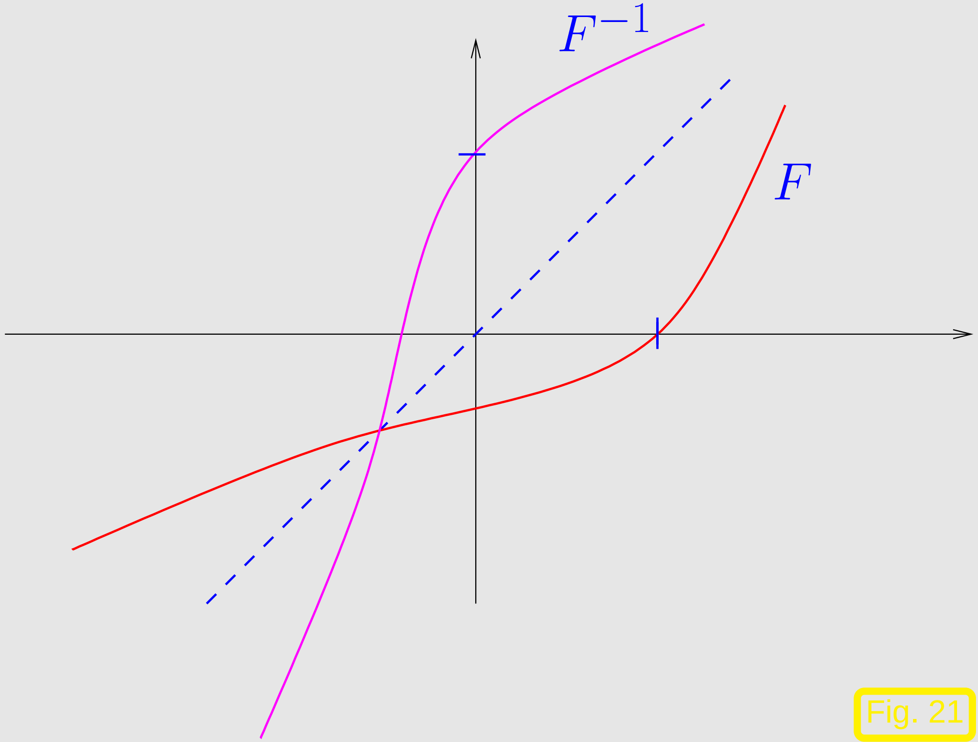
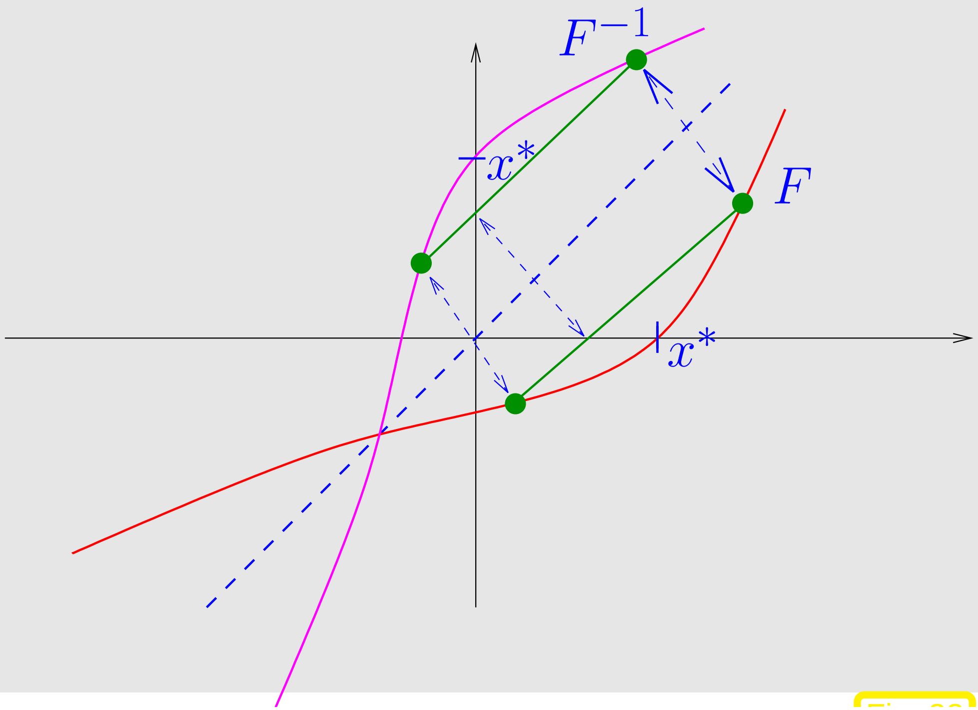


Fig. 21

Case $m = 1$ ➢ secant method



Gradinaru
D-MATH

Case $m = 2$: quadratic inverse interpolation, see [?, Sect. 4.5]

MAPLE code:
```
p := x-> a*x^2+b*x+c;
solve({p(f0)=x0,p(f1)=x1,p(f2)=x2},{a,b,c});
assign(%); p(0);
```

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

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

Example 1.3.11 (quadratic inverse interpolation). $F(x) = xe^x - 1$, $x^{(0)} = 0$, $x^{(1)} = 2.5$,$x^{(2)} = 5$.

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	

Also in this case the numerical experiment hints at a fractional rate of convergence, as in the case of the secant method, see Rem. 1.3.9.

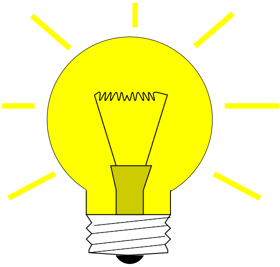
$\Diamond$

1.4 Newton's Method

Non-linear system of equations: for $F : D \subset \mathbb{R}^n \mapsto \mathbb{R}^n$ find $\mathbf{x}^* \in D$: $F(\mathbf{x}^*) = 0$

Assume: $F : D \subset \mathbb{R}^n \mapsto \mathbb{R}^n$ continuously differentiable

1.4.1 The Newton iteration



Idea ($\longrightarrow$ Sect. 1.3.2.1): local linearization:

Given $\mathbf{x}^{(k)} \in D$ ➢ $\mathbf{x}^{(k+1)}$ as zero of affine linear model function

$$F(\mathbf{x}) \approx \widetilde{F}(\mathbf{x}) := F(\mathbf{x}^{(k)}) + DF(\mathbf{x}^{(k)})(\mathbf{x} - \mathbf{x}^{(k)}) \ ,$$

$DF(\mathbf{x}) \in \mathbb{R}^{n,n}$ = Jacobian (*ger.:* Jacobi-Matrix), $DF(\mathbf{x}) = \left(\frac{\partial F_j}{\partial x_k}(\mathbf{x})\right)_{j,k=1}^{n}$.

➤ Newton iteration: ($\leftrightarrow$ (1.3.1) for $n = 1$)

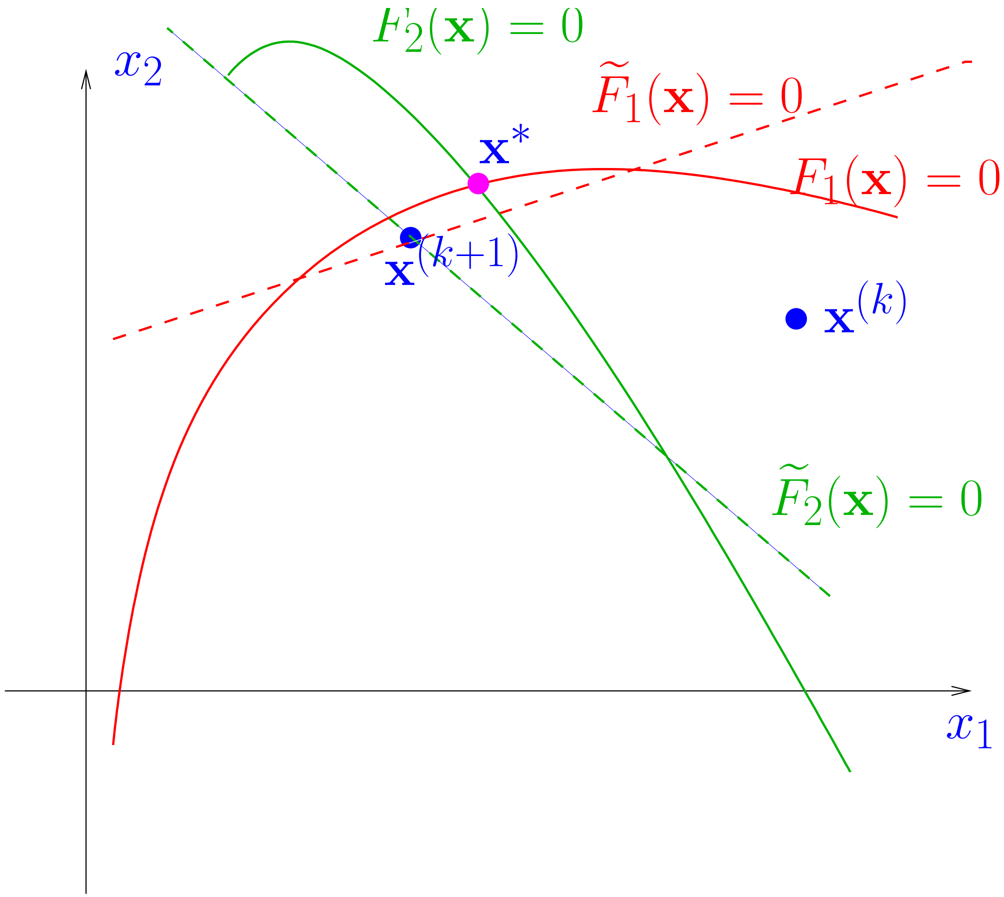
$$\boxed{\mathbf{x}^{(k+1)} := \mathbf{x}^{(k)} - DF(\mathbf{x}^{(k)})^{-1} F(\mathbf{x}^{(k)})} \quad , \quad [\text{ if } DF(\mathbf{x}^{(k)}) \text{ regular }] \tag{1.4.1}$$

Terminology: $-DF(\mathbf{x}^{(k)})^{-1} F(\mathbf{x}^{(k)})$ = Newton correction

Gradinaru
D-MATH

Illustration of idea of Newton's method for $n = 2$:

$\triangleright$



Sought: intersection point $\mathbf{x}^*$ of the curves $F_1(\mathbf{x}) = 0$ and $F_2(\mathbf{x}) = 0$.

Idea: $\mathbf{x}^{(k+1)} =$ the intersection of two straight lines ($=$ zero sets of the components of the model function) that are approximations of the original curves template for Newton method:

Solve linear system:

solve(A,b) = $\mathbf{A}^{-1}b$

F, DF: functions

A posteriori termination criterion

MATLAB-CODE: Newtonverfahren

```
def newton(x,F,DF,tol,maxit):
  for i in xrange(maxit):
      s = solve(DF(x), F(x))
      x -= s
      if norm(s) < tol*norm(x):return
```

Example 1.4.1 (Newton method in 2D).

$$F(\mathbf{x}) = \begin{pmatrix} x_1^2 - x_2^4 \\ x_1 - x_2^3 \end{pmatrix} \quad , \quad \mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \in \mathbb{R}^2 \quad \text{with solution} \quad F\begin{pmatrix} 1 \\ 1 \end{pmatrix} = 0 \; .$$

Jacobian (analytic computation):
$$DF(\mathbf{x}) = \begin{pmatrix} \partial_{x_1} F_1(x) & \partial_{x_2} F_1(x) \\ \partial_{x_1} F_2(x) & \partial_{x_2} F_2(x) \end{pmatrix} = \begin{pmatrix} 2x_1 & -4x_2^3 \\ 1 & -3x_2^2 \end{pmatrix}$$

Realization of Newton iteration (1.4.1):

1. Solve LSE

$$\begin{pmatrix} 2x_1 & -4x_2^3 \\ 1 & -3x_2^2 \end{pmatrix} \Delta\mathbf{x}^{(k)} = \begin{pmatrix} x_1^2 - x_2^4 \\ x_1 - x_2^3 \end{pmatrix} ,$$

where $\mathbf{x}^{(k)} = (x_1, x_2)^T$.

2. Set $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \Delta\mathbf{x}^{(k)}$

Code 1.4.2: Newton iteration in 2D

```
from scipy import array, diff, log, zeros, vstack
from scipy.linalg import norm, solve

F = lambda x:  array([x[0]**2 - x[1]**4, x[0]-x[1]**3])
DF = lambda x: array([[2*x[0], - 4*x[1]**3], [1, -3*x[1]**2]])

x = array([0.7, 0.7])
x0 = array([1., 1.])
tol = 1e-10

res = zeros(4); res[1:3] = x; res[3] = norm(x-x0)
print DF(x)
print F(x)
s = solve(DF(x),F(x))
x -= s
res1 = zeros(4); res[0] = 1.; res1[1:3] = x; res1[3] = norm(x-x0)
res = vstack((res,res1))
k = 2
while norm(s) > tol*norm(x):
    s = solve(DF(x),F(x))
```

```
    x -= s
    res1 = zeros(4); res1[0] = k; res1[1:3] = x; res1[3] = norm(x-x0)
    res = vstack((res,res1))
    k += 1

logdiff = diff(log(res[:,3]))
rates = logdiff[1:]/logdiff[:-1]

print res
print rates
```

k	$\mathbf{x}^{(k)}$	$\epsilon_k := \lVert\mathbf{x}^* - \mathbf{x}^{(k)}\rVert_2$
0	$(0.7,\ \ 0.7)^T$	4.24e-01
1	$(0.87850000000000,\ \ 1.064285714285714)^T$	1.37e-01
2	$(1.01815943274188,\ \ 1.00914882463936)^T$	2.03e-02
3	$(1.00023355916300,\ \ 1.00015913936075)^T$	2.83e-04
4	$(1.00000000583852,\ \ 1.00000002726552)^T$	2.79e-08
5	$(0.999999999999998,\ \ 1.000000000000000)^T$	2.11e-15
6	$(1,\ \ 1)^T$	

$\Diamond$

New aspect for $n \gg 1$ (compared to $n = 1$-dimensional case, section. 1.3.2.1):

Computation of the Newton correction is eventually costly!

Remark 1.4.3 (Affine invariance of Newton method).

An important property of the Newton iteration (1.4.1): affine invariance $\rightarrow$ [**?**, Sect .1.2.2]

set $G(\mathbf{x}) := \mathbf{A}F(\mathbf{x})$ with regular $\mathbf{A} \in \mathbb{R}^{n,n}$ so that $F(\mathbf{x}^*) = 0 \quad \Leftrightarrow \quad G(\mathbf{x}^*) = 0$.

affine invariance: Newton iteration for $G(\mathbf{x}) = 0$ is the same for all $\mathbf{A} \in GL(n)$!

This is a simple computation:

$$DG(\mathbf{x}) = \mathbf{A}DF(\mathbf{x}) \quad \Rightarrow \quad DG(\mathbf{x})^{-1}G(\mathbf{x}) = DF(\mathbf{x})^{-1}\mathbf{A}^{-1}\mathbf{A}F(\mathbf{x}) = DF(\mathbf{x})^{-1}F(\mathbf{x}) \ .$$

Use affine invariance as guideline for

- convergence theory for Newton's method: assumptions and results should be affine invariant, too.
- modifying and extending Newton's method: resulting schemes should preserve affine invariance.

$\triangle$

Gradinaru
D-MATH

Remark 1.4.4 (Differentiation rules). $\longrightarrow$ Repetition: basic analysis

Statement of the Newton iteration (1.4.1) for $F : \mathbb{R}^n \mapsto \mathbb{R}^n$ given as analytic expression entails computing the Jacobian DF. To avoid cumbersome component-oriented considerations, it is useful to know the *rules of multidimensional differentiation*:

Let V, W be finite dimensional vector spaces, $F : D \subset V \mapsto W$ sufficiently smooth. The differential $DF(\mathbf{x})$ of F in $\mathbf{x} \in V$ is the *unique*

$$\text{linear mapping} \quad DF(\mathbf{x}) : V \mapsto W \ ,$$
$$\text{such that} \quad \|F(\mathbf{x}+\mathbf{h}) - F(\mathbf{x}) - DF(\mathbf{h})\mathbf{h}\| = o(\|\mathbf{h}\|) \quad \forall \mathbf{h}\,,\ \|\mathbf{h}\| < \delta \ .$$

- For $F : V \mapsto W$ linear, i.e. $F(\mathbf{x}) = \mathbf{A}\mathbf{x}$, $\mathbf{A}$ matrix ➤ $DF(\mathbf{x}) = \mathbf{A}$.
- Chain rule: $F : V \mapsto W$, $G : W \mapsto U$ sufficiently smooth

$$D(G \circ F)(\mathbf{x})\mathbf{h} = DG(F(\mathbf{x}))(DF(\mathbf{x}))\mathbf{h}\ , \quad \mathbf{h} \in V\ , \mathbf{x} \in D\ . \tag{1.4.2}$$

• Product rule: $F : D \subset V \mapsto W$, $G : D \subset V \mapsto U$ sufficiently smooth, $b : W \times U \mapsto Z$ bilinear

$$T(\mathbf{x}) = b(F(\mathbf{x}), G(\mathbf{x})) \quad \Rightarrow \quad DT(\mathbf{x})\mathbf{h} = b(DF(\mathbf{x})\mathbf{h}, G(\mathbf{x})) + b(F(\mathbf{x}), DG(\mathbf{x})\mathbf{h}) \;, \quad \mathbf{h} \in V, \mathbf{x} \in D \;. \tag{1.4.3}$$

For $F : D \subset \mathbb{R}^n \mapsto \mathbb{R}$ the gradient $\mathbf{grad}\, F : D \mapsto \mathbb{R}^n$, and the Hessian matrix $HF(\mathbf{x}) : D \mapsto \mathbb{R}^{n,n}$ are defined as

$$\mathbf{grad}\, F(\mathbf{x})^T \mathbf{h} := DF(\mathbf{x})\mathbf{h} \;, \qquad \mathbf{h}_1^T HF(\mathbf{x})\mathbf{h}_2 := D(DF(\mathbf{x})(\mathbf{h}_1))(\mathbf{h}_2) \;, \quad \mathbf{h}, \mathbf{h}_1, \mathbf{h}_2 \in V \;.$$

△

Remark 1.4.5 (Simplified Newton method).

Simplified Newton Method: use the same $DF(\mathbf{x}^{(k)})$ for more steps

➢ (usually) merely linear convergence instead of quadratic convergence

△

Remark 1.4.6 (Numerical Differentiation for computation of Jacobian).

If $DF(\mathbf{x})$ is not available (e.g. when $F(\mathbf{x})$ is given only as a procedure):

Numerical Differentiation: $$\frac{\partial F_i}{\partial x_j}(\mathbf{x}) \approx \frac{F_i(\mathbf{x}+h\vec{e}_j) - F_i(\mathbf{x})}{h} .$$

Caution: impact of roundoff errors for small h **!**

$\triangle$

Example 1.4.7 (Roundoff errors and difference quotients).

Approximate derivative by difference quotient: $$f'(x) \approx \frac{f(x+h) - f(x)}{h} .$$

Calculus: better approximation for smaller $h > 0$, isn't it **?**

MATLAB-CODE: Numerical differentiation of exp(x)

```
h = 0.1; x = 0.
for i in xrange(16):
    df = (exp(x+h)-exp(x))/h
    print -i, df-1
    h *= 0.1
```

Recorded relative error, $f(x) = e^x$, $x = 0$ $\triangleright$

$\log_{10}(h)$	relative error
-1	0.05170918075648
-2	0.00501670841679
-3	0.00050016670838
-4	0.00005000166714
-5	0.00000500000696
-6	0.00000049996218
-7	0.00000004943368
-8	-0.00000000607747
-9	0.00000008274037
-10	0.00000008274037
-11	0.00000008274037
-12	0.00008890058234
-13	-0.00079927783736
-14	-0.00079927783736
-15	0.11022302462516
-16	-1.00000000000000

Note: An analysis based on expressions for remainder terms of Taylor expansions shows that the approximation error cannot be blamed for the loss of accuracy as $h \to 0$ (as expected).

Explanation relying on roundoff error analysis, see Sect. **??**:

MATLAB-CODE: Numerical differentiation of exp(x)

```
h = 0.1; x = 0.0
for i in xrange(16):
    df = (exp(x+h)-exp(x))/h
    print -i, df-1
    h *= 0.1
```

Obvious cancellation $\longrightarrow$ error amplification

$\log_{10}(h)$	relative error
-1	0.05170918075648
-2	0.00501670841679
-3	0.00050016670838
-4	0.00005000166714
-5	0.00000500000696
-6	0.00000049996218
-7	0.00000004943368
-8	-0.00000000607747
-9	0.00000008274037
-10	0.00000008274037
-11	0.00000008274037
-12	0.00008890058234
-13	-0.00079927783736
-14	-0.00079927783736
-15	0.11022302462516
-16	-1.00000000000000

$$\left.\begin{array}{l} f'(x) - \dfrac{f(x+h)-f(x)}{h} \longrightarrow 0 \\ \text{Impact of roundoff} \longrightarrow \infty \end{array}\right\} \quad \text{for } h \longrightarrow 0\ .$$

Analysis for $f(x) = \exp(x)$:

$$\mathtt{df} = \frac{e^{x+h}\boxed{(1+\delta_1)} - e^x \boxed{(1+\delta_2)}}{h}$$

correction factors take into account roundoff:
(→ "'axiom of roundoff analysis", Ass. **??**)

$$= e^x \left(\frac{e^h - 1}{h} + \frac{\delta_1 e^h - \delta_2}{h} \right)$$

$|\delta_1|, |\delta_2| \leq \mathtt{eps}$.

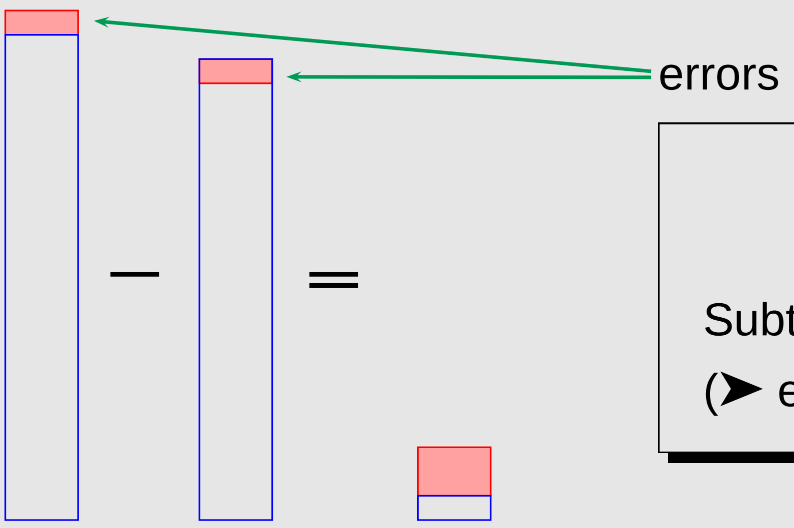
$1 + O(h) \qquad O(h^{-1})$ für $h \rightarrow 0$

▶ relative error: $\left|\dfrac{e^x - \mathtt{df}}{e^x}\right| \approx h + \dfrac{2\mathtt{eps}}{h} \rightarrow \min$ for $h = \sqrt{2\,\mathtt{eps}}$.

In double precision: $\sqrt{2\mathtt{eps}} = 2.107342425544702 \cdot 10^{-8}$

◇

What is this mysterious cancellation (*ger.:* Auslöschung) **?**



Cancellation

$\hat{=}$

Subtraction of almost equal numbers

(➤ extreme amplification of relative errors)

Example 1.4.8 (cancellation in decimal floating point arithmetic).

x, y afflicted with relative errors $\approx 10^{-7}$:

$$\begin{array}{rcll} x &=& 0.123467* & \leftarrow \text{7th digit perturbed} \\ y &=& 0.123456* & \leftarrow \text{7th digit perturbed} \\ \hline x-y &=& 0.000011* = 0.11*000 \cdot 10^{-4} & \leftarrow \text{3rd digit perturbed} \end{array}$$

padded zeroes

◇

1.4.2 Convergence of Newton's method

Newton iteration (1.4.1) $\hat{=}$ fixed point iteration ($\rightarrow$ Sect. 1.2) with

$$\Phi(\mathbf{x}) = \mathbf{x} - DF(\mathbf{x})^{-1}F(\mathbf{x}) \; .$$

[“product rule” : $D\Phi(\mathbf{x}) = \mathbf{I} - D(DF(\mathbf{x})^{-1})F(\mathbf{x}) - DF(\mathbf{x})^{-1}DF(\mathbf{x})$]

$$F(\mathbf{x}^*) = 0 \quad \Rightarrow \quad D\Phi(\mathbf{x}^*) = 0 \ .$$

1.2.7

Local quadratic convergence of Newton's method, if $DF(\mathbf{x}^*)$ regular

Example 1.4.9 (Convergence of Newton's method).

Ex. 1.4.1 cnt'd: record of iteration errors, see Code 1.4.1:

k	$\mathbf{x}^{(k)}$	$\epsilon_k := \Vert\mathbf{x}^* - \mathbf{x}^{(k)}\Vert_2$	$\dfrac{\log \epsilon_{k+1} - \log \epsilon_k}{\log \epsilon_k - \log \epsilon_{k-1}}$
0	$(0.7,\ \ 0.7)^T$	4.24e-01	
1	$(0.87850000000000,\ \ 1.06428571428571\,4)^T$	1.37e-01	1.69
2	$(1.01815943274188,\ \ 1.00914882463936)^T$	2.03e-02	2.23
3	$(1.00023355916300,\ \ 1.00015913936075)^T$	2.83e-04	2.15
4	$(1.00000000583852,\ \ 1.00000002726552)^T$	2.79e-08	1.77
5	$(0.999999999999998,\ \ 1.000000000000000)^T$	2.11e-15	
6	$(1,\ \ 1)^T$		

◇

There is a sophisticated theory about the convergence of Newton's method. For example one can find the following theorem in [**?**, Thm. 4.10], [**?**, Sect. 2.1]):

Theorem 1.4.1 (Local quadratic convergence of Newton's method)**. If***:*

(A) $D \subset \mathbb{R}^n$ *open and convex,*

(B) $F : D \mapsto \mathbb{R}^n$ *continuously differentiable,*

(C) $DF(\mathbf{x})$ *regular* $\forall \mathbf{x} \in D$,

(D) $\exists L \geq 0$: $\quad \left\| DF(\mathbf{x})^{-1}(DF(\mathbf{x}+\mathbf{v}) - DF(\mathbf{x})) \right\|_2 \leq L \left\| \mathbf{v} \right\|_2 \quad \begin{array}{l} \forall \mathbf{v} \in \mathbb{R}^n, \mathbf{v}+\mathbf{x} \in D, \\ \forall \mathbf{x} \in D \end{array}$,

(E) $\exists \mathbf{x}^*$: $\quad F(\mathbf{x}^*) = 0 \quad$ (*existence of solution* *in* D)

(F) *initial guess* $\mathbf{x}^{(0)} \in D$ *satisfies* $\quad \rho := \left\| \mathbf{x}^* - \mathbf{x}^{(0)} \right\|_2 < \dfrac{2}{L} \quad \wedge \quad B_\rho(\mathbf{x}^*) \subset D$.

then *the Newton iteration* (1.4.1) *satisfies:*

(i) $\mathbf{x}^{(k)} \in B_\rho(\mathbf{x}^*) := \{\mathbf{y} \in \mathbb{R}^n, \left\| \mathbf{y} - \mathbf{x}^* \right\| < \rho\}$ *for all* $k \in \mathbb{N}$,

(ii) $\lim\limits_{k\to\infty} \mathbf{x}^{(k)} = \mathbf{x}^*$,

(iii) $\left\| \mathbf{x}^{(k+1)} - \mathbf{x}^* \right\|_2 \leq \frac{L}{2} \left\| \mathbf{x}^{(k)} - \mathbf{x}^* \right\|_2^2 \quad$ (*local* *quadratic convergence*) .

✎ notation: ball $B_\rho(\mathbf{z}) := \{\mathbf{x} \in \mathbb{R}^n \colon \ \|\mathbf{x} - \mathbf{z}\|_2 \leq \rho\}$

Terminology: (D) $\hat{=}$ affine invariant Lipschitz condition

Problem: Usually neither ω nor x^* are known !

► In general: a priori estimates as in Thm. 1.4.1 are of little practical relevance.

1.4.3 Termination of Newton iteration

A first viable idea:

Asymptotically due to quadratic convergence:

$$\left\|\mathbf{x}^{(k+1)} - \mathbf{x}^*\right\| \ll \left\|\mathbf{x}^{(k)} - \mathbf{x}^*\right\| \quad \Rightarrow \quad \left\|\mathbf{x}^{(k)} - \mathbf{x}^*\right\| \approx \left\|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\right\| \ . \tag{1.4.4}$$

➢ quit iterating as soon as $\left\|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\right\| = \left\|DF(\mathbf{x}^{(k)})^{-1}F(\mathbf{x}^{(k)})\right\| < \tau \left\|\mathbf{x}^{(k)}\right\|$, with τ = tolerance

$\rightarrow$ uneconomical: one needless update, because $\mathbf{x}^{(k)}$ already accurate enough **!**

Remark 1.4.10. New aspect for $n \gg 1$: computation of Newton correction may be expensive **!** $\triangle$

Therefore we would like to use an a-posteriori termination criterion that dispenses with computing (and "inverting") another Jacobian $DF(\mathbf{x}^{(k)})$ just to tell us that $\mathbf{x}^{(k)}$ is already accurate enough.

Practical a-posteriori termination criterion for Newton's method:

$$\boxed{DF(\mathbf{x}^{(k-1)}) \approx DF(\mathbf{x}^{(k)}):\ \text{quit as soon as}\quad \left\|DF(\mathbf{x}^{(k-1)})^{-1}F(\mathbf{x}^{(k)})\right\| < \tau \left\|\mathbf{x}^{(k)}\right\|}$$

affine invariant termination criterion

Justification: we expect $DF(\mathbf{x}^{(k-1)}) \approx DF(\mathbf{x}^{(k)})$, when Newton iteration has converged. Then appeal to (1.4.4).

If we used the residual based termination criterion

$$\left\| F(\mathbf{x}^{(k)}) \right\| \leq \tau \ ,$$

then the resulting algorithm would not be affine invariant, because for $F(\mathbf{x}) = 0$ and $\mathbf{A}F(\mathbf{x}) = 0$, $\mathbf{A} \in \mathbb{R}^{n,n}$ regular, the Newton iteration might terminate with different iterates.

Terminology: $\Delta\bar{\mathbf{x}}^{(k)} := DF(\mathbf{x}^{(k-1)})^{-1} F(\mathbf{x}^{(k)}) \hat{=}$ simplified Newton correction

Reuse of LU-factorization of $DF(\mathbf{x}^{(k-1)})$ ➤ $\Delta\bar{\mathbf{x}}^{(k)}$ available with $O(n^2)$ operations

Summary: The Newton Method

- 🙂 converges *asymptotically* very fast: doubling of number of significant digits in each step
- 🙁 often a very small region of convergence, which requires an initial guess rather close to the solution.

Gradinaru
D-MATH

1.4.4 Damped Newton method

Example 1.4.11 (Local convergence of Newton's method).

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

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

◇

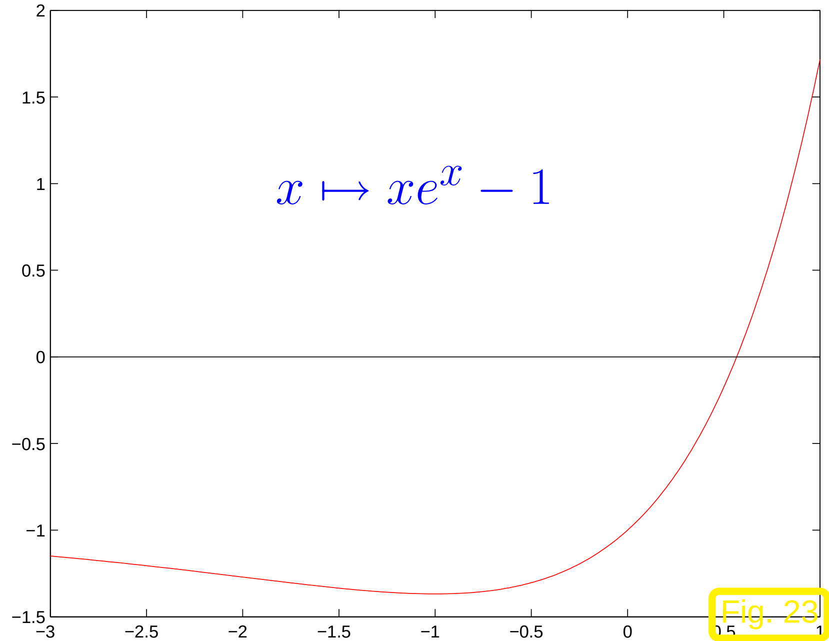


Fig. 23

Example 1.4.12 (Region of convergence of Newton method).

$$F(x) = \arctan(ax)\ , \quad a > 0, x \in \mathbb{R}$$

with zero $x^* = 0$.

◇

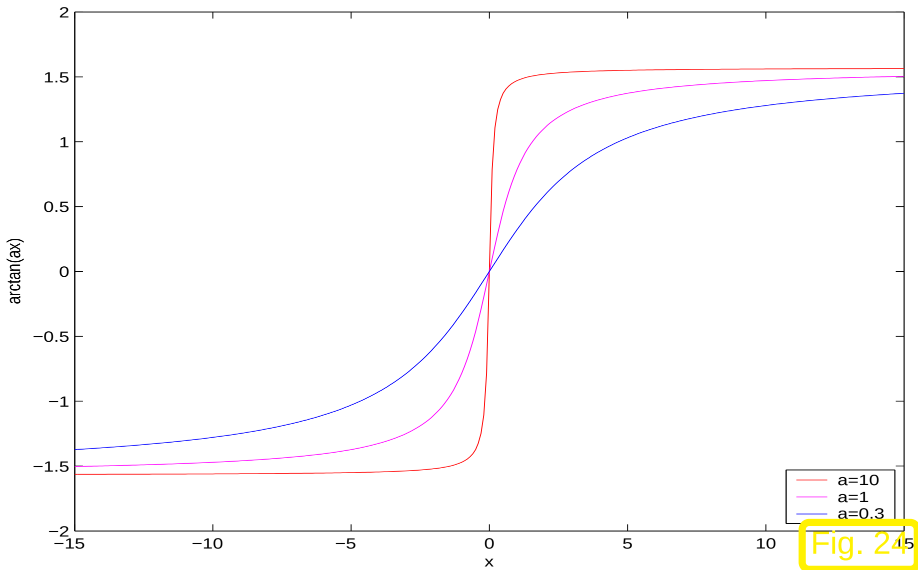
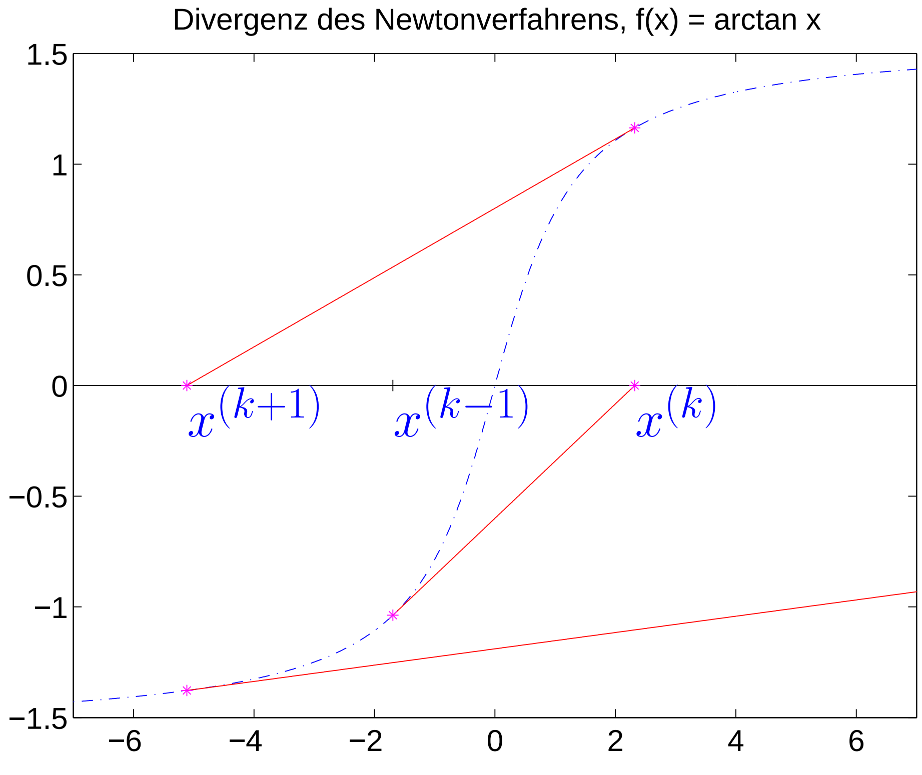
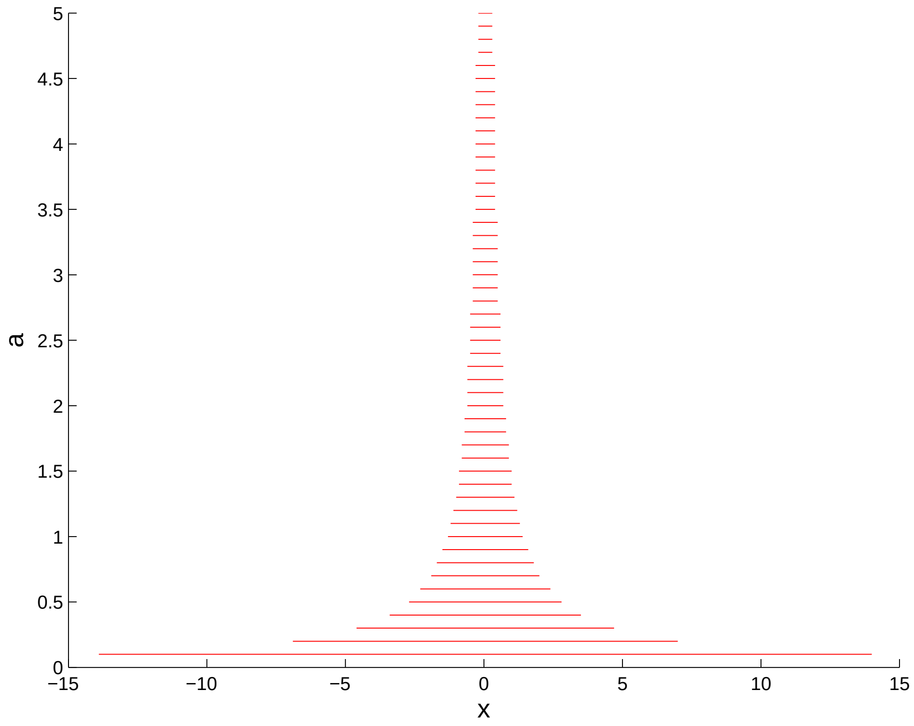
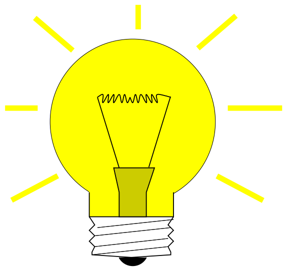


Fig. 24





red zone = $\{x^{(0)} \in \mathbb{R}, x^{(k)} \to 0\}$



▶ we observe "overshooting" of Newton correction

Idea: damping of Newton correction:

With $\lambda^{(k)} > 0$: $\quad \mathbf{x}^{(k+1)} := \mathbf{x}^{(k)} - \lambda^{(k)} DF(\mathbf{x}^{(k)})^{-1} F(\mathbf{x}^{(k)})$. (1.4.5)

Terminology: $\lambda^{(k)}$ = damping factor

Choice of damping factor: affine invariant natural monotonicity test

$$\text{“maximal”} \quad \lambda^{(k)} > 0: \qquad \left\| \Delta\overline{\mathbf{x}}(\lambda^{(k)}) \right\| \leq (1 - \frac{\lambda^{(k)}}{2}) \left\| \Delta\mathbf{x}^{(k)} \right\|_2 \tag{1.4.6}$$

where
$$\Delta\mathbf{x}^{(k)} := DF(\mathbf{x}^{(k)})^{-1} F(\mathbf{x}^{(k)}) \quad \rightarrow \text{current Newton correction ,}$$
$$\Delta\overline{\mathbf{x}}(\lambda^{(k)}) := DF(\mathbf{x}^{(k)})^{-1} F(\mathbf{x}^{(k)} + \lambda^{(k)} \Delta\mathbf{x}^{(k)}) \quad \rightarrow \text{tentative simplified Newton correction .}$$

Heuristics: convergence $\Leftrightarrow$ size of Newton correction decreases

Code 1.4.13: Damped Newton method

```
from scipy.linalg import lu_solve, lu_factor, norm
from scipy import array, arctan, exp

def dampnewton(x,F,DF,q=0.5,tol=1e-10):
    cvg = []
    lup = lu_factor(DF(x))
    s = lu_solve(lup,F(x))
    xn = x-s
    lam = 1
    st = lu_solve(lup,F(xn))
    while norm(st) > tol*norm(xn): #a posterori termination criteria
        while norm(st) > (1-lam*0.5)*norm(s): #natural monotonicity test
            lam *= 0.5 # reduce damping factor
            if lam < 1e-10:
                cvg = -1
                print 'DAMPED NEWTON: Failure of convergence'
```

```
            return x, cvg
        xn = x-lam*s
        st = lu_solve(lup,F(xn)) #simplified Newton cf. Sect. 1.4.3
    cvg += [[lam, norm(xn), norm(F(xn))]]
    x = xn
    lup = lu_factor(DF(x))
    s = lu_solve(lup,F(x))
    lam = min(lam/q, 1.)
    xn = x-lam*s
    st = lu_solve(lup,F(xn)) #simplified Newton cf. Sect. 1.4.3
  x = xn
  return x, array(cvg)

if __name__=='__main__':
  print '----------------- 2D F -----------------'
  F = lambda x:  array([x[0]**2 - x[1]**4, x[0]-x[1]**3])
  DF = lambda x: array([[2*x[0], - 4*x[1]**3], [1, -3*x[1]**2]])
  x = array([0.7, 0.7])
  x0 = array([1., 1.])
  x, cvg = dampnewton(x,F,DF)
  print x
  print cvg

  print '----------------- arctan -----------------'
```

```python
F = lambda x:  array([arctan(x[0])])
DF = lambda x: array([[1./(1.+x[0]**2)]])
x = array([20.])
x, cvg = dampnewton(x,F,DF)
print x
print cvg

print '-------------------- x e^x - 1 --------------------'
F = lambda x:  array([x[0]*exp(x[0]) -1.])
DF = lambda x: array([[exp(x[0])*(x[0]+1.)]])
x = array([-1.5])
x, cvg = dampnewton(x,F,DF)
print x
print cvg
```

Policy: Reduce damping factor by a factor $q \in]0,1[$ (usually $q = \frac{1}{2}$) until the affine invariant natural monotonicity test (1.4.6) passed.

Example 1.4.14 (Damped Newton method). (⟶ Ex. 1.4.12)

$F(x) = \arctan(x)$,

- $x^{(0)} = 20$
- $q = \frac{1}{2}$
- `LMIN = 0.001`

Observation: asymptotic quadratic convergence

k	$\lambda^{(k)}$	$x^{(k)}$	$F(x^{(k)})$
1	0.03125	0.94199967624205	0.75554074974604
2	0.06250	0.85287592931991	0.70616132170387
3	0.12500	0.70039827977515	0.61099321623952
4	0.25000	0.47271811131169	0.44158487422833
5	0.50000	0.20258686348037	0.19988168667351
6	1.00000	-0.00549825489514	-0.00549819949059
7	1.00000	0.00000011081045	0.00000011081045
8	1.00000	-0.00000000000001	-0.00000000000001

◇

Example 1.4.15 (Failure of damped Newton method).

- As in Ex. 1.4.11:
  $F(x) = xe^x - 1$,
- Initial guess for damped Newton method $x^{(0)} = -1.5$

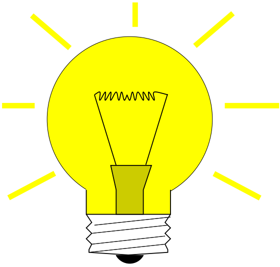
k	$\lambda^{(k)}$	$x^{(k)}$	$F(x^{(k)})$
1	0.25000	-4.4908445351690	-1.0503476286303
2	0.06250	-6.1682249558799	-1.0129221310944
3	0.01562	-7.6300006580712	-1.0037055902301
4	0.00390	-8.8476436930246	-1.0012715832278
5	0.00195	-10.5815494437311	-1.0002685596314
	Bailed out because of `lambda < LMIN`!		

Observation: Newton correction pointing in “wrong direction” so no convergence.

◇

1.4.5 Quasi-Newton Method

What to do when $DF(\mathbf{x})$ is not available and numerical differentiation (see remark 1.4.6) is too expensive**?**



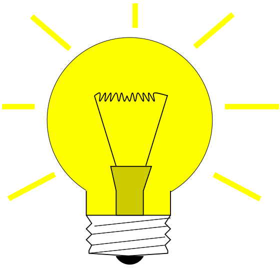
Idea: in one dimension ($n = 1$) apply the secant method (1.3.4) of section 1.3.2.3

$$F'(x^{(k)}) \approx \frac{F(x^{(k)}) - F(x^{(k-1)})}{x^{(k)} - x^{(k-1)}} \quad \text{"difference quotient"} \tag{1.4.7}$$

already computed **!** $\longrightarrow$ cheap

Generalisation for $n > 1$ **?**

Gradinaru
D-MATH



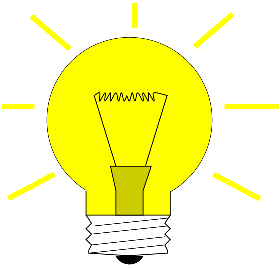
Idea: rewrite (1.4.7) as a secant condition for the approximation $\mathbf{J}_k \approx DF(\mathbf{x}^{(k)})$, $\mathbf{x}^{(k)} \hat{=}$ iterate:

$$\mathbf{J}_k(\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}) = F(\mathbf{x}^{(k)}) - F(\mathbf{x}^{(k-1)}) \; . \tag{1.4.8}$$

BUT: many matrices $\mathbf{J}_k$ fulfill (1.4.8)

Hence: we need more conditions for $\mathbf{J}_k \in \mathbb{R}^{n,n}$

Idea: get $\mathbf{J}_k$ by a modification of $\mathbf{J}_{k-1}$



Broyden conditions: $\mathbf{J}_k\mathbf{z} = \mathbf{J}_{k-1}\mathbf{z} \quad \forall \mathbf{z}\colon \mathbf{z} \perp (\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)})$. (1.4.9)

i.e.:
$$\boxed{\mathbf{J}_k := \mathbf{J}_{k-1} + \frac{F(\mathbf{x}^{(k)})(\mathbf{x}^{(k)}-\mathbf{x}^{(k-1)})^T}{\left\|\mathbf{x}^{(k)}-\mathbf{x}^{(k-1)}\right\|_2^2}} \tag{1.4.10}$$

Broydens Quasi-Newton Method for solving $F(\mathbf{x}) = 0$:

$$\mathbf{x}^{(k+1)} := \mathbf{x}^{(k)} + \Delta\mathbf{x}^{(k)},\ \Delta\mathbf{x}^{(k)} := -\mathbf{J}_k^{-1}F(\mathbf{x}^{(k)})\ ,\ \mathbf{J}_{k+1} := \mathbf{J}_k + \frac{F(\mathbf{x}^{(k+1)})(\Delta\mathbf{x}^{(k)})^T}{\left\|\Delta\mathbf{x}^{(k)}\right\|_2^2} \tag{1.4.11}$$

Initialize $\mathbf{J}_0$ e.g. with the exact Jacobi matrix $DF(\mathbf{x}^{(0)})$.

Remark 1.4.16 (Minimal property of Broydens rank 1 modification).

Let $\mathbf{J} \in \mathbb{R}^{n,n}$ fulfill (1.4.8) and $\mathbf{J}_k$, $\mathbf{x}^{(k)}$ from (1.4.11) then $(\mathbf{I} - \mathbf{J}_k^{-1}\mathbf{J})(\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}) = -\mathbf{J}_k^{-1}F(\mathbf{x}^{(k+1)})$

Gradinaru
D-MATH

and hence

$$
\begin{aligned}
\left\|\mathbf{I}-\mathbf{J}_k^{-1}\mathbf{J}_{k+1}\right\|_2 &= \left\|\frac{-\mathbf{J}_k^{-1}F(\mathbf{x}^{(k+1)})\Delta\mathbf{x}^{(k)}}{\left\|\Delta\mathbf{x}^{(k)}\right\|_2^2}\right\|_2 = \left\|(\mathbf{I}-\mathbf{J}_k^{-1}\mathbf{J})\frac{\Delta\mathbf{x}^{(k)}(\Delta\mathbf{x}^{(k)})^T}{\left\|\Delta\mathbf{x}^{(k)}\right\|_2^2}\right\|_2 \\
&\leq \left\|\mathbf{I}-\mathbf{J}_k^{-1}\mathbf{J}\right\|_2 \ .
\end{aligned}
$$

In conlcusion,

(1.4.10) gives the $\|\cdot\|_2$-minimal relative correction of $\mathbf{J}_{k-1}$, such that the secant condition (1.4.8) holds.

△

Example 1.4.17 (Broydens Quasi-Newton Method: Convergence).

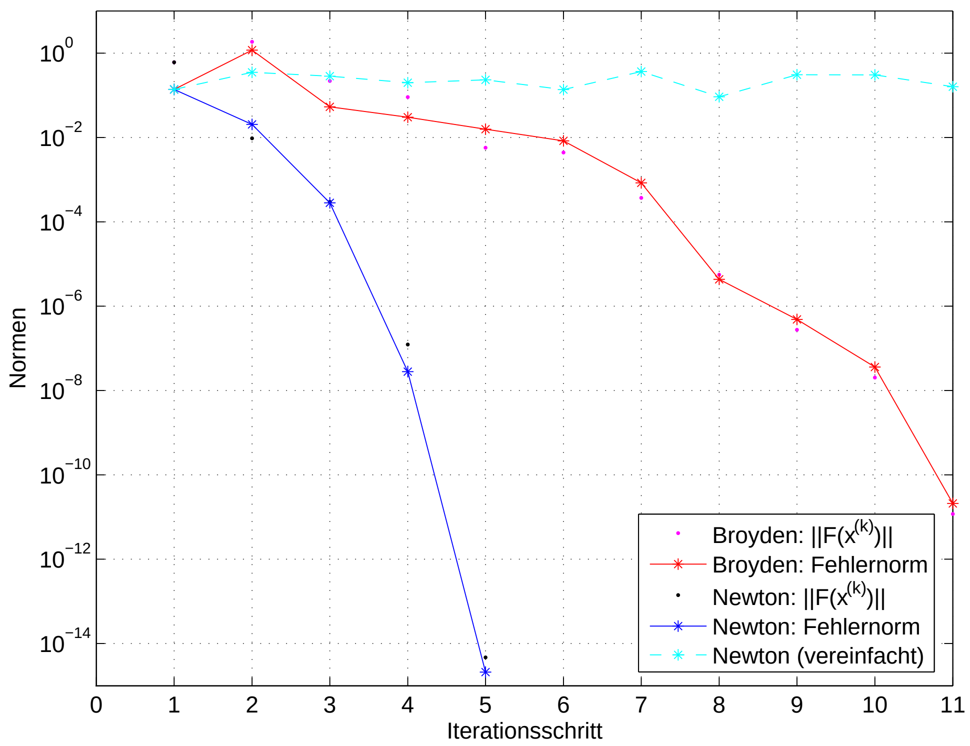
- In the non-linear system of the example 1.4.1, $n = 2$ take $\mathbf{x}^{(0)} = (0.7.0.7)^T$ and $\mathbf{J}_0 = DF(\mathbf{x}^{(0)})$

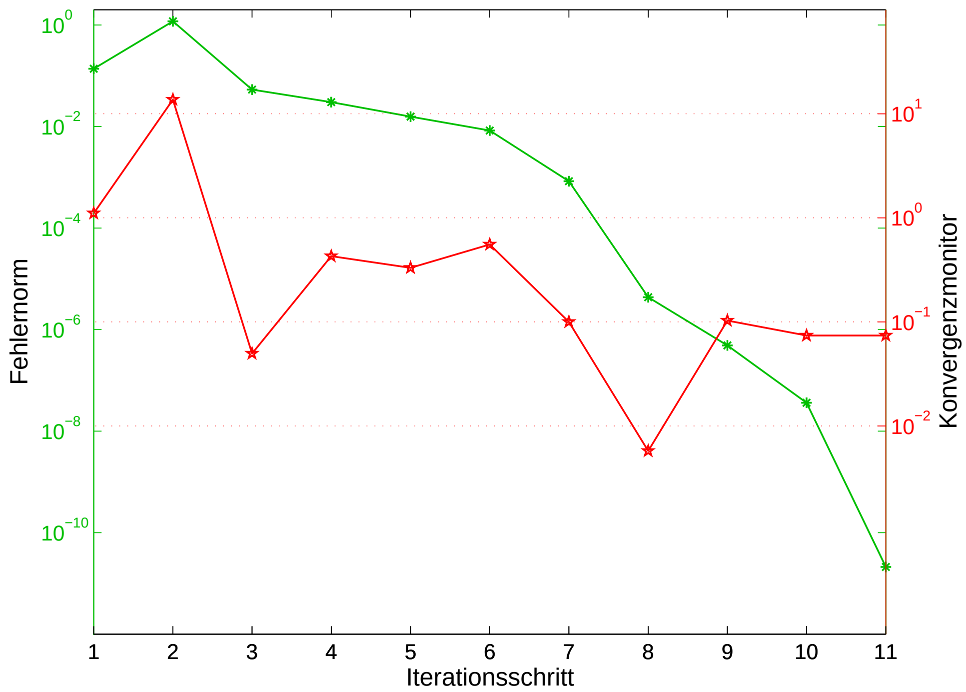
The numerical example shows that the method is:

slower than Newton method (1.4.1), but

better than simplified Newton method (see remark. 1.4.5)

$\triangleright$





convergence monitor

=

quantity that displays difficulties in the convergence of an iteration

Here:

$$\mu := \frac{\left\| \mathbf{J}_{k-1}^{-1} F(\mathbf{x}^{(k)}) \right\|}{\left\| \Delta \mathbf{x}^{(k-1)} \right\|}$$

Heuristics: no convergence whenever $\mu > 1$

◇

Remark 1.4.18. Option: damped Broyden method (as for the Newton method, section 1.4.4)

△

Implementation of (1.4.11): with Sherman-Morrison-Woodbury Update-Formula

$$\mathbf{J}_{k+1}^{-1} = \left(\mathbf{I} - \frac{\mathbf{J}_k^{-1} F(\mathbf{x}^{(k+1)}) (\Delta\mathbf{x}^{(k)})^T}{\left\| \Delta\mathbf{x}^{(k)} \right\|_2^2 + \Delta\mathbf{x}^{(k)} \cdot \mathbf{J}_k^{-1} F(\mathbf{x}^{(k+1)})} \right) \mathbf{J}_k^{-1} = \left(\mathbf{I} + \frac{\Delta\mathbf{x}^{(k+1)} (\Delta\mathbf{x}^{(k)})^T}{\left\| \Delta\mathbf{x}^{(k)} \right\|_2^2} \right) \mathbf{J}_k^{-1} \tag{1.4.12}$$

that makes sense in the case that

$$\left\| \mathbf{J}_k^{-1} F(\mathbf{x}^{(k+1)}) \right\|_2 < \left\| \Delta\mathbf{x}^{(k)} \right\|_2$$

"simplified Quasi-Newton correction"

Code 1.4.19: Broyden method

```
from scipy.linalg import lu_solve, lu_factor, norm, solve
from scipy import dot, zeros

def fastbroyd(x0, F, J, tol=1e-12, maxit=20):
    x = x0.copy()
    lup = lu_factor(J)
    k = 0; s = lu_solve(lup,F(x))
    x -= s; f = F(x); sn = dot(s,s)
    dx = zeros((maxit,len(x)))
    dxn = zeros(maxit)
    dx[k] = s; dxn[k] = sn
    k += 1; tol *= tol
    while sn > tol and k < maxit:
        w = lu_solve(lup,f)
```

```
        for r in xrange(1,k):
            w +=  dx[r]*( dot(dx[r-1],w) )/dxn[r-1]
        z = dot(s,w)
        s = (1+z/(sn-z))*w
        sn = dot(s,s)
        dx[k] = s; dxn[k] = sn
        x -= s; f = F(x); k+=1

    return x, k
```

Computational cost : N steps

- $O(N^2 \cdot n)$ operations with vectors, (Level I)
- 1 LU-decomposition of **J**, $N\times$ solutions of SLEs, see section **??**
- N evalutations of F **!**

Memory cost : N steps

- LU-factors of **J** + auxiliary vectors $\in \mathbb{R}^n$
- N vectors $\mathbf{x}^{(k)} \in \mathbb{R}^n$

Example 1.4.20 (Broyden method for a large non-linear system).

$$F(\mathbf{x}) = \begin{cases} \mathbb{R}^n & \mapsto \mathbb{R}^n \\ \mathbf{x} & \mapsto \operatorname{diag}(\mathbf{x})\mathbf{A}\mathbf{x} - \mathbf{b} \,, \end{cases}$$

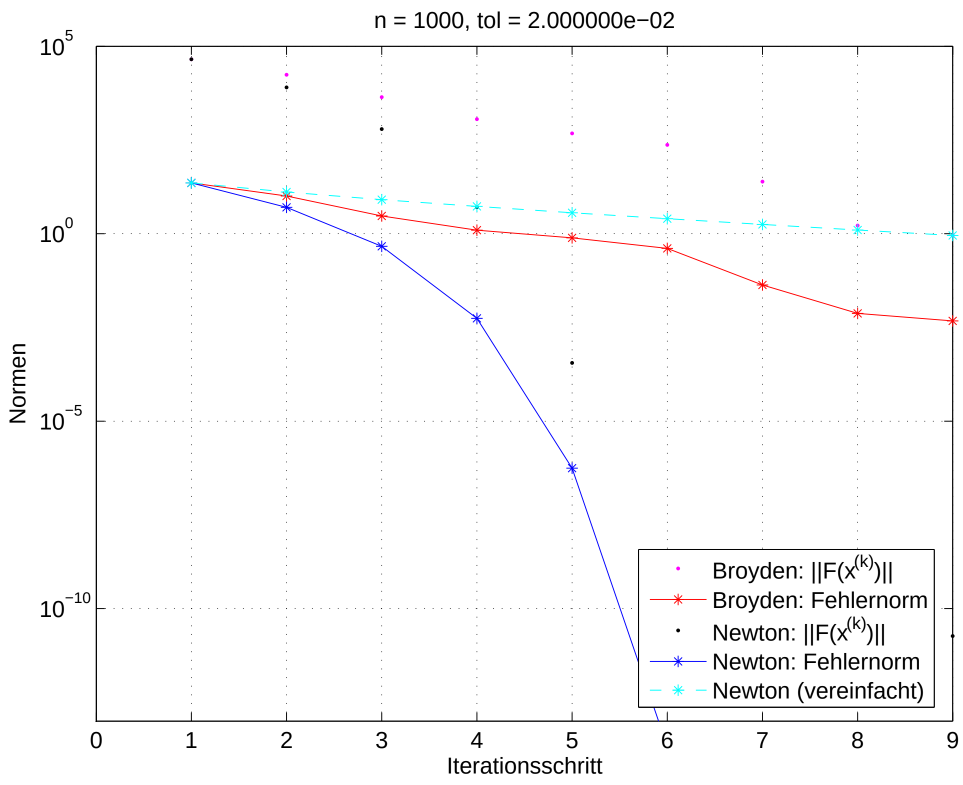
$$\mathbf{b} = (1, 2, \dots, n) \in \mathbb{R}^n \,,$$

$$\mathbf{A} = \mathbf{I} + \mathbf{a}\mathbf{a}^T \in \mathbb{R}^{n,n} \,,$$

$$\mathbf{a} = \frac{1}{\sqrt{\mathbf{1} \cdot \mathbf{b} - 1}}(\mathbf{b} - \mathbf{1}) \,.$$

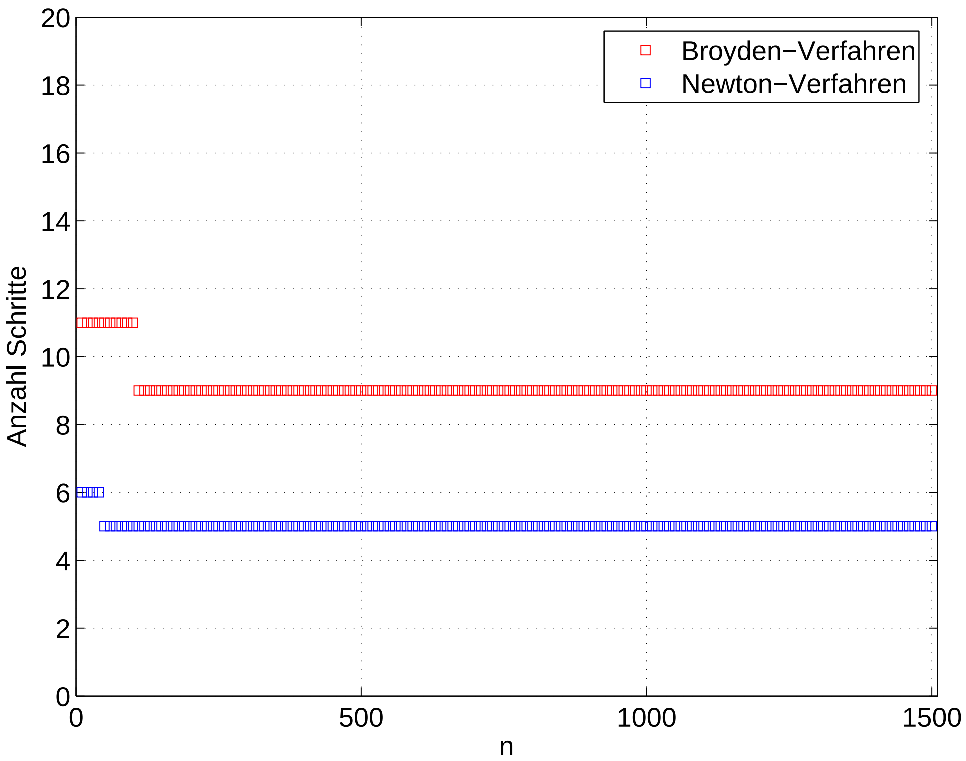
The interpretation of the results resemble the example 1.4.17 ▷

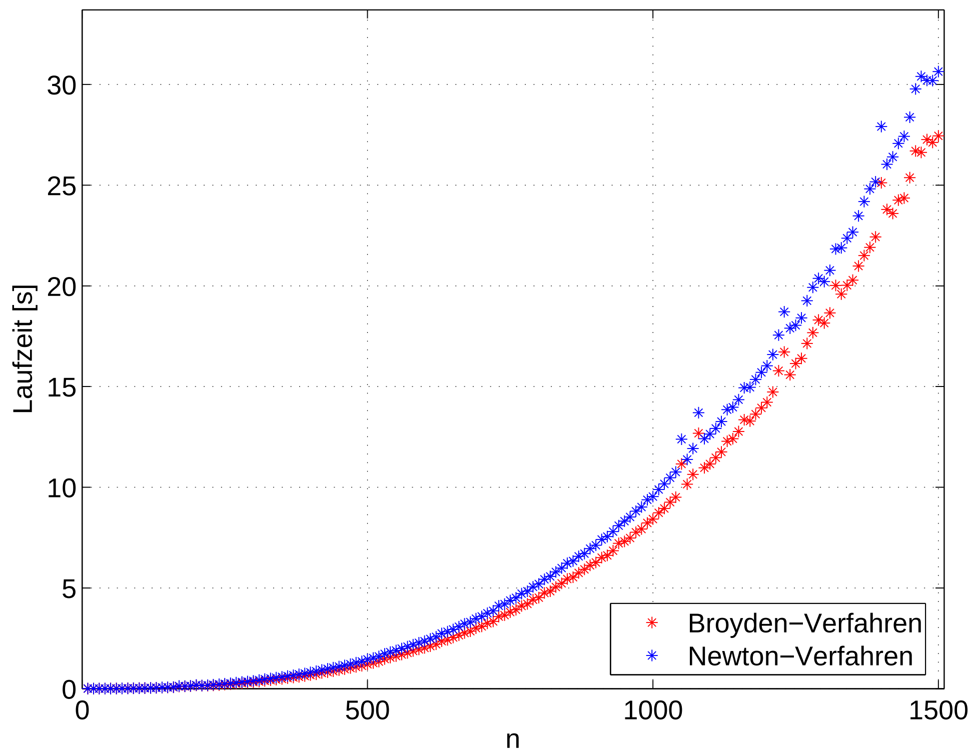
```
h = 2/n; x0 = (2:h:4-h)';
```



Efficiency comparison: Broyden method $\longleftrightarrow$ Newton method:

(in case of dimension n use tolerance $\texttt{tol} = 2n \cdot 10^{-5}$, `h = 2/n; x0 = (2:h:4-h)';`)





In conclusion,

the Broyden method is worthwile for dimensions $n \gg 1$ and low accuracy requirements.

◇

1.5 Essential Skills Learned in Chapter 1

You should know:

- what is a linear convergent iteration, its rate and dependence of the choice of the norm
- what is the the order of convergence and how to recognize it from plots or from error data
- possible termination criteria and their risks
- how to use fixed-point iterations; convergence criteria
- bisection-method: pros and contras
- Newton-iteration: pros and contras
- the idea behind multi-point methods and an example
- how to use the Newton-method in several dimensions and how to reduce its computational effort (simplified Newton, quasi-Newton, Broyden method)