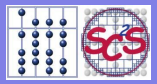


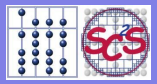
# 6. Iterative Verfahren: Nullstellen und Optima

*Besser, schneller, höher, weiter!*

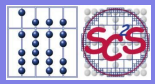
# 6.1. Große, schwach besetzte lineare Gleichungssysteme I – Relaxationsverfahren

## Einführung

- Numerisch zu lösende lineare Gleichungssysteme stammen oftmals von der Diskretisierung gewöhnlicher (bei Randwertproblemen, siehe Kapitel 5) oder partieller Differentialgleichungen. Die in Kapitel 4 studierten **direkten** Lösungsverfahren kommen hierfür in der Regel nicht infrage:
  - Erstens ist $n$ meistens so groß (i.A. ist $n$ ja direkt mit der Zahl der Gitterpunkte korreliert, was insb. bei instationären partiellen Differentialgleichungen (drei Orts- und eine Zeitvariable) zu sehr großem $n$ führt), dass ein Rechenaufwand der Größenordnung $O(n^3)$ nicht akzeptabel ist.
  - Zweitens sind solche Matrizen in der Regel **dünn besetzt** (d.h. $O(1)$ oder allenfalls $O(\log n)$ Nicht-Nullen pro Zeile) und weisen eine bestimmte **Struktur** auf (tridiagonal, Bandstruktur, Blockstruktur etc.), was sich natürlich Speicher und Rechenzeit mindernd auswirkt; Eliminationsverfahren zerstören diese Struktur typischerweise und machen die Vorteile so zunichte.
- Für große und dünn besetzte Matrizen bzw. lineare Gleichungssysteme werden deshalb **iterative** Verfahren vorgezogen:
  - Diese beginnen mit einer *Startnäherung* $x^{(0)}$ und erzeugen daraus eine Folge von Näherungswerten $x^{(i)}$, $i = 1, 2, \ldots$, die im Konvergenzfall gegen die exakte Lösung $x$ konvergieren.



– Ein Iterationsschritt kostet bei dünn besetzten Matrizen typischerweise $O(n)$ Rechenoperationen (weniger geht kaum, wenn man in jedem Schritt jede Vektorkomponente von $x^{(i)}$ updaten will). Somit wird es bei der Konstruktion iterativer Algorithmen darauf ankommen, wie viele Iterationsschritte benötigt werden, um eine bestimmte vorgegebene Genauigkeit zu erreichen.

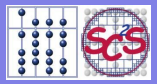


# Überblick über Relaxationsverfahren

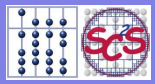
- Die wohl ältesten iterativen Verfahren zur Lösung linearer Gleichungssysteme $Ax = b$ mit $A \in \mathbb{R}^{n,n}$ und $x, b \in \mathbb{R}^n$ sind die sogenannten **Relaxations-** oder **Glättungsverfahren**:
  - das **Richardson**-Verfahren,
  - das **Jacobi**-Verfahren,
  - das **Gauß-Seidel**-Verfahren sowie
  - SOR, die **Successive Over Relaxation**.
- Grundprinzipien:
  - Für alle der genannten Verfahren ist der Ausgangspunkt das bereits eingeführte **Residuum** $r^{(i)}$,

$$r^{(i)} := b - Ax^{(i)} = Ax - Ax^{(i)} = A\left(x - x^{(i)}\right) = -Ae^{(i)},$$

    wobei $x^{(i)}$ die aktuelle Näherung an die exakte Lösung $x$ nach $i$ Iterationsschritten und $e^{(i)}$ den jeweiligen Fehler bezeichnen.
  - Weil $e^{(i)}$ nicht verfügbar ist (der Fehler kann ohne Kenntnis der exakten Lösung $x$ nicht ermittelt werden), erweist es sich aufgrund obiger Beziehung als vernünftig, den Vektor $r^{(i)}$ als *Richtung* zu nehmen, in der wir nach einer Verbesserung von $x^{(i)}$ suchen wollen.



– Das Richardson-Verfahren nimmt das Residuum direkt als Korrektur für $x^{(i)}$. Mehr Mühe geben sich das Jacobi- und das Gauß-Seidel-Verfahren, deren Idee für die Korrektur der $k$-ten Komponente von $x^{(i)}$ die Elimination von $r_k^{(i)}$ ist. Das SOR-Verfahren bzw. sein Pendant, die **gedämpfte** Relaxation, berücksichtigen zusätzlich, dass eine solche Korrektur oft über das Ziel hinausschießt bzw. nicht ausreicht.



# Wichtige Relaxationsverfahren

- **Richardson-Iteration**:

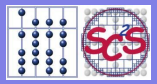
```
for i = 0,1,...
        for k = 1,...,n:
```
$$x_k^{(i+1)} := x_k^{(i)} + r_k^{(i)}$$

  Hier wird einfach das Residuum $r^{(i)}$ komponentenweise als Korrektur für die aktuelle Näherung $x^{(i)}$ herangezogen.

- **Jacobi-Iteration**:

```
for i = 0,1,...
        for k = 1,...,n:
        for k = 1,...,n:
```
$$y_k := \frac{1}{a_{kk}} \cdot r_k^{(i)}$$
$$x_k^{(i+1)} := x_k^{(i)} + y_k$$

  - In jedem Teilschritt $k$ eines Schritts $i$ wird eine Korrektur $y_k$ berechnet und gespeichert.
  - Sofort angewendet, würde diese zum (momentanen) Verschwinden der $k$-Komponente des Residuums $r^{(i)}$ führen (leicht durch Einsetzen zu verifizieren).
  - Gleichung $k$ wäre mit dieser aktuellen Näherung für $x$ somit exakt gelöst – ein Fortschritt, der im folgenden Teilschritt zu Gleichung $k+1$ natürlich gleich wieder verloren ginge.
  - Allerdings werden diese Komponentenkorrekturen nicht sofort, sondern erst am Ende eines Iterationsschritts durchgeführt (zweite $k$-Schleife).



# Wichtige Relaxationsverfahren (2)

- **Gauß-Seidel-Iteration**:

```
for i = 0,1,...
    for k = 1,...,n:
```

$$r_k^{(i)} := b_k - \sum_{j=1}^{k-1} a_{kj} x_j^{(i+1)} - \sum_{j=k}^{n} a_{kj} x_j^{(i)}$$

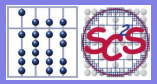
$$y_k := \frac{1}{a_{kk}} \cdot r_k^{(i)}, \quad x_k^{(i+1)} := x_k^{(i)} + y_k$$

  - Hier wird also dieselbe Korrektur wie beim Jacobi-Verfahren berechnet, der Update wird jetzt allerdings immer sofort und nicht erst am Ende des Iterationsschritts vollzogen.
  - Damit liegen beim Update von Komponente $k$ für die Komponenten $1$ bis $k-1$ bereits die modifizierten neuen Werte vor.

- Manchmal führt in jeder der drei skizzierten Methoden ein **Dämpfen** (Multiplikation der Korrektur mit einem Faktor $0 < \alpha < 1$) bzw. eine **Überrelaxation** (Faktor $1 < \alpha < 2$) zu einem besseren Konvergenzverhalten:

$$x_k^{(i+1)} := x_k^{(i)} + \alpha y_k \,.$$

  - Im Gauß-Seidel-Fall ist vor allem die Version mit $\alpha > 1$ gebräuchlich, man spricht hier von **SOR-Verfahren** (**Successive Over Relaxation**).
  - Im Jacobi-Fall wird dagegen meistens gedämpft.



# Diskussion: Additive Zerlegung der Systemmatrix

- Für eine kurze Konvergenzanalyse der obigen Verfahren benötigen wir eine algebraische Formulierung (anstelle der algorithmischen).
- Alle gezeigten Ansätze basieren auf der einfachen Idee, die Matrix $A$ als Summe $A = M + (A - M)$ zu schreiben, wobei $Mx = b$ sehr einfach zu lösen und der Unterschied $A - M$ bzgl. einer Matrixnorm nicht zu groß sein sollte.
- Mit Hilfe eines solchen geeigneten $M$ werden sich Richardson-, Jacobi-, Gauß-Seidel- und SOR-Verfahren schreiben lassen als

$$Mx^{(i+1)} + (A - M)x^{(i)} = b$$

bzw., nach $x^{(i+1)}$ aufgelöst,

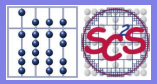
$$x^{(i+1)} := M^{-1}b - M^{-1}(A-M)x^{(i)} = M^{-1}b - (M^{-1}A - I)x^{(i)} = x^{(i)} + M^{-1}r^{(i)} .$$

- Darüber hinaus zerlegen wir $A$ additiv in seinen Diagonalteil $D_A$, seinen strikten unteren Dreiecksteil $L_A$ sowie seinen strikten oberen Dreiecksteil $U_A$:

$$A =: L_A + D_A + U_A .$$

Damit können wir die folgenden Beziehungen zeigen:

  - Richardson: $M := I$ ,
  - Jacobi: $M := D_A$ ,
  - Gauß-Seidel: $M := D_A + L_A$ ,
  - SOR: $M := \frac{1}{\alpha} D_A + L_A$ .

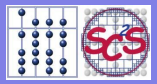


# Diskussion: Additive Zerlegung der Systemmatrix (2)

- Bei Betrachtung der algorithmischen Formulierungen für das Richardson- sowie das Jacobi-Verfahren erweisen sich die ersten beiden Gleichungen als offensichtlich:
  - Bei Richardson wird das Residuum direkt als Korrektur genommen, als Vorfaktor ergibt sich somit die Identität $I$.
  - Bei Jacobi wird das Residuum durch das Diagonalelement dividiert, als Vorfaktor ergibt sich somit die Inverse des Diagonalanteils $D_A$.
- Weil die Gauß-Seidel-Iteration ein Spezialfall des SOR-Verfahrens ist ($\alpha = 1$), reicht es aus, obige Formel für $M$ für den allgemeinen SOR-Fall zu zeigen. Aus dem Algorithmus folgt unmittelbar

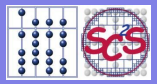
$$
\begin{aligned}
& x_k^{(i+1)} := x_k^{(i)} + \alpha \left( b_k - \sum_{j=1}^{k-1} a_{kj} x_j^{(i+1)} - \sum_{j=k}^{n} a_{kj} x_j^{(i)} \right) / a_{kk} \\
\Leftrightarrow \quad & x^{(i+1)} := x^{(i)} + \alpha D_A^{-1} \left( b - L_A x^{(i+1)} - (D_A + U_A) x^{(i)} \right) \\
\Leftrightarrow \quad & \frac{1}{\alpha} D_A x^{(i+1)} = \frac{1}{\alpha} D_A x^{(i)} + b - L_A x^{(i+1)} - (D_A + U_A) x^{(i)} \\
\Leftrightarrow \quad & \left( \frac{1}{\alpha} D_A + L_A \right) x^{(i+1)} + \left( (1 - \frac{1}{\alpha}) D_A + U_A \right) x^{(i)} = b \\
\Leftrightarrow \quad & M x^{(i+1)} + (A - M) x^{(i)} = b \,,
\end{aligned}
$$

  womit die Behauptung für das SOR-Verfahren bewiesen ist.



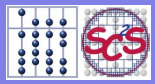
# Diskussion: Allgemeines Konvergenzverhalten

- Was die Konvergenz angeht, so gibt es zwei unmittelbare Konsequenzen aus dem Ansatz
$$Mx^{(i+1)} + (A - M)x^{(i)} = b :$$
  - Falls die Folge $\left(x^{(i)}\right)$ konvergiert, dann ist der Grenzwert die exakte Lösung $x$ unseres Systems $Ax = b$.
  - Für die Analyse werde angenommen, dass die Iterationsmatrix $-M^{-1}(A-M)$ (d.h. die Matrix, die auf $e^{(i)}$ angewandt wird, um $e^{(i+1)}$ zu erhalten; s.u.) symmetrisch sei. Dann ist der Spektralradius $\rho$ (d.h. der betragsgrößte Eigenwert) die für das Konvergenzverhalten entscheidende Größe:
$$\left(\forall x^{(0)} \in \mathbb{R}^n : \lim_{i\to\infty} x^{(i)} = x = A^{-1}b\right) \quad \Leftrightarrow \quad \rho < 1 .$$
Um das zu sehen, subtrahiere man $Mx + (A - M)x = b$ von der Gleichung ganz oben:
$$Me^{(i+1)} + (A - M)e^{(i)} = 0 \quad \Leftrightarrow \quad e^{(i+1)} = -M^{-1}(A - M)e^{(i)} .$$
Wenn alle Eigenwerte betragsmäßig kleiner 1 sind und somit $\rho < 1$ gilt, werden alle Fehlerkomponenten in jedem Iterationsschritt reduziert. Im Falle $\rho > 1$ wird sich mindestens eine Fehlerkomponente aufschaukeln.
  - Ziel bei der Konstruktion iterativer Verfahren muss natürlich ein möglichst kleiner Spektralradius der Iterationsmatrix sein (möglichst nahe bei Null).



# Diskussion: Konvergenzaussagen

- Es gibt eine Reihe von Resultaten zur Konvergenz der verschiedenen Verfahren, von denen einige bedeutende erwähnt werden sollen:
  - Notwendig für die Konvergenz des SOR-Verfahrens ist $0 < \alpha < 2$.
  - Falls $A$ positiv definit ist, dann konvergieren sowohl das SOR-Verfahren (für $0 < \alpha < 2$) als auch die Gauß-Seidel-Iteration.
  - Falls $A$ und $2D_A - A$ beide positiv definit sind, dann konvergiert das Jacobi-Verfahren.
  - Falls A strikt diagonal dominant ist (d. h. $a_{ii} > \sum_{j \neq i} \mid a_{ij} \mid$ für alle $i$), dann konvergieren das Jacobi- und das Gauß-Seidel-Verfahren.
  - In bestimmten Fällen lässt sich der optimale Parameter $\alpha$ bestimmten ($\rho$ minimal, so dass Fehlerreduktion pro Iterationsschritt maximal).
- Die Gauß-Seidel-Iteration ist nicht generell besser als das Jacobi-Verfahren (wie man aufgrund des sofort vollzogenen Updates vermuten könnte). Es gibt Beispiele, in denen Erstere konvergiert und Letzteres divergiert, und umgekehrt. In vielen Fällen kommt das Gauß-Seidel-Verfahren jedoch mit der Hälfte der Iterationsschritte des Jacobi-Verfahrens aus.

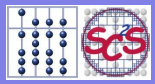


# Zum Spektralradius typischer Iterationsmatrizen

- Offensichtlich ist $\rho$ nicht nur entscheidend für die Frage, ob die Iterationsvorschrift überhaupt konvergiert, sondern auch für deren Qualität, also ihre Konvergenzgeschwindigkeit: Je kleiner $\rho$ ist, desto schneller werden alle Komponenten des Fehlers $e^{(i)}$ in jedem Iterationsschritt reduziert.
- In der Praxis haben die obigen Resultate zur Konvergenz leider eher theoretischen Wert, da $\rho$ oft so nahe bei 1 ist, dass – trotz Konvergenz – die Anzahl der erforderlichen Iterationsschritte, bis eine hinreichende Genauigkeit erreicht ist, viel zu groß ist.
- Ein wichtiges Beispielszenario ist die Diskretisierung partieller Differentialgleichungen:
  - Typisch ist, dass $\rho$ von der Problemgröße $n$ und somit von der Auflösung $h$ des zugrunde liegenden Gitters abhängt, also beispielsweise

$$\rho \;=\; O(1-h_l^2) \;=\; O\left(1-\frac{1}{4^l}\right)$$

    bei einer Maschenweite $h_l = 2^{-l}$.
  - Dies ist ein gewaltiger Nachteil: Je feiner und folglich auch genauer unser Gitter ist, umso erbärmlicher wird das Konvergenzverhalten unserer iterativen Verfahren. Bessere iterative Löser (z.B. Mehrgitterverfahren, die aber den Rahmen dieser Vorlesung sprengen würden) sind also ein Muss!



# 6.2. Große, schwach besetzte lineare Gleichungssysteme II – Minimierungsverfahren

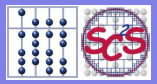
## Formulierung als Minimierungsproblem

- Eines der bekanntesten Lösungsverfahren, die Methode der **konjugierten Gradienten (cg)**, beruht auf einem anderen Prinzip als der Relaxation bzw. Glättung. Um dies zu sehen, nähern wir uns der Problematik der Lösung linearer Gleichungssysteme auf einem Umweg.
- Im Folgenden sei $A \in \mathbb{R}^{n,n}$ eine symmetrische und positiv definite Matrix, also $A = A^T$ und $x^T Ax > 0 \; \forall x \neq 0$. In diesem Fall ist die Lösung des linearen Systems $Ax = b$ äquivalent zur Minimierung der quadratischen Funktion

$$f(x) \; := \; \frac{1}{2} x^T Ax - b^T x + c$$

  für eine beliebige skalare Konstante $c \in \mathbb{R}$.
- Weil $A$ positiv definit ist, definiert die durch $z := f(x)$ gebildete Hyperfläche ein Paraboloid im $\mathbb{R}^{n+1}$ mit $n$-dimensionalen Ellipsoiden als Isoflächen $f(x) = const.$, und $f$ hat ein globales Minimum in $x$. Die Äquivalenz der Probleme ist offenkundig:

$$f'(x) \; = \; \frac{1}{2} A^T x + \frac{1}{2} Ax - b \; = \; Ax - b \; = \; -r(x) \; = \; 0 \quad \Leftrightarrow \quad Ax = b \, .$$

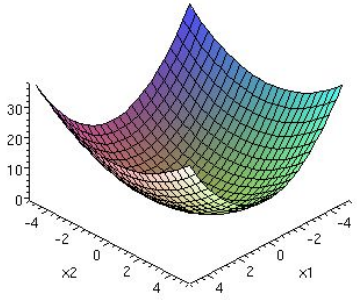


Anschaulich: Jede Störung $e \neq 0$ der Lösung $x$ von $Ax = b$ erhöht den Wert von $f$, der Punkt $x$ ist also tatsächlich Minimalstelle von $f$:

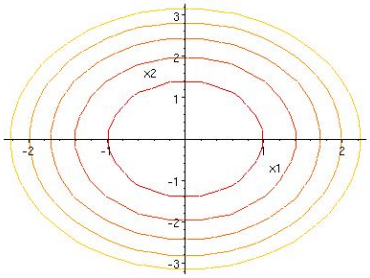
$$\begin{aligned} f(x+e) &= \frac{1}{2}x^T Ax + e^T Ax + \frac{1}{2}e^T Ae - b^T x - b^T e + c \\ &= f(x) + e^T(Ax - b) + \frac{1}{2}e^T Ae \\ &= f(x) + \frac{1}{2}e^T Ae \; > \; f(x)\,. \end{aligned}$$

- Beispiel: $n = 2, A = \begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix}, \; b = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \; c = 0$

$$f(x) = x_1^2 + \frac{1}{2}x_2^2$$




Graph von $f(x)$




Höhenlinien (Isolinien) $f(x) = c$

# Die Methode des steilsten Abstiegs

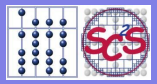
- Was bringt die Umformulierung? Man kann jetzt auch *Optimierungsmethoden* einsetzen und somit das Spektrum möglicher Lösungsverfahren erweitern.
- Betrachten wir also Techniken der Minimumsuche. Eine nahe liegende Möglichkeit liefert die **Methode des steilsten Abstiegs (steepest descent)**. Für $i = 0, 1, \ldots$, wiederhole

$$\begin{aligned} r^{(i)} &:= b - Ax^{(i)}, \\ \alpha_i &:= \frac{{r^{(i)}}^T r^{(i)}}{{r^{(i)}}^T A r^{(i)}}, \\ x^{(i+1)} &:= x^{(i)} + \alpha_i r^{(i)}, \end{aligned}$$

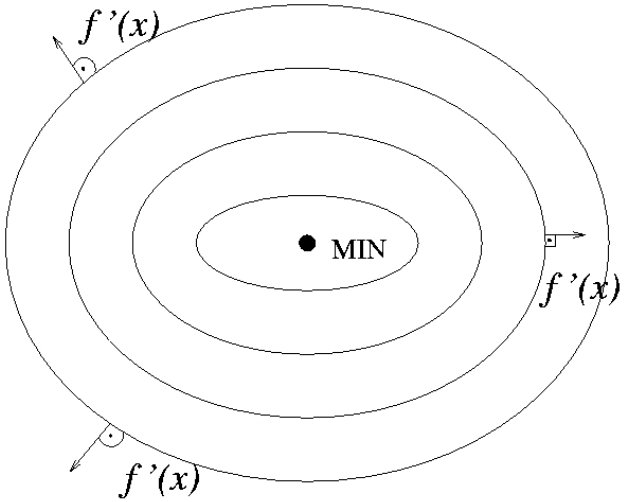
oder beginne mit $r^{(0)} := b - Ax^{(0)}$ und wiederhole für $i = 0, 1, \ldots$

$$\begin{aligned} \alpha_i &:= \frac{{r^{(i)}}^T r^{(i)}}{{r^{(i)}}^T A r^{(i)}}, \\ x^{(i+1)} &:= x^{(i)} + \alpha_i r^{(i)}, \\ r^{(i+1)} &:= r^{(i)} - \alpha_i A r^{(i)}, \end{aligned}$$

was eines der beiden Matrix-Vektor-Produkte (der einzige wirklich teure Schritt im Algorithmus) erspart.

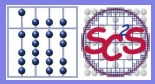


- Die Methode des steilsten Abstiegs sucht nach einer Verbesserung in Richtung des negativen Gradienten $-f'(x^{(i)}) = r^{(i)}$, der in der Tat den steilsten Abstieg anzeigt (daher der Name). Besser wäre natürlich, in Richtung des Fehlers $e^{(i)} := x^{(i)} - x$ zu suchen, aber den kennt man ja leider nicht.
- Doch die Richtung allein genügt nicht, man braucht auch eine passende Marschierweite. Dazu suchen wir das Minimum von $f(x^{(i)} + \alpha_i r^{(i)})$ als Funktion von $\alpha_i$ (partielle Ableitung nach $\alpha_i$ auf Null setzen), was nach kurzer Rechnung den obigen Wert für $\alpha_i$ liefert.



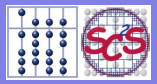
# Diskussion des steilsten Abstiegs

- Wenn man statt der Residuen $r^{(i)}$ alternierend die Einheitsvektoren in den Koordinatenrichtungen als Suchrichtungen wählt und wieder bzgl. dieser Suchrichtungen die optimalen Schrittweiten bestimmt, landet man übrigens bei der Gauß-Seidel-Iteration. Es gibt also trotz aller Verschiedenheit Verwandtschaft zwischen dem Relaxations- und dem Minimierungsansatz!
- Das Konvergenzverhalten der Methode des steilsten Abstiegs ist bescheiden. Einer der wenigen trivialen Sonderfälle ist die Einheitsmatrix: Hier sind die Isoflächen Sphären, der Gradient zeigt stets in den Mittelpunkt (das Minimum), und man ist in einem Schritt am Ziel! Allgemein landet man zwar irgendwann beliebig nahe am Minimum, das kann aber auch beliebig lange dauern (weil wir immer wieder etwas vom bisher Erreichten kaputt machen können).
- Um diesen Missstand zu beheben, bleiben wir bei unserem Minimierungsansatz, halten aber nach besseren Suchrichtungen Ausschau. Wären alle Suchrichtungen orthogonal, und wäre der Fehler nach $i$ Schritten orthogonal zu allen bisherigen Suchrichtungen, dann könnte schon Erreichtes nie wieder verloren gehen, und nach höchstens $n$ Schritten wäre man im Minimum – wie bei einem direkten Löser. Aus diesem Grunde nennt man das cg-Verfahren und Derivate auch **semi-iterative Methoden**. Wir gehen an dieser Stelle aber nicht näher auf diese Verfahrensklasse ein (im Master muss es ja auch noch etwas zu tun geben ...).



# 6.3. Nichtlineare Gleichungen

## Einführende Bemerkungen



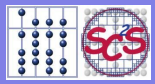
- Jetzt wissen wir also, wie man Systeme linearer Gleichungen löst, direkt oder iterativ. Die Lösung von $Ax = b$ kann man übrigens auch interpretieren als Nullstellensuche zur linearen Funktion $F : \mathbb{R}^n \to \mathbb{R}^n,\ F(x) := b - Ax$.
- Leider ist die Welt nicht linear, sondern nichtlinear. Deshalb muss sich ein Numeriker auch mit der Lösung **nichtlinearer Gleichungen** befassen. Das geht in aller Regel gar nicht mehr direkt, sondern nur noch iterativ. Da wir an dieser Stelle nicht die Analysis mehrerer Veränderlicher benutzen wollen, beschränken wir uns im Folgenden auf den einfachen Fall $n = 1$ (d.h. *eine* nichtlineare Gleichung).
- Betrachte also eine stetig differenzierbare (nichtlineare) Funktion $f : \mathbb{R} \to \mathbb{R}$, die eine Nullstelle $\bar{x} \in ]a, b[$ habe (man denkt dabei an Nullstellen- oder Extremalstellensuche).
- Gehe von einer (noch zu bestimmenden) Iterationsvorschrift aus:
  - Diese liefert eine Folge von Näherungswerten, $(x^{(i)})$, $i = 0, 1, \ldots$, die (hoffentlich) gegen die bzw. eine Nullstelle $\bar{x}$ von $f(x)$ konvergiert.

– Als Maß für die **Konvergenzgeschwindigkeit** betrachtet man die Reduktion des Fehlers in jedem Schritt und spricht im konvergenten Fall,

$$| x^{(i+1)} - \bar{x} | \ \leq \ c \cdot | x^{(i)} - \bar{x} |^{\alpha} ,$$

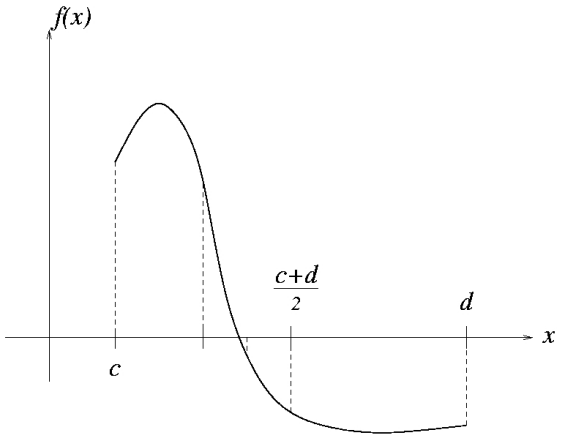
je nach maximal möglichem Wert des Parameters $\alpha$ von **linearer** ($\alpha = 1$ und zusätzlich $0 < c < 1$) oder **quadratischer** ($\alpha = 2$) Konvergenz usw.

– Es gibt **bedingt** oder **lokal** konvergente Verfahren, bei denen die Konvergenz nur bei Vorliegen eines bereits hinreichend guten Startwerts $x^{(0)}$ sichergestellt ist, und **unbedingt** oder **global** konvergente Verfahren, bei denen die Iteration unabhängig von der Wahl des Startpunkts zu einer Nullstelle von $f$ führt.

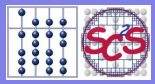


# Drei einfache Verfahren

- **Bisektionsverfahren**:
  - Gehe aus von $[c, d] \subseteq [a, b]$ mit $f(c) \cdot f(d) \leq 0$ – die Stetigkeit von $f$ garantiert die Existenz (mindestens) einer Nullstelle in $[c, d]$. Die Halbierung des Intervalls $[c, d]$ erlaubt – je nach Vorzeichen von $f((c + d)/2)$ – die Suche auf eines der Teilintervalle $[c, (c + d)/2]$ oder $[(c + d)/2, d]$ einzuschränken, usw.
  - Man bricht ab bei Erreichen einer Nullstelle bzw. bei Unterschreiten einer Toleranz $\varepsilon$ für die aktuelle Intervallbreite $d - c$.

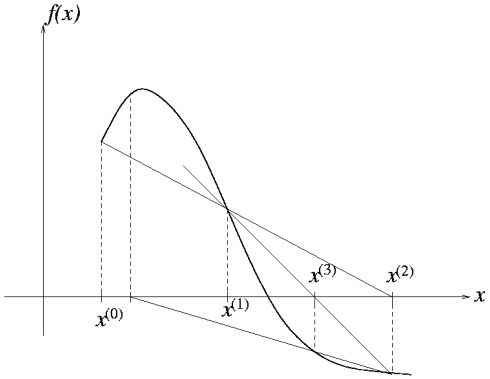


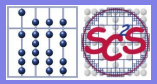
- **Regula falsi**:
  - Variante der Bisektion, bei der nicht die Intervallmitte, sondern der Nulldurchgang der Verbindung der beiden Punkte $(c, f(c))$ und $(d, f(d))$ als ein Endpunkt des neuen und kleineren Intervalls gewählt wird (kann auch langsamer als die einfache Bisektion sein!).



- **Sekantenverfahren**:
  - Beginne mit *zwei* Startnäherungen $x^{(0)}$ und $x^{(1)}$; im Folgenden wird dann $x^{(i+1)}$ aus $x^{(i-1)}$ und $x^{(i)}$ bestimmt, indem man die Nullstelle der Geraden durch $(x^{(i-1)}, f(x^{(i-1)}))$ und $(x^{(i)}, f(x^{(i)}))$ (der *Sekante* $s(x)$) sucht:

$$
\begin{aligned}
s(x) &:= f(x^{(i)}) + \frac{f(x^{(i)}) - f(x^{(i-1)})}{x^{(i)} - x^{(i-1)}} \cdot (x - x^{(i)}), \\
0 = s(x^{(i+1)}) &= f(x^{(i)}) + \frac{f(x^{(i)}) - f(x^{(i-1)})}{x^{(i)} - x^{(i-1)}} \cdot (x^{(i+1)} - x^{(i)}), \\
x^{(i+1)} &:= x^{(i)} - (x^{(i)} - x^{(i-1)}) \cdot \frac{f(x^{(i)})}{f(x^{(i)}) - f(x^{(i-1)})}.
\end{aligned}
$$



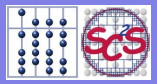



# Das Newton-Verfahren, Bemerkungen zur Konvergenzgeschwindigkeit

- **Newton-Verfahren**:
  - Hier beginnt man mit *einer* Startnäherung $x^{(0)}$ und bestimmt dann im Folgenden $x^{(i+1)}$ aus $x^{(i)}$, indem man die Nullstelle der *Tangente* $t(x)$ an $f(x)$ im Punkt $x^{(i)}$ sucht (**Linearisierung**: ersetze $f$ durch seine Tangente bzw. durch sein Taylorpolynom ersten Grades):

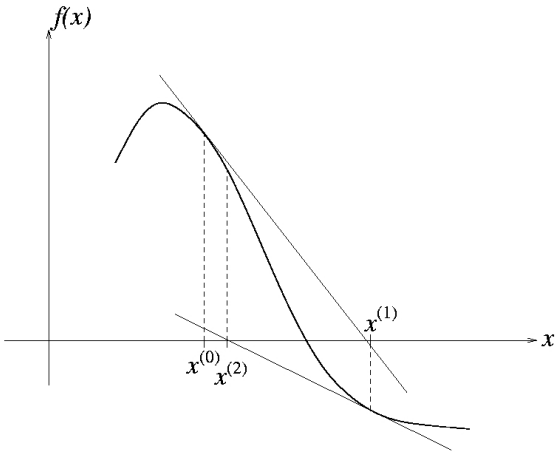
$$\begin{aligned} t(x) &:= f(x^{(i)}) + f'(x^{(i)}) \cdot (x - x^{(i)}), \\ 0 = t(x^{(i+1)}) &= f(x^{(i)}) + f'(x^{(i)}) \cdot (x^{(i+1)} - x^{(i)}), \\ x^{(i+1)} &:= x^{(i)} - \frac{f(x^{(i)})}{f'(x^{(i)})}. \end{aligned}$$

  - Das Newton-Verfahren entspricht dem Sekantenverfahren im Grenzfall $x^{(i-1)} = x^{(i)}$.
- **Konvergenzordnung** der vorgestellten Verfahren:
  - global linear für Bisektion und regula falsi,
  - lokal quadratisch für Newton,
  - lokal 1.618 für das Sekantenverfahren.
  - Da ein Newton-Schritt aber je eine Auswertung von $f$ und $f'$ erfordert, ist er mit *zwei* Schritten der anderen Verfahren zu vergleichen. Zudem ist die Berechnung der Ableitung oft ein Problem.



– Das Sekantenverfahren steht somit sehr gut da!

- Schaubild zum Newton-Verfahren:




- **Bemerkung**: Für Nullstellen von Polynomen höheren Grades (ein oft extrem schlecht konditioniertes Problem) gibt es spezielle Verfahren.

# Systeme nichtlinearer Gleichungen

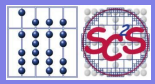
- In den meisten Anwendungen in der Praxis ist $n \gg 1$ (Unbekannte sind ja bspw. Funktionswerte an Gitterpunkten!), so dass wir es also mit sehr großen nichtlinearen Gleichungssystemen zu tun haben.
- Im mehrdimensionalen Fall tritt an die Stelle der einfachen Ableitung $f'(x)$ die sogenannte **Jacobi-Matrix** $F'(x)$, die Matrix der partiellen Ableitungen aller Vektorkomponenten von $F$ nach allen Variablen.
- Die Newton-Iterationsvorschrift lautet somit

$$x^{(i+1)} := x^{(i)} - F'(x^{(i)})^{-1} F(x^{(i)}),$$

  wobei natürlich die Matrix $F'(x^{(i)})$ nicht invertiert, sondern das entsprechende lineare Gleichungssystem mit der rechten Seite $-F(x^{(i)})$ (direkt) gelöst wird:

berechne $F'(x)$;
zerlege $F'(x) := LR$;
löse $LRs = -F(x)$;
aktualisiere $x := x + s$;
werte aus $F(x)$;

- Das wiederholte Berechnen der Jacobi-Matrix ist sehr aufwändig, ja es ist oft nur näherungsweise möglich, da meist numerisch differenziert werden muss. Auch das direkte Lösen eines linearen Gleichungssystems in jedem Newton-Schritt ist teuer. Deshalb ist das Newton-Verfahren nur der Ausgangspunkt für eine Vielzahl algorithmischer Entwicklungen gewesen.



# Vereinfachungen

- **Newton-chord-** bzw. **Shamanskii-Methode**:
  Hier wird die Jacobi-Matrix nicht in jedem Newton-Schritt berechnet und invertiert, sondern es wird immer $F'(x^{(0)})$ verwendet (chord) bzw. ein $F'(x^{(i)})$ immer für mehrere Newton-Schritte benutzt (Shamanskii).

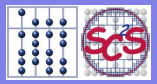
- **inexaktes Newton-Verfahren**:
  Hier wird das lineare Gleichungssystem in jedem Newton-Schritt nicht direkt (also exakt, etwa mittels $LR$-Zerlegung), sondern iterativ gelöst; man spricht von einer **inneren** Iteration im Rahmen der (**äußeren**) Newton-Iteration.

- **Quasi-Newton-Verfahren**:
  Hier wird eine Folge $B^{(i)}$ von Näherungen für $F'(\bar{x})$ erzeugt, und zwar nicht mittels teuerer Neuberechnung, sondern mittels billiger **Updates**. Man nutzt aus, dass ein **Rang-1-Update** $(B + uv^T)$ mit zwei beliebigen Vektoren $u, v \in \mathbb{R}^n$ (invertierbar genau dann, wenn $1 + v^T B^{-1} u \neq 0$) ggfs. leicht zu invertieren ist:

$$\left(B + uv^T\right)^{-1} = \left(I - \frac{(B^{-1}u)v^T}{1 + v^T B^{-1} u}\right) B^{-1}.$$

  Broyden gab eine passende Wahl für $u$ und $v$ an ($s$ wie oben im Algorithmus):

$$B^{(i+1)} := B^{(i)} + \frac{F(x^{(i+1)})s^T}{s^T s}.$$



# 6.4. Anwendungsbeispiele zu Iterationen

## Iterative Verfahren in der Informatik

- **Partielle Differentialgleichungen (PDE)**:
  Bei deren Diskretisierung mittels Finiter Differenzen oder Finiter Elemente entstehen große, schwach besetzte Gleichungssysteme, die ggfs. linearisiert und dann iterativ gelöst werden müssen. PDE treten bspw. in der numerischen Simulation oder in der Bildverarbeitung auf.

- **Computergrafik**:
  Fraktale (selbstähnliche) Kurven und Flächen werden über Iterationen erzeugt. Sie werden in der Computergrafik zur Modellierung natürlicher Objekte wie Pflanzen, Wolken oder Gebirgszüge verwendet.

- **Neuronale Netze**:
  Neuronale Netze dienen der Modellierung komplexer Systeme und sind der Funktionsweise des menschlichen Gehirns nachempfunden. Im einfachsten Fall besteht das Netz aus $n$ Knoten $N_i$, die jeweils ein Eingabesignal $x_i$ empfangen, mit einem Gewicht $w_i$ verstärken und an einen gemeinsamen Ausgabeknoten weiterleiten, der alle eingehenden Signale aufsummiert zu $y := \sum_{i=1}^{n} w_i x_i$. Ein neuronales Netz ist somit hauptsächlich durch seine Gewichte $w_i$ bestimmt, die so zu wählen sind, dass das Netz die gesuchte Lösung eines Problems finden und angeben kann. Diese Aufgabe übernimmt ein iterativer Lernalgorithmus, der anhand von Testbeispielen mit vorgegebener Lösung $y$ das Netz so optimiert, dass dieses möglichst die gesuchte Antwort liefert.

- ...