I Lineare Gleichungssysteme

1. Gauß Algorithmus

Gegeben: Ein *lineares Gleichungssystem* (GLS) $A\vec{x} = \vec{b}$ mit einer *regulären* $(n,n)-$ Koeffizientenmatrix A und der rechten Seite $\vec{b}$.
Da A regulär, ist das GLS *eindeutig* lösbar.

Die erweiterte Matrix

$$\left(\begin{array}{ccc|c} a_{11} & \dots & a_{1n} & b_1 \\ \vdots & \ddots & \vdots & \vdots \\ a_{n1} & \dots & a_{nn} & b_n \end{array}\right)$$

läßt sich mit Hilfe des *Gauß Algorithmus* auf folgende Form bringen:

$$\left(\begin{array}{ccc|c} r_{11} & \dots & r_{1n} & c_1 \\ & \ddots & \vdots & \vdots \\ 0 & & r_{nn} & c_n \end{array}\right)$$

Also erhalten wir: $R\vec{x} = \vec{c}$ mit einer oberen Dreiecksmatrix R und der neuen rechten Seite $\vec{c}$.
Bei diesem GLS $R\vec{x} = \vec{c}$ lassen sich durch *Rückwärtseinsetzen* (d.h.: man muß in der letzten Zeile beginnen) die Unbekannten $x_n, x_{n-1}, \dots, x_1$ berechnen.

Folgende Gauß-Schritte sind für die Umwandlung des GLS notwendig:

a) *In der ersten Spalte Nullen erzeugen*:

$$\left(\begin{array}{cccc|c} a_{11} & a_{12} & \dots & a_{1n} & b_1 \\ a_{21} & a_{22} & \dots & a_{2n} & b_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n1} & a_{n,n-1} & \dots & a_{nn} & b_n \end{array}\right)$$

Falls $a_{11} \neq 0$ (sonst vorher Zeilentausch), berechne:
$(i-$te Zeile$) - \dfrac{a_{i1}}{a_{11}} * (1.$ Zeile), also mit $l_{i1} = \dfrac{a_{i1}}{a_{11}}$

$$\tilde{a}_{ik} = a_{ik} - l_{i1} * a_{1k} \quad , \quad \tilde{a}_{i1} = 0$$
$$\tilde{b}_i = b_i - l_{i1} * b_1 \quad , \quad l_{i1} = \frac{a_{i1}}{a_{11}} \quad , \quad (i = 2, \dots, n \ ; \ k = 2, \dots, n)$$

Da wir die neuen Werte $\tilde{a}_{ik}$ und $\tilde{b}_i$ wieder auf den gleichen Speicherplätzen speichern, schreiben wir i.f. wieder a_{ik} , b_i anstelle von $\tilde{a}_{ik}$, $\tilde{b}_i$. Da die neuen Werte $a_{i1} = 0 \quad (i = 2, \ldots, n)$, und wir diese Nullen nicht mehr benötigen, speichern wir auf diesen Speicherplätzen die Vorfaktoren $l_{i1} = \dfrac{a_{i1}}{a_{11}}$. Damit erhalten wir nach dem 1. Schritt folgendes neue Schema:

$$\left(\begin{array}{c|ccc|c} \underline{a_{11}} & a_{12} & \ldots & a_{1n} & b_1 \\ \overline{l_{21}\,|} & a_{22} & \ldots & a_{2n} & b_2 \\ \vdots\,| & \vdots & \ddots & \vdots & \vdots \\ l_{n1}\,| & a_{n2} & \ldots & a_{nn} & b_n \end{array}\right)$$

b) *In der zweiten Spalte Nullen erzeugen*:

Falls $a_{22} \neq 0$ (sonst vorher Zeilentausch), berechne:
($i-$te Zeile) $-\dfrac{a_{i2}}{a_{22}} * (2.\text{ Zeile})$, also mit $\quad l_{i2} = \dfrac{a_{i2}}{a_{22}}$

$$\boxed{\begin{array}{l} \tilde{a}_{ik} = a_{ik} - l_{i2} * a_{2k} \quad , \quad \tilde{a}_{i2} = 0 \\ \tilde{b}_i = b_i - l_{i2} * b_2 \quad , \quad l_{i2} = \dfrac{a_{i2}}{a_{22}} \quad , \quad (i = 3, \ldots, n \ ; \ k = 3, \ldots, n) \end{array}}$$

Wir speichern wieder die Vorfaktoren $l_{i2} = \dfrac{a_{i2}}{a_{22}}$ auf den Speicherplätzen von a_{i2}.

Fahren wir so fort bis zur vorletzten Spalte, so erhalten wir nach dem letzten Schritt das folgende Schema:

$$\left(\begin{array}{c|ccccc} \underline{r_{11}} & r_{12} & \ldots & r_{1,n-1} & r_{1n} & c_1 \\ \overline{l_{21}\,|} & \underline{r_{22}} & \ldots & r_{2,n-1} & r_{2n} & c_2 \\ l_{31} & l_{32}\,| & \ddots & r_{3,n-1} & r_{3n} & c_3 \\ \vdots & & \ddots & \ddots & \vdots & \vdots \\ l_{n1} & l_{n2} & \ldots & \overline{l_{n,n-1}\,|} & r_{nn} & c_n \end{array}\right)$$

Mit der Linksdreiecksmatrix L und der Rechtsdreiecksmatrix R

$$L = \begin{pmatrix} 1 & & & 0 \\ l_{21} & 1 & & \\ \vdots & \ddots & \ddots & \\ l_{n1} & \ldots & l_{n,n-1} & 1 \end{pmatrix} \quad , \quad R = \begin{pmatrix} r_{11} & r_{12} & \ldots & r_{1n} \\ & r_{22} & \ldots & r_{2n} \\ & & \ddots & \vdots \\ 0 & & & r_{nn} \end{pmatrix}$$

gilt: $\boxed{A = LR}$

Denn: z.B.: (2. Zeile)$*$($k-$te Spalte) ergibt:

$l_{21}r_{1k} + r_{2k} = l_{21}a_{1k} + \tilde{a}_{2k} = a_{2k} \quad (k = 1, \ldots, n)$,
da $r_{1k} = a_{1k}$ (1. Zeile wird nicht verändert), und $r_{2k} = \tilde{a}_{2k} = a_{2k} - l_{21}a_{1k}$ (vgl. Schritt a)).

Also gilt

$$A\vec{x} = \vec{b} \quad \Leftrightarrow \quad LR\vec{x} = \vec{b} \quad \Leftrightarrow \quad \{\, L\vec{c} = \vec{b}\ ,\ R\vec{x} = \vec{c}\,\}$$

Zur Lösung des linearen GLS $A\vec{x} = \vec{b}$ sind also 3 Schritte durchzuführen:

1.	$A = LR$	$(LR-$ Zerlegung)
2.	$L\vec{c} = \vec{b}$	(Vorwärtseinsetzen)
3.	$R\vec{x} = \vec{c}$	(Rückwärtseinsetzen)

Pivotstrategien

Das Element, durch das bei der Berechnung der Vorfaktoren l_{ij} *dividiert* wird, wird *Pivot* genannt. Falls jeweils die auftretenden Diagonalelemente $\neq 0$ sind, so können diese als Pivot genommen werden. Diese *Pivotstrategie* nennt man *Diagonalstrategie*:

1. Diagonalstrategie

Ist das jeweilige Diagonalelement $\neq 0$, so benutze man dieses Diagonalelement als Pivotelement.

Die *Diagonalstrategie* ist nur sinnvoll bei *diagonaldominanten* Matrizen. Dabei heißt eine $(n,n)-$Matrix *diagonaldominant*, wenn gilt:

$$|a_{ii}| > \sum_{k=1,k\neq i}^{n} |a_{ik}| \quad \forall i = 1, \ldots, n$$

In allen anderen Fällen kann die *Diagonalstrategie* zu großen *Rundungsfehlern* führen, wie das folgende Beispiel zeigt:

Beispiel

$$\left(\begin{array}{cc|c} 0.00035 & 1 & 1.2224 \\ 1 & 1 & 2.333 \end{array}\right)$$

Die exakte Lösung lautet: $x_1 = 1.111$, $x_2 = 1.222$.

5−stellige Rechnug ergibt bei Diagonalstrategie

$$\tilde{a}_{22} = a_{22} - \frac{a_{21}}{a_{11}} * a_{12} = 1 - \frac{1}{0.00035} = -2856.1$$

$$\tilde{b}_2 = b_2 - \frac{a_{21}}{a_{11}} * b_1 = 2.333 - \frac{1.2224}{0.00035} = -3490.3 \quad \text{, also}$$

$$\left(\begin{array}{cc|c} 0.00035 & 1 & 1.2224 \\ 0 & -2856.1 & -3490.3 \end{array}\right)$$

$$\Rightarrow \quad x_2 = \frac{3490.3}{2856.1} = 1.2221 \quad , \quad x_1 = \frac{1.2224 - 1.2221}{0.00035} = \frac{0.0003}{0.00035} = 0.85714 \ .$$

Wir erhalten also in diesem Beispiel bei Benutzung der Diagonalstrategie sehr ungenaue Ergebnisse. Vertauschen wir vorher die 1. Zeile mit der 2. Zeile, so erhalten wir ein wesentlich besseres Resultat:

$$\left(\begin{array}{cc|c} 1 & 1 & 2.333 \\ 0.00035 & 1 & 1.2224 \end{array}\right) \quad \Rightarrow \quad \left(\begin{array}{cc|c} 1 & 1 & 2.333 \\ 0 & 0.99965 & 1.2216 \end{array}\right)$$

$$\Rightarrow \quad x_2 = \frac{1.2216}{0.99965} = 1.222 \quad , \quad x_1 = 2.333 - 1.222 = 1.111.$$

Also ist es sinnvoll, die Zeilen zu vertauschen, wenn das Pivotelement betraglich klein ist. Numerisch günstig ist es, wenn das Pivotelement betraglich möglichst groß ist, weil dann der Vorfaktor l_{ij} betraglich klein und damit der Rundungsfehler nicht zu groß wird. Diese Überlegung führt zur *Spalten-Maximum-Strategie*:

2. Spalten-Maximum-Strategie

Sollen in der $j-$ten Spalte Nullen erzeugt werden, so bestimme man das betraglich größte Element in dieser Spalte (ab Diagonalelement), d.h.: $|a_{kj}| = \max\limits_{j \le i \le n} |a_{ij}|$. Dann vertausche man die $j-$te Zeile mit der $k-$ten Zeile.

$$\begin{pmatrix} a_{11} & \dots & \dots & \dots & a_{1n} \\ & \ddots & & & \\ & & a_{jj} & \dots & a_{jn} \\ & & \vdots & \ddots & \vdots \\ 0 & & a_{kj} & \dots & a_{kn} \\ & & \vdots & \ddots & \vdots \\ & & a_{nj} & \dots & a_{nn} \end{pmatrix}$$

Sind die Matrix-Elemente in den einzelnen Zeilen sehr unterschiedlich groß, so sollte man vor der Suche des Spaltenmaximums eine *"Normierung"* vornehmen, d.h.:

$$\tilde{a}_{ij} = \frac{a_{ij}}{\sum\limits_{k=1}^{n} |a_{ik}|} \quad , \quad 1 \le i, j \le n.$$

Diese "Normierung" wird nicht explizit durchgeführt, sondern bei der *Pivotwahl* berücksichtigt. Also wird bei der Pivotwahl in der $j-$ten Spalte folgendes bestimmt:

$$\boxed{\max_{j \le i \le n} \frac{|a_{ij}|}{\sum\limits_{k=j}^{n} |a_{ik}|}}$$

Berechnung der Determinante

Beachtet man die Zeilenvertauschungen, so läßt sich aus der Matrix R einfach die Determinante von A berechnen. Es gilt: $\det A = (-1)^s \prod_{i=1}^{n} r_{ii}$, wobei s die Anzahl der Zeilenvertauschungen ist, denn es gilt:

$\pm \det A = \det(LR) = \det L \det R = \prod_{i=1}^{n} r_{ii}$ (da $\det L = 1$).

Daß die $LR-$Zerlegung und die Veränderung der rechten Seite (Vorwärtseinsetzen) in zwei getrennten Schritten durchgeführt wird, hat folgenden *Vorteil*:

Sollen mehrere GLS mit *gleicher Koeffizientenmatrix* A für *verschiedene* rechte Seiten $\vec{b}_1, \vec{b}_2, \ldots, \vec{b}_m$ gelöst werden, so muß nur *einmal* die $LR-$Zerlegung durchgeführt werden. Nur die Schritte "Vorwärtseinsetzen" und "Rückwärtseinsetzen" müssen mehrmals ausgeführt werden.

Berechnung der inversen Matrix A^{-1}

Es gilt: $AX = E \quad \Leftrightarrow \quad A\vec{s}_i = \vec{e}_i \quad , \quad (i = 1, \ldots, n)$

Hiebei ist E die Einheitsmatrix, X die gesuchte inverse Matrix A^{-1}, $\vec{s}_i$ der $i-$te Spaltenvektor von A^{-1} und $\vec{e}_i$ der $i-$te Einheitsvektor. Also müssen zur Berechnung der inversen Matrix A^{-1} n GLS mit gleicher Koeffizientenmatrix A, aber verschiedenen rechten Seiten $\vec{e}_1, \vec{e}_2, \ldots, \vec{e}_n$ gelöst werden.

Im folgenden Algorithmus werden die Zeilenvertauschungen auf einem Vektor $\vec{p}$ gespeichert. Dies benötigt man für das Vorwärtseinsetzen, da ja erst hier die rechte Seite umgeformt wird.

Die Werte l_{ij} werden auf a_{ij} gespeichert, der Vektor $\vec{c}$ und der Lösungsvektor $\vec{x}$ werden auf $\vec{b}$ gespeichert. Die Werte a_{ij} der Matrix A und die Werte b_i der rechten Seite werden also bei der Durchführung des Gauß-Algorithmus verändert. Benötigt man die Werte der Matrix oder der rechten Seite nach Beendigung des Algorithmus noch für ander Zwecke, so müssen diese Werte vorher gesondert gespeichert werden.

Gauß-Algorithmus

1. LR−Zerlegung $A = LR$

$det = 1$
für $k = 1$ bis $(n-1)$
    {Pivotsuche}
    $max = 0\ ;\ p_k = 0$
    für $i = k$ bis n
        $s = 0$
        für $j = k$ bis n : $s = s + |a_{ij}|$: Ende j−Schleife
        $q = |a_{ik}|/s$
        falls $q > max \Rightarrow max = q\ ;\ p_k = i$
    Ende i−Schleife
    falls $max = 0 \Rightarrow$ Matrix singulär, STOP
    falls $p_k \neq k \Rightarrow$ {Vorzeichenwechsel, Zeilentausch}
        $det = -det$
        für $j = 1$ bis n
            $h = a_{kj}\ ;\ a_{kj} = a_{p_k,j}\ ;\ a_{p_k,j} = h$
        Ende j−Schleife
    $det = det * a_{kk}$
    {Nullen in der k−ten Spalte erzeugen}
    für $i = (k+1)$ bis n
        $a_{ik} = a_{ik}/a_{kk}$
        für $j = (k+1)$ bis n
            $a_{ij} = a_{ij} - a_{ik} * a_{kj}$
        Ende j−Schleife
    Ende i−Schleife
Ende k−Schleife
$det = det * a_{nn}$
falls $det = 0 \Rightarrow$ Matrix singulär, STOP
{Ende LR−Zerlegung}

2. Vorwärtseinsetzen $L\vec{c} = \vec{b}$

für $k = 1$ bis $(n-1)$
    falls $p_k \neq k \Rightarrow h = b_k\ ;\ b_k = b_{p_k}\ ;\ b_{p_k} = h$
Ende k−Schleife
für $i = 2$ bis n
    für $j = 1$ bis $(i-1)$: $b_i = b_i - a_{ij} * b_j$: Ende j−Schleife
Ende i−Schleife
{Ende Vorwärtseinsetzen}

3. Rückwärtseinsetzen $\quad R\vec{x} = \vec{c}$

für $i = n$ bis 1 (rückwärts)
$\quad s = b_i$
$\quad$ für $k = (i+1)$ bis n : $\quad s = s - a_{ik} * b_k$: $\quad$ Ende $k-$Schleife
$\quad b_i = s/a_{ii}$
Ende $i-$Schleife
{Ende Rückwärtseinsetzen}

Rechenaufwand

Der Rechenaufwand beim Gauß-Algorithmus bei einem $(n,n)-$GLS beträgt bei der $LR-$Zerlegung $\boxed{\approx n^3}$, da 3 ineinandergeschachtelte Schleifen durchlaufen werden. Rechnet man genauer nach, so ergibt sich ein Rechenaufwand von $\frac{2}{3}n^3$.

Beim Vorwärts- und Rückwärtseinsetzen benötigt man nur noch einen Rechenaufwand von $\approx n^2$ (2 ineinandergeschachtelte Schleifen).

Der Rechenaufwand von $\approx n^3$ in der Zerlegungsphase bedeutet "lange" Rechenzeiten bei großen GLS. Z.B. steigt bei Verdoppelung der Unbekannten der Rechenaufwand um den Faktor $2^3 = 8$. Wir werden später iterative Verfahren behandeln, die zur Bewältigung sehr großer GLS besser geeignet sind als der Gauß-Algorithmus.

Fehlerrechnung

Trotz guter Pivotstrategie treten bei der Berechnung der Lösung eines linearen GLS *Rundungsfehler* auf. Wir wollen versuchen, die auftretenden Rundungsfehler abzuschätzen. Dazu benötigen wir ein "Maß" für eine Matrix A :
die "Norm von A" : $\|A\|$.

Matrixnormen

Ist $A\vec{x} = \vec{b}$, so gilt $\|A\vec{x}\| = \|\vec{b}\|$ mit irgendeiner Vektornorm $\|\dots\|$. Für die Matrixnorm $\|A\|$ soll gelten:
$\|A\vec{x}\| \le \|A\|\|\vec{x}\|$, also $\|A\| \ge \dfrac{\|A\vec{x}\|}{\|\vec{x}\|}$, falls $\vec{x} \neq \vec{0}$.

Also definieren wir

Definition 1.1 : *Matrixnorm*

$$\boxed{\|A\| = \max_{\vec{x}\neq\vec{0}} \frac{\|A\vec{x}\|}{\|\vec{x}\|} = \max_{\|\vec{x}\|=1} \|A\vec{x}\|}$$

Die so definierte Matrixnorm heißt die zur Vektornorm $\|\vec{x}\|$ *zugeordnete Matrixnorm.* Es gelten folgende Eigenschaften für die Matrixnorm:

Satz 1.2 : *Eigenschaften der Matrixnorm*

$\|A\| \geq 0$ für alle A , $\|A\| = 0 \quad \Leftrightarrow \quad A = 0$
$\|cA\| = |c|\|A\|$, $(c \in I\!R)$
$\|A + B\| \leq \|A\| + \|B\|$
$\|AB\| \leq \|A\|\|B\|$.

Da in der Anwendung unterschiedliche Vektornormen benutzt werden, erhalten wir auch unterschiedliche Matrixnormen. Wir wollen zwei Beispiele behandeln:

Beispiel 1.: $\infty-$Norm

$$\|\vec{x}\|_\infty = \max_{1\leq i\leq n} |x_i| \ ,$$

$$\|A\|_\infty = \max_{\|\vec{x}\|_\infty=1} \|A\vec{x}\|_\infty = \max_{\|\vec{x}\|_\infty=1} \max_{1\leq i\leq n} |\sum_{k=1}^{n} a_{ik}x_k|$$

$$= \max_{1\leq i\leq n} \max_{\|\vec{x}\|_\infty=1} |\sum_{k=1}^{n} a_{ik}x_k| \leq \max_{1\leq i\leq n} \sum_{k=1}^{n} |a_{ik}| \quad (\text{da } |x_k| \leq \|\vec{x}\|_\infty = 1 \quad \forall k = 1, \ldots, n).$$

Das Gleichheitszeichen wird angenommen für den Vektor $\vec{x}$ mit $x_k = sign(a_{ik}) \cdot 1$.
Also gilt für die $\infty-$Norm

$$\boxed{\|A\|_\infty = \max_{1\leq i\leq n} \sum_{k=1}^{n} |a_{ik}| \quad (\textit{Zeilensummennorm})}$$

Beispiel 2.: $2-$Norm

$$\|\vec{x}\|_2 = \sqrt{\sum_{i=1}^{n} x_i^2} = \sqrt{<\vec{x}, \vec{x}>} \quad (\text{Euklidnorm oder } 2-\text{Norm}).$$

Bevor wir $\|A\|_2$ untersuchen, benötigen wir einige Eigenschaften über Eigenwerte (EW), Eigenvektoren (EV) und quadratische Formen:

Hilfssatz 1.3

Sei A eine $(n, n)-$Matrix und $\vec{x}$, $\vec{y} \in I\!R^n$, dann gilt
$< A\vec{x}, \vec{y} >=< \vec{x}, A^T\vec{y} >$.
Ist insbesondere die Matrix A *symmetrisch*, also $A = A^T$, so gilt
$< A\vec{x}, \vec{y} >=< \vec{x}, A\vec{y} >=< A\vec{y}, \vec{x} >$.

Beweis

$$< A\vec{x}, \vec{y} >= \sum_{i=1}^{n}\Big(\sum_{j=1}^{n} a_{ij}x_j\Big)y_i = \sum_{j=1}^{n}\Big(\sum_{i=1}^{n} a_{ij}y_i\Big)x_j = \sum_{j=1}^{n}\Big(\sum_{i=1}^{n} (A^T)_{ji}y_i\Big)x_j$$
$$=< \vec{x}, A^T\vec{y} > .$$

Satz 1.4

Sei A eine $(n,n)-$Matrix mit EW λ und zugehörigem EV $\vec{x}$, also $A\vec{x} = \lambda\vec{x}$, $\vec{x} \neq \vec{0}$. Dann gilt

a) λ EW von $A \Leftrightarrow \det(A - \lambda E) = 0$.

b) A^m hat den EW λ^m mit gleichem EV $\vec{x}$.

c) $(A - \alpha E)$ hat den EW $(\lambda - \alpha)$ mit gleichem EV $\vec{x}$.

d) A ist regulär $\Leftrightarrow \det A \neq 0 \Leftrightarrow$ alle EW von A sind $\neq 0$.

e) A^{-1} hat den EW $\dfrac{1}{\lambda}$ mit gleichem EV $\vec{x}$.

f) Ist A *ähnlich* zu B , d.h.: $\exists\ C$ (regulär) mit $C^{-1}AC = B$, so gilt:
Die EW von A und B sind gleich. Für die EV gilt:
Ist $\vec{x}$ ein EV von A zum EW $\lambda \Rightarrow (C^{-1}\vec{x})$ ist EV von B zum gleichen EW λ.

g) Ist A symmetrisch, so existiert eine orthogonale Matrix C mit $C^T AC = D$ (D Diagonalmatrix), also A ist orthogonal ähnlich zur Diagonalmatrix D.
Die EW von A sind alle reell. Die EV zu verschiedenen EW stehen senkrecht aufeinander.
In der Diagonalen von D stehen die EW von A. Die Spaltenvektoren von C sind normierte EV zu den EW von A.

h) Ist A symmetrisch, so definiert $Q(\vec{x}) =< A\vec{x}, \vec{x} >$ eine *quadratische Form*.
Gilt $Q(\vec{x}) > 0 \quad \forall \vec{x} \neq \vec{0}$, so heißt die Matrix A *positiv definit*. In diesem Fall sind alle EW von A positiv, also $\lambda > 0 \quad \forall\lambda$ (EW von A).

Die meisten dieser Eigenschaften sind aus der Vorlesung "Mathematik für Elektrotechniker I" bekannt. Die Eigenschaften b) – f) möchte ich hier nochmal beweisen:

zu b) $A\vec{x} = \lambda\vec{x} \Rightarrow A^2\vec{x} = \lambda A\vec{x} = \lambda^2\vec{x}$
Mit Induktion folgt sofort: $A^m\vec{x} = \lambda^m\vec{x}$.

zu c) $(A - \alpha E)\vec{x} = A\vec{x} - \alpha\vec{x} = \lambda\vec{x} - \alpha\vec{x} = (\lambda - \alpha)\vec{x}$.

zu d) $\lambda = 0$ EW $\Leftrightarrow \det(A - 0E) = \det A = 0 \Leftrightarrow A$ nicht regulär.

zu e) $A\vec{x} = \lambda\vec{x} \Rightarrow A^{-1}A\vec{x} = A^{-1}(\lambda\vec{x}) \Rightarrow \vec{x} = \lambda A^{-1}\vec{x} \Rightarrow A^{-1}\vec{x} = \dfrac{1}{\lambda}\vec{x}$.

zu f) $A\vec{x} = \lambda\vec{x} \Rightarrow C^{-1}A\vec{x} = \lambda C^{-1}\vec{x} \Rightarrow$ (mit $\vec{x} = C\vec{u}$, also $\vec{u} = C^{-1}\vec{x}$)
$C^{-1}AC\vec{u} = \lambda\vec{u}$.

Nun können wir die zur Vektornorm $\|\vec{x}\|_2$ zugeordnete Matrixnorm $\|A\|_2$ untersuchen:

$$\|A\|_2 = \max_{\|\vec{x}\|_2=1} \|A\vec{x}\|_2 = \max_{\|\vec{x}\|_2=1} \sqrt{< A\vec{x}, A\vec{x} >} = \max_{\|\vec{x}\|_2=1} \sqrt{< A^TA\vec{x}, \vec{x} >}$$

$$\Rightarrow \|A\|_2 = \max_{\|\vec{x}\|_2=1} \sqrt{Q(\vec{x})} \quad \text{mit der quadratischen Form } Q(\vec{x}) =< A^TA\vec{x}, \vec{x} >.$$

Die Matrix (A^TA) ist symmetrisch und positiv semidefinit, denn:
$(A^TA)^T = A^TA^{TT} = A^TA$, $< A^TA\vec{x}, \vec{x} >=< A\vec{x}, A\vec{x} >= \|A\vec{x}\|_2^2 \geq 0$.

Ist A regulär, so ist die Matrix A^TA sogar positiv definit, denn
$\|A\vec{x}\|_2 = 0 \quad \Leftrightarrow \quad A\vec{x} = \vec{0} \quad \Leftrightarrow \quad \vec{x} = \vec{0}$ (da A regulär).

Damit ist also $Q(\vec{x}) =< A^TA\vec{x}, \vec{x} >$ eine positiv semidefinte quadratische Form, die wir uns nun genauer ansehen wollen.
Da (A^TA) symmetrisch und positiv semidefinit, gilt:
Alle EW von (A^TA) sind reell und ≥ 0. Die EV zu verschiedenen EW stehen senkrecht aufeinander.
Seien $\mu_1 \geq \mu_2 \geq \ldots \geq \mu_n \geq 0$ die EW von A^TA, und seien

$$\vec{x}_1 \ , \ \vec{x}_2 \ , \ \ldots \ , \ \vec{x}_n \ \text{ zugehörige EV mit } < \vec{x}_i, \vec{x}_j >= \begin{cases} 0 & ,\text{falls } i \neq j \\ 1 & ,\text{falls } i = j \end{cases} ,$$

so bilden die EV $\vec{x}_i$ eine Basis des $\mathbb{R}^n$, und jeder Vektor $\vec{x} \in \mathbb{R}^n$ mit $\|\vec{x}\|_2 = 1$ läßt sich als Linearkombination der EV $\vec{x}_i$ schreiben:

$$\vec{x} = \sum_{i=1}^{n} c_i\vec{x}_i \quad \text{mit}$$

$$\|\vec{x}\|_2^2 =< \vec{x}, \vec{x} >= \Big\langle \sum_{i=1}^{n} c_i\vec{x}_i \ , \ \sum_{j=1}^{n} c_j\vec{x}_j \Big\rangle = \sum_{i=1}^{n}\sum_{j=1}^{n} c_ic_j < \vec{x}_i, \vec{x}_j >= \sum_{i=1}^{n} c_i^2 = 1 \ ,$$

$$\text{da } \ < \vec{x}_i, \vec{x}_j >= \begin{cases} 0 & ,\text{falls } i \neq j \\ 1 & ,\text{falls } i = j \end{cases} .$$

Also gilt für $Q(\vec{x})$:

$$Q(\vec{x}) =< A^TA\vec{x}, \vec{x} >= \Big\langle \sum_{i=1}^{n} c_iA^TA\vec{x}_i \ , \ \sum_{j=1}^{n} c_j\vec{x}_j \Big\rangle = \sum_{i=1}^{n}\sum_{j=1}^{n} c_ic_j\mu_i < \vec{x}_i, \vec{x}_j >$$

$$= \sum_{i=1}^{n} c_i^2\mu_i \leq \mu_1 \sum_{i=1}^{n} c_i^2 = \mu_1 \quad , \text{ da } A^TA\vec{x}_i = \mu_i\vec{x}_i \ .$$

Das Gleichheitszeichen wird angenommen für $\vec{x} = \vec{x}_1$ (EV zu μ_1).
Also gilt

$$\boxed{\|A\|_2 = \max_{\|\vec{x}\|_2=1} \sqrt{Q(\vec{x})} = \sqrt{\mu_1} \ , \ (\mu_1 \text{ größter EW von } (A^TA))}$$

Wir wollen nun $\|A^{-1}\|_2$ untersuchen, falls A regulär ist:

$\|A^{-1}\|_2 = \sqrt{\varrho_1}$ mit ϱ_1 ist der größte EW von
$(A^{-1})^TA^{-1} = (A^T)^{-1}A^{-1} = (AA^T)^{-1}$.
AA^T ist *ähnlich* zu A^TA , denn: $A^{-1}(AA^T)A = (A^{-1}A)A^TA = A^TA$.
Also hat AA^T die gleichen EW wie A^TA , also auch $\mu_1 \geq \mu_2 \geq \ldots \geq \mu_n$.
Die Matrix $(AA^T)^{-1}$ hat dann die reziproken EW $\dfrac{1}{\mu_n} \geq \dfrac{1}{\mu_{n-1}} \geq \ldots \geq \dfrac{1}{\mu_1}$.
Der größte dieser EW ist $\dfrac{1}{\mu_n}$, wobei μ_n der kleinste EW von A^TA ist. Also gilt

$$\boxed{\|A^{-1}\|_2 = \frac{1}{\sqrt{\mu_n}} \ , \ (\mu_n \text{ kleinster EW von } (A^TA))}$$

Spezialfall: *A symmetrisch*

Ist A symmetrisch, so gilt: $A^TA = A^2$. Ist λ ein EW von A, so ist λ^2 EW von $A^TA = A^2$. Also gilt für symmetrische Matrizen

$$\|A\|_2 = |\lambda_1| \quad , \quad (\lambda_1 \text{ betraglich größter EW von } A)$$
$$\|A^{-1}\|_2 = \frac{1}{|\lambda_n|} \quad , \quad (\lambda_n \text{ betraglich kleinster EW von } A)$$

Bemerkung

Um $\|A^{-1}\|_\infty$ (Zeilensummennorm) berechnen zu können, benötigt man die inverse Matrix A^{-1}. Bei der Berechnung von $\|A^{-1}\|_2$ benötigt man nur den kleinsten EW von A^TA. Ein Verfahren zur Berechnung des kleinsten EW werden wir später behandeln (vgl. Wielandt-Verfahren).

Fehlerrechnung

Gegeben sei das GLS $A\vec{x} = \vec{b}$ mit der regulären Koeffizientenmatrix A.

Sei $\vec{x}$ die exakte Lösung dieses GLS.

$\vec{x}_0$ sei die berechnete Lösung , $\vec{\Delta x} = \vec{x}_0 - \vec{x}$ der Fehler und $\vec{r} = A\vec{x}_0 - \vec{b}$.
Dann gilt

$A(\vec{\Delta x}) = A\vec{x}_0 - A\vec{x} = \vec{r} + \vec{b} - \vec{b} = \vec{r}$

$\Rightarrow \quad A(\vec{\Delta x}) = \vec{r} \quad \Rightarrow \quad \vec{\Delta x} = A^{-1}\vec{r}$. Also gilt bzg. einer beliebigen Norm

$$\frac{\|\vec{\Delta x}\|}{\|\vec{x}\|} = \frac{\|A^{-1}\vec{r}\|}{\|\vec{x}\|} \leq \frac{\|A^{-1}\|\|\vec{r}\|}{\frac{\|\vec{b}\|}{\|A\|}} = \|A\|\|A^{-1}\|\frac{\|\vec{r}\|}{\|\vec{b}\|} \ ,$$

denn: $\|\vec{b}\| = \|A\vec{x}\| \leq \|A\|\|\vec{x}\| \quad \Rightarrow \quad \|\vec{x}\| \geq \frac{\|\vec{b}\|}{\|A\|}$.

Definition 1.5 : *Konditionszahl einer Matrix*

Bzgl. einer beliebigen Matrixnorm definieren wir als *Konditionszahl* einer Matrix A

$$k(A) = \|A\|\|A^{-1}\|$$

Mit dieser Konditionszahl lautet die obige Fehlerabschätzung für den relativen Fehler

$$\frac{\|\vec{\Delta x}\|}{\|\vec{x}\|} \leq k(A)\frac{\|\vec{r}\|}{\|\vec{b}\|}$$

Der relative Fehler der Lösung eines linearen GLS hängt also wesentlich von der Konditionszahl der Koeffizientenmatrix A ab.

Da $\|AA^{-1}\| = \|E\| = 1 \leq \|A\|\|A^{-1}\| = k(A)$, gilt

$$\boxed{k(A) \geq 1}$$

Die Konditionszahl läßt sich für die $\|\ldots\|_\infty-$Norm nur sehr aufwendig berechnen, da man für $\|A^{-1}\|_\infty$ die komplette inverse Matrix A^{-1} benötigt. Einfacher ist die Berechnung der Konditionszahl für die $\|\ldots\|_2-$Norm, denn es gilt

$$\boxed{k(A) = \sqrt{\frac{\mu_1}{\mu_n}} \quad , \quad (\mu_1 \text{ größter - }, \ \mu_n \text{ kleinster EW von } \ A^TA)}$$

Spezialfall: *A symmetrisch*

$$\boxed{k(A) = \frac{|\lambda_1|}{|\lambda_n|} \quad , \quad (\lambda_1 \text{ betraglich größter - }, \ \lambda_n \text{ betraglich kleinster EW von } \ A)}$$

Störung der Eingangsdaten

Wir führen nun eine genauere Fehleranalyse für den Fall durch, daß die Eingabedaten im Computer nicht exakt dargestellt werden.

$(A + \Delta A)(\vec{x} + \vec{\Delta x}) = (\vec{b} + \vec{\Delta b}) \quad , \quad (\vec{\Delta x}$ Fehler der Lösung)
$\Rightarrow \quad (A + \Delta A)(\vec{\Delta x}) = \vec{b} + \vec{\Delta b} - A\vec{x} - \Delta A\vec{x} = \vec{\Delta b} - \Delta A\vec{x} \quad (\text{da } A\vec{x} = \vec{b})$
$\Rightarrow \quad \vec{\Delta x} = (A + \Delta A)^{-1}(\vec{\Delta b} - \Delta A\vec{x}) = [A(E + A^{-1}\Delta A)]^{-1}(\vec{\Delta b} - \Delta A\vec{x})$
$\Rightarrow \quad \|\vec{\Delta x}\| \leq \|A^{-1}\|\|(E + A^{-1}\Delta A)^{-1}\|\Big(\|\vec{\Delta b}\| + \|\Delta A\|\|\vec{x}\|\Big) \ .$

Nach einem Hilfsatz, den ich hier ohne Beweis zitieren möchte, gilt

$$\|(E + A^{-1}\Delta A)^{-1}\| \leq \frac{1}{1 - \|A^{-1}\|\|\Delta A\|} \quad , \quad \text{falls } \ \|A^{-1}\|\|\Delta A\| < 1 \ .$$

Also gilt unter der Voraussetzung, daß $\|A^{-1}\|\|\Delta A\| < 1$ ist:

$$\begin{aligned}\frac{\|\vec{\Delta x}\|}{\|\vec{x}\|} &\leq \frac{\|A^{-1}\|}{1 - \|A^{-1}\|\|\Delta A\|}\left(\frac{\|\vec{\Delta b}\|}{\|\vec{x}\|} + \|\Delta A\|\right) \\ &\leq \frac{\|A^{-1}\|\|A\|}{1 - \|A^{-1}\|\|\Delta A\|}\left(\frac{\|\vec{\Delta b}\|}{\|A\|\|\vec{x}\|} + \frac{\|\Delta A\|}{\|A\|}\right) \quad \Rightarrow \ (\text{da } \|A\|\|\vec{x}\| \geq \|\vec{b}\|)\end{aligned}$$

$$\boxed{\frac{\|\vec{\Delta x}\|}{\|\vec{x}\|} \leq \frac{k(A)}{1 - k(A)\frac{\|\Delta A\|}{\|A\|}}\left(\frac{\|\vec{\Delta b}\|}{\|\vec{b}\|} + \frac{\|\Delta A\|}{\|A\|}\right)}$$

Ist die Rechengenauigkeit (Genauigkeit der Darstellung) $\approx 10^{-d}$, also
$\frac{\|\vec{\Delta b}\|}{\|\vec{b}\|} \approx 5 \cdot 10^{-d} \quad , \quad \frac{\|\Delta A\|}{\|A\|} \approx 5 \cdot 10^{-d} \quad , \quad k(A) \approx 10^{\alpha} \quad \text{mit } \ 5 \cdot 10^{\alpha - d} \ll 1$, so gilt

$$\frac{\|\vec{\Delta x}\|}{\|\vec{x}\|} \leq 10^{\alpha-d+1} \quad , \quad k(A) = 10^{\alpha} \text{ Konditionszahl} \quad , \quad 10^{-d} \text{ Rechengenauigkeit}$$

Ergebnis

Bei großer Konditionszahl (also großem α) erhält man einen großen relativen Fehler. Dieser Fehler kann kleiner gehalten werden, wenn man die Rechengenauigkeit (also d) erhöht (z.B.: Extended REAL-Variable benutzen). Es sollte immer $d \gg \alpha$ sein.

Auch wenn die Eingabe exakt erfolgt, so treten doch bei jedem Gauß-Schritt Rundungsfehler im Bereich der Rechengenauigkeit auf. Also kann die gleiche Fehlerabschätzung benutzt werden.

Beispiel 1.

$$A = \begin{pmatrix} 0.990005 & 0.979996 \\ 0.979996 & 0.970004 \end{pmatrix} \quad , \quad \vec{b} = \begin{pmatrix} 1.9584083 \\ 1.9385935 \end{pmatrix} \quad \Rightarrow \quad \vec{x} = \begin{pmatrix} 1.8 \\ 0.18 \end{pmatrix}$$

(exakte Lösung).

$5-$*stellige Rechnung*, also $d = 5$, ergibt

$$A + \Delta A = \begin{pmatrix} 0.99 & 0.98 \\ 0.98 & 0.97 \end{pmatrix} \quad , \quad \vec{b} + \vec{\Delta b} = \begin{pmatrix} 1.9584 \\ 1.9386 \end{pmatrix}$$

$$\Rightarrow \quad \left(\begin{array}{cc|c} 0.99 & 0.98 & 1.9584 \\ 0 & -0.0001 & 0 \end{array}\right) \quad \Rightarrow \quad \vec{x} + \vec{\Delta x} = \begin{pmatrix} 1.9782 \\ 0 \end{pmatrix} .$$

Da A symmetrisch $\Rightarrow \quad k(A) = \dfrac{|\lambda_1|}{|\lambda_2|}$.

Die EW einer $(2,2)-$Matrix lassen sich einfach berechnen. Es gilt für

$A = \begin{pmatrix} a & c \\ c & b \end{pmatrix} \quad \Rightarrow \quad \lambda_{1,2} = \dfrac{a+b}{2} \pm \sqrt{c^2 - ab + \left(\dfrac{a+b}{2}\right)^2}$ sind die EW.

In unserem Beispiel erhalten wir

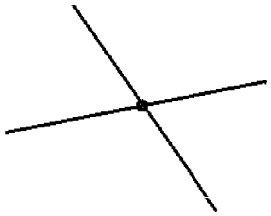
$\lambda_1 = 1.9600515$, $\lambda_2 = -0.0000425 \quad \Rightarrow \quad k(A) = 46092 \approx 4 \cdot 10^4 \quad \Rightarrow \quad \alpha \approx 4$

$\Rightarrow \quad \dfrac{\|\vec{\Delta x}\|}{\|\vec{x}\|} \leq 10^{4-5+1} = 1$,

also ist keine Stelle nach dem Komma garantiert.

Geometrische Deutung: $(2,2)-$GLS

Bei zwei linearen Gleichungen mit zwei Unbekannten ergibt die Lösung den Schnittpunkt zweier Geraden:



$k(A)$ klein , A gut konditioniert *$k(A)$ groß , A schlecht konditioniert*

Beispiel 2. *Hilbert-Matrix*

Ein Beispiel für schlecht konditionierte Matrizen liefern die *Hilbert-Matrizen* :

Definition 1.6 : *Hilbert-Matrix*

Die $(n.n)-$Matrix A mit $a_{ij} = \dfrac{1}{i+j-1}$, $(1 \le i,j \le n)$, heißt *Hilbert-Matrix* (der Größe n).

Die Inverse der Hilbert-Matrix ist $A^{-1} = B$ mit

$b_{ij} = \dfrac{(-1)^{i+j}}{i+j-1} c_i c_j$ mit $c_i = \dfrac{(n+i-1)!}{(i-1)!^2 (n-i)!}$.

Beispiel: $n = 4$

$$A = \underbrace{\begin{pmatrix} 1 & 1/2 & 1/3 & 1/4 \\ & 1/3 & 1/4 & 1/5 \\ & & 1/5 & 1/6 \\ & & & 1/7 \end{pmatrix}}_{\text{(symmetrisch)}} \quad , \quad A^{-1} = \underbrace{\begin{pmatrix} 16 & -120 & 240 & -140 \\ & 1200 & -2700 & 1680 \\ & & 6480 & -4200 \\ & & & 2800 \end{pmatrix}}_{\text{(symmetrisch)}} .$$

Als betraglich größten bzw. betraglich kleinsten EW von A erhalten wir

$\lambda_1 = 1.50021$, $\lambda_4 = 9.67 \cdot 10^{-5}$ $\Rightarrow$ $k(A) = \dfrac{|\lambda_1|}{|\lambda_4|} = 1.6 \cdot 10^4$ $\Rightarrow$ $\alpha = 4.$

Für den relativen Fehler folgt hieraus die Fehlerabschätzung

$\dfrac{\|\vec{\Delta x}\|}{\|\vec{x}\|} \le 10^{5-d}$. d sollte also viel größer als 5 sein.

Beispiel: $n = 8$

Für die Hilbertmatrix der Größe $n = 8$ erhalten wir

$\lambda_1 = 1.696$, $\lambda_8 = 1.11 \cdot 10^{-10}$ $\Rightarrow$ $k(A) = \dfrac{|\lambda_1|}{|\lambda_8|} = 1.5 \cdot 10^{10}$ $\Rightarrow$ $\alpha = 10$.

Für den relativen Fehler folgt hieraus die Fehlerabschätzung

$\dfrac{\|\vec{\Delta x}\|}{\|\vec{x}\|} \le 10^{11-d}$. d sollte also viel größer als 11 sein.

Für die rechte Seite $\vec{b} = (1, 1, \ldots, 1)^T$ erhält man als Lösung des GLS $A\vec{x} = \vec{b}$ in TURBO-PASCAL mit REAL-Variablen (d.h.: d=12) folgendes Ergebnis

exakte L.	berechnete Lösung
-8	-8.01
504	504.5
-7560	-7566.9
46200	46236.8

exakte L.	berechnete Lösung
-138600	-138698.2
216216	216354.0
-168168	-168265.7
51480	51507.4

Abschätzung der Konditionszahl

Da die Berechnung der Konditionszahl relativ aufwendig ist (vgl. Verfahren zur Berechnung von EW, Kap. III), begnügt man sich in der Praxis mit Abschätzungen.

Möglichkeit 1.

Sei $\vec{x}_0$ die berechnete Lösung, $\vec{x}$ die exakte Lösung und $\vec{\Delta x} = \vec{x}_0 - \vec{x}$ der Fehler.
Mit $\vec{r} = A\vec{x}_0 - \vec{b}$ gilt dann
$A(\vec{\Delta x}) = A\vec{x}_0 - A\vec{x} = \vec{r} + \vec{b} - \vec{b} = \vec{r} \quad \Rightarrow \quad \vec{\Delta x} = A^{-1}\vec{r}$

$$\Rightarrow \quad \|\vec{\Delta x}\| \leq \|A^{-1}\|\|\vec{r}\| \quad \Rightarrow \quad \|A^{-1}\| \geq \frac{\|\vec{\Delta x}\|}{\|\vec{r}\|} \quad \Rightarrow$$

$$\boxed{\tilde{k}(A) = \frac{\|A\|\|\vec{\Delta x}\|}{\|\vec{r}\|} \leq k(A)}$$

Um $\tilde{k}(A)$ zu berechnen, bestimmt man zunächst den Vektor $\vec{r} = A\vec{x}_0 - \vec{b}$, löst dann das GLS $A(\vec{\Delta x}) = \vec{r}$ (Die $LR-$Zerlegung ist durch die Berechnung von $\vec{x}_0$ bekannt. Also muß nur noch das Vorwärts- und Rückwärtseinsetzen durchgeführt werden).
Als Norm benutzt man hier am einfachsten die $\|\ldots\|_\infty-$Norm.

Möglichkeit 2.

$$\boxed{\tilde{k}(A) = \frac{\alpha_1 \cdot \ldots \cdot \alpha_n}{|\det A|} \quad \text{mit} \quad \alpha_i = \sqrt{\sum_{k=1}^{n} a_{ik}^2}}$$

Dieser Wert $\tilde{k}(A)$ ist ebenfalls ein "Maß" für die Kondition der Matrix A, $\tilde{k}(A)$ läßt sich aber nicht durch $k(A)$ abschätzen.

Daß $\tilde{k}(A)$ ein "Maß" für die Kondition von A ist, sieht man folgendermaßen:

Man bildet die Matrix B mit $b_{ij} = \dfrac{a_{ij}}{\alpha_i}$. Dadurch erhält man eine "normierte" Matrix.

Ist $|\det B| = \dfrac{1}{\tilde{k}(A)}$ sehr klein, so ist B "fast" singulär und damit die Matrix A schlecht konditioniert.

Beispiel hierzu

Hilbert-Matrix: $n = 4$:
$\det A = 1.6 \cdot 10^{-7}$, $\tilde{k}(A) \approx 10^6$, $k(A) \approx 10^4$.

Hilbert-Matrix: $n = 8$:
$\det A = 2.7 \cdot 10^{-33}$, $\tilde{k}(A) \approx 10^{29}$, $k(A) \approx 10^{10}$.

Dieser Wert $\tilde{k}(A)$ läßt sich sehr einfach berechnen, da $\det A$ beim Gauß-Algorithmus mitberechnet wird.

2. Cholesky-Verfahren für *symmetrische , positiv definite* Matrizen

Ist A *symmetrisch* und *positiv definit*, so läßt sich A zerlegen in

$$\boxed{A = LDL^T}$$

mit der unteren Dreiecksmatrix L und der Diagonalmatrix D

$$L = \begin{pmatrix} 1 & & & 0 \\ l_{21} & 1 & & \\ \vdots & & \ddots & \\ l_{n1} & \dots & l_{n,n-1} & 1 \end{pmatrix} \quad , \quad D = \begin{pmatrix} d_{11} & & & 0 \\ & d_{22} & & \\ & & \ddots & \\ 0 & & & d_{nn} \end{pmatrix} .$$

Für $A\vec{x} = \vec{b}$ gilt dann: $LDL^T\vec{x} = \vec{b}$.
Mit $\vec{d} = L^T\vec{x}$, $\vec{c} = D\vec{d}$ erhalten wir dann

1.	$A = LDL^T$	(LDL^T – Zerlegung)
2.	$L\vec{c} = \vec{b}$	(Vorwärtseinsetzen)
3.	$D\vec{d} = \vec{c}$	(Division durch d_{ii})
4.	$L^T\vec{x} = \vec{d}$	(Rückwärtseinsetzen)

Daß die Zerlegung $A = LDL^T$ bei symmetrischen, positiv definiten Matrizen möglich ist, zeigt der folgende Cholesky-Algorithmus :

Cholesky-Algorithmus

für $i = 1$ bis n
  für $k = 1$ bis i

$$\tilde{l}_{ik} = a_{ik} - \sum_{j=1}^{k-1} \tilde{l}_{ij} l_{kj}$$

  falls $k < i \quad \Rightarrow \quad l_{ik} = \tilde{l}_{ik}/d_{kk}$
  falls $k = i \quad \Rightarrow \quad d_{ii} = \tilde{l}_{ii}$
  falls $d_{ii} \leq 0 \quad \Rightarrow$ STOP , Matrix nicht positiv definit
  Ende $k-$Schleife
Ende $i-$Schleife

Denn:

$$a_{ik} = \tilde{l}_{ik} + \sum_{j=1}^{k-1} \tilde{l}_{ij} l_{kj} = \sum_{j=1}^{k} \tilde{l}_{ij} l_{kj} \quad (\text{da } l_{kk} = 1)$$

$$= \sum_{j=1}^{k} l_{ij} d_{jj} l_{kj} \quad (\text{da } \tilde{l}_{ij} = l_{ij} d_{jj})$$

$$= \sum_{j=1}^{k} (L)_{ij} (D)_{jj} (L^T)_{jk} = (LDL^T)_{ik}$$

Bemerkung 1.7 :

Gilt $d_{ii} > 0 \quad \forall i = 1, \ldots, n \quad \Rightarrow \quad A$ ist *positiv definit*.

Denn:

$$< A\vec{x}, \vec{x} > = < LDL^T\vec{x}, \vec{x} > = < DL^T\vec{x}, L^T\vec{x} > = < D\vec{u}, \vec{u} > = \sum_{i=1}^{n} d_{ii} u_i^2 > 0$$

$\forall \vec{u} = L^T\vec{x} \neq \vec{0}$, also auch $\forall \vec{x} \neq \vec{0}$, falls $d_{ii} > 0 \quad \forall i = 1, \ldots, n$.

Bemerkung 1.8 :

Man kann also mit Hilfe der LDL^T-Zerlegung feststellen, ob eine symmetrische Matrix A *positiv definit* ist.

Bemerkung 1.9 :

Damit man nicht zusätzlichen Speicherplatz für die Elemente $\tilde{l}_{ij}$ benötigt, wird der Algorithmus noch so verbessert, daß man mit nur einem zusätzlichen Speicherplatz *diag* auskommt. Von der Matrix A benötigt man damit nur die untere Dreiecksmatrix, also a_{ij} , $j \leq i$. Die Elemente d_{ii} der Diagonalmatrix D werden auf der Diagonalen von A, die Elemente l_{ij} , $j < i$, der unteren Dreiecksmatrix L unterhalb der Diagonalen von A gespeichert.

Vorteil *des Cholesky-Verfahrens gegenüber dem Gauß-Algorithmus*

Der Vorteil des Cholesky-Verfahrens liegt darin, daß man für die Matrix A nur $\frac{n(n+1)}{2}$ Speicherplätze (anstelle von n^2) benötigt, da die Symmetrie von A ausgenutzt wird. Der Rechenaufwand veringert sich ein wenig ($\frac{1}{3}n^3$ anstelle von $\frac{2}{3}n^3$ in der Zerlegungsphase), ist aber immer noch proportional n^3.

Bei symmetrischen, positiv definiten Matrizen ist eine spezielle Pivotstrategie nicht notwendig und auch nicht sinnvoll.

2. Cholesky-Verfahren für *symmetrische , positiv definite* Matrizen

1. LDL^T-Zerlegung

für $i = 1$ bis n
    für $k = 1$ bis $(i-1)$
        $s = 0$
        für $j = 1$ bis $(k-1)$
            $s = s + a_{ij} * a_{kj}$
        Ende $j-$Schleife
        $a_{ik} = a_{ik} - s$
    Ende $k-$Schleife
    $diag = a_{ii}$
    für $k = 1$ bis $(i-1)$
        $h = a_{ik}/a_{kk}$
        $diag = diag - a_{ik} * h$
        $a_{ik} = h$
    Ende $k-$Schleife
    falls $diag < 10^{-10} * |a_{ii}|$ $\Rightarrow$ STOP , Matrix nicht positiv definit
    $a_{ii} = diag$
Ende $i-$Schleife

2. Vorwärtseinsetzen und Division durch d_{ii}

für $i = 1$ bis n
    $s = 0$
    für $j = 1$ bis $(i-1)$
        $s = s + a_{ij} * b_j$
    Ende $j-$Schleife
    $b_i = b_i - s$
Ende $i-$Schleife
für $i = 1$ bis n
    $b_i = b_i/a_{ii}$
Ende $i-$Schleife

3. Rückwärtseinsetzen

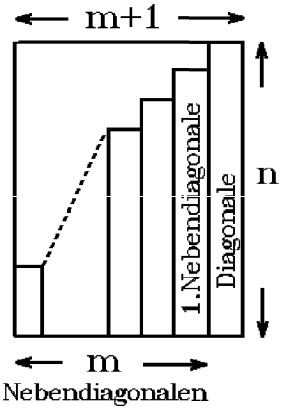
für $i = n$ bis 1 (rückwärts)
    $s = 0$
    für $j = (i+1)$ bis n
        $s = s + a_{ji} * b_j$
    Ende $j-$Schleife
    $b_i = b_i - s$
Ende $i-$Schleife

Bandmatrizen

In der Anwendung treten oft symmetrische, positiv definite *Bandmatrizen* auf. Eine Bandmartix A hat die *Bandbreite* m, wenn nur m Nebendiagonalen besetzt sind (alle anderen Nebendiagonalen enthalten nur Nullen).
Eine solche Bandmatrix der Bandbreite m wird zweckmäßigerweise anders gespeichert:

$$\begin{pmatrix} * & * & \dots & * & & 0 \\ * & \ddots & & & \ddots & \\ \vdots & & \ddots & & & * \\ * & & & \ddots & & \vdots \\ & \ddots & & & \ddots & * \\ 0 & & * & \dots & * & * \end{pmatrix}$$

m Nebendiagonalen



Bandmatrix der Bandbreite m *Speicherung auf einem $(n, m+1)-$Array*

Diese Speicherung hat den Vorteil, daß der Zeilenindex i unverändert bleibt, nur der Spaltenindex ändert sich. Es gilt

$$\underbrace{a_{i,k}}_{(n,n)-\text{Matrix}} \hat{=} \underbrace{a_{i,k-i+m+1}}_{(n,m+1)-\text{Matrix}}$$

Nutzt man die Bandstruktur beim Cholesky-Verfahren aus, so benötigt man nur noch $\approx nm^2$ Rechenoperationen (anstelle von n^3) in der Zerlegungsphase.
Im Programm muß eine andere Index-Rechnung durchgeführt werden. Hierbei verändern sich die Grenzen der inneren Schleifen (siehe nächste Seite).

Tridiagonalmatrizen

Eine Bandmatrix der Bandbreite $m = 1$ heißt *Tridiagonalmatrix*.
Zur Lösung eines linearen GLS mit *Tridiagonalmatrix* kann der Gauß-Algorithmus in spezieller Form (vgl- Literatur), oder bei symmetrischen, positiv definiten Tridiagonalmatrizen das Cholesky-Verfahren für Bandmatrizen der Bandbreite $m = 1$ benutzt werden.

$$\begin{pmatrix} * & * & & 0 \\ * & \ddots & \ddots & \\ & \ddots & \ddots & * \\ 0 & & * & * \end{pmatrix}$$

Tridiagonalmatrix

2. Cholesky-Verfahren für *symmetrische , positiv definite Bandmatrizen der Bandbreite m*

1. LDL^T-Zerlegung

für $i = 1$ bis n
  für $k = \max(1, i-m)$ bis $(i-1)$
    $s = 0$
    für $j = \max(1, i-m)$ bis $(k-1)$
      $s = s + a_{i,j-i+m+1} * a_{k,j-k+m+1}$
    Ende $j-$Schleife
    $a_{i,k-i+m+1} = a_{i,k-i+m+1} - s$
  Ende $k-$Schleife
  $diag = a_{i,m+1}$
  für $k = \max(1, i-m)$ bis $(i-1)$
    $h = a_{i,k-i+m+1}/a_{k,m+1}$
    $diag = diag - a_{i,k-i+m+1} * h$
    $a_{i,k-i+m+1} = h$
  Ende $k-$Schleife
  falls $diag < 10^{-10} * |a_{i,m+1}|$ $\Rightarrow$ STOP , Matrix nicht positiv definit
  $a_{i,m+1} = diag$
Ende $i-$Schleife

2. Vorwärtseinsetzen und Division durch d_{ii}

für $i = 1$ bis n
  $s = 0$
  für $j = \max(1, i-m)$ bis $(i-1)$
    $s = s + a_{i,j-i+m+1} * b_j$
  Ende $j-$Schleife
  $b_i = b_i - s$
Ende $i-$Schleife
für $i = 1$ bis n
  $b_i = b_i/a_{i,m+1}$
Ende $i-$Schleife

3. Rückwärtseinsetzen

für $i = n$ bis 1 (rückwärts)
  $s = 0$
  für $j = (i+1)$ bis $\min(n, i+m)$
    $s = s + a_{j,i-j+m+1} * b_j$
  Ende $j-$Schleife
  $b_i = b_i - s$
Ende $i-$Schleife

Schwach besetzte Matrizen

Eine Matrix heißt *schwach besetzt*, wenn sehr viele Nullen auftreten, aber keine spezielle Bandstruktur vorhanden ist. Für solche *schwach besetzten* Matrizen benutzt man eine spezielle Speichertechnik, die *Skyline-Technik*:

Beispiel: *symmetrische Matrix, nur der obere Teil wird gespeichert*

$$\begin{pmatrix} 10 & 2 & & & & & 6 & \\ & 20 & & 4 & & & 0 & \\ & & 30 & 0 & & & 0 & \\ & & & 40 & 5 & & 6 & \\ & & & & 50 & 0 & 7 & \\ & & & & & 60 & 7 & \\ & & & & & & 70 & \end{pmatrix}$$

$A = (10, 20, 2, 30, 40, 0, 4, 50, 5, 60, 0, 6, 0, 0, 6, 70, 7, 7)$.

Man speichert zusätzlich die Position der Diagonalelemente auf einem Index-Feld:
$IND = (1, 2, 4, 5, 8, 10, 16, 19)$.

3. Iterative Verfahren

Da der Rechenaufwand beim Gauß-Algorithmus und auch beim Cholesky-Verfahren $\approx n^3$ Operationen (bei voll besetzten Matrizen) beträgt, erhält man bei großen linearen GLS *sehr lange Rechenzeiten*. Um diesen Rechenaufwand zu verringern, benutzt man zur Lösung *großer GLS* besser *iterative Verfahren*. In diesem Kapitel werden wir iterative Verfahren behandeln, deren Grundlage das *Fixpunktverfahren* ist. In einem späteren Kapitel (Kap. VII , Gradientenverfahren) behandeln wir iterative Verfahren, deren Grundlage das Gradientenverfahren ist. Zunächst befassen wir uns mit dem allgemeinen *Fixpunktverfahren*:

Fixpunktverfahren

Gegeben: $\vec{g} : B \subset I\!R^n \to B \quad \text{mit} \quad \vec{g} = \begin{pmatrix} g_1 \\ \vdots \\ g_n \end{pmatrix} \quad , \quad \vec{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} \in B \ .$

Gesucht: *Fixpunkt* $\vec{c} \in B$, d.h.: $\vec{g}(\vec{c}) = \vec{c}$.

Fixpunktverfahren

$\vec{x}_0 \in B$ Startvektor
$\vec{x}_{r+1} = \vec{g}(\vec{x}_r) \quad , \quad r = 0, 1, \ldots$

Fragen:

1. Unter welchen Voraussetzungen *konvergiert* das Fixpunktverfahren gegen den gesuchten Fixpunkt $\vec{c}$? Ist $\vec{c}$ der einzige Fixpunkt von $\vec{g}$ in B ?
2. Wie groß ist im Falle der Konvergenz der Fehler $\|\vec{x}_r - \vec{c}\|_\infty$?

Antworten auf diese Fragen liefert der folgende *Fixpunktsatz*:

Satz 1.10 : *Fixpunktsatz*

Die vektorwertige Funktion $\vec{g} : B \subset \mathbb{R}^n \to B$ besitze stetige partielle Ableitungen in B. Für die *Funktionalmatrix* $J_{\vec{g}}(\vec{x}) = \begin{pmatrix} \frac{\partial g_1}{\partial x_1}(\vec{x}) & \dots & \frac{\partial g_1}{\partial x_n}(\vec{x}) \\ \vdots & \ddots & \vdots \\ \frac{\partial g_n}{\partial x_1}(\vec{x}) & \dots & \frac{\partial g_n}{\partial x_n}(\vec{x}) \end{pmatrix}$ gelte

$$\boxed{L = \max_{\vec{x} \in B} \|J_{\vec{g}}(\vec{x})\|_\infty < 1}$$

(L heißt *Lipschitzkonstante*).

Dann gilt für das Fixpunktverfahren $\vec{x}_{r+1} = \vec{g}(\vec{x}_r)$ mit beliebigem Startvektor $\vec{x}_0 \in B$:

$$\boxed{\lim_{r \to \infty} \vec{x}_r = \vec{c} \quad , \quad (\vec{c} \text{ Fixpunkt in } B)}$$

$\vec{c}$ ist der *einzige* Fixpunkt von $\vec{g}$ in B.

Es gelten die Fehlerabschätzungen

$$\boxed{\|\vec{x}_r - \vec{c}\|_\infty \le \frac{L^r}{1-L}\|\vec{x}_1 - \vec{x}_0\|_\infty \quad , \quad \|\vec{x}_r - \vec{c}\|_\infty \le \frac{L}{1-L}\|\vec{x}_r - \vec{x}_{r-1}\|_\infty}$$

Beweis :

Wir benutzen die $\infty-$Norm: $\|\vec{x}\|_\infty = \max\limits_{1 \le i \le n} |x_i|$. Die zugehörige Matrixnorm ist dann die Zeilensummennorm: $\|A\|_\infty = \max\limits_{1 \le i \le n} \sum\limits_{j=1}^{n} |a_{ij}|$.

Es gilt

$$\begin{aligned} &\|\vec{x}_{r+1} - \vec{x}_r\|_\infty = \|\vec{g}(\vec{x}_r) - \vec{g}(\vec{x}_{r-1})\|_\infty = \max_{1 \le i \le n} |g_i(\vec{x}_r) - g_i(\vec{x}_{r-1})| \\ &\le \max_{1 \le i \le n} \sum_{j=1}^{n} \Big| \frac{\partial g_i(\vec{\xi})}{\partial x_j} ((\vec{x}_r)_j - (\vec{x}_{r-1})_j) \Big| \le \max_{1 \le i \le n} \sum_{j=1}^{n} \Big| \frac{\partial g_i(\vec{\xi})}{\partial x_j} \Big| \|\vec{x}_r - \vec{x}_{r-1}\|_\infty \\ &\le L\|\vec{x}_r - \vec{x}_{r-1}\|_\infty \ . \end{aligned}$$

Hieraus folgt, wenn man diese Eigenschaft mehrmals anwendet:

$$\|\vec{x}_{r+1} - \vec{x}_r\|_\infty \le L\|\vec{x}_r - \vec{x}_{r-1}\|_\infty \le L^2\|\vec{x}_{r-1} - \vec{x}_{r-2}\|_\infty \le \ldots \le L^r\|\vec{x}_1 - \vec{x}_0\|_\infty \ .$$

Für beliebige $m > r$ gilt dann

$$\begin{aligned} &\|\vec{x}_m - \vec{x}_r\|_\infty \le \|\vec{x}_m - \vec{x}_{m-1}\|_\infty + \|\vec{x}_{m-1} - \vec{x}_{m-2}\|_\infty + \ldots + \|\vec{x}_{r+1} - \vec{x}_r\|_\infty \\ &\le (L^{m-1} + L^{m-2} + \ldots + L^r)\|\vec{x}_1 - \vec{x}_0\|_\infty = \frac{L^r - L^m}{1-L}\|\vec{x}_1 - \vec{x}_0\|_\infty \end{aligned}$$

$\Rightarrow \quad \|\vec{x}_m - \vec{x}_r\|_\infty \leq \dfrac{L^r}{1-L}\|\vec{x}_1 - \vec{x}_0\|_\infty \to 0 \quad$ für $m, r \to \infty \;$ (da $\; 0 < L < 1$).

Also ist $(\vec{x}_r)$ eine konvergente Cauchy-Folge in $B \subset \mathbb{R}^n$. Da $\mathbb{R}^n$ vollständiger normierter Raum (d.h. Banachraum), folgt hieraus:

Es existiert ein Grenzwert $\vec{c} \in B$ mit $\lim\limits_{r\to\infty} \vec{x}_r = \vec{c}$.

Da $\vec{g}$ stetig in B, gilt: $\lim\limits_{r\to\infty} \vec{g}(\vec{x}_r) = \vec{g}(\vec{c})$, also

$\vec{c} = \lim\limits_{r\to\infty} \vec{x}_{r+1} = \lim\limits_{r\to\infty} \vec{g}(\vec{x}_r) = \vec{g}(\vec{c}) \quad \Rightarrow \quad \vec{g}(\vec{c}) = \vec{c}$.

Also ist $\vec{c}$ Fixpunkt von $\vec{g}$ in B.

Daß $\vec{c}$ der einzige Fixpunkt in B ist, sieht man folgendermaßen:

Annahme: $\vec{d}$ sei weiterer Fixpunkt von $\vec{g}$ in B, dann gilt

$\|\vec{d} - \vec{c}\|_\infty = \|\vec{g}(\vec{d}) - \vec{g}(\vec{c})\|_\infty \leq L\|\vec{d} - \vec{c}\|_\infty < \|\vec{d} - \vec{c}\|_\infty$

(nach obiger Eigenschaft (siehe letzte Seite), und weil $L < 1$ ist)

$\Rightarrow \quad \|\vec{d} - \vec{c}\|_\infty < \|\vec{d} - \vec{c}\|_\infty \quad \Rightarrow$ Widerspruch $\Rightarrow \quad \vec{c}$ ist einziger Fixpunkt.

Aus der obigen Ungleichung

$$\|\vec{x}_m - \vec{x}_r\|_\infty \leq \frac{L^r}{1-L}\|\vec{x}_1 - \vec{x}_0\|_\infty \quad \text{folgt für} \quad m \to \infty$$

$$\|\vec{c} - \vec{x}_r\|_\infty \leq \frac{L^r}{1-L}\|\vec{x}_1 - \vec{x}_0\|_\infty \; .$$

Wählen wir $\vec{x}_{r-1}$ als Startvektor, so folgt hieraus die Fehlerabschätzung

$$\|\vec{c} - \vec{x}_r\|_\infty \leq \frac{L}{1-L}\|\vec{x}_r - \vec{x}_{r-1}\|_\infty \; .$$

Anwendung auf lineare GLS

Wir benutzen nun das Fixpunktverfahren, um gewisse lineare GLS näherungsweise zu lösen:

1. Gesamtschrittverfahren

Gegeben sei das lineare GLS $A\vec{x} = \vec{b}$ mit der regulären $(n,n)-$Matrix A. Wir zerlegen die Matrix A folgendermaßen

$$A = U + D + W \quad \text{mit} \quad U = \begin{pmatrix} 0 & \dots & \dots & 0 \\ a_{21} & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ a_{n1} & \dots & a_{n,n-1} & 0 \end{pmatrix} \; ,$$

$$D = \begin{pmatrix} a_{11} & 0 & \dots & 0 \\ 0 & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ 0 & \dots & 0 & a_{nn} \end{pmatrix} \; , \quad W = \begin{pmatrix} 0 & a_{12} & \dots & a_{1n} \\ \vdots & \ddots & & \vdots \\ \vdots & & \ddots & a_{n-1,n} \\ 0 & \dots & \dots & 0 \end{pmatrix} \; .$$

Dann erhalten wir anstelle des GLS $A\vec{x} = \vec{b}$:

$(U + D + W)\vec{x} = \vec{b} \quad \Rightarrow \quad D\vec{x} = \vec{b} - U\vec{x} - W\vec{x} \quad \Rightarrow$

$$\boxed{\vec{x} = D^{-1}(\vec{b} - U\vec{x} - W\vec{x}) =: \vec{g}(\vec{x}) \quad (\vec{g} \text{ Fixpunktfunktion})}$$

Koordinatenweise ergibt dies

$$\boxed{x_i = \frac{1}{a_{ii}}\Big(b_i - \sum_{j=1}^{i-1} a_{ij}x_j - \sum_{j=i+1}^{n} a_{ij}x_j\Big) = g_i(\vec{x}) \quad , \quad (i = 1, \ldots, n)}$$

Für die Elemente $\frac{\partial g_i}{\partial x_j}(\vec{x})$ der Funktionalmatrix $J_{\vec{g}}(\vec{x})$ erhalten wir:

$$\frac{\partial g_i}{\partial x_j}(\vec{x}) = \begin{cases} 0 & \text{,falls } j = i \\ -\dfrac{a_{ij}}{a_{ii}} & \text{,falls } j \neq i \ . \end{cases}$$

Also gilt für die Lipschitzkonstante L

$L = \max\limits_{1 \leq i \leq n} \sum\limits_{j=1, j\neq i}^{n} \dfrac{|a_{ij}|}{|a_{ii}|}$. Die *Konvergenzvoraussetzung* $L < 1$ bedeutet also, daß die Matrix A *diagonaldominant* sein muß:

$$\boxed{|a_{ii}| > \sum_{j=1, j\neq i}^{n} |a_{ij}| \quad \forall\ 1 \leq i \leq n \quad \text{(d.h.: } A \text{ ist diagonaldominant)}}$$

Ist also A *diagonaldominant*, so konvergiert dieses Fixpunktverfahren. Es wird *Gesamtschrittverfahren* (oder auch *Jacobi-Verfahren*) genannt. Koordinatenweise lautet das

Gesamtschrittverfahren

Wähle einen Startvektor $\vec{x}_0$ (z.B.: $(\vec{x}_0)_i = 1 \ , \ i = 1, \ldots, n)$).
Berechne für $r = 0, 1, \ldots$

$$\boxed{(\vec{x}_{r+1})_i = \frac{1}{a_{ii}}\Big(b_i - \sum_{j=1}^{i-1} a_{ij}(\vec{x}_r)_j - \sum_{j=i+1}^{n} a_{ij}(\vec{x}_r)_j\Big) \quad , \quad (i = 1, \ldots, n)}$$

bis $\|\vec{x}_{r+1} - \vec{x}_r\|_\infty < \epsilon$ oder r zu groß (d.h.: keine Konvergenz).

Es gilt mit obiger Lipschitzkonstanten L die Fehlerabschätzung für die Näherung $\vec{x}_{r+1}$, wenn $\vec{x}^*$ die gesuchte Lösung des GLS $A\vec{x} = \vec{b}$ ist:

$$\|\vec{x}_{r+1} - \vec{x}^*\|_\infty \leq \frac{L^{r+1}}{1 - L}\|\vec{x}_1 - \vec{x}_0\|_\infty \ .$$

Je kleiner L, desto besser ist die Konvergenz des Gesamtschrittverfahrens, also:
je größer die "Diagonaldominanz" von A ist, desto besser ist die Konvergenz.

Rechenaufwand

Bei jedem Iterationsschritt wird eine Multiplikation "Matrix $*$ Vektor" durchgeführt. Das sind $\approx n^2$ Rechenoperationen. Werden weniger als n Iterationsschritte benötigt, so ist der Gesamtrechenaufwand $< n^3$ Operatinen, also weniger als beim Gauß-Algorithmus. Ein weiterer Vorteil des Gesamtschrittverfahrens liegt darin, daß es sehr gut zur "Parallelisierung" geignet ist, da jede Multiplikation "Spalte $*$ Vektor" ($\hat{=}$ Skalarprodukt) getrennt durchgeführt werden kann, also auf einem Parallelrechner gut zu realisieren ist.

2. Einzelschrittverfahren

Eine andere Aufteilung der Matrix A führt zum *Einzelschrittverfahren* (oder *Gauß-Seidel-Verfahren*):

$(U + D)\vec{x} = \vec{b} - W\vec{x} \quad \Rightarrow$

$$\boxed{\vec{x} = (U + D)^{-1}(\vec{b} - W\vec{x}) =: \vec{g}(\vec{x}) \quad (\vec{g} \text{ Fixpunktfunktion})}$$

Das Fixpunktverfahren lautet dann:

$\vec{x}_{r+1} = (U + D)^{-1}(\vec{b} - W\vec{x}_r) \quad \Rightarrow \quad (U + D)\vec{x}_{r+1} = \vec{b} - W\vec{x}_r$
$\Rightarrow \quad \vec{x}_{r+1} = D^{-1}(\vec{b} - U\vec{x}_{r+1} - W\vec{x}_r)$.

Koordinatenweise erhalten wir damit das

Einzelschrittverfahren

Wähle einen Startvektor $\vec{x}_0$ (z.B.: $(\vec{x}_0)_i = 1 \ , \ i = 1, \ldots, n)$).
Berechne für $r = 0, 1, \ldots$

$$\boxed{(\vec{x}_{r+1})_i = \frac{1}{a_{ii}}\Big(b_i - \sum_{j=1}^{i-1} a_{ij}(\vec{x}_{r+1})_j - \sum_{j=i+1}^{n} a_{ij}(\vec{x}_r)_j\Big) \ , \quad (i = 1, \ldots, n)}$$

bis $\|\vec{x}_{r+1} - \vec{x}_r\|_\infty < \epsilon$ oder r zu groß (d.h.: keine Konvergenz).

Der Unterschied zum Gesamtschrittverfahren besteht darin, daß auf der rechten Seite schon die Koordinaten des neuen Iterationsvektors $\vec{x}_{r+1}$ auftreten. Dabei ist zu beachten, daß nur die schon berechneten Werte auf der rechten Seite benutzt werden. Für $i = 1$ ist die erste Summe auf der rechten Seite eine leere Summe; für $i = n$ ist die zweite Summe auf der rechten Seite eine leere Summe.

Konvergenzvoraussetzung

Man kann zeigen (vgl. Literatur), daß auch das Einzelschrittverfahren konvergiert, wenn die Matrix A *diagonaldominant* ist. In diesem Fall gilt die gleiche Fehlerabschätzung wie beim Gesamtschrittverfahren.

Man kann sogar zeigen, daß eine etwas schwächere Voraussezung für die Konvergenz genügt: Die Matrix A muß "fast" diagonaldominant sein, d.h. $\approx$: in einigen Zeilen genügt die Voraussetzung: $|a_{ii}| = \sum_{j=1, j\neq i}^{n} |a_{ij}|$ anstelle des echten $>$ Zeichens.

Beispiel (aus Kap. VIII , Differenzenverfahren)

A sei die folgende symmetrische $(25, 25)-$Matrix

$$A = \begin{pmatrix} B & D & & & \\ D & B & D & & \\ & D & B & D & \\ & & D & B & D \\ & & & D & B \end{pmatrix}$$

mit den Teilmatrizen (jeweils symmetrische $(5, 5)-$Matrizen)

$$B = \begin{pmatrix} 4 & -1 & & & \\ -1 & 4 & -1 & & \\ & -1 & 4 & -1 & \\ & & -1 & 4 & -1 \\ & & & -1 & 4 \end{pmatrix} \quad , \quad D = \begin{pmatrix} -1 & & & & \\ & -1 & & & \\ & & -1 & & \\ & & & -1 & \\ & & & & -1 \end{pmatrix} .$$

A ist also eine Bandmatrix der Bandbreite $m = 5$.

A ist "fast" diagonaldominant, in den äußeren Zeilen gilt $|a_{ii}| > \sum_{j=1, j\neq i}^{n} |a_{ij}|$,

in den mittleren Zeilen gilt nur "=".

Die rechte Seite sei $\vec{b}$ mit $b_i = -1/18 \quad \forall i = 1, \ldots, n$.

Beim Startvektor $\vec{x}_0$ mit $(\vec{x}_0)_i = 1 \quad \forall i = 1, \ldots, n$ und einem Fehler $\epsilon = 10^{-8}$ wurden 68 Iterationen benötigt. Beim Gesamtschrittverfahren bekommt man die gleiche Lösung sogar erst nach 120 Iterationen. Dieses Beispiel zeigt uns, daß es sinnvoll ist, das Einzelschrittverfahren so zu verbessern, daß die Anzahl der benötigten Iterationen wesentlich verkleinert wird. Dieses Ziel wird mit Hilfe eines *Relaxationsparameters* ω erreicht. Das zugehörige Verfahren heißt

3. Überrelaxationsverfahren (oder SOR-Verfahren)

Wir führen einen zusätzlichen *Relaxationsparameter* ω folgendermaßen ein:

$((1 - \frac{1}{\omega} + \frac{1}{\omega})D + U + W)\vec{x} = \vec{b} \quad \Rightarrow \quad (U + \frac{1}{\omega}D)\vec{x} = \vec{b} - (1 - \frac{1}{\omega})D\vec{x} - W\vec{x} \quad \Rightarrow$

$$\boxed{\vec{x} = (U + \tfrac{1}{\omega}D)^{-1}(\vec{b} - (1 - \tfrac{1}{\omega})D\vec{x} - W\vec{x}) =: \vec{g}(\vec{x}) \quad (\vec{g} \text{ Fixpunktfunktion})}$$

Das Fixpunktverfahren lautet dann:

$(U + \frac{1}{\omega}D)\vec{x}_{r+1} = \vec{b} - (1 - \frac{1}{\omega})D\vec{x}_r - W\vec{x}_r$

$\Rightarrow \quad \vec{x}_{r+1} = \omega D^{-1}(\vec{b} - (1 - \frac{1}{\omega})D\vec{x}_r - U\vec{x}_{r+1} - W\vec{x}_r)$.

Koordinatenweise erhalten wir damit das *Überrelaxationsverfahren*

Überrelaxationsverfahren

Wähle einen Startvektor $\vec{x}_0$ (z.B.: $(\vec{x}_0)_i = 1$, $i = 1, \ldots, n$)).
Berechne für $r = 0, 1, \ldots$

$$(\vec{x}_{r+1})_i = \frac{\omega}{a_{ii}}\Big(b_i - (1 - \frac{1}{\omega})a_{ii}(\vec{x}_r)_i - \sum_{j=1}^{i-1} a_{ij}(\vec{x}_{r+1})_j - \sum_{j=i+1}^{n} a_{ij}(\vec{x}_r)_j\Big)$$

$i = 1, \ldots, n$

bis $\|\vec{x}_{r+1} - \vec{x}_r\|_\infty < \epsilon$ oder r zu groß (d.h.: keine Konvergenz).

Für $\omega = 1$ erhalten wir das Einzelschrittverfahren. Man versucht $\omega \geq 1$ so zu wählen, daß das Verfahren möglichst schnell konvergiert.

Algorithmus: *Überrelaxationsverfahren*

Wähle Startvektor x_i , $i = 1, \ldots, n$ (z.B.: $x_i = 1$, $i = 1, \ldots, n$)
Berechne für $k = 1, 2, \ldots$ {k Anzahl der Iterationen}
  für $i = 1$ bis n
    $s = 0$
    für $j = 1$ bis $(i - 1)$ {leere Schleife für $i = 1$}
      $s = s - a_{ij} * u_j$
    Ende $j-$Schleife
    für $j = (i + 1)$ bis n {leere Schleife für $i = n$}
      $s = s - a_{ij} * x_j$
    Ende $j-$Schleife
    $u_i = \omega * (b_i + s)/a_{ii} + (1 - \omega) * x_i$
  Ende $i-$Schleife
  $max = 0$
  für $i = 1$ bis n
    falls $|u_i - x_i| > max \quad \Rightarrow \quad max = |u_i - x_i|$
  Ende $i-$Schleife
  für $i = 1$ bis n: $x_i = u_i$
bis $max < \epsilon$ oder k zu groß

Beispiel von S.26

Startvektor: $\vec{x}_0$ mit $(\vec{x}_0)_i = 1 \quad \forall i = 1, \ldots, n; \quad , \quad \epsilon = 10^{-8}$

$\omega = 1 \quad \Rightarrow$ 68 Iterationen (Einzelschrittverfahren)
$\omega = 1.3 \quad \Rightarrow$ 28 Iterationen
$\omega = 1.35 \quad \Rightarrow$ 22 Iterationen (optimales ω für dieses Beispiel)
$\omega = 1.4 \quad \Rightarrow$ 23 Iterationen.

Für bestimmte Matrizen existieren Aussagen über das optimale ω (vgl. Literatur).

In Kap. VII (Gradientenverfahren) werden wir ein weiteres iteratives Verfahren zur Lösung von linearen GLS behandeln: das cg- bzw. pcg-Verfahren.

II Nichtlineare Gleichungssysteme

Newton-Verfahren

Gegeben: n Gleichungen mit n Unbekannten

$$\begin{aligned} f_1(x_1,\ldots,x_n) &= 0 \\ f_2(x_1,\ldots,x_n) &= 0 \\ &\vdots \\ f_n(x_1,\ldots,x_n) &= 0 \ . \end{aligned}$$

In Matrizenschreibweise erhalten wir für $\vec{f} = \begin{pmatrix} f_1 \\ \vdots \\ f_n \end{pmatrix}$, $\vec{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$,

$$\boxed{\vec{f}(\vec{x}) = \vec{0}}$$

Gesucht: $\vec{x}^*$ Nullstelle von $\vec{f}(\vec{x}) = \vec{0}$.

Voraussetzungen:

Die Funktionen f_i seien in einer Umgebung der gesuchten Nullstelle $\vec{x}^*$ $2-$mal stetig differenzierbar, und die Funktionalmatrix

$$J_{\vec{f}}(\vec{x}) = \begin{pmatrix} \frac{\partial f_1}{\partial x_1}(\vec{x}) & \ldots & \frac{\partial f_1}{\partial x_n}(\vec{x}) \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1}(\vec{x}) & \ldots & \frac{\partial f_n}{\partial x_n}(\vec{x}) \end{pmatrix}$$

von $\vec{f}$ sei in dieser Umgebung *regulär* , d.h.: $J_{\vec{f}}^{-1}(\vec{x})$ existiert.

Sei $\vec{x}_0$ ein Startvektor. Dann führen wir für die Funktionen f_i eine *Taylorentwicklung* bis zur 1.Ordnung durch, d.h.: $\vec{f}$ wird durch das Taylorpolynom 1.Ordnung *linearisiert*:

$$f_i(\vec{x}) = f_i(\vec{x}_0) + \sum_{j=1}^{n} \frac{\partial f_i(\vec{x}_0)}{\partial x_j}((\vec{x})_j - (\vec{x}_0)_j) + R \quad , \quad i = 1,\ldots,n \ .$$

Setzen wir nicht $f_i(\vec{x}) = 0$, sondern das Taylorpolynom 1.Grades $= 0$, so erhalten wir

$$\sum_{j=1}^{n} \frac{\partial f_i(\vec{x}_0)}{\partial x_j}((\vec{x})_j - (\vec{x}_0)_j) = -f_i(\vec{x}_0) \quad , \quad i = 1,\ldots,n \ .$$

Führen wir den Vektor $\vec{z} = \vec{x} - \vec{x}_0$ ein, so lauten diese Gleichungen

$$\left(J_{\vec{f}}(\vec{x}_0)\,\vec{z}\right)_i = -f_i(\vec{x}_0) \quad , \quad i = 1,\ldots,n \ ,$$

mit der Funktionalmatrix $J_{\vec{f}}(\vec{x}_0)$ von $\vec{f}$. In Matrizenschreibweise erhalten wir

$$\boxed{J_{\vec{f}}(\vec{x}_0)\,\vec{z} = -\vec{f}(\vec{x}_0)}$$

Die Lösung dieses linearen GLS sei der Vektor $\vec{z}$,
$\vec{x}_1 = \vec{x}_0 + \vec{z}$ ergibt dann die nächste Näherungslösung.

Allgemein erhalten wir somit das Newton-Verfahren:

Newton-Verfahren

Wähle Startvektor $\vec{x}_0$.
Berechne für $r = 0, 1, \ldots$

$$\boxed{\begin{array}{l} J_{\vec{f}}(\vec{x}_r)\,\vec{z} = -\vec{f}(\vec{x}_r) \quad \text{lineares GLS lösen} \\ \vec{x}_{r+1} = \vec{x}_r + \vec{z} \end{array}}$$

bis $\|\vec{f}(\vec{x}_{r+1})\| < \epsilon$ und $\|\vec{z}\| < \delta$ (oder r zu groß (keine Konvergenz)).

Schreiben wir das lineare GLS $J_{\vec{f}}(\vec{x}_r)\,\vec{z} = -\vec{f}(\vec{x}_r)$ anders auf:

$J_{\vec{f}}(\vec{x}_r)(\vec{x}_{r+1} - \vec{x}_r) = -\vec{f}(\vec{x}_r) \quad \Leftrightarrow \quad \vec{x}_{r+1} = \vec{x}_r - \left(J_{\vec{f}}(\vec{x}_r)\right)^{-1} \vec{f}(\vec{x}_r)$, so sieht man, daß das Newton-Verfahren ein *Fixpunktverfahren* ist. Die Fixpunktfunktion lautet:

$$\boxed{\vec{g}(\vec{x}) = \vec{x} - \left(J_{\vec{f}}(\vec{x})\right)^{-1} \vec{f}(\vec{x})}$$

Um die Konvergenz des Newton-Verfahrens nachzuweisen, müssen wir die Voraussetzungen des Fixpunktsatzes (vgl. Satz 1.10 , S.22) überprüfen.
$\vec{g}$ ist stetig differenzierbar in einer Umgebung $U(\vec{x}^*)$ ($\vec{x}^*$ gesuchte Nullstelle von $\vec{f}$ und damit Fixpunkt von $\vec{g}$) , da $\vec{f}$ $2-$mal stetig differenzierbar ist. Weiter muß gelten: $L = \max\limits_{\vec{x} \in U_0(\vec{x}^*)} \|J_{\vec{g}}(\vec{x})\|_\infty < 1$.
Aus obiger Gleichung folgt:
$J_{\vec{f}}(\vec{x})\,\vec{g}(\vec{x}) = J_{\vec{f}}(\vec{x})\,\vec{x} - \vec{f}(\vec{x})$.
Differentiation nach x_i ergibt

$$\frac{\partial}{\partial x_i}\left(J_{\vec{f}}(\vec{x})\right)\vec{g}(\vec{x}) + J_{\vec{f}}(\vec{x})\,\frac{\partial \vec{g}(\vec{x})}{\partial x_i} = \frac{\partial}{\partial x_i}\left(J_{\vec{f}}(\vec{x})\right)\vec{x} + J_{\vec{f}}(\vec{x})\,\vec{e}_i - \frac{\partial \vec{f}(\vec{x})}{\partial x_i} \ .$$

Setzen wir in diese Gleichung $\vec{x} = \vec{x}^*$ (gesuchte Nullstelle von $\vec{f}$) ein, so erhalten wir, da $\vec{f}(\vec{x}^*) = \vec{0}$ und $\vec{g}(\vec{x}^*) = \vec{x}^*$ (da $\vec{x}^*$ gleichzeitig Fixpunkt von $\vec{g}$)

$$J_{\vec{f}}(\vec{x}^*)\,\frac{\partial \vec{g}(\vec{x}^*)}{\partial x_i} = J_{\vec{f}}(\vec{x}^*)\,\vec{e}_i - \frac{\partial \vec{f}(\vec{x}^*)}{\partial x_i} = \vec{0} \ ,$$

da $J_{\vec{f}}(\vec{x}^*)\,\vec{e}_i$ und $\dfrac{\partial \vec{f}(\vec{x}^*)}{\partial x_i}$ jeweils die $i-$te Spalte von $J_{\vec{f}}(\vec{x}^*)$ ist.
Da $J_{\vec{f}}(\vec{x}^*)$ regulär vorausgesetzt wurde, folgt hieraus

$$\boxed{\frac{\partial \vec{g}(\vec{x}^*)}{\partial x_i} = \vec{0} \quad \forall i = 1, \ldots, n \quad \Leftrightarrow \quad J_{\vec{g}}(\vec{x}^*) = 0 \ \text{ (Nullmatrix)}}$$

Da $J_{\vec{g}}(\vec{x}^*) = 0 \quad \Rightarrow \quad \|J_{\vec{g}}(\vec{x}^*)\|_\infty = 0$.
Da $\dfrac{\partial g_i}{\partial x_j}$ stetig in einer Umgebung von $\vec{x}^*$, existiert eine Umgebung $U_0(\vec{x}^*)$ mit
$L = \max\limits_{\vec{x} \in U_0(\vec{x}^*)} \|J_{\vec{g}}(\vec{x})\|_\infty < 1$.
Also gilt der folgende Satz:

Satz 2.1 : *Konvergenz des Newton-Verfahrens*
Ist $\vec{f}$ 2−mal stetig differenzierbar in einer Umgebung der gesuchten Nullstelle $\vec{x}^*$, und ist dort die Funktionalmatrix $J_{\vec{f}}(\vec{x})$ von $\vec{f}$ *regulär*, so *konvergiert* das Newton-Verfahren gegen die gesuchte Nullstelle $\vec{x}^*$, falls der Startvektor $\vec{x}_0$ "nahe" genug bei $\vec{x}^*$ gewählt wird.

Problematik: Da man in der Praxis nicht weiß, was "nahe" genug heißt, ist die Wahl eines geeigneten Startvektors $\vec{x}_0$ oft nicht sehr einfach (vgl. spätere Beispiele).

Unter der Voraussetzung der Konvergenz des Newton-Verfahrens und unter der zusätzlichen Voraussetzung, daß $\vec{f}$ sogar 3−mal stetig differenzierbar in einer Umgebung von $\vec{x}^*$ ist, kann man sogar quadratische Konvergenz zeigen:

Satz 2.2 : *Quadratische Konvergenz*
Ist $\vec{f}$ 3−mal stetig differenzierbar in einer Umgebung der gesuchten Nullstelle $\vec{x}^*$, und konvergiert das Newton-Verfahren, so gilt *quadratische Konvergenz* , d.h.:

$$\boxed{\|\vec{x}_{r+1} - \vec{x}^*\|_\infty \leq K \|\vec{x}_r - \vec{x}^*\|_\infty^2 \quad , \quad K > 0}$$

Quadratische Konvergenz bedeutet: In jedem Iterationsschritt wird die Anzahl der richtigen Stellen ungefähr *verdoppelt*, denn sei
$\|\vec{x}_r - \vec{x}^*\|_\infty < 10^{-d} \quad \Rightarrow \quad \|\vec{x}_{r+1} - \vec{x}^*\|_\infty < K \cdot 10^{-2d}$.

Beweis von Satz 2.2
Wir führen für die Fixpunktfunktion $\vec{g}$ eine Taylorentwicklung um $\vec{x}^*$ bis zu 2.Ordnung durch

$$(\vec{x}_{r+1})_i = g_i(\vec{x}_r) = g_i(\vec{x}^*) + \sum_{j=1}^{n} \frac{\partial g_i(\vec{x}^*)}{\partial x_j}((\vec{x}_r)_j - (\vec{x}^*)_j)$$
$$+\frac{1}{2}\sum_{j=1}^{n}\sum_{k=1}^{n} \frac{\partial^2 g_i(\vec{\xi})}{\partial x_j \partial x_k}((\vec{x}_r)_j - (\vec{x}^*)_j)((\vec{x}_r)_k - (\vec{x}^*)_k) \ .$$

Da $g_i(\vec{x}^*) = (\vec{x}^*)_i$ und $\dfrac{\partial g_i(\vec{x}^*)}{\partial x_j} = 0 \quad \forall 1 \leq i, j \leq n$ (da $J_{\vec{g}}(\vec{x}^*) = 0$), folgt hieraus

$$|(\vec{x}_{r+1})_i - (\vec{x}^*)_i| \le \frac{1}{2}\sum_{j=1}^{n}\sum_{k=1}^{n}\Big|\frac{\partial^2 g_i(\vec{\xi})}{\partial x_j \partial x_k}\Big| \|\vec{x}_r - \vec{x}^*\|_\infty^2 \quad \forall i = 1,\dots,n$$

$$\Rightarrow \quad \|\vec{x}_{r+1} - \vec{x}^*\|_\infty \le K\|\vec{x}_r - \vec{x}^*\|_\infty^2 \ .$$

Beispiel $n = 2$: zwei Gleichungen mit zwei Unbekannten

$f_1(x,y) = 0$
$f_2(x,y) = 0$.

Die Funktionalmatrix von $\vec{f} = \begin{pmatrix} f_1 \\ f_2 \end{pmatrix}$ lautet: $J_{\vec{f}}(x,y) = \begin{pmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y} \\ \frac{\partial f_2}{\partial x} & \frac{\partial f_2}{\partial y} \end{pmatrix}_{(x,y)}$.

Das lineare GLS $J_{\vec{f}}(x_r,y_r)\,\vec{z} = -\vec{f}(x_r,y_r)$ geht über in
$\vec{z} = -J_{\vec{f}}^{-1}(x_r,y_r)\,\vec{f}(x_r,y_r)$.
Mit $\begin{pmatrix} x_{r+1} \\ y_{r+1} \end{pmatrix} = \begin{pmatrix} x_r \\ y_r \end{pmatrix} + \vec{z}$ folgt hieraus

$$\begin{pmatrix} x_{r+1} \\ y_{r+1} \end{pmatrix} = \begin{pmatrix} x_r \\ y_r \end{pmatrix} - J_{\vec{f}}^{-1}(x_r,y_r) \begin{pmatrix} f_1(x_r,y_r) \\ f_2(x_r,y_r) \end{pmatrix} .$$

Für eine $(2,2)-$Matrix läßt sich sehr einfach die inverse Matrix berechnen, es gilt

$$A = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \quad \Rightarrow \quad A^{-1} = \frac{1}{ad-bc}\begin{pmatrix} d & -b \\ -c & a \end{pmatrix} .$$

Damit lautet das Newton-Verfahren für den
Spezialfall von $n = 2$ Gleichungen:
$$\begin{aligned} f_1(x,y) &= 0 \\ f_2(x,y) &= 0 \end{aligned}$$

$$\boxed{\begin{pmatrix} x_{r+1} \\ y_{r+1} \end{pmatrix} = \begin{pmatrix} x_r \\ y_r \end{pmatrix} - \frac{1}{f_{1x}f_{2y} - f_{1y}f_{2x}} \begin{pmatrix} f_{2y} & -f_{1y} \\ -f_{2x} & f_{1x} \end{pmatrix}_{(x_r,y_r)} \begin{pmatrix} f_1 \\ f_2 \end{pmatrix}_{(x_r,y_r)}}$$

Beispiel hierzu

$f_1(x,y) = 3y - 2xy - y^2$
$f_2(x,y) = 3x - x^2 - 2xy$.

In diesem Beispiel kann man die Nullstellen von $\vec{f}$ exakt berechnen:
$\begin{pmatrix} 0 \\ 0 \end{pmatrix}$, $\begin{pmatrix} 3 \\ 0 \end{pmatrix}$, $\begin{pmatrix} 0 \\ 3 \end{pmatrix}$, $\begin{pmatrix} 1 \\ 1 \end{pmatrix}$ sind die exakten Nullstellen von $\vec{f}$.

Funktionalmatrix von $\vec{f}$: $J_{\vec{f}}(x,y) = \begin{pmatrix} -2y & 3-2x-2y \\ 3-2x-2y & -2x \end{pmatrix}$

$$J_{\vec{f}}^{-1}(x,y) = \frac{1}{\det J_{\vec{f}}(x,y)} \begin{pmatrix} -2x & -3+2x+2y \\ -3+2x+2y & -2y \end{pmatrix} .$$

Für dieses Beispiel erhalten wir folgende Ergebnisse

r	x_r	y_r
0	1.0000000000	2.0000000000
1	−1.0000000000	4.0000000000
2	−0.2000000000	3.2000000000
3	−0.0117647059	3.0117647059
4	−0.0000457771	3.0000457771
5	−0.0000000007	3.0000000007
6	0.0000000000	3.0000000000

r	x_r	y_r
0	5.0000000000	2.0000000000
1	3.1481481481	1.0370370370
2	2.5603843739	0.4272538510
3	3.0996747240	−0.0935314446
4	3.0034317253	−0.0030725371
5	3.0000046482	−0.0000038721
6	3.0000000000	0.0000000000

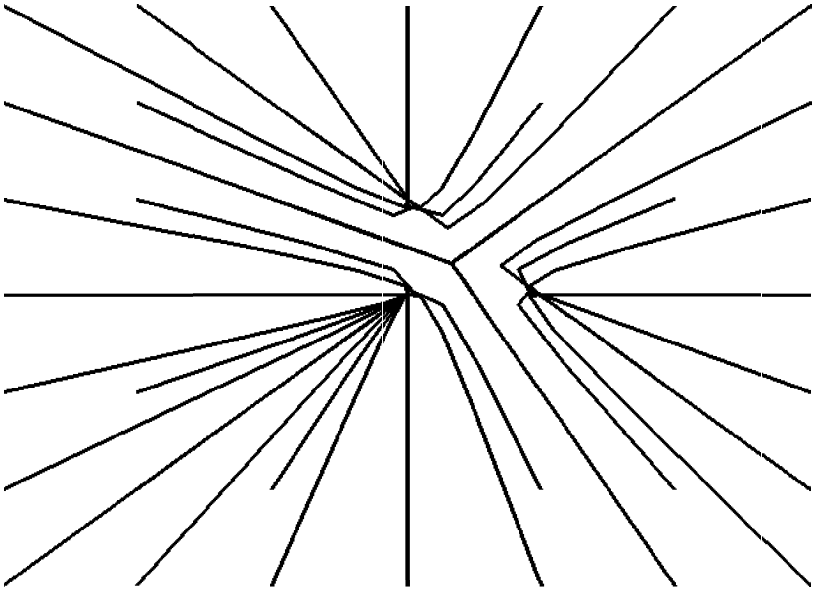
r	x_r	y_r
0	1.0000000000	1.8000000000
1	3.9090909091	−2.7818181818
2	2.5958621188	−0.5797602927
3	2.5024042686	0.2206499611
4	3.2447414925	−0.1140206816
5	3.0240283147	−0.0114461709
6	3.0002816995	−0.0001363896
7	3.0000000397	−0.0000000194
8	3.0000000000	0.0000000000

r	x_r	y_r
0	−2.0000000000	−2.0000000000
1	−0.8000000000	−0.8000000000
2	−0.2461538462	−0.2461538462
3	−0.0406026963	−0.0406026963
4	−0.0015247602	−0.0015247602
5	−0.0000023178	−0.0000023178
6	0.0000000000	0.0000000000

r	x_r	y_r
0	1.0000000000	1.4000000000
1	1.1355932203	0.8779661017
2	0.9910564603	0.9975685216
3	0.9999924172	1.0000660352
4	1.0000000026	0.9999999983
5	1.0000000000	1.0000000000

r	x_r	y_r
0	2.0000000000	2.0000000000
1	1.3333333333	1.3333333333
2	1.0666666667	1.0666666667
3	1.0039215686	1.0039215686
4	1.0000152590	1.0000152590
5	1.0000000002	1.0000000002
6	1.0000000000	1.0000000000

Im folgenden Bild sind die "Wege" aufgezeichnet, die bei der Durchführung des Newton-Verfahrens bei verschiedenen Startwerten zurückgelegt werden:



Nachteil des Newton-Verfahrens

Bei jedem Iterationsschritt müssen alle partiellen Ableitungen $\frac{\partial f_i(\vec{x}_r)}{\partial x_j}$ berechnet werden. Die Koeffizientenmatrix $J_{\vec{f}}(\vec{x}_r)$ ist also bei jedem Iterationsschritt neu zu berechnen, und damit ist in jedem Iterationsschritt ein komplettes lineares GLS zu lösen.

Vereinfachungen des Newton-Verfahrens

1. Man berechnet nur für den Startvektor $\vec{x}_0$ die Matrix $J_{\vec{f}}(\vec{x}_0)$ und ersetzt dann $J_{\vec{f}}(\vec{x}_r)$ immer durch $J_{\vec{f}}(\vec{x}_0)$. Dann muß nur einmal die LR-Zerlegung berechnet werden. Bei den einzelnen Iterationsschritten muß dann nur noch das Vorwärts- und Rückwärtseinsetzen durchgeführt werden.
Dieses Verfahren ist nur sinnvoll bei sehr guten Startvektoren. Man erhält schlechtere Konvergenz oder Divergenz.

2. Man berechnet nur alle $3-5$ Schritte die Funktionalmatrix $J_{\vec{f}}(\vec{x}_r)$ neu. Auch dies führt i.a. zu schlechterer Konvergenz oder Divergenz.

3. Man ersetzt die partiellen Ableitungen durch Differenzenquotienten. Diese Möglichkeit sollte nur dann benutzt werden, wenn die partiellen Ableitungen nicht oder nur sehr aufwendig berechnet werden können. Auch diese Methode führt i.a. zu schlechterer Konvergenz oder Divergenz.

Verbesserung: **Gedämpftes Newton-Verfahren**

Bei ungünstigen Startvektoren kann es passieren, daß das Newton-Verfahren divergiert. Um sicher zu gehen, daß der nächste Iterationsvektor $\vec{x}_{r+1}$ "besser" ist als $\vec{x}_r$, führt man einen *Dämpfungsparameter* α ein:

Gedämpftes Newton-Verfahren

Wähle Startvektor $\vec{x}_0$.
Berechne für $r = 0, 1, \ldots$

$J_{\vec{f}}(\vec{x}_r)\,\vec{z} = -\vec{f}(\vec{x}_r)$ lineares GLS lösen

wähle $\alpha = 1, 1/2, 1/4, 1/8 \ldots,$ bis $\|\vec{f}(\vec{x}_r + \alpha\vec{z})\| < (1 - \frac{\alpha}{4})\|\vec{f}(\vec{x}_r)\|$

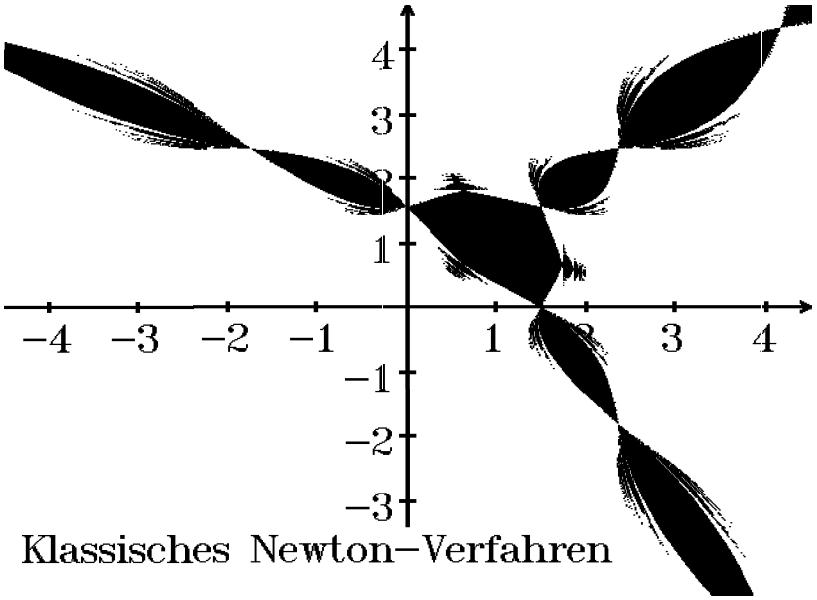
$\vec{x}_{r+1} = \vec{x}_r + \alpha\vec{z}$

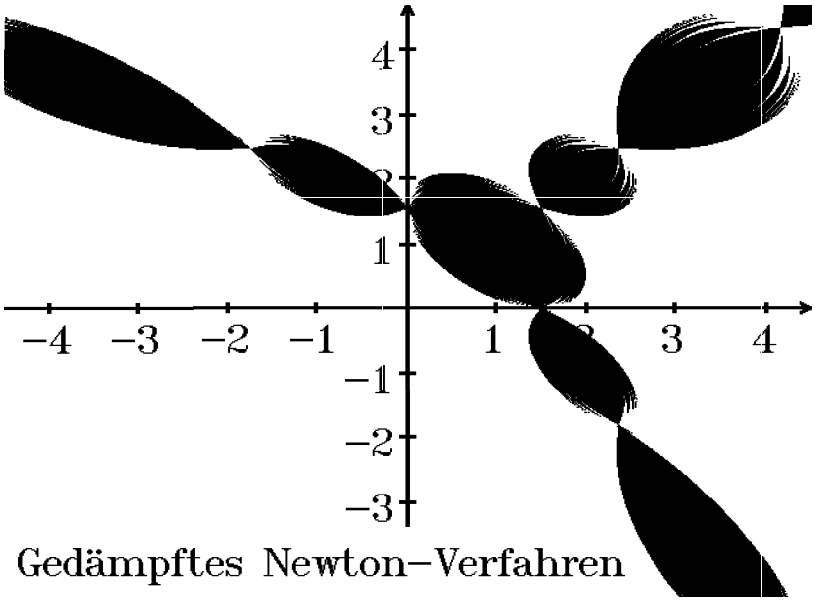
bis $\|\vec{f}(\vec{x}_{r+1})\| < \epsilon$ und $\|\vec{z}\| < \delta$
(oder α zu klein oder r zu groß (keine Konvergenz)).

Damit ist gesichert, daß $\vec{x}_{r+1}$ eine "bessere" Näherung als $\vec{x}_r$ ist. Wird hierbei α zu klein, so ist die Konvergenz nur sehr langsam. Liegt α im Bereich der Rechengenauigkeit, so ist das Verfahren zu beenden. In diesem Fall sollte man einen anderen

Startvektor wählen. In der obigen Ungleichung tritt der Faktor $(1 - \frac{\alpha}{4})$ auf, damit nicht nur auf Grund von Rundungsfehlern der Wert "besser" geworden ist.

In den beiden folgenden Bildern ist der Unterschied zwischen klassischem - und gedämpftem Newton-Verfahren zu erkennen. Für das Beispiel von S.31 sind jeweils die Punkte gekennzeichnet, die als Startvektoren die Nullstelle $\binom{1}{1}$ angesteuert haben:





Anwendungsbeispiel *Berechnung von Extrema*

Gesucht: Die *relativen Extrema* einer Funktion $h(x_1, x_2, \ldots, x_n)$.

Notwendige Bedingung: $\operatorname{grad} h(\vec{x}) = \vec{0}$.

Das sind n Gleichungen für die n Unbekannten $x_1, \ldots, x_n$, also

$h_{x_1}(x_1, \ldots, x_n) = 0$
$h_{x_2}(x_1, \ldots, x_n) = 0$
$\vdots$
$h_{x_n}(x_1, \ldots, x_n) = 0$.

Die Funktionalmatrix enthält dann die 2−ten partiellen Ableitungen $\left(\dfrac{\partial^2 h(\vec{x})}{\partial x_i \partial x_j}\right)$.

Beispiel hierzu: $n = 2$: *Gegeben*: $h(x, y)$, 2−mal stetig differenzierbar.
Gesucht: relative Extrema.

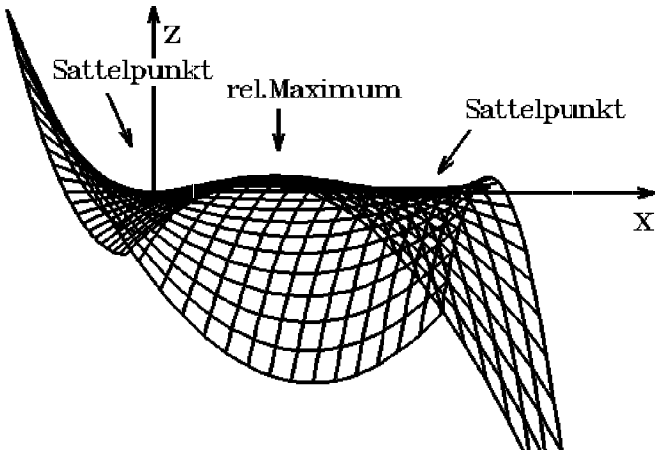
Notwendige Bedingung:
$$\begin{aligned} h_x(x, y) &= 0 \\ h_y(x, y) &= 0 \ . \end{aligned}$$

Für diesen Fall lautet das Newton-Verfahren

$$\begin{pmatrix} x_{r+1} \\ y_{r+1} \end{pmatrix} = \begin{pmatrix} x_r \\ y_r \end{pmatrix} - \frac{1}{h_{xx}h_{yy} - h_{xy}^2} \begin{pmatrix} h_{yy} & -h_{xy} \\ -h_{xy} & h_{xx} \end{pmatrix}_{(x_r, y_r)} \begin{pmatrix} h_x \\ h_y \end{pmatrix}_{(x_r, y_r)} .$$

Beispiel hierzu: $h(x, y) = xy(3 - x - y) = 3xy - x^2y - xy^2$
$\Rightarrow \quad h_x = 3y - 2xy - y^2 \quad , \quad h_y = 3x - x^2 - 2xy$ (vgl. Beispiel S.31).

$h_x = h_y = 0 \quad \Rightarrow \quad \begin{pmatrix} 0 \\ 0 \end{pmatrix}$, $\begin{pmatrix} 3 \\ 0 \end{pmatrix}$, $\begin{pmatrix} 0 \\ 3 \end{pmatrix}$, $\begin{pmatrix} 1 \\ 1 \end{pmatrix}$ sind mögliche Exrema. Mit Hilfe der hinreichenden Bedingung kann man sehen, daß nur bei $\begin{pmatrix} 1 \\ 1 \end{pmatrix}$ ein relatives Extremum (Maximum) vorliegt. An den anderen Stellen sind keine Extrema sondern Sattelpunkte.



Bemerkung : Bei Extremwertaufgaben können auch andere Verfahren benutzt werden, z.B.: Gradientenverfahren (vgl. Kap. VII).

III Eigenwertberechnung bei Matrizen

Gegeben: Eine reelle $(n,n)-$Matrix A (mit $a_{ij} \in \mathbb{R}$).

Gesucht: $\lambda \in \mathbb{C}$ und $\vec{x} \in \mathbb{C}^n \backslash \{0\}$ mit $A\vec{x} = \lambda\vec{x}$.

λ heißt EW (Eigenwert) , $\vec{x}$ zugehöriger EV (Eigenvektor) von A.

Anwendungsbeispiel

Gegeben: Lineares DGL-System mit konstanten Koeffizienten $\vec{y}\,' = A\vec{y} + \vec{b}(x)$.

Gesucht: Lösung des homogenen Systems $\vec{y}\,' = A\vec{y}$.

Ansatz: $\vec{y}(x) = \vec{c}\,\mathrm{e}^{\lambda x} \Rightarrow \lambda\vec{c}\,\mathrm{e}^{\lambda x} = A\vec{c}\,\mathrm{e}^{\lambda x} \Rightarrow A\vec{c} = \lambda\vec{c}$ (da $\mathrm{e}^{\lambda x} \neq 0$)
$\Rightarrow$ λ ist EW , $\vec{c}$ zugehöriger EV von A.

Im folgenden benutzen wir viele Eigenschaften, die wir bereits in Satz 1.4, S.9, gezeigt haben.

Für sehr kleine Matrizen $(n \leq 3)$ kann man die EW einer Matrix als Nullstellen des *charakteristischen Polynoms* $\det(A - \lambda E)$ bestimmen. Bei größeren Matrizen ist der Rechenaufwand zur Berechnung der Determinante $(\hat{=} n!)$ zu groß, und die Berechnung der Nullstellen des charakteristischen Polynoms *numerisch sehr unstabil*, d.h.: bei kleiner Ungenauigkeit der Koeffizienten des Polynoms treten große Fehler bei der Berechnung der Nullstellen auf. Deshalb müssen in der Praxis andere numerische Verfahren zur Berechnung der EW einer Matrix benutzt werden. Wir werden drei wichtige Verfahren behandeln: 1. von Mises-Verfahren , 2. Wielandt-Verfahren , 3. QR-Algorithmus , wobei das Wielandt-Verfahren eine Variante des von Mises-Verfahrens ist.

1. von Mises-Verfahren

Ziel: Berechnung des *betraglich größten EW* von A und eines zugehörigen (normierten) EV.

Voraussetzung: A *diagonalisierbar* , d.h: es existiert C (regulär) mit $C^{-1}AC = D$, (D Diagonalmatrix).

Seien $\lambda_1, \lambda_2, \ldots, \lambda_n$ die EW von A mit $|\lambda_1| > |\lambda_2| \geq \ldots \geq |\lambda_n|$ und
$\vec{s}_1$, $\vec{s}_2$, $\ldots$, $\vec{s}_n$ zugehörige EV,

$\vec{x}_0$ mit $\vec{x}_0 = \sum_{i=1}^{n} c_i \vec{s}_i$, $c_1 \neq 0$, sei Startvektor. Dann gilt

$$A^r \vec{x}_0 = \sum_{i=1}^{n} c_i A^r \vec{s}_i = \sum_{i=1}^{n} c_i \lambda_i^r \vec{s}_i = \lambda_1^r \Big(c_1 \vec{s}_1 + c_2 \Big(\frac{\lambda_2}{\lambda_1}\Big)^r \vec{s}_2 + \ldots + c_n \Big(\frac{\lambda_n}{\lambda_1}\Big)^r \vec{s}_n \Big) .$$

Es gilt $\Big(\frac{\lambda_i}{\lambda_1}\Big)^r \to 0$ für $r \to \infty$, da $|\frac{\lambda_i}{\lambda_1}| < 1 \quad \forall i \geq 2$.

Also gilt für große r: $A^r \vec{x}_0 \approx K \vec{s}_1$.

Da $K\vec{s}_1$ EV zu λ_1 $\Rightarrow$ $A^r \vec{x}_0$ konvergiert für $r \to \infty$ gegen einen EV zum betraglich größten EW λ_1 von A.

Also müssen wir die Matrix A mehrmals auf den Startvektor $\vec{x}_0$ anwenden, um näherungsweise einen EV zu λ_1 zu erhalten.

Nachteil: $\|K\vec{s}_1\|$ wird groß, falls $|\lambda_1| > 1$. Deshalb muß nach jedem Schritt der neue Iterationsvektor *normiert* werden. Das Verfahren ist also sehr einfach durchzuführen: Auf den Startvektor $\vec{x}_0$ wird sukzessive die Matrix A angewendet, und nach jedem Schritt wird der neue Vektor normiert. Wir benutzen die $\|\ldots\|_\infty$.

von Mises-Verfahren

Wähle Startvektor $\vec{x}_0$ (z.B.: $(\vec{x}_0)_i = 1 \quad \forall i = 1, \ldots, n$).
Bestimme Index i mit $|(\vec{x}_0)_i| = \max\limits_{1 \leq j \leq n} |(\vec{x}_0)_j|$.
Berechne für $r = 0, 1, \ldots$

$\vec{u}_{r+1} = A\vec{x}_r$
bestimme Index k mit $|(\vec{u}_{r+1})_k| = \max\limits_{1 \leq j \leq n} |(\vec{u}_{r+1})_j|$

$$\mu_{r+1} = \frac{(\vec{u}_{r+1})_i}{(\vec{x}_r)_i} \quad , \quad \vec{x}_{r+1} = \frac{\vec{u}_{r+1}}{|(\vec{u}_{r+1})_k|}$$

$i = k$

bis $|\mu_{r+1} - \mu_r| < \epsilon$ und $\||\vec{x}_{r+1}| - |\vec{x}_r|\| < \delta$ (hierbei sei $|\vec{x}| = (|x_1|, \ldots, |x_n|)^T$).

Satz 3.1 : *Konvergenz des von Mises-Verfahrens*

Sei A diagonalisierbar; für die EW λ_i von A gelte $|\lambda_1| > |\lambda_2| \geq \ldots \geq |\lambda_n|$, $\vec{s}_i$ seien zugehörige EV zu λ_i. Für den Startvektor $\vec{x}_0$ gelte $\vec{x}_0 = \sum_{i=1}^{n} c_i \vec{s}_i$ mit $c_1 \neq 0$.

Dann konvergiert das von Mises-Verfahren für $r \to \infty$, und zwar

$\mu_{r+1} \to \lambda_1$ (betraglich größter EW von A)
$\vec{x}_r \to \vec{s}_1$ (zugehöriger EV zu λ_1)

Beweis :

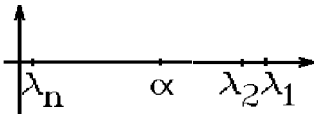
Aus obiger Herleitung folgt, daß $\vec{x}_r \to \vec{s}_1$. Da $(\mu_{r+1}\vec{x}_r)_i = (\vec{u}_{r+1})_i = (A\vec{x}_r)_i$ und $\vec{x}_r \to \vec{s}_1$ und damit $A\vec{x}_r \to A\vec{s}_1 \;\Rightarrow\; (\vec{\mu}_{r+1}\vec{x}_r)_i \to (A\vec{s}_1)_i = (\lambda_1\vec{s}_1)_i \;\Rightarrow\; \mu_{r+1} \to \lambda_1$.

Bemerkung : Ist eine der Voraussetzungen des Satzes 3.1 nicht erfüllt, so kann das von Mises-Verfahren divergieren.

Konvergenzverbesserung, Spektralverschiebung

Die Konvergenz des von Mises-Verfahrens hängt wesentlich ab von den Quotienten $\dfrac{|\lambda_i|}{|\lambda_1|}$, $i \geq 2$. Je kleiner diese Quotienten sind, desto besser konvergiert das von Mises-Verfahren. Durch Verschiebung des Spektrums (Menge der EW) kann man versuchen, diese Quotienten zu verkleinern:

Betrachtet man anstelle der Matrix A die Matrix $B = A - \alpha E$, so hat B die EW $\mu_i = \lambda_i - \alpha$. Wendet man das von Mises-Verfahren auf die Martix B an, so erhält man die Quotienten $\dfrac{|\mu_i|}{|\mu_1|} = \dfrac{|\lambda_i - \alpha|}{|\lambda_1 - \alpha|}$.
Ziel ist es, α so zu wählen, daß $\dfrac{|\lambda_i - \alpha|}{|\lambda_1 - \alpha|} < \dfrac{|\lambda_i|}{|\lambda_1|}$ gilt.

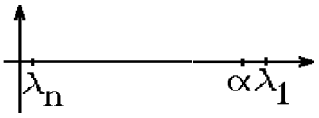


Das von Mises-Verfahren, angewendet auf die Martix B, konvergiert dann (unter den entsprechenden Voraussetzungen) gegen den betraglich größten EW $\mu_k = \lambda_k - \alpha$ und einen zugehörigen EV. Man kann mit dieser Spektralverschiebung auch andere (als den betraglich größten) EW von A berechnen.

Beispiel

Es gelte $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n > 0$.

Sei λ_1 schon berechnet. Wähle $\alpha \approx \lambda_1$.
Dann erhält man den kleinsten EW λ_n von A, denn $\mu_n = \lambda_n - \alpha$ ist dann der betraglich größte EW der Matrix $B = A - \alpha E$.



Beispiele

1. $A = \begin{pmatrix} 1 & 2 & 4 & 8 \\ 2 & 4 & 8 & 1 \\ 4 & 8 & 1 & 2 \\ 8 & 1 & 2 & 4 \end{pmatrix}$

EW: $\lambda_1 = 15$, $\lambda_{2,3} = \pm 3\sqrt{5} \approx \pm 6.7082039$, $\lambda_4 = -5$.

Ergebnis:

$\lambda_1 = 15$ nach 21 Iterationen, Verschiebung $\alpha = 0$,
$\lambda_4 = -5$ nach 3 Iterationen, Verschiebung $\alpha = 15$.

2. $A = \begin{pmatrix} 1 & -2 & -1 \\ -4 & -7 & 7 \\ -2 & -8 & 5 \end{pmatrix}$

EW: $\lambda_{1,2} = \pm 3i$, $\lambda_3 = -1$. Es gilt nicht: $|\lambda_1| > |\lambda_2|$.

Ergebnis:

Keine Konvergenz bei Verschiebung $\alpha = 0$,
$\lambda_2 = -3i$ nach 42 Iterationen, Verschiebung $\alpha = 0.5 + 2.5i$.

3. $A = \begin{pmatrix} 5 & 4 & 2 \\ 4 & 5 & 2 \\ 2 & 2 & 2 \end{pmatrix}$

EW: $\lambda_1 = 10$, $\lambda_{2,3} = 1$.

Ergebnis:

$\lambda_1 = 10$ nach 10 Iterationen, Verschiebung $\alpha = 0$,
$\lambda_{2,3} = 1$ nach 3 Iterationen, Verschiebung $\alpha = 10$.

4. *Hilbertmatrix* (vgl. S.14)

$n = 4$: $\lambda_1 = 1.50021$ nach 12 Iterationen, Verschiebung $\alpha = 0$.
Keine Konvergenz bei Verschiebung $\alpha = 1.5$.

$n = 8$: $\lambda_1 = 1.69594$ nach 15 Iterationen, Verschiebung $\alpha = 0$.
Keine Konvergenz bei Verschiebung $\alpha = 1.7$.

Wenn man den betraglich kleinsten EW von A sucht, ist es besser, das folgende *Wielandt-Verfahren* zu benutzen, als eine Spektralverschiebung beim *von Mises-Verfahren* direkt duchzuführen.

2. Wielandt-Verfahren

Ziel: Berechnung des *am nächsten bei einer vorgegebenen Zahl α gelegenen EW* von A und eines zugehörigen (normierten) EV.

Man wendet das von Mises-Verfahren nicht auf die Matrix A, sondern auf die Matrix $B = (A - \alpha E)^{-1}$ an. Die Matrix $B = (A - \alpha E)^{-1}$ hat die EW $\dfrac{1}{\lambda_i - \alpha}$ mit zugehörigen EV $\vec{s}_i$, wenn λ_i die EW von A mit zugehörigen EV $\vec{s}_i$ sind. Der betraglich größte EW von B, also $\max \dfrac{1}{|\lambda_i - \alpha|} = \dfrac{1}{\min |\lambda_i - \alpha|}$, liefert dann den am nächsten bei α gelegenen EW λ_i von A. Gleichzeitig erhält man einen zugehörigen EV $\vec{s}_i$.
Man benutzt beim von Mises-Verfahren nicht die inverse Matrix $B = (A - \alpha E)^{-1}$, sondern löst jeweils das lineare GLS $(A - \alpha E)\vec{u}_{r+1} = \vec{x}_r$.

Wielandt-Verfahren α vorgeben

Wähle Startvektor $\vec{x}_0$ (z.B.: $(\vec{x}_0)_i = 1 \ \ \forall i = 1, \ldots, n$).
Berechne für $r = 0, 1, \ldots$

$$(A - \alpha E)\vec{u}_{r+1} = \vec{x}_r \qquad \text{lineares GLS lösen}$$
$$\text{bestimme Index } i \text{ mit } |(\vec{u}_{r+1})_i| = \max_{1 \le j \le n} |(\vec{u}_{r+1})_j|$$
$$\mu_{r+1} = \alpha + \frac{(\vec{x}_r)_i}{(\vec{u}_{r+1})_i} \quad , \quad \vec{x}_{r+1} = \frac{\vec{u}_{r+1}}{|(\vec{u}_{r+1})_i|}$$

bis $|\mu_{r+1} - \mu_r| < \epsilon$ und $\| |\vec{x}_{r+1}| - |\vec{x}_r| \| < \delta$ (hierbei sei $|\vec{x}| = (|x_1|, \ldots, |x_n|)^T$).

Unter den entsprechenden Voraussetzungen (analog den Voraussetzungen beim von Mises-Verfahren) konvergiert

$\mu_r \to \lambda_i$ (am nächsten bei α gelegener EW von A)
$\vec{x}_r \to \vec{s}_i$ (zugehöriger EV).

Beispiele

1. $$A = \begin{pmatrix} 1 & 2 & 4 & 8 \\ 2 & 4 & 8 & 1 \\ 4 & 8 & 1 & 2 \\ 8 & 1 & 2 & 4 \end{pmatrix}$$

EW: $\lambda_1 = 15$, $\lambda_{2,3} = \pm 3\sqrt{5} \approx \pm 6.7082039$, $\lambda_4 = -5$.

Ergebnis:

$\alpha = -4 \ \Rightarrow \ \lambda_4 = -5$ nach 26 Iterationen,
$\alpha = 6 \ \Rightarrow \ \lambda_2 = 6.7082$ nach 24 Iterationen,
$\alpha = 6.7 \ \Rightarrow \ \lambda_2 = 6.7082$ nach 10 Iterationen.

2. *Hilbertmatrix* (vgl. S.14)

$n = 4$: $\alpha = 0 \ \Rightarrow \ \lambda_1 = 0.0000967$ nach 8 Iterationen (betraglich kleinster EW),

$n = 8$: $\alpha = 0 \ \Rightarrow \ \lambda_1 = 1.111 \cdot 10^{-10}$ nach 8 Iterationen (betr. kleinster EW).

Bemerkung : *Berechnung eines zum EW λ gehörenden EV $\vec{s}$*

Das Wielandt-Verfahren kann auch dazu benutzt werden, zu einem EW λ einen zugehörigen EV $\vec{s}$ zu berechnen. In diesem Fall wählt man $\alpha \approx \lambda$ als Verschiebung.

Bemerkung : *Berechnung der Konditionszahl $k(A)$ einer Matrix A* (vgl. S.12)

Beim Wielandt-Verfahren erhält man mit der Verschiebung $\alpha = 0$ den betraglich kleinsten EW. Das von Mises-Verfahren liefert den betraglich größten EW. Zusammen erhält man damit die Konditionszahl $k(A)$ einer Matrix A, denn es gilt bzgl. der $\| \ldots \|_2$:

$k(A) = \sqrt{\frac{\mu_1}{\mu_n}}$ mit μ_1 ist der größte EW von $A^T A$, μ_n der kleinste EW von $A^T A$.
Im Spezialfall einer symmetrischen Matrix A gilt:
$k(A) = \frac{|\lambda_1|}{|\lambda_n|}$ mit λ_1 ist der betraglich größte EW von A , λ_n der betraglich kleinste EW von A.

Das Wielandt- und von Mises-Verfahren berechnen jeweils nur *einen* bestimmten EW (mit zugehörigem EV). Sollen *alle* EW einer Matrix A berechnet werden, so muß ein anderes Verfahren benutzt werden. Das Verfahren, das für die meisten Matrizen am besten geeignet ist, ist der QR-Algorithmus.

3. QR-Algorithmus

Ziel: Berechnung aller EW einer Matrix A.

Die zugehörigen EV können dann mit Hilfe des Wielandt-Verfahrens berechnet werden.

1. Schritt *Orthogonale Transformation auf obere Hessenbergform*

Wir bestimmen eine orthogonale Matrix U so, daß

$$U^T AU = \begin{pmatrix} * & * & \dots & \dots & * \\ * & * & \dots & \dots & * \\ 0 & \ddots & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & * \\ 0 & \dots & 0 & * & * \end{pmatrix} .$$

Eine Matrix, die die Form der Matrix auf der rechten Seite annimmt, heißt *obere Hessenbergform*, d.h.: alle Elemente unterhalb der ersten Nebendiagonalen sind $= 0$.

Die Transformation auf Hessenbergform wird durchgeführt, damit der folgende *QR-Algorithmus* schneller ist.

Die orthogonale Matrix U kann als Produkt von orthogonalen *Drehmatrizen* U_{ij} , $(i = 3, \dots, n$, $j = 1, \dots, n-2)$, konstruiert werden. Dabei soll die orthogonale Transformation $U_{ij}^T \tilde{A} U_{ij}$ das Element a_{ij} zu Null machen. Vorher entstandene Nullen dürfen dabei nicht verändert werden. Die gesamte orthogonale Matrix U ist dann das Produkt dieser orthogonalen Drehmatrizen:
$U^T AU = (U_{31} \dots U_{n,n-2})^T A (U_{31} \dots U_{n,n-2}) = U_{n,n-2}^T \dots (U_{31}^T A U_{31}) \dots U_{n,n-2}$.

Konstruktion der Drehmatrix $U_{i,j}$

In allen Zeilen der Matrix U_{ij} stehen Einheitsvektoren bis auf die Zeile $(j+1)$ und i. An der Stelle $(j+1, j+1)$ und (i, i) steht $c = \cos\alpha$, an der Stelle $(j+1, i)$ steht $s = \sin\alpha$ und an der Stelle $(i, j+1)$: $-s = -\sin\alpha$.

$$U_{ij} = \begin{pmatrix} 1 & & & & & & & \\ & \ddots & & & & & & \\ & & 1 & & & & & \\ & & & c & & s & & \\ & & & & 1 & & & \\ & & & & & \ddots & & \\ & & & & & & 1 & \\ & & & -s & & & & c \\ & & & & & & & & 1 \\ & & & & & & & & & \ddots \\ & & & & & & & & & & 1 \end{pmatrix} \begin{matrix} \\ \\ \\ \leftarrow \quad (j+1) \\ \\ \\ \\ \leftarrow \quad i \\ \\ \\ \end{matrix}$$

$$\begin{matrix} & & \uparrow & & \uparrow & & \\ & & (j+1) & & i & & \end{matrix}$$

In der Matrix A seien bereits Nullen erzeugt in den Spalten von 1 bis $(j-1)$. In der $j-$ten Spalte seien Nullen erzeugt bis zum Element $a_{i-1,j}$, also

$$\tilde{A} = \begin{pmatrix} a_{11} & \dots & & & & & \\ a_{21} & & & & & & \\ 0 & \ddots & \ddots & & & & \\ 0 & & a_{j+1,j} & a_{j+1,j+1} & \dots & a_{j+1,i} & \\ & & 0 & & & & \\ \vdots & & \vdots & \vdots & \ddots & \vdots & \\ & & 0 & & & & \\ & & a_{ij} & a_{i,j+1} & \dots & a_{ii} & \\ & & & & & & \\ \vdots & & \vdots & \vdots & & \vdots & \ddots \\ 0 & & & & & & \end{pmatrix}$$

Bei der Multiplikation von links mit U_{ij}^T treten nur Veränderungen in den Zeilen $(j+1)$ und i der Matrix $\tilde{A}$ auf:

$$\boxed{\begin{aligned} a'_{j+1,k} &= c * a_{j+1,k} - s * a_{ik} \quad , \quad k = j, \dots, n \\ a'_{ik} &= s * a_{j+1,k} + c * a_{ik} \quad , \quad k = j, \dots, n \end{aligned}}$$

Bei der Multiplikation von rechts mit U_{ij} treten nur Veränderungen in den Spalten $(j+1)$ und i auf:

$$\boxed{\begin{aligned} a''_{k,j+1} &= c * a'_{k,j+1} - s * a'_{ki} \quad , \quad k = 1, \dots, n \\ a''_{ki} &= s * a'_{k,j+1} + c * a'_{ki} \quad , \quad k = 1, \dots, n \end{aligned}}$$

Damit das Element $a'_{ij} = 0$ wird, werden $c = \cos\alpha$ und $s = \sin\alpha$ folgendermaßen gewählt:

Wahl von c und s

Falls $a_{j+1,j} = 0 \quad \Rightarrow \quad c = 0 \ , \ s = 1$
sonst $\quad w = sign(a_{j+1,j})\sqrt{a_{j+1,j}^2 + a_{ij}^2} \quad , \quad c = \dfrac{a_{j+1,j}}{w} \quad , \quad s = \dfrac{-a_{ij}}{w}$

Dann gilt: $c^2 + s^2 = 1$ und $a'_{ij} = s * a_{j+1,j} + c * a_{ij} = 0$ und
$a'_{j+1,j} = c * a_{j+1,j} - s * a_{ij} = \dfrac{a_{j+1,j}^2}{w} + \dfrac{a_{ij}^2}{w} = \dfrac{w^2}{w} = w$.

2. Schritt *Iterationsverfahren "QR−Algorithmus"*

Wir setzen nun voraus, daß die Matrix A *obere Hessenbergform* besitzt.

QR-Algorithmus

$A_0 = A$, berechne für $m = 0, 1, \ldots$
$A_m = Q_m R_m \qquad (QR - Zerlegung)$
$A_{m+1} = R_m Q_m$

Hierbei ist Q_m eine *orthogonale* Matrix , R_m eine *obere Dreiecksmatrix*.

Da Q_m orthogonal $\Rightarrow$ $Q_m^{-1} = Q_m^T$ $\Rightarrow$ (aus $A_m = Q_m R_m$) $R_m = Q_m^T A_m$.
Setzen wir dies in die zweite Gleichung $A_{m+1} = R_m Q_m$ ein, so erhalten wir
$A_{m+1} = Q_m^T A_m Q_m$ $\Rightarrow$ A_{m+1} ist orthogonal ähnlich zu A_m.
Analog ist A_m orthogonal ähnlich zu A_{m-1} , A_{m-1} orthogonal ähnlich zu A_{m-2} , ... , A_1 orthogonal ähnlich zu $A_0 = A$ $\Rightarrow$
A_m ist orthogonal ähnlich zu A für alle $m = 1, 2, \ldots$.
Also verändern sich beim QR−Algorithmus die EW von A nicht (die EV werden beim QR-Verfahren verändert).

Ziel des QR−Algorithmus: $A_m \to D$ für $m \to \infty$ (D "fast Dreiecksform"),
d.h.: $a_{kk}^{(m)} \to \lambda_k$, $a_{k,k-1}^{(m)} \to 0$, falls λ_k reeller EW von A ,
oder $\begin{pmatrix} a_{kk}^{(m)} & a_{k,k+1}^{(m)} \\ a_{k+1,k}^{(m)} & a_{k+1,k+1}^{(m)} \end{pmatrix} \to B$ mit B hat die komplexen EW λ_k, λ_{k+1} von A
mit $\lambda_{k+1} = \overline{\lambda}_k$.

Berechnung der QR−Zerlegung

Gesucht: orthogonale Matrix Q mit $A = QR$, R obere Dreiecksmatrix.
Da Q orthogonal, also $Q^{-1} = Q^T$ $\Rightarrow$ $Q^T A = R$.
Die Matrix Q^T setzt sich wieder aus orthogonalen Drehmatrizen P_{ij} zusammen:

$Q^T = P_{n,n-1} P_{n-1,n-2} \dots P_{21}$.

Die Matrix $P_{k+1,k}$ muß das Element $a_{k+1,k}$ zu Null machen. Dabei dürfen die vorher erzeugten Nullen nicht verändert werden.

Konstruktion von $P_{k+1,k}$

$P_{k+1,k}\tilde{A} =$

$$\left(\begin{array}{c|c|c} 1 & & \\ \ddots & & \\ \hline & \begin{matrix} c & -s \\ s & c \end{matrix} & \\ \hline & & \begin{matrix} 1 & \\ & \ddots \end{matrix} \end{array}\right) \left(\begin{array}{c|c|c} a_{11} & & \\ \ddots & & \\ \hline & \begin{matrix} a_{kk} & a_{k,k+1} \\ a_{k+1,k} & a_{k+1,k+1} \end{matrix} & \\ \hline & & \ddots \end{array}\right)$$

$$= \left(\begin{array}{c|c|c} a_{11} & & \\ \ddots & & \\ \hline & \begin{matrix} a'_{kk} & a'_{k,k+1} \\ 0 & a'_{k+1,k+1} \end{matrix} & \\ \hline & & \ddots \end{array}\right)$$

Veränderungen treten nur in den Zeilen k und $(k+1)$ auf (ab Spaltenindex k , da vorher nur Nullen in der Matrix $\tilde{A}$ vorkommen), also

$$\boxed{\begin{aligned} a'_{kj} &= c * a_{kj} - s * a_{k+1,j} \quad , \quad j = k, \dots, n \\ a'_{k+1,j} &= s * a_{kj} + c * a_{k+1,j} \quad , \quad j = k, \dots, n \end{aligned}}$$

c und s müssen so gewählt werden, daß $a'_{k+1,k} = 0$ wird:

Wahl von c und s

$$\boxed{\begin{aligned} &\text{Falls} \quad a_{kk} = 0 \quad \Rightarrow \quad w = |a_{k+1,k}| \quad , \quad c = 0 \ , \ s = -sign(a_{k+1,k}) \\ &\text{sonst} \quad w = \sqrt{a_{kk}^2 + a_{k+1,k}^2} \quad , \quad c = \frac{a_{kk}}{w} \quad , \quad s = \frac{-a_{k+1,k}}{w} \end{aligned}}$$

Dann gilt: $c^2 + s^2 = 1$ und $a'_{k+1,k} = s * a_{k,k} + c * a_{k+1,k} = 0$ und

$a'_{kk} = c * a_{kk} - s * a_{k+1,k} = \dfrac{a_{kk}^2}{w} + \dfrac{a_{k+1,k}^2}{w} = \dfrac{w^2}{w} = w$.

Da $A_{neu} = Q^T A_{alt} Q$, gilt mit $Q^T = P_{n,n-1} \dots P_{21}$

$A_{neu} = P_{n,n-1} \dots P_{21} A_{alt} (P_{n,n-1} \dots P_{21})^T = P_{n,n-1} \dots P_{21} A_{alt} P_{21}^T \dots P_{n,n-1}^T \quad \Rightarrow$

$A_{neu} = P_{n,n-1} \dots (P_{32} P_{21} A_{alt}) P_{21}^T P_{32}^T \dots P_{n,n-1}^T$.

Nach zwei Multiplikationen von links kann eine Multiplikation von rechts ausgeführt werden, da die Elemente, die sich hierbei verändern, bei den weiteren Multiplikationen

von links nicht mehr verändert werden. Man muß also immer nur zwei cos − und sin −Werte abspeichern.

Bei der Multiplikation von rechts mit $P^T_{k,k-1}$ erhält man folgende Veränderungen:

$$\left(\begin{array}{c|c|c} a_{11} & & \\ \ddots & & \\ \hline & a'_{k-1,k-1} \quad a'_{k-1,k} & \\ & 0 \qquad\quad a'_{kk} & \\ \hline & & \\ & & \ddots \end{array}\right) \left(\begin{array}{c|c|c} 1 & & \\ \ddots & & \\ \hline & c \quad s & \\ & -s \quad c & \\ \hline & & 1 \\ & & \ddots \end{array}\right)$$

Veränderungen treten nur in den Spalten $(k-1)$ und k auf.

$$\begin{aligned} a''_{j,k-1} &= c * a'_{j,k-1} - s * a'_{jk} \quad , \quad j = 1, \ldots, k \\ a''_{jk} &= s * a'_{j,k-1} + c * a'_{jk} \quad , \quad j = 1, \ldots, k \end{aligned}$$

Konvergenz, Konvergenzverbesserungen

Gilt für die EW von A: $|\lambda_1| \geq |\lambda_2| \geq \ldots \geq |\lambda_n|$, so hängt die Konvergenz von $a^{(m)}_{k+1,k} \to 0$ wesentlich ab vom Quotienten $\frac{|\lambda_{k+1}|}{|\lambda_k|}$ (vgl. auch Konvergenzbeweis für von Mises-Verfahren). Um *Konvergenzverbesserung* zu erhalten, führen wir eine *Spektralverschiebung* α so durch, daß
$\frac{|\lambda_{k+1} - \alpha|}{|\lambda_k - \alpha|} < \frac{|\lambda_{k+1}|}{|\lambda_k|}$ gilt. α sollte also möglichst nahe bei λ_{k+1} sein. Dann erhalten wir anstelle der Matrix A_m die Matrix $(A_m - \alpha E)$, also
$(A_m - \alpha E) = Q_m R_m \quad , \quad A_{m+1} = R_m Q_m + \alpha E$.
Dann ist wieder A_{m+1} orthogonal ähnlich zu A_m, denn:
Aus $(A_m - \alpha E) = Q_m R_m \quad \Rightarrow \quad R_m = Q^T_m (A_m - \alpha E) \quad \Rightarrow$
$A_{m+1} = R_m Q_m + \alpha E = Q^T_m (A_m - \alpha E) Q_m + \alpha E = Q^T_m A_m Q_m - \alpha E + \alpha E = Q^T_m A_m Q_m$.

Wahl von α:

Berechne die EW der Matrix $\begin{pmatrix} a^{(m)}_{n-1,n-1} & a^{(m)}_{n-1,n} \\ a^{(m)}_{n,n-1} & a^{(m)}_{n,n} \end{pmatrix}$

1.Fall: *Einfacher QR - Schritt*

Hat diese Matrix zwei *reelle* EW $\Rightarrow$ wähle $\alpha =$ einen der beiden EW.
In diesem Fall gilt: $a^{(m)}_{n,n-1} \to 0$ und $a^{(m)}_{n,n} \to \lambda_n$ (EW von A) für $m \to \infty$.

2.Fall: *Doppel - QR - Schritt*

Hat obige Matrix zwei *komplexe* EW μ_1 und μ_2 mit Imaginärteil $\neq 0$ und
$\mu_2 = \overline{\mu}_1 \quad \Rightarrow$

2 Schritte ausführen: 1.Schritt mit $\alpha = \mu_1$ und 2.Schritt mit $\alpha = \mu_2$.
Beide Schritte können so zusammengefaßt werden, daß die Rechnung *reell* bleibt.

In diesem Fall gilt: $a^{(m)}_{n-1,n-2} \to 0$ und die EW der unteren Teilmatrix
$\begin{pmatrix} a^{(m)}_{n-1,n-1} & a^{(m)}_{n-1,n} \\ a^{(m)}_{n,n-1} & a^{(m)}_{n,n} \end{pmatrix}$ gehen gegen die EW λ_{n-1} , λ_n von A.

Zerlegung der Matrix A_m

Sind die Elemente $a^{(m)}_{n,n-1}$ (im 1.Fall) bzw. $a^{(m)}_{n-1,n-2}$ (im 2.Fall) betraglich klein geworden, so kann man die Matrix A_m *zerlegen*:

$$\left(\begin{array}{cccc|c} * & * & \dots & & * \\ * & & & & \\ & \ddots & \ddots & & \vdots \\ & & * & * & * \\ \hline & & & \epsilon & a^{(m)}_{nn} \end{array}\right) \quad \text{oder} \quad \left(\begin{array}{cccc|cc} * & * & \dots & & * & \\ * & & & & & \\ & \ddots & \ddots & & \vdots & \\ & & * & * & * & \\ \hline & & & \epsilon & a^{(m)}_{n-1,n-1} & a^{(m)}_{n-1,n} \\ & & & & a^{(m)}_{n,n-1} & a^{(m)}_{n,n} \end{array}\right).$$

(1.Fall) (2.Fall)

Ist ϵ betraglich sehr klein, so liefert im 1.Fall das Element $a^{(m)}_{nn}$ einen EW λ_n von A.
Im 2.Fall liefert die untere $(2,2)-$Matrix zwei EW λ_{n-1}, λ_n von A.

Anschließend führt man den $QR-$Algorithmus mit der kleineren
oberen $(n-1, n-1)-$Matrix (im 1.Fall) bzw. $(n-2, n-2)-$Matrix (im 2.Fall) fort.
Die Matrix A wird solange zerlegt, bis man nur noch eine $(2,2)-$Matrix (bzw. $(1,1)-$Matrix) hat.
Die EW dieser $(2,2)-$Matrix können dann direkt berechnet werden.

Auf diesem Weg erhält man dann alle EW $\lambda_1, \dots, \lambda_n$.

QR - Algorithmus

Wir geben für die einzelnen Schritte des QR-Algorithmus jeweils *Prozeduren* an:

Prozedur 1.: *Transformation auf obere Hessenbergform*

Prozedur 2.: *Berechnung der EW der unteren $(2,2)-$Matrix*

Prozedur 3.: *Zerlegung der Matrix*

Prozedur 4.: *Einfacher QR - Schritt*

Prozedur 5.: *Doppel - QR - Schritt*

QR - Algorithmus

Führe Prozedur 1. aus (Transformation auf Hessenbergform)

Berechne bis $n = 2$ oder $n = 1$:
Führe Prozedur 2. aus (EW der unteren $(2,2)-$Matrix berechnen)
falls $d \geq 0 \Rightarrow$ Prozedur 4. ausführen (einfacher QR-Schritt)
sonst Prozedur 5. ausführen (Doppel-QR-Schritt)
Führe Prozedur 3. aus (Zerlegung überprüfen)
falls $|a^{(m)}_{n,n-1}|$ oder $|a^{(m)}_{n-1,n-2}|$ klein genug $\Rightarrow$ $n = n-1$ bzw. $n = n-2$ setzen
Berechne mit Prozedur 2. die EW der restlichen $(2,2)-$Matrix
(oder falls $n = 1 \Rightarrow$ das letzte Element $a^{(m)}_{11}$ ist EW).

Prozedur 1.: Transformation auf obere Hessenbergform

für $j = 1$ bis $(n-2)$
für $i = (j+2)$ bis n
falls $a_{ij} \neq 0$
dann
falls $|a_{j+1,j}| \geq \delta * |a_{ij}|$
dann $w = sign(a_{j+1,j})\sqrt{a^2_{j+1,j} + a^2_{ij}}$, $c = \frac{a_{j+1,j}}{w}$, $s = -\frac{a_{ij}}{w}$
sonst $w = -a_{ij}$, $c = 0$, $s = 1$
$a_{j+1,j} = w$, $a_{ij} = 0$
für $k = (j+1)$ bis n
$h = c * a_{j+1,k} - s * a_{ik}$
$a_{ik} = s * a_{j+1,k} + c * a_{ik}$
$a_{j+1,k} = h$
Ende $k-$Schleife
für $k = 1$ bis n
$h = c * a_{k,j+1} - s * a_{ki}$
$a_{ki} = s * a_{k,j+1} + c * a_{ki}$
$a_{k,j+1} = h$
Ende $k-$Schleife
Ende $i-$Schleife
Ende $j-$Schleife

Prozedur 2.: Berechnung der EW der unteren $(2,2)-$Matrix

$d_1 = 0.5 * (a_{n-1,n-1} + a_{nn})$
$d = d_1^2 - (a_{n-1,n-1} * a_{nn} - a_{n,n-1} * a_{n-1,n})$
falls $d \geq 0 \Rightarrow$ $l_1 = d_1 + \sqrt{d}$, $l_2 = d_1 - \sqrt{d}$ (reelle EW)
sonst $l_1 = d_1 + i\sqrt{-d}$, $l_2 = d_1 - i\sqrt{-d}$ (komplexe EW)

Prozedur 3.: Zerlegung der Matrix

falls $|a_{n,n-1}| < \delta * |a_{nn}|$ oder $|a_{n,n-1}| < \epsilon$
$\Rightarrow$ $\lambda_n = a_{nn}$ ausgeben , $n = n - 1$ setzen
falls $|a_{n-1,n-2}| < \delta * |a_{n-1,n-1}|$ oder $|a_{n-1,n-2}| < \epsilon$
$\Rightarrow$ λ_n, λ_{n-1} mit Prozedur 2. berechnen und ausgeben , $n = n - 2$ setzen

Prozedur 4.: Einfacher QR - Schritt

$\sigma = l_1$, falls $|l_2 - a_{nn}| < |l_1 - a_{nn}|$ $\Rightarrow$ $\sigma = l_2$
$a_{11} = a_{11} - \sigma$
für $i = 1$ bis n
falls $i < n$ (sonst weiter bei "falls $i > 1$")
dann
falls $|a_{ii}| < \delta * |a_{i+1,i}|$ oder $|a_{ii}| < \epsilon$
dann $w = |a_{i+1,i}|$, $c = 0$, $s = -sign(a_{i+1,i})$
sonst $w = \sqrt{a_{ii}^2 + a_{i+1,i}^2}$, $c = \frac{a_{ii}}{w}$, $s = -\frac{a_{i+1,i}}{w}$
$a_{ii} = w$, $a_{i+1,i} = 0$, $a_{i+1,i+1} = a_{i+1,i+1} - \sigma$
für $j = (i + 1)$ bis n
$h = c * a_{ij} - s * a_{i+1,j}$
$a_{i+1,j} = s * a_{ij} + c * a_{i+1,j}$
$a_{ij} = h$
Ende $j-$Schleife
falls $i > 1$ (sonst weiter bei "$c_1 = c$, $s_1 = s$")
dann
für $j = 1$ bis i
$h = c_1 * a_{j,i-1} - s_1 * a_{ji}$
$a_{ji} = s_1 * a_{j,i-1} + c_1 * a_{ji}$
$a_{j,i-1} = h$
Ende $j-$Schleife
$a_{i-1,i-1} = a_{i-1,i-1} + \sigma$
$c_1 = c$, $s_1 = s$
Ende $i-$Schleife
$a_{nn} = a_{nn} + \sigma$

Prozedur 5.: Doppel - QR - Schritt

$\sigma = a_{n-1,n-1} + a_{nn}$
$\tau = a_{n-1,n-1} * a_{nn} - a_{n-1,n} * a_{n,n-1}$
$x_1 = (a_{11} - \sigma) * a_{11} + a_{12} * a_{21} + \tau$
$x_2 = a_{21} * (a_{11} + a_{22} - \sigma)$
$x_3 = a_{21} * a_{32}$
für $p = 1$ bis $(n - 1)$

..... (siehe nächste Seite)

für $p = 1$ bis $(n-1)$
  für $i = 2$ bis 3
    falls $|x_1| < \delta * |x_i|$ oder $|x_1| < \epsilon$
      dann $w = -x_i\ ,\ c = 0\ ,\ s = 1$
      sonst $w = \sqrt{x_1^2 + x_i^2}\ ,\ c = \frac{x_1}{w}\ ,\ s = -\frac{x_i}{w}$
    $x_1 = w$
    $q = p + i - 1$
    falls $q \leq n$ (sonst Ende $i-$Schleife)
    dann
      für $k = (q-1)$ bis n
        $h = c * a_{pk} - s * a_{qk}$
        $a_{qk} = s * a_{pk} + c * a_{qk}$
        $a_{pk} = h$
      Ende $k-$Schleife
      falls $i = 3$
      dann
        $h = c * a_{pp} - s * x_2$
        $x_2 = s * a_{pp} + c * x_2$
        $a_{pp} = h$
      für $j = 1$ bis $(p+1)$
        $h = c * a_{jp} - s * a_{jq}$
        $a_{jq} = s * a_{jp} + c * a_{jq}$
        $a_{jp} = h$
      Ende $j-$Schleife
      falls $i = 2$ und $p < (n-1)$
        dann $x_2 = -s * a_{p+2,q}\ ,\ a_{p+2,q} = c * a_{p+2,q}$
      falls $i = 3$
      dann
        $h = c * x_2 - s * a_{qq}$
        $a_{qq} = s * x_2 + c * a_{qq}$
        $x_2 = h$
        falls $p < (n-2)$
          dann $x_3 = -s * a_{p+3,q}\ ,\ a_{p+3,q} = c * a_{p+3,q}$
          sonst $x_3 = 0$
  Ende $i-$Schleife
  falls $p > 1$ dann $a_{p,p-1} = x_1$
  $x_1 = a_{p+1,p}$
Ende $p-$Schleife

Rundungsfehler

Da alle Transformationen mit *orthogonalen* Matrizen durchgeführt werden, ist der QR-Algorithmus *numerisch sehr stabil* (d.h.: die Rundungsfehler werden nicht seht groß), denn eine *orthogonale* Matrix Q hat bzg. der $2-$Norm die *Konditionszahl* $k(Q) = 1$,

denn: Q orthogonal $\Rightarrow \quad Q^T Q = E$,

E hat nur den ($n-$fachen) EW $\mu = 1 \quad \Rightarrow \quad k(Q) = \sqrt{\dfrac{\mu_1}{\mu_n}} = 1.$

Beispiele

1. $$A = \begin{pmatrix} 1 & 2 & 4 & 8 \\ 2 & 4 & 8 & 1 \\ 4 & 8 & 1 & 2 \\ 8 & 1 & 2 & 4 \end{pmatrix}$$

$\lambda_1 = -5$ nach 5 Iterationen
$\lambda_2 = 15$ nach einer weiteren Iteration
$\lambda_{3,4} = \pm 6.7082039 \approx \pm 3\sqrt{5}$.

2. $$A = \begin{pmatrix} 1 & -2 & -1 \\ -4 & -7 & 7 \\ -2 & -8 & 5 \end{pmatrix}$$

$\lambda_1 = -1$ nach 7 Iterationen
$\lambda_{2,3} = \pm 3i$.

3. *Hilbertmatrix*

$n = 4$:

$\lambda_1 = 9.6702 \cdot 10^{-5}$ nach 4 Iterationen
$\lambda_2 = 6.7383 \cdot 10^{-3}$ nach 1 weiteren Iteration
$\lambda_3 = 0.16914 \quad , \quad \lambda_4 = 1.50021$.

$n = 8$:

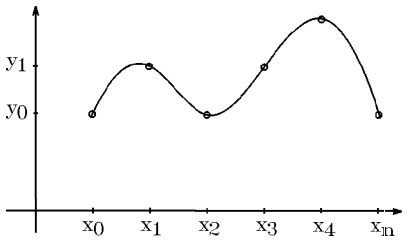
$\lambda_1 = 1.1113 \cdot 10^{-10}$ nach 3 Iterationen
$\lambda_2 = 1.7989 \cdot 10^{-8}$ nach 1 weiteren Iteration
$\lambda_3 = 1.2943 \cdot 10^{-6}$ nach 1 weiteren Iteration
$\lambda_4 = 5.4369 \cdot 10^{-5}$ nach 1 weiteren Iteration
$\lambda_5 = 1.4677 \cdot 10^{-3}$ nach 1 weiteren Iteration
$\lambda_6 = 0.262128$ nach 1 weiteren Iteration
$\lambda_7 = 0.298125 \quad , \quad \lambda_8 = 1.695939$.

Bemerkung : Benötigt man zusätzlich zu den EW auch EV, so kann man das Wielandt-Verfahren benutzen, indem man jeweils bei der Spektralverschiebung $\alpha =$"berechneter EW" wählt. Das Wielandt-Verfahren konvergiert dann sehr schnell.

IV Spline - Interpolation

Gegeben: $(n+1)$ *Stützstellen* $x_0 < x_1 < \ldots < x_n$
und zugehörige *Stützwerte* $y_0\ ,\ y_1\ ,\ \ldots\ ,\ y_n\ .$

Gesucht: Eine "einfache" glatte Funktion s, deren Graph diese Punkte $\begin{pmatrix} x_i \\ y_i \end{pmatrix}\ ,\ i = 0, 1, \ldots, n,$ verbindet, für die also gilt: $s(x_i) = y_i \quad \forall 0 \leq i \leq n.$



Ist die Anzahl der Stützstellen klein, so kann man zur Interpolation das *Lagrange-Interpolationspolynom* benutzen. Die Berechnung dieses Interpolationspolynoms erfolgt mit dem *Newton-Schema* (vgl. Vorlesung "Mathematik für Elektrotechniker II"). Bei großer Anzahl von Stützstellen erhält man aber hierbei ein Polynom großen Grades, das besonders an den Rändern zu starken *Oszillationen* neigt und deshalb für die Anwendung nicht geeignet ist (vgl. Bild auf S.56). Deshalb bedient man sich bei größerer Anzahl von Stützstellen der *kubischen Splines*.

Idee der kubischen Splines

In den Teilintervallen $[x_i, x_{i+1}]$ zwischen den einzelnen Stützstellen sei $s(x)$ ein Polynom $s_i(x)$ vom Grad ≤ 3. An den Endpunkten der Teilintervalle, also an den Stützstellen, sollen die Polynome $s_i(x)$ *glatt* aneinanderstoßen, d.h. bei kubischen Splines : die Übergänge sollen $2-$*mal stetig differenzierbar* sein.
Damit erhalten wir insgesamt eine $2-$*mal stetig differenzierbare* Interpolationsfunktion $s(x)$, die stückweise aus Polynomen vom Grad ≤ 3 besteht.

Konstruktion des kubischen Splines

Im Intervall $[x_i, x_{i+1}]$ mit der Intervalllänge $h_i = x_{i+1} - x_i$ setzen wir folgendes Polynom vom Grad ≤ 3 an:

$$\boxed{s_i(x) = a_i(x - x_i)^3 + b_i(x - x_i)^2 + c_i(x - x_i) + d_i}$$

An den beiden Endpunkten des Intervalls $[x_i, x_{i+1}]$ erhalten wir die folgenden 6 Gleichungen:

(1) $s_i(x_i) = d_i = y_i$
(2) $s_i(x_{i+1}) = a_i h_i^3 + b_i h_i^2 + c_i h_i + d_i = y_{i+1}$
(3) $s_i'(x_i) = c_i = y_i'$
(4) $s_i'(x_{i+1}) = 3a_i h_i^2 + 2b_i h_i + c_i = y_{i+1}'$
(5) $s_i''(x_i) = 2b_i = y_i''$
(6) $s_i''(x_{i+1}) = 6a_i h_i + 2b_i = y_{i+1}''\ .$

Aus den Gleichungen (1), (2), (5), (6) erhalten wir

$$(1) \quad \Rightarrow \quad d_i = y_i \quad , \quad (5) \quad \Rightarrow \quad b_i = \frac{1}{2} y_i'' \quad , \quad (6) \quad a_i = \frac{1}{6h_i}(y_{i+1}'' - y_i'')$$

$$(2) \quad \Rightarrow \quad c_i = \frac{1}{h_i}(y_{i+1} - y_i) - h_i(b_i + a_i h_i) = \frac{1}{h_i}(y_{i+1} - y_i) - h_i\Big(\frac{y_i''}{2} + \frac{y_{i+1}''}{6} - \frac{y_i''}{6}\Big)$$

$$\Rightarrow \quad c_i = \frac{1}{h_i}(y_{i+1} - y_i) - \frac{h_i}{6}(y_{i+1}'' + 2y_i'') \ .$$

Also erhalten wir für $0 \le i \le n-1$

$$\boxed{\begin{aligned} a_i &= \frac{1}{6h_i}(y_{i+1}'' - y_i'') \\ b_i &= \frac{1}{2} y_i'' \\ c_i &= \frac{1}{h_i}(y_{i+1} - y_i) - \frac{h_i}{6}(y_{i+1}'' + 2y_i'') \\ d_i &= y_i \end{aligned}}$$

Damit haben wir die unbekannten Koeffizienten a_i, b_i, c_i, d_i , $0 \le i \le n-1$, durch die bekannten Stützwerte y_i und die noch unbekannten Werte y_i'' (2. Ableitung an den Stellen x_i) ausgedrückt. Können wir diese Werte y_i'' bestimmen, so sind alle Koeffizienten der Teilpolynome $s_i(x)$ bekannt.

Die Werte y_i'' lassen sich mit Hilfe der Gleichungen (3) und (4) berechnen. Es sollen ja die folgenden *Glattheitsbedingungen* gelten:

$s_{i-1}'(x_i) = s_i'(x_i) \quad , \quad (1 \le i \le n-1).$

Aus Gleichung (3) folgt

$$s_i'(x_i) = c_i = \frac{1}{h_i}(y_{i+1} - y_i) - \frac{h_i}{6}(y_{i+1}'' + 2y_i'') \ .$$

Aus Gleichung (4) folgt für $(i-1)$ anstelle von i

$$\begin{aligned} s_{i-1}'(x_i) &= 3a_{i-1}h_{i-1}^2 + 2b_{i-1}h_{i-1} + c_{i-1} \\ &= \frac{h_{i-1}}{2}(y_i'' - y_{i-1}'') + h_{i-1}y_{i-1}'' + \frac{1}{h_{i-1}}(y_i - y_{i-1}) - \frac{h_{i-1}}{6}(y_i'' + 2y_{i-1}'') \\ &= \frac{h_{i-1}}{6}(2y_i'' + y_{i-1}'') + \frac{1}{h_{i-1}}(y_i - y_{i-1}) \ . \end{aligned}$$

Aus $s_{i-1}'(x_i) = s_i'(x_i)$ folgt mit diesen beiden Gleichungen

$$h_{i-1}(2y_i'' + y_{i-1}'') + h_i(2y_i'' + y_{i+1}'') = \frac{6}{h_i}(y_{i+1} - y_i) - \frac{6}{h_{i-1}}(y_i - y_{i-1})$$

$$\boxed{\Rightarrow \quad h_{i-1}y_{i-1}'' + 2(h_{i-1} + h_i)y_i'' + h_i y_{i+1}'' = d_i \quad , \quad (1 \le i \le n-1)}$$

mit $\quad d_i = \dfrac{6}{h_i}(y_{i+1} - y_i) - \dfrac{6}{h_{i-1}}(y_i - y_{i-1}) \ .$

Das sind $(n-1)$ Gleichungen für die $(n+1)$ Unbekannten y_i'' , $(0 \le i \le n)$.

y_0'' und y_n'' sind *frei wählbar*, da hierfür keine Anschlußbedingungen existieren.

Damit erhalten wir in Matrizenschreibweise folgendes lineare GLS

$$\begin{pmatrix} 2(h_0+h_1) & h_1 & & & & \\ h_1 & 2(h_1+h_2) & h_2 & & & \\ & h_2 & \ddots & \ddots & & \\ & & \ddots & \ddots & h_{n-2} & \\ & & & h_{n-2} & 2(h_{n-2}+h_{n-1}) \end{pmatrix} \begin{pmatrix} y_1'' \\ y_2'' \\ \vdots \\ y_{n-2}'' \\ y_{n-1}'' \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_{n-2} \\ b_{n-1} \end{pmatrix}$$

mit

$$b_1 = d_1 - h_0 y_0'' = \frac{6}{h_1}(y_2 - y_1) - \frac{6}{h_0}(y_1 - y_0) - h_0 y_0''$$

$$b_i = d_i = \frac{6}{h_i}(y_{i+1} - y_i) - \frac{6}{h_{i-1}}(y_i - y_{i-1}) \quad , \quad (2 \le i \le n-2)$$

$$b_{n-1} = d_{n-1} - h_{n-1} y_n'' = \frac{6}{h_{n-1}}(y_n - y_{n-1}) - \frac{6}{h_{n-2}}(y_{n-1} - y_{n-2}) - h_{n-1} y_n'' \; .$$

Die Koeffizientenmatrix ist eine symmetrische, diagonaldominante, positiv definte Tridiagonalmatrix (Bandmatrix der Bandbreite 1). Das GLS läßt sich mit Hilfe des Cholesky-Verfahrens für Bandmatrizen lösen.
Als Lösung dieses GLS erhalten wir die Werte y_i'' , $(1 \le i \le n-1)$, wenn man vorher die beiden Werte y_0'' und y_n'' gewählt hat. Aus den Werten y_i'' lassen sich dann die Koeffizienten a_i, b_i, c_i, d_i , $(0 \le i \le n-1)$, der Teilpolynome $s_i(x)$ mit Hilfe der Formeln auf S.52 bestimmen. Damit ist die kubische Spline-Funktion $s(x)$ bekannt.

Spezialfall: *Äquidistante Stützstellen*

Sind alle $h_i = h$ (h = Schrittweite), also $x_i = x_0 + ih$, $0 \le i \le n$, so lautet das lineare GLS (nach Division aller Gleichungen durch h):

$$\begin{pmatrix} 4 & 1 & & & \\ 1 & 4 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & 4 & 1 \\ & & & 1 & 4 \end{pmatrix} \begin{pmatrix} y_1'' \\ y_2'' \\ \vdots \\ y_{n-2}'' \\ y_{n-1}'' \end{pmatrix} = \frac{6}{h^2} \begin{pmatrix} y_2 - 2y_1 + y_0 \\ y_3 - 2y_2 + y_1 \\ \vdots \\ y_{n-1} - 2y_{n-2} + y_{n-3} \\ y_n - 2y_{n-1} + y_{n-2} \end{pmatrix} - \begin{pmatrix} y_0'' \\ 0 \\ \vdots \\ 0 \\ y_n'' \end{pmatrix}$$

Wahl von y_0'' und y_n''

1. Man kann $y_0'' = y_n'' = 0$ wählen. Dann erhält man den *natürlichen Spline*. Diese Wahl ist ungünstig, wenn an den Enden des Gesamtintervalls $[x_0, x_n]$ starke Krümmung vorliegt.

2. Man kann die Krümmung an den Endpunkten in Beziehung zur Krümmung in den benachbarten Punkten setzen, d.h:

$y_0'' = \alpha y_1''$, $y_n'' = \beta y_{n-1}''$ mit $\alpha, \beta \in \mathbb{R}$.

z.B.: $\alpha = \beta = 1$ oder $\alpha = \beta = 0.5$. (Für $\alpha = \beta = 0$ erhält man Wahl 1.)

Diese Wahl verändert im GLS nur die 1. und letzte Zeile:

$$\begin{pmatrix} a_{11} & h_1 & & & \\ h_1 & 2(h_1+h_2) & h_2 & & \\ & h_2 & \ddots & \ddots & \\ & & \ddots & \ddots & h_{n-2} \\ & & & h_{n-2} & a_{n-1,n-1} \end{pmatrix} \begin{pmatrix} y_1'' \\ y_2'' \\ \vdots \\ y_{n-2}'' \\ y_{n-1}'' \end{pmatrix} = \begin{pmatrix} d_1 \\ d_2 \\ \vdots \\ d_{n-2} \\ d_{n-1} \end{pmatrix}$$

mit $\quad a_{11} = 2(h_0+h_1) + \alpha h_0 \quad , \quad a_{n-1,n-1} = 2(h_{n-2}+h_{n-1}) + \beta h_{n-1} \qquad$ und

$$d_i = \frac{6}{h_i}(y_{i+1}-y_i) - \frac{6}{h_{i-1}}(y_i - y_{i-1}) \quad , \quad (1 \le i \le n-1).$$

3. Sind die Werte der 1. Ableitung an den Endpunkten des Intervalls $[x_0, x_n]$ bekannt, so kann man diese Werte y_0' und y_n' vorgeben, also

$s_0'(x_0) = y_0' \quad , \quad s_{n-1}'(x_n) = y_n' \ .$

Dann erhält man im GLS oben und unten jeweils eine neue zusätzliche Gleichung

$$\begin{pmatrix} 2h_0 & h_0 & & & & \\ h_0 & 2(h_0+h_1) & h_1 & & & \\ & \ddots & \ddots & \ddots & & \\ & & h_{n-2} & 2(h_{n-2}+h_{n-1}) & h_{n-1} & \\ & & & h_{n-1} & 2h_{n-1} \end{pmatrix} \begin{pmatrix} y_0'' \\ y_1'' \\ \vdots \\ y_{n-1}'' \\ y_n'' \end{pmatrix} = \begin{pmatrix} b_0 \\ b_1 \\ \vdots \\ b_{n-1} \\ b_n \end{pmatrix}$$

mit

$$b_0 = \frac{6}{h_0}(y_1 - y_0) - 6y_0'$$

$$b_i = d_i = \frac{6}{h_i}(y_{i+1}-y_i) - \frac{6}{h_{i-1}}(y_i - y_{i-1}) \quad , \quad (1 \le i \le n-1)$$

$$b_n = 6y_n' - \frac{6}{h_{n-1}}(y_n - y_{n-1}) \ .$$

Denn:

Aus Gleichung (3) von S.51 folgt

$$s_0'(x_0) = c_0 = \frac{1}{h_0}(y_1 - y_0) - \frac{h_0}{6}(y_1'' + 2y_0'') = y_0'$$

$$\Rightarrow \quad 2h_0 y_0'' + h_0 y_1'' = \frac{6}{h_0}(y_1 - y_0) - 6y_0' \ .$$

Aus Gleichung (4) von S.51 folgt

$$\begin{aligned} s_{n-1}'(x_n) &= 3a_{n-1}h_{n-1}^2 + 2b_{n-1}h_{n-1} + c_{n-1} \\ &= \frac{h_{n-1}}{2}(y_n'' - y_{n-1}'') + h_{n-1}y_{n-1}'' + \frac{1}{h_{n-1}}(y_n - y_{n-1}) - \frac{h_{n-1}}{6}(y_n'' + 2y_{n-1}'') \\ &= \frac{h_{n-1}}{6}(2y_n'' + y_{n-1}'') + \frac{1}{h_{n-1}}(y_n - y_{n-1}) = y_n' \end{aligned}$$

$$\Rightarrow \quad h_{n-1}y_{n-1}'' + 2h_{n-1}y_n'' = 6y_n' - \frac{6}{h_{n-1}}(y_n - y_{n-1}) \ .$$

Algorithmus für den Fall der Wahl: $y_0'' = \alpha y_1''$, $y_n'' = \beta y_{n-1}''$

Belegung der Speicherplätze

Diagonale der Koeffizientenmatrix $\rightarrow$ $a_1, \ldots, a_{n-1}$
Nebendiagonale der Koeff.-Matrix $\rightarrow$ $c_1, \ldots, c_{n-2}$
rechte Seite $\rightarrow$ $b_1, \ldots, b_{n-1}$
$y_i'' \rightarrow b_i$, $(0 \le i \le n)$

Eingabe

x_i , y_i , $(i = 0, 1, \ldots, n)$
α , β

Belegung des GLS

$h_i = x_{i+1} - x_i$, $(i = 0, 1, \ldots, n-1)$
$a_i = 2(h_{i-1} + h_i)$, $(i = 1, 2, \ldots, n-1)$
$a_1 = a_1 + \alpha h_0$, $a_{n-1} = a_{n-1} + \beta h_{n-1}$
$c_i = h_i$, $(i = 1, 2, \ldots, n-2)$
$b_i = 6 * \Big((y_{i+1} - y_i)/h_i - (y_i - y_{i-1})/h_{i-1} \Big)$, $(i = 1, 2, \ldots, n-1)$

Berechnung der Lösung des GLS

LDL^T-Zerlegung

für $i = 2$ bis $(n-1)$
    $h = c_{i-1}/a_{i-1}$, $a_i = a_i - c_{i-1} * h$, $c_{i-1} = h$
Ende $i-$Schleife

Vorwärtseinsetzen

für $i = 2$ bis $(n-1)$
    $b_i = b_i - c_{i-1} * b_{i-1}$
Ende $i-$Schleife
für $i = 1$ bis $(n-1)$
    $b_i = b_i / a_i$
Ende $i-$Schleife

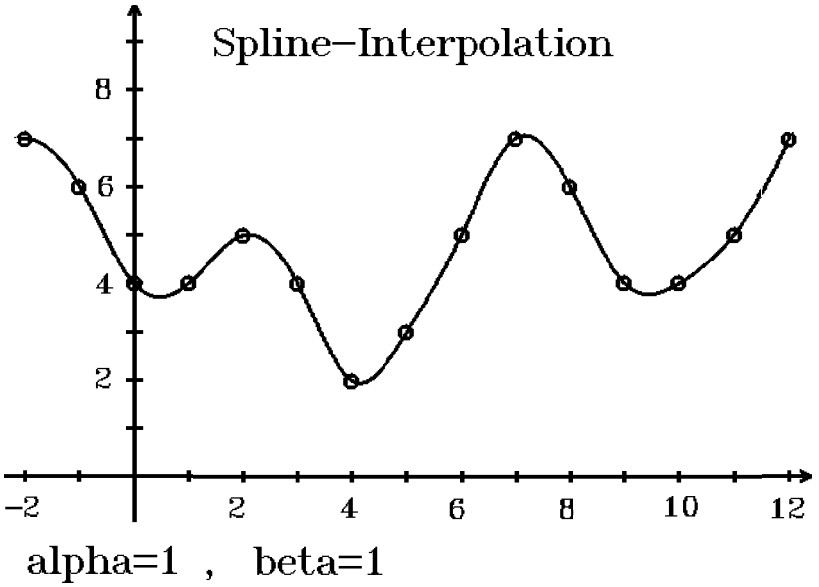
Rückwärtseinsetzen

für $i = (n-2)$ bis 1 (rückwärts)
    $b_i = b_i - c_i * b_{i+1}$
Ende $i-$Schleife
$b_0 = \alpha b_1$, $b_n = \beta b_{n-1}$

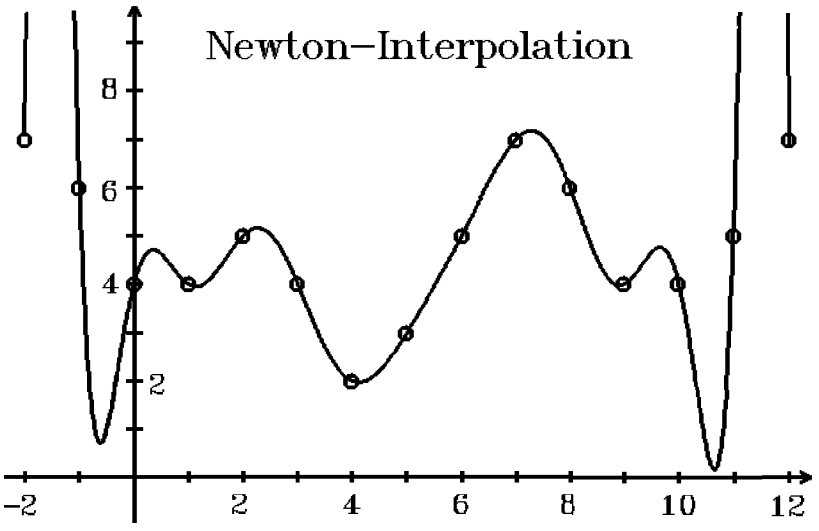
Die Speicherplätze b_i enthalten die gesuchten Werte y_i'' .

Beispiel 15 Interpolationspunkte

$x_i = -2\ ,\ -1\ ,\ 0\ ,\ 1\ ,\ 2\ ,\ 3\ ,\ 4\ ,\ 5\ ,\ 6\ ,\ 7\ ,\ 8\ ,\ 9\ ,\ 10\ ,\ 11\ ,\ 12$
$y_i = \ \ 7\ ,\ \ 6\ ,\ \ 4\ ,\ 4\ ,\ 5\ ,\ 4\ ,\ 2\ ,\ 3\ ,\ 5\ ,\ 7\ ,\ 6\ ,\ 4\ ,\ \ 4\ ,\ \ 5\ ,\ \ 7$



Im nächsten Bild sehen Sie zum Vergleich die an den Enden sehr stark oszillierende Kurve der Lagrange-Interpolation.



Eigenschaften der kubischen Splines

Sei $s(x)$ der laut Vorlesung konstruierte interpolierende kubische Spline in $[a,b] = [x_0, x_n]$, d.h. es gilt:

$s(x) = s_i(x) \quad \forall x \in [x_i, x_{i+1}]$ (s_i Polynom von Grad ≤ 3) , $(i = 0, 1, \ldots, n-1)$
$s(x_i) = y_i \quad \forall i = 0, 1, \ldots, n$
$y_0'' = \alpha y_1''$, $y_n'' = \beta y_{n-1}''$ oder $s'(x_0) = y_0'$, $s'(x_n) = y_n'$,

dann gelten folgende Eigenschaften :

1. $s(x)$ ist $2-$*mal stetig differenzierbar* in $[a,b]$ (per Konstruktion, da $s_{i-1}''(x_i) = s_i''(x_i) = y_i'' \quad \forall i = 1 \leq i \leq n-1$).

2. Falls $\alpha, \beta \geq -2$ (fest gewählt), so ist $s(x)$ *eindeutig bestimmt*, da dann das GLS eindeutig lösbar ist.
D.h.: Bei festem $\alpha, \beta \geq -2$ gibt es nur eine $2-$mal stetig differenzierbare Funktion in $[x_0, x_n]$, die in den Teilintervallen $[x_i, x_{i+1}]$ ein Polynom vom Grad ≤ 3 ist und für die $s(x_i) = y_i \quad \forall i = 0, 1, \ldots, n$ gilt.

3. Sei $y_0'' = y_n'' = 0$ oder $s'(x_0) = y_0'$, $s'(x_n) = y_n'$ gewählt und sei $B = \{\varphi \in C^2[a,b] \ : \ \varphi(x_i) = y_i \quad \forall i = 0, 1, \ldots, n\}$, dann gilt

$\|s''\|_2 \leq \|\varphi''\|_2 \quad \forall \varphi \in B$.

D.h. ungefähr: Die Krümmung des kubischen Splines ist *im quadratischen Mittel minimal*, wenn man $y_0'' = y_n'' = 0$ wählt oder die Ableitungen y_0' , y_n' vorgibt.

4. Sei $f \in C^2[a,b]$. Die Funktion f soll durch kubische Splines an den Stellen $a = x_0 < x_1 < x_2 < \ldots < x_n = b$ interpoliert werden. Wählen wir die Zerlegung $Z_n = \{x_0, x_1, \ldots, x_n\}$ immer feiner, d.h. $n \to \infty$, $|Z_n| \to 0$, also $|x_{i+1} - x_i| \to 0$, und ist s_n der zur Zerlegung Z_n gehörende kubische Spline, so gilt

$\lim\limits_{n\to\infty} \|s_n - f\|_\infty = 0$ und $\lim\limits_{n\to\infty} \|s_n' - f'\|_\infty = 0$.

D.h.: Es gilt gleichmäßige Konvergenz von $s_n \to f$ und $s_n' \to f'$ in $[a,b]$.

(Beweis zu 3. und 4. vgl. Literatur).

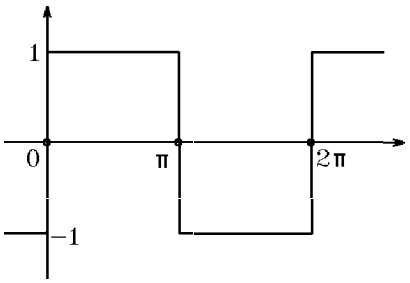
V Fourierreihen , Fast-Fourier-Transformation (FFT)

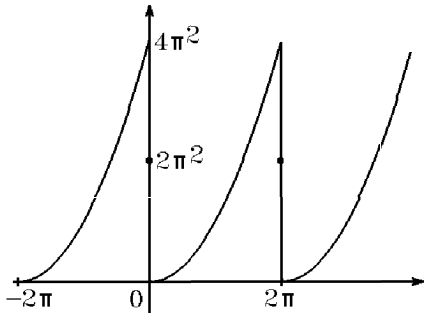
Sei $f : \mathbb{R} \to \mathbb{R}$ $2\pi-$periodisch, stückweise stetig, stückweise glatt.

An den Sprungstellen sei f folgendermaßen definiert :

$f(x_0) = \frac{1}{2}\Big(\lim\limits_{x\to x_0+} f(x) + \lim\limits_{x\to x_0-} f(x)\Big)$ (Mittelwert aus rechts- und linksseitigem Grenzwert).

Beispiele





$$f(x) = \begin{cases} 1 & \text{,falls } 0 < x < \pi \\ -1 & \text{,falls } \pi < x < 2\pi \end{cases}$$

$f(0) = f(\pi) = f(2\pi) = 0$.

$f(x) = x^2 \quad , \ 0 < x < 2\pi$

$f(0) = f(2\pi) = 2\pi^2$.

Sei $s(x) = \frac{a_0}{2} + \sum\limits_{k=1}^{\infty}\Big(a_k \cos kx + b_k \sin kx\Big)$ die *Fourierreihe* von f mit den *Fourierkoeffizienten*

$$a_k = \frac{1}{\pi}\int_0^{2\pi} f(x)\cos kx \ dx \quad , \quad b_k = \frac{1}{\pi}\int_0^{2\pi} f(x)\sin kx \ dx \quad , \quad (k = 0, 1, \ldots)$$

Unter den oben gemachten Voraussetzungen gilt für die Funktion f :

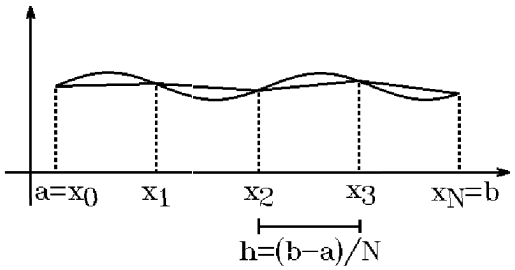
$f(x) = s(x) \quad \forall x \in \mathbb{R}$, d.h.: Die Funktion f wird durch ihre Fourierreihe s dargestellt (vgl. Vorlesung "Mathematik für Elektrotechniker II").

Sind die Integrale der Fourierkoeffizienten nur sehr schwierig zu berechnen, oder ist die Funktion f nur über Abtastwerte gegeben, so müssen die Integrale von a_k und b_k näherungsweise berechnet werden.

Näherungsweise Berechnung der a_k , b_k *mit Hilfe der Trapezregel*

Gesucht:
$\int_a^b g(x)\,dx$.

Mit der Schrittweite $h = \dfrac{b-a}{N}$
und den Unterteilungspunkten
$x_i = a + ih$, $i = 0, 1, \ldots, N$
erhalten wir die *Trapezregel* :



$$T(h) = \frac{h}{2}\Big(g(x_0) + 2\sum_{i=1}^{N-1} g(x_i) + g(x_N)\Big) .$$

Mit $x_j = jh$, $h = \dfrac{2\pi}{N}$ erhält man mit Hilfe der Trapezformel für a_k den Näherungswert

$$a_k^* = \frac{1}{\pi}\cdot\frac{2\pi}{2N}\Big\{f(x_0)\cos kx_0 + 2\sum_{j=1}^{N-1} f(x_j)\cos kx_j + f(x_N)\cos kx_N\Big\}$$

$$= \frac{1}{N}\Big\{2\sum_{j=0}^{N-1} f(x_j)\cos kx_j\Big\} \quad , \text{ da } f(x_N)\cos kx_N = f(x_0)\cos kx_0 \text{ (da } 2\pi\text{−period.).}$$

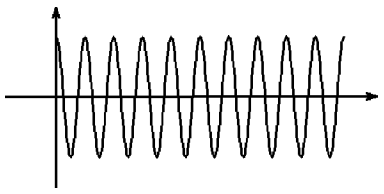
Also erhalten wir als Näherungswerte für a_k , b_k (für b_k analog)

$$a_k^* = \frac{2}{N}\sum_{j=0}^{N-1} f(x_j)\cos kx_j \quad , \quad b_k^* = \frac{2}{N}\sum_{j=0}^{N-1} f(x_j)\sin kx_j \quad , \quad (k = 0, 1, \ldots)$$

(hierbei ist $b_0^* = 0$).

Frage: Wie groß muß N (Anzahl der Teilintervalle bei der Trapezregel) gewählt werden, damit a_k^* , b_k^* die gesuchten Werte a_k , b_k möglichst gut annähern ?

Da $\cos kx$ und $\sin kx$
für große k sehr stark
oszillierende Funktionen
sind, muß N sicher sehr groß
gewählt werden.



Eine genauere Aussage können wir mit Hilfe der folgenden Fehlerabschätzung machen. Als Hilfsmittel für die Herleitung dieser Fehlerabschätzung benötigen wir zunächst die folgenden *Orthogonalitätseigenschaften* :

Satz 5.1 : *Orthogonalitätseigenschaften*

Sei $x_j = jh$, $h = \dfrac{2\pi}{N}$, dann gilt

a) $$\sum_{j=0}^{N-1} \cos kx_j = \begin{cases} 0 & ,\text{falls } \dfrac{k}{N} \notin \mathbb{Z} \\ N & ,\text{falls } \dfrac{k}{N} \in \mathbb{Z}. \end{cases}$$

b) $$\sum_{j=0}^{N-1} \sin kx_j = 0 \quad \forall k \in \mathbb{Z}.$$

c) $$\sum_{j=0}^{N-1} \cos kx_j \cos lx_j = \begin{cases} 0 & ,\text{falls } \dfrac{k+l}{N} \notin \mathbb{Z} \text{ und } \dfrac{k-l}{N} \notin \mathbb{Z} \\ \dfrac{N}{2} & ,\text{falls } \dfrac{k+l}{N} \in \mathbb{Z} \text{ oder } \dfrac{k-l}{N} \in \mathbb{Z} \\ N & ,\text{falls } \dfrac{k+l}{N} \in \mathbb{Z} \text{ und } \dfrac{k-l}{N} \in \mathbb{Z}. \end{cases}$$

d) $$\sum_{j=0}^{N-1} \sin kx_j \sin lx_j = \begin{cases} 0 & ,\text{falls beide} \notin \mathbb{Z} \text{ oder beide} \in \mathbb{Z} \\ -\dfrac{N}{2} & ,\text{falls } \dfrac{k+l}{N} \in \mathbb{Z} \text{ und } \dfrac{k-l}{N} \notin \mathbb{Z} \\ \dfrac{N}{2} & ,\text{falls } \dfrac{k+l}{N} \notin \mathbb{Z} \text{ und } \dfrac{k-l}{N} \in \mathbb{Z}. \end{cases}$$

e) $$\sum_{j=0}^{N-1} \cos kx_j \sin lx_j = 0 \quad \forall k, l \in \mathbb{N}_0 \ .$$

Beweis :

Zu a),b) : Mit $x_j = j(2\pi/N) = 2j\pi/N$ gilt:

$$\sum_{j=0}^{N-1} (\cos kx_j + i \sin kx_j) = \sum_{j=0}^{N-1} \mathrm{e}^{ikx_j} = \sum_{j=0}^{N-1} \left(\mathrm{e}^{i2k\pi/N}\right)^j = \frac{1 - (\mathrm{e}^{i2k\pi/N})^N}{1 - \mathrm{e}^{i2k\pi/N}}$$

$$= \frac{1 - \mathrm{e}^{i2k\pi}}{1 - \mathrm{e}^{i2k\pi/N}} = 0 \ , \ \text{falls } \mathrm{e}^{i2k\pi/N} \neq 1 \quad (\text{da } \mathrm{e}^{i2k\pi} = 1).$$

Ist $\mathrm{e}^{i2k\pi/N} = 1 \quad \Rightarrow \quad \sum_{j=0}^{N-1} \mathrm{e}^{ikx_j} = \sum_{j=0}^{N-1} \left(\mathrm{e}^{i2k\pi/N}\right)^j = \sum_{j=0}^{N-1} 1 = N$.

Es gilt $\mathrm{e}^{i2k\pi/N} = 1 \quad \Leftrightarrow \quad \cos(2k\pi/N) = 1$ und $\sin(2k\pi/N) = 0 \quad \Leftrightarrow \quad \dfrac{k}{N} \in \mathbb{Z}$.

Also erhalten wir insgesamt

$$\sum_{j=0}^{N-1} (\cos kx_j + i \sin kx_j) = \begin{cases} 0 & ,\text{falls } \dfrac{k}{N} \notin \mathbb{Z} \\ N & ,\text{falls } \dfrac{k}{N} \in \mathbb{Z}. \end{cases}$$

Für Real- bzw. Imaginärteil folgt dann a) bzw. b).

Zu c),d),e) :

Mit Hilfe der Additionstheoreme

$$\cos kx_j \cos lx_j = \frac{1}{2}\Big(\cos(k-l)x_j + \cos(k+l)x_j\Big)$$
$$\sin kx_j \sin lx_j = \frac{1}{2}\Big(\cos(k-l)x_j - \cos(k+l)x_j\Big)$$
$$\cos kx_j \sin lx_j = \frac{1}{2}\Big(\sin(k+l)x_j - \sin(k-l)x_j\Big)$$
folgen sofort aus a) und b) die Aussagen c),d),e).

Mit Hilfe dieser Orthogonalitätseigenschaften lassen sich nun die folgenden Fehlerabschätzungen zeigen:

Satz 5.2 : *Fehlerabschätzungen*

Für $N = 2n$ gelten die folgenden Fehlerabschätzungen zwischen den exakten Fourierkoeffizienten a_k (bzw. b_k) und den mit Hilfe der Trapezregel näherungsweise berechneten Werten a_k^* (bzw. b_k^*):

$$|a_k^* - a_k| \le \sum_{\mu=1}^{\infty}\Big(|a_{\mu N-k}| + |a_{\mu N+k}|\Big) \quad , \quad (0 \le k \le n-1),$$
$$|b_k^* - b_k| \le \sum_{\mu=1}^{\infty}\Big(|b_{\mu N-k}| + |b_{\mu N+k}|\Big) \quad , \quad (1 \le k \le n-1).$$

Beweis : für $0 < k \le n-1$ (für $k = 0$ analog)

$$a_k^* = \frac{2}{N}\sum_{j=0}^{N-1} f(x_j)\cos kx_j \quad \text{(für } f(x_j) \text{ setzen wir die Fourierreihe } s(x_j) \text{ ein)}$$
$$= \frac{2}{N}\sum_{j=0}^{N-1}\Big\{\frac{a_0}{2} + \sum_{l=1}^{\infty}(a_l \cos lx_j + b_l \sin lx_j)\Big\}\cos kx_j$$
$$= \frac{2}{N}\Big\{\frac{a_0}{2}\underbrace{\sum_{j=0}^{N-1}\cos kx_j}_{=0} + \sum_{l=1}^{\infty}\Big(a_l \sum_{j=0}^{N-1}\cos lx_j \cos kx_j + b_l \underbrace{\sum_{j=0}^{N-1}\sin lx_j \cos kx_j}_{=0}\Big)\Big\}$$
$$= \frac{2}{N}\sum_{l=1}^{\infty} a_l \sum_{j=0}^{N-1}\cos lx_j \cos kx_j \quad \text{(da } k/N = k/(2n) \notin \mathbb{Z} \text{ für } 0 < k \le n-1).$$

Für $k = l$ gilt: $\frac{k-l}{N} \in \mathbb{Z}$, $\frac{k+l}{N} = \frac{2k}{2n} = \frac{k}{n} \notin \mathbb{Z}$ (da $0 < k \le n-1$)).

Für $k \ne l$ und $\frac{k-l}{N} \in \mathbb{Z} \;\Rightarrow\; l-k = \mu N \;\Rightarrow\; \frac{l+k}{N} = \frac{\mu N + 2k}{N} = \mu + \frac{k}{n} \notin \mathbb{Z}$ (da $0 < k \le n-1$)).

Für $k \ne l$ und $\frac{k+l}{N} \in \mathbb{Z} \;\Rightarrow\; l+k = \mu N \;\Rightarrow\; \frac{l-k}{N} = \frac{\mu N - 2k}{N} = \mu - \frac{k}{n} \notin \mathbb{Z}$ (da $0 < k \le n-1$)).

Also gilt nach Satz 5.1 c): $\displaystyle\sum_{j=0}^{N-1}\cos lx_j \cos kx_j = \begin{cases} 0 & \text{,falls } l \ne \mu N \pm k \;\; , \; \mu \in \mathbb{N}_0 \\ \frac{N}{2} & \text{,falls } l = \mu N \pm k \;\; , \; \mu \in \mathbb{N}_0. \end{cases}$

Insgesamt folgt dann

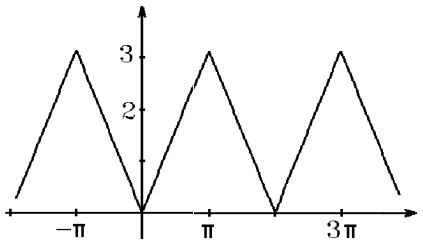
$$a_k^* = a_k + \sum_{\mu=1}^{\infty}(a_{\mu N-k} + a_{\mu N+k}) \quad \Rightarrow \quad |a_k^* - a_k| \le \sum_{\mu=1}^{\infty}(|a_{\mu N-k}| + |a_{\mu N+k}|) \ .$$

(Abschätzung für b_k analog).

Weiß man, wie die Koeffizienten a_k , b_k gegen 0 konvergieren, so kann man aus diesen Fehlerabschätzungen das N (Anzahl der notwendigen Teilintervalle bei der Trapezregel) bestimmen, das einen Fehler $|a_k^* - a_k| < \delta$ garantiert.

Beispiel

$f(x) = |x|$, falls $-\pi < x < \pi$.



Die exakten Fourierkoeffizienten sind

$a_k = -\dfrac{4}{\pi k^2}$, falls k ungerade,

$a_k = 0$, falls k gerade , $b_k = 0 \quad \forall k \ge 1$.

In diesem Fall erhalten wir als Fehlerabschätzung

$$\begin{aligned}
|a_k^* - a_k| &\le \frac{4}{\pi}\sum_{\mu=1}^{\infty}\Big(\frac{1}{(\mu N-k)^2} + \frac{1}{(\mu N+k)^2}\Big) \\
&= \frac{4}{\pi}\sum_{\mu=1}^{\infty}\frac{\mu^2N^2 + 2k\mu N + k^2 + \mu^2N^2 - 2k\mu N + k^2}{(\mu^2N^2 - k^2)^2} \\
&= \frac{8}{\pi}\sum_{\mu=1}^{\infty}\frac{\mu^2N^2 + k^2}{(\mu^2N^2 - k^2)^2} \approx \frac{8}{\pi}\sum_{\mu=1}^{\infty}\frac{1}{\mu^2N^2} \\
&= \frac{8}{\pi N^2}\sum_{\mu=1}^{\infty}\frac{1}{\mu^2} = \frac{8}{\pi N^2}\cdot\frac{\pi^2}{6} = \frac{4\pi}{3N^2} < 10^{-6} \quad \Rightarrow \quad N > 2046 \ .
\end{aligned}$$

Also benötigen wir in diesem Beispiel sehr viele Teilintervalle. Die Funktion in diesem Beispiel ist *stetig* und *stückweise glatt*. Hat die Funktion f sogar *Sprungstellen*, so muß N noch größer gewählt werden, um die gleiche Genauigkeit zu erreichen, denn dann konvergieren die Fourierkoeffizienten nur wie $\dfrac{1}{k}$.

Allgemein gilt folgende Aussage über das Konvergenzverhalten der Fourierkoeffizienten einer $2\pi-$periodischen, stückweise stetigen, stückweise glatten Funktion f :

Satz 5.3 : *Konvergenzverhalten der Fourierkoeffizienten*

f sei $2\pi-$periodisch und $(r-1)-$mal stetig differenzierbar.
Die $r-$te Ableitung von f sei *stückweise stetig* und *stückweise glatt*. Dann gilt für die Fourierkoeffizienten von f :

$$|a_k| \le \frac{M}{k^{r+1}} \quad , \quad |b_k| \le \frac{M}{k^{r+1}} \quad , \quad (M > 0) \ .$$

Es gilt also folgende *Faustregel*:

Sprünge in der $r-$ten Ableitung bedeuten $(r+1)-$te Ordnung der Fourierkoeffizienten.

Beispiel $f(x) = |x|$, $(-\pi < x < \pi)$ (vgl. S.62)

f ist stetig, Sprünge treten in der 1. Ableitung auf $\Rightarrow$ $a_k \approx \frac{1}{k^2}$.

Ergebnis:

Man muß sehr viele Unterteilungsintervalle wählen, also N sehr groß wählen, um genügend genaue Ergebnisse für a_k^* , b_k^* zu erhalten.
Sollen alle a_k^* , b_k^* , $k = 0, 1, \ldots, N/2,$ berechnet werden, so wächst der Rechenaufwand mit wachsendem N proportional N^2 , also:

Rechenaufwand $\approx N^2$.

Wir werden später einen Algorithmus (Fast-Fourier-Transformation (FFT)) behandeln, der diesen Rechenaufwand wesentlich verringert. Zu diesem Zweck führen wir zunächst eine *diskrete Fourier-Transformation* ein, aus der wir dann die gesuchten Fourierkoeffizienten a_k^* , b_k^* berechnen können:

Zurückführung auf diskrete Fourier-Transformation

Sei i.f. :

$$N = 2^\gamma \ , \ n = \frac{N}{2} \ , \ x_j = j\frac{2\pi}{N} = \frac{j\pi}{n} \quad , \quad (0 \le j \le N-1)$$

$$a'_k = \sum_{j=0}^{N-1} f(x_j) \cos kx_j \quad , \quad b'_k = \sum_{j=0}^{N-1} f(x_j) \sin kx_j \quad , \quad (k = 0, 1, \ldots n)$$

dann gilt: $b'_0 = b'_n = 0$, $a_k^* = \frac{2}{N} a'_k$, $b_k^* = \frac{2}{N} b'_k$.

Wir führen die folgenden komplexen Größen y_j ein:

$$y_j = f(x_{2j}) + if(x_{2j+1}) \quad , \quad (j = 0, 1, \ldots, n-1)$$

$$w_n = \mathrm{e}^{-i\frac{2\pi}{n}} = \cos\frac{2\pi}{n} - i\sin\frac{2\pi}{n}$$

Die komplexe Zahl w_n liegt auf dem Einheitskreis, denn es gilt: $|w_n| = 1$.

Definition 5.4 : *Diskrete Fourier-Transformation*

Zusammen mit y_j und w_n definieren wir nun die *diskrete Fourier-Transformation* der Ordnung n

$$c_k = \sum_{j=0}^{n-1} y_j w_n^{kj} \quad , \quad (k = 0, 1, \ldots, n-1)$$

Satz 5.5 : *Beziehungen zwischen* a'_k , b'_k *und* c_k

Es gelten folgende Beziehungen zwischen den a'_k , b'_k und den c_k :

$$(1) \qquad a'_k - ib'_k = \frac{1}{2}(c_k + \overline{c}_{n-k}) + \frac{1}{2i}(c_k - \overline{c}_{n-k})\mathrm{e}^{-ik\pi/n}$$

$$(2) \qquad a'_{n-k} + ib'_{n-k} = \frac{1}{2}(c_k + \overline{c}_{n-k}) - \frac{1}{2i}(c_k - \overline{c}_{n-k})\mathrm{e}^{-ik\pi/n}$$

für $k = 0, 1, \ldots, n$, falls $b'_0 = b'_n = 0$, $c_n = c_0$

Beweis :

Zu (1) : Mit $w_n = \mathrm{e}^{-i2\pi/n}$ und $x_j = \dfrac{j\pi}{n}$ gilt

$$\frac{1}{2}(c_k + \overline{c}_{n-k}) = \frac{1}{2}\sum_{j=0}^{n-1}\left(y_j w_n^{kj} + \overline{y}_j w_n^{-(n-k)j}\right) = \frac{1}{2}\sum_{j=0}^{n-1}(y_j + \overline{y}_j)w_n^{kj} \ ,$$

da $w_n^{-nj} = \mathrm{e}^{i2\pi j} = 1$.

Analog gilt: $\dfrac{1}{2i}(c_k - \overline{c}_{n-k}) = \dfrac{1}{2i}\displaystyle\sum_{j=0}^{n-1}(y_j - \overline{y}_j)w_n^{kj}$.

Da $y_j + \overline{y}_j = 2f(x_{2j})$ und $y_j - \overline{y}_j = 2if(x_{2j+1})$, folgt

$$\begin{aligned}
&\frac{1}{2}(c_k + \overline{c}_{n-k}) + \frac{1}{2i}(c_k - \overline{c}_{n-k})\mathrm{e}^{-ik\pi/n} \\
&= \sum_{j=0}^{n-1} f(x_{2j})\mathrm{e}^{-i(2jk\pi/n)} + \sum_{j=0}^{n-1} f(x_{2j+1})\mathrm{e}^{-i(2j+1)k\pi/n} \\
&= \sum_{j=0}^{n-1} f(x_{2j})(\cos kx_{2j} - i\sin kx_{2j}) + \sum_{j=0}^{n-1} f(x_{2j+1})(\cos kx_{2j+1} - i\sin kx_{2j+1}) \\
&= \sum_{l=0}^{N-1} f(x_l)\cos kx_l - i\sum_{l=0}^{N-1} f(x_l)\sin kx_l = a'_k - ib'_k \ .
\end{aligned}$$

Zu (2) : Mit $\mathrm{e}^{in\pi/n} = \mathrm{e}^{i\pi} = -1$ gilt

$$\frac{1}{2}(c_k + \overline{c}_{n-k}) - \frac{1}{2i}(c_k - \overline{c}_{n-k})\mathrm{e}^{-ik\pi/n}$$

$$= \sum_{j=0}^{n-1} f(x_{2j})\mathrm{e}^{-i(2jk\pi/n)} + \sum_{j=0}^{n-1} f(x_{2j+1})\mathrm{e}^{-i(2j+1)k\pi/n}\,\mathrm{e}^{in\pi/n}$$

Multiplikation mit $\mathrm{e}^{i2jn\pi/n} = 1$ ergibt:

$$= \sum_{j=0}^{n-1} f(x_{2j})\mathrm{e}^{i(n-k)2j\pi/n} + \sum_{j=0}^{n-1} f(x_{2j+1})\mathrm{e}^{i(n-k)(2j+1)\pi/n}$$

$$= \sum_{j=0}^{n-1} f(x_{2j})\mathrm{e}^{i(n-k)x_{2j}} + \sum_{j=0}^{n-1} f(x_{2j+1})\mathrm{e}^{i(n-k)x_{2j+1}}$$

$$= \sum_{j=0}^{n-1} f(x_{2j})(\cos(n-k)x_{2j} + i\sin(n-k)x_{2j})$$

$$+ \sum_{j=0}^{n-1} f(x_{2j+1})(\cos(n-k)x_{2j+1} + i\sin(n-k)x_{2j+1})$$

$$= \sum_{l=0}^{N-1} f(x_l)\cos(n-k)x_l + i\sum_{l=0}^{N-1} f(x_l)\sin(n-k)x_l = a'_{n-k} + ib'_{n-k}\ .$$

Mit Hilfe von (1) und (2) von Satz 5.5 kann man nun sehr einfach aus den c_k die a'_k , b'_k und damit die gesuchten Werte a^*_k , b^*_k folgendermaßen berechnen:

$(1) + (2) \quad \Rightarrow \quad (a'_k + a'_{n-k}) + i(b'_{n-k} - b'_k) = c_k + \overline{c}_{n-k} \quad \Rightarrow$

(3)	$a'_k + a'_{n-k} = Re(c_k + \overline{c}_{n-k})$
(4)	$b'_{n-k} - b'_k = Im(c_k + \overline{c}_{n-k})$

$(1) - (2) \quad \Rightarrow \quad (a'_k - a'_{n-k}) - i(b'_k + b'_{n-k}) = \dfrac{1}{i}(c_k - \overline{c}_{n-k})\mathrm{e}^{-ik\pi/n}$

Multiplikation mit i ergibt:

$$(b'_k + b'_{n-k}) + i(a'_k - a'_{n-k}) = (c_k - \overline{c}_{n-k})\mathrm{e}^{-ik\pi/n}\ .$$

Mit $u_n = \mathrm{e}^{-i\pi/n}$ folgt hieraus:

(5)	$a'_k - a'_{n-k} = Im\Big((c_k - \overline{c}_{n-k})u_n^k\Big)$
(6)	$b'_{n-k} + b'_k = Re\Big((c_k - \overline{c}_{n-k})u_n^k\Big)$

Die Potenzen u_n^k kann man sukzessive durch Multiplikation berechnen.

Indem wir die Gleichungen (3), (4), (5), (6) addieren bzw. subtrahieren und berücksichtigen, daß $a_k^* = \frac{2}{N}a'_k = \frac{2}{2n}a'_k$, $b_k^* = \frac{2}{N}b'_k = \frac{2}{2n}b'_k$ gilt, erhalten wir den folgenden Algorithmus für die Berechnung der a_k^* , b_k^* aus den c_k :

Algorithmus zur Berechnung der a_k^* , b_k^* aus den c_k

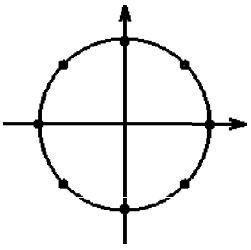
$u = \mathrm{e}^{-i\pi/n} = \cos\frac{\pi}{n} - i\sin\frac{\pi}{n}$
$u1 = 1 + i \cdot 0$
für $k = 0$ bis $n/2$
    $h1 = c_k + \overline{c}_{n-k}$
    $h2 = (c_k - \overline{c}_{n-k}) * u1$
    $a_k^* = (Re(h1) + Im(h2))/(2n)$
    $a_{n-k}^* = (Re(h1) - Im(h2))/(2n)$
    $b_k^* = (Re(h2) - Im(h1))/(2n)$
    $b_{n-k}^* = (Re(h2) + Im(h1))/(2n)$
    $u1 = u1 * u$
Ende $k-$Schleife.

FFT (Fast-Fourier-Transformation)

Das Problem, das nun noch übrig bleibt, ist die Auswertung der *diskreten Fourier-Transformation*: $c_k = \sum_{j=0}^{n-1} y_j w_n^{kj}$, $(k = 0, 1, \ldots, n-1)$, mit $w_n = \mathrm{e}^{-i(2\pi/n)}$.

Um den *Rechenaufwand* von n^2 zu verringern, bedient man sich des *FFT-Algorithmus (Fast-Fourier-Transformation)*. Hierbei wird wesentlich die Eigenschaft ausgenutzt, daß die Werte w_n^{kj} auf dem Einheitskreis liegen und sich nach jeweils $n-$Schritten wiederholen. Es gilt:

$w_n^{k+n} = w_n^k \quad \Rightarrow \quad \mathrm{mod}\ n$,
$w_n^n = \mathrm{e}^{-i(2\pi/n)n} = 1$,
$w_n^{n/2} = \mathrm{e}^{-i\pi} = -1$.



Die Idee des FFT-Algorithmus besteht darin, daß man eine diskrete Fourier-Transformation der Ordnung n auf zwei diskrete Fourier-Transformationen der Ordnung $n/2$ reduzieren kann. Diese Fourier-Transformationen der Ordnung $n/2$ lassen sich dann wieder jeweils auf zwei der Ordnung $n/4$ reduzieren, usw., bis man bei der Ornung 2 angelangt ist. Um die Reduzierung duchzuführen, faßt man die c_k mit geradem Index und die c_k mit ungeradem Index zusammen. Es gilt:

FFT-Algorithmus *Sukzessive Reduktion*

$$c_{2k} = \sum_{j=0}^{n-1} y_j w_n^{2kj} = \sum_{j=0}^{n/2-1} y_j w_n^{2kj} + \sum_{\substack{j=n/2 \\ j \to j+n/2}}^{n-1} y_j w_n^{2kj}$$

$$= \sum_{j=0}^{n/2-1} y_j w_n^{2kj} + \sum_{j=0}^{n/2-1} y_{j+n/2} w_n^{2k(j+n/2)} = \sum_{j=0}^{n/2-1} (y_j + y_{j+n/2})(w_n^2)^{kj} \quad (\text{da } w_n^{kn} = 1)$$

$$= \sum_{j=0}^{m-1} z_j (w_n^2)^{kj} \ .$$

Dies ist eine diskrete Fourier-Transformation der Ordnung $m = \dfrac{n}{2}$ mit $z_j = (y_j + y_{j+n/2})$ und w_n^2 anstelle von w_n.

$$c_{2k+1} = \sum_{j=0}^{n-1} y_j w_n^{(2k+1)j} = \sum_{j=0}^{n/2-1} y_j w_n^{(2k+1)j} + \sum_{\substack{j=n/2 \\ j \to j+n/2}}^{n-1} y_j w_n^{(2k+1)j}$$

$$= \sum_{j=0}^{n/2-1} y_j w_n^{(2k+1)j} + \sum_{j=0}^{n/2-1} y_{j+n/2} w_n^{(2k+1)(j+n/2)}$$

$$= \sum_{j=0}^{n/2-1} y_j w_n^{2kj} w_n^j + \sum_{j=0}^{n/2-1} y_{j+n/2} w_n^{2kj} w_n^j w_n^{nk} w_n^{n/2}$$

$$= \sum_{j=0}^{n/2-1} \Big((y_j - y_{j+n/2}) w_n^j \Big) (w_n^2)^{kj} \quad (\text{da } w_n^{kn} = 1 \text{ und } w_n^{n/2} = -1)$$

$$= \sum_{j=0}^{m-1} \tilde{z}_j (w_n^2)^{kj} \ .$$

Dies ist eine diskrete Fourier-Transformation der Ordnung $m = \dfrac{n}{2}$ mit $\tilde{z}_j = (y_j - y_{j+n/2}) w_n^j$ und w_n^2 anstelle von w_n.

Damit haben wir die diskrete Fourier-Transformation der Ordnung n auf zwei diskrete Fourier-Transformationen der Ordnung $n/2$ zurückgeführt. Diese lassen sich dann wieder auf jeweils zwei diskrete Fourier-Transformationen der Ordnung $n/4$ zurückführen, usw., bis man bei der Ordnung 2 angelangt ist.
Da $n = 2^{\gamma-1} \Rightarrow$ nach $\gamma - 1 = \log_2 n$ Schritten hat man nur noch diskrete Fourier-Transformationen der Ordnung 2 . Das ergibt folgenden Rechenaufwand

Rechenaufwand $= n \log_2 n$ (anstelle von n^2)

z.B.: $n = 2048 \Rightarrow n = 2^{11}$

$\Rightarrow$ Rechenaufwand $11 \cdot 2048 = 22\,528$ anstelle von $n^2 = (2048)^2 = 4\,194\,304$.

Beispiel : $n = 4\ ,\ N = 8\ ,\ w = w_4 = \mathrm{e}^{-2\pi i/4} = \mathrm{e}^{-i\pi/2}\ ,$

$c_k = \sum_{j=0}^{3} y_j w^{kj} \quad , \quad (k = 0, 1, 2, 3)\ , \quad$ ergibt

$$\boxed{\begin{aligned} c_0 &= y_0 + y_1 + y_2 + y_3 \quad , \quad (k = 0\ ,\ w^{kj} = w^0 = 1) \\ c_1 &= y_0 + y_1 w + y_2 w^2 + y_3 w^3 \quad , \quad (k = 1\ ,\ w^{kj} = w^j) \\ c_2 &= y_0 + y_1 w^2 + y_2 w^4 + y_3 w^6 \quad , \quad (k = 2\ ,\ w^{kj} = w^{2j}) \\ c_3 &= y_0 + y_1 w^3 + y_2 w^6 + y_3 w^9 \quad , \quad (k = 3\ ,\ w^{kj} = w^{3j}) \end{aligned}}$$

Da $w^4 = w_4^4 = 1 \quad \Rightarrow \quad w^{k+4} = w^k$, also $w^4 = w^0 = 1\ ,\ w^5 = w^1 = w\ ,$
$w^6 = w^2\ ,\ w^9 = w^5 = w\ .$

Damit gilt

$$\begin{aligned} c_0 &= y_0 + y_1 + y_2 + y_3 \\ c_1 &= y_0 + y_1 w + y_2 w^2 + y_3 w^3 \\ c_2 &= y_0 + y_1 w^2 + y_2 + y_3 w^2 \\ c_3 &= y_0 + y_1 w^3 + y_2 w^2 + y_3 w^5\ . \end{aligned}$$

Durch Vertauschen der Zeilen erhalten wir, da $w^2 = w_4^2 = -1\ ,\ w^3 = w\, w^2 = -w$

$$\boxed{\begin{aligned} c_0 &= (y_0 + y_2) + (y_1 + y_3) \\ c_2 &= (y_0 + y_2) + (y_1 + y_3) w^2 \end{aligned}}$$

$$\boxed{\begin{aligned} c_1 &= (y_0 - y_2) + \Big((y_1 - y_3) w\Big) \\ c_3 &= (y_0 - y_2) + \Big((y_1 - y_3) w\Big) w^2 \end{aligned}}$$

Mit $z_0 = y_0 + y_2 \quad , \quad z_1 = y_1 + y_3 \quad , \quad \tilde{z}_0 = y_0 - y_2 \quad , \quad \tilde{z}_1 = (y_1 - y_3) w \quad \Rightarrow$

$$\boxed{\begin{aligned} \tilde{c}_0 &= z_0 + z_1 \\ \tilde{c}_1 &= z_0 + z_1 (w^2) \end{aligned}} \qquad \boxed{\begin{aligned} \tilde{c}_2 &= \tilde{z}_0 + \tilde{z}_1 \\ \tilde{c}_3 &= \tilde{z}_0 + \tilde{z}_1 (w^2) \end{aligned}}$$

Also erhalten wir zwei diskrete Fourier-Transformationen der Ordnung $n/2 = 2$ mit $c_0 = \tilde{c}_0\ ,\ c_1 = \tilde{c}_2\ ,\ c_2 = \tilde{c}_1\ ,\ c_3 = \tilde{c}_3\ .$

Bitumkehr

Um die c_k-Werte wieder in richtiger Reihenfolge zu erhalten, muß zum Schluß zurückgetauscht werden. Das geschieht mit Hilfe der *Bitumkehr* der binären Darstellung der Indizes:

Beispiel *Bitumkehr*

j	binäre Darstellung	Bitumkehr	k	$\tilde{c}_j$	c_j
0	0 0	0 0	0	$\tilde{c}_0$	c_0
1	0 1	1 0	2	$\tilde{c}_1$	c_2
2	1 0	0 1	1	$\tilde{c}_2$	c_1
3	1 1	1 1	3	$\tilde{c}_3$	c_3

falls $k > j \quad \Rightarrow$ vertauschen, also hier: c_2 mit c_1 vertauschen.

Bitumkehr: z.B.: 1 0 1 1 $\rightarrow$ 1 1 0 1 (von hinten nach vorne lesen).

Man kann auf die Bitumkehr verzichten, wenn man im Algorithmus eine *Index-Rechnung* durchführt. In diesem Fall benötigt man ein zusätzliches Index-Feld.

Im folgenden werden zwei unterschiedliche FFT - Algorithmen angegeben, der eine mit Bitumkehr, der andere mit Index-Rechnung:

FFT - Algorithmus (mit Bitumkehr)

Gegeben: $N = 2^\gamma$, $n = 2^{\gamma-1}$
Funktion $f(x)$ von 0 bis 2π definieren !
An den Sprungstellen x_0 : $f(x_0) =$"Mittelwert" definieren !

Berechnen:

$y_j = f(2j\pi/n) + i\, f((2j+1)\pi/n) \quad , \quad (j = 0, 1, \ldots, n-1)$
$m = \dfrac{n}{2} \quad , \quad p = \gamma - 1 \qquad (= \mathrm{int}(\ln n / \ln 2 \ + 0.5))$
$w = \cos\dfrac{\pi}{m} - i \sin\dfrac{\pi}{m}$
$w_0 = 1 + i \cdot 0$
für $j = 1$ bis $(m-1)$: $\quad w_j = w_{j-1} * w \quad$: Ende $j-$Schleife
$f = m \quad , \quad g = 1$
für $l = 1$ bis p
    $a = 0$
    für $r = 1$ bis g
        $d = 0$
        für $i = 0$ bis $(f-1)$
            $i1 = a + i \quad , \quad i2 = i1 + f$
            $h = y_{i1} - y_{i2} \quad , \quad y_{i1} = y_{i1} + y_{i2} \quad , \quad y_{i2} = h * w_d$
            $d = d + g$
        Ende $i-$Schleife
        $a = a + 2 * f$
    Ende $r-$Schleife
    $f = f/2 \quad , \quad g = 2 * g$
Ende $l-$Schleife

Bitumkehr (Umordnung der y_k-Werte)

für $j = 0$ bis $(n-1)$
  $r = j \;\;,\;\; k = 0 \;\;,\;\; l = m$
  für $i = 1$ bis p
    $k = k + (r \bmod 2) * l$ (mod $\hat{=}$ Divisionsrest)
    $r = \text{int}(r/2) \;\;,\;\; l = l/2$ (Integer-Division, d.h: abschneiden)
  Ende $i-$Schleife
  falls $k > j \;\; \Rightarrow \;\; h = y_j \;\;,\;\; y_j = y_k \;\;,\;\; y_k = h$
Ende $j-$Schleife

Die gesuchten c_k-Werte sind auf den y_k gespeichert. Mit Hilfe des Algorithmus von S.66 können nun die gesuchten Fourierkoeffizienten a_k^* , b_k^* berechnet werden. Vorher muß $y_n = y_0$ gesetzt werden. Es gilt für $k = n$: $b_n^* = 0$. Dieser Wert entspricht nicht dem Fourierkoeffizienten b_n .

FFT - Algorithmus (mit Index-Rechnung)

Gegeben: $N = 2^\gamma$, $n = 2^{\gamma-1}$
  Funktion $f(x)$ von 0 bis 2π definieren !
  An den Sprungstellen x_0 : $f(x_0)$ ="Mittelwert" definieren !

Berechnen:

$y_j = f(2j\pi/n) + i\, f((2j+1)\pi/n) \;\;,\;\; (j = 0, 1, \ldots, n-1)$
für $j = 0$ bis $(n-1)$: $ind_j = 0$: Ende $j-$Schleife (Index-Feld vorbelegen)
$w = \cos\dfrac{2\pi}{n} - i\,\sin\dfrac{2\pi}{n}$
$p = 1$
für $m = 0$ bis $\gamma - 1$ $(\gamma - 1 = \text{int}(\ln n/\ln 2 \; + 0.5))$
  $w1 = 1 + i \cdot 0$
  $q = n/(2p)$
  für $j = 0$ bis $(q-1)$
    für $l = j$, $j + 2q$, $j + 4q$, $j + 6q$, ... , solange $l < n$
      $z = y_l + y_{l+q} \;\;,\;\; y_{l+q} = (y_l - y_{l+q}) * w1 \;\;,\;\; y_l = z$
      $ind_{l+q} = ind_{l+q} + p$
    Ende $l-$Schleife
    $w1 = w1 * w$
  Ende $j-$Schleife
  $p = 2 * p \;\;,\;\; w = w * w$
Ende $m-$Schleife

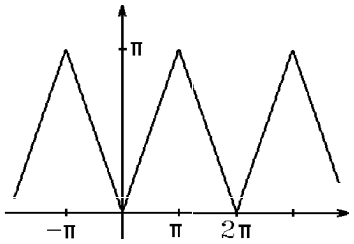
Umordnung der y_k-Werte

für $j = 0$ bis $(n-1)$
  falls $j < ind_j \;\; \Rightarrow \;\; z = y_{ind_j} \;\;,\;\; y_{ind_j} = y_j \;\;,\;\; y_j = z$
Ende $j-$Schleife

Die gesuchten c_k-Werte sind auf den y_k gespeichert. Mit Hilfe des Algorithmus von S.66 können nun die gesuchten Fourierkoeffizienten a_k^* , b_k^* berechnet werden. Vorher muß $y_n = y_0$ gesetzt werden. Es gilt für $k = n$: $b_n^* = 0$. Dieser Wert entspricht nicht dem Fourierkoeffizienten b_n .

Beispiele

Beispiel 1. (vgl. S.62)



$f(x) = |x|$, falls $-\pi \le x \le \pi$

$$\Rightarrow \quad f(x) = \begin{cases} x & ,\text{falls } 0 \le x \le \pi \\ 2\pi - x & ,\text{falls } \pi < x \le 2\pi. \end{cases}$$

Exakte Fourierkoeffizienten:

$$a_0 = \pi \quad , \quad a_k = \begin{cases} 0 & ,\text{falls } k \text{ gerade } (\neq 0) \\ -\dfrac{4}{\pi k^2} & ,\text{falls } k \text{ ungerade} \end{cases}$$

$b_k = 0 \quad \forall k \in I\!N.$

Die Berechnung der a_k^* , b_k^* mit Hilfe des FFT-Algorithmus und mit Hilfe des Algorithmus von S.66 ergab für unterschiedliche $n-$Werte folgende Ergebnisse. Hierbei wurden die a_k^* , b_k^* nur bis maximal $k = 16$ ausgegeben. Der Fehler ist jeweils der (betragliche) Unterschied zwischen exaktem und berechnetem Wert. Für $n = 2048$ ist der Fehler in der Größenordnung $3 \cdot 10^{-7}$. Dies entspricht ungefähr dem theoretisch hergeleiteten Fehler von S.62.

Ergebnisse

Fourierkoeffizienten $(n = 8)$

k	a(k)	Fehler	b(k)	Fehler
0	3.141592653592	$3.6E-0012$	0.000000000000	$0.0E+0000$
1	−1.289728947597	$1.6E-0002$	−0.000000000001	$1.2E-0012$
2	−0.000000000001	$7.8E-0013$	−0.000000000001	$7.8E-0013$
3	−0.159034724761	$1.8E-0002$	−0.000000000001	$5.1E-0013$
4	0.000000000000	$0.0E+0000$	0.000000000000	$0.0E+0000$
5	−0.071003071366	$2.0E-0002$	−0.000000000001	$5.1E-0013$
6	−0.000000000000	$1.3E-0013$	0.000000000000	$1.3E-0013$
7	−0.051029583072	$2.5E-0002$	−0.000000000000	$2.6E-0013$
8	0.000000000000	$0.0E+0000$	0.000000000000	$0.0E+0000$

Fourierkoeffizienten $(n = 64)$

k	a(k)	Fehler	b(k)	Fehler
0	3.141592653596	$7.3E-0012$	0.000000000000	$0.0E+0000$
1	−1.273495239007	$2.6E-0004$	−0.000000000003	$3.4E-0012$
2	−0.000000000000	$2.3E-0013$	0.000000000000	$6.9E-0014$
3	−0.141727001446	$2.6E-0004$	−0.000000000001	$8.9E-0013$
4	−0.000000000000	$4.8E-0014$	−0.000000000000	$1.4E-0013$
5	−0.051186017148	$2.6E-0004$	−0.000000000000	$4.1E-0013$
6	0.000000000000	$2.1E-0013$	0.000000000000	$1.3E-0013$
7	−0.026241660367	$2.6E-0004$	−0.000000000001	$8.5E-0013$
8	−0.000000000000	$3.8E-0014$	0.000000000000	$4.6E-0013$

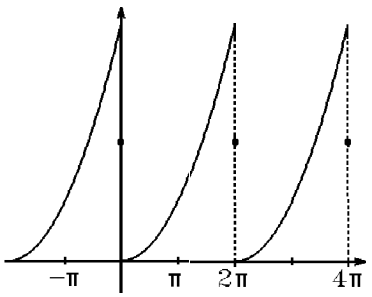
Fourierkoeffizienten $(n = 256)$

k	a(k)	Fehler	b(k)	Fehler
0	3.141592653596	$7.3E-0012$	0.000000000000	$0.0E+0000$
1	−1.273255523756	$1.6E-0005$	−0.000000000006	$6.5E-0012$
2	−0.000000000000	$2.2E-0013$	0.000000000000	$1.4E-0013$
3	−0.141487040567	$1.6E-0005$	−0.000000000015	$1.5E-0011$
4	0.000000000000	$4.2E-0014$	0.000000000000	$2.2E-0015$
5	−0.050945563762	$1.6E-0005$	−0.000000000011	$1.1E-0011$
6	−0.000000000000	$4.7E-0014$	−0.000000000000	$1.2E-0014$
7	−0.026000465368	$1.6E-0005$	−0.000000000011	$1.1E-0011$
8	0.000000000000	$4.3E-0014$	0.000000000000	$1.2E-0013$
9	−0.015734995442	$1.6E-0005$	−0.000000000010	$9.6E-0012$
10	0.000000000000	$6.9E-0014$	−0.000000000000	$1.4E-0014$
11	−0.010538634401	$1.6E-0005$	−0.000000000005	$5.3E-0012$
12	−0.000000000000	$5.1E-0014$	−0.000000000000	$3.5E-0014$
13	−0.007549961124	$1.6E-0005$	−0.000000000005	$4.8E-0012$
14	−0.000000000000	$1.2E-0014$	0.000000000000	$2.6E-0014$
15	−0.005674848497	$1.6E-0005$	−0.000000000007	$7.5E-0012$
16	−0.000000000000	$2.0E-0013$	−0.000000000000	$1.1E-0013$

Fourierkoeffizienten $(n = 2048)$

k	a(k)	Fehler	b(k)	Fehler
0	3.141592653596	$7.3E-0012$	0.000000000000	$0.0E+0000$
1	−1.273239795271	$2.5E-0007$	0.000000000020	$2.0E-0011$
2	−0.000000000000	$2.1E-0013$	0.000000000000	$1.2E-0013$
3	−0.141471310294	$2.5E-0007$	0.000000000059	$5.9E-0011$
4	0.000000000000	$4.7E-0014$	0.000000000000	$7.4E-0015$
5	−0.050929831495	$2.5E-0007$	0.000000000054	$5.4E-0011$
6	−0.000000000000	$2.6E-0014$	−0.000000000000	$1.1E-0014$
7	−0.025984730195	$2.5E-0007$	0.000000000066	$6.6E-0011$
8	−0.000000000000	$2.0E-0014$	0.000000000000	$1.1E-0013$
9	−0.015719256409	$2.5E-0007$	0.000000000063	$6.3E-0011$
10	0.000000000000	$7.6E-0014$	−0.000000000000	$1.1E-0014$
11	−0.010522890547	$2.5E-0007$	0.000000000027	$2.7E-0011$
12	−0.000000000000	$1.7E-0014$	−0.000000000000	$5.4E-0014$
13	−0.007534211484	$2.5E-0007$	0.000000000026	$2.6E-0011$
14	−0.000000000000	$8.6E-0015$	0.000000000000	$1.5E-0014$
15	−0.005659092102	$2.5E-0007$	0.000000000064	$6.4E-0011$
16	−0.000000000000	$1.9E-0014$	−0.000000000000	$1.7E-0013$

Beispiel 2.



$f(x) = x^2$, falls $0 < x < 2\pi$

$f(0) = f(2\pi) = 2\pi^2$.

Exakte Fourierkoeffizienten:

$a_0 = \frac{8\pi^2}{3}$, $a_k = \frac{4}{k^2}$, $b_k = -\frac{4\pi}{k}$.

Ergebnisse

Fourierkoeffizienten $(n = 8)$

k	a(k)	Fehler	b(k)	Fehler
0	26.370349259174	$5.1E-0002$	0.000000000000	$0.0E+0000$
1	4.051802986884	$5.2E-0002$	−12.404462994324	$1.6E-0001$
2	1.053029287543	$5.3E-0002$	−5.956833200078	$3.3E-0001$
3	0.499622322979	$5.5E-0002$	−3.692726705474	$5.0E-0001$
4	0.308425137533	$5.8E-0002$	−2.467401100261	$6.7E-0001$
5	0.223062727386	$6.3E-0002$	−1.648664706378	$8.6E-0001$
6	0.180671262597	$7.0E-0002$	−1.022030999538	$1.1E+0000$
7	0.160314163302	$7.9E-0002$	−0.490796594131	$1.3E+0000$
8	0.154212568785	$9.2E-0002$	0.000000000000	$1.6E+0000$

Fourierkoeffizienten $(n = 64)$

k	a(k)	Fehler	b(k)	Fehler
0	26.319748260081	$8.0E-0004$	0.000000000000	$0.0E+0000$
1	4.000803287257	$8.0E-0004$	−12.563847215788	$2.5E-0003$
2	1.000803577699	$8.0E-0004$	−6.278137901834	$5.0E-0003$
3	0.445248506576	$8.0E-0004$	−4.181217575584	$7.6E-0003$
4	0.250804741120	$8.0E-0004$	−3.131492973211	$1.0E-0002$
5	0.160805615440	$8.1E-0004$	−2.500644951608	$1.3E-0002$
6	0.111917797223	$8.1E-0004$	−2.079233385331	$1.5E-0002$
7	0.082440607428	$8.1E-0004$	−1.777497865560	$1.8E-0002$
8	0.063309421674	$8.1E-0004$	−1.550557874296	$2.0E-0002$
9	0.050193805778	$8.1E-0004$	−1.373479508533	$2.3E-0002$
10	0.040812960501	$8.1E-0004$	−1.231302169585	$2.5E-0002$
11	0.033872887406	$8.2E-0004$	−1.114505239490	$2.8E-0002$
12	0.028595096942	$8.2E-0004$	−1.016741418984	$3.0E-0002$
13	0.024488451263	$8.2E-0004$	−0.933616216615	$3.3E-0002$
14	0.021230681525	$8.2E-0004$	−0.861990513739	$3.6E-0002$
15	0.018603218334	$8.3E-0004$	−0.799562103079	$3.8E-0002$
16	0.016453582620	$8.3E-0004$	−0.744604150007	$4.1E-0002$

Fourierkoeffizienten $(n = 256)$

k	a(k)	Fehler	b(k)	Fehler
0	26.318995269045	$5.0E-0005$	0.000000000000	$0.0E+0000$
1	4.000050199349	$5.0E-0005$	-12.566212907215	$1.6E-0004$
2	1.000050200802	$5.0E-0005$	-6.282869891620	$3.2E-0004$
3	0.444494647149	$5.0E-0005$	-4.188317075597	$4.7E-0004$
4	0.250050205398	$5.0E-0005$	-3.140961803831	$6.3E-0004$
5	0.160050208822	$5.0E-0005$	-2.512485542829	$7.9E-0004$
6	0.111161324084	$5.0E-0005$	-2.093448780321	$9.5E-0004$
7	0.081682870927	$5.0E-0005$	-1.794091723572	$1.1E-0003$
8	0.062550223584	$5.0E-0005$	-1.569534475413	$1.3E-0003$
9	0.049432946105	$5.0E-0005$	-1.394843758138	$1.4E-0003$
10	0.040050237219	$5.0E-0005$	-1.255059604622	$1.6E-0003$
11	0.033108096398	$5.0E-0005$	-1.140662034550	$1.7E-0003$
12	0.027828031643	$5.0E-0005$	-1.045304393856	$1.9E-0003$
13	0.023718902392	$5.0E-0005$	-0.964592844143	$2.1E-0003$
14	0.020458436816	$5.0E-0005$	-0.895388928948	$2.2E-0003$
15	0.017828062297	$5.0E-0005$	-0.835391112681	$2.4E-0003$
16	0.015675296311	$5.0E-0005$	-0.782873243286	$2.5E-0003$

Fourierkoeffizienten $(n = 2048)$

k	a(k)	Fehler	b(k)	Fehler
0	26.318945854029	$7.8E-0007$	0.000000000000	$0.0E+0000$
1	4.000000787950	$7.9E-0007$	-12.566368158761	$2.5E-0006$
2	1.000000785973	$7.9E-0007$	-6.283180383412	$4.9E-0006$
3	0.444445229892	$7.9E-0007$	-4.188782815341	$7.4E-0006$
4	0.250000784988	$7.8E-0007$	-3.141582798977	$9.9E-0006$
5	0.160000784664	$7.8E-0007$	-2.513261803950	$1.2E-0005$
6	0.111111895999	$7.8E-0007$	-2.094380319017	$1.5E-0005$
7	0.081633438211	$7.9E-0007$	-1.795178554343	$1.7E-0005$
8	0.062500784619	$7.8E-0007$	-1.570776614351	$2.0E-0005$
9	0.049383500261	$7.8E-0007$	-1.396241225259	$2.2E-0005$
10	0.040000784498	$7.8E-0007$	-1.256612420742	$2.5E-0005$
11	0.033058635884	$7.8E-0007$	-1.142370223582	$2.7E-0005$
12	0.027778562387	$7.8E-0007$	-1.047167981838	$3.0E-0005$
13	0.023669423634	$7.8E-0007$	-0.966611859892	$3.2E-0005$
14	0.020408948017	$7.8E-0007$	-0.897563403271	$3.4E-0005$
15	0.017778562826	$7.9E-0007$	-0.837721079044	$3.7E-0005$
16	0.015625784508	$7.8E-0007$	-0.785358736861	$3.9E-0005$

Die Fehler in diesem Beispiel sind größer als in Beispiel 1., da die gegebene Funktion in diesem Beispiel im Gegensatz zur Funktion von Beispiel 1. *nicht stetig* ist, sondern *Sprungstellen* besitzt. Das entspricht den Bemerkungen zu den Fehlerabschätzungen von S.61/62.

Auswertung trigonometrischer Polynome

Der FFT-Algorithmus kann auch benutzt werden, um *trigonometrische Polynome* an diskreten Stellen auszuwerten:

Gegeben: Das *trigonometrische Polynom*

$$f(x) = \frac{a_0}{2} + \sum_{k=1}^{N-1} (a_k \cos kx + b_k \sin kx) + a_N \cos Nx \ .$$

Gesucht: $f(x_l)$ an den Stellen $x_l = \dfrac{2\pi l}{n}$, $(l = 0, 1, \ldots, n-1)$ mit $\underline{n = 2N}$.

Mit Hilfe der komplexen Darstellung von $\cos x$ und $\sin x$ erhalten wir

$$\begin{aligned} f(x_l) &= \frac{a_0}{2} + \sum_{k=1}^{N-1} \Big(\frac{a_k}{2} (\mathrm{e}^{ikx_l} + \mathrm{e}^{-ikx_l}) + \frac{b_k}{2i} (\mathrm{e}^{ikx_l} - \mathrm{e}^{-ikx_l}) \Big) + a_N \mathrm{e}^{iNx_l} \\ &\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad (\text{da } \sin Nx_l = 0) \\ &= \frac{a_0}{2} + \sum_{k=1}^{N-1} \Big(\frac{1}{2}(a_k - ib_k)\mathrm{e}^{ikx_l} + \frac{1}{2}(a_k + ib_k)\mathrm{e}^{-ikx_l} \Big) + a_N \mathrm{e}^{iNx_l} \\ &= \sum_{k=0}^{N} y_k \mathrm{e}^{ikx_l} + \sum_{k=1}^{N-1} y_{n-k} \mathrm{e}^{-ikx_l} \end{aligned}$$

mit

$$\boxed{\begin{aligned} &y_k = \frac{1}{2}(a_k - ib_k) \ , \quad y_{n-k} = \frac{1}{2}(a_k + ib_k) \quad , \quad (k = 0, 1, \ldots, N-1) \\ &y_N = a_N \ , \quad b_0 = 0 \end{aligned}}$$

Also erhalten wir insgesamt, da $\mathrm{e}^{inx_l} = 1$

$$\begin{aligned} f(x_l) &= \sum_{k=0}^{N} y_k \mathrm{e}^{ikx_l} + \sum_{k=1}^{N-1} y_{n-k} \mathrm{e}^{-ikx_l}\, \mathrm{e}^{inx_l} \\ &= \sum_{k=0}^{N} y_k \mathrm{e}^{ikx_l} + \sum_{k=1}^{N-1} y_{n-k} \mathrm{e}^{i(n-k)x_l} \\ &\qquad\qquad (n-k \to m) \\ &= \sum_{k=0}^{N} y_k \mathrm{e}^{ikx_l} + \sum_{m=N+1}^{2N-1} y_m \mathrm{e}^{imx_l} = \sum_{k=0}^{2N-1} y_k \mathrm{e}^{ikx_l} \end{aligned}$$

$$\boxed{\Rightarrow \quad f(x_l) = \sum_{k=0}^{n-1} y_k w_n^{kl} \quad \text{mit} \quad w_n = \mathrm{e}^{i2\pi/n} \ , \quad n = 2N}$$

Also erhalten wir für $f(x_l)$ eine diskrete Fourier-Transformation der Ordnung n. Diese kann mit Hilfe des FFT-Algorithmus ausgewertet werden:

Gegeben: a_k , $(k = 0, 1, \ldots, N)$; $b_0 = 0$; b_k , $(k = 1, \ldots, N-1)$

Berechne:

$y_k = \frac{1}{2}(a_k - ib_k)$, $(k = 0, 1, \ldots, N-1)$

$y_{n-k} = \frac{1}{2}(a_k + ib_k)$, $(k = 0, 1, \ldots, N-1)$

$y_N = a_N$, $n = 2N$

$w = \mathrm{e}^{i2\pi/n} = \cos\frac{2\pi}{n} + i\sin\frac{2\pi}{n}$

FFT-Algorithmus von S.69 oder S.70 benutzen, mit $n = 2^{\gamma-1}$.

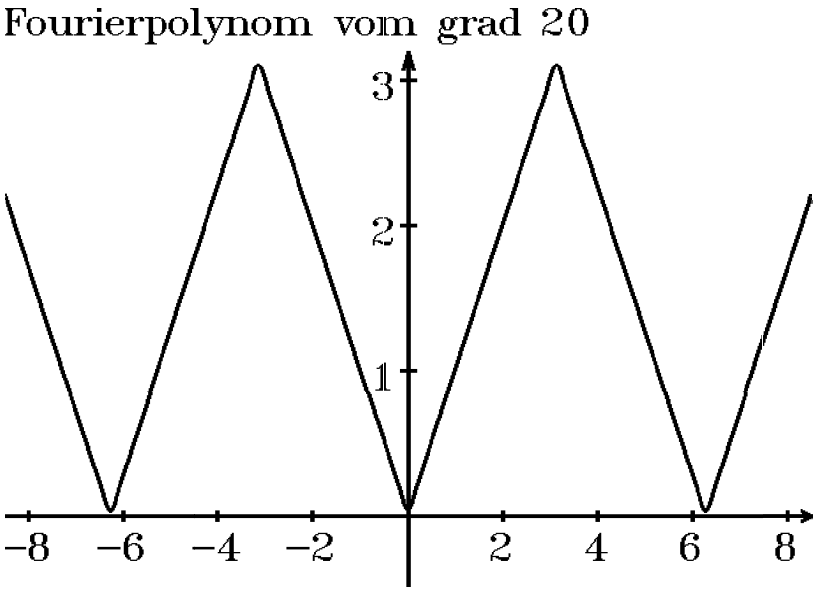
Die Werte y_k ergeben die Funktionswerte $f(x_k)$ für $x_k = \frac{2k\pi}{n}$, $k = 0, 1, \ldots, n-1$.
Es gilt $f(x_n) = f(x_0)$, da f $2\pi-$periodisch.

Beispiel (vgl. S.62)

Eingabe: $n = 256$

$$a_k = \begin{cases} 0 & \text{,falls } k \text{ gerade } (\neq 0) \\ -\frac{4}{\pi k^2} & \text{,falls } k \text{ ungerade} \end{cases} , \quad k \leq 20$$

$a_0 = \pi$, $a_k = 0$ $\forall k > 20$, $b_k = 0$ $\forall k \in I\!N$



VI Ausgleichsrechnung

Problemstellung

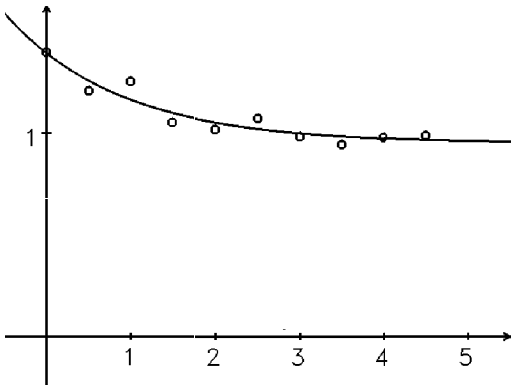
Gegeben:

N Meßpunkte (x_i, y_i) , $i = 1, 2, \ldots, N$.

Gesucht:

Eine von n Parametern $d_1, d_2, \ldots, d_n$ abhängige Funktion $f(x, \vec{d})$ mit

$$F(\vec{d}) = \sum_{i=1}^{N} \Big(f(x_i, \vec{d}) - y_i \Big)^2 \quad \textit{minimal}$$



d.h.: *minimal im quadratischen Mittel.*

Hierbei haben wir die gesuchten Parameter zu einem Vektor $\vec{d} = (d_1, \ldots, d_n)^T$ zusammengefaßt.

Beispiele

1. Lineare Probleme

$f(x, \vec{d}) = \sum_{j=1}^{n} d_j h_j(x)$, wobei $h_j(x)$ gegebene Funktionen sind.

In diesem Fall ist f *linear abhängig* von den gesuchten Parametern $d_1, \ldots, d_n$.

Beispiel hierzu:

$h_j(x) = x^{j-1} \quad \Rightarrow \quad f(x, \vec{d}) = \sum_{j=1}^{n} d_j x^{j-1} \quad \Rightarrow \quad f$ ist Polynom vom Grad $\leq n-1$.

2. Nichtlineare Probleme

f ist *nicht linear abhängig* von den gesuchten Parametern $d_1, \ldots, d_n$.

Beispiele hierzu:

$f(x, \vec{d}) = d_1 + d_2 \mathrm{e}^{d_3 x}$ (allgemeine Exponentialfunktion)

$f(x, \vec{d}) = d_1 \sin(d_2 x + d_3)$ (allgemeine Sinus-Schwingung)

$f(x, \vec{d}) = d_1 + \dfrac{d_2}{d_3 + x}$ (allgemeine Hyperbel-Funktion).

F ist in beiden Fällen eine Funktion von den unbekannten Parametern $d_1, \ldots, d_n$:

$F(\vec{d}) = \sum_{i=1}^{N} \Big(f(x_i, \vec{d}) - y_i \Big)^2$ minimal.

Notwendige Bedingung für das Minimum ist: $\dfrac{\partial F}{\partial d_i} = 0 \quad \forall i = 1, \ldots, n$.

Das ist ein Gleichungssystem für die gesuchten Parameter $d_1, \ldots, d_n$.

Wie wir gleich zeigen werden, erhalten wir im *linearen Fall* ein *lineares GLS*, im *nichtlinearen Fall* ein *nichtlineares GLS*.

1. Lineare Probleme

Gegeben: $h_j(x)$, $j = 1, \ldots, n$.

Gesucht: $\vec{d} = (d_1, \ldots, d_n)^T$ mit

$$F(\vec{d}) = \sum_{i=1}^{N} \Big(f(x_i, \vec{d}) - y_i \Big)^2 = \sum_{i=1}^{N} \Big(\sum_{j=1}^{n} d_j h_j(x_i) - y_i \Big)^2 \quad \text{minimal.}$$

Mit

$$\boxed{c_{ij} = h_j(x_i)}$$

erhalten wir die *Fehlergleichungen*

$$r_i = f(x_i, \vec{d}) - y_i = \sum_{j=1}^{n} d_j h_j(x_i) - y_i = \sum_{j=1}^{n} c_{ij} d_j - y_i = (C\vec{d} - \vec{y})_i \quad , \quad i = 1, 2, \ldots, N \ ,$$

mit der $(N, n)-$Matrix $C = \Big(\ c_{ij} \ \Big)$ und den Vektoren $\vec{d} = (d_1, \ldots, d_n)^T$ und $\vec{y} = (y_1, \ldots, y_N)^T$.

Für den *Fehlervektor* $\vec{r} = (r_1, \ldots, r_N)^T$ erhalten wir dann

$$\boxed{\vec{r} = C\vec{d} - \vec{y}}$$

Die Fehlerfunktion $F(\vec{d})$ lautet damit

$$F(\vec{d}) = \sum_{i=1}^{N} (f(x_i, \vec{d}) - y_i)^2 = \sum_{i=1}^{N} r_i^2 = <\vec{r}, \vec{r}> = <C\vec{d} - \vec{y}, C\vec{d} - \vec{y}>$$

$$= <C\vec{d}, C\vec{d}> - 2 <C\vec{d}, \vec{y}> + <\vec{y}, \vec{y}> \qquad \text{(Linearität des Skalarprodukts)}$$

$$= <C^T C\vec{d}, \vec{d}> - 2 <C^T\vec{y}, \vec{d}> + |\vec{y}|^2 \qquad \text{(nach Hilfsatz 1.3, S.8).}$$

Mit

$$\boxed{A = C^T C \quad , \quad \vec{b} = C^T \vec{y}}$$

erhalten wir für die Fehlerfunktion F :

$$\boxed{F(\vec{d}) = <A\vec{d}, \vec{d}> - 2 <\vec{b}, \vec{d}> + |\vec{y}|^2}$$

$A = C^T C$ ist eine *symmetrische* $(n, n)-$Matrix , denn:
$A^T = (C^T C)^T = C^T C^{TT} = C^T C = A$.

A ist *positiv definit*, falls rang $C = n$ ist, was wir ab jetzt voraussetzen wollen, denn:

$$< A\vec{d}\,,\,\vec{d} >=< C^T C\vec{d}\,,\,\vec{d} >=< C\vec{d}\,,\,C\vec{d} >= |C\vec{d}|^2 \geq 0 \quad , \quad = 0 \quad \Leftrightarrow \quad \vec{d} = \vec{0}$$
(falls rang $C = n$).

Damit ist $< A\vec{d}\,,\,\vec{d} >$ eine positiv definite quadratische Form (falls rang $C = n$).

Für die partiellen Ableitungen $\dfrac{\partial F}{\partial d_i}$ erhalten wir

$$\frac{\partial F}{\partial d_i} = \frac{\partial}{\partial d_i}\Big(< A\vec{d}\,,\,\vec{d} > -2 < \vec{b}\,,\,\vec{d} >\Big)$$
$$=< A\vec{e}_i\,,\,\vec{d} > + < A\vec{d}\,,\,\vec{e}_i > -2 < \vec{b}\,,\,\vec{e}_i > \qquad \text{(nach der Produktregel)},$$
$$\text{da} \quad \frac{\partial \vec{d}}{\partial d_i} = \frac{\partial}{\partial d_i}\begin{pmatrix} d_1 \\ \vdots \\ d_i \\ \vdots \\ d_n \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{pmatrix} = \vec{e}_i \ .$$

Da $< A\vec{e}_i\,,\,\vec{d} >=< \vec{e}_i\,,\,A^T\vec{d} >=< \vec{e}_i\,,\,A\vec{d} >=< A\vec{d}\,,\,\vec{e}_i >$ (nach Hilfssatz 1.3, und weil A symmetrisch) , folgt

$$\frac{\partial F}{\partial d_i} = 2 < A\vec{d}\,,\,\vec{e}_i > -2 < \vec{b}\,,\,\vec{e}_i >= 2 < A\vec{d} - \vec{b}\,,\,\vec{e}_i >= 2(A\vec{d} - \vec{b})_i = 0 \quad \forall i = 1, \ldots, n \ .$$

Also gilt:

$$\boxed{\frac{\partial F}{\partial d_i} = 0 \quad \forall i = 1, \ldots, n \quad \Leftrightarrow \quad A\vec{d} = \vec{b}}$$

mit

$$\boxed{A = C^T C \quad , \quad \vec{b} = C^T \vec{y} \quad , \quad C = \Big(\, c_{ij} \,\Big) = \Big(\, h_j(x_i) \,\Big)}$$

Hinreichendes Kriterium:

$$\Big(\frac{\partial^2 F}{\partial d_i \partial d_j}\Big) = \Big(\, 2a_{ij} \,\Big) = 2A \ .$$

Da A positiv definit (falls rang $C = n$) $\Rightarrow$ Minimum.

Also liefert der Lösungsvektor $\vec{d}$ des linearen GLS $A\vec{d} = \vec{b}$ das Mimimum der Fehlerfunktion $F(\vec{d})$ und damit die gesuchten Parameter $d_1, \ldots, d_n$.

Da die Matrix A *symmetrisch* und *positiv definit* ist, läßt sich das lineare GLS $A\vec{d} = \vec{b}$ mit Hilfe des Cholesky-Algorithmus berechnen.

Beispiel

Gegeben: N Meßpunkte (x_i, y_i) , $i = 1, 2, \ldots, N$.

Gesucht: Polynom vom Grad ≤ 2, also $f(x, \vec{d}) = d_1 + d_2 x + d_3 x^2$

$\Rightarrow \quad h_j(x) = x^{j-1}$, also: $h_1(x) = 1$, $h_2(x) = x$, $h_3(x) = x^2$

$\Rightarrow \quad c_{ij} = h_j(x_i) = x_i^{j-1}$, $i = 1, \ldots, N$, $j = 1, 2, 3$.

Damit erhalten wir

$$C = \begin{pmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ \vdots & \vdots & \vdots \\ 1 & x_N & x_N^2 \end{pmatrix} \quad , \quad \vec{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{pmatrix}$$

$$A = C^T C = \begin{pmatrix} 1 & 1 & \ldots & 1 \\ x_1 & x_2 & \ldots & x_N \\ x_1^2 & x_2^2 & \ldots & x_N^2 \end{pmatrix} \begin{pmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ \vdots & \vdots & \vdots \\ 1 & x_N & x_N^2 \end{pmatrix} = \begin{pmatrix} N & \sum x_i & \sum x_i^2 \\ \sum x_i & \sum x_i^2 & \sum x_i^3 \\ \sum x_i^2 & \sum x_i^3 & \sum x_i^4 \end{pmatrix}$$

$$\vec{b} = C^T \vec{y} = \begin{pmatrix} 1 & 1 & \ldots & 1 \\ x_1 & x_2 & \ldots & x_N \\ x_1^2 & x_2^2 & \ldots & x_N^2 \end{pmatrix} \begin{pmatrix} y_1 \\ \vdots \\ y_N \end{pmatrix} = \begin{pmatrix} \sum y_i \\ \sum x_i y_i \\ \sum x_i^2 y_i \end{pmatrix}$$

wobei $\sum$ abkürzend für $\displaystyle\sum_{i=1}^{N}$ steht.

$A\vec{d} = \vec{b}$ lösen $\Rightarrow \quad d_1, d_2, d_3$ gesuchte Koeffizienten des Polynoms
$f(x) = d_1 + d_2 x + d_3 x^2$ (beste Approximation im quadratischen Mittel an die gegebenen Meßpunkte (x_i, y_i) , $i = 1, \ldots, N$).

Beispiel hierzu

Meßpunkte

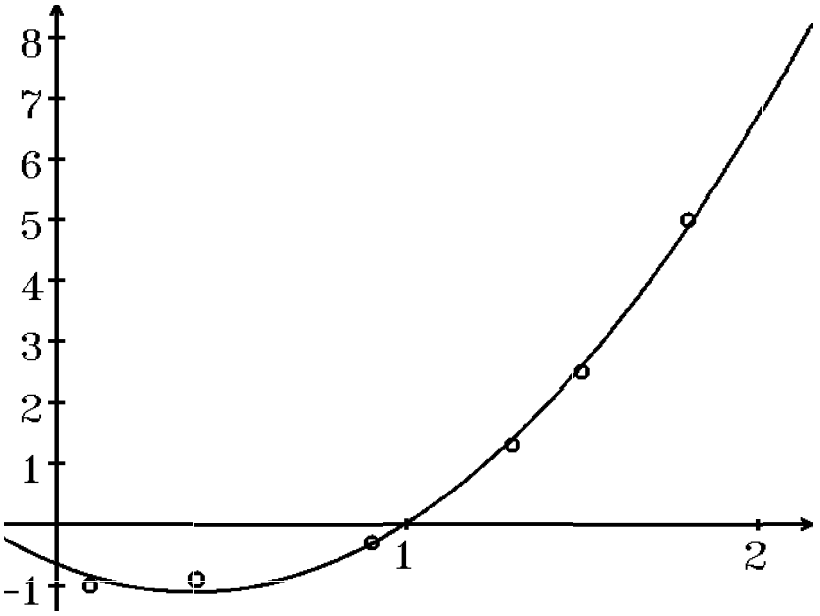
x_i	0.1	0.4	0.9	1.3	1.5	1.8
y_i	-1	-0.9	-0.3	1.3	2.5	5

Gesucht: $f(x) = d_1 + d_2 x + d_3 x^2$ (Polynom bester Approximation im quadratischen Mittel).

Für die Matrix A erhält man: $A = \begin{pmatrix} 6 & 6 & 8.16 \\ 6 & 8.16 & 12.198 \\ 8.16 & 12.198 & 19.098 \end{pmatrix}$.

Bei einfach genauer Rechnung (REAL-Variable in TURBO-PASCAL) erhält man das Ergebnis:

$d_1 = -0.660513009$, $d_2 = -2.327016054$, $d_3 = 3.005536076$.



Problematik:

Die Matrix A ist *schlecht konditioniert*, denn die EW von A sind: $\lambda_1 = 31.08$, $\lambda_2 = 2.09$, $\lambda_3 = 0.09 \quad \Rightarrow \quad k(A) = \dfrac{31.08}{0.09} = 345.3 \approx 3 \cdot 10^2$.

Man sollte also das GLS $A\vec{d} = \vec{b}$ mit *möglichst großer Rechengenauigkeit* lösen.

Dieses Problem der *schlechten Kondition* von A tritt bei fast allen Ausgleichsrechnungen auf. Also sollte man grundsätzlich hier mit *möglichst großer Rechengenauigkeit* rechnen (z.B. extended REAL-Variable).

2. Nichtlineare Probleme

Ist die Abhängigkeit der Funktion f von mindestens einem gesuchten Parameter d_i *nicht-linear*, so erhält man bei den partiellen Ableitungen $\dfrac{\partial F}{\partial d_i} = 0$ *nichtlineare Gleichungen.*

Beispiel

$f(x, \vec{d}) = d_1 + d_2 \mathrm{e}^{d_3 x}$

$$\Rightarrow \quad F(\vec{d}) = \sum_{i=1}^{N} \Big(d_1 + d_2 \mathrm{e}^{d_3 x_i} - y_i \Big)^2$$

$$\Rightarrow \quad \frac{\partial F}{\partial d_2} = 2 \sum_{i=1}^{N} \Big(d_1 + d_2 \mathrm{e}^{d_3 x_i} - y_i \Big) \mathrm{e}^{d_3 x_i} = 0 \quad \Rightarrow \text{ nichtlineare Gleichung.}$$

Also erhalten wir allgemein im nichtlinearen Fall ein *nichtlineares GLS*

$$\frac{\partial F}{\partial d_i} = 0 \quad , \quad i = 1, \ldots, n \ .$$

Dieses nichtlineare GLS müßte mit Hilfe des *Newton-Verfahrens* näherungsweise gelöst werden. Beim Newton-Verfahren wird mit Hilfe der Taylorentwicklung bis zur Ordnung 1 das nichtlineare GLS *linearisiert*.
Einfacher ist es hier, direkt die Koordinaten des *Fehlervektors*
$r_i = f(x_i, \vec{d}) - y_i \ , \ i = 1, \ldots, N$, mit Hilfe der Taylorentwicklung zu *linearisieren*.

Mit

$$\boxed{g_i(\vec{d}) := f(x_i, \vec{d}) \ , \ (i = 1, \ldots, N)}$$

lauten die Koordinaten des Fehlervektors: $r_i = g_i(\vec{d}) - y_i \ , \ (i = 1, \ldots, N)$.

Mit $\vec{g}(\vec{d}) = \begin{pmatrix} g_1(\vec{d}) \\ \vdots \\ g_N(\vec{d}) \end{pmatrix}$ erhalten wir den Fehlervektor

$$\boxed{\vec{r} = \vec{g}(\vec{d}) - \vec{y}}$$

Sei $\vec{d}^{(0)}$ eine Anfangsnäherung, so gilt mit dem Korrekturvektor $\vec{k} = \vec{d} - \vec{d}^{(0)}$:

$$g_i(\vec{d}) \approx g_i(\vec{d}^{(0)}) + \sum_{j=1}^{n} \frac{\partial g_i(\vec{d}^{(0)})}{\partial d_j} \cdot k_j \quad , \quad (i = 1, \ldots, N)$$

(*Taylorentwicklung* um $\vec{d}^{(0)}$ bis zur Ornung 1).

Damit lauten die *linearisierten* Koordinaten des *Fehlervektors*

$$r_i \approx \sum_{j=1}^{n} \frac{\partial g_i(\vec{d}^{(0)})}{\partial d_j} \cdot k_j + g_i(\vec{d}^{(0)}) - y_i \quad , \quad (i = 1, \ldots, N).$$

Mit

$$\boxed{c_{ij}^{(0)} = \frac{\partial g_i(\vec{d}^{(0)})}{\partial d_j} \quad \text{und} \quad z_i^{(0)} = y_i - g_i(\vec{d}^{(0)})}$$

gilt also mit der Matrix $C^{(0)} = \left(c_{ij}^{(0)} \right)$ und dem Vektor $\vec{z}^{(0)} = \left(z_i^{(0)} \right)$

$$\boxed{\vec{r} = C^{(0)}\vec{k} - \vec{z}^{(0)}}$$

Analog zum *linearen Fall* (vgl. S.78/79) führt dies, wenn die Fehlerfunktion $F(\vec{k})$ minimal werden soll, auf das lineare GLS

$$\boxed{A^{(0)}\vec{k} = \vec{b}^{(0)} \quad \text{mit} \quad A^{(0)} = {C^{(0)}}^T C^{(0)} \quad \text{und} \quad \vec{b}^{(0)} = {C^{(0)}}^T \vec{z}^{(0)}}$$

Als Lösung dieses GLS $A^{(0)}\vec{k} = \vec{b}^{(0)}$ erhält man den Korrekturvektor $\vec{k}$.
Dann wählen wir als nächste Näherung:

$$\vec{d}^{(1)} = \vec{d}^{(0)} + \alpha\vec{k} \quad , \quad \alpha = 1, 1/2, 1/4, \ldots$$
$$\text{bis} \quad F(\vec{d}^{(1)}) < (1 - \frac{\alpha}{4})F(\vec{d}^{(0)})$$

α heißt *Dämpfungsparameter* (vgl. *gedämpftes Newton-Verfahren*).
Wird α zu klein, so muß das Verfahren beendet werden.

Algorithmus

Wähle Ausgangsnäherung $\vec{d}$.

Berechne für $l = 1, 2, \ldots$

$c_{ij} = \dfrac{\partial g_i(\vec{d})}{\partial d_j} \quad , \quad (i = 1, \ldots, N \ , \ j = 1, \ldots, n)$

$z_i = y_i - g_i(\vec{d}) \quad , \quad (i = 1, \ldots, N)$

$A = C^T C \quad , \quad \vec{b} = C^T \vec{z}$

GLS $A\vec{k} = \vec{b}$ lösen (mit Gauß-Algorithmus)

Setze $\alpha = 1, 1/2, 1/4, \ldots$

bis $F(\vec{d} + \alpha\vec{k}) < (1 - \dfrac{\alpha}{4})F(\vec{d})$ (oder $\alpha < \epsilon$)

$h = |F(\vec{d} + \alpha\vec{k}) - F(\vec{d})|$

$\vec{d} = \vec{d} + \alpha\vec{k}$

bis $h < \delta$ oder $\alpha < \epsilon$ oder l zu groß (keine Konvergenz).

Beispiel 1.

Meßpunkte: $(x_i, y_i) \quad , \quad i = 1, \ldots, N.$

Gesucht: $f(x, \vec{d}) = d_1 + d_2 \mathrm{e}^{d_3 x}$.

Definiere folgende Funktionen (mit globalen Variablen d_1, d_2, d_3)

$g(x) = d_1 + d_2 \mathrm{e}^{d_3 x}$

$g1(x) = \dfrac{\partial g}{\partial d_1}(x) \quad , \quad g2(x) = \dfrac{\partial g}{\partial d_2}(x) \quad , \quad g3(x) = \dfrac{\partial g}{\partial d_3}(x)$

also in diesem Beispiel:

$g1(x) = 1 \quad , \quad g2(x) = \mathrm{e}^{d_3 x} \quad , \quad g3(x) = x d_2 \mathrm{e}^{d_3 x}$.

Dann erhält man

$$C = \begin{pmatrix} g1(x_1) & g2(x_1) & g3(x_1) \\ g1(x_2) & g2(x_2) & g3(x_2) \\ \vdots & \vdots & \vdots \\ g1(x_N) & g2(x_N) & g3(x_N) \end{pmatrix} \quad , \quad C^T = \begin{pmatrix} g1(x_1) & g1(x_2) & \ldots & g1(x_N) \\ g2(x_1) & g2(x_2) & \ldots & g2(x_N) \\ g3(x_1) & g3(x_2) & \ldots & g3(x_N) \end{pmatrix}$$

Berechne für $l = 1, 2, \ldots$

$$A = C^T C = \begin{pmatrix} \sum g1(x_i)g1(x_i) & \sum g1(x_i)g2(x_i) & \sum g1(x_i)g3(x_i) \\ & \sum g2(x_i)g2(x_i) & \sum g2(x_i)g3(x_i) \\ \text{symmetrisch} & & \sum g3(x_i)g3(x_i) \end{pmatrix}$$

$$\vec{b} = C^T \vec{z} = C^T(\vec{y} - \vec{g}(\vec{d})) = \begin{pmatrix} \sum g1(x_i)(y_i - g(x_i)) \\ \sum g2(x_i)(y_i - g(x_i)) \\ \sum g3(x_i)(y_i - g(x_i)) \end{pmatrix}$$

herbei ist $\sum$ eine Abkürzung für $\sum_{i=1}^{N}$.

Löse das lineare GLS $A\vec{k} = \vec{b} \quad \Rightarrow \quad$ *Korrekturvektor* $\vec{k}$.

Wähle $\alpha = 1, 1/2, 1/4, \ldots$

bis $F(\vec{d} + \alpha\vec{k}) < (1 - \alpha/4)F(\vec{d})$ (beenden, falls $\alpha < \epsilon$)

mit $F(\vec{d}) = \sum_{i=1}^{N}(g(x_i) - y_i)^2$

Berechne $h = |F(\vec{d} + \alpha\vec{k}) - F(\vec{d})|$

Setze $\vec{d} := \vec{d} + \alpha\vec{k}$

bis $h < \delta$ oder $\alpha < \epsilon$ oder l zu groß (keine Konvergenz).

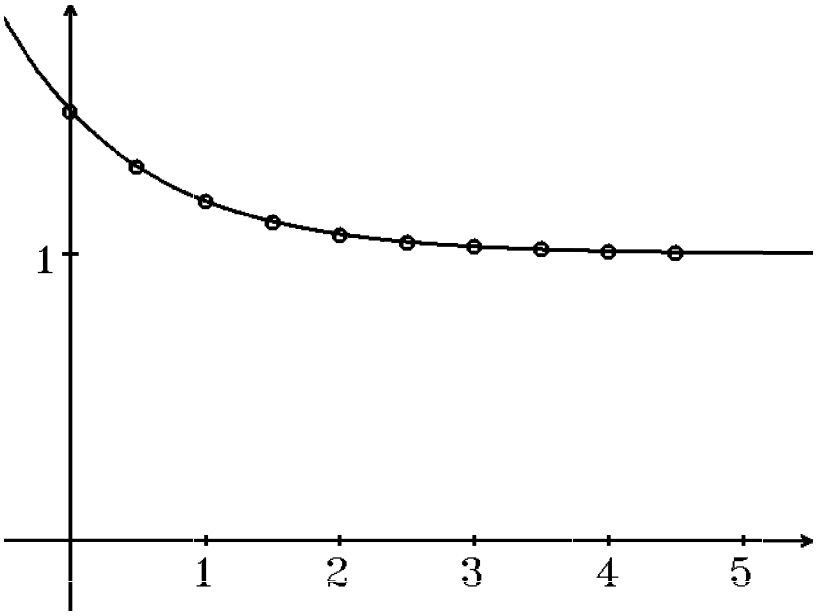
Beispiel hierzu

Vorgegeben: $x_i = (i-1) * 0.5$, $y_i = f(x_i) = 1 + 0.5\mathrm{e}^{-x_i}$, $(i = 1, \ldots, N)$.

Die Punkte liegen also alle *exakt* auf dem Graphen der e−Funktion mit $f(x) = 1 + 0.5\mathrm{e}^{-x}$, also $d_1 = 1$, $d_2 = 0.5$, $d_3 = -1$.

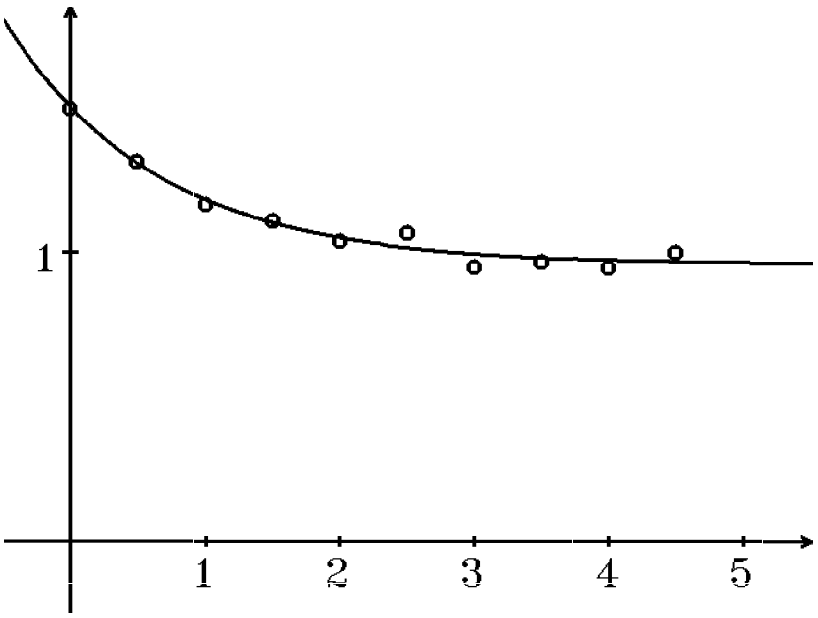
Für $N = 10$ erhält man im Falle geeigneter Startwerte nach l Schritten die gesuchten Werte $d_1 = 1$, $d_2 = 0.5$, $d_3 = -1$. Bei ungeeigneten Startwerten erhält man keine Konvergenz:

Startwerte			Konvergenz nach l Schritten
d_1	d_2	d_3	l
2	1	–2	5
3	5	–4	5
1	1	1	keine Konvergenz



Ausgleichsrechnung für e−*Funktion*

Im folgenden Bild wurden die y_i-Werte mit Hilfe eines Zufallszahlengenerators leicht gestört:



Ausgleichsrechnung für e−*Funktion*

Beispiel 2.

Gesucht: $f(x) = d_1 \sin(d_2 x + d_3)$.

Definiere folgende Funktionen (mit globalen Variablen d_1, d_2, d_3)
$g(x) = d_1 \sin(d_2 x + d_3)$

$$g1(x) = \frac{\partial g}{\partial d_1}(x) = \sin(d_2 x + d_3)$$
$$g2(x) = \frac{\partial g}{\partial d_2}(x) = x d_1 \cos(d_2 x + d_3)$$
$$g3(x) = \frac{\partial g}{\partial d_3}(x) = d_1 \cos(d_2 x + d_3)\ .$$

Dann kann der gleiche Algorithmus wie in Beispiel 1. benutzt werden.

Beispiel hierzu

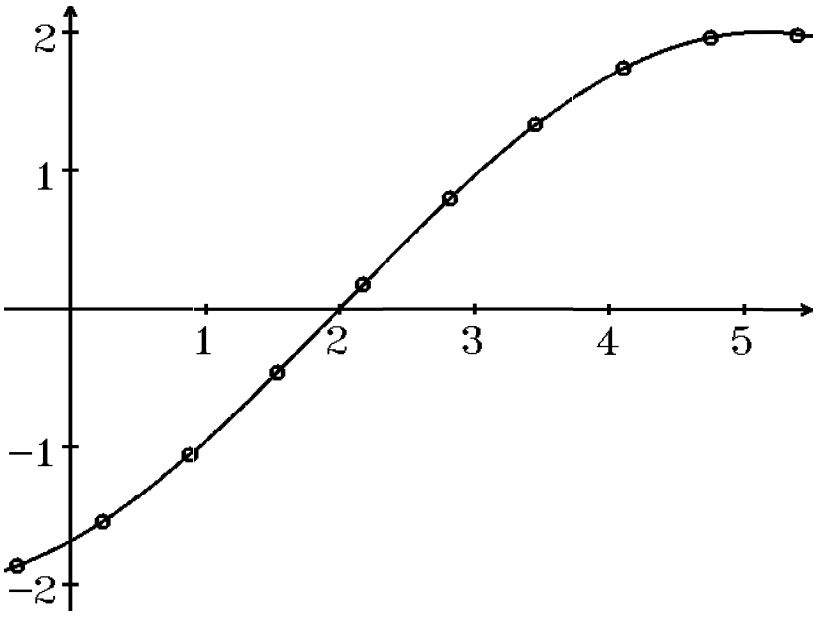
Vorgegeben: $x_i = a + (i-1) * (b-a)/(N-1)$, $y_i = f(x_i) = 2\sin(0.5x_i - 1)$,
$(i = 1, \ldots, N)$, $(a, b$ Intervallgrenzen).

Die Punkte liegen also alle *exakt* auf dem Graphen der sin −Funktion mit
$f(x) = 2\sin(0.5x - 1)$, also $d_1 = 2$, $d_2 = 0.5$, $d_3 = -1$.

Für $N = 10$ erhält man im Falle geeigneter Startwerte nach l Schritten die folgenden Werte d_1 , d_2 , d_3:

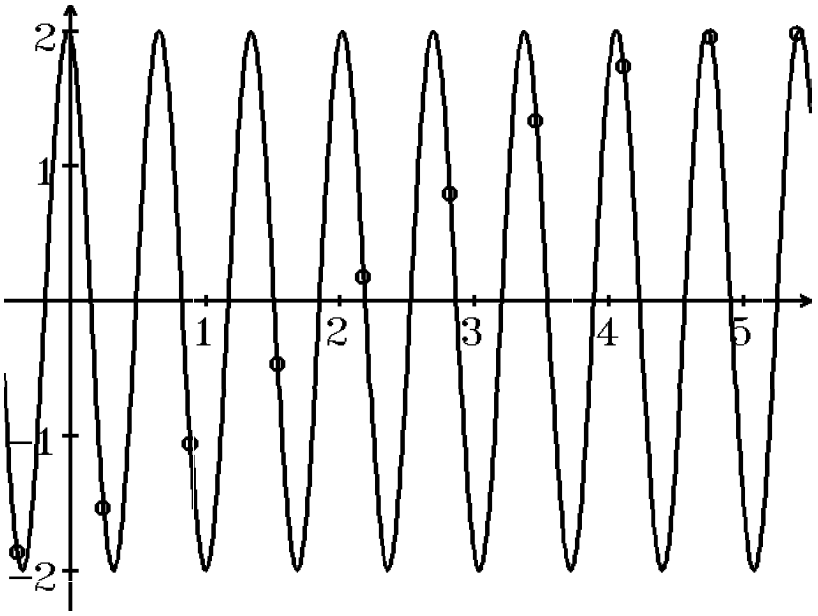
Startwerte			Konvergenz nach l Schritten			
d_1	d_2	d_3	l	d_1	d_2	d_3
1	1	–2	6	2	0.5	-1
3	1	0	7	-2	0.5	$\pi - 1$
1	9	–2	9	2	8.924	-3.712

Die Werte in der zweiten Zeile ergeben die gleiche sin −Funktion, denn:
$-2\sin(0.5x + (\pi - 1)) = 2\sin(0.5x - 1)$.



Ausgleichsrechnung für sin −*Funktion*

Wird der Startwert für die Frequenz, also d_2 , ungünstig gewählt (vgl. 3. Zeile in obiger Tabelle auf S.86), so kann als Ergebnis eine andere (als die vorgegebene) Frequenz auftreten, wie das folgende Bild zeigt:



Ausgleichsrechnung für $\sin -$*Funktion für die Startwerte* $d_1 = 1\ ,\ d_2 = 9\ ,\ d_3 = -2$

Beispiel 3.

Gesucht: $f(x) = d_1 + \dfrac{d_2}{x + d_3}$.

Definiere folgende Funktionen (mit globalen Variablen d_1, d_2, d_3)

$$g(x) = d_1 + \frac{d_2}{x + d_3}$$

$$g1(x) = \frac{\partial g}{\partial d_1}(x) = 1$$
$$g2(x) = \frac{\partial g}{\partial d_2}(x) = \frac{1}{x + d_3}$$
$$g3(x) = \frac{\partial g}{\partial d_3}(x) = -\frac{d_2}{(x + d_3)^2}\ .$$

Dann kann wieder der gleiche Algorithmus wie in Beispiel 1. benutzt werden.

Beispiel 4. *Ausgleichsrechnung für den Kreis*

Gegeben: N Meßpunkte $(x_i.y_i)$, $(i = 1, \ldots, N)$.

Gesucht: Parameter x_0 , y_0 , r mit

$$F = \sum_{i=1}^{N} \Big((x_i - x_0)^2 + (y_i - y_0)^2 - r^2 \Big)^2 \quad \text{minimal.}$$

Dies ist ein *nichtlineares* Ausgleichsproblem. Durch geschickte Wahl anderer Parameter, aus denen sich dann die gesuchten Werte x_0, y_0, r berechnen lassen, kann dieses *nichtlineare* Ausgleichsproblem in ein *lineares* Ausgleichsproblem übergeführt werden:

Es gilt

$$\begin{aligned}(x_i - x_0)^2 + (y_i - y_0)^2 - r^2 &= x_i^2 - 2x_0x_i + x_0^2 + y_i^2 - 2y_0y_i + y_0^2 - r^2 \\ &= (x_i^2 + y_i^2) - (r^2 - (x_0^2 + y_0^2)) - 2x_0x_i - 2y_0y_i \ .\end{aligned}$$

Also folgt mit $d_1 = r^2 - (x_0^2 + y_0^2)$, $d_2 = 2x_0$, $d_3 = 2y_0$

$$F(d_1, d_2, d_3) = \sum_{i=1}^{N} \Big(d_1 + d_2x_i + d_3y_i - (x_i^2 + y_i^2) \Big)^2 \quad \text{minimal.}$$

Damit haben wir ein *lineares* Ausgleichsproblem erhalten mit

$h_1(x, y) = 1$, $h_2(x, y) = x$, $h_3(x, y) = y$.

Das ergibt mit $c_{ij} = h_j(x_i, y_i)$ folgende Matrizen

$$C = \begin{pmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ \vdots & \vdots & \vdots \\ 1 & x_N & y_N \end{pmatrix} \quad , \quad C^T = \begin{pmatrix} 1 & 1 & \ldots & 1 \\ x_1 & x_2 & \ldots & x_N \\ y_1 & y_2 & \ldots & y_N \end{pmatrix}$$

$$A = C^TC = \begin{pmatrix} N & \sum x_i & \sum y_i \\ \sum x_i & \sum x_i^2 & \sum x_iy_i \\ \sum y_i & \sum x_iy_i & \sum y_i^2 \end{pmatrix}$$

$$\vec{b} = C^T \begin{pmatrix} x_1^2 + y_1^2 \\ x_2^2 + y_2^2 \\ \vdots \\ x_N^2 + y_N^2 \end{pmatrix} = \begin{pmatrix} \sum (x_i^2 + y_i^2) \\ \sum x_i(x_i^2 + y_i^2) \\ \sum y_i(x_i^2 + y_i^2) \end{pmatrix} .$$

Die Lösung des linearen GLS $A\vec{d} = \vec{b}$ ergibt d_1, d_2, d_3 und damit

$$x_0 = \frac{d_2}{2} \quad , \quad y_0 = \frac{d_3}{2} \quad , \quad r = \sqrt{d_1 + (x_0^2 + y_0^2)} \ .$$

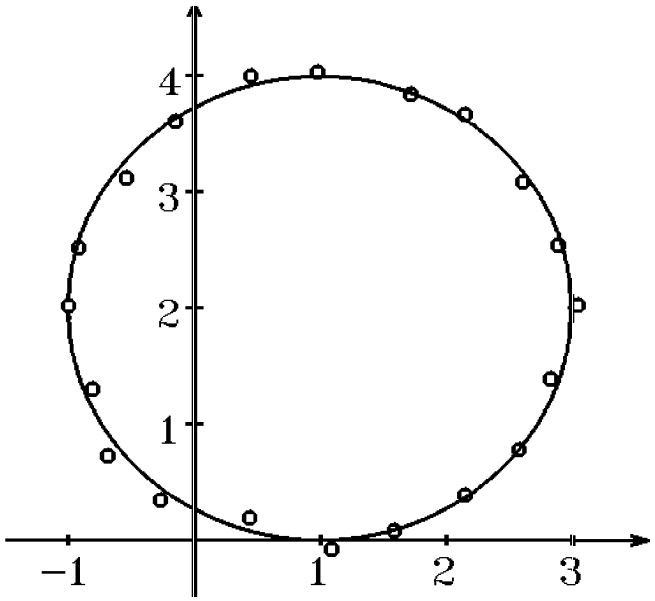
Beispiel hierzu

Vorgegeben: N Punkte auf dem Kreis $(x-1)^2 + (y-2)^2 = 2^2$, also
$x_i = 1 + 2\cos t_i$, $y_i = 2 + 2\sin t_i$, $t_i = 2\pi * (i-1)/N$, $(i = 1, \ldots, N)$.

Ergebnis: $x_0 = 1$, $y_0 = 2$, $r = 2$.

Im folgenden Bild wurden diese Punkte zusätzlich mit Hilfe eines Zufallszahlengenerators leicht gestört.

Ergebnis: $x_0 = 0.992$, $y_0 = 1.992$, $r = 1.995$.



Ausgleichsrechnung für den Kreis

VII Gradientenverfahren , Konjugierte Gradientenverfahren

Problemstellung

Gegeben: $f : D \subset \mathbb{R}^n \to \mathbb{R}$, f stetig differenzierbar in D .

Gesucht: $\min\limits_{\vec{x} \in D} f(\vec{x})$.

Notwendige Bedingung: $\operatorname{grad} f(\vec{x}) = \vec{0}$.

1. Möglichkeit:

Newton-Verfahren zur Lösung des GLS $\operatorname{grad} f(\vec{x}) = \vec{0}$ (vgl. S.35).

Die Funktionalmatrix lautet dann $\left(\dfrac{\partial^2 f}{\partial x_i \partial x_j}\right)_{1 \le i,j \le n}$.

Bemerkung: Diese 1. Möglichkeit ist nur sinnvoll bei relativ kleinen Optimierungsproblemen.

2. Möglichkeit: **Gradientenverfahren**

Wir benutzen i.f. die $2-$Norm $\|\vec{x}\|_2$ und schreiben hierfür kurz $\|\vec{x}\|$.

Die Richtungsableitung von f an der Stelle $\vec{x}$ in Richtung $\vec{a}$ mit $\|\vec{a}\| = 1$ lautet
$\dfrac{\partial f}{\partial \vec{a}}(\vec{x}) =< \operatorname{grad} f(\vec{x}), \vec{a} >= \|\operatorname{grad} f(\vec{x})\| \, \|\vec{a}\| \cos\alpha = \|\operatorname{grad} f(\vec{x})\| \cos\alpha$ (da $\|\vec{a}\| = 1$).
Die Richtungsableitung wird maximal $\Leftrightarrow$ $\cos\alpha = 1$ $\Leftrightarrow$ $\alpha = 2k\pi$, $(k \in \mathbb{Z})$
$\Leftrightarrow$ $\vec{a} = \lambda \operatorname{grad} f(\vec{x})$, $(\lambda \in \mathbb{R})$.

Also gilt: $\operatorname{grad} f(\vec{x})$ zeigt in die Richtung des stärksten Anstiegs
$\Rightarrow$ $-\operatorname{grad} f(\vec{x})$ zeigt in die Richtung des stärksten Gefälles.

Diese Aussage führt zu folgendem *Gradientenverfahren*:

Gradientenverfahren

Wähle Startvektor $\vec{x}_0$.
Für $k = 0, 1, 2, \ldots$ berechne
$\vec{d}_k := -\operatorname{grad} f(\vec{x}_k)$ (*Bestimmung der Richtung*)
Bestimme α_k mit $\min\limits_{\alpha_K} f(\vec{x}_k + \alpha_k \vec{d}_k)$ (*Liniensuche*)
$\vec{x}_{k+1} := \vec{x}_k + \alpha_k \vec{d}_k$
bis $\|\vec{d}_k\| < \epsilon$ oder k zu groß (keine Konvergenz).

Die *Liniensuche* ist im allgemeinen Fall nicht exakt durchzuführen. Man halbiert deshalb den Faktor α (ähnlich wie beim gedämpften Newton-Verfahren (vgl. S.33)) bis man einen besseren Wert $\vec{x}_{k+1}$ gefunden hat:

Gradientenverfahren (mit Halbierung)

Wähle Startvektor $\vec{x}_0$ und Schrittweite α_0 .

Für $k = 0, 1, 2, \ldots$ berechne

$\vec{d}_k := -\text{grad}\, f(\vec{x}_k)$ (*Bestimmung der Richtung*)

Setze $\alpha = \alpha_0, \alpha_0/2, \alpha_0/4, \ldots$ bis $f(\vec{x}_k + \alpha\vec{d}_k) \leq f(\vec{x}_k) - \frac{\alpha}{4}\|\vec{d}_k\|^2$
(oder bis $\alpha < 2^{-10}\alpha_0$)

$\vec{x}_{k+1} := \vec{x}_k + \alpha\vec{d}_k$

bis $\|\vec{d}_k\| < \epsilon$ oder $\alpha < 2^{-10}\alpha_0$ oder k zu groß (keine Konvergenz).

Beispiel $f(x, y) = xy(x + y - 3)$ (vgl. S.35),

f hat ein relatives Minimum bei $\begin{pmatrix}1\\1\end{pmatrix}$.

Ergebnisse: *Gradientenverfahren mit Halbierung*

Startvektor $\begin{pmatrix}1.3\\1.3\end{pmatrix} \rightarrow \begin{pmatrix}1\\1\end{pmatrix}$ nach 9 Iterationsschritten

Startvektor $\begin{pmatrix}1.3\\-0.7\end{pmatrix} \rightarrow \begin{pmatrix}1\\1\end{pmatrix}$ nach 18 Iterationsschritten

Startvektor $\begin{pmatrix}1.5\\1.5\end{pmatrix}$ keine Konvergenz.

Spezialfall: *Lineares GLS* $A\vec{x} = \vec{b}$ *lösen*

Sei A eine *symmetrische, positiv definite* $(n, n)-$Matrix und $\vec{b}$ ein fester Vektor aus $\mathbb{R}^n$, dann gilt für die folgende Funktion f

$$f(\vec{x}) = \frac{1}{2} < A\vec{x}, \vec{x} > - < \vec{b}, \vec{x} > \quad , \quad \vec{x} \in \mathbb{R}^n$$

$\text{grad}\, f(\vec{x}) = A\vec{x} - \vec{b}$ (vgl. S.79)

und $\left(\frac{\partial^2 f}{\partial x_i \partial x_j}\right) = A$ positiv definit.

Damit gilt: f hat bei $\vec{x}$ ihr absolutes Minimum
$\Leftrightarrow$ $\vec{x}$ ist Lösung des linearen GLS $A\vec{x} = \vec{b}$.

Also kann man auch umgekehrt zur Bestimmung der Lösung des linearen GLS $A\vec{x} = \vec{b}$ für die Funktion $f(\vec{x}) = \frac{1}{2} < A\vec{x}, \vec{x} > - < \vec{b}, \vec{x} >$ mit Hilfe des Gradientenverfahrens das Minimum suchen. In diesem Fall ist die *Liniensuche* besonders einfach, wenn die Richtung $\vec{d}$ vorgegeben ist:

Gesucht: α mit $\min_{\alpha} f(\vec{x} + \alpha\vec{d})$.

Sei $p(\alpha) = f(\vec{x} + \alpha\vec{d})$. *Gesucht*: $\min_\alpha p(\alpha)$.

Notwendige Bedingung: $p'(\alpha) = 0$.

Es gilt

$$p'(\alpha) = \frac{d}{d\alpha} f(\vec{x} + \alpha\vec{d}) = \sum_{k=1}^{n} \frac{\partial f}{\partial x_k}(\vec{x} + \alpha\vec{d}) \cdot d_k \quad \text{(Kettenregel)}$$

$$=< \operatorname{grad} f(\vec{x} + \alpha\vec{d}), \vec{d} >=< A(\vec{x} + \alpha\vec{d}) - \vec{b}\,,\ \vec{d} >=< A\vec{x} - \vec{b}, \vec{d} > +\alpha < A\vec{d}, \vec{d} >= 0$$

$$\Rightarrow \quad \alpha = -\frac{< A\vec{x} - \vec{b}, \vec{d} >}{< A\vec{d}, \vec{d} >} \ .$$

$p''(\alpha) =< A\vec{d}, \vec{d} >> 0$ für $\vec{d} \neq \vec{0}$, da A positiv definit.

Also gilt

$$\boxed{\min_\alpha f(\vec{x} + \alpha\vec{d}) \quad \Leftrightarrow \quad \alpha = -\frac{< A\vec{x} - \vec{b}, \vec{d} >}{< A\vec{d}, \vec{d} >}}$$

Um die weiteren Eigenschaften besser beschreiben zu können, führen wir zunächst ein neues Skalarprodukt ein:

Definition 7.1 : *Skalarprodukt*

$< \vec{x}, \vec{y} >_A :=< A\vec{x}, \vec{y} >$

ist ein *Skalarprodukt*, falls A symmetrisch und positiv definit.

Beweis :

a) $< \vec{x}, \vec{y} >_A =< A\vec{x}, \vec{y} >=< \vec{x}, A^T\vec{y} >=< \vec{x}, A\vec{y} >=< A\vec{y}, \vec{x} >=< \vec{y}, \vec{x} >_A$
(da A symmetrisch)

b) $< \lambda\vec{x}, \vec{y} >_A =< A\lambda\vec{x}, \vec{y} >= \lambda < A\vec{x}, \vec{y} >= \lambda < \vec{x}, \vec{y} >_A$

c) $< \vec{x} + \vec{y}, \vec{z} >_A =< A(\vec{x} + \vec{y}), \vec{z} >=< A\vec{x}, \vec{z} > + < A\vec{y}, \vec{z} >=< \vec{x}, \vec{z} >_A + < \vec{y}, \vec{z} >_A$

d) $< \vec{x}, \vec{x} >_A =< A\vec{x}, \vec{x} >> 0 \quad \forall \vec{x} \neq \vec{0}$ (da A positiv definit)
$< \vec{x}, \vec{x} >_A =< A\vec{x}, \vec{x} >= 0 \quad \Leftrightarrow \quad \vec{x} = \vec{0}$.

Also sind alle Eigenschaften eines Skalarprodukts erfüllt. Mit Hilfe dieses Skalarprodukts ist damit eine neue Vektornorm definiert:

Definition 7.2 : *Vektornorm*

$\|\vec{x}\|_A := \sqrt{< \vec{x}, \vec{x} >_A} = \sqrt{< A\vec{x}, \vec{x} >}$.

Mit Hilfe dieser Norm kann man für das *Gradientenverfahren* folgende *Fehlerabschätzung* zeigen (vgl. Literatur)

$$\boxed{\|\vec{x}_k - \vec{x}\|_A \leq \Big(\frac{k(A) - 1}{k(A) + 1}\Big)^k \|\vec{x}_0 - \vec{x}\|_A}$$

wobei $\vec{x}$ die gesuchte Lösung des GLS $A\vec{x} = \vec{b}$, $\vec{x}_k$ der $k-$te Iterationsvektor und $k(A)$ die *Konditionszahl* $k(A) = \dfrac{|\lambda_1|}{|\lambda_n|}$ von A ist (vgl. S.12).

Hat A eine große Konditionszahl, so ist der Faktor $\dfrac{k(A)-1}{k(A)+1}$ fast 1, und die Konvergenz damit sehr langsam. Z.B. bei $k(A) = 1000$ benötigt man ≈ 350 Schritte, um den Fehler zu halbieren. Deshalb muß, um ein für die Praxis geeignetes Verfahren zu erhalten, das einfache Gradientenverfahren verbessert werden.

Konjugierte Gradientenverfahren (cg-Verfahren)

Sei A eine *symmetrische, positiv definite* $(n,n)-$Matrix und $\vec{b} \in I\!R^n$.

Gesucht: Lösung $\vec{x}$ des linearen GLS $A\vec{x} = \vec{b}$

$$\Leftrightarrow \quad \min_{\vec{x}} f(\vec{x}) \quad \text{mit} \quad f(\vec{x}) = \frac{1}{2} < A\vec{x}, \vec{x} > - < \vec{b}, \vec{x} > .$$

Idee des cg-Verfahrens

Man wählt die Richtungen $\vec{d}_k$ bzgl. des Skalarprodukts $< \vec{x}, \vec{y} >_A$ *orthogonal*, also $< \vec{d}_{k+1}, \vec{d}_k >_A = < A\vec{d}_{k+1}, \vec{d}_k > = 0$.

Da $\operatorname{grad} f(\vec{x}_k) = A\vec{x}_k - \vec{b}$, soll für die neue Richtung $\vec{d}_{k+1}$ gelten:

1) $\vec{d}_{k+1} = -\vec{g}_{k+1} + \beta_k \vec{d}_k \quad \text{mit} \quad \vec{g}_k = A\vec{x}_k - \vec{b}$

2) $< \vec{d}_{k+1}, \vec{d}_k >_A = < A\vec{d}_{k+1}, \vec{d}_k > = 0$,

also:

$< \vec{d}_{k+1}, \vec{d}_k >_A = < A\vec{d}_{k+1}, \vec{d}_k > = < \vec{d}_{k+1}, A\vec{d}_k >$ (da A symmetrisch)

$= < -\vec{g}_{k+1} + \beta_k \vec{d}_k , A\vec{d}_k > = - < \vec{g}_{k+1}, A\vec{d}_k > + \beta_k < A\vec{d}_k, \vec{d}_k > = 0 \quad \Rightarrow$

$$\boxed{\beta_k = \frac{< \vec{g}_{k+1}, A\vec{d}_k >}{< A\vec{d}_k, \vec{d}_k >}}$$

Bei vorgegebener Richtung $\vec{d}_k$ erhält man bei der *Liniensuche* für $\min\limits_{\alpha_k} f(\vec{x}_k + \alpha_k \vec{d}_k)$ (vgl. S.92)

$$\boxed{\alpha_k = -\frac{< A\vec{x}_k - \vec{b}, \vec{d}_k >}{< A\vec{d}_k, \vec{d}_k >} = -\frac{< \vec{g}_k, \vec{d}_k >}{< A\vec{d}_k, \vec{d}_k >}}$$

Damit erhalten wir das folgende *cg-Verfahren*

cg-Verfahren

Wähle Startvektor $\vec{x}_0$,
$\vec{g}_0 := A\vec{x}_0 - \vec{b}$, $\vec{d}_0 = -\vec{g}_0$.
Für $k = 0, 1, 2, \ldots$ berechne

$$\alpha_k = -\frac{<\vec{g}_k, \vec{d}_k>}{<A\vec{d}_k, \vec{d}_k>}$$

$$\vec{x}_{k+1} := \vec{x}_k + \alpha_k \vec{d}_k$$

$$\vec{g}_{k+1} := A\vec{x}_{k+1} - \vec{b} = A\vec{x}_k - \vec{b} + \alpha_k A\vec{d}_k = \vec{g}_k + \alpha_k A\vec{d}_k$$

$$\beta_k := \frac{<\vec{g}_{k+1}, A\vec{d}_k>}{<A\vec{d}_k, \vec{d}_k>}$$

$$\vec{d}_{k+1} := -\vec{g}_{k+1} + \beta_k \vec{d}_k$$

bis $\|\vec{d}_{k+1}\| < \epsilon$ oder k zu groß (keine Konvergenz).

Da die neuen Werte auf den Speicherplätzen der alten Werte gespeichert werden, benötigt man für die Programmierung keine Indizes. Ohne Indizes lautet das *cg-Verfahren*:

cg-Verfahren

Wähle Startvektor $\vec{x}$,
$\vec{g} := A\vec{x} - \vec{b}$, $\vec{d} = -\vec{g}$.
Für $k = 0, 1, 2, \ldots$ berechne

$$\vec{c} := A\vec{d} \quad , \quad \gamma :=< \vec{c}, \vec{d} > \quad , \quad \alpha := -\frac{<\vec{g}, \vec{d}>}{\gamma}$$

$$\vec{x} := \vec{x} + \alpha\vec{d} \quad , \quad \vec{g} := \vec{g} + \alpha\vec{c}$$

$$\beta := \frac{<\vec{g}, \vec{c}>}{\gamma} \quad , \quad \vec{d} := -\vec{g} + \beta\vec{d}$$

bis $\|\vec{d}\| < \epsilon$ oder k zu groß (keine Konvergenz).

Für dieses *cg-Verfahren* lautet die *Fehlerabschätzung* (vgl. Literatur):

$$\|\vec{x}_k - \vec{x}\|_A \leq 2\Big(\frac{\sqrt{k(A)} - 1}{\sqrt{k(A)} + 1}\Big)^k \|\vec{x}_0 - \vec{x}\|_A$$

Beispiel

$k(A) = 1000 \quad \Rightarrow \quad \|\vec{x}_k - \vec{x}\|_A \leq 2(0.9387)^k \|\vec{x}_0 - \vec{x}\|_A$

Der Fehler wird in diesem Beispiel nach 22 Schritten halbiert (vgl. 350 Schritte beim einfachen Gradientenverfahren, S.93).

Beispiel A $(25,25)-$Matrix von Beispiel S.26

$$A = \begin{pmatrix} B & D & & & \\ D & B & D & & \\ & D & B & D & \\ & & D & B & D \\ & & & D & B \end{pmatrix}$$

mit den Teilmatrizen (jeweils symmetrische $(5,5)-$Matrizen)

$$B = \begin{pmatrix} 4 & -1 & & & \\ -1 & 4 & -1 & & \\ & -1 & 4 & -1 & \\ & & -1 & 4 & -1 \\ & & & -1 & 4 \end{pmatrix} \quad , \quad D = \begin{pmatrix} -1 & & & & \\ & -1 & & & \\ & & -1 & & \\ & & & -1 & \\ & & & & -1 \end{pmatrix}$$

$$\vec{b} = -\frac{1}{18}(1,1,1,\ldots,1)^T \ .$$

Mit dem Startvektor $\vec{x}_0 = (1,1,\ldots,1)^T$ benötigt man nur 5 Iterationen, um die gesuchte Lösung $\vec{x}$ des GLS $A\vec{x} = \vec{b}$ mit einem Fehler von $\approx 10^{-11}$ anzunähern (vgl. 22 Iterationen beim Überrelaxationsverfahren, S.27).

Vorkonditionierung

Eine noch bessere Konvergenz erhält man, wenn man zusätzlich eine *Vorkonditionierung* vornimmt.

Wählt man anstelle des Vektors $\vec{g}_k = A\vec{x}_k - \vec{b}$ den Vektor $\vec{h}_k = C^{-1}\vec{g}_k$ mit einer regulären Matrix C , so erhält man im 1.Iterationsschritt

$$\begin{aligned} \vec{x}_1 &= \vec{x}_0 + \alpha(-C^{-1}\vec{g}_0) = \vec{x}_0 - \alpha C^{-1}(A\vec{x}_0 - \vec{b}) \\ &= (\vec{x}_0 - \alpha C^{-1}A\vec{x}_0) + \alpha C^{-1}\vec{b} \ . \end{aligned}$$

Mit $\alpha = 1$ und $C = A$ erhielte man im 1.Iterationsschritt: $\vec{x}_1 = A^{-1}\vec{b}$, also schon die gesuchte Lösung des GLS $A\vec{x} = \vec{b}$. Mit $C = E$ erhält man das *cg-Verfahren.* Ist $C \approx A$, so erhält man schon nach wenigen Schritten die gesuchte Lösung.
C ist so zu wählen, daß das GLS $C\vec{h}_k = \vec{g}_k$ einfach zu lösen ist.

In der Praxis führt man für A eine *"unvollständige" $\tilde{L}\tilde{D}\tilde{L}^T-$Cholesky-Zerlegung* durch: $C = \tilde{L}\tilde{D}\tilde{L}^T$,

z.B.: nur für einige Nebendiagonalen oder

z.B.: nur für die Elemente von A, die von Null verschieden sind (bei schwach besetzten Matrizen).

Man erhält dann anstelle von $k(A)$ die Konditionszahl $k(C^{-1}A)$.

pcg-Verfahren *(preconditioning conjugate gradients)*

Wähle Startvektor $\vec{x}$, $\vec{g} := A\vec{x} - \vec{b}$,

$C\vec{h} = \vec{g}$ GLS lösen , $\vec{d} = -\vec{h}$.

Für $k = 0, 1, 2, \ldots$ berechne

$$\vec{c} := A\vec{d} \ , \ \gamma :=< \vec{c}, \vec{d} > \ , \ \alpha := -\frac{< \vec{g}, \vec{d} >}{\gamma}$$

$$\vec{x} := \vec{x} + \alpha\vec{d} \ , \ \vec{g} := \vec{g} + \alpha\vec{c}$$

$C\vec{h} = \vec{g}$ GLS lösen

$$\beta := \frac{< \vec{h}, \vec{c} >}{\gamma} \ , \ \vec{d} := -\vec{h} + \beta\vec{d}$$

bis $\|\vec{d}\| < \epsilon$ oder k zu groß (keine Konvergenz).

Bemerkung :

Ist die Matrix A nicht symmetrisch oder nicht positiv definit, so kann man die Matrix $A^T A$ betrachten, denn $A^T A$ ist symmetrisch und, falls A regulär, auch positiv definit

denn:

$(A^T A)^T = A^T A^{TT} = A^T A$

$< A^T A\vec{x}, \vec{x} >=< A\vec{x}, A\vec{x} >= \|A\vec{x}\|^2 \geq 0$, $= 0 \ \Leftrightarrow \vec{x} = \vec{0}$ (falls A regulär).

Multipliziert man das GLS $A\vec{x} = \vec{b}$ von links mit A^T , so erhält man das neue GLS $A^T A\vec{x} = A^T\vec{b}$ mit der symmetrischen, positiv definiten Koeffizientenmatrix $A^T A$ und der neuen rechten Seite $A^T\vec{b}$.

Allgemeine Minimierungsprobleme

Wir kommen nochmal zurück auf *allgemeine Minimierungsprobleme*:

Gesucht: $\min\limits_{\vec{x} \in D} f(\vec{x})$

für eine stetig differenzierbare Funktion $f : D \subset I\!R^n \to I\!R$.

Man kann das *cg-Verfahren* so abändern, daß man es auch auf solche allgemeinen Minimierungsprobleme anwenden kann. Man muß jeweils $A\vec{x} - \vec{b}$ durch $\operatorname{grad} f(\vec{x})$ ersetzen. Dann erhält man das Verfahren von *Fletcher – Reeves*. Eine Variante hiervon stammt von *Polak – Ribière*.

Verfahren von Fletcher – Reeves

Wähle Startvektor $\vec{x}$,

$\vec{g} := \operatorname{grad} f(\vec{x})$, $\vec{d} = -\vec{g}$.

Für $k = 0, 1, 2, \ldots$ berechne

α mit $\min\limits_{\alpha} f(\vec{x} + \alpha\vec{d})$ (evt. mit Schrittweitenhalbierung annähern)

$\vec{x} := \vec{x} + \alpha\vec{d}$, $\vec{\tilde{g}} := \operatorname{grad} f(\vec{x})$

$\beta := \dfrac{< \vec{\tilde{g}}, \vec{\tilde{g}} >}{< \vec{g}, \vec{g} >}$

$\vec{g} := \vec{\tilde{g}}$, $\vec{d} := -\vec{g} + \beta\vec{d}$

bis $\|\vec{d}\| < \epsilon$ oder k zu groß (keine Konvergenz).

Variante von Polak – Ribière

Anstelle von obigem β wird β folgendermaßen berechnet

$\beta := \dfrac{< \vec{\tilde{g}} - \vec{g}, \vec{\tilde{g}} >}{< \vec{g}, \vec{g} >}$.

Beispiel $f(x, y) = xy(x + y - 3)$ (vgl. S.35 und S.91),

f hat ein relatives Minimum bei $\begin{pmatrix} 1 \\ 1 \end{pmatrix}$.

Ergebnisse: *Verfahren von Fletcher – Reeves*

Startvektor $\begin{pmatrix} 1.3 \\ 1.3 \end{pmatrix} \rightarrow \begin{pmatrix} 1 \\ 1 \end{pmatrix}$ nach 6 Iterationsschritten

Startvektor $\begin{pmatrix} 1.3 \\ -0.7 \end{pmatrix} \rightarrow \begin{pmatrix} 1 \\ 1 \end{pmatrix}$ nach 15 Iterationsschritten

Startvektor $\begin{pmatrix} 1.5 \\ 1.5 \end{pmatrix}$ keine Konvergenz.

Eine andere Möglichkeit, das obige Minimierungsproblem zu lösen, bietet das *Newton-Verfahren* (vgl. S.35) oder das *Quasi-Newton-Verfahren.*
Beim *Quasi-Newton-Verfahren* wird in jedem Iterationsschritt die Funktionalmatrix $\left(\dfrac{\partial^2 f}{\partial x_i \partial x_j}\right)$ angenähert (vgl. Literatur, z.B.: Schabak – Werner, Numerische Mathematik, S.312 ff).

VIII Differenzenverfahren, Finite Elemente

Problemstellung

Gegeben: Partielle Randwertaufgabe

Beispiel *Elliptische DGL*

Gegeben: $G \subset \mathbb{R}^2$ (Normalbereich)

$$\boxed{u_{xx} + u_{yy} = f \quad , \quad u_{|\partial G} = \varphi}$$

hierbei sei $f : G \to \mathbb{R}$ stetig in G , $\varphi : \partial G \to \mathbb{R}$ stetig in ∂G.

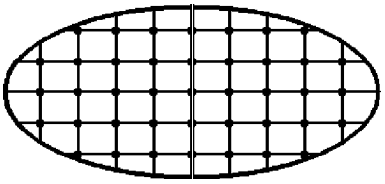
Gesucht: Lösung u in G mit $u \in C^2(G)$ und $u \in C(\overline{G})$.

1. Differenzenverfahren

Man legt ein Netz über G
und erhält die Gitterpunkte (x_i, y_j) .

u_{ij} sei die *Näherungslösung* für $u(x_i, y_j)$.

Falls (x_i, y_j) *Randpunkt*
$\Rightarrow \quad u(x_i, y_j) = \varphi(x_i, y_j)$ als Randwert gegeben.



Die Ableitungen u_{xx} und u_{yy} werden durch *Differenzenquotienten* ersetzt:

$$u_{xx}(x_i, y_j) \approx \frac{1}{h^2}(u_{i-1,j} - 2u_{ij} + u_{i+1,j})$$

$$u_{yy}(x_i, y_j) \approx \frac{1}{k^2}(u_{i,j-1} - 2u_{ij} + u_{i,j+1})$$

jeweils für die inneren Gitterpunkte.
Hierbei sei h (bzw. k) die Schrittweite in $x-$ (bzw. $y-$) Richtung.

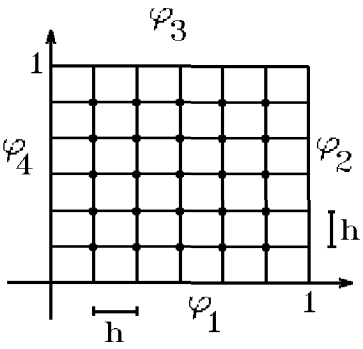
Beispiel *Rechteck*

Sei $h = k = \frac{1}{n}$,

$x_i = ih$, $y_j = jh$,

$f_{ij} = f(x_i, y_j)$

$(0 \le i, j \le n)$.



Folgende Punkte sind Randpunkte:

$u_{i,0} = u(x_i, 0) = \varphi_1(x_i) = \varphi(x_i, 0)$, $u_{i,n} = u(x_i, 1) = \varphi_3(x_i) = \varphi(x_i, 1)$, $(0 \le i \le n)$,
$u_{0,j} = u(0, y_j) = \varphi_4(y_j) = \varphi(0, y_j)$, $u_{n,j} = u(1, y_j) = \varphi_2(y_j) = \varphi(1, y_j)$, $(0 \le j \le n)$.

Nach Multiplikation mit h^2 erhalten wir die folgenden Gleichungen

$$(u_{i-1,j} - 2u_{ij} + u_{i+1,j}) + (u_{i,j-1} - 2u_{ij} + u_{i,j+1}) = h^2 f_{ij} \quad , \quad (1 \le i, j \le n-1).$$

Das sind $(n-1)^2$ viele Gleichungen für die $(n-1)^2$ Unbekannten
$u_{11}, u_{12}, \ldots, u_{n-1,n-1}$, also die Näherungslösungen in den inneren Gitterpunkten.

Z.B. erhalten wir für $i = j = 1$ die folgende Gleichung:

$$\underbrace{u_{01}}_{\text{R.P.}} - 2u_{11} + u_{21} + \underbrace{u_{10}}_{\text{R.P.}} - 2u_{11} + u_{12} = h^2 f_{11}$$

wobei u_{01} und u_{10} Randpunkte (R.P.) sind.
Bringen wir diese Randpunkte auf die rechte Seite und multiplizieren anschließend mit (-1), so erhalten wir die Gleichung:

$$4u_{11} - u_{12} - u_{21} = -h^2 f_{11} + u_{01} + u_{10} \ .$$

In Matrizenschreibweise erhalten wir insgesamt z.B. für $n = 5$ folgendes lineare GLS

$$\begin{pmatrix}
4 & -1 & & & -1 & & & & \\
-1 & 4 & -1 & & & -1 & & & \\
 & -1 & 4 & -1 & & & -1 & & \\
 & & -1 & 4 & & & & -1 & \\
-1 & & & & 4 & -1 & & & \ddots \\
 & -1 & & & -1 & 4 & -1 & & \\
 & & -1 & & & -1 & 4 & -1 & \\
 & & & -1 & & & -1 & 4 & \\
 & & & & \ddots & & & & \ddots
\end{pmatrix}
\begin{pmatrix} u_{11} \\ u_{12} \\ u_{13} \\ u_{14} \\ u_{21} \\ u_{22} \\ u_{23} \\ u_{24} \\ \vdots \end{pmatrix}
= -h^2 \begin{pmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{14} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{24} \\ \vdots \end{pmatrix} + \vec{r}$$

mit dem Vektor $\vec{r}$ der Randpunkte

$$\vec{r} = \begin{pmatrix} u_{01} + u_{10} \\ u_{02} \\ u_{03} \\ u_{04} + u_{15} \\ u_{20} \\ 0 \\ 0 \\ u_{25} \\ \vdots \end{pmatrix} .$$

Die Koeffizientenmatrix A ist eine $((n-1)^2, (n-1)^2)-$Bandmatrix der Bandbreite $(n-1)$.
Sie ist *symmetrisch, positiv definit* und *"fast" diagonaldominant.*

Das GLS läßt sich für kleine n mit Hilfe des *Cholesky-Verfahrens* für Bandmatrizen lösen. Für große n sollte man ein *iteratives Verfahren* benutzen, z.B. das *Überrelaxations-Verfahren* (vgl S.27) oder das *cg– oder pcg–Verfahren* (vgl. S.94,96).

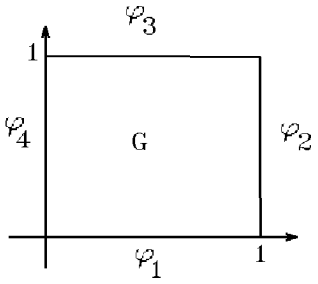
Beispiel 1

$G = \left\{ \begin{pmatrix} x \\ y \end{pmatrix} \in I\!R^2 \ : \ 0 < x, y < 1 \right\}$ (Rechteck)

$u_{xx} + u_{yy} = 0 \quad \text{in } G.$

Randbedingung:

$u(x,0) = \varphi_1(x) = x^2$
$u(x,1) = \varphi_3(x) = x^2 - 1$
$u(0,y) = \varphi_4(y) = -y^2$
$u(1,y) = \varphi_2(x) = 1 - y^2.$



Exakte Lösung: $u(x,y) = x^2 - y^2.$

Ergebnis: Differenzenverfahren

Maximaler Fehler zwischen u_{ij} und $u(x_i, y_j)$ in Abhängigkeit der Anzahl der Unterteilungen n :

n	maximaler Fehler
5	$2 \cdot 10^{-12}$
10	$5 \cdot 10^{-12}$
20	$1.5 \cdot 10^{-11}$

Beispiel 2

$u_{xx} + u_{yy} = 0 \quad \text{in } G = \left\{ \begin{pmatrix} x \\ y \end{pmatrix} \in I\!R^2 \ : \ 0 < x, y < 1 \right\}$ (Rechteck).

Randbedingung: $u_{|\partial G} = \sin \pi x + \sin \pi y \ .$

Exakte Lösung:

$$u(x,y) = \frac{\sin \pi x}{\sinh \pi} \Big(\sinh \pi y - \sinh \pi (y-1) \Big) + \frac{\sin \pi y}{\sinh \pi} \Big(\sinh \pi x - \sinh \pi (x-1) \Big) \ .$$

Ergebnis: Differenzenverfahren

Maximaler Fehler zwischen u_{ij} und $u(x_i, y_j)$ in Abhängigkeit der Anzahl der Unterteilungen n :

n	maximaler Fehler
5	$3.4 \cdot 10^{-2}$
10	$9.4 \cdot 10^{-3}$
20	$2.4 \cdot 10^{-3}$

Ein weiteres Beispiel befindet sich im nächsten Abschnitt über *Finite Elemente* (vgl. S.126).

Algorithmus für $u_{xx} + u_{yy} = f$ in G mit

$G = \left\{ \begin{pmatrix} x \\ y \end{pmatrix} \in I\!R^2 \ : \ 0 < x, y < 1 \right\}$ (Rechteck) , $h = \frac{1}{n}$,

mit den Randbedingungen:

$u(x,0) = \varphi_1(x)$, $u(x,1) = \varphi_3(x)$, $u(0,y) = \varphi_4(y)$, $u(1,y) = \varphi_2(y)$.

Erzeugung der Matrix (Bandmatrix für *Cholesky-Verfahren*)

Bandbreite $M = n - 1$, $N = M^2$

$(N, M+1)-$Matrix A mit 0 vorbelegen

für $i = 1$ bis N : $a_{i,M+1} = 4$: Ende der $i-$Schleife
für $i = 2$ bis N : $a_{i,M} = -1$: Ende der $i-$Schleife
für $i = 1$ bis N *step* M : $a_{i,M} = 0$: Ende der $i-$Schleife
für $i = n$ bis N : $a_{i,1} = -1$: Ende der $i-$Schleife

Erzeugung der rechten Seite

für $i = 1$ bis M
  für $j = 1$ bis M
    $x = i * h$, $y = j * h$, $k = M * (i-1) + j$
    $b_k = -h^2 f(x,y)$
    falls $i = 1 \ \Rightarrow \ b_k = b_k + \varphi_4(y)$
    falls $i = M \ \Rightarrow \ b_k = b_k + \varphi_2(y)$
    falls $j = 1 \ \Rightarrow \ b_k = b_k + \varphi_1(x)$
    falls $j = M \ \Rightarrow \ b_k = b_k + \varphi_3(x)$
  Ende $j-$Schleife
Ende $i-$Schleife

Cholesky-Verfahren oder *iteratives Verfahren* anwenden $\Rightarrow$ u_{ij} .

Bemerkung :

Liegen andere Grundgebiete (als Rechteck) vor, so müssen evt. in der Nähe des Randes andere Differenzenquotienten gewählt werden (siehe z.B.: Schwarz: Numerische Mathematik). Dann erhält man andere Koeffizientenmatrizen. Diese sind aber i.a. immer noch symmetrische Bandmatrizen.

2. Finite Elemente

Zusammenhang zwischen Differentialgleichung und Variationsaufgabe

Sei $M = \{f : [a,b] \to \mathbb{R} \ : \ f \quad 2-\text{mal stetig differenzierbar} \ , \ f(a) = \alpha \ , \ f(b) = \beta\}$ die Menge aller auf $[a,b]$ 2–mal stetig differenzierbaren Funktionen, die die *Randbedingungen* $f(a) = \alpha \ , \ f(b) = \beta$ erfüllen.

Sei $F : D_F \subset \mathbb{R}^3 \to \mathbb{R}$ 2–mal stetig differenzierbar. Dann definieren wir das *Funktional* $J : M \to \mathbb{R}$ mit

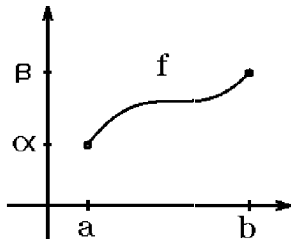
$$\boxed{J(f) := \int_a^b F(x, f(x), f'(x)) \ dx}$$

Hierbei wird jeder Funktion $f \in M$ die reelle Zahl $J(f)$ zugeordnet.

Gesucht: $f \in M$ mit $J(f)$ *minimal* , also: $J(f) \leq J(g) \quad \forall g \in M$.

Beispiel

Gesucht ist die Funktion $f \in C^2[a,b]$, deren Graph die kürzeste Verbindung zwischen den Punkten $\begin{pmatrix} a \\ \alpha \end{pmatrix}$ und $\begin{pmatrix} b \\ \beta \end{pmatrix}$ ergibt.



Kurvenlänge: $J(f) = \int_a^b \sqrt{1 + (f'(x))^2} \ dx \quad , \quad f(a) = \alpha \ , \ f(b) = \beta \ ,$

also $F(x, f, f') = \sqrt{1 + (f'(x))^2}$.

Idee der **Variationsrechnung**

f sei die gesuchte Extremalfunktion von J , also $J(f) \leq J(g) \quad \forall g \in M$.

$u : [a,b] \to \mathbb{R}$ sei eine in $[a,b]$ 2–mal stetig differenzierbare Funktion mit $u(a) = u(b) = 0$, dann gilt:

$(f + \lambda u) \in M \quad \forall \lambda \in \mathbb{R}$ und für kleine $|\lambda|$ ist $(f + \lambda u)$ in der "Nachbarschaft" von f.
Setzen wir $(f + \lambda u)$ in J ein, so erhalten wir eine Funktion $\varphi : \mathbb{R} \to \mathbb{R}$ von λ

$$\boxed{\varphi(\lambda) = J(f + \lambda u) = \int_a^b F(x, f + \lambda u, f' + \lambda u') \ dx}$$

Notwendige Bedingung für ein *Minimum* von $\varphi(\lambda)$ bei $\lambda = 0$ ist $\varphi'(0) = 0$, also

$$\varphi'(0) = \int_a^b \Big(F_f \cdot u + F_{f'} \cdot u' \Big) \ dx = 0 \ .$$

Mit partieller Integration folgt hieraus

$$\varphi'(0) = \int_a^b F_f \cdot u \; dx + \underbrace{u\, F_{f'}\Big]_{x=a}^{b}}_{=0 \ , \ \text{da } u(a)=u(b)=0} - \int_a^b u \frac{d}{dx}(F_{f'}) \; dx$$

$$= \int_a^b \Big(F_f - \frac{d}{dx}(F_{f'}) \Big) u \; dx = 0 \ .$$

Da diese Gleichung für alle 2–mal stetig differenzierbaren Funktionen $u : [a,b] \to I\!R$ mit $u(a) = u(b) = 0$ gilt, folgt aus folgendem Hilfssatz:

$$\boxed{\frac{d}{dx}(F_{f'}) - F_f = 0}$$

Dies ist die zur Variationsaufgabe gehörende *Eulersche Differentialgleichung.*

Ergebnis 8.1 : Damit f das Funktional $J(f)$ *minimal* macht, muß f notwendigerweise Lösung der zugehörigen *Eulersche DGL* $\frac{d}{dx}(F_{f'}) - F_f = 0$ mit $f(a) = \alpha \ , \ f(b) = \beta$ sein.

Hilfssatz 8.2

Sei $g : [a,b] \to I\!R$ stetig in $[a,b]$, und es gelte für alle 2–mal stetig differenzierbaren Funktionen $u : [a,b] \to I\!R$ mit $u(a) = u(b) = 0$:

$$\int_a^b g(x)u(x) \; dx = 0 \ .$$

Dann gilt: $g(x) = 0 \quad \forall x \in [a,b]$.

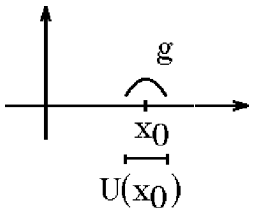
Beweis :

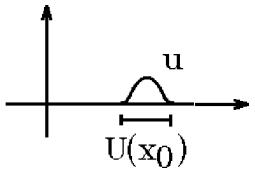
Annahme:

$\exists x_0 \in [a,b]$ mit $g(x_0) \neq 0$
$\Rightarrow \quad \exists U(x_0)$ mit
$g(x) \neq 0 \quad \forall x \in U(x_0)$ (da g stetig).
Wählen wir $u(x) = 0$ außerhalb von $U(x_0)$,
aber $u(x) > 0$ im Innern von $U(x_0)$, so folgt

$$\int_a^b g(x)u(x) \; dx \neq 0$$

$\Rightarrow$ Widerspruch zu $= 0$.





Beispiel (von S.102)

$$J(f) = \int_a^b \sqrt{1 + (f'(x))^2} \; dx \ .$$

Die zugehörige Eulersche DGL lautet, da $F_f = 0$:

$$\frac{d}{dx}\Big(F_{f'}\Big) = \frac{d}{dx}\Big(\frac{f'}{\sqrt{1+f'^2}}\Big) = 0 \quad \Rightarrow \quad \frac{f'}{\sqrt{1+f'^2}} = c$$

$\Rightarrow \quad f'^2 = c^2(1+f'^2) \quad \Rightarrow \quad f'^2(1-c^2) = c^2 \quad \Rightarrow \quad f'^2 = d \quad \Rightarrow \quad f' = c_1$
$\Rightarrow \quad f(x) = c_1 x + c_2 \quad$ mit $\quad f(a) = \alpha \ , \ f(b) = \beta$
$\Rightarrow \quad f$ ist *Gerade* zwischen $\begin{pmatrix} a \\ \alpha \end{pmatrix}$ und $\begin{pmatrix} b \\ \beta \end{pmatrix}$.

Wir nutzen das *Ergebnis 8.1* folgendermaßen:

Wollen wir eine *Randwertaufgabe* lösen, deren DGL sich als *Eulersche DGL* eines *Variationsproblems* darstellen läßt, so versuchen wir, das zugehörige *Variationsproblem* (näherungsweise) zu lösen. Die Lösung des *Variationsproblems* muß dann auch Lösung der gegebenen *Randwertaufgabe* sein.

Beispiel

Gegebene Randwertaufgabe: (lineare DGL 2.Ordnung)

$$y'' + a(x)y' + b(x)y = c(x) \quad , \quad y(a) = \alpha \ , \ y(b) = \beta$$

Gesucht: zugehörige Variationsaufgabe

$$J(f) = \int_a^b \Big(pf'^2 - qf^2 + 2rf \Big) \, dx \quad , \quad f(a) = \alpha \ , \ f(b) = \beta$$

also $\quad F(x, f, f') = pf'^2 - qf^2 + 2rf$.
Zugehörige Eulersche DGL:
$\frac{d}{dx}(2pf') - (-2qf + 2r) = 0 \quad \Rightarrow \quad 2p'f' + 2pf'' + 2qf - 2r = 0$
$\Rightarrow \quad pf'' + p'f' + qf = r \quad \Rightarrow \quad f'' + \frac{p'}{p} f' + \frac{q}{p} f = \frac{r}{p}$.
Es muß gelten: $\frac{p'(x)}{p(x)} = a(x)$, $\frac{q(x)}{p(x)} = b(x)$, $\frac{r(x)}{p(x)} = c(x)$.
Aus der DGL $\frac{p'}{p} = a(x)$ folgt $\int \frac{1}{p} \, dp = \int a(x) \, dx \quad \Rightarrow \quad \ln|p| = \int a(x) \, dx \quad \Rightarrow$

$$p(x) = \mathrm{e}^{\int a(x) \, dx} \quad , \quad q(x) = p(x)b(x) \quad , \quad r(x) = p(x)c(x)$$

Beispiel hierzu

$y'' - y = x \quad , \quad y(0) = y(1) = 0$,
$\Rightarrow \quad p(x) = \mathrm{e}^{\int 0 \, dx} = 1 \quad , \quad q(x) = -1 \quad , \quad r(x) = x$.
Also lautet die zugehörige Variationsaufgabe

$$J(f) = \int_0^1 (f'^2 + f^2 + 2xf)\ dx \quad , \quad f(0) = f(1) = 0 \ .$$

Eine Lösung dieser Variationsaufgabe ist auch Lösung der gegebenen Randwertaufgabe $y'' - y = x \quad , \quad y(0) = y(1) = 0$.

Näherungslösung des Variationsproblems

$\varphi_i : [a,b] \to I\!R \quad , \quad (i = 1, 2, \ldots n)$, seien vorgegebene geeignete *Grundfunktionen*.
f werde angenähert durch

$$\boxed{f(x) \approx \sum_{i=1}^{n} c_i \varphi_i(x)}$$

Gesucht: die unbekannten Koeffizienten $c_i \quad , \quad (i = 1, \ldots, n)$.

Setzen wir anstelle von f diese Annäherung $\sum_{i=1}^{n} c_i \varphi_i(x)$ in das Funktional $J(f)$ ein, so erhalten wir eine Funktion von den unbekannten Koeffizienten c_i

$$J(f) \approx L(c_1, \ldots, c_n) = \int_a^b F(x\ ,\ \sum_{i=1}^{n} c_i \varphi_i\ ,\ \sum_{i=1}^{n} c_i \varphi_i')\ dx \ .$$

Damit $L(c_1, \ldots, c_n)$ minimal wird, muß gelten: $\dfrac{\partial L}{\partial c_k} = 0 \quad \forall k = 1, \ldots, n$, also

$$\boxed{\frac{\partial L}{\partial c_k} = \int_a^b (F_f \varphi_k + F_{f'} \varphi_k')\ dx = 0 \quad \forall k = 1, \ldots, n}$$

Dies ist ein GLS für die unbekannten Koeffizienten $c_i \quad , \ (i = 1, \ldots, n)$.
Die Lösung ergibt $\sum_{i=1}^{n} c_i \varphi_i(x)$ als Näherung für die gesuchte Lösung f des Variationsproblems, und damit eine Näherungslösung der zugehörigen Eulerschen DGL einschließlich der gegebenen Randbedingungen.

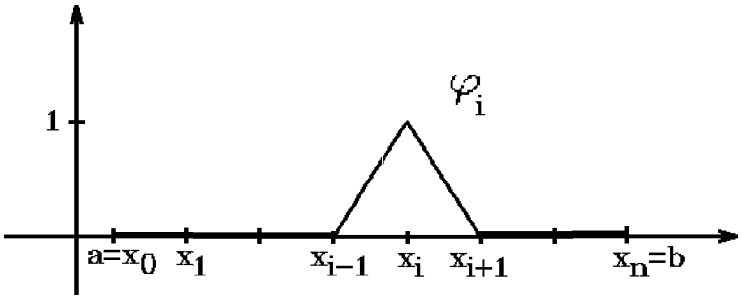
Wahl der Grundfunktionen

Sind die Randbedingungen $f(a) = f(b) = 0$ gegeben, und läßt man in M alle in $[a,b]$ stetigen und stückweise glatten Funktionen zu, also

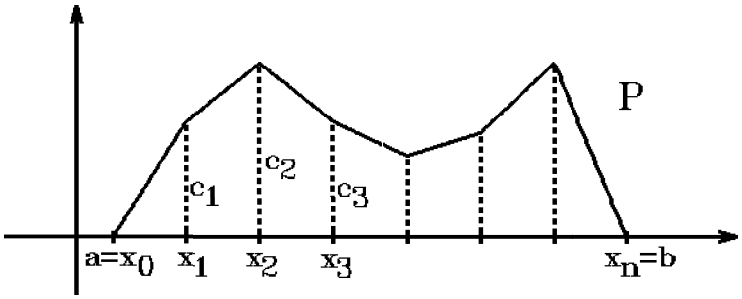
$$M = \Big\{ f : [a,b] \to I\!R\ ,\ f \text{ stetig und stückweise glatt}\ ,\ f(a) = f(b) = 0 \Big\} \ ,$$

so können als *Grundfunktionen* $\varphi_i(x)$ folgende einfache Funktionen gewählt werden:
Seien $x_i = a + ih \quad , \ (i = 0, 1, \ldots, n)$, die Teilpunkte des Intervalls $[a,b]$ mit der Schrittweite $h = \dfrac{b-a}{n}$, dann definieren wir als *Grundfunktionen* $\varphi_i \ , \ (1 \le i \le n-1)$:

$$\varphi_i(x) = \begin{cases} (x - x_{i-1})/h & \text{,falls } x_{i-1} \le x \le x_i \\ (x_{i+1} - x)/h & \text{,falls } x_i < x \le x_{i+1} \\ 0 & \text{, sonst} \end{cases}$$



Dann erhalten wir für $P(x) = \sum_{i=1}^{n-1} c_i \varphi_i(x)$ folgenden Graphen:



Da $\varphi_i(x_i) = 1$ und $\varphi_i(x_j) = 0 \ \ \forall j \neq i$, gilt:

$$P(x_i) = \sum_{j=1}^{n-1} c_j \varphi_j(x_i) = c_i \quad \forall\, 1 \le i \le n-1 \ .$$

Beispiel

Gegeben: Variationsaufgabe

$$J(f) = \int_a^b (pf'^2 - qf^2 + 2rf)\ dx \quad \text{minimal} \quad , \quad f(a) = f(b) = 0 \ .$$

Gesucht: Näherungslösung $\quad P(x) = \sum_{i=1}^{n-1} c_i \varphi_i(x)$.

$$L(c_1, \ldots, c_{n-1}) = \int_a^b \Big(p\Big(\sum_{i=1}^{n-1} c_i \varphi_i'\Big)^2 - q\Big(\sum_{i=1}^{n-1} c_i \varphi_i\Big)^2 + 2r \sum_{i=1}^{n-1} c_i \varphi_i \Big)\ dx \ .$$

Notwendige Bedingung für Minimum:

$$\frac{\partial L}{\partial c_k} = \int_a^b \Big(2p\Big(\sum_{i=1}^{n-1} c_i\varphi_i'\Big)\varphi_k' - 2q\Big(\sum_{i=1}^{n-1} c_i\varphi_i\Big)\varphi_k + 2r\varphi_k\Big)\ dx$$

$$= 2\sum_{i=1}^{n-1} c_i \underbrace{\Big[\int_a^b p\varphi_i'\varphi_k'\ dx - \int_a^b q\varphi_i\varphi_k\ dx\Big]}_{a_{ki}} + 2\underbrace{\int_a^b r\varphi_k\ dx}_{-b_k} = 0$$

$$\Rightarrow \quad \sum_{i=1}^{n-1} a_{ki}c_i = b_k \quad \forall k = 1,\ldots,n-1 \quad \Rightarrow \quad \boxed{A\vec{c} = \vec{b}}$$

mit

$$a_{ki} = \int_a^b p\varphi_i'\varphi_k'\ dx - \int_a^b q\varphi_i\varphi_k\ dx \quad , \quad b_k = -\int_a^b r\varphi_k\ dx\ .$$

Berechnet man diese Integrale (z.B. numerisch mit Hilfe Gaußscher Quadraturformeln (siehe Literatur)), so erhält man die Werte a_{ki} der Koeffizientenmatrix A und die Werte b_k des Vektors $\vec{b}$ der rechten Seite des linearen GLS $A\vec{c} = \vec{b}$.
Die Lösung dieses linearen GLS $A\vec{c} = \vec{b}$ ergibt die gesuchten Koeffizienten c_i und damit die Näherungslösung $P(x) = \sum_{i=1}^{n-1} c_i\varphi_i(x)$.

Beispiel hierzu

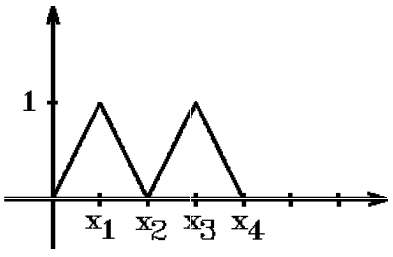
$y'' - y = x \ , \ y(0) = y(1) = 0$

Zugehörige Variationsaufgabe (vgl. S.104/105)

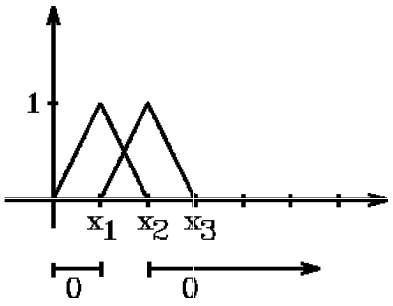
$J(f) = \int_0^1 (f'^2 + f^2 + 2xf)\ dx \ , \ f(0) = f(1) = 0 \ ,$
also $p(x) \equiv 1\ ,\ q(x) \equiv -1\ ,\ r(x) = x \quad \Rightarrow$

$$a_{ki} = \int_0^1 \varphi_i'\varphi_k'\ dx + \int_0^1 \varphi_i\varphi_k\ dx \quad , \quad b_k = -\int_0^1 x\varphi_k(x)\ dx\ .$$

Es gilt: $a_{ki} = 0$ für $k \le i-2$ und $k \ge i+2$



z.B.: $\varphi_1\varphi_3 \equiv 0$

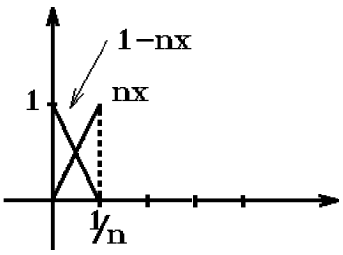


$\varphi_1(x)\varphi_2(x) = 0$ für $x \le x_1$ und $x \ge x_2$

Weiter gilt: $a_{ki} = a_{ik} \quad \Rightarrow \quad A$ symmetrisch.

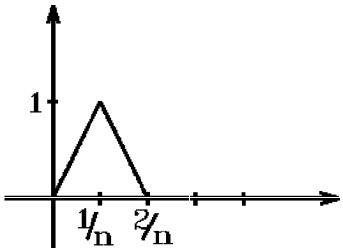
Berechnung von $a_{k,k-1}$:

$$\int_0^1 \varphi_k \varphi_{k-1}\, dx = \int_0^{1/n} nx(1-nx)\, dx$$
$$= n\Big[\frac{1}{2n^2} - \frac{n}{3n^3}\Big] = \frac{1}{6n}$$
$$\int_0^1 \varphi_k' \varphi_{k-1}'\, dx = \int_0^{1/n} n(-n)\, dx = -n$$
$$\Rightarrow \quad a_{k,k-1} = a_{k-1,k} = \frac{1}{6n} - n = \frac{1-6n^2}{6n}\ .$$



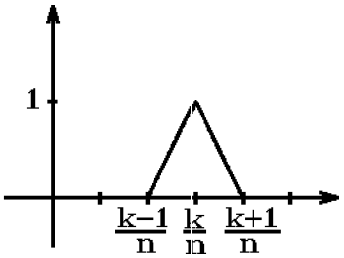
Berechnung von a_{kk} :

$$\int_0^1 \varphi_k^2\, dx = 2\int_0^{1/n} (nx)^2\, dx = \frac{2}{3n}$$
$$\int_0^1 (\varphi_k')^2\, dx = 2\int_0^{1/n} n^2\, dx = 2n$$
$$\Rightarrow \quad a_{kk} = \frac{2}{3n} + 2n = \frac{4+12n^2}{6n}\ .$$



Berechnung von b_k :

$$b_k = -\int_0^1 x\varphi_k(x)\, dx$$
$$= -\int_{(k-1)/n}^{k/n} x(nx-(k-1))\, dx$$
$$-\int_{k/n}^{(k+1)/n} x(k+1-nx)\, dx$$
$$\Rightarrow \quad b_k = -\frac{k}{n^2}\ .$$



Nach Multiplikation mit $6n$ erhalten wir folgendes GLS

$$\begin{pmatrix} 4+12n^2 & 1-6n^2 & & & \\ 1-6n^2 & 4+12n^2 & 1-6n^2 & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & 1-6n^2 \\ & & & 1-6n^2 & 4+12n^2 \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_{n-2} \\ c_{n-1} \end{pmatrix} = -\frac{6}{n} \begin{pmatrix} 1 \\ 2 \\ \vdots \\ n-2 \\ n-1 \end{pmatrix}\ .$$

Die Koeffizientenmatrix ist eine symmetrische Tridiagonalmatrix.

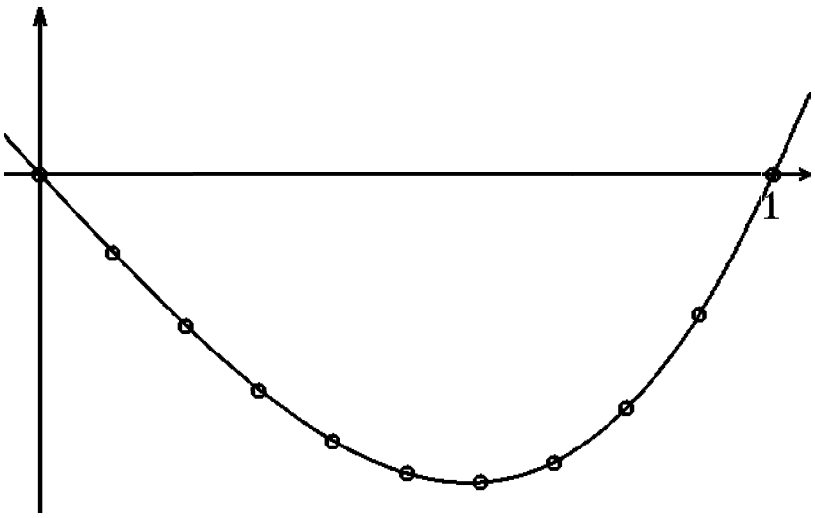
Die exakte Lösung der gegebenen Randwertaufgabe $y'' - y = x \quad , \quad y(0) = y(1) = 0$, lautet: $y(x) = \dfrac{\sinh(x)}{\sinh 1} - x$.

Ergebnis: Näherungslösung

Das lineare GLS $A\vec{c} = \vec{b}$ wurde für verschiedene n gelöst. In der folgenden Tabelle ist jeweils der maximale Fehler zwischen exakter Lösung $y(x)$ und Näherungslösung $P(x) = \sum_{i=1}^{n-1} c_i \varphi_i(x)$ an den Unterteilungsstellen x_i angegeben, also $\max_{1 \le i \le n-1} |y(x_i) - P(x_i)| = \max_{1 \le i \le n-1} |y(x_i) - c_i|$:

n	maximaler Fehler
4	$1.8 \cdot 10^{-4}$
10	$3.6 \cdot 10^{-5}$
50	$1.7 \cdot 10^{-6}$
100	$4.3 \cdot 10^{-7}$
200	$1.1 \cdot 10^{-7}$

Verbindet man die Punkte $\begin{pmatrix} x_i \\ c_i \end{pmatrix}$ mit Hilfe eines kubischen Splines, so erhält man den Graphen einer Näherungslösung, z.B. für $n = 10$:



Mehrdimensionale Probleme

Sei $G \subset \mathbb{R}^2$ ein *Greenscher Bereich*, d.h.: der Rand von G setze sich aus glatten Kurven zusammen, und sei

$M = \Big\{ u : G \to \mathbb{R} \ , \ u \quad 2-\text{mal stetig differenzierbar} \ , \ u_{|\partial G} = h \Big\}$

die Menge aller in G 2–mal stetig differenzierbaren Funktionen mit $u_{|\partial G} = h$.

Dann betrachten wir die Variationsaufgabe

$$\boxed{J(u) = \int_G F(x, y, u(x,y), u_x(x,y), u_y(x,y)) \ d(x,y) \ \text{ minimal } \ , \quad u \in M}$$

Hierbei sei F bzgl. aller Variablen 2–mal stetig partiell differenzierbar.

Sei u die *Minimalfunktion* und $v : G \to \mathbb{R}$ 2–mal stetig differenzierbar mit $v_{|\partial G} = 0$, dann gilt: $(u + \lambda v) \in M \quad \forall \lambda \in \mathbb{R}$.

Damit die Funktion

$\mu(\lambda) = J(u + \lambda v) = \int_G F(x, y, u + \lambda v, u_x + \lambda v_x, u_y + \lambda v_y) \ d(x,y)$

für u (also für $\lambda = 0$) minimal ist, muß gelten: $\mu'(0) = 0$, also

$\mu'(0) = \int_G (F_u \cdot v + F_{u_x} \cdot v_x + F_{u_y} \cdot v_y) \ d(x,y) = 0.$

Anstelle der partiellen Integration im 1-dimensionalen Fall benutzen wir nun den *Greenschen Satz* :

Mit $\vec{V} = \begin{pmatrix} V_1 \\ V_2 \end{pmatrix}$, $V_1 = -vF_{u_y}$, $V_2 = vF_{u_x}$ gilt nach dem *Greenschen Satz*

$$\int_G (V_{2x} - V_{1y}) \ d(x,y) = \int_{\partial G} V_1 dx + V_2 dy \ \text{ , also}$$

$$\int_G \Big(v_x F_{u_x} + v \frac{\partial}{\partial x}(F_{u_x}) + v_y F_{u_y} + v \frac{\partial}{\partial y}(F_{u_y}) \Big) \ d(x,y)$$

$$= \int_{\partial G} v \Big(-F_{u_y} dx + F_{u_x} dy \Big) = 0 \quad (\text{da } \ v_{|\partial G} = 0).$$

Also gilt

$$\int_G (F_{u_x} \cdot v_x + F_{u_y} \cdot v_y) \ d(x,y) = - \int_G v \Big(\frac{\partial}{\partial x}(F_{u_x}) + \frac{\partial}{\partial y}(F_{u_y}) \Big) \ d(x,y) \ .$$

Somit erhalten wir für $\mu'(0) = 0$:

$$\mu'(0) = \int_G \Big(F_u - \frac{\partial}{\partial x}(F_{u_x}) - \frac{\partial}{\partial y}(F_{u_y}) \Big) \, v \ d(x,y) = 0$$

für alle 2–mal stetig differenzierbaren Funktionen v mit $v_{|\partial G} = 0$.

Analog zum 1-dimensionalen Fall muß dann gelten:

$$\boxed{\frac{\partial}{\partial x}(Fu_x) + \frac{\partial}{\partial y}(Fu_y) - F_u = 0}$$

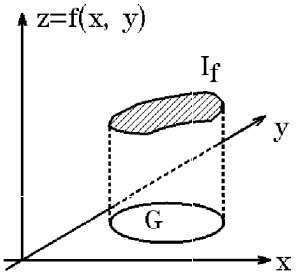
Das ist die zur Variationsaufgabe gehörende *Eulersche partielle DGL.*

Ergebnis:
Ist $u \in M$ Lösung der *Variationsaufgabe* $\Rightarrow$ u ist Lösung der zugehörigen *Eulerschen partiellen DGL.*

Beispiel *Minimalfläche*

Gegeben: Ein Greenscher Bereich $G \subset \mathbb{R}^2$ und eine stetige Randfunktion $h : \partial G \to \mathbb{R}$.

Gesucht: $f : G \to \mathbb{R}$ mit $f_{|\partial G} = h$,
für die die Fläche
$$I_f = \left\{ \begin{pmatrix} x \\ y \\ f(x,y) \end{pmatrix} : \begin{pmatrix} x \\ y \end{pmatrix} \in G \right\}$$
einen *minimalen* Flächeninhalt hat.



$\vec{\varphi}(x,y) = \begin{pmatrix} x \\ y \\ f(x,y) \end{pmatrix}$, $\begin{pmatrix} x \\ y \end{pmatrix} \in G$, ist eine Parameterdarstellung der Fläche I_f mit

$\vec{\varphi}_x = \begin{pmatrix} 1 \\ 0 \\ f_x \end{pmatrix}$, $\vec{\varphi}_y = \begin{pmatrix} 0 \\ 1 \\ f_y \end{pmatrix}$, $\vec{\varphi}_x \times \vec{\varphi}_y = \begin{pmatrix} -f_x \\ -f_y \\ 1 \end{pmatrix}$, $|\vec{\varphi}_x \times \vec{\varphi}_y| = \sqrt{1 + f_x^2 + f_y^2}$.

Damit beträgt der Flächeninhalt $\mu(I_f)$ der Fläche I_f

$$\mu(I_f) = \int_G \sqrt{1 + f_x^2 + f_y^2}\, d(x,y) \quad \text{minimal} \quad , \quad f_{|\partial G} = h \ .$$

Dies ist eine Variationsaufgabe mit $F(x,y,f,f_x,f_y) = \sqrt{1 + f_x^2 + f_y^2}$.
Die zugehörige Eulersche DGL ist eine nichtlineare partielle DGL.
Wir wollen das Problem annähern, indem wir die Funktion F durch das Taylorpolynom 2. Grades ersetzen. Es gilt für die Funktion $g(x,y) = \sqrt{1 + x^2 + y^2}$:

$$g_x(x,y) = \frac{x}{\sqrt{1 + x^2 + y^2}} \ , \ g_y(x,y) = \frac{y}{\sqrt{1 + x^2 + y^2}}$$

$$g_{xx}(x,y) = \frac{1}{\sqrt{1 + x^2 + y^2}} + x\Big(-\frac{x}{\sqrt{1 + x^2 + y^2}^3}\Big) \ , \ g_{xy}(x,y) = x\Big(-\frac{y}{\sqrt{1 + x^2 + y^2}^3}\Big) \ ,$$

$$g_{yy}(x,y) = \frac{1}{\sqrt{1 + x^2 + y^2}} + y\Big(-\frac{y}{\sqrt{1 + x^2 + y^2}^3}\Big)$$

$\Rightarrow$ $g(0,0) = 1$, $g_x(0,0) = 0$, $g_y(0,0) = 0$,
$g_{xx}(0,0) = 1$, $g_{xy}(0,0) = 0$, $g_{yy}(0,0) = 1$.

Damit lautet die Taylorentwicklung von g um $(0,0)$:

$g(x,y) = 1 + \frac{1}{2}(x^2+y^2) + R$.

Also gilt für die Funktion F :

$F(x,y,f,f_x,f_y) = \sqrt{1+f_x^2+f_y^2} \approx 1+\frac{1}{2}(f_x^2+f_y^2)$ bei kleinen Deformationen f_x, f_y .

Damit erhält man die folgende angenäherte Variationsaufgabe

$\mu(I_f) \approx \int_G (1+\frac{1}{2}(f_x^2+f_y^2))\ d(x,y)$ minimal , $f_{|\partial G} = h$.

Die zugehörige *Eulersche partielle DGL* lautet

$$\frac{\partial}{\partial x}(F_{f_x}) + \frac{\partial}{\partial y}(F_{f_y}) - F_f = \frac{\partial}{\partial x}(f_x) + \frac{\partial}{\partial y}(f_y) = f_{xx} + f_{yy} = \Delta f = 0 \ .$$

Also liefert die Lösung des Dirichlet-Problems $\Delta f = 0$, $f_{\partial G} = h$, (näherungsweise) die *Minimalfläche* über G mit der Randbedingung $f_{|\partial G} = h$.

Wir wollen nun den Zusammenhang zwischen *Variationsaufgabe* und zugehöriger *Eulerscher partieller DGL* folgendermaßen nutzen:

Um eine gegebene partielle Randwertaufgabe (d.h: partielle DGL mit Randbedingung) zu lösen, suchen wir zunächst die zugehörige Variationsaufgabe, d.h.: die zur Variationsaufgabe gehörende Eulersche DGL soll gleich der gegebenen partiellen DGL sein. Dann lösen wir näherungsweise die Variationsaufgabe und erhalten somit eine Näherungslösung der gegebenen partiellen Randwertaufgabe.

Beispiel *Elliptische DGL*

Gegeben:

$$\boxed{u_{xx} + u_{yy} + \varrho u = a \quad \text{in } G \ , \ u_{|\partial G} = h}$$

Zugehörige *Variationsaufgabe*

$$\boxed{J(u) = \int_G \Big(\frac{1}{2}(u_x^2+u_y^2) - \frac{1}{2}\varrho u^2 + au\Big)\ d(x,y) \quad \text{minimal} \ , \ u_{|\partial G} = h}$$

denn mit $F(x,y,u,u_x,u_y) = \frac{1}{2}(u_x^2+u_y^2) - \frac{1}{2}\varrho u^2 + au$ gilt die *Eulersche DGL*

$$\frac{\partial}{\partial x}(F_{u_x}) + \frac{\partial}{\partial y}(F_{u_y}) - F_u = \frac{\partial}{\partial x}(u_x) + \frac{\partial}{\partial y}(u_y) - (-\varrho u + a) = u_{xx} + u_{yy} + \varrho u - a = 0$$

$$\Rightarrow \quad u_{xx} + u_{yy} + \varrho u = a \ .$$

Zur Berechnung der Näherungslösung des Variationsproblems erweitern wir wieder den Funktionenraum M :

$$M = \Big\{ u : G \to I\!R \ , \ u \text{ stetig, stückweise stetig differenzierbar } , \ u_{|\partial G} = h \Big\} \ .$$

Dann ist allerdings mathematisch nicht mehr sicher, ob Lösungen des Variationsproblems auch Lösungen der zugehörigen Eulerschen DGL sind. Oft kann man aber das physikalische Problem direkt als Variationsproblem formulieren. Dann entfällt diese (mathematische) Problematik.

Näherungslösung des Variationsproblems

Wir legen wieder ein Netz über den Grundbereich G und betrachten geeignete *Grundfunktionen* $\varphi_i(x,y)$ über den Teilbereichen. Dann nähern wir die gesuchte Lösung u durch $u(x,y) \approx \sum_{i=1}^{n} c_i \varphi_i(x,y)$ an. Wir erhalten dann

$$L(c_1,\ldots,c_n) = \int_G F(x,y,\sum_{i=1}^{n} c_i\varphi_i, \sum_{i=1}^{n} c_i\varphi_{ix}, \sum_{i=1}^{n} c_i\varphi_{iy}) \ d(x,y) \quad \text{minimal} \ .$$

Für das Minimum muß gelten: $\dfrac{\partial L}{\partial c_k} = 0 \quad \forall k = 1,\ldots,n$.

Damit erhalten wir ein GLS für die unbekannten Koeffizienten c_i .

Beispiel

$$J(u) = \int_G \Big(\frac{1}{2}(u_x^2 + u_y^2) - \frac{1}{2}\varrho u^2 + au \Big) \ d(x,y) \quad \text{minimal} \quad \Rightarrow$$

$$L(\vec{c}) = \int_G \Big(\frac{1}{2}\Big(\sum_{i=1}^{n} c_i\varphi_{ix}\Big)^2 + \frac{1}{2}\Big(\sum_{i=1}^{n} c_i\varphi_{iy}\Big)^2 - \frac{1}{2}\varrho\Big(\sum_{i=1}^{n} c_i\varphi_i\Big)^2 + a\sum_{i=1}^{n} c_i\varphi_i \Big) \ d(x,y)$$

$$\frac{\partial L}{\partial c_k} = \int_G \Big(\sum_{i=1}^{n} c_i\varphi_{ix}\varphi_{kx} + \sum_{i=1}^{n} c_i\varphi_{iy}\varphi_{ky} - \varrho\sum_{i=1}^{n} c_i\varphi_i\varphi_k + a\varphi_k \Big) \ d(x,y)$$

$$= \sum_{i=1}^{n} c_i \int_G \Big\{ \varphi_{ix}\varphi_{kx} + \varphi_{iy}\varphi_{ky} - \varrho\varphi_i\varphi_k \Big\} \ d(x,y) + \int_G a\varphi_k \ d(x,y) = 0$$

$$\Rightarrow \quad \sum_{i=1}^{n} a_{ki}c_i = b_k \quad \forall k = 1,\ldots,n \quad \Rightarrow \quad A\vec{c} = \vec{b} \ \text{ mit}$$

$$\boxed{a_{ki} = a_{ik} = \int_G \varphi_{ix}\varphi_{kx} d(x,y) + \int_G \varphi_{iy}\varphi_{ky} d(x,y) - \int_G \varrho\varphi_i\varphi_k d(x,y)}$$

$$\boxed{b_k = -\int_G a\varphi_k d(x,y)}$$

Nach Berechnung dieser Integrale (z.B. numerisch mit Gaußschen Quadraturformeln (vgl. Literatur)) erhält man das lineare GLS $A\vec{c} = \vec{b}$. Die Lösung dieses GLS liefert die gesuchten Koeffizienten $c_1,\ldots,c_n$.

Wahl der Grundfunktionen

Beispiel 1 *"Lineare" Grundfunktionen*

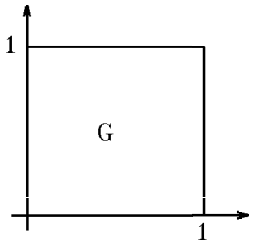
Gegeben: *Randwertaufgabe auf einem Rechteck mit homogener Randbedingung*

$u_{xx} + u_{yy} + \varrho u = a \quad$ in

$G = \left\{ \begin{pmatrix} x \\ y \end{pmatrix} \; : \; 0 < x < 1 \; , \; 0 < y < 1 \right\}$

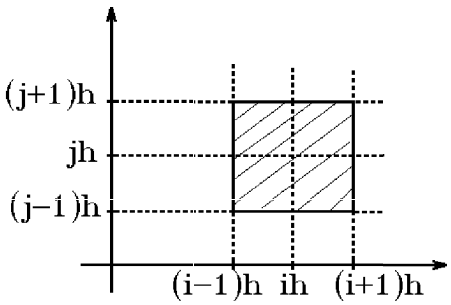
mit der homogenen Randbedingung

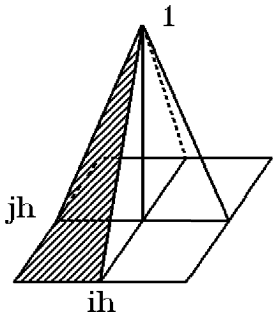
$u_{|\partial G} = 0$.



Legen wir über G ein Rechteckgitter mit gleicher Schrittweite h in $x-$ und $y-$Richtung, so können wir folgende *Grundfunktionen* $\varphi_{ij}(x,y)$, $(1 \le i,j \le n-1)$, wählen:

Schrittweite in $x-$ und $y-$Richtung: $h = \dfrac{1}{n}$, $x_i = ih$, $y_j = jh$, $(0 \le i,j \le n)$





"lineare" Grundfunktion φ_{ij}

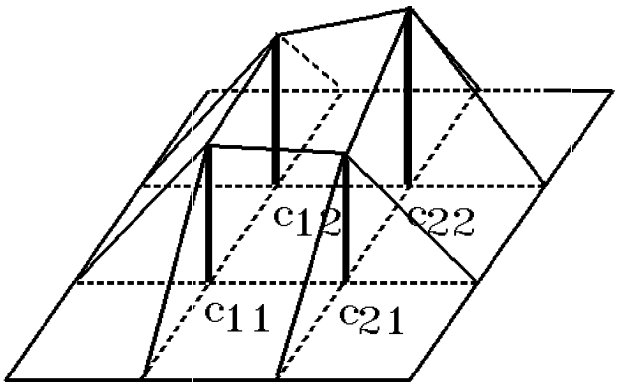
Außerhalb des schraffierten Quadrats sei die Grundfunktion $\varphi_{ij}(x,y) = 0$.

$$\varphi_{ij}(x,y) = \begin{cases} \dfrac{1}{h^2}(x-(i-1)h)(y-(j-1)h) & \text{,falls } (i-1)h \le x \le ih \\ & \quad \text{und } (j-1)h \le y \le jh \\ \dfrac{1}{h^2}(x-(i-1)h)((j+1)h-y) & \text{,falls } (i-1)h \le x \le ih \\ & \quad \text{und } jh \le y \le (j+1)h \\ \dfrac{1}{h^2}((i+1)h-x)(y-(j-1)h) & \text{,falls } ih \le x \le (i+1)h \\ & \quad \text{und } (j-1)h \le y \le jh \\ \dfrac{1}{h^2}((i+1)h-x)((j+1)h-y) & \text{,falls } ih \le x \le (i+1)h \\ & \quad \text{und } jh \le y \le (j+1)h \\ 0 & \text{sonst} \end{cases}$$

Für die Annäherung $P(x,y) = \sum_{i,j} c_{ij}\varphi_{ij}(x,y)$ an $u(x,y)$ gilt dann:

$P(x_i, y_j) = c_{ij} \approx u(x_i, y_j) \quad \forall 1 \leq i,j \leq n-1$. Also sind die gesuchten Koeffizienten c_{ij} Näherungswerte für die Lösungsfunktion u an den Stellen (x_i, y_j) .

Der Graph von $P(x,y)$ ergibt geometrisch eine "Zeltdachkonstruktion"



Sind die Grundfunktionen φ_{ij} und φ_{kl} nicht benachbart, so gilt $\varphi_{ij} \cdot \varphi_{kl} \equiv 0$. Für benachbarte Grundfunktionen kann man die Integrale

$$a_{ki} = a_{ik} = \int_G \varphi_{ix}\varphi_{kx} d(x,y) + \int_G \varphi_{iy}\varphi_{ky} d(x,y) - \int_G \varrho\varphi_i\varphi_k d(x,y)$$

und

$$b_k = -\int_G a\varphi_k d(x,y)$$

leicht berechnen, falls ϱ und a konstant sind.

Beispiel hierzu $\varrho = 0$ und $a \in \mathbb{R}$, also $u_{xx} + u_{yy} = a$ in G , $u_{|\partial G} = 0$.

Für benachbarte Grundfunktionen φ_{ij} , φ_{kl} erhalten wir folgende Integrale:

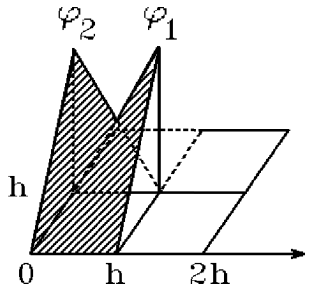
mit $\varphi_1(x,y) = \dfrac{xy}{h^2}$, $0 \leq x,y \leq h$

$\varphi_2(x,y) = \dfrac{(h-x)y}{h^2}$, $0 \leq x,y \leq h$

gilt

$\varphi_{1x} = \dfrac{y}{h^2}$, $\varphi_{1y} = \dfrac{x}{h^2}$

$\varphi_{2x} = \dfrac{-y}{h^2}$, $\varphi_{2y} = \dfrac{h-x}{h^2}$,



also gilt für benachbarte $\varphi_{ij}-$Grundfunktionen:

$$\int_0^1\int_0^1 \varphi_{ij_x}\varphi_{kl_x}\,dxdy = \frac{2}{h^4}\int_0^h\int_0^h (-y^2)\,dxdy = \frac{2}{h^3}(-\frac{h^3}{3}) = -\frac{2}{3}$$
$$\int_0^1\int_0^1 \varphi_{ij_y}\varphi_{kl_y}\,dxdy = \frac{2}{h^4}\int_0^h\int_0^h (xh-x^2)\,dydx = \frac{2}{h^3}(\frac{h^3}{2}-\frac{h^3}{3}) = \frac{1}{3}\ ,$$

zusammen ergibt das, falls zwei Grundfunktionen benachbart sind und $\varrho = 0$ ist:

$$= -\frac{1}{3}\ .$$

Für die Diagonalelemente der Matrix A erhalten wir, falls $\varrho = 0$ ist:

$$\int_0^1\int_0^1 (\varphi_{ij_x})^2\,dxdy + \int_0^1\int_0^1 (\varphi_{ij_y})^2\,dxdy = 2\int_0^1\int_0^1 (\varphi_{ij_x})^2\,dxdy$$
$$= 8\int_0^h\int_0^h \frac{y^2}{h^4}\,dxdy = \frac{8}{h^3}\cdot\frac{h^3}{3} = \frac{8}{3}\ .$$

Für die Koordinaten des Vektor $\vec{b}$ der rechten Seite des GLS $A\vec{c} = \vec{b}$ erhalten wir, falls $a \in \mathbb{R}$ (konstant):

$$b_k = -a\int_0^1\int_0^1 \varphi_{ij}\,dxdy = -4a\int_0^h\int_0^h \frac{xy}{h^2}\,dxdy = -\frac{4a}{h^2}\Big(\int_0^h x\,dx\Big)^2$$
$$= -\frac{4a}{h^2}\Big(\frac{h^2}{2}\Big)^2 = -ah^2$$
$$\Rightarrow \quad b_k = -ah^2 \quad \forall\ k\ .$$

Damit erhalten wir für den Fall, daß $\varrho = 0$ und $a \in \mathbb{R}$ ist, nach Multiplikation mit dem Faktor 3, folgendes GLS :

$$\begin{pmatrix}
8 & -1 & & & & -1 & -1 & & & & \\
-1 & 8 & -1 & & & -1 & -1 & -1 & & & \\
 & \ddots & \ddots & \ddots & & & \ddots & \ddots & \ddots & & \\
 & & -1 & 8 & -1 & & & -1 & -1 & -1 & \\
 & & & -1 & 8 & & & & -1 & -1 & \\
-1 & -1 & & & & 8 & -1 & & & & \ddots \\
-1 & -1 & -1 & & & -1 & 8 & -1 & & & \\
 & \ddots & \ddots & \ddots & & & \ddots & \ddots & \ddots & & \\
 & & & & & & & & & & \\
 & & & \ddots & & & & & \ddots & & \ddots
\end{pmatrix}
\begin{pmatrix} c_{11} \\ c_{12} \\ \vdots \\ c_{1,n-2} \\ c_{1,n-1} \\ c_{21} \\ c_{22} \\ \vdots \\ \\ \vdots \end{pmatrix} = \vec{b}$$

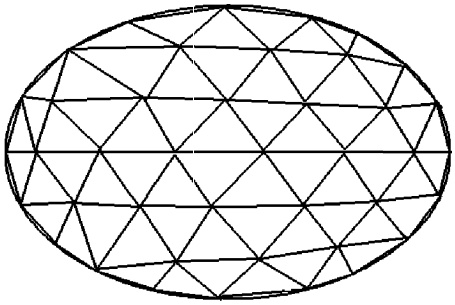
mit $\vec{b} = -3ah^2(1,1,1,\ldots,1)^T$, $h = \frac{1}{n}$.

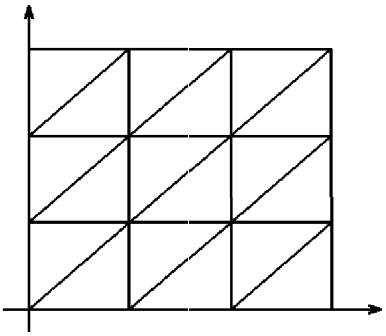
Die Koeffizientenmatrix A ist eine *symmetrische, positiv definte, fast diagonaldominante Bandmatrix* der Bandbreite n . Das GLS kann für kleine n mit dem *Cholesky-Verfahren* und für große n mit dem *Überrelaxationsverfahren* oder besser noch mit dem *pcg-Verfahren* gelöst werden.

Die Lösungen c_{ij} des GLS ergeben Näherungswerte für die Lösungsfunktion u der Randwertaufgabe $u_{xx}+u_{yy} = a$, $u_{|\partial G} = 0$, an den inneren Gitterpunkten (x_i, y_j) , also $c_{ij} \approx u(x_i, y_j) = u(ih, jh)$ (Ergebnisse zu diesem Beispiel siehe später S.126).

Beispiel 2 *Triangulierung*

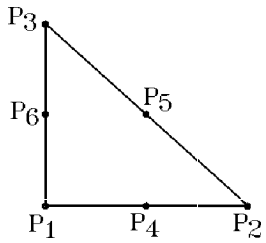
Für die meisten Grundgebiete ist eine *Triangulierung* am besten geeignet, also ein Gitter aus Dreiecken, z.B.:





Dann betrachtet man auf jedem Dreieck *Grundfunktionen* in Form eines Polynoms, z.B.: 2.Grades.

Beispiel *quadratische Grundfunktionen*



Auf jedem Dreieck wählen wir 6 Punkte
$P_1, P_2, \ldots, P_6$ in folgender Reihenfolge:

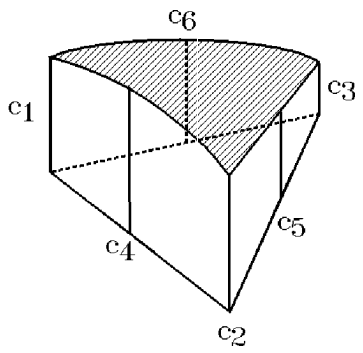
Dann wählen wir für jedes Dreieck
6 *Grundfunktionen* φ_i , $(i = 1, 2, \ldots, 6)$
mit folgenden Eigenschaften:

$$\varphi_i(P_i) = 1 \ , \ \varphi_i(P_j) = 0 \ \text{ für } \ j \neq i$$

und $\varphi_i(x) = a_i + b_i x + c_i y + d_i x^2 + e_i y^2 + f_i xy$ (Polynom 2.Grades) ,
außerhalb des Dreiecks $= 0$.

Diese 6 *Grundfunktionen* lauten für folgendes *Normdreieck*

$$\begin{aligned}
\varphi_1(x,y) &= (1-x-y)(1-2x-2y) \\
\varphi_2(x,y) &= x(2x-1) \\
\varphi_3(x,y) &= y(2y-1) \\
\varphi_4(x,y) &= 4x(1-x-y) \\
\varphi_5(x,y) &= 4xy \\
\varphi_6(x,y) &= 4y(1-x-y)
\end{aligned}$$

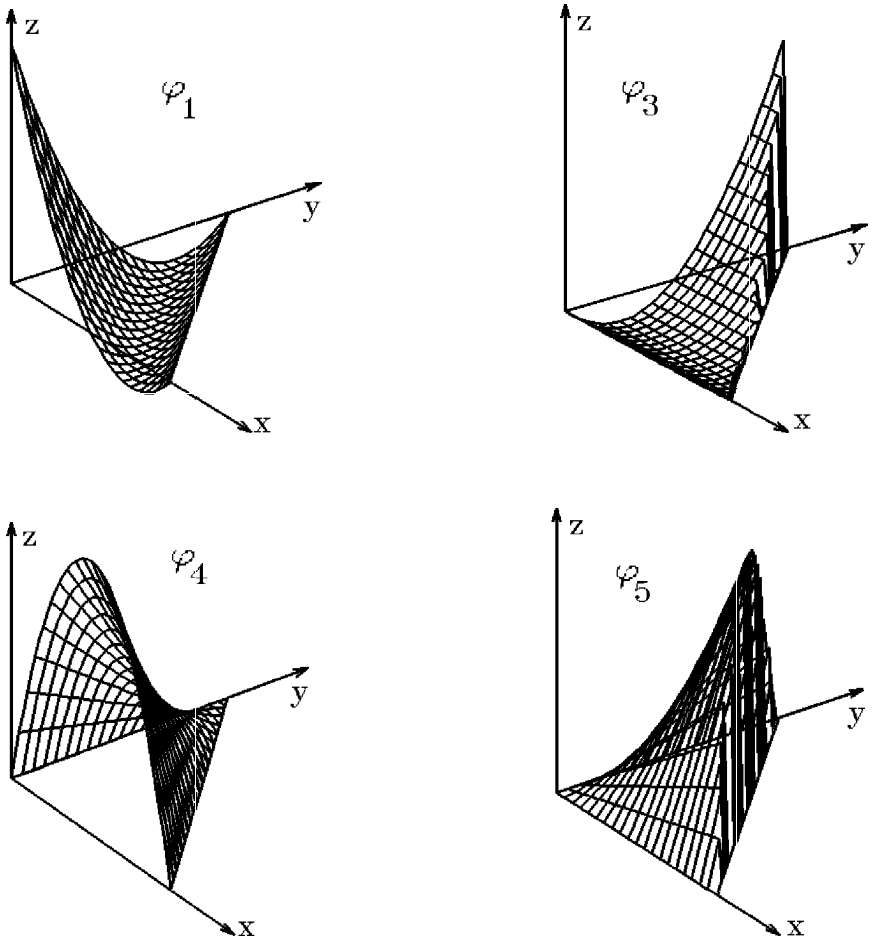


Bildet man

$$p(x,y) = \sum_{i=1}^{6} c_i \varphi_i(x,y) \ ,$$

so erhält man: $p(P_i) = c_i$.

Die Funktionen $p(x,y)$ für benachbarte Dreiecke stoßen *stetig* aneinander, da sie auf der gemeinsamen Dreiecksseite in 3 Punkten übereinstimmen (denn ein Polynom 2.Grades von 1 Variablen ist durch 3 Punkte eindeutig bestimmt).



einige Grundfunktionen für das Dreieck

Die einzelnen Dreiecke der *Triangulierung* können durch eine einfache Transformation auf das *Normdreieck* zurückgeführt werden. Addiert man für alle Dreiecke die Anzätze $\sum_{i=1}^{6} c_i \varphi_i(x,y)$, so erhält man insgesamt:

$$u(x,y) \approx \sum_{i=1}^{N} c_i \varphi_i(x,y) \quad \text{mit} \quad u(P_i) \approx c_i \ .$$

Hierbei müssen die Stützpunkte P_i auf den Dreiecksseiten geeignet durchnumeriert werden.

Beispiel

Gegeben: Randwertaufgabe

$u_{xx} + u_{yy} + \varrho u = a \quad , \quad u_{|\partial G} = h \ .$

Die gegebene DGL ist Eulersche DGL zu folgender Variationsaufgabe:

$J(u) = \int_G \Big(\frac{1}{2}(u_x^2 + u_y^2) - \frac{1}{2}\varrho u^2 + au \Big) \, d(x,y)$ minimal , $u_{|\partial G} = h$ (vgl. S.112).

Wir lösen diese Variationsaufgabe näherungsweise. Der Ansatz $u \approx \sum_i c_i \varphi_i$ führt dann auf das lineare GLS $A\vec{c} = \vec{b}$ mit

$a_{ij} = \int_G (\varphi_{ix}\varphi_{jx} + \varphi_{iy}\varphi_{jy}) \, d(x,y) - \int_G \varrho\varphi_i\varphi_j \, d(x,y)$ und

$b_i = -\int_G a\varphi_i \, d(x,y)$ (vgl. S.113).

Wir setzen ab jetzt voraus, daß ϱ und a *konstant* sind, also

Voraussetzung: $\varrho \in \mathbb{R}$, $a \in \mathbb{R}$ (*konstant*)

Dann gilt

$a_{ij} = \int_G (\varphi_{ix}\varphi_{jx} + \varphi_{iy}\varphi_{jy}) \, d(x,y) - \varrho \int_G \varphi_i\varphi_j \, d(x,y)$ und

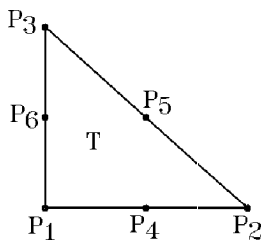
$b_i = -a \int_G \varphi_i \, d(x,y) \ .$

Also müssen für jedes Dreieck T bzgl. seiner 6 Grundfunktionen φ_i die folgenden Integrale berechnet werden:

(1) $s_{ij} = \int_T (\varphi_{ix}\varphi_{jx} + \varphi_{iy}\varphi_{jy}) \, d(x,y) \quad , \quad (1 \le i,j \le 6)$

(2) $m_{ij} = \int_T \varphi_i\varphi_j \, d(x,y) \quad , \quad (1 \le i,j \le 6)$

(3) $e_i = \int_T \varphi_i \, d(x,y) \quad , \quad (1 \le i \le 6).$



Für das *Normdreieck* T erhält man durch einfaches Nachrechnen:

(1) *Steifigkeitsmatrix* (Integrale s_{ij})

$$S = \frac{1}{6} \begin{pmatrix} 6 & 1 & 1 & -4 & 0 & -4 \\ 1 & 3 & 0 & -4 & 0 & 0 \\ 1 & 0 & 3 & 0 & 0 & -4 \\ -4 & -4 & 0 & 16 & -8 & 0 \\ 0 & 0 & 0 & -8 & 16 & -8 \\ -4 & 0 & -4 & 0 & -8 & 16 \end{pmatrix}$$

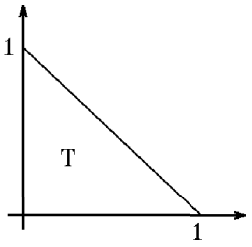
(2) *Massenelementmatrix* (Integrale m_{ij})

$$M = \frac{J}{360}\begin{pmatrix} 6 & -1 & -1 & 0 & -4 & 0 \\ -1 & 6 & -1 & 0 & 0 & -4 \\ -1 & -1 & 6 & -4 & 0 & 0 \\ 0 & 0 & -4 & 32 & 16 & 16 \\ -4 & 0 & 0 & 16 & 32 & 16 \\ 0 & -4 & 0 & 16 & 16 & 32 \end{pmatrix}$$

(3) *Elementvektor* (Integrale e_i)

$\vec{e} = \frac{J}{6}(0,0,0,1,1,1)^T$ mit:

J ist die *doppelte Dreiecksfläche*.



Z.B.:

$$e_2 = \int_T \varphi_2 \; d(x,y) = \int_0^1 \int_0^{1-x} x(2x-1) \; dydx$$

$$= \int_0^1 x(2x-1)(1-x) \; dx = \int_0^1 (3x^2 - 2x^3 - x) \; dx = 1 - \frac{2}{4} - \frac{1}{2} = 0 \; ,$$

$$e_5 = \int_T \varphi_5 \; d(x,y) = \int_0^1 \int_0^{1-x} 4xy \; dydx = 2\int_0^1 x(1-x)^2 \; dx$$

$$= 2\int_0^1 (x^3 - 2x^2 + x) \; dx = 2(\frac{1}{4} - \frac{2}{3} + \frac{1}{2}) = \frac{1}{6} \quad (J = 1 \text{ doppelte Dreiecksfläche}).$$

Die Matrizen S, M und der Vektor $\vec{e}$ hängen nicht von der Lage der Dreiecke ab, sondern nur von der Numerierung der Stützpunkte.

Hat man den Grundbereich G in Dreiecke zerlegt, so muß man die zugehörigen Stützpunkte auf den Dreiecksseiten in geeigneter Form durchnumerieren und dann mit Hilfe der Matrizen S und M die Koeffizientenmatrix A und mit Hilfe des Vektors $\vec{e}$ die rechte Seite des GLS $A\vec{c} = \vec{b}$ für die Unbekannten c_i bestimmen. Dabei muß beachtet werden, daß für Stützpunkte, die auf dem *Rand des Gebietes* G liegen, die Werte c_i durch die Randbedingung $u_{|\partial G} = h$ vorgegeben sind. Diese *Randpunkte* behandelt man am besten zunächst wie alle inneren Stützpunkte, d.h.: man *numeriert sie mit*. Dann erhält man die Unbekannten $\vec{c} = (c_1, \ldots, c_M)^T$, eine Koeffizientenmatrix $\tilde{A}$ und eine rechte Seite $\vec{\tilde{b}}$. Das GLS $\tilde{A}\vec{c} = \vec{\tilde{b}}$ enthält dann im Unbekanntenvektor $\vec{c}$ auch die (bekannten) *Randpunkte*.

Ist P_k ein *Randpunkt*, so subtrahiert man das $h(P_k)-$fache der $k-$ten Spalte von der rechten Seite und setzt dann alle Elemente (bis auf das Diagonalelement) der $k-$ten Zeile und $k-$ten Spalte gleich 0, das Diagonalelement $\tilde{a}_{kk} = 1$ und das Element $\tilde{b}_k = h(P_k)$.

Auf diese Weise erhält man das neue GLS $A\vec{c} = \vec{b}$ mit den trivialen Gleichungen $1 \cdot c_k = h(P_k)$ für die Randpunkte. Das hat den Vorteil, daß dann der Lösungsvektor $\vec{c}$ alle Näherungswerte $u(P_i)$ einschließlich der Randpunkte enthält.

Beispiele hierzu

Beispiel 1

Gegeben: Randwertaufgabe

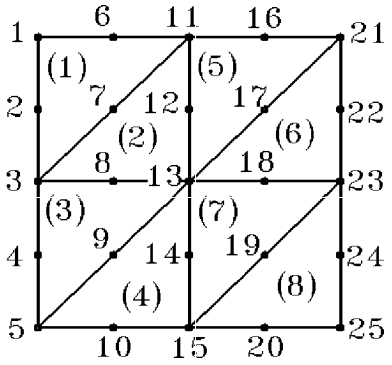
$$u_{xx} + u_{yy} = 0 \quad \text{in } G = \left\{ \begin{pmatrix} x \\ y \end{pmatrix} \; : \; 0 < x, y < 1 \right\}$$
$$u_{|\partial G} = h(x,y) = x^2 - y^2$$

$\Rightarrow \quad \varrho = a = 0$, also bleiben die Matrix M und der Vektor $\vec{e}$ unberücksichtigt.

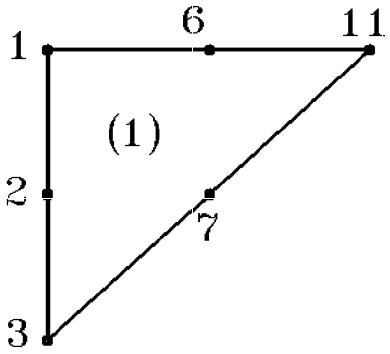
Die *exakte Lösung* lautet: $u(x,y) = x^2 - y^2$.

Für die Berechnung einer Näherungslösung führen wir zunächst eine *Triangulierung* durch:

Zerlegung in 8 Dreiecke



z.B.: Dreieck (1)



Reihenfolge der Punkte: $1, 3, 11, 2, 7, 6$

Reihenfolge der Punkte für die einzelnen Dreiecke:

(1) :	$1,\ 3, 11,\ 2,\ 7,\ 6$	(5) :	$11, 13, 21, 12, 17, 16$
(2) :	$13, 11,\ 3, 12,\ 7,\ 8$	(6) :	$23, 21, 13, 22, 17, 18$
(3) :	$3,\ 5, 13,\ 4,\ 9,\ 8$	(7) :	$13, 15, 23, 14, 19, 18$
(4) :	$15, 13,\ 5, 14,\ 9, 10$	(8) :	$25, 23, 15, 24, 19, 20$

Randpunkte:

$1, 2, 3, 4, 5, 6, 10, 11, 15, 16, 20, 21, 22, 23, 24, 25.$

Beispiel für das Aufstellen der Matrix A :

7.−te Zeile:

Der Punkt 7 liegt auf dem Dreieck (1) und (2) jeweils als 5. Punkt, also erhält man aus der Matrix S die beiden Gleichungen (jeweils die 5. Zeile):

$-8c_2 + 16c_7 - 8c_6 = 0$
$-8c_{12} + 16c_7 - 8c_8 = 0.$

Addition dieser beiden Gleichungen ergibt: $\underbrace{-8c_2 - 8c_6}_{\text{Randpunkte}} + 32c_7 - 8c_8 - 8c_{12} = 0.$

Werden die Randpunkte auf die rechte Seite gebracht, so erhält man:

$32c_7 - 8c_8 - 8c_{12} = 8h(P_2) + 8h(P_6)$ (dies ist die 7. Zeile des GLS $A\vec{c} = \vec{b}$).

8.−te Zeile:

Der Punkt 8 liegt auf dem Dreieck (2) und (3) jeweils als 6. Punkt, also erhält man aus der Matrix S die beiden Gleichungen (jeweils die 6. Zeile):

$-4c_{13} - 4c_3 - 8c_7 + 16c_8 = 0$
$-4c_3 - 4c_{13} - 8c_9 + 16c_8 = 0.$

Addition dieser beiden Gleichungen ergibt: $\underbrace{-8c_3}_{\text{Randpunkt}} - 8c_7 + 32c_8 - 8c_9 - 8c_{13} = 0.$

Wird der Randpunkt auf die rechte Seite gebracht, so erhält man:

$-8c_7 + 32c_8 - 8c_9 - 8c_{13} = 8h(P_3)$ (dies ist die 8. Zeile des GLS $A\vec{c} = \vec{b}$).

Insgesamt erhalten wir folgende symmetrische Koeffizientenmatrix A

$$
\begin{array}{cccccc|ccc|ccc|ccc|ccc|cccccc}
1 & & & & & & & & & & & & & & & & & & & & & & & \\
0 & 1 & & & & & & & & & & & & & & & & & & & & & & \\
0 & 0 & 1 & & & & & & & & & & & & & & & & & & & & & \\
0 & 0 & 0 & 1 & & & & & & & & & & & & & & & & & & & & \\
0 & 0 & 0 & 0 & 1 & & & & & & & & & & & & & & & & & & & \\
0 & 0 & 0 & 0 & 0 & 1 & & & & & & & & & & & & & & & & & & \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & 32 & & & & & & & & & & & & & & & & & \\
0 & 0 & 0 & 0 & 0 & 0 & -8 & 32 & & & & & & & & & & & & & & & & \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -8 & 32 & & & & & & & & & & & & & & & \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & & & & & & & & & & & & & & \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & & & & & & & & & & & & & \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & -8 & 0 & 0 & 0 & 0 & 32 & & & & & & & & & & & & \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -8 & 0 & 0 & 0 & -8 & 24 & & & & & & & & & & & \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -8 & 0 & 0 & 0 & -8 & 32 & & & & & & & & & & \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & & & & & & & & & \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & & & & & & & & \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -8 & 0 & 0 & 0 & 0 & 32 & & & & & & & \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -8 & 0 & 0 & 0 & -8 & 32 & & & & & & \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -8 & 0 & 0 & 0 & -8 & 32 & & & & & \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & & & & \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & & & \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & & \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{array}
$$

Die rechte Seite des GLS $A\vec{c} = \vec{b}$ lautet in diesem Beispiel:

```
-1.0000 -0.5625 -0.2500 -0.0625  0.0000 -0.9375-12.0000
-2.0000  0.0000  0.0625 -0.7500 -6.0000  0.0000  2.0000
 0.2500 -0.4375  0.0000  6.0000 12.0000  0.5625  0.0000
 0.4375  0.7500  0.9375  1.0000
```

Das GLS $A\vec{c} = \vec{b}$ enthält die trivialen Gleichungen $1 \cdot c_i = h(P_i)$ für die Randpunkte.

Die Koeffizientenmatrix A ist eine *symmetrische, positiv definte, fast diagonaldominante Bandmatrix*. Also kann das GLS für kleine n mit dem *Cholesky-Verfahren* und für große n mit dem *Überrelaxations-* oder *pcg-Verfahren* gelöst werden.

Ergebnis für dieses Beispiel:

Die Ergebnisse sind in der gleichen Reihenfolge wie die Lage der Gitterpunkte aufgelistet:

```
-1.00000000  -0.93750000  -0.75000000  -0.43750000   0.00000000
-0.56250000  -0.50000000  -0.31250000  -0.00000000   0.43750000
-0.25000000  -0.18750000   0.00000000   0.31250000   0.75000000
-0.06250000   0.00000000   0.18750000   0.50000000   0.93750000
 0.00000000   0.06250000   0.25000000   0.56250000   1.00000000
```

Der maximale Fehler zwischen Finite-Elemente–Lösung und exakter Lösung beträgt: $9.1 \cdot 10^{-13}$.

Bemerkung :

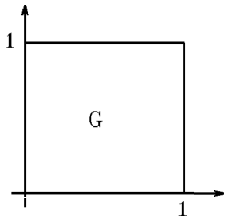
Es existieren Algorithmen, um eine optimale Numerierung der Gitterpunkte zu erhalten, damit die Bandbreite der Bandmatrix A möglichst klein ist.

Ebenfalls existieren Algorithmen, die die Koeffizientenmatrix A und die rechte Seite $\vec{b}$ erstellen.

Für unser Beispiel

$u_{xx} + u_{yy} = a \quad \text{in} \quad G$

$u_{|\partial G} = h$



könnte man die Matrix A und die rechte Seite $\vec{b}$ folgendermaßen aufstellen:

Erzeugung des GLS $A\vec{c} = \vec{b}$:

Gegeben:

n Anzahl der Gitterpunkte (einschließlich Randpunkte)

p Anzahl der Randpunkte
d Anzahl der Dreiecke
D_i doppelte Dreiecksfläche für das $i-$te Dreieck
a rechte Seite der DGL
s_{ij} Elemente der Matrix $6S$
e_i Elemente des Vektors $(0,0,0,1,1,1)^T$
c_{ij} , $(j = 1, \ldots, 6)$, Indizes der 6 Punkte des $i-$ten Dreiecks in der richtigen Reihenfolge
r_i Indizes der Randpunkte
$w_i = h(P_{ri})$ Wert im Randpunkt P_{ri} .

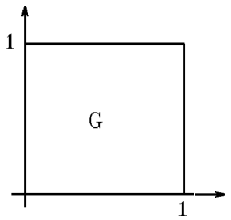
Vektor $\vec{b}$ und Matrix A mit Nullen vorbelegen
für $i = 1$ bis d
  für $j = 1$ bis 6
    $l = c_{ij}$, $b_l = b_l - D_i * a * e_j$
    für $k = j$ bis 6
      $l = c_{ij}$, $m = c_{ik}$
      falls $l > m$ $\Rightarrow$ m und l vertauschen
      $a_{lm} = a_{lm} + s_{jk}$
    Ende $k-$Schleife
  Ende $j-$Schleife
Ende $i-$Schleife
für $i = 1$ bis n (Symmetrie von A)
  für $j = i$ bis n : $a_{ji} = a_{ij}$: Ende $j-$Schleife
Ende $i-$Schleife
für $i = 1$ bis p ($w_i * r_i-$te Spalte von der rechten Seite subtrahieren)
  für $j = 1$ bis n : $b_j = b_j - w_i * a_{j,r_i}$: Ende $j-$Schleife
Ende $i-$Schleife
für $i = 1$ bis p (triviale Gleichungen erzeugen)
  $b_{r_i} = w_i$
  für $j = 1$ bis n : $a_{r_i,j} = 0$, $a_{j,r_i} = 0$: Ende $j-$Schleife
  $a_{r_i,r_i} = 1$
Ende $i-$Schleife

Beispiele hierzu

Beispiel 1

$u_{xx} + u_{yy} = 2$ in $G = \left\{ \begin{pmatrix} x \\ y \end{pmatrix} : 0 < x, y < 1 \right\}$

$u_{|\partial G} = 0.$

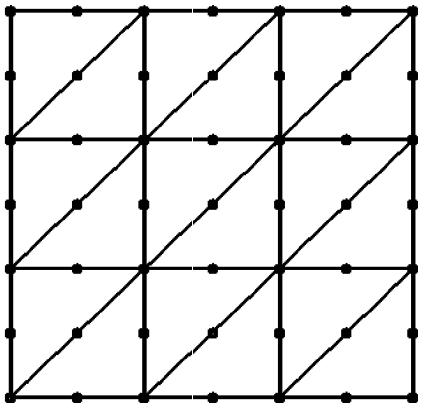


Exakte Lösung:

$$u(x,y) = \frac{4}{\pi^3}\sum_{k=0}^{\infty}(\sinh(2k+1)\pi(1-y)+\sinh(2k+1)\pi y)\frac{\sin(2k+1)\pi x}{(2k+1)^3\sinh(2k+1)\pi}$$
$$+\frac{4}{\pi^3}\sum_{k=0}^{\infty}(\sinh(2k+1)\pi(1-x)+\sinh(2k+1)\pi x)\frac{\sin(2k+1)\pi y}{(2k+1)^3\sinh(2k+1)\pi}$$
$$-\frac{1}{2}x(1-x)-\frac{1}{2}y(1-y)\ .$$

Finite-Elemente – Näherungslösung für $n = 49$:

Im folgenden ist die Lösung in den inneren Gitterpunkten angegeben, und zwar in der Reihenfolge ihrer Lage (von oben nach unten, wie eingezeichnet).



Die Lösung mit Finiten-Elementen (*quadratische Grundfunktionen*) lautet

```
-0.05351593  -0.07925408  -0.08673271  -0.07925408  -0.05351593
-0.07925408  -0.12121212  -0.13286713  -0.12121212  -0.07925408
-0.08673271  -0.13286713  -0.14675602  -0.13286713  -0.08673271
-0.07925408  -0.12121212  -0.13286713  -0.12121212  -0.07925408
-0.05351593  -0.07925408  -0.08673271  -0.07925408  -0.05351593
```

Zum Vergleich die Werte der exakten Lösung (auch näherungsweise, da unendliche Reihe)

```
-0.05455991  -0.07982901  -0.08730273  -0.07982901  -0.05455991
-0.07982901  -0.12069185  -0.13317246  -0.12069185  -0.07982901
-0.08730273  -0.13317246  -0.14734271  -0.13317246  -0.08730273
-0.07982901  -0.12069185  -0.13317246  -0.12069185  -0.07982901
-0.05455991  -0.07982901  -0.08730273  -0.07982901  -0.05455991
```

Der maximale Fehler zwischen Finite-Elemente–Lösung und exakter Lösung beträgt: $1.0 \cdot 10^{-3}$.

Zum Vergleich geben wir auch die Ergebnisse an, die man mit *linearen Grundfunktionen* (vgl. Beispiel S.115/116) erhält , und die Ergebnisse, die man mit *Differenzenverfahren* (vgl. S.98-101) erhält:

Die Lösung mit Finiten-Elementen (*lineare Grundfunktionen*) lautet

```
-0.05671548 -0.08175704 -0.08918233 -0.08175704 -0.05671548
-0.08175704 -0.12354312 -0.13619168 -0.12354312 -0.08175704
-0.08918233 -0.13619168 -0.15070073 -0.13619168 -0.08918233
-0.08175704 -0.12354312 -0.13619168 -0.12354312 -0.08175704
-0.05671548 -0.08175704 -0.08918233 -0.08175704 -0.05671548
```

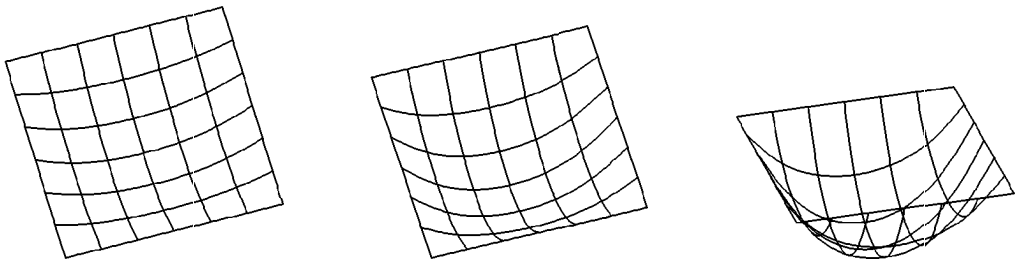
maximaler Fehler (zur exakten Lösung): $3.4 \cdot 10^{-3}$.

Die Lösung mit *Differenzenverfahren* lautet

```
-0.05288462 -0.07799145 -0.08547009 -0.07799145 -0.05288462
-0.07799145 -0.11805556 -0.13034188 -0.11805556 -0.07799145
-0.08547009 -0.13034188 -0.14423077 -0.13034188 -0.08547009
-0.07799145 -0.11805556 -0.13034188 -0.11805556 -0.07799145
-0.05288462 -0.07799145 -0.08547009 -0.07799145 -0.05288462
```

maximaler Fehler (zur exakten Lösung): $3.1 \cdot 10^{-3}$.

Verbindet man die Werte c_i mit Hilfe *kubischer Splines* (jeweils in $x-$ und $y-$Richtung), so erhält man die ungefähre Gestalt der Lösungsfläche (i.f. aus verschiedenen Aufsichten):



Beispiel 2

$$u_{xx} + u_{yy} = 2 \quad \text{in} \quad G = \left\{ \begin{pmatrix} x \\ y \end{pmatrix} \; : \; 0 < x, y < 1 \right\}$$

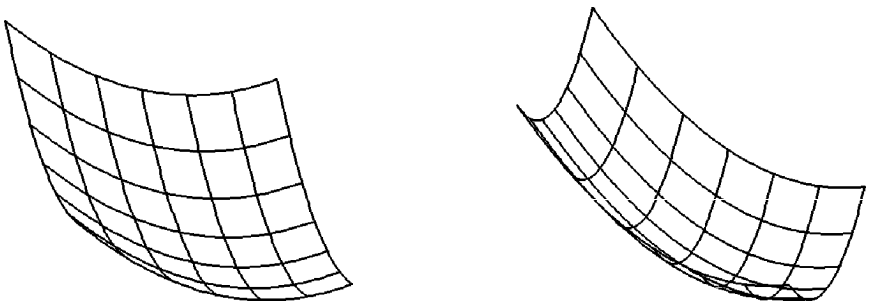
$$u_{|\partial G} = \frac{1}{2}(x^2 + y^2).$$

Exakte Lösung: $u(x, y) = \frac{1}{2}(x^2 + y^2)$.

Die Lösung mit Finiten-Elementen (*quadratische Grundfunktionen*) lautet

```
0.36111111   0.40277778   0.47222222   0.56944444   0.69444444
0.23611111   0.27777778   0.34722222   0.44444444   0.56944444
0.13888889   0.18055556   0.25000000   0.34722222   0.47222222
0.06944444   0.11111111   0.18055556   0.27777778   0.40277778
0.02777778   0.06944444   0.13888889   0.23611111   0.36111111
```

maximaler Fehler (zur exakten Lösung): $1.8 \cdot 10^{-12}$.



Inhaltsverzeichnis

Literaturverzeichnis

Schwarz, H.R.: Numerische Mathematik

Werner, J.: Numerische Mathematik

Werner, H.: Praktische Mathematik I

Werner, H./ Schaback, R.: Praktische Mathematik II

Törnig, W./ Spellucci, P.: Numerische Mathematik für Ingenieure und Physiker

Maess, G.: Vorlesungen über Numerische Mathematik

Engeln-Müllges, G./ Reutter, F.: Formelsammlung zur Numerischen Mathematik mit Pascal-Programmen

Jordan-Engeln, G./ Reutter, F.: Numerische Mathematik für Ingenieure

Jordan-Engeln, G./ Reutter, F.: Formelsammlung zur Numerischen Mathematik mit Fortran Programmen

Spellucci, P./ Törnig, W.: Eigenwertberechnung in den Ingenieurwissenschaften

Törnig, W./ Gipser, M./ Kaspar, B.: Numerische Lösung von partiellen DGL der Technik (mit: große schwach besetzte GLS)

Schwarz, H.R.: Methode der finiten Elemente

Schwarz, H.R.: Fortran Programme zur Methode der finiten Elemente

Braess, D.: Finite Elemente

Cuvelier,C./ Segal, A.: Finite Elements Methods and Navier-Stokes Equations

Berg, L.: Lineare Gleichungssysteme mit Bandstruktur

Hackbusch, W.: Theorie und Numerik elliptischer DGL