HSB
Hochschule Bremen
City University of Applied Sciences

Höhere Mathematik 4

Kapitel 16

Numerische Mathematik

Prof. Dr.-Ing. Dieter Kraus

Höhere Mathematik 4

Kapitel 16

Inhaltsverzeichnis

16 Numerische Mathematik

16.1 Lineare Gleichungssysteme

16.1.1 Gauß Algorithmus

Gegeben sei ein lineares Gleichungssystem $\mathbf{Ax} = \mathbf{b}$ mit einer regulären $n \times n$ Koeffizientenmatrix $\mathbf{A}$ und der rechten Seite $\mathbf{b}$. Da $\mathbf{A}$ regulär ist, ist das Gleichungssystem eindeutig lösbar. Die erweiterte Matrix

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

lässt sich mit Hilfe des Gauß Algorithmus auf folgende Form bringen:

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

Somit erhalten wir das lineare Gleichungssystem $\mathbf{Rx} = \mathbf{c}$ mit einer oberen Dreiecksmatrix $\mathbf{R}$ und der neuen rechten Seite $\mathbf{c}$.

Bei diesem Gleichungssystem $\mathbf{Rx} = \mathbf{c}$ lassen sich durch Rückwärtseinsetzen, d.h. beginnend mit der letzten Zeile, die Unbekannten $x_n, x_{n-1}, \ldots, x_1$ berechnen.

Für die Umwandlung des Gleichungssystems sind die folgenden Gauß-Schritte notwendig:

a) In der ersten Spalte Nullen erzeugen

$$\left(\begin{array}{cccc|c} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ a_{21} & a_{22} & \cdots & a_{2n} & b_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} & b_n \end{array}\right)$$

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

$$\tilde{a}_{ik} = a_{ik} - l_{i1} \cdot a_{1k}, \quad \tilde{a}_{i1} = 0$$

$$\tilde{b}_i = b_i - l_{i1} \cdot b_1, \quad l_{i1} = \frac{a_{i1}}{a_{11}}, \quad (i = 2,\ldots,n;\, k = 2,\ldots,n).$$

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

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

b) In der zweiten Spalte Nullen erzeugen

Falls $a_{22} \neq 0$ (sonst vorher Zeilentausch), berechne:
(i-te Zeile) $- a_{i2}/a_{22} \cdot$ (2te Zeile), also mit $l_{i2} = a_{i2}/a_{22}$

$$\tilde{a}_{ik} = a_{ik} - l_{i2} \cdot a_{2k}, \quad \tilde{a}_{i2} = 0$$

$$\tilde{b}_i = b_i - l_{i2} \cdot b_2, \quad l_{i2} = \frac{a_{i2}}{a_{22}}, \quad (i = 3,\ldots,n;\, k = 3,\ldots,n).$$

Wir speichern wieder die Vorfaktoren $l_{i2} = a_{i2}/a_{22}$ auf den Speicherplätzen von a_{i2}. Fahren wir so fort bis zur vorletzten Spalte, so erhalten wir nach dem letzten Schritt das folgende Schema:

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

Mit der Linksdreiecksmatrix $\mathbf{L}$ und der Rechtsdreiecksmatrix $\mathbf{R}$

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

gilt

$$\mathbf{A} = \mathbf{LR}$$

denn z.B. liefert die 2te Zeile mal der k-ten Spalte

$$l_{21}\, r_{1k} + r_{2k} = l_{21}\, a_{1k} + \tilde{a}_{2k} = a_{2k} \quad (k = 1,\ldots,n),$$

da $r_{1k} = a_{1k}$ (1te Zeile wird nicht verändert) und

$$r_{2k} = \tilde{a}_{2k} = a_{2k} - l_{21}\, a_{1k} \quad (k = 1,\ldots,n),$$

vgl. Schritt a). Also gilt

$$\mathbf{Ax} = \mathbf{b} \;\Leftrightarrow\; \mathbf{LRx} = \mathbf{b} \;\Leftrightarrow\; \{\mathbf{Lc} = \mathbf{b},\ \mathbf{Rx} = \mathbf{c}\}$$

Zur Lösung des linearen Gleichungssystems $\mathbf{Ax} = \mathbf{b}$ sind also 3 Schritte durchzuführen:

1) $\mathbf{A} = \mathbf{LR}$ (**LR**-Zerlegung)
2) $\mathbf{Lc} = \mathbf{b}$ (Vorwärtseinsetzen)
3) $\mathbf{Rx} = \mathbf{c}$ (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 verwendet werden. Diese Pivotstrategie nennt man Diagonalstrategie.

1) Diagonalstrategie

Ist das jeweilige Diagonalelement ≠ 0, so benutze man dieses Diagonalelement als Pivotelement. Die Diagonalstrategie ist nur sinnvoll bei diagonaldominanten Matrizen. Dabei heißt eine $n \times n$ 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 $x_1 = 1.111,\ x_2 = 1.222$. Eine Rechnung mit 5stelliger Genauigkeit liefert bei Diagonalstrategie

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

$$\tilde{b}_2 = b_2 - \frac{a_{21}}{a_{11}} b_1 = 2.333 - \frac{1.2224}{0.00035} = -3490.3$$

also

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

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

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

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

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

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 ab dem Diagonalelement betraglich größte Element in dieser Spalte, d.h.

$$\left|a_{kj}\right| = \max_{j \le i \le n} \left|a_{ij}\right| \quad \text{und} \quad k = \arg\max_{j \le i \le n} \left|a_{ij}\right|$$

Dann vertausche man die j-te Zeile mit der k-ten Zeile.

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

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

$$\tilde{a}_{ij} = \frac{a_{ij}}{\sum_{l=1}^{n} |a_{il}|}, \quad 1 \le i, j \le n$$

Diese Normierung wird nicht explizit durchgeführt, sondern bei der Pivotwahl berücksichtigt. Das Pivotelement der j-ten Spalte ermittelt man somit wie folgt

$$k = \arg\max_{j \le i \le n} \frac{|a_{ij}|}{\sum_{l=1}^{n} |a_{il}|} \quad \Rightarrow \quad a_{kj} = \text{Pivotelement}$$

Berechnung der Determinante

Beachtet man die Zeilenvertauschungen, so lässt sich aus der Matrix $\mathbf{R}$ einfach die Determinante von $\mathbf{A}$ berechnen. Es gilt

$$\det \mathbf{A} = (-1)^s \prod_{i=1}^{n} r_{ii}, \quad \text{da} \quad \pm \det \mathbf{A} = \det(\mathbf{LR}) = \det \mathbf{L} \det \mathbf{R} = \prod_{i=1}^{n} r_{ii},$$

wobei s die Anzahl der Zeilenvertauschungen angibt.

Die **LR**-Zerlegung und die Veränderung der rechten Seite (Vorwärtseinsetzen) werden in zwei getrennten Schritten durchgeführt. Dies hat den Vorteil, dass die **LR**-Zerlegung nur einmal durchgeführt werden muss, wenn mehrere lineare Gleichungssysteme mit der gleichen Koeffizientenmatrix $\mathbf{A}$ aber verschiedenen rechten Seiten $\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_m$ gelöst werden sollen. Also nur die Schritte Vorwärtseinsetzen und Rückwärtseinsetzen sind mehrmals auszuführen.

Berechnung der inversen Matrix $\mathbf{A}^{-1}$

Es gilt

$$\mathbf{AX} = \mathbf{E} \quad \Leftrightarrow \quad \mathbf{As}_i = \mathbf{e}_i, \quad (i = 1,\ldots,n).$$

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

Im folgenden Algorithmus werden die Zeilenvertauschungen auf einem Vektor $\mathbf{p}$ 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 $\mathbf{c}$ und der Lösungsvektor $\mathbf{x}$ werden auf $\mathbf{b}$ gespeichert. Die Werte a_{ij} der Matrix $\mathbf{A}$ und die Werte b_i der rechten Seite $\mathbf{b}$ 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 andere Zwecke, so müssen diese Werte vorher gesondert gespeichert werden.

Matlab-Programm (Gauß-Algorithmus)

```
% LR-Zerlegung   (A = LR)
    det = 1;
    for k = 1:n–1              %Pivotsuche
        max = 0;
        p(k) = 0;
        for i = k:n
            s = 0;
            for j = k:n
                s = s + abs(a(i,j));
            end
            q = abs(a(i,k))/s;
            if   q > max
```

```
            max = q;
            p(k) = i;
        end
    end
    if  max == 0
        det = 0;          %Matrix singulär
        break;
    end
    if  p(k) ~= k         %Vorzeichenwechsel, Zeilentausch
        det = –det
        for j = 1:n
            h = a(k,j);
            a(k,j) = a(p(k),j);
            a(p(k),j) = h;
        end
    end
    det = det * a(k,k);
```



```
    for i = k+1:n         %Nullen in der k-ten Spalte erzeugen
        a(i,k) = a(i,k)/a(k,k);
        for j = k+1:n
            a(i,j) = a(i,j) – a(i,k) * a(k,j);
        end
    end
end
det = det * a(n,n);
if  det == 0
    disp('Matrix singulär')
else
%   Vorwärtseinsetzen  Lc = b
    for k = 1:n–1
        if  p(k) ~= k
            h = b(k);
            b(k) = b(p(k));
            b(p(k)) = h;
```

```
            end
        end
        for i = 2:n
            for j = 1:i−1
                b(i) = b(i) − a(i,j) ∗ b(j);
            end
        end
% Rückwärtseinsetzen Rx = c
        for i = n:−1:1
            s = b(i);
            for k = i+1:n
                s = s − a(i,k) ∗ b(k);
            end
            b(i) = s/a(i,i);
        end
    end
```


Rechenaufwand

Bei der **LR**−Zerlegung (3 ineinander geschachtelte Schleifen) beträgt die Anzahl wesentlicher Rechenoperationen für ein $n \times n$ Gleichungssystem

$$1/3\ (n^3 - n) = O(n^3).$$

Beim Vorwärts- und Rückwärtseinsetzen (je 2 ineinander geschachtelte Schleifen) benötigt man noch

$$1/2\ (n^2 - n) = O(n^2) \quad \text{bzw.} \quad 1/2\ (n^2 + n) = O(n^2)$$

Rechenoperationen. Der vollständige Gauß-Algorithmus erfordert somit

$$1/3\ n^3 + n^2 - 1/3\ n$$

wesentliche Operationen.

Der Aufwand in der Zerlegungsphase bedingt einen hohen Rechenaufwand bei großen Gleichungssystemen. So bedeutet z.B. eine Verdoppelung der Zahl der Unbekannten eine Verachtfachung des Rechenaufwands. Wir werden später iterative Verfahren behandeln, die zur Bewältigung sehr großer Gleichungssysteme besser geeignet sind als der Gauß-Algorithmus.

16.1.2 Genauigkeitsfragen, Fehlerabschätzung

Trotz guter Pivotstrategie treten bei der Berechnung der Lösung eines linearen Gleichungssystems Rundungsfehler auf. Wir wollen versuchen, die auftretenden Rundungsfehler abzuschätzen. Um Aussagen über die Fehler machen zu können, benötigen wir neben den schon bekannten Maßzahlen für die Größe eines Vektors (Vektornormen, $\|\mathbf{x}\|$) auch Maßzahlen für die Größe einer Matrix (Matrixnormen, $\|\mathbf{A}\|$).

Matrixnormen

Ist $\mathbf{Ax} = \mathbf{b}$, so gilt $\|\mathbf{Ax}\| = \|\mathbf{b}\|$ mit irgendeiner Vektornorm $\|\cdot\|$. Für die Matrixnorm $\|\mathbf{A}\|$ soll gelten

$$\|\mathbf{Ax}\| \le \|\mathbf{A}\|\,\|\mathbf{x}\|, \quad \text{also } \|\mathbf{A}\| \ge \frac{\|\mathbf{Ax}\|}{\|\mathbf{x}\|}.$$

Dies motiviert die folgende Definition.

Definition 16-1: (*Matrixnorm*)
Der zu einer gegebenen Vektornorm definierte Zahlenwert

$$\|\mathbf{A}\| := \max_{\mathbf{x}\neq\mathbf{0}} \frac{\|\mathbf{Ax}\|}{\|\mathbf{x}\|} = \max_{\|\mathbf{x}\|=1} \|\mathbf{Ax}\|$$

heißt zugeordnete oder natürliche Matrixnorm.

Für die Matrixnorm gelten die folgenden Eigenschaften.

Satz 16-1: (*Eigenschaften der Matrixnorm*)

$\|\mathbf{A}\| \ge 0$ für alle $\mathbf{A}$, $\|\mathbf{A}\| = 0 \Leftrightarrow \mathbf{A} = \mathbf{0}$

$\|c\mathbf{A}\| = |c|\,\|\mathbf{A}\|, \quad c \in \mathbb{R}$

$\|\mathbf{A} + \mathbf{B}\| \le \|\mathbf{A}\| + \|\mathbf{B}\|$

$\|\mathbf{AB}\| \le \|\mathbf{A}\|\,\|\mathbf{B}\|$

Da in der Anwendung unterschiedliche Vektornormen benutzt werden, erhalten wir auch entsprechend unterschiedliche Matrixnormen. Im folgenden werden zwei Beispiele behandelt.

1) ∞-Norm

$$\|\mathbf{x}\|_\infty = \max_{1\le i\le n} |x_i|, \quad \text{(Maximumnorm)}$$

$$\|\mathbf{A}\|_\infty = \max_{\|\mathbf{x}\|_\infty=1} \|\mathbf{A}\mathbf{x}\|_\infty = \max_{\|\mathbf{x}\|_\infty=1} \max_{1\le i\le n} \left|\sum_{k=1}^{n} a_{ik}x_k\right| = \max_{1\le i\le n} \max_{\|\mathbf{x}\|_\infty=1} \left|\sum_{k=1}^{n} a_{ik}x_k\right| \le \max_{1\le i\le n} \sum_{k=1}^{n} |a_{ik}|$$

da $|x_k| \le \|\mathbf{x}\|_\infty$ für alle $k = 1,\ldots,n$. Das Gleichheitszeichen wird angenommen für den Vektor $\mathbf{x}$ mit $x_k = \operatorname{sgn}(a_{ik})$. Also gilt für die ∞-Norm

$$\|\mathbf{A}\|_\infty = \max_{1\le i\le n} \sum_{k=1}^{n} |a_{ik}| \quad \text{(Zeilensummennorm)}$$

2) 2−Norm

$$\|\mathbf{x}\|_2 = \sqrt{\mathbf{x}^T\mathbf{x}} = \sqrt{\sum_{i=1}^{n} x_i^2} \quad \text{(euklidsche Norm)}$$

$$\|\mathbf{A}\|_2 = \max_{\|\mathbf{x}\|_2=1} \|\mathbf{A}\mathbf{x}\|_2 = \max_{\|\mathbf{x}\|_2=1} \sqrt{(\mathbf{A}\mathbf{x})^T\mathbf{A}\mathbf{x}} = \max_{\|\mathbf{x}\|_2=1} \sqrt{\mathbf{x}^T\mathbf{A}^T\mathbf{A}\mathbf{x}} \quad \text{(spektral Norm)}$$

Die Matrix $\mathbf{A}^T\mathbf{A}$ ist symmetrisch und positiv semidefinit, denn

$(\mathbf{A}^T\mathbf{A})^T = \mathbf{A}^T(\mathbf{A}^T)^T = \mathbf{A}^T\mathbf{A}$ und

$$\|\mathbf{A}\mathbf{x}\|_2^2 = \mathbf{x}^T(\mathbf{A}^T\mathbf{A})\mathbf{x} = (\mathbf{A}\mathbf{x})^T(\mathbf{A}\mathbf{x}) = \mathbf{y}^T\mathbf{y} = \|\mathbf{y}\|_2^2 \ge 0.$$

Damit ist also $q(\mathbf{x}) = \mathbf{x}^T(\mathbf{A}^T\mathbf{A})\,\mathbf{x}$ eine positiv semidefinite quadratische Form, die wir im folgenden genauer untersuchen wollen.

<u>Anmerkung:</u>
Ist $\mathbf{A}$ regulär, so ist die Matrix $\mathbf{A}^T\mathbf{A}$ sogar positiv definit, denn

$$\|\mathbf{A}\mathbf{x}\|_2 = 0 \quad \Leftrightarrow \quad \mathbf{A}\mathbf{x} = 0 \quad \Leftrightarrow \quad \mathbf{x} = \mathbf{0} \quad \text{(da } \mathbf{A} \text{ regulär)}$$

Da $(\mathbf{A}^T\mathbf{A})$ symmetrisch und positiv semidefinit ist, sind alle Eigenwerte von $(\mathbf{A}^T\mathbf{A})$ reell und ≥ 0. Die zu Eigenwerten zugehörigen Eigenvektoren stehen senkrecht aufeinander.

Es seien $\mu_1 \geq \mu_2 \ldots \geq \mu_n \geq 0$ die Eigenwerte von $\mathbf{A}^T\mathbf{A}$ und $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$ die zugehörigen Eigenvektoren mit

$$\mathbf{x}_i^T \mathbf{x}_j = \begin{cases} 1 & \text{für } i = j \\ 0 & \text{für } i \neq j \end{cases}.$$

Die Eigenvektoren $\mathbf{x}_i$ bilden eine Basis des $\mathbb{R}^n$. Somit lässt sich jeder Vektor $\mathbf{x} \in \mathbb{R}^n$ mit $\|\mathbf{x}\|_2 = 1$ als Linearkombination der Eigenvektoren $\mathbf{x}_i$ schreiben, d.h.

$$\mathbf{x} = \sum_{i=1}^{n} c_i \mathbf{x}_i$$

$$\text{mit} \quad \|\mathbf{x}\|_2^2 = \mathbf{x}^T\mathbf{x} = \left(\sum_{i=1}^{n} c_i \mathbf{x}_i^T \right) \left(\sum_{j=1}^{n} c_j \mathbf{x}_j \right) = \sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j \mathbf{x}_i^T \mathbf{x}_j = \sum_{i=1}^{n} c_i^2 = 1$$

Also gilt

$$q(\mathbf{x}) = \mathbf{x}^T \mathbf{A}^T \mathbf{A}\mathbf{x} = \left(\sum_{i=1}^{n} c_i \mathbf{x}_i^T \mathbf{A}^T \right) \left(\sum_{j=1}^{n} c_j \mathbf{A}\mathbf{x}_j \right) = \sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j \mathbf{x}_i^T \mathbf{A}^T \mathbf{A}\mathbf{x}_j$$

$$= \sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j \mu_j \mathbf{x}_i^T \mathbf{x}_j = \sum_{i=1}^{n} c_i^2 \mu_i \leq \mu_1 \sum_{i=1}^{n} c_i^2 = \mu_1, \quad \text{da } \mathbf{A}^T\mathbf{A}\mathbf{x}_j = \mu_j \mathbf{x}_j$$

Das Gleichheitszeichen wird für $\mathbf{x} = \mathbf{x}_1$ (Eigenvektor zu μ_1) angenommen und es gilt

$$\|\mathbf{A}\|_2 = \max_{\|\mathbf{x}\|_2 = 1} \sqrt{q(\mathbf{x})} = \max_{\|\mathbf{x}\|_2 = 1} \sqrt{\mathbf{x}^T \mathbf{A}^T \mathbf{A}\mathbf{x}} = \sqrt{\mu_1},$$

wobei μ_1 den größten Eigenwert von $\mathbf{A}^T\mathbf{A}$ angibt. Für $\mathbf{A}$ regulär soll nun $\|\mathbf{A}^{-1}\|_2$ untersucht werden. Wir können schreiben

$$\|\mathbf{A}^{-1}\|_2 = \sqrt{\eta_1},$$

wobei η_1 den größten Eigenwert von

$$(\mathbf{A}^{-1})^T \mathbf{A}^{-1} = (\mathbf{A}^T)^{-1} \mathbf{A}^{-1} = (\mathbf{A}\mathbf{A}^T)^{-1}$$

bezeichnet. Da $\mathbf{A}\mathbf{A}^T$ wegen

$$\mathbf{A}^{-1}(\mathbf{A}\mathbf{A}^T)\mathbf{A} = (\mathbf{A}^{-1}\mathbf{A})\mathbf{A}^T\mathbf{A} = \mathbf{A}^T\mathbf{A}$$

zu $\mathbf{A}^T\mathbf{A}$ ähnlich ist, besitzen $\mathbf{A}\mathbf{A}^T$ und $\mathbf{A}^T\mathbf{A}$ die gleichen Eigenwerte $\mu_1 \geq \mu_2 \ldots \geq \mu_n > 0$. Die Matrix $(\mathbf{A}\mathbf{A}^T)^{-1}$ hat dann die reziproken Eigenwerte $1/\mu_n \geq 1/\mu_{n-1} \ldots \geq 1/\mu_1 > 0$. Der größte dieser Eigenwerte ist $1/\mu_n$, wobei μ_n der kleinste Eigenwert von $\mathbf{A}^T\mathbf{A}$ ist. Folglich gilt

$$\left\|\mathbf{A}^{-1}\right\|_2 = \sqrt{\eta_1} = \frac{1}{\sqrt{\mu_n}}$$

mit μ_n dem kleinsten Eigenwert von $\mathbf{A}^T\mathbf{A}$.

Spezialfall:
Für $\mathbf{A}$ symmetrisch gilt: $\mathbf{A}^T\mathbf{A} = \mathbf{A}^2$. Ist λ Eigenwert von $\mathbf{A}$, so ist λ^2 Eigenwert von $\mathbf{A}^T\mathbf{A} = \mathbf{A}^2$. Also gilt für symmetrische Matrizen

$$\left\|\mathbf{A}\right\|_2 = \left|\lambda_1\right| \quad \text{und} \quad \left\|\mathbf{A}^{-1}\right\|_2 = \frac{1}{\left|\lambda_n\right|}$$

wobei λ_1 bzw. λ_n den betraglich größten bzw. kleinsten Eigenwert von $\mathbf{A}$ bezeichnen.

Anmerkung:
Um $\|\mathbf{A}^{-1}\|_\infty$, d.h. die Zeilensummennorm, berechnen zu können, benötigt man die inverse Matrix $\mathbf{A}^{-1}$. Bei der Berechnung von $\|\mathbf{A}^{-1}\|_2$ benötigt man nur den kleinsten Eigenwert von $\mathbf{A}^T\mathbf{A}$. Ein Verfahren zur Berechnung des kleinsten Eigenwerts wird später behandelt (vgl. Wielandt-Verfahren).

Fehlerrechnung

Gegeben sei das lineare Gleichungssystem $\mathbf{A}\mathbf{x} = \mathbf{b}$ mit der regulären Koeffizientenmatrix $\mathbf{A}$. Ferner sei $\mathbf{x}$ die exakte, $\mathbf{x}_*$ die berechnete Lösung des Gleichungssystems, $\Delta\mathbf{x} = \mathbf{x}_* - \mathbf{x}$ der Fehler und $\mathbf{r} = \mathbf{A}\mathbf{x}_* - \mathbf{b}$. Dann gilt

$$\mathbf{A}\Delta\mathbf{x} = \mathbf{A}(\mathbf{x}_* - \mathbf{x}) = \mathbf{A}\mathbf{x}_* - \mathbf{A}\mathbf{x} = \mathbf{A}\mathbf{x}_* - \mathbf{b} = \mathbf{r} \quad \Rightarrow \quad \Delta\mathbf{x} = \mathbf{A}^{-1}\mathbf{r}.$$

Also gilt bezüglich einer beliebigen Norm

$$\frac{\|\Delta\mathbf{x}\|}{\|\mathbf{x}\|} = \frac{\|\mathbf{A}^{-1}\mathbf{r}\|}{\|\mathbf{x}\|} \le \frac{\|\mathbf{A}^{-1}\|\,\|\mathbf{r}\|}{\|\mathbf{b}\|/\|\mathbf{A}\|} = \|\mathbf{A}\|\,\|\mathbf{A}^{-1}\|\frac{\|\mathbf{r}\|}{\|\mathbf{b}\|}, \text{ da } \|\mathbf{b}\| = \|\mathbf{A}\mathbf{x}\| \le \|\mathbf{A}\|\,\|\mathbf{x}\| \Rightarrow \|\mathbf{x}\| \ge \frac{\|\mathbf{b}\|}{\|\mathbf{A}\|}$$

Definition 16-2: (*Konditionszahl einer Matrix*)
Der Zahlenwert

$$\text{cond}(\mathbf{A}) = \|\mathbf{A}\|\,\|\mathbf{A}^{-1}\|$$

heißt die Konditionszahl der Matrix $\mathbf{A}$ bezüglich der verwendeten Norm.

Mit dieser Konditionszahl lautet die obige Fehlerabschätzung für den relativen Fehler

$$\frac{\|\Delta\mathbf{x}\|}{\|\mathbf{x}\|} \le \text{cond}(\mathbf{A})\frac{\|\mathbf{r}\|}{\|\mathbf{b}\|}.$$

Der relative Fehler der Lösung eines linearen Gleichungssystems hängt also wesentlich von der Konditionszahl der Koeffizientenmatrix $\mathbf{A}$ ab.

Die Konditionszahl cond($\mathbf{A}$) ist immer ≥ 1, denn es gilt stets

$$1 = \|\mathbf{E}\| = \|\mathbf{A}\mathbf{A}^{-1}\| \le \|\mathbf{A}\|\,\|\mathbf{A}^{-1}\| = \text{cond}(\mathbf{A}).$$

Die Konditionszahl lässt sich für die ∞-Norm nur sehr aufwendig berechnen, da man für $\|\mathbf{A}^{-1}\|_\infty$ die komplette inverse Matrix $\mathbf{A}^{-1}$ benötigt.

Einfacher ist die Berechnung der Konditionszahl für die 2-Norm, denn es gilt

$$\text{cond}(\mathbf{A}) = \sqrt{\frac{\mu_1}{\mu_n}} \quad \text{mit} \quad \begin{matrix}\mu_1 \text{ größter Eigenwert} \\ \mu_n \text{ kleinster Eigenwert}\end{matrix} \quad \text{von } \mathbf{A}^T\mathbf{A}$$

sowie für symmetrisches $\mathbf{A}$

$$\text{cond}(\mathbf{A}) = \frac{|\lambda_1|}{|\lambda_n|} \quad \text{mit} \quad \begin{matrix}\lambda_1 \text{ betraglich größter Eigenwert} \\ \lambda_n \text{ betraglich kleinster Eigenwert}\end{matrix} \quad \text{von } \mathbf{A} = \mathbf{A}^T$$

Störung in den Koeffizienten

Wir führen nun eine genauere Fehleranalyse für den Fall durch, dass die Matrix $\mathbf{A}$ um $\Delta\mathbf{A}$ und die rechte Seite $\mathbf{b}$ um $\Delta\mathbf{b}$, z.B. aufgrund der endlichen Darstellungsgenauigkeit im Computer, verändert werden.

$$(\mathbf{A}+\Delta\mathbf{A})(\mathbf{x}+\Delta\mathbf{x}) = (\mathbf{b}+\Delta\mathbf{b}) \quad (\Delta\mathbf{x} \text{ Fehler der Lösung})$$

$$\Rightarrow (\mathbf{A}+\Delta\mathbf{A})\Delta\mathbf{x} = \mathbf{b}+\Delta\mathbf{b}-\mathbf{A}\mathbf{x}-\Delta\mathbf{A}\mathbf{x} = \Delta\mathbf{b}-\Delta\mathbf{A}\mathbf{x} \quad (\text{da } \mathbf{A}\mathbf{x}=\mathbf{b})$$

$$\Rightarrow \Delta\mathbf{x} = (\mathbf{A}+\Delta\mathbf{A})^{-1}(\Delta\mathbf{b}-\Delta\mathbf{A}\mathbf{x}) = \left(\mathbf{A}(\mathbf{E}+\mathbf{A}^{-1}\Delta\mathbf{A})\right)^{-1}(\Delta\mathbf{b}-\Delta\mathbf{A}\mathbf{x})$$

$$\Rightarrow \|\Delta\mathbf{x}\| \le \|\mathbf{A}^{-1}\| \, \|(\mathbf{E}+\mathbf{A}^{-1}\Delta\mathbf{A})^{-1}\| \left(\|\Delta\mathbf{b}\| + \|\Delta\mathbf{A}\| \, \|\mathbf{x}\| \right)$$

$$\Rightarrow \frac{\|\Delta\mathbf{x}\|}{\|\mathbf{x}\|} \le \|\mathbf{A}^{-1}\| \, \|(\mathbf{E}+\mathbf{A}^{-1}\Delta\mathbf{A})^{-1}\| \left(\frac{\|\Delta\mathbf{b}\|}{\|\mathbf{x}\|} + \|\Delta\mathbf{A}\| \right)$$

Um den mittleren Ausdruck auf der rechten Seite weiter abschätzen zu können benötigen wir die Ungleichung

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

die hier nicht bewiesen werden soll. Also gilt unter der Voraussetzung, dass $\|\mathbf{A}^{-1}\| \, \|\Delta\mathbf{A}\| < 1$ ist die Abschätzung

$$\frac{\|\Delta\mathbf{x}\|}{\|\mathbf{x}\|} \le \frac{\|\mathbf{A}^{-1}\|}{1-\|\mathbf{A}^{-1}\| \, \|\Delta\mathbf{A}\|} \left(\frac{\|\Delta\mathbf{b}\|}{\|\mathbf{x}\|} + \|\Delta\mathbf{A}\| \right) = \frac{\|\mathbf{A}^{-1}\| \, \|\mathbf{A}\|}{1-\|\mathbf{A}^{-1}\| \, \|\Delta\mathbf{A}\|} \left(\frac{\|\Delta\mathbf{b}\|}{\|\mathbf{A}\| \, \|\mathbf{x}\|} + \frac{\|\Delta\mathbf{A}\|}{\|\mathbf{A}\|} \right)$$

$$\le \frac{\text{cond}(\mathbf{A})}{1-\text{cond}(\mathbf{A}) \, \|\Delta\mathbf{A}\|/\|\mathbf{A}\|} \left(\frac{\|\Delta\mathbf{b}\|}{\|\mathbf{b}\|} + \frac{\|\Delta\mathbf{A}\|}{\|\mathbf{A}\|} \right), \quad \text{da } \|\mathbf{A}\| \, \|\mathbf{x}\| \ge \|\mathbf{b}\|$$

Bei einer *d*-stelligen Gleitpunktrechnung sind die relativen Fehler für beliebige kompatible Normen von der Größenordnung

$$\frac{\|\Delta\mathbf{b}\|}{\|\mathbf{b}\|} \approx 5 \cdot 10^{-d}, \quad \frac{\|\Delta\mathbf{A}\|}{\|\mathbf{A}\|} \approx 5 \cdot 10^{-d}.$$

Ist die Konditionszahl

$$\text{cond}(\mathbf{A}) \approx 10^{\alpha} \quad \text{mit} \quad 5 \cdot 10^{\alpha - d} \ll 1,$$

so kann man den relativen Fehler qualitativ durch

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

abschätzen.

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. Es sollte immer $d \gg \alpha$ gelten.

Beispiel:

$$\mathbf{A} = \begin{pmatrix} 0.990005 & 0.979996 \\ 0.979996 & 0.970004 \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} 1.95840828 \\ 1.93859352 \end{pmatrix} \quad \Rightarrow \quad \mathbf{x} = \begin{pmatrix} 1.8 \\ 0.18 \end{pmatrix}$$

Rechnung mit 5-stelliger Genauigkeit, d.h. $d = 5$, liefert

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

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

Da $\mathbf{A}$ symmetrisch ist, ergibt sich die Konditionszahl bei Verwendung der 2-Norm zu $\text{cond}(\mathbf{A}) = |\lambda_1|/|\lambda_2|$. Die Eigenwerte einer 2×2 Matrix lassen sich wie folgt berechnen.

$$\mathbf{A} = \begin{pmatrix} a & c \\ c & b \end{pmatrix}, \quad \det(\mathbf{A} - \lambda\mathbf{E}) = \det\begin{pmatrix} a - \lambda & c \\ c & b - \lambda \end{pmatrix} = \lambda^2 - (a+b)\lambda + ab - c^2$$

$$\Rightarrow \quad \lambda_{1,2} = \frac{a+b}{2} \pm \sqrt{\left(\frac{a+b}{2}\right)^2 - ab + c^2}$$

Für unser Zahlenbeispiel erhalten wir

$$\lambda_1 = 1.9600515, \quad \lambda_2 = -0.0000425$$

und

$$\text{cond}(\mathbf{A}) = 46092 \approx 4 \cdot 10^4 \;\Rightarrow\; \alpha \approx 4.$$

Mit diesen Werten ergibt sich die Abschätzung des relativen Fehlers zu

$$\frac{\|\Delta \mathbf{x}\|_2}{\|\mathbf{x}\|_2} \le 10^{4-5+1} = 1.$$

Die Abschätzung zeigt, dass das Ergebnis nicht einmal auf eine Dezimalstelle nach dem Komma genau ist.

Definition 16-3: (*Hilbert Matrix*)
Die $n \times n$ Matrix

$$\mathbf{A} = \left(a_{ij}\right)_{i,j=1,\ldots,n} \quad \text{mit} \;\; a_{ij} = \frac{1}{i+j-1}$$

heißt Hilbert Matrix.

Die Inverse der Hilbert-Matrix ergibt sich zu

$$\mathbf{A}^{-1} = \mathbf{B} = \left(b_{ij}\right)_{i,j=1,\ldots,n} \quad \text{mit} \;\; b_{ij} = \frac{(-1)^{i+j}}{i+j-1} \cdot \frac{(n+i-1)!}{(i-1)!^2 (n-i)!} \cdot \frac{(n+j-1)!}{(j-1)!^2 (n-j)!}$$

Hilbert-Matrizen liefern ein Beispiel für schlecht konditionierte Matrizen.

Beispiel:

1) $n = 4$

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

Als betraglich größten bzw. kleinsten Eigenwert von $\mathbf{A}$ erhalten wir

$$\lambda_1 = 1.50021, \;\; \lambda_4 = 9.67 \cdot 10^{-5} \;\Rightarrow\; \text{cond}(\mathbf{A}) = \frac{|\lambda_1|}{|\lambda_4|} = 1.6 \cdot 10^4 \;\Rightarrow\; \alpha = 4$$

Für den relativen Fehler folgt hieraus die Fehlerabschätzung

$$\|\Delta\mathbf{x}\|/\|\mathbf{x}\| \le 10^{5-d}, \quad \text{d.h. } d \text{ sollte sehr viel größer als 5 sein}$$

2) $n = 8$

Für die Hilbert-Matrix der Größe $n = 8$ erhalten wir

$$\lambda_1 = 1.696,\ \lambda_8 = 1.11 \cdot 10^{-10} \ \Rightarrow\ \text{cond}(\mathbf{A}) = \frac{|\lambda_1|}{|\lambda_8|} = 1.5 \cdot 10^{10} \ \Rightarrow\ \alpha = 10$$

Für den relativen Fehler folgt hieraus die Fehlerabschätzung

$$\|\Delta\mathbf{x}\|/\|\mathbf{x}\| \le 10^{11-d}, \quad \text{d.h. } d \text{ sollte sehr viel größer als 11 sein}$$

Für die rechte Seite $\mathbf{b} = (1, 1, \ldots, 1)^T$ erhält man als Lösung des linearen Gleichungssystems $\mathbf{Ax} = \mathbf{b}$ für $d = 12$ folgendes Ergebnis

exakte Lsg.	–8	504	–7560	46200	–138600	216216	–168168	51480
berechn. Lsg.	–8.01	504.5	–7566.9	46236.8	–138698.2	216354.0	–168265.7	51507.4

16.1.3 Cholesky-Zerlegung

Ist $\mathbf{A}$ symmetrisch und positiv definit, dann existiert die Zerlegung

$$\mathbf{A} = \mathbf{L}\mathbf{L}^T$$

mit der unteren Dreiecksmatrix

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

Für $\mathbf{Ax} = \mathbf{b}$ gilt folglich $\mathbf{L}\mathbf{L}^T\mathbf{x} = \mathbf{b}$. Mit dem Hilfsvektor $\mathbf{c} = \mathbf{L}^T\mathbf{x}$ kann dann die Lösung von $\mathbf{Ax} = \mathbf{b}$ in den folgenden drei Schritten erfolgen.

1) $\mathbf{A} = \mathbf{L}\mathbf{L}^T$ ($\mathbf{L}\mathbf{L}^T$ – Zerlegung)
2) $\mathbf{Lc} = \mathbf{b}$ (Vorwärtseinsetzen)
3) $\mathbf{L}^T\mathbf{x} = \mathbf{c}$ (Rückwärtseinsetzen)

Dass die Zerlegung $\mathbf{A} = \mathbf{L}\mathbf{L}^T$ bei symmetrischen, positiv definiten Matrizen möglich ist, zeigt der folgende nach Cholesky benannte Algorithmus.

Cholesky-Zerlegung

Für $i = 1$ bis n

$$s = a_{ii} - \sum_{j=1}^{i-1} l_{ij}^2$$

Falls $s \le 0 \Rightarrow$ STOP, Matrix nicht positiv definit

$$l_{ii} = \sqrt{s}$$

Für $j = i+1$ bis n

$$l_{ji} = \frac{1}{l_{ii}} \left(a_{ji} - \sum_{k=1}^{i-1} l_{jk} l_{ik} \right)$$

Ende der j-Schleife

Ende der i-Schleife

denn

$$a_{ji} = l_{ji} l_{ii} + \sum_{k=1}^{i-1} l_{jk} l_{ik} = \sum_{k=1}^{i} l_{jk} l_{ik} = \sum_{k=1}^{i} (\mathbf{L})_{jk} \left(\mathbf{L}^T\right)_{ki} = \left(\mathbf{L}\mathbf{L}^T\right)_{ji}$$

Anmerkung:

Eine Zerlegung der Form $\mathbf{A} = \tilde{\mathbf{L}}\mathbf{D}\tilde{\mathbf{L}}^T$ ist ohne Berechnung der Wurzeln und damit auch für symmetrische nicht notwendigerweise positiv definite Matrizen möglich, wobei $\tilde{\mathbf{L}}$ eine untere Dreiecksmatrix mit Einsen auf der Hauptdiagonalen und $\mathbf{D}$ eine Diagonalmatrix bezeichnet.

Anmerkung:

Man kann mit Hilfe der Cholesky-Zerlegung feststellen ob eine symmetrische Matrix positiv definit ist.

Denn $\mathbf{A} = \mathbf{L}\mathbf{L}^T = \tilde{\mathbf{L}}\mathbf{D}\tilde{\mathbf{L}}^T$ ist für $d_{ii} > 0 \ \forall i = 1,\ldots,n$ positiv definit, da

$$\mathbf{x}^T\mathbf{A}\mathbf{x} = \mathbf{x}^T\mathbf{L}\mathbf{L}^T\mathbf{x} = \mathbf{x}^T\tilde{\mathbf{L}}\mathbf{D}\tilde{\mathbf{L}}^T\mathbf{x} = (\tilde{\mathbf{L}}^T\mathbf{x})^T\mathbf{D}(\tilde{\mathbf{L}}^T\mathbf{x}) = \mathbf{y}^T\mathbf{D}\mathbf{y} > 0.$$

Anmerkung:

Der Rechenaufwand der Cholesky-Zerlegung beträgt

$$1/6\ (n^3 + 3n^2 - 4n) = O(n^3).$$

wesentliche Operationen (Multiplikationen, Divisionen) sowie n Wurzelberechnungen. Das Vorwärts- und Rückwärtseinsetzen benötigt jeweils

$$1/2\ (n^2 + n) = O(n^2)$$

Operationen. Der vollständige Cholesky-Algorithmus erfordert somit

$$1/6\ n^3 + 3/2\ n^2 + 1/3\ n$$

wesentliche Operationen.

Anmerkung:

Bei der Berechnung der i-ten Spalte werden die entsprechenden Elemente von **A** und nur bereits von **L** berechnete Elemente, die links von dieser Spalte stehen, verwendet. Wegen der Symmetrie ist die Matrix **A** durch ihren unteren Dreiecksteil vollständig bestimmt. Das Element a_{ji} kann während der Zerlegung direkt mit l_{ji} überschrieben werden.

Will man die Information von **A** und **L** gespeichert halten, so kann man die Einträge der oberen Dreiecksmatrix von **A** verwenden und **L** bis auf die Diagonale, die in einem Zusatzvektor zu speichern ist, im strikten unteren Teil von **A** speichern.

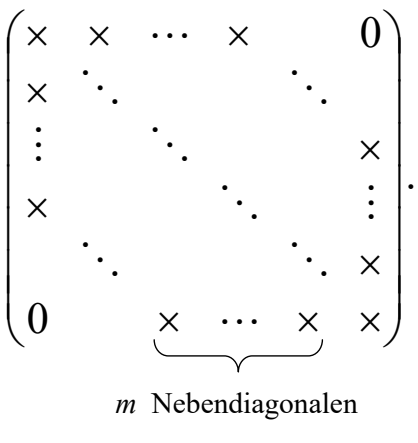
Matlab Programm (Cholesky-Algorithmus)

```
% Cholesky Zerlegung
    c = zeros(n,1); x = zeros(n,1); pd = 1;
    for i = 1:n
        s = a(i,i) – a(i,1:i–1) * a(i,1:i–1)';
        if  s <= 0
            pd = 0;          % Matrix nicht positiv definit
            break;
        end
        diag(i) = sqrt(s);
        for j = i+1:n
```

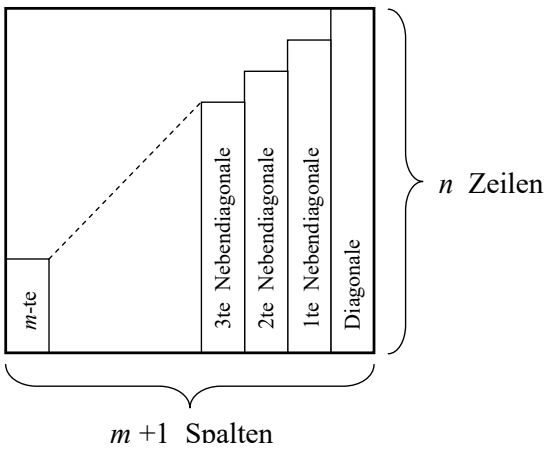
```
            a(j,i) = (a(j,i) – a(j,1:i–1) ∗ a(i,1:i–1)')/diag(i);
        end
    end
    if  pd == 0
        disp('Matrix nicht positiv definit')
    else
% Vorwärtseinsetzen  Lc = b
        for i = 1:n
            c(i) = (b(i) – a(i,1:i–1) ∗ c(1:i–1))/diag(i);
        end
% Rückwärtseinsetzen L^T x = c
        for i = n: –1:1
            x(i) = (c(i) – a(i+1:n, i)' ∗ x(i+1:n))/diag(i);
        end
    end
```


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 wie folgt gespeichert.



Bandmatrix der Bandbreite *m*



Speicherung auf einem $n \times m+1$ Feld

Diese Speicherung hat den Vorteil, dass der Zeilenindex j unverändert bleibt und sich nur der Spaltenindex ändert. Es gilt

$$\underbrace{a_{ji}}_{n\times n \text{ Matrix}} = \underbrace{a_{j,i-j+m+1}}_{n\times m+1 \text{ Matrix}}$$

Nutzt man die Bandstruktur beim Cholesky-Verfahren aus, so benötigt man nur noch Rechenoperationen der Ordnung $O(nm^2)$ in der Zerlegungsphase. Im Programm muss eine andere Index-Rechnung durchgeführt werden. Hierbei verändern sich die Grenzen der inneren Schleifen.

Tridiagonalmatrizen

Eine Bandmatrix der Bandbreite $m = 1$ heißt Tridiagonalmatrix. Zur Lösung eines linearen Gleichungssystem 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.

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 die sogenannte Skyline- Speichertechnik.

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).

Zusätzlich speichert man die Position der Diagonalelemente in

IND = (1, 2, 4, 5, 8, 10, 16).

16.1.4 Iterative Verfahren

Da der Rechenaufwand beim Gauß-Algorithmus und auch beim Cholesky-Verfahren von der Ordnung $O(n^3)$ ist, erhält man bei großen linearen Gleichungssystemen sehr lange Rechenzeiten. Um diesen Rechenaufwand zu verringern, benutzt man zur Lösung großer Gleichungssysteme besser iterative Verfahren. In diesem Kapitel werden wir iterative Verfahren behandeln, deren Grundlage das Fixpunktverfahren ist. In einem späteren Kapitel behandeln wir iterative Verfahren, deren Grundlage das Gradientenverfahren ist. Zunächst befassen wir uns mit dem allgemeinen Fixpunktverfahren.

Fixpunktverfahren

Gegeben sei $\mathbf{g} : B \subset \mathbb{R}^n \to B$ mit

$$\mathbf{g}(\mathbf{x}) = (g_1(\mathbf{x}),\ldots, g_n(\mathbf{x}))^T, \qquad \mathbf{x} = (x_1,\ldots,x_n)^T \in B$$

Gesucht ist der Fixpunkt $\tilde{\mathbf{x}} \in B$ mit

$$\mathbf{g}(\tilde{\mathbf{x}}) = \tilde{\mathbf{x}}.$$

Fixpunktverfahren

$\mathbf{x}_0 \in B$ Startvektor

$\mathbf{x}_{k+1} = \mathbf{g}(\mathbf{x}_k) \qquad k = 0,1,\ldots$

Satz 16-2: *Fixpunktsatz*

Die vektorwertige Funktion $\mathbf{g} : B \subset \mathbb{R}^n \to B$ besitze stetige partielle Ableitungen in B. Für die Funktionalmatrix

$$\mathbf{J}_\mathbf{g}(\mathbf{x}) = \begin{pmatrix} \frac{\partial g_1}{\partial x_1}(\mathbf{x}) & \cdots & \frac{\partial g_1}{\partial x_n}(\mathbf{x}) \\ \vdots & \ddots & \vdots \\ \frac{\partial g_n}{\partial x_1}(\mathbf{x}) & \cdots & \frac{\partial g_n}{\partial x_n}(\mathbf{x}) \end{pmatrix}$$

gelte

$$L = \max_{\mathbf{x}\in B} \left\| \mathbf{J}_\mathbf{g}(\mathbf{x}) \right\|_\infty < 1 \qquad (L \text{ heißt Lipschitzkonstante}).$$

Dann gilt für das Fixpunktverfahren $\mathbf{x}_{k+1} = g(\mathbf{x}_k)$ mit beliebigem Startvektor $\mathbf{x}_0 \in B$

$$\lim_{k\to\infty} \mathbf{x}_k = \tilde{\mathbf{x}},$$

mit $\tilde{\mathbf{x}}$ dem einzigen Fixpunkt von $\mathbf{g}$ in B. Überdies gelten die a priori Fehlerabschätzung

$$\left\|\mathbf{x}_k - \tilde{\mathbf{x}}\right\|_\infty \le \frac{L^k}{1-L}\left\|\mathbf{x}_1 - \mathbf{x}_0\right\|_\infty$$

und a posteriori Fehlerabschätzung

$$\left\|\mathbf{x}_k - \tilde{\mathbf{x}}\right\|_\infty \le \frac{L}{1-L}\left\|\mathbf{x}_k - \mathbf{x}_{k-1}\right\|_\infty .$$

Anwendung auf lineare Gleichungssysteme

Wir benutzen nun das Fixpunktverfahren, um gewisse lineare Gleichungssysteme näherungsweise zu lösen.

Gesamtschrittverfahren (Jacobi-Verfahren)

Gegeben sei das lineare Gleichungssystem $\mathbf{Ax} = \mathbf{b}$ mit der regulären $n \times n$ Matrix $\mathbf{A}$. Wir zerlegen die Matrix $\mathbf{A}$ folgendermaßen

$\mathbf{A} = \mathbf{L} + \mathbf{D} + \mathbf{R}$ mit

$$\mathbf{L} = \begin{pmatrix} 0 & \cdots & \cdots & 0 \\ a_{21} & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ a_{n1} & \cdots & a_{n,n-1} & 0 \end{pmatrix}, \quad \mathbf{D} = \begin{pmatrix} a_{11} & 0 & \cdots & 0 \\ 0 & \ddots & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & a_{n,n} \end{pmatrix}, \quad \mathbf{R} = \begin{pmatrix} 0 & a_{12} & \cdots & a_{1,n} \\ \vdots & \ddots & & \vdots \\ \vdots & & \ddots & a_{n-1,n} \\ 0 & \cdots & \cdots & 0 \end{pmatrix}$$

Dann erhalten wir anstelle des linearen Gleichungssystems $\mathbf{Ax} = \mathbf{b}$

$$(\mathbf{L} + \mathbf{D} + \mathbf{R})\mathbf{x} = \mathbf{b} \;\Rightarrow\; \mathbf{Dx} = \mathbf{b} - \mathbf{Lx} - \mathbf{Rx} \;\Rightarrow\; \mathbf{x} = \mathbf{g}(\mathbf{x}) := \mathbf{D}^{-1}(\mathbf{b} - \mathbf{Lx} - \mathbf{Rx})$$

Koordinatenweise ergibt dies

$$x_i = g_i(\mathbf{x}) = \frac{1}{a_{ii}}\left(b_i - \sum_{j=1}^{i-1} a_{ij}x_j - \sum_{j=i+1}^{n} a_{ij}x_j \right), \quad (i = 1,\ldots,n).$$

Für die Elemente der Funktionalmatrix $\mathbf{J_g}(\mathbf{x})$ erhalten wir

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

Also gilt für die Lipschitzkonstante L

$$L = \max_{1\le i\le n} \sum_{j=1, j\neq i}^{n} |a_{ij}| / |a_{ii}|.$$

Die Konvergenzvoraussetzung $L < 1$ bedeutet also, dass die Matrix $\mathbf{A}$ diagonaldominant sein muss, d.h.

$$|a_{ii}| > \sum_{j=1, j\neq i}^{n} |a_{ij}| \quad \forall i = 1,\ldots,n.$$

Ist also $\mathbf{A}$ diagonaldominant, so konvergiert dieses Fixpunktverfahren. Es wird Gesamtschrittverfahren oder auch Jacobi-Verfahren genannt.

Nach wählen eines Startvektors $\mathbf{x}_0$, z.B. $x_{i,0} = 1$, $i = 1,\ldots,n$ berechne

$$x_{i,k+1} = \frac{1}{a_{ii}}\left(b_i - \sum_{j=1}^{i-1} a_{ij}x_{j,k} - \sum_{j=i+1}^{n} a_{ij}x_{j,k} \right), \quad (i = 1,\ldots,n)$$

für $k = 0,1,\ldots$ bis $\|\mathbf{x}_{k+1} - \mathbf{x}_k\|_\infty < \varepsilon$ oder k zu groß (d.h. keine Konvergenz).

Ist $\tilde{\mathbf{x}}$ die gesuchte Lösung des linearen Gleichungssystems $\mathbf{Ax} = \mathbf{b}$ dann gilt mit der obigen Lipschitzkonstanten L für die Näherung $\mathbf{x}_{k+1}$ die a priori Fehlerabschätzung

$$\|\mathbf{x}_k - \tilde{\mathbf{x}}\|_\infty \le \frac{L^k}{1-L}\|\mathbf{x}_1 - \mathbf{x}_0\|_\infty .$$

Je kleiner L, d.h. je größer die ”Diagonaldominanz” von $\mathbf{A}$, desto besser ist die Konvergenz des Gesamtschrittverfahrens.

Rechenaufwand

Bei jedem Iterationsschritt wird eine Multiplikation "Matrix × Vektor" mit $O(n^2)$ Rechenoperationen durchgeführt. Werden weniger als n Iterationsschritte benötigt, so ist der Gesamtrechenaufwand kleiner $O(n^3)$, also weniger als beim Gauß-Algorithmus.

Ein weiterer Vorteil des Gesamtschrittverfahrens liegt darin, dass es sehr gut zur "Parallelisierung" geeignet ist, da jede Multiplikation "Spalte × Vektor" (Skalarprodukt) getrennt durchgeführt werden kann, also auf einem Parallelrechner gut zu realisieren ist.

<u>Einzelschrittverfahren (Seidel-Verfahren)</u>

Die Umformung des linearen Gleichungssystems

$$\mathbf{A}\mathbf{x} = (\mathbf{L} + \mathbf{D} + \mathbf{R})\mathbf{x} = \mathbf{b}$$

gemäß

$$(\mathbf{L} + \mathbf{D})\,\mathbf{x} = \mathbf{b} - \mathbf{R}\mathbf{x} \;\Rightarrow\; \mathbf{x} = \mathbf{g}(\mathbf{x}) := (\mathbf{L} + \mathbf{D})^{-1}(\mathbf{b} - \mathbf{R}\mathbf{x})$$

führt zum Einzelschritt- oder Gauß-Seidel-Verfahren.

Die Fixpunktiterationsvorschrift lautet dann

$$\mathbf{x}_{k+1} = \mathbf{g}(\mathbf{x}_k) = (\mathbf{L} + \mathbf{D})^{-1}(\mathbf{b} - \mathbf{R}\mathbf{x}_k) \;\Rightarrow\; (\mathbf{L} + \mathbf{D})\,\mathbf{x}_{k+1} = \mathbf{b} - \mathbf{R}\mathbf{x}_k$$

$$\Rightarrow\; \mathbf{x}_{k+1} = \mathbf{D}^{-1}(\mathbf{b} - \mathbf{L}\mathbf{x}_{k+1} - \mathbf{R}\mathbf{x}_k).$$

Also nach wählen eines Startvektors $\mathbf{x}_0$ berechne

$$x_{i,k+1} = \frac{1}{a_{ii}}\left(b_i - \sum_{j=1}^{i-1} a_{ij} x_{j,k+1} - \sum_{j=i+1}^{n} a_{ij} x_{j,k} \right), \quad (i = 1, \ldots, n)$$

für $k = 0,1,\ldots$ bis $\left\| \mathbf{x}_{k+1} - \mathbf{x}_k \right\|_\infty < \varepsilon$ oder k zu groß (d.h. keine Konvergenz).

Der Unterschied zum Gesamtschrittverfahren besteht darin, dass auf der rechten Seite schon die Koordinaten des neuen Iterationsvektors $\mathbf{x}_{k+1}$ 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 $\mathbf{A}$ diagonaldominant ist (hinreichendes Kriterium). In diesem Fall gilt die gleiche Fehlerabschätzung wie beim Gesamtschrittverfahren.

Ferner kann man zeigen, dass das Gauß-Seidel-Verfahren für alle symmetrischen positiv definiten Matrizen $\mathbf{A}$ konvergiert.

Überrelaxationsverfahren (Successive Over Relaxation (SOR)-Verfahren)

Hierbei führt man den zusätzlichen Relaxationsparameter ω wie folgt ein.

$$((1 - 1/\omega + 1/\omega)\mathbf{D} + \mathbf{L} + \mathbf{R})\mathbf{x} = \mathbf{b}$$

$$\Rightarrow\ (\mathbf{L} + 1/\omega\,\mathbf{D})\mathbf{x} = \mathbf{b} - (1 - 1/\omega)\mathbf{D}\mathbf{x} - \mathbf{R}\mathbf{x}$$

$$\Rightarrow\ \mathbf{x} = \mathbf{g}(\mathbf{x}) := (\mathbf{L} + 1/\omega\,\mathbf{D})^{-1}(\mathbf{b} - (1 - 1/\omega)\mathbf{D}\mathbf{x} - \mathbf{R}\mathbf{x}).$$

Als Fixpunktiterationsvorschrift erhält man dann

$$\mathbf{x}_{k+1} = \mathbf{g}(\mathbf{x}_k) = (\mathbf{L} + 1/\omega\,\mathbf{D})^{-1}(\mathbf{b} - (1 - 1/\omega)\mathbf{D}\mathbf{x}_k - \mathbf{R}\mathbf{x}_k)$$

$$\Rightarrow\ (\mathbf{L} + 1/\omega\,\mathbf{D})\mathbf{x}_{k+1} = \mathbf{b} - (1 - 1/\omega)\mathbf{D}\mathbf{x}_k - \mathbf{R}\mathbf{x}_k$$

$$\Rightarrow\ \mathbf{x}_{k+1} = \omega\,\mathbf{D}^{-1}(\mathbf{b} - (1 - 1/\omega)\mathbf{D}\mathbf{x}_k - \mathbf{L}\mathbf{x}_{k+1} - \mathbf{R}\mathbf{x}_k).$$

Koordinatenweise berechnet man ausgehend von einem Startvektor $\mathbf{x}_0$

$$x_{i,k+1} = \frac{\omega}{a_{ii}}\left(b_i - \left(1 - \frac{1}{\omega}\right) a_{ii} x_{i,k} - \sum_{j=1}^{i-1} a_{ij} x_{j,k+1} - \sum_{j=i+1}^{n} a_{ij} x_{j,k} \right), \quad (i = 1, \ldots, n)$$

für $k = 0,1,\ldots$ bis $\|\mathbf{x}_{k+1} - \mathbf{x}_k\|_\infty < \varepsilon$ oder k 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.

Matlab Programm (Überrelaxationsverfahren)

```
% Überrelaxationsverfahren
    xk = ones(n,1);              % Festlegen des Startvektors
    while  (k <= kmax & delta_max > epsilon)
           for i=1:n
               s = a(i,1:i–1) * xkp1(1:i–1) + a(i,i+1:n) * xk(i+1:n);
               xkp1(i) = (omega / a(i,i)) * (b(i) – s) + (1 – omega) * xk(i);
           end
           delta_max = max(abs(xk – xkp1));
           xk = xkp1;  k = k +1;
    end
```


Beispiel:

$\mathbf{A}$ sei die folgende symmetrische 25×25 Matrix

$$\mathbf{A} = \begin{pmatrix} \mathbf{T} & -\mathbf{E} & & & \\ -\mathbf{E} & \mathbf{T} & -\mathbf{E} & & \\ & -\mathbf{E} & \mathbf{T} & -\mathbf{E} & \\ & & -\mathbf{E} & \mathbf{T} & -\mathbf{E} \\ & & & -\mathbf{E} & \mathbf{T} \end{pmatrix} \quad \text{mit} \quad \mathbf{T} = \begin{pmatrix} 4 & -1 & & & \\ -1 & 4 & -1 & & \\ & -1 & 4 & -1 & \\ & & -1 & 4 & -1 \\ & & & -1 & 4 \end{pmatrix}$$

und $\mathbf{E}$ der 5×5 Einheitsmatrix, d.h. $\mathbf{A}$ ist eine Bandmatrix der Bandbreite $m = 5$. Außerdem sei die rechte Seite $\mathbf{b}$ mit $b_i = -1/18$ für $i = 1,\ldots,25$ gegeben.

Auf ein solches lineares Gleichungssystem wird man geführt, wenn man die Poisson-Gleichung mit Hilfe des Differenzenverfahrens auf einem rechteckigen Gebiet lösen möchte. Man kann auch leicht zeigen, dass die Matrix $\mathbf{A}$ positiv definit ist.

Differenzenverfahren zur Lösung elliptischer Differentialgleichungen führen häufig auf lineare Gleichungssysteme mit symmetrischer positiv definiter Koeffizientenmatrix $\mathbf{A}$.

Für den Startvektor $\mathbf{x}_0$ mit $x_{i,0} = 1,\ i = 1,\ldots,25$ und $\varepsilon = 10^{-8}$ liefert das SOR-Verfahren für verschiedene ω die folgenden Iterationsaufwände.

$\omega = 1 \quad \Rightarrow$ 62 Iterationen (Einzelschrittverfahren)

$\omega = 1.3 \quad \Rightarrow$ 28 Iterationen

$\omega = 1.35 \Rightarrow$ 22 Iterationen (optimales ω für dieses Beispiel)

$\omega = 1.4 \quad \Rightarrow$ 23 Iterationen.

Anmerkung:
Für bestimmte Matrizen $\mathbf{A}$ (obiges Beispiel eingeschlossen) existieren Aussagen zur optimalen Wahl des Relaxationsparameters ω (vgl. Literatur).

16.2 Nichtlineare Gleichungssysteme

16.2.1 Einführung

Gegeben seien n Gleichungen mit n Unbekannten

$$\mathbf{f}(\mathbf{x}) = \mathbf{0} \quad \text{mit} \quad \mathbf{f}(\mathbf{x}) = \begin{pmatrix} f_1(\mathbf{x}) \\ f_2(\mathbf{x}) \\ \vdots \\ f_n(\mathbf{x}) \end{pmatrix} = \begin{pmatrix} f_1(x_1,\ldots x_n) \\ f_2(x_1,\ldots x_n) \\ \vdots \\ f_n(x_1,\ldots x_n) \end{pmatrix} \text{ und } \mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$$

Gesucht wird mit $\tilde{\mathbf{x}}$ das $\mathbf{x}$, das das nichtlineare Gleichungssystem $\mathbf{f}(\mathbf{x}) = \mathbf{0}$ befriedigt.

Voraussetzungen

Die Funktionen f_i seien in einer Umgebung der gesuchten Nullstelle $\tilde{\mathbf{x}}$ 2mal stetig differenzierbar, und die Funktionalmatrix

$$\mathbf{J_f}(\mathbf{x}) = \begin{pmatrix} \frac{\partial f_1}{\partial x_1}(\mathbf{x}) & \cdots & \frac{\partial f_1}{\partial x_n}(\mathbf{x}) \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1}(\mathbf{x}) & \cdots & \frac{\partial f_n}{\partial x_n}(\mathbf{x}) \end{pmatrix} = \begin{pmatrix} (\operatorname{grad} f_1(\mathbf{x}))^T \\ \vdots \\ (\operatorname{grad} f_n(\mathbf{x}))^T \end{pmatrix}$$

von $\mathbf{f}$ sei in dieser Umgebung regulär, d.h. $\mathbf{J}_\mathbf{f}^{-1}(\mathbf{x})$ existiert.

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

$$\begin{aligned} f_i(\mathbf{x}) &= f_i(\mathbf{x}_0) + \sum_{j=1}^{n} \frac{\partial f_i(\mathbf{x}_0)}{\partial x_j}(x_j - x_{j,0}) + R_{i,1,\mathbf{x}_0} \\ &= \underbrace{f_i(\mathbf{x}_0) + (\operatorname{grad} f_i(\mathbf{x}_0))^T (\mathbf{x} - \mathbf{x}_0)}_{T_{i,1,\mathbf{x}_0}(\mathbf{x})} + R_{i,1,\mathbf{x}_0} \end{aligned}$$

Setzen wir nun nicht $f_i(\mathbf{x}) = 0$, sondern das Taylorpolynom $T_{i,1,\mathbf{x}_0}(\mathbf{x}) = 0$, so erhalten wir

$$(\operatorname{grad} f_i(\mathbf{x}_0))^T (\mathbf{x} - \mathbf{x}_0) = -f_i(\mathbf{x}_0) \quad i = 1, \ldots, n$$

Führen wir den Vektor $\Delta\mathbf{x} = \mathbf{x} - \mathbf{x}_0$ ein, so lauten diese Gleichungen in Matrixschreibweise

$$\mathbf{J_f}(\mathbf{x}_0)\Delta\mathbf{x} = -\mathbf{f}(\mathbf{x}_0).$$

Die Lösung dieses linearen Gleichungssystems sei der Vektor $\Delta\mathbf{x}$, dann ergibt sich die nächste Näherungslösung zu

$$\mathbf{x}_1 = \mathbf{x}_0 + \Delta\mathbf{x}.$$

Durch Fortsetzen dieser Vorgehensweise, erhalten wir das im folgenden näher beschriebene Newton-Verfahren.

16.2.2 Newton-Verfahren

Wähle einen Startvektor $\mathbf{x}_0$ und berechne für $k = 0,1,\ldots$

$\mathbf{J_f}(\mathbf{x}_k)\Delta\mathbf{x}_k = -\mathbf{f}(\mathbf{x}_k)$ (lineares Gleichungssystem lösen)

$\mathbf{x}_{k+1} = \mathbf{x}_k + \Delta\mathbf{x}_k$

bis $\|\mathbf{f}(\mathbf{x}_{k+1})\| < \varepsilon$ und $\|\Delta\mathbf{x}_k\| < \delta$ oder k zu groß (d.h. keine Konvergenz).

Umformulieren des linearen Gleichungssystems $\mathbf{J_f}(\mathbf{x}_k)\Delta\mathbf{x}_k = -\mathbf{f}(\mathbf{x})$ liefert

$$\mathbf{J_f}(\mathbf{x}_k)(\mathbf{x}_{k+1} - \mathbf{x}_k) = -\mathbf{f}(\mathbf{x}_k) \quad \Leftrightarrow \quad \mathbf{x}_{k+1} = \mathbf{x}_k - \mathbf{J_f}^{-1}(\mathbf{x}_k)\,\mathbf{f}(\mathbf{x}_k)$$

und damit die Erkenntnis, dass das Newton-Verfahren ein Fixpunktverfahren ist. Die Fixpunktfunktion lautet

$$\mathbf{g}(\mathbf{x}) = \mathbf{x} - \mathbf{J_f}^{-1}(\mathbf{x})\,\mathbf{f}(\mathbf{x})$$

Um die Konvergenz des Newton-Verfahrens nachzuweisen, müssen wir noch die Voraussetzungen des Fixpunktsatzes (vgl. Satz 16-2) überprüfen.

$\mathbf{g}$ ist stetig differenzierbar in einer Umgebung um $\tilde{\mathbf{x}}$ ($\tilde{\mathbf{x}}$ gesuchte Nullstelle von $\mathbf{f}$ und damit Fixpunkt von $\mathbf{g}$), da $\mathbf{f}$ 2mal stetig differenzierbar ist. Weiter muss gelten

$$L = \max_{\mathbf{x}\in U(\tilde{\mathbf{x}})} \|\mathbf{J_g}(\mathbf{x})\|_\infty < 1.$$

Aus obiger Gleichung folgt

$$\mathbf{J_f}(\mathbf{x})\,\mathbf{g}(\mathbf{x}) = \mathbf{J_f}(\mathbf{x})\,\mathbf{x} - \mathbf{f}(\mathbf{x}).$$

Differentiation nach x_i ergibt

$$\frac{\partial}{\partial x_i}\big(\mathbf{J_f}(\mathbf{x})\big)\mathbf{g}(\mathbf{x}) + \mathbf{J_f}(\mathbf{x})\frac{\partial \mathbf{g}(\mathbf{x})}{\partial x_i} = \frac{\partial}{\partial x_i}\big(\mathbf{J_f}(\mathbf{x})\big)\mathbf{x} + \mathbf{J_f}(\mathbf{x})\,\mathbf{e}_i - \frac{\partial \mathbf{f}(\mathbf{x})}{\partial x_i}.$$

Setzen wir in diese Gleichung $\mathbf{x} = \tilde{\mathbf{x}}$ (gesuchte Nullstelle von $\mathbf{f}$) ein, so erhalten wir, da $\mathbf{f}(\tilde{\mathbf{x}}) = \mathbf{0}$ und $\mathbf{g}(\tilde{\mathbf{x}}) = \tilde{\mathbf{x}}$ ($\tilde{\mathbf{x}}$ gleichzeitig Fixpunkt von $\mathbf{g}$) und weil $\mathbf{J_f}(\mathbf{x})\,\mathbf{e}_i$ und $\partial\mathbf{f}(\mathbf{x})/\partial x_i$ jeweils der i-ten Spalte von $\mathbf{J_f}(\mathbf{x})$ entsprechen,

$$\mathbf{J_f}(\tilde{\mathbf{x}})\frac{\partial \mathbf{g}(\tilde{\mathbf{x}})}{\partial x_i} = \mathbf{J_f}(\tilde{\mathbf{x}})\,\mathbf{e}_i - \frac{\partial \mathbf{f}(\tilde{\mathbf{x}})}{\partial x_i} = \mathbf{0}.$$

Ausnutzen, dass $\mathbf{J_f}(\mathbf{x})$ in einer Umgebung von $\tilde{\mathbf{x}}$ als regulär vorausgesetzt wurde, liefert

$$\frac{\partial \mathbf{g}(\tilde{\mathbf{x}})}{\partial x_i} = \mathbf{0} \quad i = 1,\ldots,n \quad \Leftrightarrow \quad \mathbf{J_g}(\tilde{\mathbf{x}}) = \mathbf{0} \quad \text{(Nullmatrix)}.$$

und damit

$$\mathbf{J_g}(\tilde{\mathbf{x}}) = \mathbf{0} \quad \Rightarrow \quad \left\|\mathbf{J_g}(\tilde{\mathbf{x}})\right\|_\infty = 0.$$

Aufgrund der Stetigkeit von $\partial g_i(\mathbf{x})/\partial x_j$ in einer Umgebung von $\tilde{\mathbf{x}}$, existiert eine Umgebung $U(\tilde{\mathbf{x}})$ mit

$$\mathbf{J_g}(\tilde{\mathbf{x}}) = \mathbf{0} \quad \Rightarrow \quad \max_{\mathbf{x}\in U(\tilde{\mathbf{x}})} \left\|\mathbf{J_g}(\tilde{\mathbf{x}})\right\|_\infty < 1.$$

und es gilt der folgende Satz.

Satz 16-3: (*Konvergenz des Newton-Verfahrens*)
Ist $\mathbf{f}$ 2mal stetig differenzierbar in einer Umgebung der gesuchten Nullstelle $\tilde{\mathbf{x}}$ und ist dort die Funktionalmatrix $\mathbf{J_f}(\mathbf{x})$ von $\mathbf{f}$ regulär, so konvergiert das Newton-Verfahren gegen die gesuchte Nullstelle $\tilde{\mathbf{x}}$, falls der Startvektor $\mathbf{x}_0$ "nahe" genug bei $\tilde{\mathbf{x}}$ gewählt wird.

Problematik: Da man in der Praxis nicht weiß, was "nahe" genug heißt, ist die Wahl eines geeigneten Startvektors $\mathbf{x}_0$ oft nicht trivial.

Unter der Voraussetzung der Konvergenz des Newton-Verfahrens und unter der zusätzlichen Voraussetzung, dass $\mathbf{f}$ 3mal stetig differenzierbar in einer Umgebung von $\tilde{\mathbf{x}}$ ist, kann man sogar quadratische Konvergenz zeigen.

Satz 16-4: (*Quadratische Konvergenz*)
Ist $\mathbf{f}$ 3mal stetig differenzierbar in einer Umgebung der gesuchten Nullstelle $\tilde{\mathbf{x}}$ und konvergiert das Newton-Verfahren, so gilt quadratische Konvergenz, d.h.

$$\left\|\mathbf{x}_{k+1} - \tilde{\mathbf{x}}\right\|_\infty \le K \left\|\mathbf{x}_k - \tilde{\mathbf{x}}\right\|_\infty^2, \quad K > 0.$$

Beweis:
Wir führen für die Fixpunktfunktion $\mathbf{g}$ eine Taylorentwicklung um $\tilde{\mathbf{x}}$ bis zur 2ten Ordnung durch, d.h.

$$x_{i,k+1} = g_i(\mathbf{x}_k) = g_i(\tilde{\mathbf{x}}) + (\mathbf{x}_k - \tilde{\mathbf{x}})^T \operatorname{grad} g_i(\tilde{\mathbf{x}}) + \frac{1}{2}(\mathbf{x}_k - \tilde{\mathbf{x}})^T \mathbf{H}_{g_i}(\boldsymbol{\xi}_i)(\mathbf{x}_k - \tilde{\mathbf{x}})$$

mit

$$\mathbf{H}_{g_i}(\boldsymbol{\xi}_i) = \left(\frac{\partial^2 g_i}{\partial x_j \partial x_k}(\boldsymbol{\xi}_i) \right)_{j,k=1,\ldots,n} \qquad \text{für} \quad i = 1,\ldots,n$$

Wegen $g_i(\tilde{\mathbf{x}}) = \tilde{x}_i$ und $\operatorname{grad} g_i(\tilde{\mathbf{x}}) = \mathbf{0}$ für $i = 1,\ldots,n$ folgt

$$\left| x_{i,k+1} - \tilde{x}_i \right| \le \frac{1}{2} \sum_{j=1}^{n} \sum_{k=1}^{n} \left| \frac{\partial^2 g_i}{\partial x_j \partial x_k}(\boldsymbol{\xi}_i) \right| \left\| \mathbf{x}_k - \tilde{\mathbf{x}} \right\|_\infty^2 \quad \text{für} \quad i = 1,\ldots,n$$

$$\Rightarrow \quad \left\| \mathbf{x}_{k+1} - \tilde{\mathbf{x}} \right\|_\infty \le K \left\| \mathbf{x}_k - \tilde{\mathbf{x}} \right\|_\infty^2$$

Anmerkung:
Quadratische Konvergenz bedeutet, dass sich mit jedem Iterationsschritt die Anzahl der richtigen Stellen ungefähr verdoppelt, denn sei

$$\left\| \mathbf{x}_k - \tilde{\mathbf{x}} \right\|_\infty < 10^{-d} \Rightarrow \left\| \mathbf{x}_{k+1} - \tilde{\mathbf{x}} \right\|_\infty < K \cdot 10^{-2d}$$

Beispiel: (*zwei Gleichungen mit zwei Unbekannten*)

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

Die Funktionalmatrix von $\mathbf{f} = (f_1, f_2)^T$ lautet

$$\mathbf{J}_\mathbf{f}(x,y) = \begin{pmatrix} \frac{\partial f_1}{\partial x}(x,y) & \frac{\partial f_1}{\partial y}(x,y) \\ \frac{\partial f_2}{\partial x}(x,y) & \frac{\partial f_2}{\partial y}(x,y) \end{pmatrix}$$

Das lineare Gleichungssystem

$$\mathbf{J_f}(x_k, y_k)\begin{pmatrix}\Delta x_k\\ \Delta y_k\end{pmatrix} = -\begin{pmatrix}f_1(x_k, y_k)\\ f_2(x_k, y_k)\end{pmatrix}$$

geht über in

$$\begin{pmatrix}\Delta x_k\\ \Delta y_k\end{pmatrix} = -\mathbf{J}_\mathbf{f}^{-1}(x_k, y_k)\begin{pmatrix}f_1(x_k, y_k)\\ f_2(x_k, y_k)\end{pmatrix}$$

Mit

$$\begin{pmatrix}x_{k+1}\\ y_{k+1}\end{pmatrix} = \begin{pmatrix}x_k\\ y_k\end{pmatrix} + \begin{pmatrix}\Delta x_k\\ \Delta y_k\end{pmatrix}$$

folgt hieraus

$$\begin{pmatrix}x_{k+1}\\ y_{k+1}\end{pmatrix} = \begin{pmatrix}x_k\\ y_k\end{pmatrix} - \mathbf{J}_\mathbf{f}^{-1}(x_k, y_k)\begin{pmatrix}f_1(x_k, y_k)\\ f_2(x_k, y_k)\end{pmatrix}$$

Für eine 2×2 Matrix lässt sich sehr einfach die inverse Matrix berechnen, es gilt

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

Damit lautet das Newton-Verfahren für den Spezialfall von zwei Gleichungen mit zwei unbekannten

$$\begin{pmatrix}x_{k+1}\\ y_{k+1}\end{pmatrix} = \begin{pmatrix}x_k\\ y_k\end{pmatrix} - \frac{1}{f_{1,x}f_{2,y} - f_{1,y}f_{2,x}}\begin{pmatrix}f_{2,y} & -f_{1,y}\\ -f_{2,x} & f_{1,x}\end{pmatrix}\begin{pmatrix}f_1\\ f_2\end{pmatrix}\Bigg|_{x=x_k,\, y=y_k}$$

Untersuchen Sie das Newton-Verfahren anhand des folgenden Beispiels.

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

Für dieses Beispiel lassen sich die Nullstellen von $\mathbf{f}$ auch exakt berechnen. Sie lauten

$$\begin{pmatrix}0\\ 0\end{pmatrix}, \quad \begin{pmatrix}3\\ 0\end{pmatrix}, \quad \begin{pmatrix}0\\ 3\end{pmatrix}, \quad \begin{pmatrix}1\\ 1\end{pmatrix}.$$

Nachteil des Newton-Verfahrens

Bei jedem Iterationsschritt müssen alle partiellen Ableitungen $\partial f_i(\mathbf{x}_k)/\partial x_j$ berechnet werden. Die Koeffizientenmatrix $\mathbf{J_f}(\mathbf{x}_k)$ ist also bei jedem Iterationsschritt neu zu berechnen, und damit ist in jedem Iterationsschritt ein komplettes lineares Gleichungssystem zu lösen.

Vereinfachungen des Newton-Verfahrens

a) Man berechnet nur für den Startvektor $\mathbf{x}_0$ die Matrix $\mathbf{J_f}(\mathbf{x}_0)$ und ersetzt dann $\mathbf{J_f}(\mathbf{x}_k)$ immer durch $\mathbf{J_f}(\mathbf{x}_0)$, d.h. die LR-Zerlegung ist nur einmal am Anfang zu berechnen. Während der Iterationsschritte 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.

b) Man berechnet nur jeden $3-5$ Iterationsschritt die Funktionalmatrix $\mathbf{J_f}(\mathbf{x}_k)$ neu. Auch dies führt im allgemeinen zu schlechterer Konvergenz oder Divergenz.

c) 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 im allgemeinen zu schlechterer Konvergenz oder Divergenz.

16.3.3 Gedämpftes Newton-Verfahren

Bei ungünstigen Startvektoren kann es passieren, dass das Newton-Verfahren divergiert. Um sicher zu gehen, dass der nächste Iterationsvektor $\mathbf{x}_{k+1}$ "besser" ist als $\mathbf{x}_k$, führt man einen Dämpfungsparameter α ein.

Gedämpftes Newton-Verfahren

Wähle einen Startvektor $\mathbf{x}_0$ und berechne für $k = 0,1,...$

$\mathbf{J_f}(\mathbf{x}_k)\Delta\mathbf{x}_k = -\mathbf{f}(\mathbf{x}_k)$ lineares Gleichungssystem lösen

wähle $\alpha = 1,\ 1/2,\ 1/4,\ 1/8,\ldots$ bis $\left\|\mathbf{f}(\mathbf{x}_k + \alpha\,\Delta\mathbf{x}_k)\right\| < (1-\alpha/4)\left\|\mathbf{f}(\mathbf{x}_k)\right\|$

$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha\,\Delta\mathbf{x}_k$

bis $\|\mathbf{f}(\mathbf{x}_{k+1})\| < \varepsilon$ und $\|\Delta\mathbf{x}_k\| < \delta$ oder α zu klein oder k zu groß (keine Konvergenz).

Damit ist gesichert, dass $\mathbf{x}_{k+1}$ eine "besser" Näherung als $\mathbf{x}_k$ 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 $(1 - \alpha/4)$ auf, damit nicht nur auf Grund von Rundungsfehlern der Wert "besser" geworden ist.

Beispiel:
Untersuchen Sie anhand des vorangegangenen Beispiels vergleichend das klassische und das gedämpfte Newton-Verfahren. Ermitteln Sie hierzu für beide Verfahren das Gebiet der Startvektoren für die Konvergenz zur Nullstelle $(1,1)^T$ erreicht wird.

16.2.4 Anwendungsbeispiel, Berechnung von Extrema

Gesucht seien die relativen Extrema einer Funktion $h(x_1,x_2,\ldots,x_n)$. Die notwendige Bedingung $\operatorname{grad} h(\mathbf{x}) = \mathbf{0}$ liefert n im allgemeinen nichtlineare Gleichungen für die n Unbekannten $x_1,x_2,\ldots,x_n$, d.h.

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

Die Funktionalmatrix enthält dann die 2ten partiellen Ableitungen

$$\left(h_{x_i x_j}(\mathbf{x})\right)_{i=1,\ldots,n;\,j=1,\ldots,n} = \left(\frac{\partial^2 h(\mathbf{x})}{\partial x_i \partial x_j}\right)_{i=1,\ldots,n;\,j=1,\ldots,n}.$$

Beispiel:
Gegeben sei die 2mal stetig differenzierbare Funktion $h(x,y)$. Gesucht sind deren relative Extrema.

Notwendige Bedingung:

$$h_x(x,y) = 0$$
$$h_y(x,y) = 0.$$

Für diesen Fall lautet das Newton-Verfahren

$$\begin{pmatrix} x_{k+1} \\ y_{k+1} \end{pmatrix} = \begin{pmatrix} x_k \\ y_k \end{pmatrix} - \frac{1}{h_{xx}h_{yy} - h_{xy}^2} \begin{pmatrix} h_{yy} & -h_{xy} \\ -h_{xy} & h_{xx} \end{pmatrix} \begin{pmatrix} h_x \\ h_y \end{pmatrix} \Bigg|_{x=x_k, y=y_k}$$

Wenden Sie das Newton-Verfahren auf das folgende Beispiel an.

$$h(x,y) = xy(3-x-y) = 3xy - x^2y - xy^2$$

Mit den partiellen Ableitungen

$$h_x(x,y) = 3y - 2xy - y^2, \quad h_y(x,y) = 3x - x^2 - 2xy$$

findet man aus den notwendigen Bedingungen

$$h_x(x,y) = 0, \quad h_y(x,y) = 0$$

die möglichen Extrema an den Stellen

$$\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \quad \begin{pmatrix} 3 \\ 0 \end{pmatrix}, \quad \begin{pmatrix} 0 \\ 3 \end{pmatrix}, \quad \begin{pmatrix} 1 \\ 1 \end{pmatrix}.$$

Mit Hilfe der hinreichenden Bedingung kann man sehen, dass nur bei $(1,1)^T$ ein relatives Extremum (Maximum) vorliegt. An den anderen Stellen befinden sich keine Extrema sondern Sattelpunkte.

Anmerkung:

Bei Extremwertaufgaben können auch andere gradientenbasierte Verfahren benutzt werden, z.B. das klassische Gradientenverfahren, das cg- und pcg-Verfahren sowie die Quasi-Newton-Verfahren.

16.3 Eigenwertberechnung bei Matrizen

16.3.1 Grundlagen

Gegeben sei eine reelle $n \times n$ Matrix $\mathbf{A}$ (mit $a_{ij} \in \mathbb{R}$). Gesucht ist $\lambda \in \mathbb{C}$ und $\mathbf{x} \in \mathbb{C}^n \backslash \{\mathbf{0}\}$ mit $\mathbf{A}\mathbf{x} = \lambda\mathbf{x}$. λ heißt Eigenwert und $\mathbf{x}$ der zu λ zugehörige Eigenvektor von $\mathbf{A}$.

Anwendungsbeispiel

Gegeben sei ein lineares inhomogenes Differentialgleichungssystem mit konstanten Koeffizienten

$$\mathbf{y}'(t) = \mathbf{A}\mathbf{y}(t) + \mathbf{b}(t).$$

Gesucht ist die Lösung des zugehörigen homogenen Differentialgleichungssystems

$$\mathbf{y}'(t) = \mathbf{A}\mathbf{y}(t).$$

Einsetzen des Lösungsansatzes

$$\mathbf{y}(t) = \mathbf{c}\, e^{\lambda t}$$

in das homogene Differentialgleichungssystem liefert

$$\mathbf{c}\,\lambda\, e^{\lambda t} = \mathbf{A}\mathbf{c}\, e^{\lambda t} \;\Rightarrow\; \mathbf{A}\mathbf{c} = \lambda\,\mathbf{c} \quad (\text{da } e^{\lambda t} \neq 0),$$

d.h. λ ist Eigenwert und $\mathbf{c}$ der zugehörige Eigenvektor von $\mathbf{A}$.

Für sehr kleine Matrizen ($n \leq 3$) kann man die Eigenwerte einer Matrix als Nullstellen des charakteristischen Polynoms $\det(\mathbf{A} - \lambda\,\mathbf{E})$ bestimmen. Bei größeren Matrizen ist der Rechenaufwand zur Berechnung der Determinante ($O(n!)$) zu groß. Außerdem ist die Berechnung der Nullstellen des charakteristischen Polynoms numerisch sehr empfindlich, da schon kleinste Störungen in den Polynomkoeffizienten große Änderungen bei den Nullstellen hervorrufen. Deshalb müssen in der Praxis andere numerische Verfahren zur Berechnung der Eigenwerte einer Matrix benutzt werden.

Im folgenden werden zwei wichtige Verfahren, das von Mises-Verfahren und das Wielandt-Verfahren behandelt, wobei das Wielandt-Verfahren eine Variante des von Mises-Verfahrens ist.

16.3.2 von Mises-Verfahren

Ziel ist die Berechnung des betraglich größten Eigenwerts von $\mathbf{A}$ und eines zugehörigen (normierten) Eigenvektors.

Vorausgesetzt sei die Diagonalisierbarkeit von $\mathbf{A}$, d.h. es existiert eine reguläre Matrix $\mathbf{C}$ mit $\mathbf{C}^{-1}\mathbf{A}\mathbf{C} = \mathbf{D}$, wobei $\mathbf{D}$ eine Diagonalmatrix bezeichnet.

Es seien $\lambda_1,\lambda_2,\ldots,\lambda_n$ die Eigenwerte von $\mathbf{A}$ mit $|\lambda_1| > |\lambda_2| \geq \ldots \geq |\lambda_n|$ und $\mathbf{s}_1,\mathbf{s}_2,\ldots,\mathbf{s}_n$ die korrespondierenden Eigenvektoren. Ferner sei $\mathbf{x}_0$ ein Startvektor mit

$$\mathbf{x}_0 = \sum_{i=1}^{n} c_i \mathbf{s}_i$$

dann gilt

$$\mathbf{A}^k \mathbf{x}_0 = \sum_{i=1}^{n} c_i \mathbf{A}^k \mathbf{s}_i = \sum_{i=1}^{n} c_i \lambda_i^k \mathbf{s}_i = \lambda_1^k \left(c_1 \mathbf{s}_1 + c_2 \left(\frac{\lambda_2}{\lambda_1} \right)^k \mathbf{s}_2 + \cdots + c_n \left(\frac{\lambda_n}{\lambda_1} \right)^k \mathbf{s}_n \right).$$

Wegen $|\lambda_i / \lambda_1| < 1$ für $i = 2,\ldots,n$ folgt

$$\left(\frac{\lambda_i}{\lambda_1} \right)^k \xrightarrow[k \to \infty]{} 0 \qquad i = 2,\ldots,n$$

Also gilt für große k

$$\mathbf{A}^k \mathbf{x}_0 \approx K \mathbf{s}_1$$

Da $K\mathbf{s}_1$ der zu λ_1 zugehörige Eigenvektor ist, konvergiert $\mathbf{A}^k\mathbf{x}_0$ für $k \to \infty$ gegen einen Eigenvektor zum betraglich größten Eigenwert λ_1 von $\mathbf{A}$. Also müssen wir die Matrix $\mathbf{A}$ mehrmals auf den Startvektor $\mathbf{x}_0$ anwenden, um näherungsweise einen Eigenvektor zu λ_1 zu erhalten.

Problematik: $\|K\mathbf{s}_1\|$ wird groß, falls $|\lambda_1| > 1$. Deshalb muss nach jedem Schritt der neue Iterationsvektor normiert werden.

Das Verfahren ist sehr einfach durchzuführen. Auf den Startvektor $\mathbf{x}_0$ wird sukzessive die Matrix $\mathbf{A}$ angewendet und nach jedem Schritt wird der neue Vektor normiert. Wir benutzen hier die ∞-Norm ($\|\cdot\|_\infty$).

von Mises-Verfahren

Wähle Startvektor $\mathbf{u}_0$ und normiere $\mathbf{u}_0$ gemäß

$$\mathbf{x}_0 = \frac{\mathbf{u}_0}{|u_{m,0}|} \quad \text{mit} \quad m = \arg\max_{1 \le i \le n} |u_{i,0}|.$$

Anschließend berechne für $k = 0,1,\ldots$

$$\mathbf{u}_{k+1} = \mathbf{A}\mathbf{x}_k\,, \qquad m = \arg\max_{1 \le i \le n} |u_{i,k+1}|$$

$$\mu_{k+1} = \frac{u_{m,k+1}}{x_{m,k}}, \qquad \mathbf{x}_{k+1} = \frac{\mathbf{u}_{k+1}}{|u_{m,k+1}|},$$

bis $|\mu_{k+1} - \mu_k| < \varepsilon$ und $\|(|x_{1,k+1}|,\ldots,|x_{n,k+1}|)^T - (|x_{1,k}|,\ldots,|x_{n,k}|)^T\| < \delta$.

Satz 16-5: (*Konvergenz des von Mises-Verfahrens*)

Es sei $\mathbf{A}$ diagonalisierbar und für die Eigenwerte λ_i von $\mathbf{A}$ gelte $|\lambda_1| > |\lambda_2| \ge \ldots \ge |\lambda_n|$. Ferner seien $\mathbf{s}_i$ $(i = 1,\ldots,n)$ die zu λ_i $(i = 1,\ldots,n)$ zugehörigen Eigenvektoren und für den Startvektor $\mathbf{x}_0$ gelte

$$\mathbf{x}_0 = \sum_{i=1}^{n} c_i \mathbf{s}_i.$$

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

$\mu_{k+1} \to \lambda_1$ (gegen den betraglich größten Eigenwert von $\mathbf{A}$)

$\mathbf{x}_k \to \mathbf{s}_1$ (gegen den zu λ_1 zugehörigen Eigenvektor)

Beweis:

Aus obiger Herleitung folgt, dass $\mathbf{x}_k \to \mathbf{s}_1$. Da $\mu_{k+1} x_{i,k} = u_{i,k+1} = (\mathbf{A}\mathbf{x}_k)_i$ und $\mathbf{x}_k \to \mathbf{s}_1$ und damit $\mathbf{A}\mathbf{x}_k \to \mathbf{A}\mathbf{s}_1 \;\Rightarrow\; \mu_{k+1} x_{i,k} \to (\mathbf{A}\mathbf{s}_1)_i = \lambda_1 s_{i,1} \;\Rightarrow\; \mu_{k+1} \to \lambda_1$.

Anmerkung:

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

Konvergenzverbesserung, Spektralverschiebung

Die Konvergenz des von Mises-Verfahrens hängt wesentlich ab von den Quotienten $|\lambda_i|/|\lambda_1|$ $(i = 2,\ldots,n)$. Je kleiner diese Quotienten sind, desto besser konvergiert das von Mises-Verfahren. Durch Verschiebung des Spektrums kann man versuchen, diese Quotienten zu verkleinern.

Betrachtet man anstelle der Matrix $\mathbf{A}$ die Matrix $\mathbf{B} = \mathbf{A} - \alpha\,\mathbf{E}$, so hat $\mathbf{B}$ die Eigenwerte $\mu_i = \lambda_i - \alpha$. Wendet man das von Mises-Verfahren auf die Matrix $\mathbf{B}$ an, so erhält man die Quotienten

$$\frac{|\mu_i|}{|\mu_1|} = \frac{|\lambda_i - \alpha|}{|\lambda_1 - \alpha|}.$$

Ziel ist es, α so zu wählen, dass

$$\frac{|\lambda_i - \alpha|}{|\lambda_1 - \alpha|} < \frac{|\lambda_i|}{|\lambda_1|}$$

gilt.

Das von Mises-Verfahren, angewendet auf die Matrix **B**, konvergiert dann (unter den entsprechenden Voraussetzungen) gegen den betraglich größten Eigenwert $\mu_m = \lambda_m - \alpha$ und einen zugehörigen Eigenvektor. Man kann mit dieser Spektralverschiebung auch andere Eigenwerte von $\mathbf{A}$ berechnen.

Es gelte $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n > 0$ und λ_1 sei schon berechnet. Wähle $\alpha \approx \lambda_1$. Dann erhält man den kleinsten Eigenwert λ_n von $\mathbf{A}$, denn $\mu_n = \lambda_n - \alpha$ ist dann der betraglich größte Eigenwert der Matrix $\mathbf{B} = \mathbf{A} - \alpha\,\mathbf{E}$.

Beispiel:

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

Eigenwerte von $\mathbf{A}$ sind $\lambda_1 = 15$, $\lambda_{2,3} = \pm 3\sqrt{5} \approx \pm 6.7082039$, $\lambda_4 = -5$.

Anwenden des von Mises-Verfahrens liefert für $\mathbf{u}_0 = (1,1,0.5,0.5)^T$

$\lambda_1 = 15$ nach 27 Iterationen, Verschiebung $\alpha = 0$,

$\lambda_3 = -3\sqrt{5}$ nach 22 Iterationen, Verschiebung $\alpha = 15$.

2) $\mathbf{B} = \begin{pmatrix} 1 & -2 & -1 \\ -4 & -7 & 7 \\ -2 & -8 & 5 \end{pmatrix}$

Eigenwerte von $\mathbf{B}$ sind $\lambda_{1,2} = \pm 3j$, $\lambda_3 = -1$. Es gilt nicht $|\lambda_1| > |\lambda_2|$.

Anwenden des von Mises-Verfahrens liefert für $\mathbf{u}_0 = (1,1,1)^T$

keine Konvergenz bei Verschiebung $\alpha = 0$,

$\lambda_{1,2} = \pm 3j$ nach 34 Iterationen, Verschiebung $\alpha = 0.5 \mp 2.5\,j$.

3) $\mathbf{C} = \begin{pmatrix} 5 & 4 & 2 \\ 4 & 5 & 2 \\ 2 & 2 & 2 \end{pmatrix}$

Eigenwerte von $\mathbf{C}$ sind $\lambda_1 = 10$, $\lambda_{2,3} = 1$.

Anwenden des von Mises-Verfahrens liefert $\mathbf{u}_0 = (1,1,1)^T$

$\lambda_1 = 10$ nach 10 Iterationen, Verschiebung $\alpha = 0$,

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

4) Hilbertmatrix (vgl. S.33)

Anwenden des von Mises-Verfahrens liefert für $\mathbf{u}_0 = (1,\ldots,1)^T$

$n = 4$: $\lambda_1 = 1.50021$ nach 11 Iterationen, Verschiebung $\alpha = 0$,
keine Konvergenz bei Verschiebung $\alpha = 1.5$.

$n = 8$: $\lambda_1 = 1.69594$ nach 13 Iterationen, Verschiebung $\alpha = 0$,
keine Konvergenz bei Verschiebung $\alpha = 1.7$.

Wenn man den betraglich kleinsten Eigenwert von $\mathbf{A}$ sucht, ist es besser, das folgende Wielandt-Verfahren zu benutzen, als eine Spektralverschiebung beim von Mises-Verfahren direkt durchzuführen.

16.3.3 Wielandt-Verfahren

Ziel ist die Berechnung des am nächsten bei einer vorgegebenen Zahl α gelegenen Eigenwerts von $\mathbf{A}$ und eines zugehörigen (normierten) Eigenvektors.

Man wendet das von Mises-Verfahren nicht auf die Matrix $\mathbf{A}$, sondern auf die Matrix $\mathbf{B} = (\mathbf{A} - \alpha\,\mathbf{E})^{-1}$ an. Die Matrix $\mathbf{B} = (\mathbf{A} - \alpha\,\mathbf{E})^{-1}$ hat die Eigenwerte $1/(\lambda_i - \alpha)$ mit zugehörigen Eigenvektoren $\mathbf{s}_i$, wenn λ_i die Eigenwerte von $\mathbf{A}$ mit zugehörigen Eigenvektoren $\mathbf{s}_i$ sind. Der betraglich größte Eigenwert von $\mathbf{B}$, also

$$\max_{1\le i\le n} \frac{1}{\left|\lambda_i - \alpha\right|} = \frac{1}{\min\limits_{1\le i\le n}\left|\lambda_i - \alpha\right|}$$

liefert dann den am nächsten bei α gelegenen Eigenwert λ_i von $\mathbf{A}$. Gleichzeitig erhält man einen zugehörigen Eigenvektor $\mathbf{s}_i$. Man benutzt bei den Iterationsschritten nicht die inverse Matrix $\mathbf{B} = (\mathbf{A} - \alpha\,\mathbf{E})^{-1}$, sondern löst $(\mathbf{A} - \alpha\,\mathbf{E})\mathbf{u}_{k+1} = \mathbf{x}_k$ mit Hilfe der LR-Zerlegung von $(\mathbf{A} - \alpha\,\mathbf{E})$.

Wielandt-Verfahren

α vorgeben, Startvektor $\mathbf{u}_0$ wählen und gemäß

$$\mathbf{x}_0 = \frac{\mathbf{u}_0}{\left|u_{m,0}\right|} \quad \text{mit} \quad m = \arg\max_{1\le i\le n}\left|u_{i,0}\right|$$

normieren. Anschließend berechne für $k = 0,1,\ldots$

$$\mathbf{LR}\,\mathbf{u}_{k+1} = \mathbf{x}_k \quad \text{mit} \quad \mathbf{LR} = (\mathbf{A} - \alpha\,\mathbf{E})$$

$$\mu_{k+1} = \alpha + \frac{x_{l,k}}{u_{l,k+1}}, \quad \mathbf{x}_{k+1} = \frac{\mathbf{u}_{k+1}}{\left|u_{l,k+1}\right|}, \quad \text{mit} \quad l = \arg\max_{1\le i\le n}\left|u_{i,k+1}\right|$$

bis $|\mu_{k+1} - \mu_k| < \varepsilon$ und $\|(|x_{1,k+1}|,\ldots,|x_{n,k+1}|)^T - (|x_{1,k}|,\ldots,|x_{n,k}|)^T\| < \delta$.

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

$\mu_{k+1} \to \lambda_1$ (gegen den am nächsten bei α gelegenen Eigenwert von $\mathbf{A}$)

$\mathbf{x}_k \to \mathbf{s}_1$ (gegen den zu λ_1 zugehörigen Eigenvektor).

Beispiel:

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

Eigenwerte von $\mathbf{A}$ sind $\lambda_1 = 15,\ \lambda_{2,3} = \pm 3\sqrt{5} \approx \pm 6.7082039,\ \lambda_4 = -5.$

Anwenden des Wielandt-Verfahrens liefert für $\mathbf{u}_0 = (1, 1, 0.5, 0.5)^T$

$\alpha = 14 \quad \Rightarrow \ \lambda_1 = 15$ nach 11 Iterationen.

$\alpha = 6 \quad \Rightarrow \ \lambda_2 = 6.7082$ nach 10 Iterationen,

$\alpha = -6 \quad \Rightarrow \ \lambda_3 = -6.7082$ nach 8 Iterationen,

$\alpha = -4.5 \ \Rightarrow \ \lambda_4 = -5$ nach 39 Iterationen,

2) Hilbertmatrix (vgl. S.33)

Anwenden des Wielandt-Verfahrens liefert für $\mathbf{u}_0 = (1, \ldots, 1)^T$

$n = 4$: $\alpha = 0 \ \Rightarrow \ \lambda_1 = 0.0000967$ nach 6 Iterationen, betraglich kleinster Eigenwert.

$n = 8$: $\alpha = 0 \ \Rightarrow \ \lambda_1 = 1.111 \cdot 10^{-10}$ nach 6 Iterationen, betraglich kleinster Eigenwert.

Anmerkung:

Berechnen eines zum Eigenwert λ gehörenden Eigenvektors $\mathbf{s}$

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

Berechnen der Konditionszahl $\mathrm{cond}(\mathbf{A})$ *einer Matrix* $\mathbf{A}$

Beim Wielandt-Verfahren erhält man mit der Verschiebung $\alpha = 0$ den betraglich kleinsten Eigenwert. Das von Mises-Verfahren liefert den betraglich größten Eigenwert. Zusammen erhält man damit die Konditionszahl $\mathrm{cond}(\mathbf{A})$ einer Matrix $\mathbf{A}$, denn es gilt bzgl. der $\|\cdot\|_2$,

$$\text{cond}(\mathbf{A}) = \sqrt{\frac{\mu_1}{\mu_n}}$$

wobei μ_1 den größten und μ_n den kleinsten Eigenwert von $\mathbf{A}^T\mathbf{A}$ angibt. Im Spezialfall einer symmetrischen Matrix $\mathbf{A}$ gilt

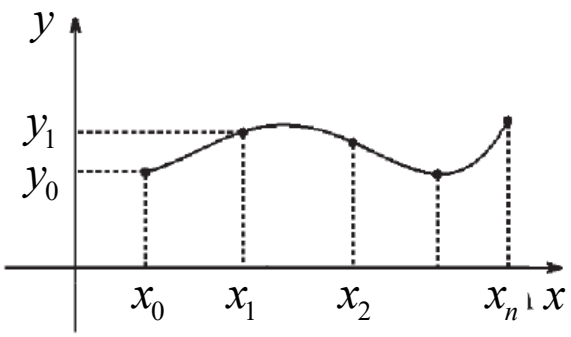
$$\text{cond}(\mathbf{A}) = \frac{|\lambda_1|}{|\lambda_n|}$$

mit λ_1 ist der größte und λ_n der kleinste Eigenwert von $\mathbf{A}$.

Das Wielandt- und von Mises-Verfahren berechnen jeweils nur einen bestimmten Eigenwert mit zugehörigem Eigenvektor. Sollen alle Eigenwerte einer Matrix $\mathbf{A}$ berechnet werden, so muss ein anderes Verfahren benutzt werden. Das Verfahren, das für die meisten Matrizen am besten geeignet ist, ist der QR-Algorithmus. Die zugehörigen Eigenvektoren müssen dann anschließend noch mit dem Wielandt-Verfahren berechnet werden.

15.4 Interpolation

Gegeben seien $n + 1$ verschiedene Stützstellen $x_0, x_1, \ldots, x_n \in \mathbb{R}$ mit den $n + 1$ dazugehörigen Stützwerten $y_0, y_1, \ldots, y_n \in \mathbb{R}$.



Gesucht wird ein Polynom $p(x)$ vom grad $p \le n$ und $p(x_i) = y_i$ für $i = 0,1,...,n$.

<u>Anwendungsbeispiele</u>

a) Messreihe: An den $n + 1$ Stellen $x_0, x_1, \ldots, x_n$ werden die Werte $y_0, y_1, \ldots, y_n$ gemessen.
   Gesucht: Polynom $p(x)$, das diese Messpunkte $(x_i,y_i)^T$ verbindet (interpoliert).

b) Funktion: $f: [a,b] \to \mathbb{R}$.

Gesucht: Polynom $p(x)$, das an $n+1$ Stellen $x_0, x_1,\ldots,x_n$ mit der Funktion übereinstimmt, d.h.: $p(x_i) = f(x_i)$ für $i = 0,1,...,n$, d.h.: Die Funktion f wird durch p interpoliert.

Im folgenden Satz wird gezeigt, dass das Interpolationsproblem (wie oben angegeben) eindeutig lösbar ist.

16.4.1 Lagrange-Interpolation

Satz 16-6: (*Interpolationsproblem*)

Gegeben seien $n+1$ verschiedene Stützstellen $x_0, x_1,\ldots,x_n \in \mathbb{R}$ mit den $n+1$ dazugehörigen Stützwerten $y_0, y_1,\ldots, y_n \in \mathbb{R}$. Dann existiert eindeutig ein Polynom $p(x)$ mit grad $p \le n$ und $p(x_i) = y_i$ für $i = 0,1,...,n$. $p(x)$ heißt Interpolationspolynom zu den Punkten $(x_i,y_i)^T$, $i = 0,1,...,n$. Es kann in der Darstellungsform von Lagrange wie folgt angegeben werden.

$$p(x) = \sum_{k=0}^{n} y_k l_k(x) \quad \text{mit} \quad l_k(x) = \frac{(x-x_0)\cdots(x-x_{k-1})(x-x_{k+1})\cdots(x-x_n)}{(x_k-x_0)\cdots(x_k-x_{k-1})(x_k-x_{k+1})\cdots(x_k-x_n)}$$

Beweis:

Existenz:
Die Lagrange-Grundpolynome l_k sind vom grad $= n \Rightarrow p$ ist ein Polynom vom grad $\le n$. Wegen

$$l_k(x_i) = \begin{cases} 0 & \text{falls } i \neq k \\ 1 & \text{falls } i = k \end{cases} \quad \Rightarrow \quad p(x_i) = \sum_{k=0}^{n} y_k l_k(x_i) = y_i, \; i = 0,1,\ldots,n$$

ist p somit Interpolationspolynom zu den Punkten $(x_i,y_i)^T$, $i = 0,1,...,n$ mit grad $p \le n$

Eindeutigkeit:
Sei r ein weiteres Interpolationspolynom mit grad $r \le n$ und $r(x_i) = y_i$ für $i = 0,1,...,n \Rightarrow q(x) = r(x) - p(x)$ ist ein Polynom vom grad $\le n$ und $q(x_i) = 0$ für $i = 0,1,...,n \Rightarrow q(x) = 0 \; \forall x \in \mathbb{R}$ (denn ein Polynom vom grad $= n$ hat höchstens n verschiedene Nullstellen) $\Rightarrow r(x) = p(x)$ für $i = 0,1,...,n$.

Beispiel:

1) $x_0 = -1, \quad x_1 = 2, \quad x_2 = 4,$

$y_0 = 15, \quad y_1 = 6, \quad y_2 = 10,$

$$l_0(x) = \frac{(x-2)(x-4)}{(-1-2)(-1-4)} = \frac{1}{15}(x-2)(x-4) = \frac{1}{15}(x^2 - 6x + 8)$$

$$l_1(x) = \frac{(x+1)(x-4)}{(2+1)(2-4)} = -\frac{1}{6}(x+1)(x-4) = -\frac{1}{6}(x^2 - 3x - 4)$$

$$l_2(x) = \frac{(x+1)(x-2)}{(4+1)(4-2)} = \frac{1}{10}(x+1)(x-2) = \frac{1}{10}(x^2 - x - 2)$$

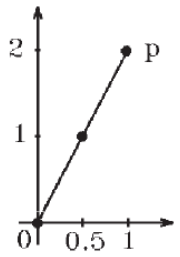
$$p(x) = y_0 l_0(x) + y_1 l_1(x) + y_2 l_2(x)$$
$$= (x^2 - 6x + 8) - (x^2 - 3x - 4) + (x^2 - x - 2) = x^2 - 4x + 10.$$

Der Grad von p kann kleiner als n werden, wie das folgende Beispiel zeigt.

2) $x_0 = 0, \quad x_1 = 1/2, \quad x_2 = 1,$

$y_0 = 0, \quad y_1 = 1, \quad y_2 = 2,$

$\Rightarrow \; p(x) = 2x$, also grad $p < 2$.



16.4.2 Approximation einer Funktion durch Interpolationspolynome

Es sei $f : [a,b] \to \mathbb{R}$, $x_0 < x_1 < x_2 < \ldots < x_n$, $x_i \in [a,b]$. Dann existiert nach Satz 16-6 das eindeutig bestimmte Interpolationspolynom p_n mit $p_n(x_i) = f(x_i)$ für $i = 0,1,...,n$.

Frage: Wie groß ist der Fehler zwischen $f(x)$ und $p_n(x)$ für $x \in [a,b]$?

Satz 16-7:

Ist $f : [a,b] \to \mathbb{R}$ $(n+1)$-mal stetig differenzierbar in $[a,b]$, und sind $n+1$ verschiedene Stützstellen $x_0 < x_1 < x_2 < \ldots < x_n$ vorgeben, dann gilt für das Interpolationspolynom p_n

$$f(x)-p_n(x)=\frac{1}{(n+1)!}f^{(n+1)}(\xi)(x-x_0)(x-x_1)\ldots(x-x_n) \text{ mit } \xi\in(a,b).$$

Beweis:

Mit $q(x)=(x-x_0)(x-x_1)\cdots(x-x_n)$ gilt

$$q(x_i)=0 \text{ für } i=0,1,\ldots,n \text{ und } q(x)\neq 0 \ \forall x\neq x_i .$$

Sei $x\in[a,b]$ mit $x\neq x_i$ für $i=0,1,\ldots,n$ fest gewählt, dann definieren wir

$$F(t)=f(t)-p_n(t)-\frac{f(x)-p_n(x)}{q(x)}q(t),\ t\in[a,b]$$

$\Rightarrow$ F ist $(n+1)$-mal stetig differenzierbar in $[a,b]$ und hat dort mindestens $n+2$ Nullstellen, denn $F(x_i)=0$ für $i=0,1,\ldots,n$ und $F(x)=0$.

Nach dem Satz von Rolle hat F' mindestens eine Nullstelle zwischen zwei Nullstellen von F, also gilt bei wiederholter Anwendung des Satzes von Rolle:

F' hat mindestens $n+1$ Nullstellen

F'' hat mindestens n Nullstellen

$\vdots$

$F^{(n+1)}$ hat mindestens 1 Nullstelle in (a,b),

d.h. $\exists\xi\in(a,b)$ mit $F^{(n+1)}(\xi)=0$.

Da

$p^{(n+1)}(t)=0 \ \forall t\in[a,b]$ (da Polynom vom grad $\leq n$)

und

$q^{(n+1)}(t)=(n+1)! \ \forall t\in[a,b]$

$$\Rightarrow F^{(n+1)}(\xi)=f^{(n+1)}(\xi)-\frac{f(x)-p_n(x)}{q(x)}(n+1)!=0$$

$$\Rightarrow f(x)-p_n(x)=\frac{1}{(n+1)!}f^{(n+1)}(\xi)q(x).$$

Satz 16-8: (*Folgerung aus Satz 16-7*)
Es gilt für den maximalen Fehler in $[a,b]$

$$\|f - p_n\|_\infty = \max_{x\in[a,b]} |f(x) - p_n(x)| \le \frac{1}{(n+1)!} \max_{x\in[a,b]} \left|f^{(n+1)}(x)\right| \max_{x\in[a,b]} |q(x)|.$$

Dieser maximale Fehler hängt von 2 Faktoren ab:

a) $\max\limits_{x\in[a,b]} \left|f^{(n+1)}(x)\right|$ (hängt nur von der Funktion f ab)

b) $\max\limits_{x\in[a,b]} |q(x)| = \max\limits_{x\in[a,b]} |(x-x_0)(x-x_1)\cdots(x-x_n)|$ (hängt nur von den Stützstellen x_i ab).

Um eine gute Approximation zu erreichen, muss man die Stützstellen x_i so wählen, dass $\max\limits_{x\in[a,b]} |q(x)|$ möglichst klein wird.

Für das Intervall $[-1,1]$ kann man zeigen, dass die Nullstellen der Tschebyscheff-Polynome $T_{(n+1)}$ diese Eigenschaft erfüllen. Man kann sogar zeigen, dass bei Wahl dieser Stützstellen die Interpolationspolynome p_n für $n \to \infty$ gleichmäßig auf $[-1,1]$ gegen f konvergieren, falls f in $[-1,1]$ stetig differenzierbar ist.

Beispiel:

1) $f(x) = e^x,\ [a,b] = [0,1],\ x_i \in [0,1],\ i = 0,1,\ldots,n.$

$$\max_{x\in[0,1]} \left|f^{(n+1)}(x)\right| = \max_{x\in[0,1]} \left|e^x\right| = e,$$

$$\max_{x\in[0,1]} |q(x)| = \max_{x\in[0,1]} |(x-x_0)(x-x_1)\cdots(x-x_n)| \le 1$$

$$\Rightarrow \|e^x - p_n(x)\|_\infty \le \frac{e}{(n+1)!}$$

$$\Rightarrow \lim_{n\to\infty} \left\| e^x - p_n(x) \right\|_\infty = 0 \qquad \text{(gleichmäßige Konvergenz in } [0,1]).$$

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

$\Rightarrow$ f ist stetig in $[0,1]$.

Wähle $x_i = \dfrac{1}{i+1}, \quad i = 0,1,\ldots,n$

$$\Rightarrow f(x_i) = \frac{1}{i+1} \sin\big((i+1)\pi\big) = 0$$

$$\Rightarrow p_n(x) = 0 \quad \forall x \in [0,1]$$

$$\Rightarrow p_n(x) \not\to f(x), \quad \text{falls } x \neq 0,\ x \neq x_i.$$

Die Berechnung der Interpolationspolynome $p_n(x)$ nach der Formel von Lagrange ist numerisch instabil, d.h. es können relativ große Rundungsfehler auftreten, denn es werden die Polynome $l_k(x)$ vom grad $= n$ zunächst mit den Faktoren y_k multipliziert und dann addiert. Bei größeren n sind die Polynome $l_k(x)$ aber sehr stark oszillierend, und es können leicht größere Rundungsfehler entstehen.

Ein weiterer Nachteil besteht darin, dass beim Hinzunehmen weiterer Stützstellen (um die Genauigkeit zu erhöhen) alle $l_k(x)$ neu berechnet werden müssen.

Der Vorteil der Lagrange-Formel liegt darin, dass bei gleicher Stützstellenwahl für verschiedene Sätze von Stützwerten die Polynome $p_n(x)$ leicht zu berechnen sind, da sich dann die $l_k(x)$ nicht verändern.

In den meisten Fällen ist aber die Berechnung mittels der Darstellungsformel von Newton vorzuziehen:

16.4.3 Newton-Interpolation

Es seien $x_0,x_1,\ldots,x_n$ $n+1$ verschiedene Stützstellen und $y_0,y_1,\ldots,y_n$ die zugehörigen Stützwerte, dann kann man das eindeutig bestimmbare Interpolationspolynom $p_n(x)$ auch in der Darstellung von Newton angegeben.

Satz 16-9: (*Newton-Darstellungsformel*)

$$p_n(x) = B_0 + B_1(x-x_0) + B_2(x-x_0)(x-x_1) + \ldots$$
$$\ldots + B_n(x-x_0)(x-x_1)\cdots(x-x_{n-1})$$

mit $B_0 = y_0$

$$B_1 = \frac{y_1 - y_0}{x_1 - x_0} := [y_0 y_1]$$

$$\vdots$$

$$B_k = \frac{[y_1 y_2 \ldots y_k] - [y_0 y_1 \ldots y_{k-1}]}{x_k - x_0} := [y_0 y_1 \ldots y_k], \quad k = 2,\ldots,n,$$

wobei

$$[y_{i-1} y_i] := \frac{y_i - y_{i-1}}{x_i - x_{i-1}}, \quad 1 \le i \le n,$$

$$[y_i y_{i+1} \ldots y_k] := \frac{[y_{i+1} y_{i+2} \ldots y_k] - [y_i y_{i+1} \ldots y_{k-1}]}{x_k - x_i}, \quad 0 \le i \le k \le n$$

die so genannten dividierten Differenzen bezeichnet.

Beweis:
Es sei p_n das Interpolationsproblem mit $p_n(x_i) = y_i$ für $i = 0,1,...,n$ und sei $x \in [a,b]$ mit $x \neq x_i$ für $i = 0,1,...,n$ eine weitere Stützstelle mit dem Stützwert $p_n(x) = y$, dann gilt

$$[y y_0] = \frac{y_0 - y}{x_0 - x} = \frac{y_0 - p_n(x)}{x_0 - x}, \quad [y y_0 y_1] = \frac{[y_0 y_1] - [y y_0]}{x_1 - x}, \quad [y y_0 y_1 y_2] = \ldots$$

$$\ldots = \frac{[y_0 y_1 y_2] - [y y_0 y_1]}{x_2 - x}, \ldots, [y y_0 y_1 \ldots y_n] = \frac{[y_0 y_1 \ldots y_n] - [y y_0 \ldots y_{n-1}]}{x_n - x}.$$

Auflösung dieser Gleichungen ergibt

$$
\begin{aligned}
p_n(x) &= y_0 + [y\,y_0](x-x_0) = y_0 + \{[y_0y_1] + [y\,y_0y_1](x-x_1)\}(x-x_0) \\
&= y_0 + [y_0y_1](x-x_0) + [y\,y_0y_1](x-x_0)(x-x_1) \\
&= y_0 + [y_0y_1](x-x_0) + [y_0y_1y_2](x-x_0)(x-x_1) \\
&\quad + [y\,y_0y_1y_2](x-x_0)(x-x_1)(x-x_2) \\
&= y_0 + [y_0y_1](x-x_0) + [y_0y_1y_2](x-x_0)(x-x_1) + \ldots \\
&\quad + [y_0y_1\ldots y_n](x-x_0)(x-x_1)\cdots(x-x_{n-1}) \\
&\quad + [y\,y_0y_1\ldots y_n](x-x_0)(x-x_1)\cdots(x-x_n)
\end{aligned}
$$

$$
\Rightarrow\ p_n(x) = r_n(x) + [y\,y_0y_1\ldots y_n](x-x_0)(x-x_1)\cdots(x-x_n)
$$

mit grad $r_n \le n$ und $p_n(x_i) = r_n(x_i),\ i = 0,1,\ldots,n.$

Da p_n eindeutig bestimmt folgt $p_n(x) = r_n(x)\ \ \forall x \in [a,b]$

$$
\Rightarrow\ p_n(x) = y_0 + [y_0y_1](x-x_0) + \ldots + [y_0y_1\ldots y_n](x-x_0)\cdots(x-x_{n-1})
$$

$$
\Rightarrow\ p_n(x) = B_0 + B_1(x-x_0) + B_2(x-x_0)(x-x_1) + \ldots
\quad \ldots + B_n(x-x_0)(x-x_1)\cdots(x-x_{n-1})
$$

Die Koeffizienten B_k werden mit folgendem Schema berechnet.

Newton-Schema $(n = 3)$

		x_0	$y_0 = B_0$			
	$x_1 - x_0$		$\frac{y_1 - y_0}{x_1 - x_0} = [y_0y_1] = B_1$			
$x_2 - x_0$		x_1	y_1	$\frac{[y_1y_2] - [y_0y_1]}{x_2 - x_0} = [y_0y_1y_2] = B_2$		
	$x_2 - x_1$		$\frac{y_2 - y_1}{x_2 - x_1} = [y_1y_2]$		$\frac{[y_1y_2y_3] - [y_0y_1y_2]}{x_3 - x_0} = B_3$	
$x_3 - x_1$		x_2	y_2	$\frac{[y_2x_3] - [y_1y_2]}{x_3 - x_1} = [y_1y_2y_3]$		
	$x_3 - x_2$		$\frac{y_3 - y_2}{x_3 - x_2} = [y_2y_3]$			
		x_3	y_3			

Beispiel:

(vgl. Beispiel auf S. 16-93)

$x_0 = -1, \quad x_1 = 2, \quad x_2 = 4,$

$y_0 = 15, \quad y_1 = 6, \quad y_2 = 10,$

Newton-Schema

$$\begin{array}{cc|l} & -1 & 15 \\ & & \frac{6-15}{3} = -3 \\ 3 & & \\ & & \frac{2-(-3)}{5} = 1 \\ 5 & 2 & 6 \\ & & \frac{10-6}{2} = 2 \\ 2 & & \\ & 4 & 10 \end{array}$$

$\Rightarrow p_2(x) = 15 - 3(x+1) + (x+1)(x-2) = x^2 - 4x + 10.$

Matlab-Programm (Newton-Interpolation)

```
% Berechnung der dividierten Differenzen
    B(1) = ys(1);
    for j = 2:n+1
        for i = n+1:–1:j
            h = xs(i)–xs(i–j+1);
            ys(i) = (ys(i)–ys(i–1))/h;
        end
        B(j) = ys(j);
    end
% Berechnung von pn(x)
    s = B(n+1);
    for i = n:–1:1
        s = s * (x–xs(i))+B(i);
    end
```

Neben der numerischen Stabilität hat die Newton-Darstellung den Vorteil, dass sich beim Hinzunehmen weiterer Stützstellen $x_{n+1}, x_{n+2}, \ldots$ die Koeffizienten $B_0, B_1, \ldots, B_n$ nicht ändern und nur die Koeffizienten $B_{n+1}, B_{n+2}, \ldots$ neu berechnet werden müssen.

Besitzen die Stützstellen alle den gleichen Abstand voneinander, d.h. äquidistante Stützstellen mit $h = x_{i+1} - x_i$ für $i = 0, \ldots, n-1$, dann folgt

$$B_0 = y_0\,, \quad B_1 = \frac{1}{h}(y_1 - y_0)\,, \quad B_2 = \frac{1}{2h^2}(y_2 - 2y_1 + y_0)\,,$$

$$B_3 = \frac{1}{3!h^3}(y_3 - 3y_2 + 3y_1 - y_0)\,, \ldots$$

und damit allgemein

$$B_k = \frac{1}{k!h^k}\sum_{l=0}^{k}\binom{k}{l}(-1)^l y_{k-l}.$$

16.4.4 Hermite-Interpolation

Zusätzlich zu den Stützwerten soll jetzt an der Stützstelle x_i auch der Ableitungswert y_i' interpoliert werden. Das Newton-Schema kann man weiterhin anwenden, wenn die Stützstelle x_i im Schema zweimal aufgeführt und die sich ergebende dividierte Differenz $[y_i y_i]$ durch den Ableitungswert y_i' ersetzt wird. Der Rest des Newton-Schemas wird wie bisher berechnet.

Das Interpolationspolynom p_{n+1} erfüllt dann die zusätzliche Eigenschaft $p'_{n+1}(x_i) = y_i'$.

Der Grad des Interpolationspolynoms wird hierdurch um eins erhöht, da ja eine Stützstelle x_i hinzugekommen ist.

	x_{i-1}	y_{i-1}	
	x_i	y_i	
0			y'_i
	x_i	y_i	
	x_{i+1}	y_{i+1}	

Die Fehlerabschätzung aus Satz 16-7, gilt analog, wobei auf der rechten Seite der Faktor $(x - x_i)$ zweimal aufzuführen ist.

Beispiel:

Die Sinus-Funktion soll in $x_0 = 0$ und in $x_1 = \pi/2$ mitsamt ihrer Ableitung interpoliert werden. Es soll also gelten

$$p(0) = \sin 0 = 0, \qquad p'(0) = \sin' 0 = \cos 0 = 1,$$
$$p(\pi/2) = \sin(\pi/2) = 1, \quad p'(\pi/2) = \sin'(\pi/2) = \cos(\pi/2) = 0.$$

Mit Hilfe des Newton-Schemas (s. nächste Seite) erhalten wir

$$p_3(x) = 0 + 1\cdot(x-0) + \frac{4-2\pi}{\pi^2}(x-0)(x-0) + \frac{4\pi-16}{\pi^3}(x-0)(x-0)(x-\pi/2)$$
$$= x + \frac{4-2\pi}{\pi^2}x^2 + \frac{4\pi-16}{\pi^3}x^2(x-\pi/2).$$

Probe:

$$p_3(0) = 0, \quad p_3(\pi/2) = 1, \quad p_3'(0) = 1, \quad p_3'(\pi/2) = 0.$$

Newton-Schema

x_i	y_i			
0	0			
		1		
0	0		$\frac{2/\pi - 1}{\pi/2} = \frac{4-2\pi}{\pi^2}$	
		$2/\pi$		$\frac{-4/\pi^2 - (4-2\pi)/\pi^2}{\pi/2} = \frac{4\pi-16}{\pi^3}$
$\pi/2$	1		$\frac{-2/\pi}{\pi/2} = -\frac{4}{\pi^2}$	
		0		
$\pi/2$	1			

Fehlerabschätzung

$$\left|f(x)-p_3(x)\right| \le \frac{1}{4!}\max_{\xi\in[0,\pi/2]}\left|f^{(4)}(\xi)\right|\left|(x-0)^2(x-\pi/2)^2\right| \le \frac{x^2(x-\pi/2)^2}{24}$$

$$\Rightarrow \max_{x\in[0,\pi/2]}\left|f(x)-p_3(x)\right| \le \frac{1}{24}\max_{x\in[0,\pi/2]}\left(x(x-\pi/2)\right)^2 \le \frac{1}{24}\left(\frac{\pi}{4}\right)^4 \le 0{,}016.$$

Bei der Polynom-Interpolation erhält man bei einer großen Anzahl von Stützstellen Polynome hohen Grades, die an den Rändern des Intervalls zu starken Oszillationen neigen.

In solchen Fällen bevorzugt man deshalb eine Spline-Interpolation, d.h. eine stückweise Interpolation durch Polynome kleinen Grades (z.B. dritten Grades bei den kubischen Splines) und an den Verknüpfungsstellen möglichst glatte Übergänge (z.B. zweimal stetige Differenzierbarkeit bei den kubischen Splines).

16.4.5 Spline-Interpolation

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 aneinander stoßen, d.h. bei kubischen Splines, dass die Übergänge 2mal stetig differenzierbar sein sollen. Damit erhalten wir insgesamt eine 2mal stetig differenzierbare Interpolationsfunktion $s(x)$, die stückweise aus Polynomen vom grad ≤ 3 besteht.

Konstruktion der 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.

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

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) und (6) erhalten wir

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

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

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

und zusammengefasst für $0 \le i \le n-1$ schließlich

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

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

Wir haben somit die unbekannten Koeffizienten a_i,b_i,c_i und d_i für $0 \le i \le n-1$, durch die bekannten Stützwerte y_i und die noch unbekannten Werte y_i'' (2te Ableitung an den Stelle 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.

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

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 $(i-1)$ statt 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

$$\begin{aligned} & h_{i-1}(2y_i'' + y_{i-1}'') + h_i(2y_i'' + y_{i+1}'') = 6(y_{i+1} - y_i)/h_i - 6(y_i - y_{i-1})/h_{i-1} \\ & \Rightarrow \; h_{i-1}y_{i-1}'' + 2(h_{i-1} + h_i)y_i'' + h_i y_{i+1}'' = \delta_i, \quad 1 \le i \le n-1 \\ & \text{mit} \quad \delta_i = 6(y_{i+1} - y_i)/h_i - 6(y_i - y_{i-1})/h_{i-1}. \end{aligned}$$

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 das folgende lineare Gleichungssystem.

$$\begin{pmatrix} 2(h_0+h_1) & h_1 & & & & \\ h_1 & 2(h_1+h_2) & h_2 & & & \\ & h_2 & \ddots & \ddots & & \\ & & \ddots & \ddots & h_{n-2} & \\ & & & h_{n-2} & 2(h_{n-2}+h_{n-1}) & \end{pmatrix} \begin{pmatrix} y_1'' \\ y_2'' \\ \vdots \\ y_{n-2}'' \\ y_{n-1}'' \end{pmatrix} = \begin{pmatrix} \tilde{\delta}_1 \\ \delta_2 \\ \vdots \\ \delta_{n-2} \\ \tilde{\delta}_{n-1} \end{pmatrix}$$

mit

$$\tilde{\delta}_1 = \delta_1 - h_0 y_0'' = \frac{6}{h_1}(y_2 - y_1) - \frac{6}{h_0}(y_1 - y_0) - h_0 y_0''$$

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

$$\tilde{\delta}_{n-1} = \delta_{n-1} - h_{n-1} y_n'' = \frac{6}{h_{n-1}}(y_n - y_{n-1}) - \frac{6}{h_{n-2}}(y_{n-1} - y_{n-2}) - h_{n-1} y_n''.$$

Die Koeffizientenmatrix ist eine symmetrische, diagonaldominante, positiv definite Tridiagonalmatrix (Bandmatrix der Brandbreite 1). Das lineare Gleichungssystem lässt sich mit Hilfe des Cholesky-Verfahrens für Bandmatrizen lösen.

Als Lösung dieses linearen Gleichungssystems erhalten wir die Werte y_i'', $1 \le i \le n-1$, wenn man vorher die beiden Werte y_0'' und y_n'' gewählte hat. Aus den Werten y_i'' lassen sich dann die Koeffizienten a_i, b_i, c_i und d_i, $0 \le i \le n-1$, der Teilpolynome $s_i(x)$ mit Hilfe der Formeln auf S. 114 bestimmen. Damit ist die kubsiche Spline-Funktion $s(x)$ bekannt.

Äquidistante Stützstellen

Sind alle $h_i = h$ (h = Schrittweite), also $x_i = x_0 + ih$, $0 \le i \le n$, so lautet das lineare Gleichungssystem (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 setzten, d.h.

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

z.B. $\alpha = \beta = 1$ oder $\alpha = \beta = 0.5$. Diese Wahl verändert im linearen Gleichungssystem nur die erste und letzte Zeile

$$\begin{pmatrix} a_{11} & h_1 & & & \\ h_1 & 2(h_1+h_2) & h_2 & & \\ & h_2 & \ddots & \ddots & \\ & & \ddots & \ddots & h_{n-2} \\ & & & h_{n-2} & a_{n-1,n-1} \end{pmatrix} \begin{pmatrix} y_1'' \\ y_2'' \\ \vdots \\ y_{n-2}'' \\ y_{n-1}'' \end{pmatrix} = \begin{pmatrix} \delta_1 \\ \delta_2 \\ \vdots \\ \delta_{n-2} \\ \delta_{n-1} \end{pmatrix}$$

mit

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

und

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

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

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

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

$$\begin{pmatrix} 2h_0 & h_0 & & & \\ h_0 & 2(h_0+h_1) & h_1 & & \\ & \ddots & \ddots & \ddots & \\ & & h_{n-2} & 2(h_{n-2}+h_{n-1}) & h_{n-1} \\ & & & h_{n-1} & 2h_{n-1}) \end{pmatrix} \begin{pmatrix} y_0'' \\ y_1'' \\ \vdots \\ y_{n-1}'' \\ y_n'' \end{pmatrix} = \begin{pmatrix} \tilde{\delta}_0 \\ \delta_1 \\ \vdots \\ \delta_{n-1} \\ \tilde{\delta}_n \end{pmatrix}$$

mit

$$\tilde{\delta}_0 = 6\big((y_1-y_0)/h_0 - y_0'\big), \quad \tilde{\delta}_n = 6\big(y_n' - (y_n - y_{n-1})/h_{n-1}\big)$$

und

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

Denn aus Gleichung (3) und (4) von S. 113 folgt

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

$$\Rightarrow\ 2h_0 y_0'' + h_0 y_1'' = 6\big((y_1 - y_0)/h_0 - y_0'\big).$$

bzw.

$$s_{n-1}'(x_n) = 3a_{n-1}h_{n-1}^2 + 2b_{n-1}h_{n-1} + c_{n-1}$$

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

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

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

Beispiel:

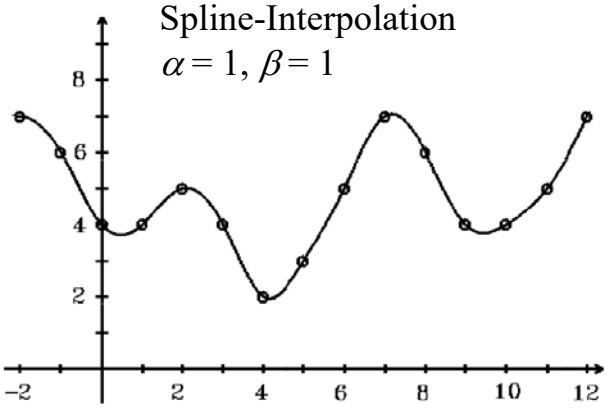
15 Interpolationspunkte

x_i = −2, −1, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12

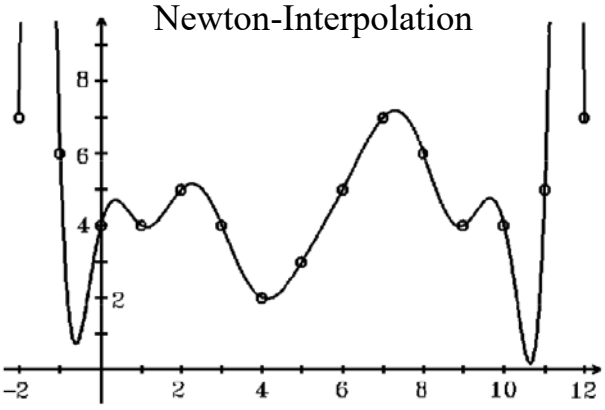
y_i = 7, 6, 4, 4, 5, 4, 2, 3, 5, 7, 6, 4, 4, 5, 7

Spline-Interpolation

$\alpha = 1, \beta = 1$



Newton-Interpolation



Der Kurvenverlauf der Newton-Interpolation weist im Vergleich zur Spline-Interpolation an den Enden starke Oszillationen auf.

16.5 Numerik Partieller Differentialgleichungen

Es sei das Randwertproblem einer partiellen Differentialgleichung zu lösen.

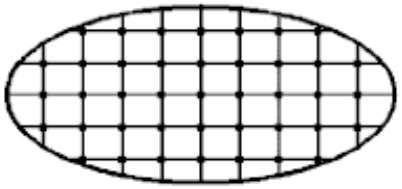
Beispiel: (*Elliptische Differentialgleichung*)
Gegeben sei $G \subset \mathbb{R}^2$ Normalbereich sowie

$$u_{xx}(x,y) + u_{yy}(x,y) = f(x,y) \quad \text{und} \quad u(x,y) = \varphi(x,y) \;\; \forall (x,y)^T \in \partial G$$

mit $f : G \to \mathbb{R}$ stetig auf G und $f : \partial G \to \mathbb{R}$ stetig auf ∂G. Gesucht wird die Lösung u auf G mit $u \in C^2(G)$ und $u \in C(G \cup \partial G)$.

16.5.1 Differenzenverfahren

Man legt ein Netz über G und erhält die Gitterpunkte (x_i, y_j).



Ferner bezeichnet man die Näherungslösung von $u(x,y)$ im Punkt (x_i, y_j) mit u_{ij} .

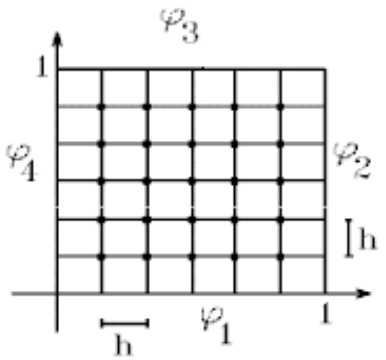
Für $(x_i, y_j) \in \partial G$, d.h. für einen Randpunkt ist $u(x_i, y_j) = \varphi(x_i, y_j)$ als Randwert gegeben.

Die Ableitungen u_{xx} und u_{yy} werden für die inneren Gitterpunkte durch Differenzenquotienten

$$u_{xx}(x,y) \approx \frac{1}{h^2}\left(u_{i-1,j} - 2u_{ij} + u_{i+1\,j}\right) \quad \text{und} \quad u_{yy}(x,y) \approx \frac{1}{h^2}\left(u_{i,j-1} - 2u_{ij} + u_{i,j+1}\right)$$

approximiert. Hierbei bezeichnen h und k die Gitterpunktabstände in x- bzw. y-Richtung.

Beispiel: (*rechteckiges/quadratisches Gebiet*)
Es sei $h = k = 1/n$, $x_i = hi$, $y_j = kj$ und
$f_{ij} = f(x_i, y_j)$ für $0 \le i, j \le n$.



Die folgenden Punkte sind Randpunkte

$u_{i,0} = u(x_i, 0) = \varphi_1(x_i) = \varphi(x_i, 0) \qquad 0 \le i \le n$
$u_{0,j} = u(1, y_j) = \varphi_2(y_j) = \varphi(1, y_j) \qquad 0 \le j \le n$

$$u_{i,n} = u(x_i, 1) = \varphi_3(x_i) = \varphi(x_i, 1) \qquad 0 \le i \le n$$
$$u_{n,j} = u(0, y_j) = \varphi_4(y_j) = \varphi(0, y_j) \qquad 0 \le j \le n.$$

Nach Multiplikation mit h^2 erhalten wir die Gleichungen

$$\left(u_{i-1,j} - 2u_{ij} + u_{i+1,j}\right) + \left(u_{i,j-1} - 2u_{ij} + u_{i,j+1}\right) = h^2 f_{ij} \qquad 1 \le i, j \le n-1.$$

Das sind $(n-1)^2$ Gleichungen für die $(n-1)^2$ Unbekannten u_{11}, u_{12},..., $u_{n-1,n-1}$, d.h. die Näherungslösungen an den inneren Gitterpunkten. Für z.B. $i = j = 1$ erhalten wir die Gleichung

$$\underbrace{u_{01}}_{\text{Randpunkt}} - 2u_{11} + u_{21} + \underbrace{u_{10}}_{\text{Randpunkt}} - 2u_{11} + u_{12} = h^2 f_{11},$$

wobei u_{01} und u_{10} Randpunkte sind. Nachdem wir diese Randpunkte auf die rechte Seite gebracht und anschließend auf beiden Seiten mit (-1) multipliziert haben, lautet die Gleichung

$$4u_{11} - u_{12} - u_{21} = -h^2 f_{11} + u_{01} + u_{10}$$

In Matrixschreibeweise lassen sich die $(n-1)^2$ Gleichungen für die $(n-1)^2$ Unbekannten dann z.B. für $n = 5$ in dem folgenden linearen GLS darstellen.

$$\mathbf{A}\mathbf{u} = -h^2\mathbf{f} + \mathbf{r}$$

mit

$$\mathbf{A} = \begin{pmatrix} \mathbf{T} & -\mathbf{E} & & \\ -\mathbf{E} & \mathbf{T} & -\mathbf{E} & \\ & -\mathbf{E} & \mathbf{T} & -\mathbf{E} \\ & & -\mathbf{E} & \mathbf{T} \end{pmatrix}, \qquad \mathbf{T} = \begin{pmatrix} 4 & -1 & & \\ -1 & 4 & -1 & \\ & -1 & 4 & -1 \\ & & -1 & 4 \end{pmatrix}$$

sowie

$$\mathbf{u} = \left(u_{11}, u_{12}, u_{13}, u_{14}, u_{21}, \ldots, u_{41}, u_{42}, u_{43}, u_{44}\right)^T$$
$$\mathbf{f} = \left(f_{11}, f_{12}, f_{13}, f_{14}, f_{21}, \ldots, f_{41}, f_{42}, f_{43}, f_{44}\right)^T$$

und

$$\mathbf{r} = \left(u_{01} + u_{10}, u_{02}, u_{03}, u_{04} + u_{15}, u_{20}, 0, 0, u_{25}, \ldots\right)^T .$$

Die Koeffizientenmatrix **A** ist eine $(n-1)^2 \times (n-1)^2$ 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.

Beispiel:

$$u_{xx}(x,y) + u_{yy}(x,y) = 0$$

$$\text{in } G = \left\{ \begin{pmatrix} x \\ y \end{pmatrix} \in \mathbb{R}^2 : \ 0 < x, y < 1 \right\} \quad \text{(Rechteck)}$$

a) *Randbedingung:*

$$u(x,0) = \varphi_1(x) = x^2, \qquad u(1,y) = \varphi_2(y) = 1 - y^2,$$
$$u(x,1) = \varphi_3(x) = x^2 - 1, \quad u(0,y) = \varphi_4(y) = -y^2.$$

Exakte Lösung:

$$u(x,y) = x^2 - y^2$$

Ergebnis mittels 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	$1{,}5 \cdot 10^{-11}$

b) *Randbedingung:*

$$u_{|\partial G} = \sin(\pi x) + \sin(\pi y)$$

Exakte Lösung:

$$u(x,y) = \Big[\sin(\pi x)\big(\sinh(\pi y) - \sinh\big(\pi(y-1)\big)\big) + \sin(\pi y)\big(\sinh(\pi x) - \sinh\big(\pi(x-1)\big)\big)\Big] \Big/ \sinh \pi$$

Ergebnis mittels 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	$2{,}4 \cdot 10^{-3}$

Algorithmus:

für $u_{xx}(x,y) + u_{yy}(x,y) = f$ in $G = \left\{ \begin{pmatrix} x \\ y \end{pmatrix} \in \mathbb{R}^2 : \ 0 < x, y < 1 \right\}$ $h = \dfrac{1}{n}$

mit den Randbedingungen

$$u(x,0) = \varphi_1(x), \quad u(1,y) = \varphi_2(y), \quad u(x,1) = \varphi_3(x), \quad u(0,y) = \varphi_4(y).$$

% Erzeugen der Bandmatrix
% Erzeugen der rechten Seite
Cholesky- oder iteratives Verfahren anwenden $\Rightarrow$ u_{ij}

Anmerkung :

Liegen andere Grundgebiete (als Rechtecke) vor, so müssen evtl. in der Nähe des Randes andere Differenzenquotienten gewählt werden (siehe z.B. Schwarz: Numerische Mathematik). Dann erhält man andere Koeffizientenmatrizen. Diese sind aber im Allgemeinen immer noch symmetrische Bandmatrizen.