Vorlesungsskript

Praktische Mathematik I

Volker Mehrmann

MA 468, Tel 314-25736

email: mehrmann@math.tu-berlin.de

Fakultät II für Mathematik und
Naturwissenschaften
Institut für Mathematik
Technische Universität Berlin

Literaturverzeichnis

[1] Bjoerck, Åke / Dahlquist, Germund *Numerische Methoden*. Oldenbourg, München 1972.

[2] Bunse, Wolfgang / Bunse–Gerstner, Angelika. *Numerische lineare Algebra*. Teubner, Stuttgart 1985.

[3] Braess, Dietrich. *Finite Elemente*. Springer, Berlin 1991.

[4] Coddington, Earl.A. / Levinson, N. *Theory of ordinary differential equations*. McGraw-Hill, New York 1955.

[5] Deuflhard, Peter / Hohmann, Andreas. *Numerische Mathematik*. de Gruyter, Berlin 1991.

[6] Deuflhard, Peter / Bornemann. *Numerische Mathematik II*. de Gruyter, Berlin 1994.

[7] Faires, J. Douglas / Burden, Richard, L. *Numerische Methoden*. Spektrum, Heidelberg 1994.

[8] Forster, Otto. *Analysis I*. 5. Aufl. Vieweg, Braunschweig 1999.

[9] Goering, Herbert / Roos, Hans-Görg / Tobiska, Lutz. *Finite Elemente Methode*. Akademie Verlag, Berlin 1988.

[10] Golub, Gene H. / Van Loan, Charles F. *Matrix Computations*. 3. Aufl., Johns Hopkins University Press, Baltimore, 1996.

[11] Golub, Gene H. / Ortega, James M. *Scientific Computing*. Teubner, Stuttgart 1996.

[12] Hairer, Ernst / Noersett, Syvert Paul / Wanner, Gerhard. *Solving ordinary differential equations*. Springer, Berlin 1987.

[13] Higham, Nicholas J. *Accuracy and Stability of Numerical Algorithms*. SIAM, Philadelphia 1996.

[14] Locher, Franz. *Numerische Mathematik für Informatiker*. Springer, Berlin 1992.

[15] Opfer, Gerhard. *Numerische Mathematik für Anfänger*. Vieweg, Braunschweig 1993.

[16] Plato, Robert. *Numerische Mathematik kompakt*. Vieweg, Braunschweig 2000.

[17] Roos, Hans-Görg / Schwetlick, Hubert. *Numerische Mathematik*. Teubner, Stuttgart 1999.

[18] Schaback, Robert / Werner, Helmut. *Numerische Mathematik*. Springer, Berlin 1992.

[19] Schäfer, Michael. *Numerische Mathematik im Maschinenbau*, Springer, Berlin 1999.

[20] Schwarz, Hans Rudolf. *Numerische Mathematik*. Teubner, Stuttgart, 1988.

[21] Schwarz, Hans Rudolf. *Methode der Finite Elemente*. Teubner, Stuttgart 1984.

[22] Schwetlick, Hubert / Kretschmar, Horst. *Numerische Mathematik für Naturwissenschaftler und Ingenieure*. Fachbuchverlag, Leipzig 1991.

[23] Stoer, Josef / Bulirsch, Roland. *Numerische Mathematik – eine Einführung, Band I,II*. Springer, Berlin 1989.

[24] Walter, Wolfgang. *Gewöhnliche Differentialgleichungen*. 3. Aufl. Springer, Heidelberg 1985.

[25] Wilkinson, James H. *Rundungsfehler*, Springer, Berlin, 1969.

Inhaltsverzeichnis

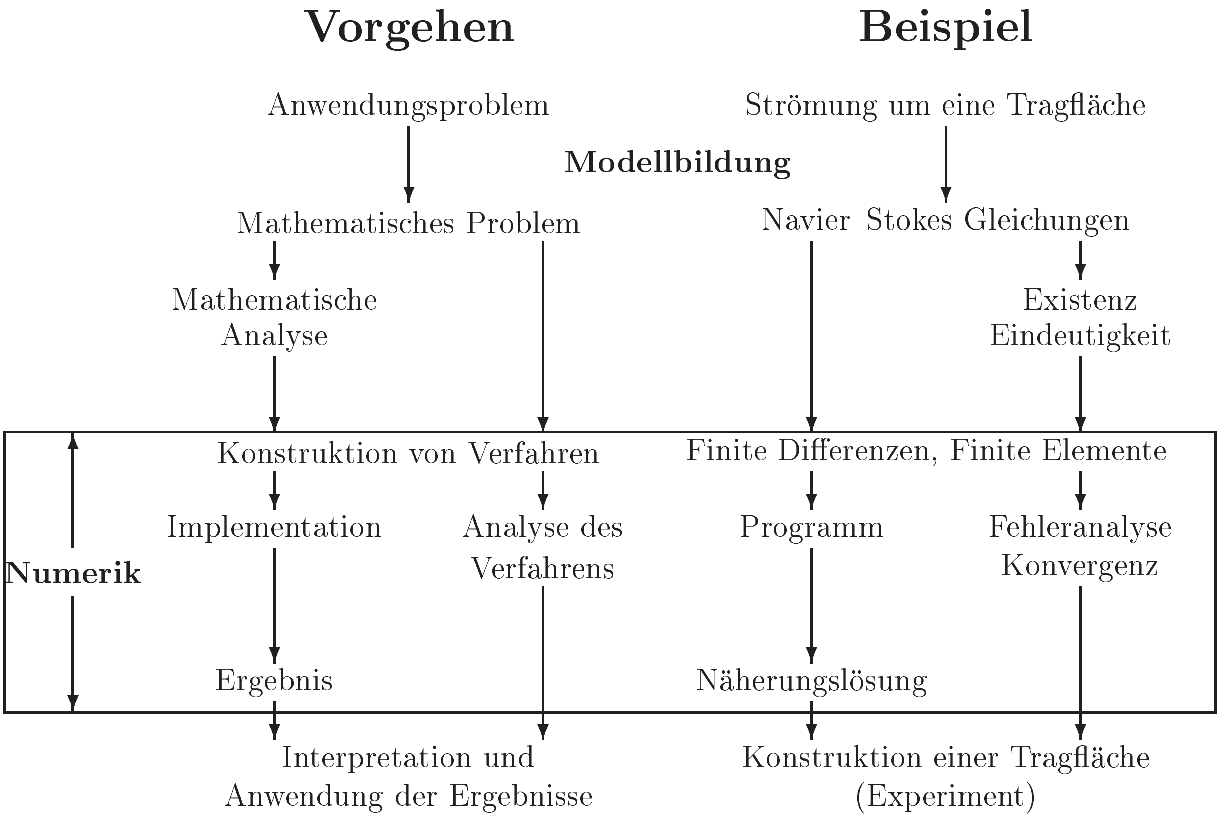
Kapitel 1

Einführung

Was ist überhaupt: **Numerische Mathematik?**

Um eine grobe Vorstellung von diesem Teilgebiet der Mathematik zu erhalten, stellen wir uns vor, wir wollen ein Flugzeug konstruieren und zu diesem Zweck erst mal verstehen, wie die Strömung um einen Tragfläche funktioniert.

Wie gehen wir vor?



Vorgehen
Beispiel
Anwendungsproblem
Strömung um eine Tragfläche
Modellbildung
Mathematisches Problem
Navier–Stokes Gleichungen
Mathematische Analyse
Existenz Eindeutigkeit
Konstruktion von Verfahren
Finite Differenzen, Finite Elemente
Implementation
Analyse des Verfahrens
Programm
Fehleranalyse Konvergenz
Numerik
Ergebnis
Näherungslösung
Interpretation und Anwendung der Ergebnisse
Konstruktion einer Tragfläche (Experiment)

Beispiel 1 Freier Fall

Wir betrachten den aus der Physik bekannten freien Fall. Ein Stein wird aus dem Turm von Pisa (Höhe 44.12 *m) fallen gelassen.* **Wann schlägt er auf?**

Modellbildung:

Als Vereinfachung vernachlässigen wir die Reibung und nehmen an, dass der Stein eine Punktmasse darstellt.

Hier machen wir einen kleinen Modell(ierungs)fehler.

Mathematisch wird die Fallbewegung als eine Funktion des zurückgelegten Weges zur Zeit t *beschrieben. Sei* $h(t)$ *die Höhe zum Zeitpunkt* t*, dann ist die Geschwindigkeit des Steines gegeben durch die erste Ableitung von* h *nach der Zeit:*

$$v(t) = \frac{dh}{dt} = h'(t).$$

Die Beschleunigung a *des Steines ist definiert als die Ableitung der Geschwindigkeit* v *nach der Zeit, also gilt:*

$$\begin{aligned} a(t) &= v'(t) \\ &= \frac{dv}{dt} \\ &= \frac{d^2h}{dt^2}. \end{aligned}$$

Für die Erdbeschleunigung g *nehmen wir den konstanten Wert* $9.81\ \frac{m}{s^2}$.

Dies ist ein Näherungswert, wir machen Fehler in den Daten, und wir vereinfachen, denn eigentlich hängt g **von der Höhe ab.**

Damit wissen wir, dass für $a(t)$ *gilt:*

$$a(t) = -g$$

Die Geschwindigkeit des Steines zur Zeit t *läßt sich nun durch Integration bestimmen:*

$$\begin{aligned} v(t) &= \int a(t)dt \\ &= -gt + c_1. \end{aligned}$$

Analog dazu wird nun durch eine weitere Integration $h(t)$ bestimmt:

$$\begin{aligned} h(t) &= \int v(t)dt \\ &= \int(-gt+c_1)dt \\ &= -\frac{g}{2}t^2+c_1t+c_2. \end{aligned}$$

Wir erhalten also als allgemeine Lösung unseres Problems, dass der Stein nach

$$\begin{aligned} t &= \frac{c_1 \pm \sqrt{c_1^2+4c_2\frac{g}{2}}}{2\frac{g}{2}} \\ &= \frac{c_1 \pm \sqrt{c_1^2+2c_2g}}{g} \end{aligned}$$

Sekunden aufschlägt.

Um dies explizit zu machen, brauchen wir die Integrationskonstanten c_1, c_2. Den Wert von c_1 können wir sofort berechnen, da bekannt ist, dass sich der Stein zum Zeitpunkt $t = 0$ im Zustand der Ruhe befindet. Es muß nur noch $v(0) = 0\frac{m}{s}$ gelöst werden:

$$v(0) = -g \cdot 0 + c_1 = 0\ \frac{m}{s}$$

$$\Rightarrow \qquad c_1 = 0.$$

Die Konstante c_2 wird analog ermittelt ($h(0) = 44.12\ m$):

$$h(0) = -\frac{g}{2} \cdot 0 + c_1 \cdot 0 + c_2 = 44.12\ m$$

$$\Rightarrow \qquad c_2 = 44.12\ m.$$

Wir können nun diese Werte und g einsetzen und erhalten als Lösung die Fallzeit:

$$t = \pm\frac{\sqrt{44.12 * 19.62}}{9.81}.$$

Nun müssen wir den Rechner bemühen, um einen Zahlenwert zu bestimmen, und wir erhalten die numerische Lösung (auf 3 Stellen genau):

$$t = 3.00\ s.$$

Hier haben wir weitere Fehler gemacht, nämlich beim Wurzelziehen und beim Runden auf die 3. Stelle.

Alle diese Fehler können bei komplizierten Problemen zu Katastrophen führen !

Was können wir tun um sicherzustellen, dass das Ergebnis annähernd korrekt ist ?

Wir könnten zB. nach Pisa fahren und ein Experiment machen. Oder wir könnten unseren Lösungsalgorithmus analysieren oder wir könnten unser Modell verbessern, indem wir zB. Reibungsterme hinzufügen.

Analyse eines Lösungverfahrens:

Anwendungsproblem	Beispiel: Freier Fall	Fehler
Modellbildung Math. Problem	 Diffgl. $h'' = -9.81$	**Modellfehler** **Datenfehler**
Math. Analyse — Numer. Methode anal. Lsg — numer. Lsg Impl. und Analyse	Lösung $t = 3.00$ Programm — Fehleranal.	**Verfahrensfehler** **Rundungsfehler**
exakte Lsg Näherung		Gesamtfehler

Wir werden uns hier mit der Entwicklung und Analyse von numerischen Methoden beschäftigen.

Kapitel 2

Anfangswertaufgaben Teil I

Wir betrachten zuerst Anfangswertaufgaben für gewöhnliche Differentialgleichungen:

$$y' = \frac{\mathrm{d}y}{\mathrm{d}t} = f(t, y(t)), \tag{2.1}$$

in einem Interval $\mathbb{I} = [t_0, t_0 + a] \subset \mathbb{R}$ mit einer Anfangsbedingung $y(t_0) = y_0 \in \mathbb{R}^n$. Die Lösung y ist dabei eine Funktion $y : \mathbb{I} \to \mathbb{R}^n$ und die rechte Seite ist eine Funktion $f : \mathbb{I} \times \mathbb{R}^n \to \mathbb{R}^n$.

Dabei kann y natürlich ein Vektor sein.

Beispiel 2 *Betrachte das Beispiel 1 und führe neue Variablen wie folgt ein. Sei* $z_1(t) = h(t)$, $z_2(t) = h'(t)$ *und* $z = \begin{bmatrix} z_1(t) \\ z_2(t) \end{bmatrix}$, *so lautet die Anfangswertaufgabe in (1) dann*

$$z'(t) = \begin{bmatrix} z_1(t) \\ z_2(t) \end{bmatrix}' = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} z_1(t) \\ z_2(t) \end{bmatrix} + \begin{bmatrix} 0 \\ -g \end{bmatrix}, \ z(0) = \begin{bmatrix} 44.12 \\ 0 \end{bmatrix}.$$

Die Lösungskurve $y(t)$ nennt man auch *Trajektorie*. Ohne die Vorgabe des Anfangswertes erhalten wir eine Schar von Trajektorien (Richtungsfeld). Durch den Anfangswert wird eine bestimmte Trajektorie festgelegt.

Beispiel 3 Abhängigkeit der Lösung vom Anfangswert.

Betrachte die Differentialgleichung:
$y' = 3y,\ y(0) = y_0.$
Als Lösung ergibt sich: $y(t) = y_0 e^{3t}$

Lösungsverlauf für Startwerte $y_0 = 0.01, 0.05, 0.1, 0.5$

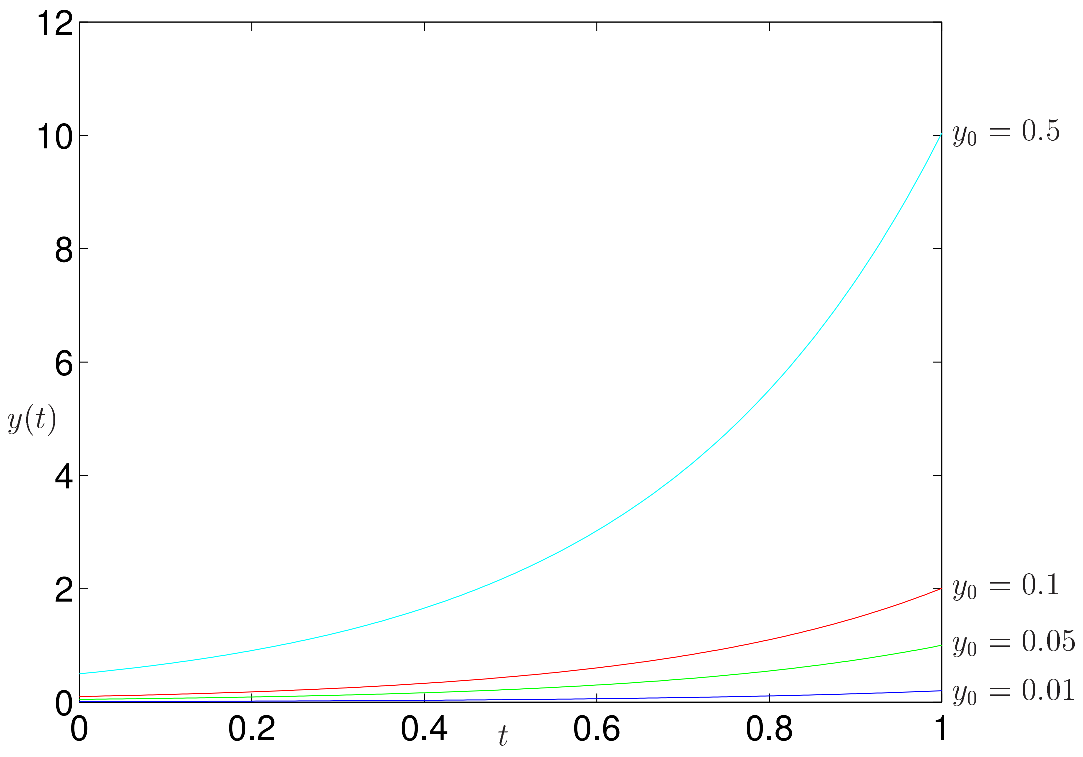


Abbildung 2.1: Richtungsfeld

Kleine Änderungen der Anfangswerte können zu sehr grossen Änderungen in der Lösung führen. Dies wird im allgemeinen *Instabilität der Differentialgleichung* genannt. (Wie wir später sehen werden, wäre ein anderer Begriff besser.)

Beispiel 4 Instabilität der Differentialgleichung

$$y' = 10(y - \frac{t^2}{1+t^2}) + \frac{2t}{(1+t^2)^2} \qquad y(0) = y_0$$

Als Lösung ergibt sich: $\quad y(t) = y_0 e^{10t} + \dfrac{t^2}{1+t^2}.$

Für $y_0 = 0$ *lautet die Lösung:* $\dfrac{t^2}{1+t^2} = \hat{y}(t)$

Für $y_0 = -\varepsilon$ *lautet die Lösung:* $-\varepsilon e^{10t} + \dfrac{t^2}{1+t^2} = \tilde{y}(t).$

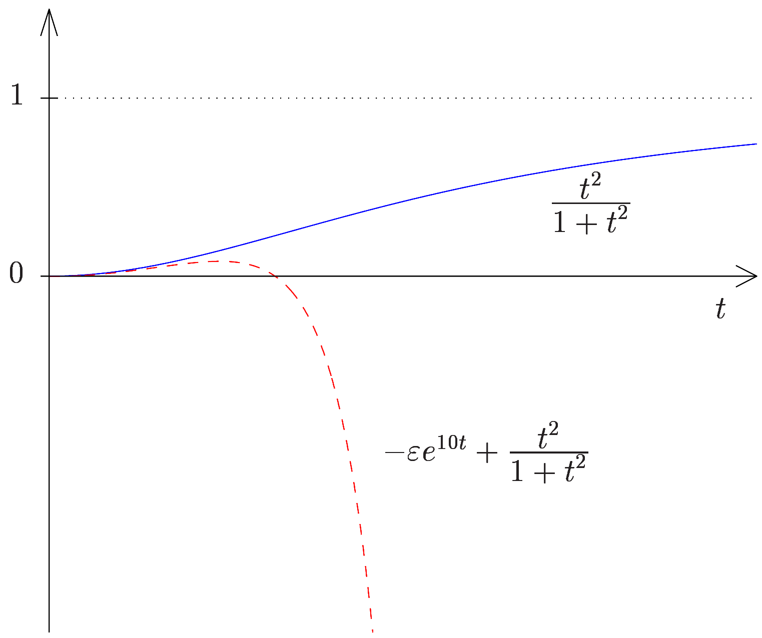


Abbildung 2.2: Instabile Differentialgleichung

Probleme bei denen dies auftritt sind auf dem Rechner sehr schwer zu lösen und man kann generell der Lösung nicht vertrauen!

Im allgemeinen sind wir nicht in der Lage die Lösung einer Differentialgleichung analytisch zu bestimmen. Wir verwenden daher numerische Verfahren.

Wie löst man nun so eine Differentialgleichung numerisch (auf dem Rechner) ?

Zur numerischen Lösung einer Differentialgleichung wird diese diskretisiert, das heißt die Lösung wird in einzelnen Schritten berechnet.

Wie gehen im einfachsten Fall wie folgt vor. Wir führen in unserem Intervall $\mathbb{I}$ ein Gitter ein.

$$\mathbb{I}_h = \{t_0, t_1, \ldots, t_N = t_0 + a\}$$

wobei wir als erste Idee einfach eine äquidistante Unterteilung mit

$$t_i = t_0 + ih, \quad i = 1, \ldots, N$$

verwenden. Dann suchen wir Werte

$$u_i = u_h(t_i), \quad i = 1, \ldots, N \tag{2.2}$$

wobei $u_h : \mathbb{I}_h \to \mathbb{R}^n$ eine Gitterfunktion ist. Diese ist so zu wählen, dass u_i den Wert

$$y_i = y(t_i) \tag{2.3}$$

der exakten Lösung von (2.1) möglichst gut annähert (approximiert).

Das heißt, auf dem Gitter $\mathbb{I}_h$ mit der Schrittweite $h \neq 0$ sollen Näherungswerte u_i für y_i an den äquidistanten Punkten $t_i = t_0 + ih$ berechnet werden.

2.1 Das Polygonzugverfahren von Euler

Das einfachste Verfahren ist das sogenannte explizite Euler-Verfahren [1].

Für die Ableitung y' gilt näherungsweise:

$$\frac{y(t+h) - y(t)}{h} \approx y'(t) = f(t, y(t)), \tag{2.4}$$

d.h.

$$y(t+h) \approx y(t) + hf(t, y(t)).$$

[1] Leonhard Euler, 1707–1783, Schweizer Mathematiker

Man erhält so an den Stellen t_i Näherungswerte u_i für y_i:

$$\begin{aligned} u_0 &= y_0 \\ \frac{u_{i+1}-u_i}{h} &= f(t_i,u_i), \qquad i=0,\ldots,N-1 \end{aligned} \tag{2.5}$$

oder umgeformt:

$$u_{i+1} = u_i + hf(t_i,u_i) \tag{2.6}$$

Die Integrationsmethode von Euler benutzt in den einzelnen Näherungspunkten (x_i,u_i) die Steigung des durch die Differentialgleichung definierten Richtungsfeldes dazu, den nächstfolgenden Näherungswert u_{i+1} zu bestimmen.

Wegen der anschaulich geometrischen Konstruktion der Näherung bezeichnet man das Verfahren auch als Polygonzugmethode. Sie ist offensichtlich sehr grob und kann sicher nur für kleine Schrittweiten h gute Näherungswerte liefern.

Beispiel 5 Das Eulerverfahren

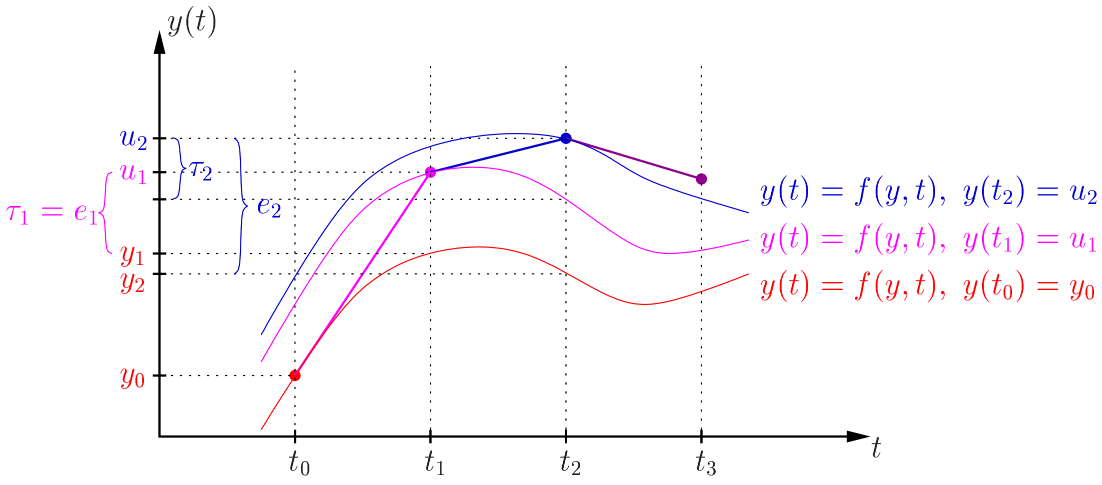


Abbildung 2.3: Euler-Verfahren

2.2 Allgemeine Einschrittverfahren

Das explizite Euler-Verfahren ist ein Spezialfall eines allgemeinen expliziten Einschrittverfahrens der Form

$$\begin{aligned} u_0 &= y_0, \\ u_{i+1} &= u_i + h\Phi(t_i, u_i, h, f). \end{aligned} \tag{2.7}$$

Beim Eulerverfahren gilt also

$$\Phi(t_i, u_i, h, f) = f(t_i, u_i). \tag{2.8}$$

Die Funktion Φ heißt Inkrementfunktion. Sie beschreibt die zugrundeliegende Methode, wie aus der bekannten Information (t_i, u_i) und für die Schrittweite h der neue Näherungswert zu berechnen ist.

Bemerkung 6 *Durch komponentenweise Betrachtung lassen sich diese Verfahren direkt auf den Vektorfall übertragen:*

$$\vec{u}_{i+1} = \begin{pmatrix} u_1 \\ \vdots \\ u_n \end{pmatrix}_{i+1} = \begin{pmatrix} u_1 \\ \vdots \\ u_n \end{pmatrix}_i + h \begin{pmatrix} \Phi_1(t_i, \vec{u}_i) \\ \vdots \\ \Phi_n(t_i, \vec{u}_i) \end{pmatrix}. \tag{2.9}$$

2.3 Fehlerbetrachtung

Wie gut ist nun so ein Verfahren ?

Gesucht ist eine Abschätzung für den *globalen Fehler*.

$$e_j = y_j - u_j = y(t_j) - u_h(t_j). \tag{2.10}$$

Um e_j abzuschätzen, muß man erst den *lokalen Fehler* (den Fehler in jedem einzelnen Schritt der Diskretisierung) abschätzen.

Sei y die exakte Lösung von (2.1) und sei $y(\hat{t}) = z$. Dann wird definiert:

$$\Delta(\hat{t}, z, h, f) := \begin{cases} \frac{y(\hat{t}+h)-z}{h} & \text{für } h \neq 0 \\ f(\hat{t}, z) & \text{für } h = 0. \end{cases} \tag{2.11}$$

Für die Näherung aus dem Einschrittverfahren (2.7) gilt:

$$\frac{u_h(\hat{t}+h)-u_h(\hat{t})}{h}=\Phi(\hat{t},z,h,f). \tag{2.12}$$

Die Größe

$$\tau(\hat{t},z,h,f):=h(\Delta(\hat{t},z,h,f)-\Phi(\hat{t},z,h,f)) \tag{2.13}$$

heißt *lokaler Diskretisierungsfehler* in $(\hat{t},z)$.

Hier ist τ ist das Ergebnis, wenn man $y(t)$ für u_h in (2.12) einsetzt. d.h., τ gibt an, wie gut die exakte Lösung die verwendete Integrationsvorschrift in einem einzelnen Schritt erfüllt. Man beachte, dass in einigen anderen Lehrbüchern der lokale Diskretisierungsfehler als $\frac{\tau}{h}$ definiert ist.

Im Fall des Euler-Verfahrens besitzt der lokale Diskretisierungsfehler die unmittelbare Bedeutung der Differenz zwischen dem exakten Wert und dem erhaltenen Näherungswert, falls zuvor vom exakten Wert ausgegangen wurde. (Siehe Beispiel 5.)

Damit ein Einschrittverfahren vernünftig ist, fordert man dass

$$\lim_{h\to 0}(\Delta(\hat{t},z,h,f)-\Phi(\hat{t},z,h,f))=\lim_{h\to 0}\frac{1}{h}\tau(\hat{t},z,h,f)=0. \tag{2.14}$$

Da aber

$$\lim_{h\to 0}\Delta(\hat{t},z,h,f)=y'(\hat{t})=f(\hat{t},z),$$

so ist (2.14) äquivalent mit

$$\Phi(\hat{t},z,0,f)=\lim_{h\to 0}\Phi(\hat{t},z,h,f)=f(\hat{t},z). \tag{2.15}$$

Definition 7 *Das Einschrittverfahren (2.7) heißt* konsistent*, wenn für alle $\hat{t}\in[t_0,t_0+a]$ und alle f, deren partielle Ableitungen nach t und y stetig und beschränkt sind, gilt*

$$\lim_{h\to 0}\frac{1}{h}\tau(\hat{t},z,h,f)=0. \tag{2.16}$$

Proposition 8 *Das Eulerverfahren ist konsistent.*

Beweis. Wir wenden die Taylorentwicklung auf y an, d.h.,

$$y(\hat{t}+h) = y(\hat{t}) + hy'(\hat{t}) + \frac{h^2}{2!}y''(\hat{t}) + \ldots + \frac{h^p}{p!}y^{(p)}(\hat{t}+\Theta h).$$

Für die Ableitungen gilt:

$$\begin{aligned}
y(\hat{t}) &= z \\
y'(\hat{t}) &= f(\hat{t}, y(\hat{t})) \\
y''(\hat{t}) &= \frac{\mathrm{d}}{\mathrm{d}t} f(t, y(t))\Big|_{t=\hat{t}} \\
&= \frac{\partial}{\partial t} f(t, y(t))\Big|_{t=\hat{t}} + \frac{\partial}{\partial y} f(t, y(t)) y'(t)\Big|_{t=\hat{t}} \\
&=: f_t(\hat{t}, z) + f_y(\hat{t}, z) f(\hat{t}, z),
\end{aligned}$$

wobei f_t die partielle Ableitung von f nach t, f_y entsprechend die partielle Ableitung nach y bezeichnet. Also folgt:

$$\begin{aligned}
\Delta(\hat{t}, z, h, f) &= y'(\hat{t}) + \frac{h}{2!}y''(\hat{t}) + \ldots + \frac{h^{p-1}}{p!}y^{(p)}(\hat{t}+\Theta h) \\
&= f(\hat{t}, z) + \frac{h}{2}\big(f_t(\hat{t}, z) + f_y(\hat{t}, z) f(\hat{t}, z)\big) + (\text{Terme h. Ord.}).
\end{aligned}$$

Beim Eulerverfahren ist $\Phi(\hat{t}, z, h, f) = f(\hat{t}, z)$. Also folgt für den lokalen Diskretisierungsfehler:

$$\begin{aligned}
\frac{\tau}{h}(\hat{t}, z, h, f) &= \Delta(\hat{t}, z, h, f) - \Phi(\hat{t}, z, h, f) \\
&= \frac{h}{2}\big(f_t(\hat{t}, z) + f_y(\hat{t}, z) f(\hat{t}, z)\big) + (\text{Terme h. Ord.}) \\
&= \mathcal{O}(h) \to 0 \text{ für } h \to 0,
\end{aligned}$$

und damit ist das Euler-Verfahren konsistent. □

2.4 Abschätzung des globalen Fehlers

Bisher haben wir nur den lokalen Fehler betrachtet.

Was können wir über den globalen Fehler sagen ?

Theorem 9 *Es genüge $\Phi(t,y,h,f)$ auf einem Streifen $\mathbb{I}\times\mathbb{R}^n$ einer Lipschitzbedingung mit Lipschitzkonstante L bezüglich y, dh.*

$$|\Phi(t,y_1,h,f)-\Phi(t,y_2,h,f)|\le L|y_1-y_2|,$$

wobei $|\ |$ der Betrag oder (falls $n>1$) eine Norm ist. Dann gilt für den globalen Fehler

$$e_m = y(t_m)-u_m, \qquad m=0,\dots,N$$

folgende Abschätzung:

$$|e_m|\le\left(|e_0|+\frac{(t_m-t_0)}{h}\tau_h\right)e^{L(t_m-t_0)}, \qquad m=0,\dots,N. \tag{2.17}$$

Dabei ist $\tau_h=\max_j|\tau_j|$ und $\tau_j=\tau(t_j,y_j,h,f)$ der lokale Diskretisierungsfehler bei t_j.

Beweis. Es gilt

$$\begin{aligned}
u_{j+1} &= u_j+h\Phi(t_j,u_j,h,f),\\
y(t_{j+1}) &= y(t_j)+h\underbrace{\left(\Phi(t_j,y(t_j),h,f)+\frac{1}{h}\tau(t_j,y(t_j),h,f)\right)}_{\Delta},
\end{aligned}$$

und damit erhalten wir

$$\begin{aligned}
e_{j+1} &= y(t_{j+1})-u_{j+1}\\
&= \underbrace{y(t_j)-u_j}_{e_j}+\tau(t_j,y(t_j),h,f)+h[\Phi(t_j,y(t_j),h,f)\\
&\qquad -\Phi(t_j,u_j,h,f)]\\
\Rightarrow\qquad |e_{j+1}| &\le |e_j|+\tau_h+hL|e_j|\\
&= (1+hL)|e_j|+\tau_h
\end{aligned}$$

Mit dem nachfolgenden Lemma 10 folgt:

$$\begin{aligned}
|e_m| &\le \left(|e_0|+\sum_{j=0}^{m-1}|\tau_j|\right)e^{mLh}\\
&\le (|e_0|+m\tau_h)e^{mLh}\\
&= (|e_0|+\frac{(t_m-t_0)}{h}\tau_h)e^{L(t_m-t_0)}
\end{aligned}$$

wegen $mh = t_m - t_0$. □

Falls $\tau_h = \mathcal{O}(h^{p+1})$ und $e_0 = 0$, so geht $|e_m| \to 0$ für $h \to 0$ wie $(t_m - t_0)h^p e^{L(t_m - t_0)}$.

Lemma 10 *Falls* $\rho_i \geq 0, \eta_i \geq 0, z_i \geq 0, i = 0, 1, 2, \ldots$ *und*

$$z_m \leq (1 + \rho_{m-1}) z_{m-1} + \eta_{m-1} \qquad \forall m = 1, 2, \ldots \tag{2.18}$$

so gilt

$$z_m \leq \left(z_0 + \sum_{j=0}^{m-1} \eta_j \right) e^{\sum\limits_{i=0}^{m-1} \rho_i} \qquad \forall m = 1, 2, \ldots \tag{2.19}$$

Beweis. Per Induktion:

$m = 1$:

$$z_1 \leq (1 + \rho_0) z_0 + \eta_0 \leq (z_0 + \eta_0) e^{\rho_0}$$

$m \to m + 1$

$$\begin{aligned} z_{m+1} &\leq (1 + \rho_m) z_m + \eta_m \\ &\overset{\text{I.V.}}{\leq} (1 + \rho_m) \left(z_0 + \sum_{j=0}^{m-1} \eta_j \right) e^{\sum\limits_{i=0}^{m-1} \rho_i} + \eta_m \end{aligned}$$

Dann gilt

$$\left(z_0 + \sum_{j=0}^{m} \eta_j \right) e^{\sum\limits_{i=0}^{m} \rho_i} = \left(z_0 + \sum_{j=0}^{m-1} \eta_j \right) e^{\sum\limits_{i=0}^{m-1} \rho_i} \underbrace{e^{\rho_m}}_{\geq 1 + \rho_m} + \eta_m \underbrace{e^{\sum\limits_{i=0}^{m} \rho_i}}_{\geq 1}$$

und damit folgt die Behauptung. □

Im Falle des Euler-Verfahrens ist die Lipschitzbedingung gerade die übliche Lipschitzbedingung an die Funktion, die typischerweise für die Existenz- und Eindeutigkeit der Lösung (siehe z.B. [4, 24]) benötigt wird.

Bemerkung 11 *Diskussion der Ergebnisse.*

- **Falls die Lipschitzkonstante L gross ist, so ist die Differentialgleichung instabil (siehe Beispiel 4). Dann wird ein Fehler am Anfangswert, aber auch jeder lokale Diskretisierungsfehler stark verstärkt.**

- **Die Konstante L ist leider im allgemeinen nicht bekannt.**

- **Ist L gross, so ist das eine schlechte Eigenschaft des mathematischen Modells, und es ist dringend notwendig zu überprüfen, ob dieses Problem im realen physikalischen Problem auch auftritt. Wenn ja, so ist dieses Anwendungsproblem mit numerischen Methoden nur sehr schlecht zugänglich. Wenn nein, so sollte die Modellierung überprüft werden.**

- **Ein weiterer Verstärkungsfaktor für den Fehler tritt auf, wenn wir über sehr lange Zeiten rechnen, d.h. wenn $t_N - t_0$ sehr gross ist, denn dann addieren sich auch viele kleine, in jedem Schritt gemachte Fehler. Man sollte überprüfen, ob das wirklich gewollt ist.**

- **Wenn aber L und $t_N - t_0$ beide nicht sehr gross sind, sollten wir einen kleinen globalen Fehler erhalten, wenn der lokale Fehler klein ist.**

Sind wir damit nicht eigentlich fertig?

Wir bräuchten ja nur $h \to 0$ schicken, dann wird $\frac{\tau_h}{h}$ klein und damit auch der Fehler.

Schauen wir mal an was auf dem Rechner passiert, wenn $h \to 0$.

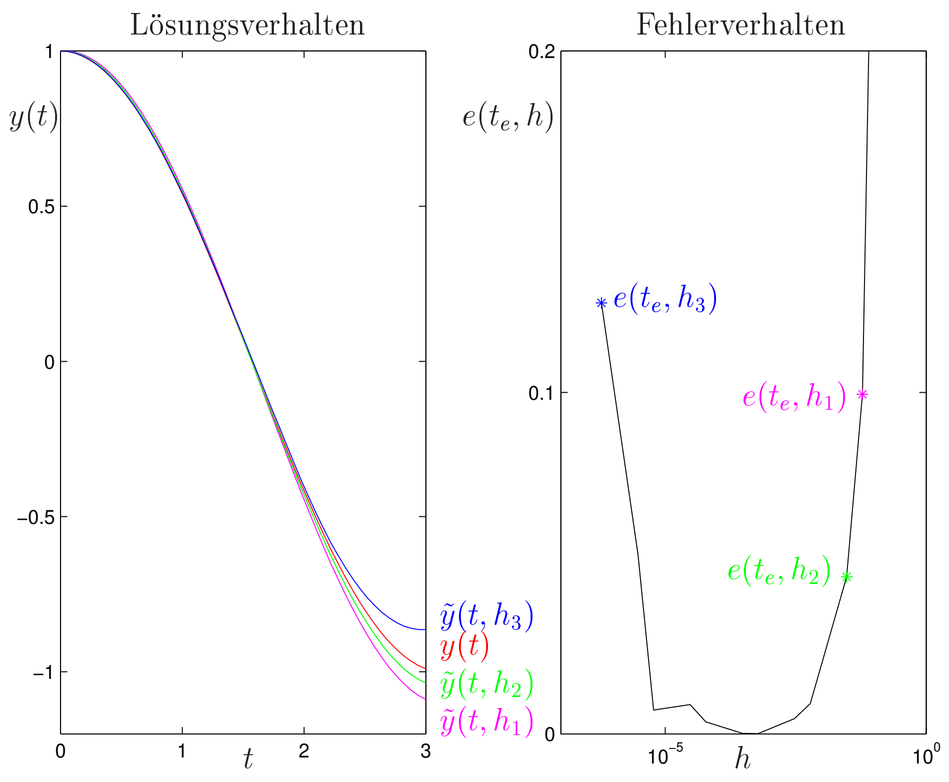
Beispiel 12 *Betrachte das Anfangswertproblem:*

$$y' = -\tan(t)y, \ y(0) = 1$$

Als Lösung ergibt sich:

$$y(t) = \cos(t).$$

Lösungsverhalten für Schrittweiten
$h_1 = 3/50,\ h_2 = 3/100,\ h_3 = 3/5000000$



Anmerkung: *Die Ergebnisse wurden mit einfacher Genauigkeit unter F77 berechnet.*

Kapitel 3

Fehleranalyse

In diesem Kapitel wollen wir uns anschauen was auf einem Rechner passiert, wenn wir ein numerisches Verfahren implementieren.

3.1 Rechnerarithmetik

Die Zahlendarstellung auf dem Rechner verwendet den folgenden Satz:

Theorem 13 (p–adische Entwicklung)
Ist $r \in \mathbb{R}, p \in \mathbb{N}, p > 1$, so gibt es ein eindeutiges $j \in \{0, 1\}$, ein eindeutiges $l \in \mathbb{Z}$ und für alle $k \in \mathbb{Z}$ mit $k \leq l$, eindeutige $\gamma_k \in \mathbb{Z}$, so dass

$$r = (-1)^j \sum_{k=-\infty}^{l} \gamma_k p^k \tag{3.1}$$

wobei $\gamma_l \neq 0$ für $r \neq 0$, $j = l = 0$ für $r = 0$ gilt, ausserdem $0 \leq \gamma_k < p$ für alle $k \leq l$ und $\gamma_k < p - 1$ für unendliche viele $k \leq l$.

Beweis. Siehe z.B. [8]. □

Bemerkung 14 *Die letzte Bedingung schließt Zahlen wie $3.9999\bar{9}$ aus.*

Beispiel 15 *Betrachte $r = 18.5$ und $p = 2$.*

$$\begin{aligned} r &= (-1)^0(1*2^{-1}+0*2^0+1*2^1+0*2^2+0*2^3+1*2^4) \\ &= 10010.1 \textit{ Dualzahldarstellung oder Binärdarstellung} \\ p &= 16 \\ r &= (-1)^0(8*16^{-1}+2*16^0+1*16) \\ &= 12.8 \textit{ Hexadezimaldarstellung} \\ & \quad \textit{Ziffern } (0123456789ABCDEF). \end{aligned}$$

Die Konvertierung zwischen verschiedenen Darstellungen geschieht durch Division durch p mit Rest für ganzzahligen Anteil und Multiplikation mit p für Nachpunktanteil. Nach Theorem 13 ist

$$r = \left[\underbrace{(-1)^j \sum_{k=-\infty}^{l} (\gamma_k p^{k-l-1})}_{a}\right] \cdot p^{l+1} \tag{3.2}$$

wobei $\frac{1}{p} \leq |a| < 1$, da $\gamma_l \neq 0$.

Definition 16 *Die Darstellung $r = a \cdot p^b$ heisst normalisierte Gleitpunktdarstellung von r bezüglich p. a heisst Mantisse, b der Exponent von r.*

$$\begin{aligned} a &= V_M \sum_{i=1}^{\infty} \alpha_i p^{-i} \qquad \textit{mit } \alpha_1 \neq 0 \\ b &= V_E \sum_{i=1}^{t} \beta_i p^{t-i}, \quad V_M, V_E \textit{ Vorzeichen }. \end{aligned}$$

Auf einem Rechner ist für die Darstellung (im allgemeinen, in normierter Sprache) nur eine **feste** Anzahl von n Stellen möglich.

Diese n Stellen werden aufgeteilt in l Stellen für die Mantisse und m Stellen für den Exponenten. Die Zahl

$$V_M(\alpha_1 p^{-1} + \alpha_2 p^{-2} + \cdots + \alpha_l p^{-l})p^{V_E(\beta_1 p^{m-1}+\cdots+\beta_m p^0)} \tag{3.3}$$

mit $\alpha_1 \neq 0$ wird durch die Angabe $V_M \alpha_1 \alpha_2 \cdots \alpha_l V_E \beta_1 \cdots \beta_m$ codiert.

Beispiel 17 *Betrachte*

$$\begin{aligned} r_1 &= -1234.567890,\ r_2 = 0.01234567890 \\ p &= 10,\ l = 9,\ m = 2. \end{aligned}$$

Die normalisierte Gleitpunktdarstellung ist

$$\begin{aligned} r_1 &= -0.123456789 * 10^4 \\ &= -123456789 + 04, \\ r_2 &= -0.123456789 * 10^{-1} \\ &= -123456789 - 01. \end{aligned}$$

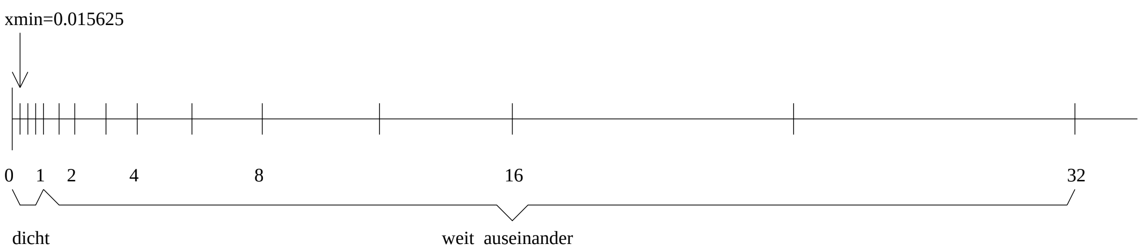
Definition 18 $\mathbb{M}(p, l, m)$ *bezeichnet die Menge der in normalisierter Gleitpunktdarstellung codierbaren (darstellbaren) Zahlen, auch Maschinenzahlen genannt, mit Basis p, mit l Stellen für die Mantisse und mit m Stellen für den Exponenten (jeweils zur Basis p).*

Beispiel 19 $\mathbb{M}(2, 3, 3)$, *d.h.* $p = 2$ *und* $l = m = 3$

verfügbare Mantissen	verfügbare Exponenten
	$\pm000 \;\widehat{=}\; \pm0 \quad 1,\ 1$
$\pm100 \;\widehat{=}\; \pm\frac{1}{2}$	$\pm001 \;\widehat{=}\; \pm1 \quad 2,\ \frac{1}{2}$
	$\pm010 \;\widehat{=}\; \pm2 \quad 4,\ \frac{1}{4}$
$\pm101 \;\widehat{=}\; \pm\frac{5}{8}$	$\pm011 \;\widehat{=}\; \pm3 \quad 8,\ \frac{1}{8}$
	$\pm100 \;\widehat{=}\; \pm4 \quad 16,\ \frac{1}{16}$
$\pm110 \;\widehat{=}\; \pm\frac{3}{4}$	$\pm101 \;\widehat{=}\; \pm5 \quad 32,\ \frac{1}{32}$
	$\pm110 \;\widehat{=}\; \pm6 \quad 64,\ \frac{1}{64}$
$\pm111 \;\widehat{=}\; \pm\frac{7}{8}$	$\pm111 \;\widehat{=}\; \pm7 \quad 128,\ \frac{1}{128}$

kleinste positive Zahl $\quad x_{\min} = \frac{1}{2}2^{-7} = 2^{-8}$
größte positive Zahl $\quad x_{\max} = \frac{7}{8}2^{7} = 112$

Maschinenzahlen sind wie folgt angeordnet



3.2 Rundungsfehler (Reduktionsfehler)

Von nun an sei p entweder 10 oder 2^k. Um eine beliebige reelle Zahl darzustellen wird sie gerundet in die Menge der Maschinenzahlen abgebildet,

$$\gamma : \mathbb{R} \longrightarrow \mathbb{M}(p,l,m).$$

Sei $x = \pm\left(\sum_{i=1}^{\infty} \alpha_i p^{-i}\right) \cdot p^t, \alpha_1 \neq 0, x \in Z = [x_{\min}, x_{\max}] \cup [-x_{\max}, -x_{\min}]$

Übliche Rundung auf l Stellen

$$\gamma(x) = \begin{cases} \pm & \left(\sum_{i=1}^{l} \alpha_i p^{-i}\right) \cdot p^t & \text{falls } \alpha_{l+1} < \frac{p}{2} \\ \pm & \left(\sum_{i=1}^{l} (\alpha_i p^{-i}) + p^{-l}\right) \cdot p^t & \text{falls } \alpha_{l+1} \geq \frac{p}{2}. \end{cases} \tag{3.4}$$

Bei Zahlen ausserhalb von Z gibt es **rechnerabhängige** Vorgehensweisen z.B.

$$|x| \;<\; x_{\min} \left\langle \begin{array}{lll} \gamma(x) & = & 0 \\ & & \\ \gamma(x) & = & \text{sign }(x) x_{\min} \end{array} \right. \quad \text{'underflow'}$$

$$|x| \;>\; x_{\max} \left\langle \begin{array}{lll} \gamma(x) & = & \text{sign }(x) x_{\max} \text{ mit Meldung 'overflow'} \\ & & \text{Warnung 'overflow'} \end{array} \right.$$

Proposition 20 Absoluter Rundungsfehler *Sei $x \in Z$. Dann gilt*

$$|\gamma(x) - x| \leq \frac{p}{2}^{-l} p^t. \tag{3.5}$$

Beweis.
Abrunden:

$$|\gamma(x) - x| \;=\; \left| \left(\sum_{i=1}^{\infty} \alpha_i p^{-i}\right) p^t - \left(\sum_{i=1}^{l} \alpha_i p^{-i}\right) p^t \right|$$

$$
\begin{aligned}
&= p^t \cdot \left| \sum_{i=l+1}^{\infty} \alpha_i p^{-i} \right| = p^t p^{-(l+1)} \left| \underbrace{\sum_{i=0}^{\infty} \alpha_{l+i+1} p^{-i}}_{\leq \frac{p}{2}} \right| \\
&\leq p^t \frac{p^{-l}}{2}.
\end{aligned}
$$

Aufrunden:

$$
\begin{aligned}
|\gamma(x) - x| &= \left| \left(\sum_{i=1}^{\infty} \alpha_i p^{-i} \right) p^t - \left(\sum_{i=1}^{l} (\alpha_i p^{-i}) + p^{-l} \right) p^t \right| \\
&= \left| \sum_{i=l+1}^{\infty} \alpha_i p^{-i} - p^{-l} \right| p^t \\
&= p^t \left| -p^{-l} + \alpha_{l+1} p^{-(l+1)} + \cdots \right| \\
&= p^t p^{-l} \left| -1 + \alpha_{l+1} p^{-1} + \alpha_{l+2} p^{-2} + \cdots \right| \\
&\leq p^t p^{-l} \left(-\frac{p}{2} p^{-1} + 1 \right) = p^t \frac{p^{-l}}{2}.
\end{aligned}
$$

□

Proposition 21 Relativer Rundungsfehler *Sei $x \in Z$. Dann gilt*

$$
\frac{|\gamma(x) - x|}{|x|} \leq \frac{\frac{p^{-l}}{2} p^t}{M_x p^t} \leq \frac{\frac{p^{-l}}{2}}{\frac{1}{p}} = \frac{p}{2} p^{-l}.
$$

Beweis. Der Beweis folgt da die Mantisse M_x von x die Ungleichung $M_x \geq \frac{1}{p}$ erfüllt. □

Beachte, bei $M_x = \frac{1}{p}$, d.h. $x = \frac{p^t}{p}$ gibt es keinen Fehler.
Die Zahl

$$
\text{eps} := \frac{p}{2} p^{-l} =: (\text{macheps}) \; =: \mathbf{u} \tag{3.6}
$$

heisst Maschinengenauigkeit (roundoff unit).

Bemerkung 22 *Es gilt immer* eps $= \min\{|\delta||\ \delta \in [x_{\min}, x_{\max}]$ *und* $|\gamma(1+\delta)| > 1\}$ *und*

$$\begin{aligned} \gamma(x) &= x \cdot (1+\varepsilon) \ \textit{mit}\ |\varepsilon| \leq \text{eps}\,, \\ \textit{wobei}\ \varepsilon &= \tfrac{\gamma(x)-x}{x}. \end{aligned}$$

Beispiel 23 *Betrachte* 0.2 *dezimal mit* $l = 6, p = 2$. *Das ergibt* $0.\overline{0011}$ *in dualer Darstellung und normalisiert* $0.11\overline{0011}*2^{-2}$. *Rundung auf* $l = 6$ *Stellen ergibt* $0.110011 * 2^{-2}$. *Rückkonvertiert ergibt sich* $\frac{51}{256} = 0.19921875$.

Rundungsfehler entstehen beim Einlesen, Speichern, Konvertieren, Rechnen.

Man hat auf dem Rechner nur eine *Pseudoarithmetik*, denn $\mathbb{M}$ ist nicht abgeschlossen bzgl. $+, -, *, \div$.

Beispiel 24 *Betrachte* $\mathbb{M}(10, 5, 1)$ *und die Zahlen*

$$\begin{aligned} x &= +25684 + 1 \ \widehat{=}\ 2.5684, \\ y &= +32791 - 2 \ \widehat{=}\ 0.0032791. \end{aligned}$$

Dann gilt

$$\begin{aligned} x + y &= \underbrace{2.5716}_{5}791 \notin \mathbb{M}, x * y = 0.00\underbrace{84220}_{5}4044 \notin \mathbb{M} \\ x/y &= \underbrace{783.26}_{5}37004 \notin \mathbb{M}. \end{aligned}$$

Die übliche Arithmetik in $\mathbb{R}$ muss so realisiert werden, dass das Ergebnis wieder in $\mathbb{M}$ ist.

Das Ergebnis der Pseudooperation $\bigcirc\!\!\!\!\nabla$, $\nabla \in \{+, -, *, \div\}$ ist im allgemeinen festgelegt durch

$$gl(x \nabla y) \equiv x \bigcirc\!\!\!\!\nabla\, y = \gamma(x \nabla y),\ \text{für}\ x, y \in \mathbb{M} \tag{3.7}$$

Hardwaremäßig wird üblicherweise mit längerer Mantisse gearbeitet als im Ergebnis, dies normalisiert und dann gerundet. **Vorsicht: dies gilt nicht auf allen Rechnern, man kann böse Überraschungen erleben. (z.B. auf Rechnern die nicht mit IEEE–Standard–Arithmetik arbeiten.)**

Proposition 25 *Es gelte (3.7). Falls $|x\nabla y|$ in Z liegt, so gilt für den relativen Fehler*

$$\left|\frac{x \circledtriangledown y - x\nabla y}{x\nabla y}\right| = \left|\frac{\gamma(x\nabla y) - x\nabla y}{x\nabla y}\right| < \text{ eps} \tag{3.8}$$

In $\mathbb{R}$ gelten Assoziativ–, Kommutativ– und Distributivgesetz, in der Rechnerarithmetik in $\mathbb{M}$ gilt im allgemeinen **nur** die Kommutativität für Addition und Multiplikation.

Beispiel 26 *Betrachte* $\mathbb{M}(10,5,1)$.

Assoziativität

$$\begin{aligned}
0.98765 + 0.012424 - 0.0065432 &= 0.\underline{99353}08 \\
0.98765 \oplus (0.012424 \ominus 0.0065432) &= \\
0.98765 \oplus 0.00588(08) &= 0.\underline{99353}(08) \\
(0.98765 \oplus 0.012424) \ominus 0.0065432 &= \\
\underbrace{1.000074}_{1.0001} \ominus 0.0065432 &= 0.\underline{99356}
\end{aligned}$$

Distributivität

$$\begin{aligned}
(4.2832 - 4.2821) * 5.7632 &= 0.00\underline{633952} \\
(4.2832 \ominus 4.2821) \otimes 5.7632 &= \\
0.0011 \otimes 5.7632 &= 0.00\underline{63395}(2) \\
(4.2832 \otimes 5.7632) \ominus (4.2821 \otimes 5.7632) &= \\
\underbrace{24.684(93824) \quad \ominus 24.678(59872)}_{24.685 \ominus 24.679 = 0.006\underline{0000}} &
\end{aligned}$$

Mathematisch äquivalente Algorithmen zur Auswertung eines rationalen Ausdrucks können bei Durchführung auf dem Rechner zu <u>wesentlich</u> verschiedenen Ergebnissen führen, selbst wenn die Eingabedaten Maschinenzahlen sind. Nichtrationale Operationen wie $\sqrt{\ }, \sin, \exp$ **sind (nicht immer gut) softwaremäßig realsiert. Auch**

hierfür gilt im allgemeinen: Das Maschinenergebnis ist Rundung des exakten Ergebnisses.

Auf einigen Maschinen ist selbst $\div$ nicht fest realisiert oder man hat die Wahl zwischen billig und schlecht oder teuer und gut.

3.3 Fehlerfortpflanzung

Unglücklicherweise planzen sich einmal gemachte Fehler (z.B. bei Rundung) auch noch fort. Seien x, y mit Fehlern $\triangle x, \triangle y$ behaftet $\left(\left|\frac{\triangle x}{x}\right|, \left|\frac{\triangle y}{y}\right| \ll 1\right)$.

Was passiert bei exakter Durchführung einer elementaren Operation mit diesen Fehlern, d.h. für $\nabla \in \{+, -, *, \div\}$ sei $x\nabla y \in \mathbb{M}$.

Fortpflanzung des relativen Fehlers:

$$(x + \triangle x)\nabla(y + \triangle y) = x\nabla y + \underbrace{\triangle(x\nabla y)}_{\text{abs. Fehler des Ergebnisses}}$$

$\pm : x \pm y + \triangle x \pm \triangle y :$

$$\frac{\triangle(x \pm y)}{x \pm y} = \frac{\triangle x \pm \triangle y}{x \pm y} = \frac{x}{x \pm y}\frac{\triangle x}{x} \pm \frac{y}{x \pm y}\frac{\triangle y}{y} \tag{3.9}$$

Bemerkung 27 *Ist $|x \pm y|$ sehr klein gegenüber $|x|$ oder $|y|$, so kann der relative Fehler ausserordentlich verstärkt werden, z.B. wenn zwei annähernd gleich große Zahlen subtrahiert werden. Dieser Effekt heisst Auslöschung. Man muss bei der Aufstellung der Algorithmen darauf achten, dass dies soweit wie möglich vermieden wird.*

Bei Multiplikation und Division tritt keine wesentliche Verstärkung des Fehlers auf.

$$* : xy + \underbrace{x\triangle y + y\triangle x + \triangle y\triangle x}_{\triangle(x*y)} :$$

$$\frac{\triangle(x * y)}{x * y} = \frac{\triangle y}{y} + \frac{\triangle x}{x} + \frac{\triangle y\triangle x}{xy} \approx \frac{\triangle y}{y} + \frac{\triangle x}{x} \tag{3.10}$$

$$\div : \frac{x+\triangle x}{y+\triangle y} = \frac{x}{y} + \frac{y\triangle x - x\triangle y}{y(y+\triangle y)} :$$

$$\begin{aligned}
\frac{\triangle(x \div y)}{x \div y} &= \frac{1}{x \div y}\left(\frac{y\triangle x - x\triangle y}{y(y+\triangle y)}\right) \qquad (3.11)\\
&= \frac{y\triangle x - x\triangle y}{x(y+\triangle y)} = \frac{\triangle x}{x}\frac{y}{y+\triangle y} - \frac{\triangle y}{y+\triangle y}\\
&\approx \frac{\triangle x}{x} - \frac{\triangle y}{y}
\end{aligned}$$

Wie beurteilt man nun numerische Verfahren?

Wesentliche Gesichtspunkte sind die **Genauigkeit und Verlässlichkeit** des Ergebnisses und die **Effizienz**, d.h. der Rechenzeitbedarf, der Speicherbedarf, die Programmlänge, die Kosten und neuerdings die Vektorisierbarkeit oder Parallelisierbarkeit.

Die Effizienz hängt stark von der Rechnerarchitektur ab, bzw. von der Systemumgebung, i.a. kann man dieses gut abschätzen oder testen.

Die Verlässlichkeit gibt an, wie ein Versagen eines Teilprogramms den Gesamtprozess beeinflusst. Bei der Genauigkeit hat man **alle möglichen auftretenden** Fehler zu berücksichtigen und ihren Einfluss auf das Ergebnis.

Fehler sind hier <u>nicht</u> Programmier– oder Hardwarefehler, sondern die Abweichung, eines nach richtig angewendeter Vorschrift erzeugten Resultats, vom gewünschten Ergebnis.

Es gibt verschiedene Fehlerarten:

1. **Rundungs– oder Reduktionsfehler**, s.o.

2. **Datenfehler:** Üblicherweise sind die Eingangsdaten ungenau, z. B. Messdaten mit Messfehlern. Das Lösen eines Problems entspricht dem Auswerten einer Funktion $f : D \longrightarrow W$ für Datenwerte $x \in D$. Anstelle von x hat man gestörten Wert $\tilde{x}$. Der <u>Datenfehler</u> ist dann $f(x) - f(\tilde{x})$.

Die Empfindlichkeit der Lösung des Problems (Auswertung von f), gegenüber kleinen Änderungen in den Daten (in x) heisst **Kondition des Problems.** Die gute oder schlechte Kondition ist eine **Eigenschaft des Problems** und **NICHT** des Verfahrens.

Eine instabile Differentialgleichung hat eine schlechte Kondition.

3. **Verfahrensfehler:** Exakte Verfahren enden nach endlich vielen Operationen (bei exakter Rechnung) mit dem exakten Ergebnis $y = f(x)$. Näherungsverfahren (z.B. das Euler-Verfahren) enden mit einer Näherung $\tilde{y}$ für die Lösung y. Verfahrensfehler: $\tilde{y} - y$

Anwender- problem	$\longrightarrow$	Mathem. Problem	$\longrightarrow$	Numer. Algorithm.	$\longrightarrow$	Prog.
Daten x $\downarrow$		Unsich. Daten $\tilde{x}$ $\downarrow$		Daten $\tilde{x}$ $\downarrow$		gerund. Daten $gl(\tilde{x})$ $\downarrow$
Lösung des Probl. y	$\longrightarrow$ $\uparrow$	exaktes Ergebn. y^*	$\longrightarrow$ $\uparrow$	Ergeb. des Alg. $\hat{y}$	$\longrightarrow$ $\uparrow$	Maschinen- Ergeb. $\tilde{y}$
	Daten- fehler		Verfahr.- fehler		Rund. fehler	

3.4 Fehleranalyse

Um herauszufinden was in einem Algorithmus passiert, machen wir eine Fehleranalyse.

Gegeben sei eine Funktion $f(a_1, \ldots, a_n)$ und eine durch Fehler gestörte Funktion $\tilde{f}(a_1, \ldots, a_n)$.

Eine *Fehleranalyse* ist die Verfolgung der Auswirkung aller Fehler, die in den einzelnen Schritten bei der Berechnung von f auftreten können.

Bei der *Vorwärtsanalyse* verfolgen wir den Fehler von Schritt zu Schritt und schätzen für jedes Teilergebnis den akkumulierten Fehler ab, d.h. wir verfolgen den Fehler, so dass das Ergebnis die Form $\tilde{f}(a_1, \ldots, a_n) = f(a_1, \ldots, a_n) + \delta$ hat, wobei δ der akkumulierte Fehler ist.

Bei der *Rückwärtsanalyse* geschieht die Verfolgung des Fehlers so, dass jedes Zwischenergebnis als *exakt berechnetes Ergebnis für gestörte Daten* interpretiert wird, d.h. der akkumulierte Fehler im Teilergebnis wird als Datenfehlereffekt interpretiert. Also: $\tilde{f}(a_1, \ldots, a_n) = f(\tilde{a}_1, \ldots, \tilde{a}_n)$ mit gestörten Daten $\tilde{a}_1, \ldots, \tilde{a}_n$. Über den Fehler im Endergebnis erhält man so zunächst nur die Aussage, dass er einem Datenfehler bestimmter Größe entspricht, dem *äquivalenten Datenfehler*. Die Größe der äquivalenten Datenfehler dient dann zur Beurteilung der Algorithmen.

Beispiel 28 *Betrachte*

$$f(a_1, a_2) = a_1 + a_2,\ \tilde{f}(a_1, a_2) = a_1 \oplus a_2.$$

$$\begin{array}{llll} a_1 \oplus a_2 & \overset{\text{Vorwärtsanalyse}}{=} & \underbrace{a_1 + a_2}_{f(a_1,a_2)} + \underbrace{\varepsilon(a_1 + a_2)}_{\delta} & |\varepsilon| \leqslant \text{eps}\,, \\ & & & |\delta| \leqslant \text{eps}\,(|a_1| + |a_2|) \\ & \overset{\text{Rückwärtsanalyse}}{=} & \underbrace{a_1(1+\varepsilon) + a_2(1+\varepsilon)}_{f(\tilde{a}_1,\tilde{a_2})} & |\varepsilon| \leqslant \text{eps} \end{array}$$

Eine Vorwärtsanalyse ist im allgemeinen kaum durchführbar, für die meisten Verfahren ist, wenn überhaupt, nur eine Rückwärtsanalyse bekannt.

Ein idealer Algorithmus im Sinne der Rückwärtsanalyse hat einen äquivalenten Datenfehler von der Größenordnung der Rundungsfehler oder Eingangsfehler. Aussagen über den Gesamtfehler erhält man dann über *Störungssätze.*

Vorteil dieser Vorgehensweise: Auswirkungen der Arithmetik und des Verfahrens werden getrennt von der Kondition des Problems.

Beispiel 29 *Betrachte die quadratische Gleichung* $x^2 - 2a_1x + a_2 = 0 \quad 0 < a_2 \ll 1 < a_1$.

Finde die kleine Lösung

$$x^* = f(a_1, a_2) = a_1 - \sqrt{a_1^2 - a_2}$$

<u>*Kondition des Problems:*</u> *Setze* $a = (a_1, a_2)$ *und betrachte Taylorentwicklung von*

$$f(a + \triangle a) = \tilde{x}^*$$

$$\begin{aligned} f(a+\triangle a) &= f(a) + \tfrac{\partial f}{\partial a_1}(a) \triangle a_1 + \tfrac{\partial f}{\partial a_2}(a) \triangle a_2 \\ &+ \text{ Terme höherer Ordnung in } \triangle a_1, \triangle a_2. \end{aligned} \tag{3.12}$$

$$\begin{aligned}
\frac{\tilde{x}^* - x^*}{x^*} &= \underbrace{\frac{\partial f(a)}{\partial a_1}\frac{a_1}{x^*}}\frac{\triangle a_1}{a_1} + \underbrace{\frac{\partial f(a)}{\partial a_2}\frac{a_2}{x^*}}\frac{\triangle a_2}{a_2} + \ T.h.0. \\
&\qquad \textit{Verstärkungsfaktoren für} \\
&\qquad \textit{relativen Fehler in Daten.} \\
\frac{\partial f}{\partial a_1}(a) &= 1 - \frac{1}{2}\frac{2a_1}{\sqrt{a_1^2 - a_2}} = \frac{\sqrt{a_1^2 - a_2} - a_1}{\sqrt{a_1^2 - a_2}} = \frac{-x^*}{\sqrt{a_1^2 - a_2}} \\
\frac{\partial f}{\partial a_2}(a) &= \frac{1}{2}\frac{1}{\sqrt{a_1^2 - a_2}} \\
\frac{\partial f}{\partial a_1}(a)\frac{a_1}{x^*} &= \frac{-x^*}{\sqrt{a_1^2 - a_2}}\frac{a_1}{x^*} \approx -1 \quad \text{da } a_2 \ll 1 < a_1 \\
\frac{\partial f}{\partial_{a_2}}(a)\frac{a_2}{x^*} &= \frac{a_2}{2\sqrt{a_1^2 - a_2}x^*} = \frac{a_2}{\frac{2a_2}{\frac{a_1}{\sqrt{a_1^2 - a_2}} + 1}} \approx 1
\end{aligned}$$

Also ist das Problem gut konditioniert. Erwarten würden wir bei einem guten Algorithmus einen relativen Fehler von $c \cdot \text{eps}$, *mit kleinem* c.

Algorithmus:	$\mathbb{M}(10,5,1) a_1 = 6.002, a_2 = 0.01$
$y_1 := a_1 * a_1$	$y_1 = 36.024$
$y_2 := y_1 - a_2$	$y_2 = 36.014$
$y_3 := \sqrt{y_2}$	$y_3 = 6.0012$
$x := y_4 = a_1 - y_3$	$\tilde{x} = y_4 = 0.00080000$
	exakt 0.00083311
	$\frac{\triangle x}{x} = 0.039747$

Eine Rückwärtsanalyse liefert

$$\begin{aligned}
& \left[a_1 - \sqrt{(a_1^2(1+\varepsilon_1) - a_2)(1+\varepsilon_2)}(1+\varepsilon_3)\right](1+\varepsilon_4),\ |\varepsilon_i| < \text{ eps}. \\
= \quad & a_1(1+\varepsilon_4) - \sqrt{a_1^2(1+\varepsilon_4)^2 \underbrace{(1+\varepsilon_1)(1+\varepsilon_2)(1+\varepsilon_3)^2}_{\substack{1+\varepsilon_5 \\ |\varepsilon_5|<4\text{ eps}.}} - a_2 \underbrace{(1+\varepsilon_2)(1+\varepsilon_3)^2(1+\varepsilon_4)^2}_{\substack{1+\varepsilon_6 \\ |\varepsilon_6|<5\text{ eps}.}}} \\
= \quad & a_1(1+\varepsilon_4) - \sqrt{(a_1(1+\varepsilon_4))^2 - a_2 \underbrace{((1+\varepsilon_6) - \frac{a_1^2(1+\varepsilon_4)^2\varepsilon_5}{a_2})}_{\substack{1+\varepsilon_7 \\ |\varepsilon_7|<\text{ eps}\,(5+4\frac{a_1^2}{|a_2|}))}}}
\end{aligned}$$

Der Rechner liefert die exakte Lösung von $x^2 - 2a_1(1+\varepsilon_4)x + a_2(1+\varepsilon_7) = 0$ *mit* $|\varepsilon_4| < \text{ eps}$ *und* $|\varepsilon_7| < \text{ eps}\,(5 + 4\underbrace{\frac{a_1^2}{|a_2|}}_{\textit{groß}}))$*, d.h. der Algorithmus ist* **ungeeignet.**

Definition 30 *Ein Algorithmus für den die Rückwärtsanalyse liefert, dass das berechnete Ergebnis das exakte Ergebnis eines gestörten Problems mit Störungen der Größenordnung* $c \cdot \text{ eps}$ *ist, heisst* numerisch rückwärts stabil. *Ein Algorithmus für den die Vorwärtsanalyse liefert, dass das berechnete Resultat einen relativen Fehler der Größenordnung* $c \cdot \text{ eps}$ *hat, heisst* numerisch vorwärts stabil. *Ansonsten ist der Algorithmus* numerisch instabil.

Faustregel:

Gut konditioniertes Problem und stabiler Algorithmus $\Longrightarrow$ gute Ergebnisse.

Schlecht konditioniertes Problem oder instabiler Algorithmus $\Longrightarrow$ unsichere Ergebnisse.

3.5 Fehleranalyse bei Einschrittverfahren

Wir wollen nun mit diesen Erkenntnissen die expliziten Einschrittverfahren untersuchen.

Wir haben schon im Beispiel gesehen, dass der Fehler wächst, wenn wir h zu klein machen. Woran liegt das?

Die Lipschitzkonstante L, gibt im wesentlich an ob das Problem schlecht konditioniert ist. Leider kennen wir diese im allgemeinen nicht.

Was passiert bei der Implementierung des Verfahrens?

Wir berechnen

$$u_{i+1} = u_i + h\Phi(t_i, u_i, h, f), \;\; u_0 = y_0.$$

Dabei machen wir einen Approximationsfehler aber auch Rundungsfehler bei der Auswertung von Φ und bei der Summe und Multiplikation.

Auf dem Rechner erhalten wir

$$\begin{aligned} \tilde{u}_0 &= gl(y_0) = y_0 + \epsilon_0, \;\; |\epsilon_0| \leq \text{ eps} \\ \tilde{u}_{i+1} &= gl(\tilde{u}_i + gl(\tilde{h} * gl(\Phi(\tilde{t}_i, \tilde{u}_i, \tilde{h}, \tilde{f})) \\ &= \tilde{u}_i + h\Phi(t_i, \tilde{u}_i, h, f) + \epsilon_{i+1}, \end{aligned} \tag{3.13}$$

wobei

$$\epsilon_{i+1} = h\Phi(t_i, \tilde{u}_i, h, f)(\alpha_{i+1} + \mu_{i+1}) + \tilde{u}_{i+1}\sigma_{i+1} + \mathcal{O}(\text{ eps}^2). \tag{3.14}$$

Dabei ist α_{i+1} der Fehler bei der Berechnung von Φ, μ_{i+1} der Fehler bei der Berechnung von $\tilde{h}\tilde{\Phi}$ und σ_{i+1} der Fehler bei der Berechnung von $\tilde{u}_i + \widetilde{\tilde{h}\tilde{\Phi}}$.

Wenn h klein ist, können wir annehmen, dass

$$h(\alpha_{i+1} + \mu_{i+1}) \ll u_{i+1}\sigma_{i+1}.$$

Damit haben wir grob, dass $\epsilon_{i+1} \approx \sigma_{i+1}\tilde{u}_{i+1}$.

Theorem 31 Rundungsfehlereinfluß beim expliziten Einzelschrittverfahren.

Seien $u_i, \; i = 0, \ldots, N$ die (theoretischen) Verfahrenswerte, die gemäß

$$u_{i+1} = u_i + h\Phi(t_i, u_i, h, f), \quad u_0 = y_0$$

gewonnen werden, und seien $\tilde{u}_i$ die vom Rechner hierfür gelieferten Werte, für die gilt

$$\tilde{u}_{i+1} = \tilde{u}_i + h\Phi(t_i, \tilde{u}_i, h, f) + \epsilon_{i+1}, \quad \tilde{u}_0 = u_0 + \epsilon_0.$$

Die Inkrementfunktion Φ genüge einer Lipschitzbedingung mit Konstante L bzgl. u und es sei $|\epsilon_i| \leq \epsilon = c\,\mathrm{eps}$ für alle $0 = 1, \ldots, N$.

Dann gilt für $r_i = \tilde{u}_i - u_i$, dass

$$|r_m| \leq e^{L(t_m - t_0)}(|r_0| + \frac{t_m - t_0}{h}\epsilon).$$

Beweis. Wir haben

$$r_{i+1} = r_i + h[\Phi(t_i, \tilde{u}_i, h, f) - \Phi(t_i, u_i, h, f] + \epsilon_{i+1}$$

also

$$|r_{i+1}| \leq (1 + hL)|r_i| + \epsilon.$$

Nach Lemma 10 gilt daher

$$|r_m| \leq (|r_0| + m\epsilon)e^{mLh} = (|r_0| + \frac{t_m - t_0}{h}\epsilon)e^{L(t_m - t_0)},$$

da $mh = t_m - t_0$. □

Damit können wir nun den Gesamtfehler $E_i := \tilde{u}_i - y(t_i)$ bei expliziten Einzelschrittverfahren charakterisieren.

Theorem 32 Gesamtfehler.

Mit den Bezeichnungen und Voraussetzungen von Theorem 31 und mit $\tau_h = \max_i |\tau_i|$ und $\tau_i = \tau(t_i, y_i, h, f)$ folgt für $m = 0, \ldots, N$, dass

$$|E_m| = |\tilde{u}_m - y(t_m)| \leq e^{L(t_m - t_0)}[|\epsilon_0| + \frac{(t_m - t_0)}{h}(\tau_h + c\,\mathrm{eps})].$$

Beweis. Wir haben $E_0 = \epsilon_0$ und

$$E_{i+1} = \tilde{u}_{i+1} - y_{i+1} = (\tilde{u}_{i+1} - u_{i+1}) + (u_{i+1} - y_{i+1}) = r_{i+1} + e_{i+1}.$$

Wir haben e_{i+1} in Theorem 9 und r_{i+1} in Theorem 31 berechnet. Damit ergibt sich

$$\begin{aligned} |E_m| &= |r_m + e_m| \le |r_m| + |e_m| \\ &\le \left(|r_0| + |e_0| + (t_m - t_0) \frac{\tau_h + c\,\mathrm{eps}}{h} \right) e^{L(t_m - t_0)}. \end{aligned}$$

□

Bemerkung 33 *Diskussion der Ergebnisse:*

- **Falls h groß ist, so ist $\frac{\tau_h}{h}$ groß und $\frac{\mathrm{eps}}{h}$ klein.**
- **Falls h klein ist, so ist $\frac{\tau_h}{h}$ klein, aber $\frac{\mathrm{eps}}{h}$ groß.**
- **Wir müssen h klein machen, damit $\frac{\tau_h}{h}$ klein ist, wir dürfen es aber nicht zu klein machen, weil sonst der Anteil des Fehlers, der von den Rundungsfehlern herrührt den Fehler dominiert.**
- **Wir brauchen daher Verfahren höherer Ordnung, d.h., Verfahren, bei denen $\frac{\tau_h}{h} = \mathcal{O}(h^p)$ mit möglichst großem p, denn dann können wir den Gesamtfehler schon mit relativ großem h klein machen.**

Kapitel 4

Interpolation

4.1 Einführung

Wir haben gesehen, dass wir für die numerische Lösung von Differentialgleichungen Verfahren höherer Ordnung benötigen. Um solche Verfahren zu erhalten, und auch aus vielen anderen Gründen wie

- Approximation von Funktionen,
- Darstellung der Lösung von Differentialgleichungen als Kurve,
- CAD (Computer Aided Design),
  CAGD (Computer Aided Graphic Design),
- Bildverarbeitung,
- Signalverarbeitung,
- ...

müssen wir uns mit Interpolationsmethoden beschäftigen.

Betrachte einen Typ von Funktionen z.B.

1. Polynome
$$P(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n; \tag{4.1}$$
2. Rationale Funktionen
$$P(x) = \frac{a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n}{a_{n+1} + a_{n+2} x + a_{n+3} x^{n+3} + \ldots + a_{n+m+1} x^m}; \tag{4.2}$$

3. Trigonometrische Polynome

$$P(x) = a_0 + a_1 e^{xi} + a_2 e^{2xi} + \ldots + a_n e^{nxi}; \tag{4.3}$$

4. oder Splines, dies sind stückweise Polynome.

Eine Interpolationsaufgabe hat dann die folgende Form:

Gegeben Werte (x_i, f_i) für $i = 0, 1, \ldots, n$ (diese heißen *Stützstellen* oder *Knoten*), finde Parameter $a_0, \ldots, a_n$, so dass

$$P(x_i, a_0, \ldots, a_n) = f_i, \qquad i = 0, \ldots, n. \tag{4.4}$$

Beispiel 34 *Interpolation durch stückweise lineare Funktionen. Gegeben seien die Stützstellen, die das Euler-Verfahren für $y' = f(t, y)$, mit Anfangswert $y(t_0) = y_0$ berechnet hat,*

$$(t_i, u_i), \; i = 0, \ldots N.$$

Bestimme eine stückweise lineare Funktion (linearer Spline), die diese Punkte verbindet.

4.2 Polynominterpolation

Eine der wichtigsten Interpolationsaufgaben in der Praxis ist die Polynominterpolation.

Mit der Bezeichnung

$$\Pi_n := \{\text{Polynome vom Grad} \leq n\}$$

erhalten wir den folgenden Satz:

Theorem 35 Lagrange–Interpolation[1]
Seien $n+1$ Stützstellen $(x_i, f_i), i = 0, \ldots, n$, mit $x_i \neq x_j$ für $i \neq j$ gegeben. Dann gibt es ein eindeutig bestimmtes Polynom $p \in \Pi_n$ mit

$$p(x_i) = f_i, \qquad i = 0, \ldots, n. \tag{4.5}$$

[1] J.L. Lagrange, Französischer Mathematiker 1736–1813

Beweis. Zuerst zeigen wir die Eindeutigkeit des Interpolationspolynoms: Angenommen, es existieren zwei Polynome $p_1, p_2 \in \Pi_n$ mit

$$p_1(x_i) = p_2(x_i) = f_i, \qquad i = 0, \dots, n.$$

Dann definiere $p := p_1(x) - p_2(x)$. Es gilt: Grad $p \leq n$ und

$$p(x_i) = p_1(x_i) - p_2(x_i) = 0, \qquad i = 0, \dots, n$$

p hat $n+1$ Nullstellen. Mit dem Hauptsatz der Algebra folgt:

$$p(x) \equiv 0$$

und damit

$$p_1 = p_2.$$

Nun zeigen wir die Existenz, indem wir das Interpolationspolynom konstruieren: Sei

$$L_i(x) = \prod_{\substack{j=0 \\ j \neq i}}^{n} \frac{x - x_j}{x_i - x_j}. \tag{4.6}$$

Für die L_i gilt an den Stützstellen x_k:

$$L_i(x_k) = \begin{cases} 1 & \text{für } i = k \\ 0 & \text{für } i \neq k \end{cases}. \tag{4.7}$$

Daraus folgt, dass

$$\begin{aligned} p(x) &= \sum_{i=0}^{n} f_i L_i(x) \\ &= \sum_{i=0}^{n} f_i \left(\prod_{\substack{j=0 \\ j \neq i}}^{n} \frac{x - x_j}{x_i - x_j} \right) \end{aligned} \tag{4.8}$$

die Interpolationsaufgabe löst. □

Die $L_i(x)$ heißen *Lagrange–Interpolationspolynome*. Sie bilden eine Basis für Π_n. Eine andere Basis ist $1, x, x^2, \dots, x^n$. Beide Basen sind jedoch für die praktische Berechnung ungeeignet. Für die numerische Berechnung verwenden wir zwei andere Ansätze.

4.2.1 Das Verfahren von Neville und Aitken

Definition 36 *Gegeben seien die Stützwerte* $(x_i, f_i), i = 0, \ldots, n$ *und eine Teilmenge* $\{i_0, \ldots, i_k\} \subset \{0, \ldots, n\}$. *Dann bezeichnet* $P_{i_0,\ldots,i_k} \in \Pi_k$ *das Polynom mit der Eigenschaft*

$$P_{i_0,\ldots,i_k}(x_{i_j}) = f_{i_j}, \qquad j = 0, \ldots, k \tag{4.9}$$

Dann erhalten wir für diese Teilpolynome die folgende Rekursionsformel:

Lemma 37 *Es gilt:*

$$\begin{aligned} P_{i_j}(x) &\equiv f_{i_j}, \qquad j = 0, \ldots, k, & (4.10) \\ P_{i_0,\ldots,i_k}(x) &= \frac{(x - x_{i_0})P_{i_1,\ldots,i_k}(x) - (x - x_{i_k})P_{i_0,\ldots,i_{k-1}}(x)}{x_{i_k} - x_{i_0}}. & (4.11) \end{aligned}$$

Beweis. Die Formel (4.10) ist klar. Zum Beweis von (4.11) sei $q(x)$ die rechte Seite von (4.11). Der Grad von $q(x)$ ist nicht größer als k, daher gilt

$$\begin{aligned} q(x_{i_0}) &= \frac{0 - (x_{i_0} - x_{i_k})P_{i_0,\ldots,i_{k-1}}(x_{i_0})}{x_{i_k} - x_{i_0}} \\ &= P_{i_0,\ldots,i_{k-1}}(x_{i_0}) \\ &= f_{i_0}, \\ q(x_{i_k}) &= \frac{(x_{i_k} - x_{i_0})P_{i_1,\ldots,i_k}(x_{i_k}) - 0}{x_{i_k} - x_{i_0}} \\ &= P_{i_1,\ldots,i_k}(x_{i_k}) \\ &= f_{i_k}. \end{aligned}$$

Weiterhin gilt für $j = 1, \ldots, k-1$, dass

$$\begin{aligned} q(x_{i_j}) &= \frac{(x_{i_j} - x_{i_0})P_{i_1,\ldots,i_k}(x_{i_j}) - (x_{i_j} - x_{i_k})P_{i_0,\ldots,i_{k-1}}(x_{i_j})}{x_{i_k} - x_{i_0}} \\ &= \frac{(x_{i_j} - x_{i_0})f_{i_j} - (x_{i_j} - x_{i_k})f_{i_j}}{x_{i_k} - x_{i_0}} \\ &= f_{i_j}. \end{aligned}$$

Die Behauptung folgt dann aus der Eindeutigkeit des Interpolationspolynoms. □

Damit können wir nun einen rekursiven Algorithmus aufbauen.

Algorithmus 1 Algorithmus von Neville–Aitken[2]
Gegeben sei eine Stelle x an der das Polynom $P_{0,1,\ldots,n}$ ausgewertet werden soll.
Konstruiere mit Hilfe der Rekursionsformel (4.10), (4.11) das folgende Tableau:

$$
\begin{array}{c|lclcc}
x_0 & f_0 = P_0(x) & & & & \\
 & & \rightarrow & P_{0,1}(x) = \frac{(x-x_0)P_1(x)-(x-x_1)P_0(x)}{x_1-x_0} & & \\
x_1 & f_1 = P_1(x) & & & \rightarrow & \\
 & & \rightarrow & P_{1,2}(x) = \frac{(x-x_1)P_2(x)-(x-x_2)P_1(x)}{x_2-x_1} & \ddots & \\
x_2 & f_2 = P_2(x) & & & & P_{0,1,\ldots,n}(x) \\
\vdots & \quad\vdots & & \quad\vdots & & \\
\vdots & \quad\vdots & & \quad\vdots & \rightarrow & \\
 & & \rightarrow & P_{n-1,n}(x) = \frac{(x-x_{n-1})P_n(x)-(x-x_n)P_{n-1}(x)}{x_n-x_{n-1}} & & \\
x_n & f_n = P_n(x) & & & &
\end{array}
$$

Beispiel 38 *Gegeben seien die Stützstellen* $(0,1),(1,3),(3,2)$. *Die Auswertung erfolgt an der Stelle* $x = 2$.

$$
\begin{array}{c|lclcl}
0 & 1 = P_0(x) & & & & \\
 & & \rightarrow & P_{0,1}(x) = \frac{(x-0)3-(x-1)1}{1-0} = 5 & & \\
1 & 3 = P_1(x) & & & \rightarrow & P_{0,1,2}(x) = \frac{(2-0)\frac{5}{2}-(2-3)5}{3-0} = \frac{10}{3} \\
 & & \rightarrow & P_{1,2}(x) = \frac{(x-1)2-(x-3)3}{3-1} = \frac{5}{2} & & \\
3 & 2 = P_2(x) & & & &
\end{array}
$$

Der Algorithmus von Neville-Aitken ist gut geeignet, um das Interpolationspolynom an einer oder wenigen Stellen auszuwerten, aber nicht um das Polynom selbst zu bestimmen, oder viele Auswertungen zu machen, (wie man sie zum Beispiel für einen Plot braucht).

4.2.2 Interpolation nach Newton

Eine Alternative zum Neville-Aitken Algorithmus stellt die Newton-Interpolation mit Hilfe von dividierten Differenzen dar.

[2] A.C. Aitken, Neuseeländischer Mathematiker, 1895–1966

Definition 39 *Seien Stützstellen $(x_i, f_i), i = 0, \ldots, n$ gegeben. Der Ausdruck $f[x_i, \ldots, x_{i+k}]$, definiert durch die Rekursion*

$$
\begin{aligned}
f[x_i] &:= f_i, && (4.12)\\
f[x_i, \ldots, x_{i+k}] &:= \frac{f[x_{i+1}, \ldots, x_{i+k}] - f[x_i, \ldots, x_{i+k-1}]}{x_{i+k} - x_i}, && (4.13)\\
& \quad\; i = 0, \ldots, n, \quad k = 0, \ldots, n-i,
\end{aligned}
$$

heißt k-te dividierte Differenz *von* $(x_i, f_i), \ldots, (x_{i+k}, f_{i+k})$.

Für die Berechnung der dividierten Differenzen haben wir wieder ein Schema:

$$
\begin{array}{c|ccccccc}
 & k=0 & & k=1 & & k=2 & & k=3 \\
\hline
x_0 & f_0 = f[x_0] & \searrow & & & & & \\
 & & & f[x_0,x_1] = \frac{f[x_1]-f[x_0]}{x_1-x_0} & \searrow & & & \\
x_1 & f_1 = f[x_1] & \begin{smallmatrix}\nearrow\\ \searrow\end{smallmatrix} & & & f[x_0,x_1,x_2] & \searrow & \\
 & & & f[x_1,x_2] = \frac{f[x_2]-f[x_1]}{x_2-x_1} & \begin{smallmatrix}\nearrow\\ \searrow\end{smallmatrix} & & & f[x_0,\ldots,x_3] \\
x_2 & f_2 = f[x_2] & \begin{smallmatrix}\nearrow\\ \searrow\end{smallmatrix} & & & f[x_1,x_2,x_3] & \begin{smallmatrix}\nearrow\\ \searrow\end{smallmatrix} & \\
 & & & f[x_2,x_3] = \frac{f[x_3]-f[x_2]}{x_3-x_2} & \begin{smallmatrix}\nearrow\\ \searrow\end{smallmatrix} & \vdots & & \\
x_3 & f_3 = f[x_3] & \begin{smallmatrix}\nearrow\\ \searrow\end{smallmatrix} & \vdots & & & & \\
x_4 & \vdots & & & & & & \\
\vdots & & & & & & &
\end{array}
$$

Als Basis von Π_n verwenden wir die Newton-Basis[3]:

$$
\begin{aligned}
N_0(x) &= 1, \\
N_i(x) &= \prod_{j=0}^{i-1}(x - x_j) && (4.14)\\
&= (x - x_0)(x - x_1)\ldots(x - x_{i-1}), \qquad i = 1, \ldots, n.
\end{aligned}
$$

[3]Sir Isaac Newton, 1643–1727, englischer Physiker und Mathematiker

Das Interpolationspolynom wird dann folgendermaßen konstruiert:

$$P_{0,\dots,n}(x) = \sum_{i=0}^{n} \gamma_i N_i(x). \tag{4.15}$$

Die Koeffizienten ergeben sich aus den dividierten Differenzen mit folgendem Theorem.

Theorem 40 *Es gilt*

$$\begin{aligned} P_{i,\dots,i+k}(x) \quad = \quad & f[x_i] + f[x_i, x_{i+1}](x - x_i) \\ & +f[x_i, x_{i+1}, x_{i+2}](x - x_i)(x - x_{i+1}) \\ & +\dots \\ & +f[x_i, x_{i+1}, \dots, x_{i+k}](x - x_i)(x - x_{i+1}) \dots (x - x_{i+k-1}). \end{aligned}$$

Insbesondere ist $\gamma_i = f[x_0, \dots, x_i]$*, d.h. die oberste Zeile im Schema der dividierten Differenzen liefert die* γ_i*.*

Beweis. Wir verwenden Induktion über k.

Für $k = 0$ ist die Behauptung offensichtlich wahr.

Sei die Behauptung wahr für $k - 1 \geq 0$. Dann gilt:

$$P_{i,\dots,i+k}(x) = P_{i,\dots,i+k-1}(x) + \gamma(x - x_i) \dots (x - x_{i+k-1}),$$

wobei γ gerade der Koeffizient von x^k ist.

Wenn wir zeigen, dass $\gamma = f[x_i, \dots, x_{i+k}]$, so gilt die Behauptung auch für k. Nach (4.11) ist

$$P_{i,i+1,\dots,i+k}(x) = \frac{(x - x_i)P_{i+1,\dots,i+k}(x) - (x - x_{i+k})P_{i,\dots,i+k-1}(x)}{x_{i+k} - x_i}$$

Nach Induktionsvoraussetzung ist $f[x_{i+1}, \dots, x_{i+k}]$ der Koeffizient zu x^{k-1} von $P_{i+1,\dots,i+k}(x)$ und $f[x_i, \dots, x_{i+k-1}]$ der Koeffizient zu x^{k-1} von $P_{i,\dots,i+k-1}(x)$. Also ist

$$\frac{f[x_{i+1}, \dots, x_{i+k}] - f[x_i, \dots, x_{i+k-1}]}{x_{i+k} - x_i} = f[x_i, \dots, x_{i+k}]$$

der Koeffizient zu x^k von $P_{i,\dots,i+k}(x)$, d.h. γ. $\square$

Beispiel 41 *Betrachte die Stützstellen*

$x_i =$	0	1	3
$f_i =$	1	3	2

.

Wir erhalten das dividierte Differenzen Schema:

$$\begin{array}{c|ccccc}
x_0 = 0 & f[x_0] = 1 & \searrow & & & \\
 & & & f[x_0, x_1] = 2 & \searrow & \\
x_1 = 1 & f[x_1] = 3 & \nearrow\searrow & & & f[x_0, x_1, x_2] = -\frac{5}{6} \\
 & & & f[x_1, x_2] = -\frac{1}{2} & \nearrow & \\
x_2 = 3 & f[x_2] = 2 & \nearrow & & &
\end{array}$$

und damit

$$P_{0,1,2}(x) = 1 + 2(x - 0) - \frac{5}{6}(x - 0)(x - 1).$$

Zur Auswertung von Polynomen verwendet man üblicherweise das Horner-Schema[4]. Dazu schreibt man das Polynom als

$$\begin{aligned}
P(x) &= a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n \\
&= (\ldots((a_n x + a_{n-1})x + a_{n-2})x + \ldots + a_1)x + a_0
\end{aligned}$$

und erhält für festes $\tilde{x}$ die Rekursion:

$$\begin{aligned}
u_{n+1} &= 0, \\
u_k &= \tilde{x} u_{k+1} + a_k, \qquad k = n, \ldots, 0, \\
P(\tilde{x}) &= u_0.
\end{aligned} \tag{4.16}$$

Der Vorteil des Horner-Schemas ist, das die Auswertung n Multiplikationen und n Additionen benötigt, also $2n$ flops kostet. Wenn man mittels der normalen Darstellung auswertet, würde das $2n^2$ flops kosten.

Für die Newton-Basis erhalten wir ein analoges Schema:

Algorithmus 2 *Hornerschema für Newton–Basis*

$$\begin{aligned}
P(x) &= \sum_{i=0}^{n} \gamma_i N_i(x), \\
N_i(x) &= (x - x_0)(x - x_1) \ldots (x - x_{i-1}), \\
&= N_{i-1}(x)(x - x_{i-1}).
\end{aligned}$$

[4] W.G. Horner, 1786–1837, englischer Mathematiker

Rekursion für festes $\tilde{x}$:

$$\begin{aligned} u_{n+1} &= 0, \\ u_k &= (\tilde{x} - x_k)u_{k+1} + \gamma_k, \qquad k = n, \dots, 0, \\ P(\tilde{x}) &= u_0. \end{aligned} \tag{4.17}$$

In diesem Fall werden zur Auswertung des Polynoms n Multiplikationen und $2n$ Additionen benötigt, die Kosten betragen also $3n$ flops.

Bemerkung 42 *Die Newton–Interpolation in Verbindung mit dem Hornerschema bietet bei der Bestimmung eines Interpolationspolynoms eine Reihe von Vorteilen:*

- *Einfache Berechnung des gesamten Interpolationspolynoms mit dividierten Differenzen.*
- *Leichte Auswertung mit dem Hornerschema.*
- *Einfaches Hinzufügen weiterer Stützstellen.*
- $f[x_0, \dots, x_n]$ *ist symmetrisch in den* x_i*, d.h. die Reihenfolge ist egal.*
- *Der Rundungsfehlereinfluß ist durch Umordnung der Stützstellen reduzierbar.*

Dividierte Differenzen lassen sich auf zusammenfallende Stützstellen $x_i = x_j$ erweitern falls $f(x_j) = f(x_i)$.

$$\begin{aligned} f[x_k, x_k] &:= \lim_{h\to 0} \frac{f[x_k + h] - f[x_k]}{x_k + h - x_k} = f'(x_k) \\ f[x_k, x_k, x_k] &:= \lim_{h\to 0} \frac{f[x_k + h, x_k + 2h] - f[x_k, x_k + h]}{x_k - 2h - x_k} \\ &= \lim_{h\to 0} \frac{f[x_k + 2h] - 2f[x_k + h] + f[x_k]}{2h^2} \\ &= \frac{1}{2} f''(x_k) \\ f[\underbrace{x_k, \dots, x_k}_{m+1 \text{ mal}}] &:= \frac{1}{m!} f^{(m)}(x_k). \end{aligned}$$

Damit kann man dann allgemeine Hermite–Interpolation[5] durchführen, bei der für mehrfache Stützstellen jeweils die entsprechenden Ableitungen mit interpoliert werden.

Beispiel 43

$$
\begin{array}{c|cccccccccccc}
 & c_0 & & c_1 & & c_2 & & c_3 & & c_4 & & c_5 \\
x_0 & f[x_0] & \searrow & & & & & & & & & \\
 & & & f'(x_0) & \searrow & & & & & & & \\
x_0 & f[x_0] & \begin{smallmatrix}\nearrow\\ \searrow\end{smallmatrix} & & & \frac{1}{2}f''(x_0) & \searrow & & & & & \\
 & & & f'(x_0) & \begin{smallmatrix}\nearrow\\ \searrow\end{smallmatrix} & & & f[x_0,x_0,x_0,x_1] & \searrow & & & \\
x_0 & f[x_0] & \begin{smallmatrix}\nearrow\\ \searrow\end{smallmatrix} & & & f[x_0,x_0,x_1] & \begin{smallmatrix}\nearrow\\ \searrow\end{smallmatrix} & & & f[x_0,x_0,x_0,x_1,x_1] & \searrow & \\
 & & & f[x_0,x_1] & \begin{smallmatrix}\nearrow\\ \searrow\end{smallmatrix} & & & f[x_0,x_0,x_1,x_1] & \begin{smallmatrix}\nearrow\\ \searrow\end{smallmatrix} & & & c_5 \\
x_1 & f[x_1] & \begin{smallmatrix}\nearrow\\ \searrow\end{smallmatrix} & & & f[x_0,x_1,x_1] & \begin{smallmatrix}\nearrow\\ \searrow\end{smallmatrix} & & & f[x_0,x_0,x_1,x_1,x_1] & \nearrow & \\
 & & & f'(x_1) & \begin{smallmatrix}\nearrow\\ \searrow\end{smallmatrix} & & & f[x_0,x_1,x_1,x_1] & \nearrow & & & \\
x_1 & f[x_1] & \begin{smallmatrix}\nearrow\\ \searrow\end{smallmatrix} & & & \frac{1}{2}f''(x_1) & \nearrow & & & & & \\
 & & & f'(x_1) & \nearrow & & & & & & & \\
x_1 & f[x_1] & \nearrow & & & & & & & & &
\end{array}
$$

Das Interpolationspolynom ist damit

$$
\begin{aligned}
P_{000111}(x) \quad = \quad & c_0 + c_1(x-x_0) + c_2(x-x_0)^2 + c_3(x-x_0)^3 \\
& + c_4(x-x_0)^3(x-x_1) + c_5(x-x_0)^3(x-x_0)^2.
\end{aligned}
$$

Beispiel 44 *Betrachte Stützstellen* $x_0, x_0, x_1, x_1, \ldots$ *. Das Schema ergibt*

$$
\begin{aligned}
P_{001122\ldots kk}(x) \quad = \quad & [c_0 + c_1(x-x_0)] + [c_2 + c_3(x-x_1)](x-x_0)^2 \\
& + [c_4 + c_5(x-x_2)](x-x_0)^2(x-x_1)^2 + \cdots \\
& + [c_{2k} + c_{2k-1}(x-x_k)](x-x_0)^2 \cdots (x-x_{k-1})^2.
\end{aligned}
$$

Es gilt für $i = 0, \ldots, k$ *die Bedingung*

$$
\begin{aligned}
P_{00\ldots kk}(x_i) &= f(x_i), \\
P'_{00\ldots kk}(x_i) &= f'(x_i).
\end{aligned}
$$

[5]C. Hermite, Französischer Mathematiker, 1822–1901

Wir wollen die Interpolation verwenden, um Verfahren höherer Ordung für die Lösung von Differentialgleichungen zu konstruieren. Dazu müssen wir die Frage klären:

Gegeben eine Funktion f und Auswertungsstellen $x_0, \ldots, x_n$ und Werte $f_i = f(x_i)$ für $i = 0, \ldots, n$.

Wie gut approximiert das entsprechende Interpolationspolynom $P_{0,\ldots,n}(x)$ die Funktion f?

Theorem 45 *Die Funktion f sei $n+1$–mal differenzierbar. Dann gibt es zu jedem $\hat{x}$ eine Zahl ξ aus dem kleinsten Intervall I, das alle x_i und $\hat{x}$ enthält (konvexe Hülle $I = \mathcal{C}[x_0, \ldots, x_n, \hat{x}]$), so dass*

$$f(\hat{x}) - P_{01\ldots n}(\hat{x}) = w(\hat{x})\frac{f^{(n+1)}(\xi)}{(n+1)!} \tag{4.18}$$

wobei $w(x) = \prod_{i=0}^{n}(x - x_i)$.

Beweis. Sei für $\hat{x} \neq x_i$

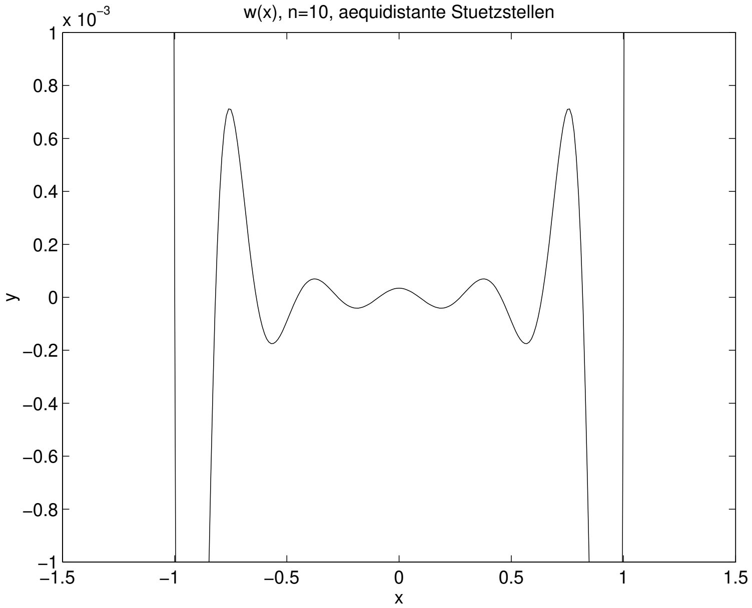
$$F(x) := f(x) - P_{0,1,\ldots,n}(x) - kw(x).$$

Bestimme k derart, daß $F(\hat{x}) = 0$. Dann hat $F(x)$ in $\mathcal{C}[x_0, \ldots, x_n, \hat{x}]$ $\quad n+2$ Nullstellen $x_0, \ldots, x_n, \hat{x}$.
Nach dem Satz von Rolle hat dann $F'(x)$ mindestens $n+1$ Nullstellen, ..., $F^{(n+1)}(x)$ mindestens eine Nullstelle $\xi \in \mathcal{C}[x_0, \ldots, x_n\hat{x}]$.

$$\begin{aligned} P^{(n+1)}_{0,1,\ldots,n}(x) \equiv 0 \quad &\Longrightarrow \quad F^{(n+1)}(\xi) = f^{(n+1)}(\xi) - k(n+1)! = 0 \\ &\Longrightarrow \quad k = \frac{f^{(n+1)}(\xi)}{(n+1)!}. \end{aligned}$$

□

Wie sieht $w(x)$ aus?



Die Funktion $w(x)$ wächst außerhalb von $\mathcal{C}[x_0, \ldots, x_n]$ stark an und oszilliert sehr stark. Man sollte daher das Interpolationspolynoms außerhalb von $\mathcal{C}[x_0, \ldots, x_n]$ nicht verwenden und damit der Fehler nicht größer wird, nicht zu viele Stützstellen haben.

Sei nun $f : [a, b] \to \mathbb{R}$ gegeben, f genügend glatt. Dann kann ich f beliebig genau durch Polynome interpolieren. Man könnte nun vermuten, daß die Interpolationspolynome gegen f konvergieren, falls ich die Intervallunterteilung feiner und feiner mache.

Betrachte eine Folge von Unterteilungen

$$\begin{aligned} \triangle_m &= \{a = x_0^{(m)} < x_1^{(m)} < \cdots < x_{n_m}^{(m)} = b\}, \\ \| \triangle_m \| &= \max_i |x_{i+1}^{(m)} - x_i^{(m)}|. \end{aligned}$$

Theorem 46 *[Satz von Faber][6] Zu jeder Folge von Intervalleinteilungen $\triangle_m$ von $[a, b]$ kann man eine auf $[a, b]$ stetige Funktion f finden, so dass die Polynome $P_{\triangle_m}$ die auf $(x_i^{(m)}, f(x_i^{(m)})), i = 0, \ldots, n_m$ interpolieren für $m \to \infty$ NICHT gleichmäßig gegen $f(x)$ konvergieren.*

[6] G. Faber Deutscher Mathematiker, 1877–1966

Nur für ganze Funktionen gilt für alle $\| \triangle_m \| \to 0$, $P_m \to f$ gleichmäßig.

Für beliebige stetige Funktionen gibt es spezielle $\triangle_m$, diese sind i.a. nicht äquidistant.

4.3 Trigonometrische Interpolation

Wenn Interpolation für ein Problem gemacht werden soll, welches periodisches Verhalten aufweist, so bietet es sich an trigonometrische Polynome zu verwenden, d.h., jetzt soll die Funktion f (oder die Folge von Stützstellen), im Intervall $[0, 2\pi]$ durch ein trigonometrisches Polynom der Form

$$P(x) = \beta_0 + \beta_1 e^{ix} + \beta_2 e^{2ix} + \ldots + \beta_{n-1} e^{(n-1)ix}, \tag{4.19}$$

durch die Stützstellen $(x_k, f_k), k = 0, \ldots, n-1$ mit

$$x_k = k\frac{2\pi}{n}, \qquad f_k \in \mathbb{C}$$

interpoliert werden. Dies nennt man auch diskrete Fouriertransformation[7].

Dabei soll $P(x)$ die Interpolationsbedingung

$$P(x_k) = f_k, \qquad k = 0, \ldots, n-1 \tag{4.20}$$

erfüllen. Aus der Periodizität von $P(x)$ folgt

$$P(x_k) = f_k, \qquad k = \ldots, -3, -2, -1, 0, 1, 2, 3, \ldots. \tag{4.21}$$

Setze

$$w = e^{ix} \qquad w_k = e^{ix_k}. \tag{4.22}$$

Dann ist die Interpolationsaufgabe identisch mit der Aufgabe

$$p(w_k) = f_k, \qquad k = 0, \ldots, n-1 \tag{4.23}$$

mit

$$p(w) = \beta_0 + \beta_1 w + \beta_2 w^2 + \ldots + \beta_{n-1} w^{n-1}. \tag{4.24}$$

Wir erhalten das folgende Theorem:

[7] J. Baron de Fourier, französischer Physiker und Mathematiker, 1768–1830

Theorem 47 *Sei* $x_k = \frac{2\pi k}{n}, f_k$ *beliebig für* $k = 0, \ldots, n-1, n \in \mathbb{N}$. *Dann gibt es ein eindeutiges trigonometrisches Polynom* $p(w) = \beta_0 + \beta_1 w + \cdots + \beta_{n-1} w^{n-1}$ *mit* $p(w_k) = f_k$, *für* $w_k = e^{ix_k}$, $k = 0, \ldots, n-1$.

Zusätzlich gilt

$$
\begin{array}{ll}
i) & w_k^j = w_j^k \quad \forall j, k \in \mathbb{Z} \\
ii) & \sum\limits_{k=0}^{n-1} w_k^j w_k^{-l} = \begin{cases} n & \text{für } j = l \\ 0 & \text{für } j \neq l, 0 \leq j, l \leq n-1. \end{cases}
\end{array}
$$

Beweis. Die Existenz und Eindeutigkeit folgt aus Theorem 35.

Teil i) ist trivial.

Um Teil ii) zu zeigen, beachte, dass für alle $k = \mathbb{Z}$, jedes w_k eine Nullstelle von

$$w^n - 1 = (w-1)(w^{n-1} + w^{n-2} + \cdots + 1) = 0$$

ist. Für $k \neq 0, \pm n, \pm 2n$, ist $w_k \neq 1$, also folgt

$$\sum_{l=0}^{n-1} w_k^l = \begin{cases} n & k = 0, \pm n, \pm 2n, \ldots \\ 0 & \text{sonst.} \end{cases}$$

□

Die n–Vektoren $\Phi_j = (w_0^j, \ldots, w_{n-1}^j)$, $j = 0, \ldots, n-1$ bilden eine Orthonormalbasis von $\mathbb{C}^n$ unter dem Skalarprodukt

$$[f, g] := \frac{1}{n} \sum_{j=0}^{n-1} f_j \bar{g}_j \tag{4.25}$$

(diskrete Version von $(f, g) = \frac{1}{\pi} \int\limits_{-\pi}^{\pi} f(x)\overline{g(x)} dx$), denn es gilt

$$[\Phi_j, \Phi_l] = \begin{cases} 1 & j = l \\ 0 & j \neq l \quad 0 \leq j, l, \leq n-1. \end{cases}$$

Damit erhält man eine explizite Formel für die Koeffizienten β_i.

Theorem 48 *Sei $p(x) = \sum_{j=0}^{n-1} \beta_j e^{ijx}$ das interpolierende trigonometische Polynom für $p(x_k) = f_k,\ k = 0, \dots, n-1$, so gilt*

$$\beta_j = \frac{1}{n}\sum_{k=0}^{n-1} f_k w_k^{-j} = \frac{1}{n}\sum_{k=0}^{n-1} f_k \exp\left\{\frac{-2\pi ikj}{n}\right\},\ j = 0, \dots, n-1. \tag{4.26}$$

Beweis. Aus dem Skalarprodukt $[\bullet, \bullet]$ folgt

$$\frac{1}{n}\sum_{k=0}^{n-1} f_k w_k^{-j} = [f, \Phi_j] = [\beta_0\Phi_0 + \dots + \beta_{n-1}\Phi_{n-1}, \Phi_j] = \beta_j.$$

□

Es gilt die folgende Minimaleigenschaft der trigonometrischen Interpolation.

Theorem 49 *Unter allen trigonometrischen Polynomen q_s der Form $q_s(x) = \gamma_0 + \gamma_1 e^{ix} + \dots + \gamma_s e^{isx},\ s \le n-1$, minimiert das s–te Abschnittspolynom $p_s(x) = \beta_0 + \beta_1 e^{xi} + \dots + \beta_s e^{sxi}$ für $s = 0, \dots, n-1$ die Fehlerquadratsumme*

$$S(q_s) = \sum_{k=0}^{n-1} |f_k - q_s(x_k)|^2. \tag{4.27}$$

Beweis. Ordne p_s, q_s die n–Tupel

$$\begin{aligned}\underline{p_s} &= (p_s(x_0), \dots, p_s(x_{n-1}))\\ \underline{q_s} &= (q_s(x_0), \dots, q_s(x_{n-1}))\end{aligned}$$

zu, die p_s, q_s eindeutig festlegen. Dann gilt $\frac{1}{n}S(q_s) = [f - \underline{q_s}, f - \underline{q_s}]$. Da aber $\beta_k = [f, \Phi_k]$ für $k = 0, \dots, n-1$ folgt für $f \in \mathbb{C}^n$, dass $[f - \underline{p_s}, \Phi_j] = [f - \sum_{k=0}^{s} \beta_k\Phi_k, \Phi_j] = \beta_j - \beta_j = 0, j = 0, \dots, s$ und

$$\begin{aligned}&[f - \underline{p_s}, \underline{p_s} - \underline{q_s}] = \sum_{j=0}^{s}[f - \underline{p_s}, (\beta_j - \gamma_j)\Phi_j] = 0,\\ &\tfrac{1}{n}S(q_s) = [f - \underline{q_s}, f - \underline{q_s}] = [(f - \underline{p_s} + \underline{p_s} - \underline{q_s}), (f - \underline{p_s} + \underline{p_s} - \underline{q_s})\\ &= [f - \underline{p_s}, f - \underline{p_s}] + [\underline{p_s} - \underline{q_s}, \underline{p_s} - \underline{q_s}]\\ &\ge [f - \underline{p_s}, f - \underline{p_s}] = \tfrac{1}{n}S(p_s)\end{aligned}$$

und Gleichheit nur für $\underline{p_s} = \underline{q_s}$, da $[\bullet, \bullet]$ eine Norm definiert. $\square$

Bisher haben wir komplex gearbeitet, es gibt aber auch eine reelle Version: Beachte, dass $e^{ix} = \cos x + i \sin x$ und setze

$$\begin{aligned} A_j &= \frac{2}{n} \sum_{k=0}^{n-1} f_k \cos \frac{2\pi jk}{n}, \\ B_j &= \frac{2}{n} \sum_{k=0}^{n-1} f_k \sin \frac{2\pi jk}{n}. \end{aligned} \tag{4.28}$$

Für reelle f_k sind A_j, B_j reell und es gilt

$$\beta_{n-j} = \frac{1}{n} \sum_{k=0}^{n-1} f_k w_k^{j-n} = \frac{1}{n} \sum_{k=0}^{n-1} f_k w_k^j, \tag{4.29}$$

$$\beta_j = \frac{1}{2}[A_j - iB_j], \beta_{n-j} = \frac{1}{2}[A_j + iB_j], \tag{4.30}$$

und

$$\beta_j w_k^j + \beta_{n-j} w_k^{n-j} = A_j \cos jx_k + B_j \sin jx_k. \tag{4.31}$$

Damit erhält man die reelle diskrete Fouriertransformation.

Die Koeffizienten A_j, B_j sind diskrete Approximationen an die Fourierkoeffizienten

$$\begin{aligned} a_k &= \tfrac{1}{\pi} \int_{-\pi}^{\pi} f(x) \cos(kx)\, dx, \quad k = 0, \ldots, n-1, \\ b_k &= \tfrac{1}{\pi} \int_{-\pi}^{\pi} f(x) \sin(kx)\, dx, \quad k = 1, \ldots, n-1. \end{aligned}$$

Theorem 50 *Gegeben* (x_k, f_k), $k = 0, \ldots, N-1$ *mit* $x_k = \frac{2\pi k}{N}$. *Man setze für* $j \in \mathbb{Z}$

$$\begin{aligned} A_j &:= \tfrac{2}{N} \sum_{k=0}^{N-1} f_k \cos(kx_j), \\ B_j &:= \tfrac{2}{N} \sum_{k=0}^{N-1} f_k \sin(kx_j). \end{aligned} \tag{4.32}$$

Definiere für ungerades $N = 2M + 1$

$$\Psi(x) = \frac{A_0}{2} + \sum_{k=1}^{M} (A_k \cos kx + B_k \sin kx) \tag{4.33}$$

und für gerades $N = 2M$

$$\Psi(x) = \frac{A_0}{2} + \sum_{k=1}^{M-1} (A_k \cos kx + B_k \sin kx) + \frac{A_M}{2} \cos Mx, \tag{4.34}$$

so gilt

$$\Psi(x_k) = f_k, \; k = 0, \ldots, N-1.$$

Beweis. Mit $N = 2M$ folgt

$$f_k = \sum_{j=0}^{N-1} \beta_j w_k^j = \beta_0 + \sum_{j=1}^{M-1} (\beta_j w_k^j + \beta_{N-j} w_k^{(N-j)}) + \beta_M w_k^M$$

und mit (4.30), (4.31) folgt

$$\begin{aligned} B_0 &= \tfrac{2}{N} \sum_{k=0}^{N-1} f_k \sin \tfrac{2\pi k \cdot 0}{N} = 0, \\ B_M &= \tfrac{2}{N} \sum_{k=0}^{N-1} f_k \sin \tfrac{2\pi k \cdot 0}{N} = 0. \end{aligned}$$

Der Beweis für ungerades N geht analog. □

Die Berechnung der Koeffizienten des trigonometrischen Polynomes geht sehr effizient mit **Hilfe der schnellen (fast) Fouriertransformation (FFT)** für $N = 2^n, \quad n > 0$.

Was ist zu berechnen?

$$\begin{aligned} \beta_j &= \frac{1}{N} \sum_{k=0}^{N-1} f_k e^{\frac{-2\pi i k j}{N}} \\ &= \frac{1}{N} \sum_{k=0}^{N-1} f_k \epsilon_N^{kj} \end{aligned}$$

mit $\epsilon_N = e^{\frac{-2\pi i}{N}}$.

Der Trick ist, die Tatsache auszunutzen, dass $e^{\frac{-2\pi i}{N}}$ eine N–te Einheitswurzel ist, d.h. alle Potenzen liegen auch auf dem Einheitskreis und die Exponenten lassen sich einfach berechnen.

Beispiel 51 *Betrachte als Beispiel die FFT für $N = 4$.*

$$\begin{bmatrix}\beta_0\\ \beta_1\\ \beta_2\\ \beta_3\end{bmatrix} = \frac{1}{N}\begin{bmatrix}\epsilon_N^0 & \epsilon_N^0 & \epsilon_N^0 & \epsilon_N^0\\ \epsilon_N^0 & \epsilon_N^1 & \epsilon_N^2 & \epsilon_N^3\\ \epsilon_N^0 & \epsilon_N^2 & \epsilon_N^4 & \epsilon_N^6\\ \epsilon_N^0 & \epsilon_N^3 & \epsilon_N^6 & \epsilon_N^9\end{bmatrix}\begin{bmatrix}f_0\\ f_1\\ f_2\\ f_3\end{bmatrix} = \frac{1}{N}\begin{bmatrix}1 & 1 & 1 & 1\\ 1 & \epsilon_N^1 & \epsilon_N^2 & \epsilon_N^3\\ 1 & \epsilon_N^2 & 1 & \epsilon_N^2\\ 1 & \epsilon_N^3 & \epsilon_N^2 & \epsilon_N^1\end{bmatrix}\begin{bmatrix}f_0\\ f_1\\ f_2\\ f_3\end{bmatrix}$$

$$\begin{bmatrix}\beta_0\\ \beta_2\\ \beta_1\\ \beta_3\end{bmatrix} = \frac{1}{N}\begin{bmatrix}1 & 1 & 1 & 1\\ 1 & \epsilon_N^2 & 1 & \epsilon_N^2\\ 1 & \epsilon_N & \epsilon_N^2 & \epsilon_N^3\\ 1 & \epsilon_N^3 & \epsilon_N^2 & \epsilon_N\end{bmatrix}\begin{bmatrix}f_0\\ f_1\\ f_2\\ f_3\end{bmatrix}$$

$$= \frac{1}{N}\left[\begin{array}{cc|cc}1 & 1 & 0 & 0\\ 1 & \epsilon_N^2 & 0 & 0\\ \hline 0 & 0 & 1 & 1\\ 0 & 0 & 1 & \epsilon_N^2\end{array}\right]\left[\begin{array}{cc|cc}1 & 0 & 1 & 0\\ 0 & 1 & 0 & 1\\ \hline 1 & 0 & \epsilon_N^2 & 0\\ 0 & \epsilon_N & 0 & \epsilon_N^3\end{array}\right]\begin{bmatrix}f_0\\ f_1\\ f_2\\ f_3\end{bmatrix}$$

Bilde zuerst

$$\begin{bmatrix}z_0\\ z_1\\ z_2\\ z_3\end{bmatrix} = \begin{bmatrix}1 & 0 & 1 & 0\\ 0 & 1 & 0 & 1\\ 1 & 0 & \epsilon_N^2 & 0\\ 0 & \epsilon_N & 0 & \epsilon_N^3\end{bmatrix}\begin{bmatrix}f_0\\ f_1\\ f_2\\ f_3\end{bmatrix}.$$

Verwende $\epsilon_N^2 = -1, \epsilon_N^3 = -\epsilon_N$

$$\begin{aligned} z_0 &= f_0 + f_2 \quad z_2 = f_0 - f_2,\\ z_1 &= f_1 + f_3 \quad z_3 = \epsilon_N(f_1 - f_3),\end{aligned}$$

$$\begin{bmatrix}\beta_0\\ \beta_2\\ \beta_1\\ \beta_3\end{bmatrix} = \frac{1}{N}\underbrace{\begin{bmatrix}1 & 1 & 0 & 0\\ 1 & \epsilon_N^2 & 0 & 0\\ 0 & 0 & 1 & 1\\ 0 & 0 & 1 & \epsilon_N^2\end{bmatrix}}_{\text{2 komplexe FT der Ordnung 2 mit } \epsilon_N^2} \begin{bmatrix}z_0\\ z_1\\ z_2\\ z_3\end{bmatrix} = \frac{1}{N}\begin{bmatrix}1 & 1 & & \\ 1 & -1 & & \\ & & 1 & 1\\ & & 1 & -1\end{bmatrix}\begin{bmatrix}z_0\\ z_1\\ z_2\\ z_3\end{bmatrix},$$

also

$$\begin{aligned} N\beta_0 &= z_0 + 1z_1, \quad N\beta_1 = z_2 + z_3,\\ N\beta_2 &= z_0 - z_1, \quad N\beta_3 = z_2 - z_3.\end{aligned}$$

Es findet jeweils eine Rückführung einer FFT der Ordnung $N = 2^n$ auf 2 FFT der Ordnung $\frac{N}{2} = 2^{n-1}$ statt. Als zusätzlicher Aufwand ergeben sich

Kosten von $\mathcal{O}(N)$ flops. Anstatt $\mathcal{O}(N^2)$ für Matrixmultiplikation erhält man dann insgesamt $\frac{1}{2}N\log_2 N = 2^{n-1} \times n$ Operationen.

Da wir die zwei FFT halber Dimension unabhängig laufen können, lassen sie sich gut auf Parallelrechner implementieren.

Für den Schritt von N auf $N/2$ erhalten wir die allgemeinen Formeln: Sei $M = \frac{N}{2}$ und setze $j = 2l$, dann ist

$$\begin{aligned} \beta_{2l} &= \sum_{k=0}^{2M-1} f_k \epsilon_N^{2lk} = \sum_{k=0}^{M-1} (f_k + f_{M+k}) \epsilon_N^{2lk} \\ &= \sum_{k=0}^{M-1} (f_k + f_{M+k}) (\epsilon_N^2)^{lk}, \end{aligned} \tag{4.35}$$

wobei wir benutzen, dass

$$\epsilon_N^{2l(M+k)} = \epsilon_N^{2lk} \epsilon_N^{2lM} = \epsilon_N^{2lk}.$$

Mit Hilfswerten

$$z_k = f_k + f_{M+k},\ k = 0, \ldots, M-1 \tag{4.36}$$

und mit $\epsilon_N^2 = \epsilon_M$ folgt

$$\beta_{2l} = \sum_{k=0}^{M-1} z_k \epsilon_M^{kl}, \quad l = 0, \ldots, M-1. \tag{4.37}$$

Für die ungeraden Indizes gilt analog

$$\beta_{2l+1} = \sum_{k=0}^{M-1} z_{M+k} \epsilon_M^{kl}, \quad l = 0, \ldots, m-1 \tag{4.38}$$

mit Hilfswerten

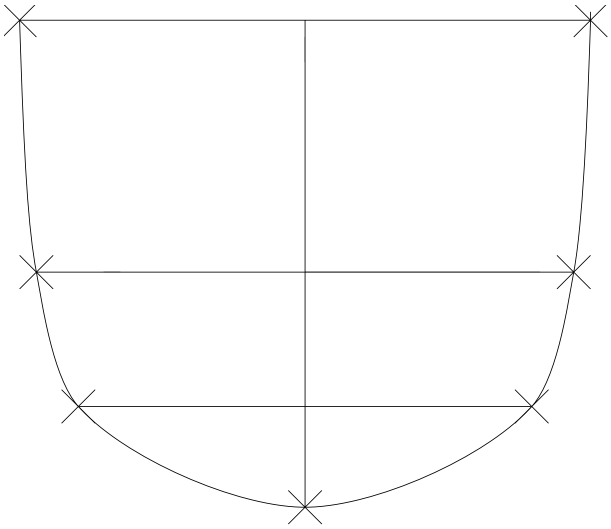
$$z_{M+k} = f_k - f_{M+k} \epsilon_N^k,\ k = 0, \ldots, M-1 \tag{4.39}$$

Der zugehörige Algorithmus von Cooley/Tukey (1965) enthält noch weitere Tricks.

4.4 Spline Interpolation

Wir haben gesehen, dass die Polynominterpolation nicht unbedingt gute Ergebnisse liefert, insbesondere dass der Fehler stark oszillieren kann.

Als Ausweg kann man die Interpolation durch stückweise Polynome durchführen. Dies ist ursprünglich schon im Schiffbau ein übliches Verfahren, daher auch der Name.



Heute sind Splines die wichtigsten Werkzeuge bei der Lösung von Randwertproblemen für gewöhnliche und partielle Differentialgleichungen, bei der Computergraphik, und vielen anderen Anwendungen, weil sie sehr schön glatte Kurven liefern.

Durch $\triangle := \{a = x_0 < x_1 \cdots < x_n = b\}$ sei eine Unterteilung von $[a, b]$ gegeben.

Definition 52 *Unter einer zu $\triangle$ gehörenden Spline–Funktion $S_\triangle$ versteht man eine reelle Funktion $S_\triangle : [a, b] \to \mathbb{R}$ mit*

a) $S_\triangle \in \mathcal{C}^{2k}[a, b]$ *(2k mal stetig differenzierbar).*

b) Auf jedem Teilintervall $[x_i, x_{i+1}]$ stimmt $S_\triangle$ mit Polynom $2k + 1$–ten Grades überein.

Hier sei im folgenden $k = 1$, das sind kubische Splines. Fast alles läßt sich übertragen auf andere k.

Sei $\kappa^n(a,b) := \big\{f : [a,b] \to \mathbb{R}; f^{(i)},\ i = 1,\ldots,n-1,$ existiert und ist absolut stetig, $f^{(n)}$ existiert fast überall auf $[a,b]$ und $f^{(n)} \in L^2[a,b]\big\}$.

Zur Erinnerung: Eine Funktion f heißt absolut stetig, falls zu jedem $\epsilon > 0$ ein $\delta > 0$ existiert, so dass für jedes endliche System von Intervallen $[a_i, b_i]$ mit $a \leq a_1 < b_1 \leq a_2 < b_2 \cdots \leq a_m < b_m \leq b$ mit $\sum\limits_i |b_i - a_i| < \delta$ gilt: $\sum\limits_i |f(b_i) - f(a_i)| < \epsilon$.

Wir haben, dass $S_\triangle \in \kappa^{2k+1}(a,b)$.

Sei weiterhin $\kappa_p^n(a,b) = \{f \in \kappa^n(a,b), f, f', \ldots, f^{(n-1)}, (b-a)-periodisch\}$, d.h. $f^{(i)}(a) = f^{(i)}(b) \quad i = 0,\ldots,n-1$. Eine Spline–Funktion heißt periodisch, falls

$$S_\triangle \in \kappa_p^{2k+1}(a,b).$$

Wir wollen mit Splines interpolieren, d.h. wir wollen, dass $S_\triangle(x_i) = f_i = f(x_i),\ i = 0,\ldots,n$. Dann bezeichnen wir den Spline mit $S_\triangle^f$. Führe als Maß für die Güte der Approximation $w(f) := \int\limits_a^b |f''(x)|^2\,dx$ ein. Es gilt $w(f) \geq 0$, jedoch $w(f) = 0$ für alle $f = cx + d$.

Theorem 53 *Sei $f \in \kappa^2(a,b), \triangle = \{a = x_0 < \cdots < x_n = b\}$ und $S_\triangle$ kubischer Spline zu $\triangle$. Dann gilt*

$$w(f - S_\triangle) = w(f) - w(S_\triangle) - 2\left[(f' - S_\triangle')S_\triangle''\Big|_a^b - \sum_{i=1}^n (f - S_\triangle)S_\triangle'''\Big|_{x_{i-1}^+}^{x_i^-}\right].$$

Beweis. Es gilt

$$w(f - S_\triangle) \;=\; \int\limits_a^b |f'' - S_\triangle''|^2\,dx$$

$$
\begin{aligned}
&= \int_a^b |f''|^2\,dx - 2\int_a^b f'' S''_\triangle dx + \int_a^b |S''_\triangle|^2\,dx \\
&= w(f) - 2\int_a^b (f'' - S''_\triangle) S''_\triangle\,dx - w(S_\triangle).
\end{aligned}
$$

Partielle Integration in Teilintervallen ergibt

$$
\begin{aligned}
\int_{x_{i-1}}^{x_i} (f'' - S''_\triangle) S''_\triangle dx &= (f' - S'_\triangle) S''_\triangle \Big|_{x_{i-1}}^{x_i} - \int_{x_{i-1}}^{x_i} (f' - S'_\triangle) S'''_\triangle dx \\
&= (f' - S'_\triangle) S''_\triangle \Big|_{x_{i-1}}^{x_i} - (f - S_\triangle) S'''_\triangle \Big|_{x_{i-1}^+}^{x_i^-} + \int_{x_{i-1}}^{x_i} (f - S_\triangle) S^{(4)}_\triangle\,dx.
\end{aligned}
$$

Da $S^{(4)} \equiv 0$ auf allen $[x_{i-1}, x_i]$ und

$$
\sum_{i=1}^{n} (f' - S'_\triangle) S''_\triangle \Big|_{x_{i-1}}^{x_i} = (f' - S'_\Delta) S''_\triangle \Big|_a^b
$$

folgt die Behauptung. □

Dieses Theorem ist Grundlage für die Minimaleigenschaft der Splines.

Theorem 54 *Sei $\triangle = \{a = x_0 < x_1 \cdots < x_n = b\}$ und eine Menge von Funktionswerten $\{f_0, \ldots, f_n\}$, gegeben, wobei $f(x_i) = f_i$, für $i = 0, \ldots, n$ und $f \in \kappa^2(a, b)$. Dann gilt*

$$
w(f - S^f_\triangle) = w(f) - w(S^f_\triangle) \geq 0 \tag{4.40}
$$

für alle interpolierenden $S^f_\triangle$, die einer der drei folgenden Bedingungen erfüllen.

$$
a) \quad S''_\triangle(a) = S''_\triangle(b) = 0; \tag{4.41}
$$

$$
\text{oder } b) \quad f \in \kappa^2_p(a, b) \quad S_\triangle \text{ periodisch}; \tag{4.42}
$$

$$
\text{oder } c) \quad f'(a) = S'_\triangle(a), f'(b) = S'_\triangle(b). \tag{4.43}
$$

Für jeden dieser Fälle ist $S_\triangle$ eindeutig bestimmt.

Beweis. In jedem dieser Fälle verschwindet

$$\left((f' - S'_\triangle)S''_\triangle\right)\Big|_b^a - \sum_{i=1}^{n} (f - S_\triangle)S'''_\triangle\Big|_{x_{i-1}^+}^{x_i^-} = 0.$$

Für die Eindeutigkeit sei angenommen, es gäbe $\tilde{S}_\triangle^f$ mit den gleichen Eigenschaften. Dann könnte man $f(x) = \tilde{S}_\triangle^f(x)$ setzen. Daraus folgt, dass

$$w(\tilde{S}_\triangle^f(x) - S_\triangle^f(x)) = w(\tilde{S}_\triangle^f(x)) - w(S_\triangle^f(x)) \geq 0$$

und genauso, dass

$$w(S_\triangle^f(x)) - w(\tilde{S}_\triangle^f(x)) \geq 0$$

also insgesamt, dass

$$\int_a^b (\tilde{S}''_\triangle - S''_\triangle)^2 dx = 0.$$

Da beide Ableitungen $\tilde{S}'', S''$ stetig sind folgt, dass

$$\tilde{S}''_\triangle = S''_\triangle \Longrightarrow \tilde{S}_\triangle^f = S_\triangle^f + cx + d,$$

und wegen $\tilde{S}_\triangle^f(x) = S_\Delta^f(x)$ für a, b folgt $c = d = 0$. □

Dieses Theorem besagt, dass die Splines unter allen Funktionen aus $\kappa^2(a,b)$ die interpolierende Funktion mit der geringsten Krümmung sind, denn es gilt $\kappa(x) = \frac{f''(x)}{\sqrt{1+f'(x)^2}^3} \approx f''(x)$ für $f'(x)$ klein.

Analoge Eigenschaften gelten für andere Splines.

Zur expliziten Berechnung der Splines setze $h_{j+1} = x_{j+1} - x_j$ für $j = 0, \ldots, n-1$ und

$$M_j = S''_\triangle(x_j),\ j = 0, \ldots, n. \tag{4.44}$$

Diese M_j heißen Momente. Wir berechnen Splines, die (4.41), (4.42) oder (4.43) erfüllen mit Hilfe der Momente. Da $S_\triangle$ kubisch in $[x_i, x_{i+1}]$ ist, folgt dass $S''_\triangle$ linear in $[x_i, x_{i+1}]$ und es gilt (2-Punkte Formel der Geradengleichung), dass

$$S''_\triangle(x) = M_j \frac{x_{j+1} - x}{h_{j+1}} + M_{j+1} \frac{x - x_j}{h_{j+1}}, \ x \in [x_j, x_{j+1}]. \tag{4.45}$$

Integration ergibt

$$\begin{aligned} S'_\triangle(x) &= -M_j \frac{(x_{j+1} - x)^2}{2h_{j+1}} + M_{j+1} \frac{(x - x_j)^2}{2h_{j+1}} + \alpha_j, \\ S_\triangle(x) &= M_j \frac{(x_{j+1} - x)^3}{6h_{j+1}} + M_{j+1} \frac{(x - x_j)^3}{6h_{j+1}} + \alpha_j (x - x_j) + \beta_j. \end{aligned}$$

Die Interpolationsbedingungen ergeben, dass

$$\begin{aligned} S_\Delta(x_j) &= f_j, \\ \Longrightarrow M_j \frac{h_{j+1}^2}{6} + \beta_j &= f_j \Longrightarrow \beta_j = f_j - M_j \frac{h_{j+1}^2}{6}, \\ S_\Delta(x_{j+1}) &= f_{j+1} \\ \Longrightarrow M_{j+1} \frac{h_{j+1}^2}{6} + \alpha_j h_{j+1} + \beta_j &= f_{j+1} \Longrightarrow \alpha_j = \frac{f_{j+1} - f_j}{h_{j+1}} - \frac{h_{j+1}}{6}(M_{j+1} - M_j), \end{aligned}$$

und damit

$$S_\Delta(x) = a_j + b_j(x - x_j) + c_j(x - x_j)^2 + d_j(x - x_j)^3, \tag{4.46}$$

mit

$$\begin{aligned} a_j &= f_j, c_j = \frac{M_j}{2}, b_j = -\frac{M_j h_{j+1}}{2} + \alpha_j = S'_\Delta(x_j) \\ &= \frac{f_{j+1} - f_j}{h_{j+1}} - \frac{(2M_j + M_{j+1})h_{j+1}}{6}, d_j = \frac{M_{j+1} - M_j}{6h_{j+1}} = \frac{S'''_\Delta(x_j^+)}{6}. \end{aligned} \tag{4.47}$$

Wir brauchen also nur die M_j zu berechnen. Wir haben gefordert, dass $S'_\Delta(x)$ an den Knoten $x = x_j, \ j = 1, \ldots, n-1$ stetig sein soll. Das ergibt $n - 1$ Gleichungen. Es gilt

$$\begin{aligned} S'_\Delta(x) &= -M_j \frac{(x_{j+1} - x)^2}{2h_{j+1}} + M_{j+1} \frac{(x - x_j)^2}{2h_{j+1}} + \frac{f_{j+1} - f_j}{h_{j+1}} \\ &\quad - \frac{h_{j+1}}{6}(M_{j+1} - M_j) \end{aligned} \tag{4.48}$$

wenn man die Werte einsetzt. Damit ergibt sich für $j = 1, \ldots, n-1$

$$\begin{aligned} S'_\Delta(x_j^-) &= \frac{f_j - f_{j-1}}{h_j} + \frac{h_j}{3} M_j + \frac{h_j}{6} M_{j-1}, \\ S'_\Delta(x_j^+) &= \frac{f_{j+1} - f_j}{h_{j+1}} - \frac{h_{j+1}}{3} M_j - \frac{h_{j+1}}{6} M_{j+1}. \end{aligned} \tag{4.49}$$

Gleichsetzen wegen der Stetigkeit impliziert, dass

$$\begin{aligned} \frac{h_j}{6} M_{j-1} + \frac{h_j + h_{j+1}}{3} M_j + \frac{h_{j+1}}{6} M_{j+1} &= \frac{f_{j+1} - f_j}{h_{j+1}} - \frac{f_j - f_{j-1}}{h_j} \\ &= f[x_j, x_{j+1}] - f[x_{j-1}, x_j], \end{aligned} \tag{4.50}$$

für $j = 1, \ldots, n-1$. Das sind $n-1$ Gleichungen für $M_0, \ldots, M_n$.
Die zwei weiteren notwendigen Bedingungen erhält man aus (4.41), (4.42), (4.43). Aus (4.41)

$$S''_\Delta(a) = M_0 = M_n = S''_\Delta(b) = 0.$$

Aus (4.42)

$$\begin{aligned} S''_\Delta(a) = S''_\Delta(b) &\implies M_0 = M_n. \\ S'_\Delta(a) = S'_\Delta(b) &\implies \frac{h_n}{6} M_{n-1} + \frac{h_n + h_1}{3} M_n + \frac{h_1}{6} M_1 \\ & \quad = \frac{f_1 - f_n}{h_1} - \frac{f_n - f_{n-1}}{h_n}. \end{aligned}$$

Aus $f_n = f_0$ und mit der Setzung $h_{n+1} := h_1$, $M_{n+1} := M_1$, $f_{n+1} := f_1$ erhält man wieder Gleichung (4.50).
Aus (4.43)

$$\begin{aligned} S'_\Delta(a) = f'_0 &\implies \frac{h_1}{3} M_0 + \frac{h_1}{6} M_1 = \frac{f_1 - f_0}{h_1} - f'_0, \\ S'_\Delta(b) = f'_n &\implies \frac{h_n}{6} M_{n-1} + \frac{h_n}{3} M_n = f'_n - \frac{f_n - f_{n-1}}{h_n}. \end{aligned}$$

Führe nun folgende Abkürzungen ein:

$$\lambda_j := \frac{h_{j+1}}{h_j + h_{j+1}}, \; \mu_j := 1 - \lambda_j = \frac{h_j}{h_j + h_{j+1}}$$

Dann erhält man das System von linearen Gleichungen

$$\mu_j M_{j-1} + 2M_j + \lambda_j M_{j+1} = 6f[x_{j-1}, x_j, x_{j+1}] =: d_j \tag{4.51}$$

d.h. die folgenden Gleichungssysteme:

$$\begin{bmatrix} 2 & \lambda_0 & & & \\ \mu_1 & 2 & \lambda_1 & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \lambda_{n-1} \\ & & & \mu_n & 2 \end{bmatrix} \begin{bmatrix} M_0 \\ \vdots \\ \vdots \\ \vdots \\ M_n \end{bmatrix} = \begin{bmatrix} d_0 \\ \vdots \\ \vdots \\ \vdots \\ d_n \end{bmatrix} \tag{4.52}$$

im Fall von Bedingung (4.41), wobei $\lambda_0 := 0, d_0 = 0, \mu_n := 0, d_n := 0$ gesetzt wurde.

$$\begin{bmatrix} 2 & \lambda_1 & & & & \mu_1 \\ \mu_2 & 2 & \lambda_2 & & & \\ & \mu_3 & \ddots & \ddots & & \\ & & \ddots & \ddots & \ddots & \\ & & & & & \lambda_{n-1} \\ \lambda_n & & & & \mu_n & 2 \end{bmatrix} \begin{bmatrix} M_1 \\ \vdots \\ \vdots \\ \vdots \\ M_n \end{bmatrix} = \begin{bmatrix} d_1 \\ \vdots \\ \vdots \\ \vdots \\ d_n \end{bmatrix} \tag{4.53}$$

im Fall von Bedingung (4.42), wobei

$$\lambda_n := \frac{h_1}{h_n + h_1}, \mu_n = 1 - \lambda_n, d_n = 6f[x_{n-1}, x_n, x_1]$$

Im Fall von Bedingung (4.43) erhält man wiederum System (4.52) nur mit

$$\lambda_0 = 1, \mu_n = 1, d_0 = \tfrac{6}{h_1}\left(f[x_0, x_1] - f_0'\right), d_n = \tfrac{6}{h_n}\left(f_n' - f[x_{n-1}, x_n]\right).$$

Für alle 3 Gleichungssysteme gilt $\lambda_i \geq 0, \mu_i \geq 0, \lambda_i + \mu_i \leqslant 1$ und das Theorem:

Theorem 55 *Die Matrizen der Systeme (4.52), (4.53) sind für jede Zerlegung Δ von $[a, b]$ nichtsingulär.*

Beweis. Sei

$$A = \begin{bmatrix} 2 & \lambda_0 & & \\ \mu_1 & \ddots & \ddots & \\ & \ddots & \ddots & \lambda_{n-1} \\ & & \mu_n & 2 \end{bmatrix}.$$

Sei $x, y \in \mathbb{R}^{n+1}$ mit $Ax = y$ und setze $\lambda_n = \mu_0 = 0$. Dann gilt

$$\max_i |x_i| \leq \max_i |y_i| \quad \text{(A ist monoton)}.$$

Denn sei r der Index für den $\max\limits_i |x_i|$ angenommen wird.
Dann gilt

$$\begin{aligned} \mu_r x_{r-1} + 2x_r + \lambda_r x_{r+1} &= y_r \\ \Longrightarrow \max_i |y_i| \geq |y_r| &\geq 2|x_r| - \mu_r |x_{r-1}| - \lambda_r |x_{r+1}| \\ &\geq 2|x_r| - (\mu_r + \lambda_r)|x_r| \\ &\geq |x_r| = \max_i |x_i|. \end{aligned}$$

Angenommen A singulär $\Longrightarrow x \neq 0$ mit $Ax = 0$.
Monotonie führt sofort zum Widerspruch. Der Beweis für die anderen Systeme geht analog. □

Wie gut approximiert so eine Spline-Funktion?

Sei $\Delta_m = \{a = x_0^{(m)} < x_1^{(m)} < \cdots < x_{n_m}^{(m)} = b\}$ eine Folge von Unterteilungen von $[a, b]$ und sei $\sigma_m = \max\limits_i (x_{i+1}^{(m)} - x_i^{(m)})$.

Theorem 56 *Sei $f \in \mathcal{C}^4[a, b]$ und $|f^{(4)}(x)| \leq L \; \forall x \in [a, b], \Delta_m$ wie oben mit $\sup\limits_{m,j} \frac{\sigma_m}{x_{j+1}^{(m)} - x_j^{(m)}} \leq \kappa < \infty$. Seien S_{Δ_m} die zu f gehörigen Splines, die die Interpolationsbedingungen*

$$\begin{aligned} S_{\Delta_m}(\zeta) &= f(\zeta) && \text{für } \zeta \in \Delta_m \\ S'_{\Delta_m}(x) &= f'(x) && \text{für } x = a, b \end{aligned} \tag{4.54}$$

erfüllen. Dann gibt es von Δ_m unabhängige Konstanten $C_i \leq 2$, so dass für alle $x \in [a, b]$

$$\left| f^{(i)}(x) - S_{\Delta_m}^{(i)}(x) \right| \leq C_i L \kappa \sigma_m^{4-i}, \; i = 0, 1, 2, 3. \tag{4.55}$$

Beweis. Betrachte feste Zerlegung Δ von $[a, b]$. Die Momente M_j erfüllen für den Fall den wir hier haben, das Gleichungssystem

$$\begin{bmatrix} 2 & \lambda_0 & & \\ \mu_1 & 2 & \ddots & \\ & \ddots & \ddots & \lambda_{n-1} \\ & & \mu_n & 2 \end{bmatrix} \begin{bmatrix} M_0 \\ \vdots \\ \vdots \\ M_n \end{bmatrix} \begin{bmatrix} d_0 \\ \vdots \\ \vdots \\ d_n \end{bmatrix}$$

$$\lambda_0 = \mu_n = 1, \lambda_j = \frac{h_{j+1}}{h_j + h_{j+1}}, \mu_j = 1 - \lambda_j,\ j = 1, \ldots, n-1$$

$$\begin{aligned} d_j &= 6f[x_{j-1}, x_j, x_{j+1}],\ j = 1, \ldots, n-1 \\ d_0 &= \frac{6}{h_1}\left(f[x_0, x_1] - f_0'\right), \\ d_n &= \frac{6}{h_n}\left(f_n' - f[x_{n-1}, x_n]\right). \end{aligned}$$

Definiere

$$F = \begin{bmatrix} f''(x_o) \\ \vdots \\ f''(x_n) \end{bmatrix}, r = d - AF, M = \begin{bmatrix} M_0 \\ \vdots \\ M_n \end{bmatrix}.$$

Dann gilt $r_0 = d_0 - 2f''(x_0) - f''(x_1)$. Schätze zuerst $\|r_0\|_\infty$ ab. Taylorentwicklung um x_0 liefert

$$\begin{aligned} r_0 &= \frac{6}{h_1}\left(f[x_0, x_1] - f_0'\right) - 2f''(x_0) - f''(x_1) \\ &= \frac{6}{h_1}\left(f'(x_0) + \frac{h_1}{2}f''(x_0) + \frac{h_1^2}{6}f'''(x_0) + \frac{h_1^3}{24}f^{(4)}(\tau_1) - f'(x_0)\right) \\ &\quad -2f''(x_0) - \left(f''(x_0) + h_1 f'''(x_0) + \frac{h_1^2}{2}f^{(4)}(\tau_2)\right) \\ &= \frac{h_1^2}{4}f^{(4)}(\tau_1) - \frac{h_1^2}{2}f^{(4)}(\tau_2),\ \text{für } \tau_1, \tau_2 \in [x_0, x_1] \end{aligned}$$

$$\Longrightarrow |r_0| \le \frac{3}{4}L\sigma_m^2. \tag{4.56}$$

Analog gilt für $r_n = d_n - f''(x_{n-1}) - 2f''(x_n)$ die Abschätzung

$$|r_n| \le \frac{3}{4}L\sigma_m^2 \tag{4.57}$$

und für $j = 1, \ldots, n-1$ mit Taylorentwicklung um x_j.

$$\begin{aligned} r_j &= d_j - \mu_j f''(x_{j-1}) - 2f''(x_j) - \lambda_j f''(x_{j+1}) \\ &= 6f[x_{j-1}, x_j, x_{j+1}] - \frac{h_j}{h_j + h_{j+1}}f''(x_{j-1}) - 2f''(x_j) - \frac{h_{j+1}}{h_j + h_{j+1}}f''(x_{j+1}) \\ &= \frac{1}{h_j + h_{j+1}}\left\{6\left(f'(x_j) + \frac{h_{j+1}}{2}f''(x_j) + \frac{h_{j+1}^2}{6}f'''(x_j) + \frac{h_{j+1}^3}{24}f^{(4)}(\tau_{j_1})\right.\right. \end{aligned}$$

$$
\begin{aligned}
& -f'(x_j) + \frac{h_j}{2} f''(x_j) - \frac{h_j^2}{6} f'''(x_j) + \frac{h_j^3}{24} f^{(4)}(\tau_{j_2}) \Big) \\
& - h_j \left(f''(x_j) - h_j f'''(x_j) + \frac{h_j^2}{2} f^{(4)}(\tau_{j_3}) \right) \\
& - 2 f''(x_j)(h_j + h_{j+1}) \\
& - h_{j+1} \left(f''(x_j) + h_{j+1} f'''(x_j) + \frac{h_{j+1}^2}{2} f^{(4)}(\tau_{j_4}) \right) \Big\} \\
= \quad & \frac{1}{h_j + h_{j+1}} \left(\frac{h_{j+1}^3}{4} f^{(4)}(\tau_{j_1}) + \frac{h_j^3}{4} f^{(4)}(\tau_{j_2}) - \frac{h_j^3}{2} f^{(4)}(\tau_{j_3}) - \frac{h_{j+1}^3}{2} f^{(4)}(\tau_{j_4}) \right),
\end{aligned}
$$

mit $\tau_{j_i} \in [x_{j-1}, x_{j+1}],\ i = 1, 2, 3, 4.$

$$\Longrightarrow |r_j| \leq \frac{3}{4} L \frac{h_{j+1}^3 + h_j^3}{h_j + h_{j+1}} \leq \frac{3}{4} L \sigma_m^2.$$

Insgesamt gilt

$$\|r\|_\infty \leq \frac{3}{4} L \sigma_m^2 \Longrightarrow$$

$$\|A(M - F)\|_\infty \leq \frac{3}{4} L \sigma_m^2 \Longrightarrow \|M - F\|_\infty \leq \frac{3}{4} L \sigma_m^2, \tag{4.58}$$

da A monoton.

Sei $x \in [x_{j-1}, x_j]$. Dann folgt

$$
\begin{aligned}
S_\Delta''' - f'''(x) &= \frac{M_j - M_{j-1}}{h_j} - f'''(x) \\
&= \frac{M_j - f''(x_j)}{h_j} - \frac{M_{j-1} - f''(x_{j-1})}{h_j} \\
&\quad + \frac{[f''(x_j) - f''(x)] - [f''(x_{j-1}) - f''(x)]}{h_j} - f'''(x).
\end{aligned}
$$

Taylorentwicklung um x und (4.58) $\Longrightarrow$

$$
\begin{aligned}
|S_\Delta'''(x) - f'''(x)| \leq \; & \frac{3}{2} L \frac{\sigma_m^2}{h_j} + \frac{1}{h_j} \Big| (x_j - x) f'''(x) + \frac{(x_j - x)^2}{2} f^{(4)}(\eta_1) - (x_{j-1} - x) f'''(x) \\
& - \frac{(x_{j-1} - x)^2}{2} f^{(4)}(\eta_2) - h_j f'''(x) \Big|
\end{aligned}
$$

$$\begin{aligned}
&\leq \frac{3}{2} L \frac{\sigma_m^2}{h_j} + \frac{1}{2} \frac{\sigma_m^2}{h_j} L, \ \tau_1, \tau_2 \in [x_{j-1}, x_j] \\
&\leq 2L\kappa\sigma_m, \ \text{wegen Vor. } \frac{\sigma_m}{h_j} \leq \kappa.
\end{aligned}$$

Hierbei wurde noch $\max_{x \in [x_{j-1}, x_j]} \{(x_j - x)^2 + (x - x_{j-1})^2\} = h_j^2$ benutzt. Nun gibt es für jedes $x \in (a, b)$ ein x_j mit

$$\begin{aligned}
|x_j - x| &\leq \frac{1}{2}\sigma_m \\
\Longrightarrow f''(x) - S''_\Delta(x) &= f''(x_j) - S''_\Delta(x_j) + \int_{x_j}^{x} (f'''(t) - S'''_\Delta(t))\, dt.
\end{aligned}$$

Mit der Abschätzung für $f''' - S'''_\Delta$ folgt, dass

$$\begin{aligned}
|f''(x) - S''_\Delta(x)| &\leq \frac{3}{4} L\sigma_m^2 + \frac{1}{2}\sigma_m \cdot 2L\kappa\sigma_m \\
&\leq \frac{7}{4} L\kappa\sigma_m^2, \ \text{da } \kappa \geq 1.
\end{aligned}$$

Damit haben wir die Abschätzung für $f'' - S''_\Delta$.
Es gilt $f(x_{j-1}) = S_\Delta(x_{j-1}), f(x_j) = S_\Delta(x_j)$. Außer den Randpunkten $\alpha_0 = a, \alpha_{n+1} = b$ gibt es also noch den Satz von Rolle n Stellen $\alpha_j \in (x_{j-1}, x_j)$ mit

$$f'(\alpha_j) = S'_\Delta(\alpha_j) \quad j = 0, \ldots, n+1.$$

Es gilt also für jedes $x \in [a, b]$ ein α_j mit $|\alpha_j - x| < \sigma_m$. Daher gilt für alle $x \in [a, b]$:

$$\begin{aligned}
f'(x) - S'_\Delta(x) &= \int_{\alpha_j(x)}^{x} (f''(t) - S''_\Delta(t))\, dt \\
\Longrightarrow |f'(x) - S'_\Delta(x)| &\leq \frac{7}{4} L\kappa\sigma_m^2 \cdot \sigma_m = \frac{7}{4} L\kappa\sigma_m^3.
\end{aligned}$$

Analog folgt für $x \in [a, b]$ (hier ist wieder $|x - x_j| \leq \frac{1}{2}\sigma_m$), dass

$$f(x) - S_\Delta(x) = \int_{x_j}^{x} f'(t) - S'_\Delta(t) dt$$

und damit

$$|f(x) - S_\Delta(x)| \leq \frac{7}{4} L\kappa\sigma_m^3 \frac{1}{2}\sigma_m = \frac{7}{8} L\kappa\sigma_m^4.$$

□

Es gäbe noch viel mehr zu Splines zu sagen, dazu fehlt hier die Zeit.

Kapitel 5

Numerische Integration

Eine weitere Technik, die wir zur Entwicklung von Verfahren höherer Ordnung benötigen, sind numerische Integrationsmethoden, oft auch Quadraturmethoden genannt.

Aufgabe der numerischen Integration (Quadratur) ist die Berechnung von

$$\int_a^b f(x)dx, \quad a, b < \infty.$$

Typischerweise ist f nicht direkt integrierbar, und wir müssen numerische Methoden verwenden. Die Idee ist nun die folgende. Approximiere $f(x)$ durch eine Funktion, die einfach zu integrieren ist; wie Polynome, trigonometrische Polynome, Splines, rationale Funktionen, Exponentialsummen, oder ähnliches, und verwende das Integral dieser approxierenden Funktion als Approximation an das gesuchte Integral.

5.1 Newton–Cotes Formeln

Die Idee der Newton–Cotes[1] Formeln, ist $f(x)$ durch ein Polynom zu interpolieren und dieses zu integrieren. Bilde äquidistante Unterteilung von $[a, b], \quad x_i = a + ih, \quad i = 0, \cdots, n$, mit $h = \frac{b-a}{n}$ $n \in \mathbb{N}$ und bilde das Interpolationspolynom in Π_n, das

$$P_n(x_i) = f(x_i) \quad i = 0, \cdots, n \tag{5.1}$$

[1]R. Cotes, Englischer Mathematiker, 1682 – 1716

n	σ_i	ns	Fehler	Name
1	1 1	2	$\frac{h^3}{12}f^{(2)}(\xi)$	Trapezregel
2	1 4 1	6	$\frac{h^5}{90}f^{(4)}(\xi)$	Simpsonregel
3	1 3 3 1	8	$\frac{h^5 3}{80}f^{(4)}(\xi)$	3/8 Regel
4	7 32 12 32 7	90	$\frac{h^7 178}{945}f^{(6)}(\xi)$	Milne–Regel
5	19 75 50 50 75 19	288	$\frac{h^7 275}{12096}f^{(6)}(\xi)$	–
6	41 216 27 272 27 216 41	840	$\frac{h^9 9}{1400}f^{(8)}(\xi)$	Weddle–Regel

Tabelle 5.1: Newton–Cotes–Formeln

erfüllt. Aus Theorem 35 folgt, dass

$$P_n(x) = \sum_{i=0}^{n} f_i L_i(x). \tag{5.2}$$

Damit ergibt sich

$$\begin{aligned}
\int_a^b P_n(x)dx &= \sum_{i=0}^{n} f_i \int_a^b L_i(x)dx \qquad \text{mit } x = a + hs \\
&= h \sum_{i=0}^{n} f_i \int_0^n \Pi_{\substack{k=0\\k\neq i}}^{n} \frac{s-k}{i-k} ds \\
&= h \sum_{i=0}^{n} f_i \alpha_i = \frac{b-a}{n} \sum_{i=0}^{n} \sigma_i f_i \quad \text{mit } \sigma_i \in \mathbb{Z}.
\end{aligned} \tag{5.3}$$

Die Gewichte α_i sind unabhängig von f_i, a, b jedoch abhängig von n und liegen tabelliert (siehe Tabelle 5.1) vor. Diese Formeln heißen *Newton–Cotes–Formeln.*

Proposition 57 *Für den Approximationsfehler bei der numerischen Inte-*

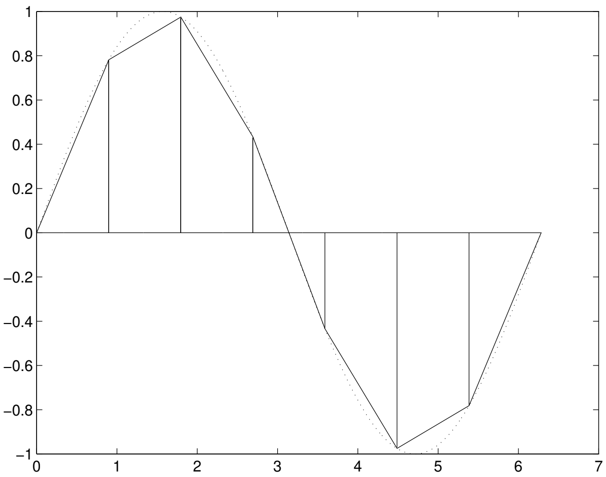
gration mit Hilfe von Newton-Cotes Formeln gilt:

$$\int_a^b (P_n(x) - f(x))\, dx = h^{p+1}\kappa \cdot f^{(p)}(\xi), \xi \in (a,b). \tag{5.4}$$

Beweis. Der Beweis folgt sofort aus Theorem 45. $\square$

Es macht keinen Sinn höheren Polynomgrad zu verwenden, weil dann negative Gewichte in den Formeln auftreten und damit die Gefahr der Auslöschung auftritt. Aber das ist auch nicht notwendig, denn um die schlechten globalen Approximationseigenschaften von Polynomen zu vermeiden werden die Regeln stückweise auf Teilintervalle angewendet und dann aufsummiert.

Beispiel 58 (Summierte Trapezregel: stückweise lineare Interpolation)



Verwende Teilintervalle $[x_i, x_{i+1}], x_i = a + ih, \quad i = 0, \cdots, N, \quad h = \frac{b-a}{N}$. Dann ist die summierte Trapezregel gegeben durch

$$T(h) \quad := \quad \sum_{i=0}^{N-1} \frac{h}{2} [f(x_i) + f(x_{i+1})] \tag{5.5}$$

$$= \quad h\left[\frac{f(a)}{2} + f(a+h) + \cdots + f(b-h) + \frac{f(b)}{2}\right].$$

Für jedes Teilintervall ist der Fehler

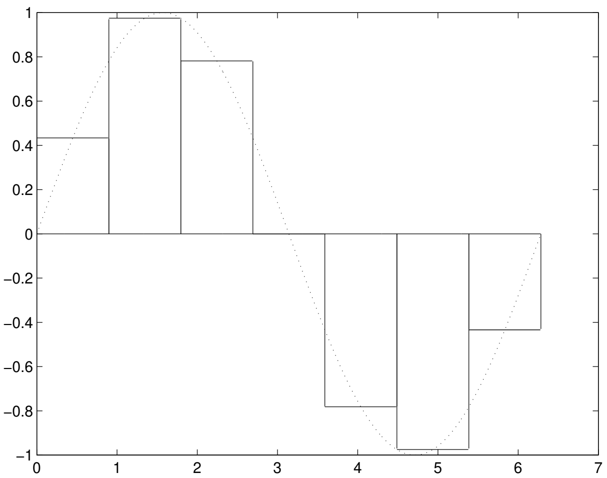
$$\frac{h}{2}\left[f(x_i) - f(x_{i+1})\right] - \int\limits_{x_i}^{x_{i+1}} f(x)dx = \frac{h^3}{12} f^{(2)}(\xi_i) \text{ für } \xi_i \in (x_i, x_{i+1}). \tag{5.6}$$

Insgesamt ergibt sich also

$$\begin{aligned} \left| T(h) - \int\limits_a^b f(x)dx \right| &= \left| \sum_{i=0}^{N-1} \frac{h^3}{12} f^{(2)}(\xi_i) \right| \qquad (5.7) \\ &\leq \frac{h^3}{12} N \cdot \underbrace{\sup_{\xi\in[a,b]} f''(\xi)}_{=M} = \frac{b-a}{12} h^2 M. \end{aligned}$$

Damit geht der Fehler mit h^2 gegen 0. Man nennt dies ein Verfahren 2. Ordnung.

Beispiel 59 (Summierte Mittelpunktregel)



Beispiel 60 *Sei N gerade. Verwende die Simpsonregel auf $[x_{2i}, x_{2i+1}, x_{2i+2}]$, $i = 0, \cdots, \frac{N}{2} - 1$. Der Näherungswert auf jedem der Teilintervalle ist*

$$\frac{h}{3}\left[f(x_{2i}) + 4f(x_{2i+1}) + f(x_{2i+2})\right].$$

Summation ergibt

$$\begin{aligned} S(h) &= \frac{h}{3}[(f(a) + 4f(a+b) + 2f(a+2h) + 4f(a+3h)+ \\ &\quad +2f(b-2h) + 4f(b-h) + f(b)] \end{aligned} \tag{5.8}$$

und als Fehler erhalten wir

$$\begin{aligned} \left| S(h) - \int_a^b f(x)dx \right| &\leq \frac{h^5}{90} \cdot \frac{N}{2} \cdot \underbrace{\sup_{\xi \in [a,b]} f^{(4)}(\xi)}_{=M} \\ &= \frac{b-a}{180} h^4 \cdot M, \end{aligned} \tag{5.9}$$

also ist das ein Verfahren 4. Ordnung.

Weitere Quadraturformeln erhält man durch andere Approximationsaufgaben für $f(x)$. Dies sind z.B. die *Gauss–Quadratur*[2] für Integrale der Form

$$\int_a^b w(x)f(x)dx \approx \sum_{i=1}^n w_i f(x_i).$$

wobei w_i, f_i so gewählt werden, dass der Fehler

$$\int_a^b w(x)f(x)dx - \sum_{i=1}^n w_i f(x).$$

für Polynome möglichst hohen Grades verschwindet.

[2] C.F. Gauß, Deutscher Mathematiker, 1777 –1855

5.2 Extrapolation

Eine weitere Technik zur numerischen Integration, die wir auch bei der Lösung von Differentialgleichungen verwenden, ist die Extrapolation. Die Idee ist dabei, Integrationsformeln für unterschiedliche Werte h_i von h zu verwenden und damit Approximationen an das Integral Q_i auszurechen, dann die Werte (h_i, Q_i) durch ein Interpolationspolynom zu interpolieren und dieses mit Hilfe des Neville-Aitken Schemas bei $h = 0$ auszuwerten. Besonderes gut funktioniert dies bei der Trapezsumme. Dies liegt an der Euler-Maclaurin-Formel[3].

Theorem 61 *Für $f \in \mathcal{C}^{2m+2}[a,b]$ hat $T(h)$ die Entwicklung*

$$T(h) = \tau_0 + \tau_1 h^2 + \tau_2 h^4 + \cdots + \tau_m h^{2m} + \alpha_{m+1}(h) h^{2m+2} \tag{5.10}$$

mit $\tau_0 := \int\limits_a^b f(x)\,dx$. Dabei sind die τ_i von h unabhängige Konstanten und $|\alpha_{m+1}(h)| \leq M$ für alle $h = \frac{b-a}{n}, n \in \mathbb{N}$.

Bevor wir dieses Theorem beweisen können brauchen brauchen wir zuerst einige Eigenschaften der *Bernoulli–Polynome*, die wie folgt rekursiv definiert sind.

Definition 62 *Die Polynome $B_k(x), k = 0, 1, 2, \cdots,$ die durch die rekursive Folge*

$$\begin{array}{l} B_0(x) = 1, B_k'(x) = k \cdot B_{k-1}(x) \quad k \geq 1, \\ \int\limits_0^1 B_k(x)\,dx = 0, \ k \geq 1 \end{array} \tag{5.11}$$

definiert sind, heißen Bernoulli–Polynome.

Eigenschaften der Bernoulli–Polynome:

$$B_k(x) = a_k + k \int_0^x B_{k-1}(t)\,dt \quad k \geq 1 \tag{5.12}$$

[3] C. Maclaurin, Englischer Mathematiker, 1698 – 1746

wobei a_k so zu bestimmen ist, daß $\int\limits_0^1 B_k(x)\,dx = 0$. Man erhält sofort

$$\begin{array}{l} B_1(x) = x - \frac{1}{2}, B_2 = x^2 - x + \frac{1}{6}, B_3(x) = x^3 - \frac{3}{2}x^2 + \frac{1}{2}x \\ B_4(x) = x^4 - 2x^3 + x^2 - \frac{1}{30}, \cdots \end{array} \tag{5.13}$$

und es gilt das folgende Lemma.

Lemma 63

i) $B_k(0) = B_k(1)$ *für* $k \geq 2$.

ii) *Die Polynome* $P_k(x) = B_k(\frac{1}{2}+x)$ *sind für gerades* k *gerade, d.h.* $P(x) = P(-x)$ *und für ungerades* k *ungerade Polynome, d.h.* $P(x) = -P(-x)$ *in* x.

iii) $B_{2k+1}(0) = 0$ *für alle* $k \geq 1$.

Beweis.

i) Aus (5.12) folgt, dass

$$B_k(1) - B_k(0) = k \int\limits_0^1 B_{k-1}(t)\,dt = 0 \text{ für } k \geq 2.$$

ii) Das Polynom $P_0(x) \equiv 1$ ist gerade. Sei $P_{2k}(x)$ für $k \geq 0$ ein gerades Polynom. Zeige, dass $P_{2k+2}(x)$ wieder gerade ist. Es gilt

$$P'_{2k+1}(x) = B'_{2k+1}(\frac{1}{2} + x) = (2k+1)B_{2k}(\frac{1}{2} + x) = (2k+1)P_{2k}(x).$$

Also

$$\int\limits_{-\frac{1}{2}}^{\frac{1}{2}} P'_{2k+1}(x)dx = 0.$$

Da $P_{2k}(x)$ gerade ist, folgt dass

$$q(x) := (2k+1) \int\limits_0^x P_{2k}(t)\,dt$$

ein ungerades Polynom ist und es gilt

$$\int_{-\frac{1}{2}}^{\frac{1}{2}} q(x)dx = -\int_{+\frac{1}{2}}^{-\frac{1}{2}} q(-x)\,dx = \int_{\frac{1}{2}}^{-\frac{1}{2}} q(x)dx = -\int_{-\frac{1}{2}}^{\frac{1}{2}} q(x)dx = 0.$$

Weiterhin ist $q'(x) = (2k+1)P_{2k}(x)$ und $\int\limits_{-\frac{1}{2}}^{\frac{1}{2}} q(x)dx = 0$. Damit folgt, dass $q(x) = P_{2k+1}(x)$ ungerade ist und da

$$P_{2k+2}(x) = (2k+2)\int_0^x P_{2k+1}(t)\,dt + a,$$

so ist P_{2k+2} wieder gerade, also gilt ii).

iii) Für $k \geq 1$ gilt

$$B_{2k+1}(1) = B_{2k+1}(0) = P_{2k+1}(-\frac{1}{2}) = -P_{2k+1}(\frac{1}{2}) = -B_{2k+1}(1)$$

und damit $B_{2k+1}(0) = B_{2k+1}(1) = 0$.

□

Die Werte der Bernoulli-Polynome bei 0, $\beta_k = (-1)^{k+1}B_{2k}(0)$ heißen *Bernoullische Zahlen*. Es gilt $\beta_1 = \frac{1}{6}, \beta_2 = \frac{1}{30}, \beta_3 = \frac{1}{42}, \beta_4 = \frac{1}{30}$.

Wir definieren nun neue Funktionen

$$S_k(x) := B_k(x-i) \text{ für } x \in E_i := \{x | i < x < i+1\},\ i \leq k. \tag{5.14}$$

Diese Funktionen sind periodisch mit Periode 1 und durch die folgende Rekursion bestimmt:

$$S_k(x) = S_k(0) + k\int_0^x S_{k-1}(t)\,dt,\ k \geq 2. \tag{5.15}$$

S_1 ist stückweise linear mit Sprungstellen bei $i \in \mathbb{N}$ und Nullstellen bei $i + \frac{1}{2}$ und es gilt wegen Lemma 63

$$\begin{aligned} &B_k(0) = S_k(0) = S_k(1) = S_k(2) = \cdots \\ &S_{2k+1}(0) = 0, S_{2k}(0) = (-1)^{k+1}\beta_k. \end{aligned} \tag{5.16}$$

Damit können wir nun die Euler–MacLaurin'sche Summenformel beweisen.
Beweis. (Theorem 61) Betrachte $g(t) := f(a + th)$. Wir formen das Integral

$$\int\limits_0^n S_1(t) g'(t)\, dt$$

durch partielle Integration. $S_1(x)$ ist stückweise linear und

$$\begin{aligned} \int\limits_0^1 S_1(t) g'(t)\, dt &= \int\limits_0^1 B_1(t) g'(t)\, dt \\ &= B_1(t) g(t)|_0^1 - \int\limits_0^1 B_0(t) g(t)\, dt = \frac{1}{2}\left[g(1) + g(0)\right] - \int\limits_0^1 g(t)\, dt. \end{aligned}$$

Analog folgt

$$\begin{aligned} \int\limits_i^{i+1} S_1(t) g'(t)\, dt &= \int\limits_0^1 B_1(t) g'(t+i)\, dt \\ &= \frac{1}{2}\left[g(i+1) + g(i)\right] - \int\limits_i^{i+1} g(t)\, dt,\ i = 0, \ldots, n-1 \end{aligned}$$

und damit

$$\frac{g(0)}{2} + g(1) + \cdots + g(n-1) + \frac{g(n)}{2} - \int\limits_0^n g(t)\, dt = \int\limits_0^n S_1(t) g'(t)\, dt \tag{5.17}$$

Für die rechte Seite gilt

$$\int\limits_0^n S_1(t) g'(t)\, dt = \frac{1}{2!}\left(S_2(t) g'(t)|_0^n - \int\limits_0^n S_2(t) g''(t)\, dt \right)$$

$$\begin{aligned}
&= \frac{\beta_1}{2!}\left[g'(n)-g'(0)\right]-\frac{1}{2!}\int\limits_0^n S_2(t)g''(t)\,dt \\
&= \frac{\beta_1}{2!}\left[g'(n)-g'(0)\right]-\frac{1}{3!}S_3(t)g''(t)|_0^n+\frac{1}{3!}\int\limits_0^n S_3(t)g^{(3)}(t)\,dt \\
&= \frac{\beta_1}{2!}\left[g'(n)-g'(0)\right]+\frac{1}{3!}\int\limits_0^n S_3(t)g^{(3)}\,dt.
\end{aligned}$$

Zweimalige partielle Integration ergibt

$$\begin{aligned}
&\tfrac{g(0)}{2}+g(1)+\cdots+g(n-1)+\tfrac{g(n)}{2}-\int\limits_0^n g(t)\,dt \\
&=\sum_{k=1}^m(-1)^{k+1}\tfrac{\beta_k}{(2k!)}\left[g^{(2k-1)}(n)-g^{(2k-1)}(0)\right]+R_{m+1}
\end{aligned} \tag{5.18}$$

mit

$$\begin{aligned}
R_{m+1} &= \frac{1}{(2m+1)!}\int\limits_0^n S_{2m+1}(t)g^{(2m+1)}(t)\,dt \\
&= \frac{1}{(2m+2)!}\left(S_{2m+2}(t)-S_{2m+2}(0)\right)g^{(2m+1)}(t)\Big|_0^n \\
&\quad -\frac{1}{(2m+2)!}\int\limits_0^n\left[S_{2m+2}(t)-S_{2m+2}(0)\right]g^{(2m+2)}(t)\,dt \\
&= -\frac{1}{(2m+2)!}\int\limits_0^n\left[S_{2m+2}(t)-S_{2m+2}(0)\right]g^{(2m+2)}(t)\,dt.
\end{aligned} \tag{5.19}$$

Es ist $\int\limits_0^n g(t)\,dt=\frac{1}{h}\int\limits_a^b f(x)dx$ und $g^{(k)}(t)=h^kf^{(k)}(a+th)\quad k=0,1,2,\cdots$ und damit folgt

$$\begin{aligned}
\frac{g(0)}{2} &+ g(1)+\cdots+g(n-1)+\frac{g(n)}{2} \\
&= \frac{f(a)}{2}+f(a+h)+\cdots+f(b-h)+\frac{f(b)}{2} \\
&= \frac{1}{h}T(h),
\end{aligned}$$

also

$$T(h) = \tau_0 + \tau_1 h^2 + \cdots + \tau_m h^{2m} + \alpha_{m+1}(h) h^{2m+2} \tag{5.20}$$

mit $\tau_0 = \int\limits_a^b f(x)dx$. Weiter folgt, dass

$$\tau_k = \frac{(-1)^{k+1}}{(2k)!} \beta_k \left[f^{(2k-1)}(b) - f^{(2k-1)}(a) \right] \tag{5.21}$$

und dass

$$\alpha_{m+1} = \frac{-1}{(2m+2)!} \int\limits_a^b f^{(2m+2)}(x) \left[S_{2m+2}\left(\frac{x-a}{n} \right) - S_{2m+2}(0) \right] dx. \tag{5.22}$$

Da S_{2m+2} stetig und periodisch ist folgt, dass es eine von h unabhängige Schranke für $|\alpha_{m+1}^{(n)}|$ gibt. □

Die Euler–MacLaurin–Formel läßt sich nun zur Extrapolation verwenden.

Wenn man das Restglied vernachlässigt, ist $T(h)$ ein Polynom in h^2, das für $h = 0$ den Wert $\tau_0 = \int\limits_a^b f(x)\, dx$ liefert.

Bestimme daher für verschiedene Schrittweiten $h_0 = b - a, h_1 = \frac{h_0}{n_1}, \cdots, h_m = \frac{h_0}{n_m},\ n_i \in \mathbb{N}$, die zugehörige Trapezsumme

$$T_{i,0} = T(h_i),\ i = 0, \cdots, m \tag{5.23}$$

und dann durch Interpolation das Polynom

$$\tilde{T}_{m,m}(h) = a_0 + a_1 h^2 + \cdots + a_m h^{2m}, \tag{5.24}$$

das die Interpolationsbedingungen

$$\tilde{T}_{m,m}(h_i) = T(h_i),\ i = 0, \ldots, m \tag{5.25}$$

erfüllt. Werte dann dieses Polynom bei $h = 0$ mit dem Neville–Aitken Schema aus. Dies liefert i.a. eine sehr gute Näherung für das Integral. Sei $\tilde{T}_{i,k}(h)$ das Polynom vom Grad k in h^2, dass

$$\tilde{T}_{i,k}(h_j) = T_{j,0},\ j = i-k, \cdots, i \tag{5.26}$$

erfüllt. Damit gilt für die extrapolieren Werte $T_{i,k} := \tilde{T}_{i,k}(0)$.

$$T_{i,k} = T_{i,k-1} + \frac{T_{i,k-1} - T_{i-1,k-1}}{\left(\frac{h_{i-k}}{h_i}\right)^2 - 1}, \; 1 \le k \le i \le m. \tag{5.27}$$

Als Extrapolationsfolgen verwendet man

a) $h_0 = b - a, h_1 = \frac{h_0}{2}, h_2 = \frac{h_1}{2}, h_3 = \frac{h_2}{2}, \cdots$ (Romberg Folge)

b) $h_0 = b - a, h_1 = \frac{h_0}{2}, h_2 = \frac{h_0}{3}, h_3 = \frac{h_1}{2}, h_4 = \frac{h_2}{2}, h_5 = \frac{h_3}{2}, \cdots$ (Bulirschfolge).

Dabei ist b) oft besser, da der Rechenaufwand weniger schnell ansteigt, denn vorherige Werte können verwendet werden.

Wie gut ist nun die Extrapolation?

Aus der Fehlerbetrachtung für die Quadraturformel folgt, dass es $\xi \in [a, b]$ gibt, so dass gilt

$$T_{mm} - \int_a^b f(x)\,dx = -\frac{1}{(2m+2)!} f^{(2m+2)}(\xi) \int_a^b K(x)\,dx \tag{5.28}$$

mit

$$K(x) = \sum_{i=0}^{m} c_{m_i} h_i^{2m+2} \left[S_{2m+2}\left(\frac{x-a}{h_i}\right) - S_{2m+2}(0) \right]. \tag{5.29}$$

Damit ergibt sich die Fehlerabschätzung:

$$T_{mm} - \int_a^b f(x)dx = (b-a) h_0^2 \cdots h_m^2 \frac{\beta_{m+1}}{(2m+1)!} f^{(2m+2)}(\xi). \tag{5.30}$$

Kapitel 6

Verfahren höherer Ordnung

Die Fehler- und Konvergenzanalyse für das Euler-Verfahren hat uns gezeigt, dass wir auf Grund der Rundungsfehler die Schrittweite nicht beliebig klein machen können. Wir brauchen daher Verfahren, die einen lokalen Diskretisierungsfehler (und damit auch einen globalen Fehler) haben, der eine Ordung $\mathcal{O}(h^p)$ mit möglichst großem p hat.

Definition 64 *Ein Einschrittverfahren der Form*

$$\begin{aligned} u_0 &= y_0, \\ u_{i+1} &= u_i + h\Phi(t_i, u_i, h, f) \end{aligned} \tag{6.1}$$

zur Lösung der Anfangswertaufgabe

$$y' = f(t, y), \;\; y(t_0) = y_0 \tag{6.2}$$

auf einem Intervall $\mathbb{I} = [t_0, t_0 + a]$, heißt konsistent von der Ordnung p, wenn für alle genügend oft differenzierbaren Funktionen f und für alle $\hat{t} \in \mathbb{I}$ und alle z, für den lokalen Diskretisierungsfehler gilt, dass

$$\tau(\hat{t}, z, h, f) = \mathcal{O}(h^{p+1}). \tag{6.3}$$

6.1 Einfache Verfahren höherer Ordnung

Einfache Verfahren höherer Ordnung erhält man mit dem Ansatz

$$\Phi(t, z, h, f) = a_1 f(t, z) + a_2 f(t + p_1 h, z + p_2 h f(t, z))$$

Man bestimmt nun a_1, a_2, p_1, p_2 so, dass die Taylor-Entwicklung von τ mit möglichst hoher $h-$Potenz beginnt.

Die Taylorentwicklung von $\Phi(t, z, h, f)$ ergibt

$$\Phi(t, z, h, f) = (a_1 + a_2)f(t, z) + a_2 h[p_1 f_t(t, z) + p_2 f_z(t, z)f(t, z)] + \mathcal{O}(h^2).$$

Um ein Verfahren 2-ter Ordnung zu erhalten, ergibt sich daher:

$$a_1 + a_2 = 1 \quad a_2 p_1 = \frac{1}{2} \quad a_2 p_2 = \frac{1}{2}.$$

Lösungen sind:

1) $a_1 = \frac{1}{2}, a_2 = \frac{1}{2}, p_1 = p_2 = 1$.Das ergibt das *Verfahren von Heun* mit

$$\Phi(t, z, h, f) = \frac{1}{2}[f(t, z) + f(t + h, z + hf(t, z))],$$

welches 2 Auswertungen von f pro Schritt hat.

2) $a_1 = 0, a_2 = 1, p_1 = p_2 = \frac{1}{2}$. Das ist das *modifizierte Euler-Verfahren* von Collatz mit

$$\Phi(t, z, h, f) = f(t + \frac{h}{2}, z + \frac{h}{2}f(t, z)),$$

das ebenfalls 2 Auswertungen von f pro Schritt hat.

Beispiel 65 *Wir wollen uns die drei Verfahren, Euler, modifizierter Euler und Heun im Vergleich anschauen, siehe Abbildung 65.*

Dazu betrachten wir die Anfangswertaufgabe

$$y' = -2xy^2, y(0) = 1$$

mit der exakten Lösung $y(x) = 1/(x^2 + 1)$. Mit der Methode von Euler erhalten wir die in der folgenden Tabelle zusammengestellten Näherungswerte y_k für verschiedene Schrittweiten h an gleichen Stellen x_k sowie die zugehörigen Fehler $e_k := y(x_k) - y_k$. Der Fehler nimmt etwa proportional zur Schrittweite h ab.

Euler-Verfahren für $y' = -2xy^2,\ y(0) = 1.$

		$h = 0.1$		$h = 0.01$		$h = 0.001$	
x_k	$y(x_k)$	y_k	e_k	y_k	e_k	y_k	e_k
0	1.00000	1.00000	–	1.00000	–	1.00000	–
0.1	0.99010	1.00000	−0.00990	0.99107	−0.00097	0.99020	−0.00010
0.2	0.96154	0.98000	−0.01846	0.96330	−0.00176	0.96171	−0.00018
0.3	0.91743	0.94158	−0.02415	0.91969	−0.00226	0.91766	−0.00022
0.4	0.86207	0.88839	−0.02632	0.86448	−0.00242	0.86231	−0.00024
0.5	0.80000	0.82525	−0.02525	0.80229	−0.00229	0.80023	−0.00023
0.6	0.73529	0.75715	−0.02185	0.73727	−0.00198	0.73549	−0.00020

Mit dem Verfahren von Heun und dem modifiziertem Euler-Verfahren für Schrittweiten $h = 0.1$ und $h = 0.05$ erhalten wir die folgenden Ergebnisse. Die Resultate zeigen die Fehlerordnung 2.

Modifiziertes Euler-Verfahren (y_k) und Verfahren von Heun ($\hat{y}_k$) für $y' = -2xy^2,\ y(0) = 1$.

	$h = 0.1$		$h = 0.05$		$h = 0.1$		$h = 0.05$	
x_k	y_k	e_k	y_k	e_k	$\hat{y}_k$	$\hat{e}_k$	$\hat{y}_k$	$\hat{e}_k$
0	1.00000	–	1.00000	–	1.00000	–	1.00000	–
0.1	0.99000	0.00010	0.99007	0.00002	0.99000	0.00010	0.99009	0.00001
0.2	0.96118	0.00036	0.96145	0.00009	0.96137	0.00017	0.96152	0.00002
0.3	0.91674	0.00069	0.91727	0.00016	0.91725	0.00019	0.91742	0.00001
0.4	0.86110	0.00096	0.86184	0.00023	0.86195	0.00011	0.86208	−0.00001
0.5	0.79889	0.00111	0.79974	0.00026	0.80003	−0.00003	0.80004	−0.00004
0.6	0.73418	0.00111	0.73503	0.00026	0.73553	−0.00023	0.73538	−0.00009
0.7	0.67014	0.00100	0.67091	0.00023	0.67159	−0.00045	0.67128	−0.00014
0.8	0.60895	0.00080	0.60957	0.00018	0.61040	−0.00064	0.60993	−0.00018
0.9	0.55191	0.00058	0.55236	0.00013	0.55329	−0.00080	0.55270	−0.00021
1.0	0.49964	0.00036	0.49992	0.00008	0.50092	−0.00092	0.50024	−0.00024

Man kann noch weitere Methoden auf ähnliche Weise konstruieren, wir wollen jedoch systematisch vorgehen.

Wir betrachten wieder eine Anfangswertaufgabe der Form

$$y' = f(t, y),\ y(t_0) = y_0 \tag{6.4}$$

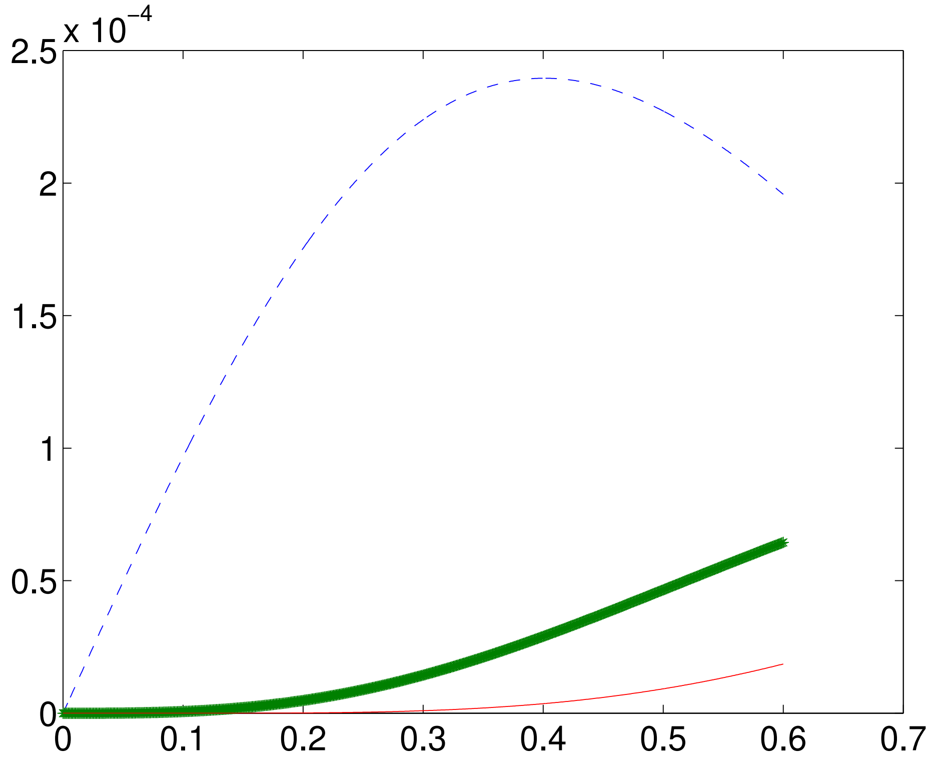


Abbildung 6.1: Vergleich Euler (-), Mod. Euler (.) und Heun (*)

auf einem Intervall $I = [t_0, t_0 + a]$. Dabei sei f eine stetige Funktion der Form

$$f : G \subset \mathbb{R}^{n+1} \to \mathbb{R}^n,$$

die einer Lipschitzbedingung genüge. Integration der Differentialgleichung ergibt für $y(t)$

$$y(t) = y(t_0) + \int_{t_0}^{t} f(s, y(s))\, ds. \tag{6.5}$$

Entsprechend erhalten wir für Teilintervalle

$$y_{j+1} = y_j + \int_{t_j}^{t_{j+1}} f(s, y(s))\, ds. \tag{6.6}$$

Das Integral

$$\int_{t_j}^{t_{j+1}} f(s, y(s))\, ds$$

ersetzen wir nun durch die Quadraturformel

$$h_j \sum_{l=1}^{m} \gamma_l f(s_l, y(s_l)), \qquad s_l \in [t_j, t_{j+1}],$$

wobei wir jetzt auch variable Schrittweiten h_j zulassen. Dabei muss $y(s_l)$ noch ausgerechnet werden. Dazu benötigen wir eine Näherung für $f(s_l, y(s_l))$, da wir $y(s_l)$ ja nicht kennen.

Dieser allgemeine Ansatz führt auf explizite Runge[1]–Kutta[2] Verfahren. Wir setzen

$$f(s_l, y(s_l)) \approx k_l(t_j, y_j)$$

mit $s_1 = t_j$ und $s_l = t_j + \alpha_l h_j$, wobei für α_l gilt:

$$\alpha_l = \sum_{r=1}^{l-1} \beta_{l,r}.$$

Daraus konstruieren wir ein Gleichungssystem der Form

$$\begin{aligned}
k_1(t_j, y_j) &= f(t_j, y_j), \\
k_2(t_j, y_j) &= f(t_j + \alpha_2 h_j, y_j + h_j \beta_{2,1} k_1(t_j, y_j)), \\
k_3(t_j, y_j) &= f(t_j + \alpha_3 h_j, y_j + h_j(\beta_{3,1} k_1(t_j, y_j) + \beta_{3,2} k_2(t_j, y_j))), \\
&\vdots \\
k_m(t_j, y_j) &= f(t_j + \alpha_m h_j, y_j + h_j(\beta_{m,1} k_1(t_j, y_j) + \ldots \\
&\qquad \ldots + \beta_{m,m-1} k_{m-1}(t_j, y_j))),
\end{aligned} \tag{6.7}$$

und damit erhalten wir für Approximationen u_j das Verfahren:

$$u_{j+1} = u_j + h_j(\gamma_1 k_1(t_j, u_j) + \ldots + \gamma_m k_m(t_j, u_j)). \tag{6.8}$$

---

[1] C.D. Runge, deutscher Mathematiker, 1856–1927
[2] M.W. Kutta, deutscher Mathematiker, 1867–1944

Theorem 66 *Ein Verfahren der Form (6.8) mit* Stufenwerten k_i *wie in (6.7) heißt* m–stufiges Runge–Kutta Verfahren. *Es wird üblicherweise durch eine* Butcher–Tabelle *der Form*

$$\begin{array}{c|cccc} 0 & & & & \\ \alpha_2 & \beta_{2,1} & & & \\ \alpha_3 & \beta_{3,1} & \ddots & & \\ \vdots & \vdots & & & \\ \alpha_m & \beta_{m,1} & \dots & \beta_{m,m-1} & \\ \hline & \gamma_1 & \dots & \gamma_{m-1} & \gamma_m \end{array} \tag{6.9}$$

dargestellt.

Beispiel 67 *Als spezielle Runge-Kutta Verfahren erhalten wir die folgenden Methoden:*

1. *Eulerverfahren, 1. Ordnung:*

$$m = 1, \quad \gamma_1 = 1$$

$$u_{j+1} = u_j + h_j f(t_j, u_j). \tag{6.10}$$

2. *Modifiziertes Eulerverfahren, (Mittelpunktsregel für Quadratur), 2. Ordnung:*

$$m = 2, \quad \gamma_1 = 0, \quad \gamma_2 = 1, \quad \beta_{2,1} = \alpha_2 = \frac{1}{2}. \tag{6.11}$$

3. *Verfahren von Heun, (Trapezregel für Quadratur), 2. Ordnung:*

$$m = 2, \quad \gamma_1 = \gamma_2 = \frac{1}{2}, \quad \beta_{2,1} = \alpha_2 = 1 \tag{6.12}$$

4. *Runge–Verfahren, 3. Ordnung:*

$$m = 3 \qquad \begin{array}{c|ccc} 0 & & & \\ 1/2 & 1/2 & & \\ 1 & 0 & 1 & \\ \hline & 0 & 0 & 1 \end{array} \tag{6.13}$$

5. *Klassisches Runge–Kutta Verfahren, (Simpsonregel für Quadratur), 4. Ordnung:*

$$m = 4 \qquad \begin{array}{c|cccc} 0 & & & & \\ 1/2 & 1/2 & & & \\ 1/2 & 0 & 1/2 & & \\ 1 & 0 & 0 & 1 & \\ \hline & 1/6 & 1/3 & 1/3 & 1/6 \end{array} \tag{6.14}$$

Was für Bedingungen müssen an die Koeffizienten gestellt werden?

Es soll ja für die k_r gelten, dass

$$\begin{aligned} k_r(t_j, y_j) &= f(t_j + \alpha_r h_j, y(t_j) + h_j(\beta_{r,1}k_1 + \ldots + \beta_{r,r-1}k_{r-1})) \\ &\approx f(t_j + \alpha_r h_j, y(t_j + \alpha_r h_j)) \\ &= f(t_j + \alpha_r h_j, y(t_j) + \alpha_r h_j y'(t_j) + \mathcal{O}(h_j^2)), \end{aligned}$$

und die Approximation soll mindestens $\mathcal{O}(h_j^2)$ sein. Für die Koeffizienten α_i, β_i und γ_i gelten dann folgende Beziehungen:

$$\beta_{r,1} + \ldots + \beta_{r,r-1} = \alpha_r, \tag{6.15}$$

$$\gamma_1 + \ldots + \gamma_m = 1. \tag{6.16}$$

Wir erhalten den folgenden Konsistenzsatz:

Theorem 68 *Ein explizites Runge–Kutta Verfahren der Form (6.9), welches die Bedingungen (6.15) und (6.16) erfüllt, ist konsistent.*

Beweis. Laut Definition gilt für den lokalen Fehler, dass $\frac{1}{h_j}\tau(\hat{t}, z, h_j, f) = \Delta(\hat{t}, z, h_j, f) - \Phi(\hat{t}, z, h_j, f)$ und damit erhalten wir unter Verwendung von (6.16) die folgende Abschätzung:

$$\begin{aligned} |\frac{1}{h_j}\tau(\hat{t}, z, h_j, f)| &= |\Delta(\hat{t}, z, h_j, f) - \Phi(\hat{t}, z, h_j, f)| \\ &= \left| \frac{y(\hat{t} + h_j) - y(\hat{t})}{h_j} - \sum_{r=1}^{m} \gamma_r k_r(\hat{t}, z) + f(\hat{t}, z) - f(\hat{t}, z) \right| \end{aligned}$$

$$
\begin{aligned}
&\leq \left|\frac{y(\hat{t}+h_j)-y(\hat{t})}{h_j}-y'(\hat{t})\right| + \left|\sum_{r=1}^{m}\gamma_r(k_r(\hat{t},z)-f(\hat{t},z))\right| \\
&\leq \left|\frac{y(\hat{t}+h_j)-y(\hat{t})}{h_j}-y'(\hat{t})\right| \\
&\qquad + \sum_{r=1}^{m}\gamma_r\left|f(\hat{t},z)-f(\hat{t}+\alpha_r h_j, y(\hat{t})+h_j\theta)\right| .
\end{aligned}
$$

Da die beiden Summanden gegen Null gehen, wenn h_j gegen Null geht, folgt die Behauptung. □

Um die Ordnung der Runge–Kutta Verfahren zu bestimmen, muss man nach Taylor entwickeln und anschließend ein nichtlineares Gleichungssystem für die Koeffizienten lösen. Die Anzahl der Parameter und damit der Funktionsauswertungen wächst sehr schnell, wie die folgende Tabelle zeigt:

Ordnung p	1	2	3	4	5	6	7	8
# Parameter	1	2	4	8	17	37	85	200

(6.17)

Die folgende Tabelle zeigt, wie groß die maximale Ordnung ist, die mit einem m–stufigen Verfahren für ein bestimmtes m erreicht werden kann.

Stufe m	1	2	3	4	5	6	7	8	9	≥ 9
Ordnung $p(m)$	1	2	3	4	4	5	6	6	7	$< m-2$

(6.18)

Beispiel 69 *Im folgenden sei $m = 4$ und $p = 4$. Dann gibt es z.B. die folgenden 4-stufigen Verfahren:*

- *Klassisches Runge–Kutta Verfahren*

$$
\begin{array}{c|cccc}
0 & & & & \\
1/2 & 1/2 & & & \\
1/2 & 0 & 1/2 & & \\
1 & 0 & 0 & 1 & \\
\hline
 & 1/6 & 1/3 & 1/3 & 1/6
\end{array}
\tag{6.19}
$$

- 3/8–*Regel*

$$\begin{array}{c|cccc} 0 & & & & \\ 1/3 & 1/3 & & & \\ 2/3 & -1/3 & 1 & & \\ 1 & 1 & -1 & 1 & \\ \hline & 1/8 & 3/8 & 3/8 & 1/8 \end{array} \tag{6.20}$$

Das klassische Runge–Kutta Verfahren (6.19) ist unter allen expliziten Runge-Kutta Verfahren der Ordnung 4 *dasjenige mit den wenigsten Parametern. Es braucht daher auch unter all diesen Methoden die wenigsten Funktionsauswertungen.*

Weitere Verfahren lassen sich durch Kombination von Runge–Kutta–Verfahren und Taylorentwicklung erzeugen. Diese heißen *Runge–Kutta–Fehlberg Verfahren.*

6.2 Implizite Runge-Kutta Formeln

In einigen Problemen läßt es sich nicht vermeiden, kompliziertere Verfahren zu verwenden. Bei den Runge-Kutta-Verfahren lassen sich weitere Verfahren sehr einfach entwickeln. Definiere hier statt (6.7)

$$k_r(t_j, y_j) = f(t_j + \alpha_r h_j, y_j + h_j(\beta_{r,1}k_1 + \ldots + \beta_{r,m}k_m)), \tag{6.21}$$

für $r = 1, \ldots, m$ und damit

$$u_{j+1} = u_j + h_j(\gamma_1 k_1(t_j, u_j) + \ldots + \gamma_m k_m(t_j, u_j)).$$

Mit $\beta_{r,l} = 0$ für $l \geq r$ ist k_r in (6.21) explizit aus $k_1, \ldots, k_{r-1}$ berechenbar, sonst ist (6.21) implizit in den Unbekannten $k_1, \ldots, k_m$. Man beachte: diese Formeln definieren kein implizites Verfahren, d.h. wir erhalten immer noch u_{i+1} explizit aus u_i, nur die Berechnung der k_r ist implizit.

Auch implizite Runge-Kutta-Verfahren werden wieder durch Butcher–Tabellen angegeben:

$$\begin{array}{c|ccc} \alpha_1 & \beta_{11} & \ldots & \beta_{1m} \\ \alpha_2 & \beta_{21} & \ldots & \beta_{2m} \\ \vdots & \vdots & \ldots & \vdots \\ \alpha_m & \beta_{m1} & \ldots & \beta_{mm} \\ \hline & \gamma_1 & \ldots & \gamma_m \end{array}$$

Bei den impliziten Verfahren sind drei verschiedene Formeltypen von besonderem Interesse.

1. Gauß-Form : Alle Parameter $\alpha_j, \beta_{jl}, \gamma_j$ beliebig wählbar

2. Radau-Form : Entweder $\alpha_1 = \beta_{11} = \beta_{12} = \ldots = \beta_{1m} = 0$ oder $\alpha_m = 1$ und $\beta_{1m} = \beta_{2m} = \ldots = \beta_{mm} = 0$.

3. Lobatto-Form : $\alpha_1 = \beta_{11} = \beta_{12} = \ldots = \beta_{1m} = \beta_{2m} = \ldots = \beta_{mm} = 0, \alpha_m = 1$.

Die Namen der drei Typen von Verfahren leiten sich aus dem Umstand her, dass sie im Spezialfall einer von y unabhängigen Funktion f in die gleichnamigen Quadraturformeln übergehen. Die Radau-Formeln haben den Vorteil, dass entweder k_1 oder k_m explizit berechnet werden kann, bei den Lobatto-Formeln können sogar k_1 und k_m explizit berechnet werden, wodurch die Zahl der in jedem Schritt zu lösenden impliziten Gleichungen verringert wird. Dafür muß man eine geringere Konsistenzordnung bei gleicher Stufenzahl in Kauf nehmen.

Beispiel 70 *Wir geben nun einige spezielle implizite Formeln der genannten Typen an.*

Gauß-Form, 2. Ordnung : $m = 1, p = 2,$

$$\begin{array}{c|c} \frac{1}{2} & \frac{1}{2} \\ \hline & 1 \end{array}$$

Mit diesen Koeffizienten erhält man das folgende Verfahren:

$$\begin{aligned} k_1 &= f(t_j + \frac{h_j}{2}, u_j + \frac{h_j}{2} k_1), \\ u_{j+1} &= u_j + h_j k_1. \end{aligned}$$

Gauß-Form, 4. Ordnung : $m = 2, p = 4,$

$$\begin{array}{c|cc} \frac{(3-\sqrt{3})}{6} & \frac{1}{4} & \frac{(3-2\sqrt{3})}{12} \\ \frac{(3+\sqrt{3})}{6} & \frac{(3+2\sqrt{3})}{12} & \frac{1}{4} \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}$$

Radau-Form, 1. Ordnung, (Euler Verfahren): $m = 1, p = 1$

$$\begin{array}{c|c} 0 & \\ \hline & 1 \end{array}$$

Radau-Form, 3. Ordnung: $m = 2, p = 3$

$$\begin{array}{c|cc} 0 & 0 & 0 \\ \frac{2}{3} & \frac{1}{3} & \frac{1}{3} \\ \hline & \frac{1}{4} & \frac{3}{4} \end{array} \qquad \begin{array}{c|cc} \frac{1}{3} & \frac{1}{3} & 0 \\ 1 & 1 & 0 \\ \hline & \frac{3}{4} & \frac{1}{4} \end{array}$$

Lobatto-Form, 2. Ordnung : $m = 2, p = 2$

$$\begin{array}{c|cc} 0 & 0 & 0 \\ 1 & 1 & 0 \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}$$

Setze man diese Werte ein, so erhält man die (explizite) Formel

$$u_{j+1} = u_j + \frac{h_j}{2}(f(t_j, u_j) + f(t_j + h_j, u_j + f(t_j, u_j))).$$

Lobatto-Form, 4. Ordnung : $m = 3, p = 4$

$$\begin{array}{c|ccc} 0 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{4} & \frac{1}{4} & 0 \\ 1 & 0 & 1 & 0 \\ \hline & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \end{array}$$

Zur Berechnung der k_r muss im allgemeinen ein nichtlineares Gleichungssystem gelöst werden. Die Existenz der Lösung erhalten wir aus dem folgenden Theorem. Zur numerischen Lösung kommen wir später.

Theorem 71 *Die Funktion f genüge für alle $(t, y_1), (t, y_2) \in \mathbb{I} \times \mathbb{R}^n$ der Lipschitz-Bedingung*

$$\|f(t, y_1) - f(t, y_2)\| \leq L\|y_1 - y_2\|.$$

Dann existiert für alle Schrittweiten h mit

$$q = L \cdot h \cdot max_{j=1,\dots,m}(\sum_{k=1}^{m} |\beta_{jk}|) < 1$$

und alle $(t, y) \in \mathbb{I} \times \mathbb{R}^n$ eine eindeutig bestimmte Lösung $k_1(t, y), \dots, k_m(t, y)$ von (6.21).

Beweis. Der Beweis folgt aus dem Banachschen Fixpunktsatz. □

Bemerkung 72 *i) Wesentlicher Vorteil der impliziten Runge-Kutta Verfahren gegenüber den expliziten Verfahren ist der größere Stabilitätsbereich der Methode.*

ii) Die Lipschitz-Bedingung läßt sich auf eine lokale Lipschitz-Bedingung abschwächen, die nur in einem Schlauch um die Lösung gilt.

iii) Um $q < 1$ zu garantieren, muss h genügend klein sein. Das führt oft dazu, dass der erhöhte Rechenaufwand die hohe Konsistenzordnung wieder aufhebt.

6.3 Schrittweitensteuerung:

So wie wir die Runge–Kutta Verfahren bisher entwickelt haben, ist nicht klar wie die Schrittweite h_j in jedem Schritt zu wählen ist. Es ist natürlich sinnvoll, die Schrittweite an den Lösungsverlauf anzupassen, d.h. bei fast linearem Lösungsverhalten sollten wir große Schritte machen (um den Rechenaufwand zu minimieren), und wenn die Lösung stark oszilliert entsprechend kleine Schritte (um die gewünschte Genauigkeit zu bekommen). Dies ist die Idee der *Schrittweitensteuerung.*

Ziel: Wähle die Schrittweite h_j von t_j auf t_{j+1} möglichst groß, aber so, dass der lokale Fehler unterhalb einer bestimmten Toleranzgrenze liegt.

Idee: Verwende eine Schätzung für den Fehler. Wird der geschätzte Fehler zu groß, so verkleinere die Schrittweite. Wird der geschätzte Fehler dagegen sehr klein, so vergrößere die Schrittweite.

Wie können wir den Fehler schätzen?

Seien $\Phi_p(t, y, h, f)$ und $\Phi_{p+1}(t, y, h, f)$ die Inkrementfunktionen zweier Verfahren der Ordnung p und $p+1$. Dann gilt für die zugehörigen Diskretisierungsfehler τ_p bzw. τ_{p+1}:

$$\begin{aligned}
\frac{1}{h_j}\tau_p(t_j, z, h_j, f) &= \Delta(t_j, z, h_j, f) - \Phi_p(t_j, z, h_j, f), \\
\frac{1}{h_j}\tau_{p+1}(t_j, z, h_j, f) &= \Delta(t_j, z, h_j, f) - \Phi_{p+1}(t_j, z, h_j, f).
\end{aligned}$$

Daraus folgt dann

$$\begin{aligned}
\tfrac{1}{h_j}\tau_p &= \Phi_{p+1} - \Phi_p + \tfrac{1}{h_j}\tau_{p+1} \\
&\approx \Phi_{p+1} - \Phi_p.
\end{aligned} \tag{6.22}$$

Diese Schätzung des lokalen Fehlers wird nun zur Schrittweitensteuerung verwendet. Ideal wäre natürlich, wenn die beiden Verfahren so gewählt wären, dass der Rechenaufwand im wesentlichen der gleiche ist, als wenn wir nur das Verfahren der Ordnung $p+1$ verwenden. Dies ist die Idee der *eingebetteten Runge-Kutta Verfahren.*

Die allgemeine Vorgehensweise läßt sich nun wie folgt beschreiben:

Schrittweitensteuerung

```
 Gegeben   t_0, u_0, a
           Anfangsschrittweite h < a
           Toleranz ε
           Intervall [t_0, t_0 + a]
           2 Inkrementfunktionen Φ_p, Φ_{p+1} zu Verfahren der Ordnung p, p+1
i = 0
WHILE t_i < t_0 + a AND t_i + h > t_i
        IF t_i + h > t_0 + a
            h = t_0 + a − t_i
        END IF
        Φ_1 = Φ_p(t_i, u_i, h, f)
        Φ_2 = Φ_{p+1}(t_i, u_i, h, f)
        τ = ‖Φ_2 − Φ_1‖
        n = ‖u_i‖ + 1
        IF τ ⩽ εn
            t_{i+1} = t_i + h
            u_{i+1} = u_i + hΦ_2 (u_{i+1} = u_i + hΦ_2)
            i = i + 1
        END IF
        IF τ > εn
            h = h (εn/τ)^{1/p}
        END IF
END WHILE
```

Man kann auf diese Weise auch Unstetigkeiten in den Lösungen aufspüren, wenn die Schrittweite immer kleiner wird.

Beispiel 73 *Wir verwenden ein Verfahren der Ordnung $p = 2$ zum Rechnen und ein Verfahren der Ordnung $p = 3$ zum Schätzen des Fehlers mit folgenden Tabellen für die Runge–Kutta Verfahren:*

$$p = 2 \qquad \begin{array}{c|cc} 0 & & \\ 1 & 1 & \\ \hline & 1/2 & 1/2 \end{array}$$

$$p = 3 \qquad \begin{array}{c|ccc} 0 & & & \\ 1 & 1 & & \\ 1/2 & 1/4 & 1/4 & \\ \hline & 1/6 & 1/6 & 2/3 \end{array}$$

Wir erhalten so für die diskreten Werte u bzw. û der beiden Verfahren folgende Terme:

$$\begin{aligned} u_{j+1} &= u_j + \frac{h_j}{2}(k_1 + k_2), \\ k_1 &= f(t_j, u_j), \\ k_2 &= f(t_j + h_j, u_j + h_j k_1) \end{aligned} \tag{6.23}$$

und

$$\begin{aligned} \hat{u}_{j+1} &= \hat{u}_j + \frac{h_j}{6}(\hat{k}_1 + \hat{k}_2 + 4\hat{k}_3), \\ \hat{k}_1 &= f(t_j, \hat{u}_j), \\ \hat{k}_2 &= f(t_j + h_j, \hat{u}_j + h_j \hat{k}_1), \\ \hat{k}_3 &= f\left(t_j + \frac{h_j}{2}, \hat{u}_j + \frac{h_j}{4}(\hat{k}_1 + \hat{k}_2)\right). \end{aligned} \tag{6.24}$$

Da wir vom gleichen Startwert starten, gilt dann $\hat{k}_1 = k_1$ und $\hat{k}_2 = k_2$ und damit brauchen wir diese Werte nur einmal zu berechnen.

Beispiel 74 *Das eingebettete Runge-Kutta-Fehlberg-4(5)-Verfahren hat die Form*

$$\begin{array}{c|cccccc} 0 & & & & & & \\ \frac{1}{4} & \frac{1}{4} & & & & & \\ \frac{3}{8} & \frac{3}{32} & \frac{9}{32} & & & & \\ \frac{12}{13} & \frac{1932}{2197} & -\frac{7200}{2197} & \frac{7296}{2197} & & & \\ 1 & \frac{439}{216} & -8 & \frac{3680}{513} & -\frac{845}{4104} & & \\ \frac{1}{2} & -\frac{8}{27} & 2 & -\frac{3544}{2565} & \frac{1859}{4104} & -\frac{11}{40} & \\ \hline y & \frac{25}{216} & 0 & \frac{1408}{2565} & \frac{2197}{4104} & -\frac{1}{5} & 0 \quad \textit{Ordnung 4} \\ \hline \hat{y} & \frac{16}{135} & 0 & \frac{6656}{12825} & \frac{28561}{56430} & -\frac{9}{50} & \frac{2}{55} \quad \textit{Ordnung 5} \end{array}$$

Dieses Verfahren braucht keine zusätzlichen Funktionsauswertungen.

Eine andere exzellente Methode ist das Verfahren von Dormand-Prince-5(4) (DOPRI5) mit den folgenden Koeffizienten

$$
\begin{array}{c|ccccccc}
0 & & & & & & & \\
\frac{1}{5} & \frac{1}{5} & & & & & & \\
\frac{3}{10} & \frac{3}{40} & \frac{9}{40} & & & & & \\
\frac{4}{5} & \frac{44}{45} & -\frac{56}{15} & \frac{32}{9} & & & & \\
\frac{8}{9} & \frac{19372}{6561} & -\frac{25360}{2187} & \frac{64448}{6561} & -\frac{212}{729} & & & \\
1 & \frac{9017}{3168} & -\frac{355}{33} & \frac{46732}{5247} & \frac{49}{176} & -\frac{5103}{18656} & & \\
1 & \frac{35}{384} & 0 & \frac{500}{1113} & \frac{125}{192} & -\frac{2187}{6784} & \frac{11}{84} & \\
\hline
y_1 & \frac{35}{384} & 0 & \frac{500}{1113} & \frac{125}{192} & -\frac{2187}{6784} & \frac{11}{84} & 0 \\
\hline
\hat{y}_1 & \frac{5179}{57600} & 0 & \frac{7571}{16695} & \frac{393}{640} & -\frac{92097}{339200} & \frac{187}{2100} & \frac{1}{40}
\end{array}
$$

Kapitel 7

Lösung von linearen Gleichungssystemen.

Fast alle Praxisprobleme, ob in der Physik, der Chemie oder den Ingenieurwissenschaften führen letztendlich auf die Lösung eines linearen Gleichungssystems welches typischerweise heute eine große Zahl von Variablen hat. Ein Beispiel haben wir schon bei der Spline-Interpolation gesehen. Ein anderes Beispiel tritt offensichtlich bei der Lösung von impliziten Runge–Kutta–Verfahren auf. Wir betrachten die folgende Aufgabe:

Berechne eine Lösung von $Ax = b$, wobei $A \in \mathbb{R}^{n,n}$ und $b \in \mathbb{R}^n$ ist. Die meisten der Ergebnisse und Algorithmen gelten auch über $\mathbb{C}$ oder sogar anderen Körpern.

7.1 Normen und andere Grundlagen

Zuerst betrachten wir als Wiederholung ein paar Grundlagen.

Eine *Vektornorm* auf $\mathbb{R}^n$ ist eine Funktion $f : \mathbb{R}^n \longrightarrow \mathbb{R}$, welche die folgenden Bedingungen erfüllt.

$$\begin{array}{rcll} f(x) & \geq & 0 & \forall x \in \mathbb{R}^n \text{ (dabei ist } f(x) = 0 \Longleftrightarrow x = 0), \\ f(x+y) & \leq & f(x) + f(y) & \forall x, y \in \mathbb{R}^n, \\ f(\alpha x) & = & |\alpha| f(x) & \forall \alpha \in \mathbb{R}, x \in \mathbb{R}^n, \end{array} \tag{7.1}$$

Wir schreiben $f(x) = \|x\|$.

Beispiel 75

$$\begin{aligned}
\|x\|_p &= (|x_1|^p + \cdots |x_n|^p)^{\frac{1}{p}},\ p \geq 1, \\
\|x\|_1 &= |x_1| + |x_2| + \cdots + |x_n|, \\
\|x\|_2 &= (|x_1|^2 + |x_2|^2 + \cdots + |x_n|^2)^{\frac{1}{2}} = (x^T x)^{\frac{1}{2}}, \\
\|x\|_\infty &= \max_{1 \leq k \leq n} |x_k|.
\end{aligned}$$

Proposition 76 *Es gilt:*

$$\begin{aligned}
|x^T y| &\leq \|x\|_p \|y\|_q,\ \text{für } \frac{1}{p} + \frac{1}{q} = 1 \ \textit{(Höldersche Ungleichung)} \\
|x^T y| &\leq \|x\|_2 \|y\|_2 \quad \textit{(Cauchy–Schwarz–Ungleichung)} \\
\|x\|_2 &\leq \|x\|_1 \leq \sqrt{n} \|x\|_2 \\
\|x\|_\infty &\leq \|x\|_2 \leq \sqrt{n} \|x\|_\infty \\
\|x\|_\infty &\leq \|x\|_1 \leq n \|x\|_\infty
\end{aligned}$$

Eine Abbildung $f : \mathbb{R}^{m,n} \longrightarrow \mathbb{R}$ heißt *Matrix Norm*, falls

$$\begin{aligned}
f(A) &\geq 0 && \forall A \in \mathbb{R}^{m,n} \ (\text{ dabei ist } f(A) = 0 \Longleftrightarrow A = 0) \\
f(A+B) &\leq f(A) + f(B) && \forall A, B \in \mathbb{R}^{m,n} \\
f(\alpha A) &= |\alpha| f(A) && \forall \alpha \in \mathbb{R}, A \in \mathbb{R}^{m,n}
\end{aligned} \tag{7.2}$$

Beispiel 77 *Beispiele sind die Frobenius Norm*

$$\|A\|_F = \left(\sum_{i=1}^m \sum_{j=1}^n |a_{ij}|^2 \right)^{\frac{1}{2}}, \tag{7.3}$$

dies ist die euklidische Norm auf $\mathbb{R}^{mn}$*, und die Matrix* p*–Normen*

$$\|A\|_p = \sup_{x \neq 0} \frac{\|Ax\|_p}{\|x\|_p} = \sup_{x \neq 0} \left\| A \frac{x}{\|x\|_p} \right\|_p = \max_{\|x\|_p = 1} \|Ax\|_p \tag{7.4}$$

Insbesondere gilt

$$\|A\|_1 = \max_{1 \leq j \leq n} \sum_{i=1}^m |a_{ij}|$$

$$\|A\|_\infty = \max_{1\le i\le m} \sum_{j=1}^{n} |a_{ij}|$$

Die Matrix 2–Norm ist nicht so leicht zu charakterisieren wie $\|\bullet\|_1, \|\bullet\|_\infty$. *Es gilt, dass* $\|A\|_2^2$ *der größte Eigenwert von* A^TA *ist.*

Matrix Normen sind abhängig von der Dimension. $\|\bullet\|_2$ auf $\mathbb{R}^{3,2}$ ist etwas anderes als $\|\bullet\|_2$ auf $\mathbb{R}^{5,6}$.

Eine Matrixnorm heißt *konsistent*, falls $\|AB\| \le \|A\|\|B\|$. Die p-Normen und die Frobenius-Norm sind konsistent, d.h.,

$$\begin{array}{rcl} \|AB\|_p & \le & \|A\|_p\|B\|_p \\ \|AB\|_F & \le & \|A\|_2\|B\|_F \le \|A\|_F\|B\|_F \end{array} \quad \forall A \in \mathbb{R}^{m,n}, B \in \mathbb{R}^{n,q} \tag{7.5}$$

Nicht alle Normen erfüllen die Konsistenzbedingung.

Proposition 78 *Es gelten folgende Ungleichungen und Gleichungen*

$$\begin{aligned} \|A\|_2 &\le \|A\|_F \le \sqrt{n}\|A\|_2, \\ \max_{i,j}|a_{ij}| &\le \|A\|_2 \le \sqrt{mn}\max_{i,j}|a_{ij}|, \\ \frac{1}{\sqrt{n}}\|A\|_\infty &\le \|A\|_2 \le \sqrt{m}\|A\|_\infty, \\ \frac{1}{\sqrt{m}}\|A\|_1 &\le \|A\|_2 \le \sqrt{n}\|A\|_1. \end{aligned} \tag{7.6}$$

Weiterhin brauchen wir noch *Absolutbeträge für Matrizen.*
Sei $A \in \mathbb{C}^{m,n}$, dann ist $|A| = B$ mit $b_{ij} = |a_{ij}|$. Wir setzen

$$B \le A, \text{ falls } b_{ij} \le a_{ij}, \ \forall i = 1:m, j = 1:n.$$

Proposition 79 *Für das Rechnen mit Beträgen gelten folgende Regeln.*

$$|A + B| \le |A| + |B|,$$

$$\begin{aligned}
|AB| &\leq |A|\,|B|, \\
A &\leq B.\ C, D \geq O \Longrightarrow CAD \leq CBD, \\
\|A\|_p &\leq \|\,|A|\,\|_p, \\
\|A\| &= \|\,|A|\,\|,\ \textit{für}\ \|\bullet\| = \|\bullet\|_1, \|\bullet\|_\infty, \|\bullet\|_F, \\
|A| &\leq |B| \Longrightarrow \|A\|_1 \leq \|B\|_1, \|A\|_\infty \leq \|B\|_\infty, \|A\|_F \leq \|B\|_F.
\end{aligned}$$

Wir haben gesehen, dass sich beim Speichern oder Runden ergibt, dass

$$[gl(A)]_{ij} = [gl(a_{ij})] = [a_{ij}(1+\varepsilon_{ij})]\ \text{ mit } |\varepsilon_{ij}| \leq \text{ eps}\,.$$

Dann folgt

$$|gl(A) - A| \leq \text{ eps}\,|A|.$$

Dieses kann auch als Normungleichung umgeschrieben werden,

$$\|gl(A) - A\|_1 \leq \text{ eps}\,\|A\|_1.$$

Heute werden vielfach für Fehleranalysen nicht Normabschätzungen sondern elementweise Abschätzungen verwendet.

7.2 Lösung von Dreieckssystemen

Zur Anfang betrachten wir erst einmal Dreiecksmatrizen. Sei nun $L = [l_{ij}] \in \mathbb{R}^{n,n}$ eine invertierbare untere Dreiecksmatrix, d.h. $l_{i,j} = 0$ für $i < j$. Wir betrachten die Lösung von $Lx = b = [b_i]$ mit $b \in \mathbb{R}^n$. Wir erhalten sofort durch Vorwärts-Einsetzen die Lösung

$$x_i = \left(b_i - \sum_{j=1}^{i-1} l_{ij}x_j\right)/l_{ii} \quad i = 1, \cdots, n. \tag{7.7}$$

Dieses Verfahren wird durch folgenden Algorithmus realisiert.

Algorithmus 3
Input: $L \in \mathbb{R}^{n,n}$ *untere Dreiecks-Matrix, nichtsingulär,* $b \in \mathbb{R}^n$.
Output: *Lösung von* $Lx = b$, *überschrieben auf* b.

```
b(1) = b(1)/L(1,1);
FOR  i = 2 : n
        b(i) = (b(i) - L(i,1 : i-1) * b(1 : i-1))/L(i,i);
END
```

Die Multiplikation $L(i, 1 : i-1) * b(1 : i-1)$ ist natürlich eine Schleife.

Die Kosten für Algorithmus 3 betragen n^2 flops.

Eine Rückwärtsanalyse für Algorithmus 3 liefert für die berechnete Lösung von $\tilde{x}$:

$$(L + F)\tilde{x} = b, \text{ wobei } |F| \leq n \cdot \text{eps} \cdot |L| + \mathcal{O}(\text{eps}^2). \tag{7.8}$$

Der analoge Algorithmus für obere Dreiecksmatrizen heißt Rückwärts–Einsetzen. Sei $U \in [u_{ij}] \in \mathbb{R}^{n,n}$ obere Dreiecksmatrix. Für die Lösung von $Ux = b = [b_i] \in \mathbb{R}^n$ erhalten wir

$$x_i = \left(b_i - \sum_{j=i+1}^{n} u_{ij} x_j \right) / u_{jj} \quad i = n, n-1, \ldots, 1 \tag{7.9}$$

durch den folgenden Algorithmus.

Algorithmus 4
Input: $U \in \mathbb{R}^{n,n}$ *nichtsinguläre obere Dreiecks-Matrix,* $b \in \mathbb{R}^n$.
Output: *Lösung von* $Ux = b$, *überschrieben auf* b.

$b(n) = b(n)/U(n, n);$
FOR $i = n-1 : -1 : 1$
$\quad b(i) = (b(i) - U(i, i+1 : n) * b(i+1 : n))/U(i, i);$
END

Für die Kosten in Algorithmus 4 erhalten wir wieder n^2 flops und die Rückwärtsanalyse liefert für Algorithmus 4, dass

$$(U + F)\tilde{x} = b, \text{ wobei } |F| \leq n \cdot \text{eps} \cdot |U| + \mathcal{O}(\text{eps}^2). \tag{7.10}$$

Es gibt Varianten dieser Algorithmen für Parallel– und Vektorrechner (spaltenorientierte Versionen) und entsprechende Block–Versionen für mehrfache rechte Seiten.

Die Algorithmen zum Vorwärts– und Rückwärts–Einsetzen bilden die Basis für die Lösung von Gleichungssystemen mittels einer LR–Zerlegung und sind in Paketen wie LAPACK implementiert.

7.3 LR–Zerlegung

Wir kommen nun zur LR–Zerlegung einer Matrix A als $A = LR$, mit L untere und R obere Dreiecks-Matrix.

Falls man so eine Zerlegung hat, so löst man $Ax = b$ mittels Algorithmus 4 und 3, indem man $Ly = b$ und $Rx = y$ nacheinander löst. Die LR–Zerlegung wird mittels des Gaußschen Eliminationsverfahrens erzeugt. Dazu verwenden wir sogenannte Gauß–Transformationen.
Sei $x \in \mathbb{R}^n$ mit $x_k \neq 0$. Sei

$$t \equiv t^{(k)} = \begin{bmatrix} \left.\begin{matrix} 0 \\ \vdots \\ 0 \end{matrix}\right\} k \\ t_{k+1} \\ \vdots \\ t_n, \end{bmatrix}, \; t_i = \frac{x_i}{x_k}, i = k+1 : n$$

und sei e_k, der k-te Einheitsvektor.

Setze

$$M_k := I - t^{(k)} e_k^T \text{ (Dyade oder äußeres Produkt).} \tag{7.11}$$

Dann gilt

$$M_k x = \begin{bmatrix} 1 & & & & & \\ & \ddots & & & & \\ & & 1 & & & \\ & & -t_{k+1} & 1 & & \\ & & \vdots & & \ddots & \\ & & -t_n & & & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_k \\ x_{k+1} \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} x_1 \\ \vdots \\ x_k \\ 0 \\ \vdots \\ 0 \end{bmatrix}. \tag{7.12}$$

M_k heißt *Gauß–Transformation.*

Algorithmus 5

Input: $x \in \mathbb{R}^n, x_1 \neq 0$.

Output: *Vektor t der Länge $n-1$, so dass für die Gauß–Transformation M mit*
$M(2:n,1) = -t$ und $y = Mx$ gilt, dass $y(2:n) = 0$.

```
FUNCTION   t = GAUSS(x)
           n = length(x);
           t = x(2 : n)/x(1);
END GAUSS
```

Dieser Algorithmus benötigt $n-1$ flops.

Wir betrachten nun die Multiplikation mit einer Gauß–Transformation

$$M_kC = (I - t^{(k)}e_k^T)C = C - t^{(k)}(e_k^TC). \tag{7.13}$$

Da $t(1:k) = 0$ wird nur $C(k+1:n,:)$ verändert. Die Multiplikation wird durch folgenden Algorithmus realisiert.

Algorithmus 6

Input: $C \in \mathbb{R}^{n,r}, M \in \mathbb{R}^{n,n}$ *Gauß–Transformation mit* $M(2:n,1) = -t$.

Output: C *überschrieben mit* MC.

```
FUNCTION      C = GAUSSAPP(C, t)
              n = size(C, 1);
              C(2 : n, :) = C(2 : n, :) − t ∗ C(1, :);
END GAUSSAPP
```

Dieser Algorithmus benötigt $2(n-1)r$ flops.

Die Fehleranalyse für Algorithmus 6 liefert für den berechneten Wert $\tilde{t}$ für t aus *GAUSS*, dass

$$\tilde{t} = t + e, \text{ wobei } |e| \leq \text{ eps } |t|. \tag{7.14}$$

Damit erhält man für das Ergebnis von *GAUSSAPP*

$$gl\left((I - \tilde{t}e_1^T)C\right) = (I - te_1^T)C + E, \tag{7.15}$$

wobei

$$|E| \leq 3 \text{ eps } (|C| + |t|\,|C(1,:)|) + \mathcal{O}(\text{ eps}^2) \tag{7.16}$$

Falls $|t|$ groß ist, dann werden die Fehler in der Aufdatierung sehr groß gegenüber $|C|$.

Die Transformation auf Dreiecksgestalt erfolgt nun durch mehrfache Anwendung von *GAUSSAPP*.

Im k–ten Schritt erhalten wir (falls alles glatt geht)

$$A^{(k-1)} := M_{k-1}\cdots M_2 M_1 A = \begin{bmatrix} a_{11}^{(k-1)} & \cdots & a_{1,k-1}^{(k-1)} & \cdots & \cdots & a_{1,n}^{(k-1)} \\ 0 & \ddots & & & & \vdots \\ \vdots & \ddots & a_{k-1,k-1}^{(k-1)} & \cdots & \cdots & a_{k-1,n}^{(k-1)} \\ \vdots & & 0 & a_{kk}^{(k-1)} & & a_{k,n}^{(k-1)} \\ \vdots & & \vdots & \vdots & & \vdots \\ 0 & \cdots & 0 & a_{n,k}^{(k-1)} & \cdots & a_{nn}^{(k-1)} \end{bmatrix}. \tag{7.17}$$

Wir fahren dann fort auf dem noch nicht reduzierten unteren Block. Die vollständige Dreiecksreduktion wird dann durch die folgende Schleife erzielt.

$$\begin{array}{ll} n = size(A,1); & \\ k = 1; & \\ \text{WHILE} & A(k,k) \neq 0 \text{ AND } k < n \\ & t = GAUSS(A(k:n,k)); \\ & A(k:n,:) = GAUSSAPP(A(k:n,:),t); \\ & k = k+1; \\ \text{END} & \end{array} \tag{7.18}$$

Die Elemente $A(k,k)$, die während des Algorithmus auf '0' überprüft werden müssen, heißen *Pivots*. Ihre relative Größe ist entscheidend für die Fehleranalyse.

Matrizentheoretisch formuliert gilt, dass wenn die Schleife 7.18 mit $k = n$ endet, so ist

$$M_{n-1}\cdots M_1 A =: R$$

mit R obere Dreiecks-Matrix.

Für jedes $M_k = I - t^{(k)}e_k^T$ gilt $M_k^{-1} = I + t^{(k)}e_k^T$, und damit gilt

$$A = M_1^{-1}\cdots M_{n-1}^{-1} R =: L \cdot R.$$

Alle M_k sind untere Dreiecks-Matrizen mit 1–Diagonale also auch L.

Theorem 80 (Existenz und Eindeutigkeit der LR–Zerlegung)
Eine Matrix $A \in \mathbb{R}^{n,n}$ hat eine LR–Zerlegung genau dann, wenn

$$\det\left(A(1:k, 1:k)\right) \neq 0 \quad \textit{für } k = 1 : n-1. \tag{7.19}$$

Falls die LR–Zerlegung existiert und A nichtsingulär ist, so ist die LR–Zerlegung eindeutig und $\det A = r_{11} \cdots r_{nn}$.

Beweis. Falls A eine LR–Zerlegung

$$A = \begin{bmatrix} 1 & & & \\ l_{21} & \ddots & & \\ \vdots & \ddots & \ddots & \\ l_{n1} & \cdots & l_{n,n+1} & 1 \end{bmatrix} \begin{bmatrix} r_{11} & \cdots & \cdots & r_{1n} \\ & \ddots & & \vdots \\ & & \ddots & \vdots \\ & & & r_{n,n} \end{bmatrix}$$

besitzt, so gilt $r_{ii} \neq 0$ für $i = 1 : n-1$, da die r_{ii} die Pivots sind.

$$\det\left(A(1:k,1:k)\right) = \det\left(L(1:k,1:k)\right) \cdot \det\left(R(1:k,1:k)\right) = 1 \cdot \prod_{i=1}^{k} r_{ii} \neq 0.$$

Die Rückrichtung beweisen wir mit Induktion über k. Der Fall $k = 1$ ist klar. Angenommen $k-1$ Schritte sind ausgeführt, $A^{(k-1)} = M_{k-1} \cdots M_1 A$ und $a_{k,k}^{(k-1)}$ ist das k–te Pivot. Dann ist

$$\begin{aligned}
\underbrace{M_{k-1} \cdots M_1}_{M^{(k-1)}} A \quad &= \left[\begin{array}{ccc|ccc} 1 & & & & & \\ * & \ddots & & & & \\ \vdots & \ddots & 1 & & & \\ \hline \vdots & & * & 1 & & \\ \vdots & & \vdots & & \ddots & \\ * & \cdots & * & & & 1 \end{array}\right] A = A^{(k-1)} \\
&= \left[\begin{array}{ccc|ccc} a_{11}^{(k-1)} & \cdots & \cdots & \cdots & \cdots & * \\ & \ddots & & & & \vdots \\ & & a_{k-1,k-1}^{(k-1)} & \cdots & \cdots & * \\ \hline & & & a_{kk}^{(k-1)} & \cdots & * \\ & & & \vdots & & \vdots \\ & & & * & \cdots & * \end{array}\right].
\end{aligned}$$

Es folgt, dass

$$\begin{aligned}\det\left(A^{(k-1)}(1:k,1:k)\right) &= \prod_{i=1}^{k} a_{ii}^{(k-1)} \\ &= \underbrace{\det\left(M^{(k-1)}(1:k,1:k),\right)}_{=1}\cdot\det\left(A(1:k,1:k)\right).\end{aligned}$$

Also folgt aus $\det\left(A(1:k,1:k)\right)\neq 0$, dass $a_{k,k}^{(k-1)}\neq 0$ ist.

Zur Eindeutigkeit: Seien $A = L_1R_1 = L_2R_2$ zwei LR–Zerlegungen der nichtsingulären Matrix A. Dann sind L_i, R_i $i = 1,2$ auch nichtsingulär. Also folgt

$$L_2^{-1}L_1 = R_2R_1^{-1}$$

ist gleichzeitig untere Dreiecks-Matrix mit Einsdiagonale und obere Dreiecks-Matrix und damit die Einheitsmatrix. Also $L_1 = L_2, R_1 = R_2$ und

$$\det A = \det L\cdot\det R = 1\cdot\det R = r_{11}\cdots r_{nn}.$$

□

Die LR–Zerlegung läßt sich auch über anderen Körpern (Ringen) durchführen.

Einige praktische Details.

- Die Gauß–Transformation braucht nur auf Spalten $k:n$ angewendet werden, und selbst bei Spalte k kennen wir das Ergebnis. Also haben wir die folgende Änderung der Schleife

$$A(k:n,k+1:n) = \text{ GAUSSAPP}\left(A(k:n,k+1:n),t\right);$$

- Die Multiplikatoren, d.h. die wichtigen Elemente von t_k können auf den entstandenen Nullen von A gespeichert werden.

Damit erhält man folgenden Algorithmus für die LR–Zerlegung.

Algorithmus 7 (Gauß–Elimination)

Input:	$A \in \mathbb{R}^{n,n}$ *mit* $A(1:k, 1:k)$ *nichtsingulär für* $k = 1:n-1$.
Output:	*Faktorisierung* $M_{n-1} \cdots M_1 A = R$, *mit* R *obere Dreiecks-Matrix und* M_i *Gauß–Transformationen,* R *wird im oberen Dreieck von* A *und die Multiplikatoren aus* M_k *in* $A(k+1:n,k)$ *gespeichert, d.h.* $A(k+1:n,k) = -M_k(k+1:n,k)$.

FOR $k = 1 : n-1$
    $t =$ *GAUSS*$(A(k:n,k))$;
    $A(k+1:n,k) = t$;
    $A(k:n,k+1:n) =$ *GAUSSAPP*$(A(k:n,k+1:n),t)$;
END

Die Kosten für den Algorithmus betragen $\frac{2n^3}{3}$ flops , jeder Durchgang durch die k–Schleife ist ein äußeres Produkt.

Wir haben gesehen, daß wir die Multiplikatoren aus $t^{(k)}$ in A speichern können. Wie erhalten wir nun L (falls wir es benötigen)?

$$\begin{aligned} L &= (M_{n-1}\cdots M_1)^{-1} = M_1^{-1}\cdots M_{n-1}^{-1} \\ &= (I + t^{(1)}e_1^T)\cdots(I + t^{(n-1)}e_{n-1}^T) = I + \sum_{k=1}^{n-1} t^{(k)}e_k^T. \end{aligned}$$

Also folgt $L(k+1:n,k) = t^{(k)}$.

Die Lösung eines linearen Gleichungssystems folgt nun basierend auf der LR–Zerlegung wie oben beschrieben.

Im wesentlichen enthält Algorithmus 7 drei Schleifen die ineinandergeschachtelt sind. Je nach Rechnerarchitektur ist es besser eine andere als die beschriebene Anordnung zu verwenden. Siehe das Buch von Golub/Van Loan: 'Matrix Computations'.

7.4 Fehleranalyse der Gauß-Elimination

Da die Lösung von Gleichungssystemen so wichtig ist, wollen wir hier exemplarisch einmal die volle Rückwärts-Analyse betrachten. Um eine Idee zu bekommen was passiert betrachten wir das parametrisierte System

$$(A + \varepsilon F)x(\varepsilon) = b + \varepsilon f,\ x(0) = x,\ F \in \mathbb{R}^{n,n}, f \in \mathbb{R}^n.$$

Falls A nichtsingulär ist, so ist $x(\varepsilon)$ differenzierbar in einer Umgebung von 0 und es gilt:

$$\dot{x}(0) = A^{-1}(f - Fx).$$

Eine Taylorentwicklung liefert

$$x(\varepsilon) = x + \varepsilon\dot{x}(0) + \mathcal{O}(\varepsilon^2).$$

Also folgt für jede Vektornorm und zugehörige konsistente Matrixnorm, dass

$$\frac{\|x(\varepsilon) - x\|}{\|x\|} \leq |\varepsilon|\|A^{-1}\| \left\{ \frac{\|f\|}{\|x\|} + \|F\| \right\} + \mathcal{O}(\varepsilon^2). \tag{7.20}$$

Für quadratische Matrizen definieren wir dann die *Konditionszahl* einer Matrix als

$$\kappa_{\|\bullet\|}(A) := \|A\|\|A^{-1}\|, \tag{7.21}$$

diese ist abhängig von der Norm, und wir setzen $\kappa(A) = \infty$ für A singulär. Mit der Konsistenzungleichung $\|b\| \leq \|A\|\|x\|$ folgt dann für den relativen Fehler

$$\frac{\|x(\varepsilon) - x\|}{\|x\|} \leq \kappa(A)(\rho_A + \rho_b) + \mathcal{O}(\varepsilon^2), \tag{7.22}$$

wobei

$$\rho_A = \varepsilon\frac{\|F\|}{\|A\|}, \rho_b = \varepsilon\frac{\|f\|}{\|b\|}$$

die relativen Fehler in A, b sind. Damit ist $\kappa(A)$ der Verstärkungsfaktor für die relativen Fehler in den Daten.

Die Abschätzung (7.22) ist noch unbefriedigend, da sie auf „ε klein“ beruht, eine genaue Analyse erhält man mit den folgenden Resultaten.

Lemma 81 *Sei $F \in \mathbb{R}^{n,n}$ und $\| \bullet \|$ eine konsistente Norm. Falls $\|F\| < 1$, so ist $I - F$ nichtsingulär und es gilt*

$$(I-F)^{-1} = \sum_{k=0}^{\infty} F^k \qquad \textit{Neumannsche Reihe} \tag{7.23}$$

und

$$\|(I-F)^{-1}\| \leq \frac{1}{1-\|F\|}. \tag{7.24}$$

Beweis. Angenommen $I - F$ ist singulär. Dann gibt es $x \neq 0$ mit

$$(I-F)x = 0 \Longrightarrow \|x\| = \|Fx\| \leq \|F\|\|x\| \Longrightarrow \|F\| \geq 1.$$

Dies ist ein Widerspruch!

Für die Partialsummen erhalten wir

$$\left(\sum_{k=0}^{N} F^k\right)(I-F) = I - F^{N+1}.$$

Da $\|F\| < 1$ und $\|F^k\| \leq \|F\|^k$, so gilt $\lim\limits_{k\to\infty} F^k = 0$, und damit

$$\begin{aligned} \left(\lim_{N\to\infty} \sum_{k=0}^{N} F^k\right)(I-F) &= I \\ \lim_{N\to\infty} \sum_{k=0}^{N} F^k &= (I-F)^{-1} \end{aligned}$$

und

$$\|(I-F)^{-1}\| \leq \sum_{k=0}^{\infty} \|F\|^k = \frac{1}{1-\|F\|}.$$

□

Mit diesem Lemma erhält man

Lemma 82 *Sei $A \in \mathbb{R}^{n,n}$ nichtsingulär und $0 \neq b \in \mathbb{R}^n$. Es gelte $Ax = b$ und $(A+\triangle A)y = b+\triangle b$ mit $\triangle A \in \mathbb{R}^{n,n}$, $\triangle b \in \mathbb{R}^n$. und $\|\triangle A\| \leq \delta\|A\|$, $\|\triangle b\| \leq \delta\|b\|$. Falls $\delta\,\kappa(A) = r < 1$, so ist $A + \triangle A$ nichtsingulär und*

$$\frac{\|y\|}{\|x\|} \leq \frac{1+r}{1-r}.$$

Beweis. Da $\|A^{-1} \bigtriangleup A\| \leq \delta\|A^{-1}\|\,\|A\| = r < 1$, so folgt, dass $A + \bigtriangleup A$ nichtsingulär nach Lemma 81. Aus

$$(I + A^{-1} \bigtriangleup A)y = x + A^{-1} \bigtriangleup b$$

ergibt sich

$$\begin{aligned} \|y\| &\leq \|(I + A^{-1} \bigtriangleup A)^{-1}\| \left(\|x\| + \delta\|A^{-1}\|\,\|b\|\right) \\ &\leq \frac{1}{1-r}(\|x\| + \delta\|A^{-1}\|\,\|Ax\|) \\ &\leq \frac{1}{1-r}\left(\|x\| + \underbrace{\delta\|A^{-1}\|\,\|A\|}_{r}\,\|x\|\right). \end{aligned}$$

□

Damit erhalten wir jetzt ein genaueres Störungsresultat:

Theorem 83 *Unter den Voraussetzungen von Lemma 82 gilt*

$$\frac{\|x-y\|}{\|x\|} \leq \frac{2r}{1-r}. \tag{7.25}$$

Beweis. Wir haben $y - x = A^{-1} \bigtriangleup b - A^{-1} \bigtriangleup Ay$, also folgt, dass

$$\begin{aligned} \|y-x\| &\leq \delta\|A^{-1}\|\,\|b\| + \delta\|A^{-1}\|\,\|A\|\,\|y\| \\ &\leq \delta\|A^{-1}\|\,\|A\|\,\|x\| + \delta\|A^{-1}\|\,\|A\|\,\|y\| = r(\|x\| + \|y\|) \end{aligned}$$

und damit ergibt sich mit Lemma 82

$$\frac{\|y-x\|}{\|x\|} \leq r\left(1 + \frac{1+r}{1-r}\right).$$

□

Es folgt, daß der relative Fehler abhängig von der Konditionszahl ist. Es kann allerdings sein, daß die Schranke in (7.25) eine grobe Überschätzung ist.

Beispiel 84

$$\begin{bmatrix} 1 & 0 \\ 0 & 10^{-6} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 10^{-6} \end{bmatrix}, \implies x = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$\kappa_\infty(A) = \|A\|_\infty \|A^{-1}\|_\infty = 10^6$. *Sei* $\triangle b = \begin{bmatrix} 10^{-7} \\ 0 \end{bmatrix}, \triangle A = 0$, *so ist* $\delta = 10^{-7}, r = 0.1$ *und (7.25) ergibt sich die grobe Überschätzung* $\frac{10^{-7}}{10^0} \leq \frac{2}{0.9} 0.1 = \frac{2}{9}$.

Jedoch für $\triangle b = \begin{bmatrix} 0 \\ 10^{-7} \end{bmatrix}, \triangle A = 0$ *ist weiterhin* $\delta = 10^{-7}, r = 0.1$ *und (7.25) lautet* $\frac{10^{-1}}{10^0} \leq \frac{2}{9}$, *also gilt fast Gleichheit.*

Eine verbesserte Fehleranalyse ist möglich, wenn man komponentenweise Störungsresultate verwendet.

Lemma 85 *Sei* $A \in \mathbb{R}^{n,n}$ *nichtsingulär und* $0 \neq b \in \mathbb{R}^n$. *Es gelte* $Ax = b$ *und* $(A + \triangle A)y = b + \triangle b$ *mit* $\triangle A \in \mathbb{R}^{n,n}$, $\triangle b \in \mathbb{R}^n$. *und* $|\triangle A| \leq \delta |A|, |\triangle b| \leq \delta |b|$.

Falls $\delta \, \| \, |A^{-1}| \, |A| \, \| = r < 1$, *so ist* $A + \triangle A$ *nichtsingulär und*

$$\frac{\|y\|}{\|x\|} \leq \frac{1+r}{1-r},$$

wobei $\| \bullet \| = \| \bullet \|_\infty, \| \bullet \|_1, \| \bullet \|_F$.

Beweis. Übung. □

Theorem 86 *Unter den Voraussetzungen von Lemma 85 gilt*

$$\frac{\|y - x\|}{\|x\|} \leq \frac{2r}{1-r}, \tag{7.26}$$

wobei $\| \bullet \| = \| \bullet \|_\infty, \| \bullet \|_1, \| \bullet \|_F$.

Beweis. Übung □

Komponentenweise Abschätzungen sind i.a. schwieriger zu bekommen, aber oft aussagefähiger.

Beispiel 87 *Wir betrachten Beispiel 84. Hier ist* $\|\,|A^{-1}|\,|A|\,\|_\infty = 1$. *Sei* $\triangle b = \begin{bmatrix} 10^{-7} \\ 0 \end{bmatrix}, \triangle A = 0$, *so ist* $\delta = r = 10^{-7}$ *und (7.26) ergibt* $\frac{10^{-7}}{10^0} \leq \frac{2}{1-10^{-7}} 10^{-7} \approx 2 \cdot 10^{-7}$. *Für* $\triangle b = \begin{bmatrix} 0 \\ 10^{-7} \end{bmatrix}, \triangle A = 0$ *ist jetzt* $\delta = r = 10^{-1}$ *und (7.26) ergibt* $\frac{10^{-1}}{10^0} \leq \frac{2}{0.9} 10^{-1} = \frac{2}{9}$.

Diese Störungssätze wenden wir nun an. Zu Anfang betrachten wir nur gestörte Daten und exakte Rechnung. Es sei

$$gl(A) = A + E, \quad gl(b) = b + e,$$

und es gelte

$$(A+E)\tilde{x} = b + e, \quad \text{wobei } \|E\|_\infty \leq \text{ eps } \|A\|_\infty, \|e\|_\infty \leq \text{ eps } \|b\|_\infty. \tag{7.27}$$

Die Störungssätze ergeben mit $\text{eps}\,\kappa_\infty(A) \leq \frac{1}{2}$, dass

$$\frac{\|x - \tilde{x}\|_\infty}{\|x\|_\infty} \leq 4\,\text{eps}\,\kappa_\infty(A). \tag{7.28}$$

Beides sind „bestmögliche“ Norm–schranken, d.h. keine allgemeine $\|\bullet\|_\infty$–Analyse kann bessere Schranken liefern.

Wir können konsequenterweise keinen Algorithmus als schlecht abtun, der ein ungenaues Ergebnis für eine Matrix mit $\text{eps}\,\kappa_\infty(A) \approx 1$ liefert. **Für schlecht konditionierte Probleme kann also auch ein guter Algorithmus schlechte Ergebnisse liefern.**

Wir betrachten nun konkret das Gauß–Verfahren in der Form von Algorithmus 7.

Theorem 88 *Sei A eine $n \times n$ Matrix von Maschinenzahlen. Falls kein 0–Pivot während der Ausführung von Algorithmus 7 auftritt, dann erfüllen die berechneten Faktoren* $\tilde{L}, \tilde{R}$

$$\tilde{L}\tilde{R} = A + H \tag{7.29}$$

mit

$$|H| \leq 3(n-1)\,\text{eps}\,(|A| + |\tilde{L}||\tilde{R}|) + \mathcal{O}(\text{ eps}^2) \tag{7.30}$$

Beweis. Beweis mit Induktion über n. Der Fall $n = 1$ ist klar. Angenommen das Resultat sei richtig für alle $(n-1) \times (n-1)$ Maschinenzahl–Matrizen. Sei

$$A = \begin{bmatrix} \alpha & w^T \\ v & B \end{bmatrix} \begin{matrix} \}1 \\ \}n-1 \end{matrix},$$

so wird im ersten Schritt von Algorithmus 7 $\tilde{z} = gl(v/\alpha)$ und $\tilde{A}_1 = gl(B - \tilde{z}w^T)$ berechnet. Also folgt

$$\tilde{z} = \frac{1}{\alpha}v + f, \text{ mit } |f| \leq \text{ eps } \frac{|v|}{\alpha} \tag{7.31}$$

und

$$\tilde{A}_1 = B - \tilde{z}w^T + F, \text{ mit } |F| \leq 2\,\text{eps}\,(|B| + |\tilde{z}|\,|w|^T) + \mathcal{O}(\text{eps}^2). \tag{7.32}$$

Dann wird die LR–Zerlegung von $\tilde{A}_1$ berechnet. Mit Induktion folgt $\tilde{A}_1 = \tilde{L}_1\tilde{R}_1$ mit

$$\tilde{L}_1\tilde{R}_1 = \tilde{A}_1 + H_1, \text{ wobei } |H_1| \leq 3(n-2)\,\text{eps}\,(|\tilde{A}_1| + |\tilde{L}_1|\,|\tilde{R}_1|) + \mathcal{O}(\text{ eps }^2). \tag{7.33}$$

Also gilt

$$\begin{aligned} \tilde{L}\tilde{R} &= \begin{bmatrix} 1 & 0 \\ \tilde{z} & \tilde{L}_1 \end{bmatrix} \begin{bmatrix} \alpha & w^T \\ 0 & \tilde{R}_1 \end{bmatrix} \\ &= A + \begin{bmatrix} 0 & 0 \\ \alpha f & H_1 + F \end{bmatrix} = A + H. \end{aligned}$$

Aus (7.32) folgt

$$|\tilde{A}_1| \leq (1 + 2\,\text{eps}\,)(|B| + |\tilde{z}|\,|w|^T) + \mathcal{O}(\text{ eps }^2).$$

Mit (7.32), (7.33) folgt

$$|H_1 + F| \leq 3(n-1)\,\text{eps}\,(|B| + |\tilde{z}|\,|w|^T + |\tilde{L}_1|\,|\tilde{R}_1|) + \mathcal{O}(\text{ eps }^2)$$

und da $|\alpha f| \leq \text{ eps } |v|$, folgt

$$|H| \leq 3(n-1)\,\text{eps}\left\{\begin{bmatrix} |\alpha| & |w|^T \\ |v| & |B| \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ |\tilde{z}| & |\tilde{L}_1| \end{bmatrix}\begin{bmatrix} |\alpha| & |w|^T \\ 0 & |\tilde{R}_1| \end{bmatrix}\right\} + \mathcal{O}(\text{ eps }^2).$$

□

Damit haben wir eine Analyse über die LR–Zerlegung, jetzt müssen wir noch analysieren, was die Dreiecks-Löser machen.

Theorem 89 *Seien $\tilde{L}, \tilde{R}$ die berechneten LR–Faktoren aus Algorithmus 7. Bei Verwendung von Algorithmus 3 bzw. 4 zur Lösung von $\tilde{L}y = b$ und $\tilde{R}x = \tilde{y}$ ergibt sich $(A+E)\tilde{x} = b$ mit*

$$|E| \leq n\,\mathrm{eps}\,(3|A| + 5|\tilde{L}|\,|\tilde{R}|) + \mathcal{O}(\,\mathrm{eps}^2). \tag{7.34}$$

Beweis. Aus (7.8) und (7.10) ergibt sich

$$\begin{array}{ll} (\tilde{L}+F)\tilde{y} = b & |F| \leq n\,\mathrm{eps}\,|\tilde{L}| + \mathcal{O}(\,\mathrm{eps}^2), \\ (\tilde{R}+G)\tilde{x} = \tilde{y} & |G| \leq n\,\mathrm{eps}\,|\tilde{R}| + \mathcal{O}(\,\mathrm{eps}^2). \end{array}$$

Also

$$(\tilde{L}+F)(\tilde{R}+G)\tilde{x} = (\tilde{L}\tilde{R} + F\tilde{R} + \tilde{L}G + FG)\tilde{x} = b.$$

Es gilt $\tilde{L}\tilde{R} = A + H$ aus Satz 88 mit

$$|H| \leq 3(n-1)\,\mathrm{eps}\,(|A| + |\tilde{L}|\,|\tilde{R}|) + \mathcal{O}(\,\mathrm{eps}^2).$$

Setze $E := H + F\tilde{R} + \tilde{L}G + FG$, so gilt

$$\begin{array}{rcl} |E| & \leq & |H| + |F|\,|\tilde{R}| + |\tilde{L}|\,|G| + \mathcal{O}(\,\mathrm{eps}^2) \\ & \leq & 3n\,\mathrm{eps}\,(|A| + |\tilde{L}|\,|\tilde{R}|) + 2n\,\mathrm{eps}\,(|\tilde{L}|\,|\tilde{R}|) + \mathcal{O}(\,\mathrm{eps}^2). \end{array}$$

□

Wäre nicht der Term $|\tilde{L}|\,|\tilde{R}|$ in der Abschätzung, welcher groß sein kann, so wäre der Algorithmus rückwärts stabil. Da wir jedoch nichts gegen kleine Pivots machen können, falls diese auftauchen, kann $|\tilde{L}|, |\tilde{R}|$ sehr groß werden, und der Algorithmus ist damit **NICHT** rückwärts stabil.

7.5 Partielle Pivotisierung (Spaltenpivotisierung)

Um zu vermeiden, dass 0–Pivots oder sehr kleine Pivots auftauchen, wird jeweils in der momentanen Spalte, das betragsmäßig maximale Element gesucht und durch eine Zeilenvertauschung (Multiplikation mit Permutationsmatrizen P_k) in die Diagonalposition gebracht.

Diese Vorgehensweise heißt *Spaltenpivotisierung oder partielle Pivotisierung* und garantiert, dass alle Multiplikatoren vom Betrag kleiner oder gleich 1 sind. Es gilt

$$|(P_k M_{k-1} \cdots M_1 P_1 A)_{kk}| = \max_{k \le i \le n} |(P_k M_{k-1} \cdots M_1 P_1 A)_{ik}|\,, \; k = 1 : n-1.$$

Falls das Pivot 0 ist, so kann man einfach den Schritt der Elimination weglassen, da dann alle zu eliminierenden Elemente in dieser Spalte auch 0 sind.

Man erhält dann den folgenden Algorithmus.

Algorithmus 8 (Gauß–Elim. mit partieller Pivotisierung)

Input: $A \in \mathbb{R}^{n,n}$.

Output: *Gauß–Transformationen* $M_1, \dots, M_{k-1}$ *und Permutationen* $P_1, \dots, P_{n-1}$, *so dass* $M_{n-1}P_{n-1} \cdots M_1 P_1 A = R$ *obere Dreiecks-Matrix.*
Keiner der Multiplikatoren hat Betrag > 1. $A(k+1:n,k)$ *wird durch* $-M_k(k+1:n,k)$, $k = 1:n-1$ *überschrieben,* $A(1:k,k)$ *wird durch* $R(1:k,k)$, $k = 1:n$ *überschrieben. Im Pivotvektor* $piv(1:n-1)$ *werden die Vertauschungen gespeichert.* P_k *vertauscht die Zeilen* k *und* $piv(k)$, $k = 1:n-1$.

```
FOR  k = 1 : n − 1
     Bestimme μ (k ≤ μ ≤ n) mit |A(μ, k)| = ‖A(k : n, k)‖∞
     A(k, k : n) ⟷ A(μ, k : n)
     piv(k) = μ;
     IF   A(k, k) ≠ 0
          t = GAUSS(A(k : n, k));
          A(k + 1 : n, k) = t;
          A(k : n, k + 1 : n) = GAUSSAPP(A(k : n, k + 1 : n), t);
     END
END
```

Die Kosten für diesen Algorithmus sind $\mathcal{O}(n^2)$ Vergleiche und $\frac{2n^3}{3}$ flops.

Die Lösung des linearen Gleichungssystem $Ax = b$ erhält man also durch

$$y = M_{n-1}P_{n-1} \cdots M_1 P_1 b$$

und danach der Lösung von $Rx = y$. Sämtliche Informationen werden in A, b und piv gespeichert.

Beispiel 90

$$
\begin{aligned}
A &= \begin{bmatrix} 3 & 17 & 10 \\ 2 & 4 & -2 \\ 6 & 18 & -12 \end{bmatrix} \\
P_1 &= \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \\
P_1 A &= \begin{bmatrix} 6 & 18 & -12 \\ 2 & 4 & -2 \\ 3 & 17 & 10 \end{bmatrix} & piv(1) &= 3 \\
M_1 &= \begin{bmatrix} 1 & 0 & 0 \\ -\frac{1}{3} & 1 & 0 \\ -\frac{1}{2} & 0 & 1 \end{bmatrix} & M_1 P_1 A &= \begin{bmatrix} 6 & 18 & -12 \\ 0 & -2 & 2 \\ 0 & 8 & 16 \end{bmatrix} \\
P_2 &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} & piv(2) &= 3 \\
M_2 &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & \frac{1}{4} & 1 \end{bmatrix} & R &= \begin{bmatrix} 6 & 18 & -12 \\ 0 & 8 & 16 \\ 0 & 0 & 6 \end{bmatrix}
\end{aligned}
$$

Was passiert mit L?

Theorem 91 *Man verwende Gauß–Elimination mit partieller Pivotisierung zur Berechnung von*

$$M_{n-1} P_{n-1} \cdots M_1 P_1 A = R \tag{7.35}$$

mit Algorithmus 8. Dann gilt

$$PA = LR,$$

mit $P = P_{n-1} \cdots P_1$ (Permutationsmatrix), L untere Dreiecks-Matrix mit 1–Diagonale, $|l_{ij}| \leq 1$. Die k–te Spalte von L ist unterhalb der Diagonalen eine permutierte Version des k–ten Gauß-Vektors. Speziell für $M_k = I - t^{(k)} e_k^T$ gilt

$$L(k+1:n, k) = g(k+1:n), \; g = P_{n-1} \cdots P_{k+1} t^{(k)}.$$

Beweis. Aus (7.35) folgt

$$\tilde{M}_{n-1} \cdots \tilde{M}_1 P A = R \quad \text{mit}$$

$$\begin{aligned} \tilde{M}_{n-1} &= M_{n-1}, \\ \tilde{M}_k &= P_{n-1} \cdots P_{k+1} M_k P_{k+1} \cdots P_{n-1}, \ k \leq n-2. \end{aligned}$$

Da P_j nur Zeilen j und $\mu \geq j$ vertauscht, so folgt $P_j(1:j-1, 1:j-1) = I_{j-1}$. Also ist $\tilde{M}_k$ eine Gauß–Transformation mit Gauß–Vektor $\tilde{t}^{(k)} = P_{n-1} \cdots P_{k+1} t^{(k)}$. □

Durch die einfache Ersetzung der Zeile

$$A(k, k:n) \longleftrightarrow A(\mu, k:n)$$

durch

$$A(k, 1:n) \longleftrightarrow A(\mu, 1:n)$$

in Algorithmus 8 gilt dann wieder, dass $A(i,j)$ die Werte $L(i,j)$ für $i > j$ nach Ende des Algorithmus enthält.

Die Fehleranalyse liefert das folgende: Da Permutationen rundungsfehlerfrei durchgeführt werden können, kann man analog zeigen, daß die berechnete Lösung $\tilde{x}$ die Gleichung $(A+E)\tilde{x} = b$ erfüllt mit

$$|E| \leq n \,\text{eps}\, \left(3|A| + 5\tilde{P}^T |\tilde{L}|\, |\tilde{R}|\right) + \mathcal{O}(\text{eps}^2), \tag{7.36}$$

wobei $\tilde{P}, \tilde{L}, \tilde{R}$ die berechneten P, L, R sind.

Durch die Pivotisierung folgt

$$\|\tilde{L}\|_\infty \leq n,$$

und damit

$$\|E\|_\infty \leq n \,\text{eps}\, \left(3\|A\|_\infty + 5n\|\tilde{R}\|_\infty\right) + \mathcal{O}(\text{eps}^2).$$

Als letztes Problem bleibt die Bestimmung einer Schranke für $\|\tilde{R}\|_\infty$.

Definiere den *Wachstumsfaktor* ρ durch

$$\rho = \max_{i,j,k} \frac{|\tilde{a}_{ij}^{(k)}|}{\|A\|_\infty}, \tag{7.37}$$

wobei $\tilde{A}^{(k)}$ die berechnete Version von $A^{(k)}$ ist. Dann folgt

$$\|E\|_\infty \leq 8n^3\rho\|A\|_\infty \text{ eps } + \mathcal{O}(\text{ eps }^2). \tag{7.38}$$

Die Fehlerschranke hängt also wesentlich von ρ ab, der Faktor n^3 ist i.a. in der Praxis vernachlässigbar, da er nicht auftritt. Der Faktor ρ ist typisch von der Ordnung 10, kann aber om schlimmsten Fall 2^{n-1} sein.

Im allgemeinen ist der Gauß-Algorithmus mit partieller Pivotisierung sehr zuverlässig und kann relativ sorglos verwendet werden.

7.6 Vollständige Pivotisierung

Um den Wachstumsfaktor zu verkleinern gibt es eine Variante, in der das Pivot aus der gesamten Matrix $A^{(k-1)}(k:n, k:n)$ ausgewählt wird, d.h. wir bestimmen eine Zerlegung

$$M_{n-1}P_{n-1}\cdots M_1P_1AQ_1\cdots Q_{n-1} = R$$

mit Permutationsmatrizen P_i, Q_i, die so bestimmt werden, daß

$$\left|(P_kA^{(k-1)}Q_k)_{kk}\right| = \max_{k\leq i,j\leq n} \left|(P_kA^{(k-1)}Q_k)_{ij}\right|.$$

Theorem 92 *Bei Verwendung von Gauß–Elimination mit vollständiger Pivotisierung zur Berechnung von*

$$M_{n-1}P_{n-1}\cdots M_1P_1AQ_1\cdots Q_{n-1} = R \tag{7.39}$$

gilt

$$PAQ = LR$$

mit $P = P_{n-1}\cdots P_1$, $Q = Q_1\cdots Q_{n-1}$ *und* L *ist untere Dreiecks-Matrix mit 1–Diagonale und* $|l_{ij}| \leq 1$. *Für* $M_k = I - t^{(k)}e_k^T$ *gilt*

$$L(k+1:n, k) = g(k+1:n) \text{ mit } g = P_{n-1}\cdots P_{k+1}t^{(k)}.$$

Beweis. Analog zu Satz 91. □

Algorithmus 9 (Gauß–Elim. mit vollständiger Pivotisierung)
Input: $A \in \mathbb{R}^{n,n}$.
Output: *LR–Zerlegung von* PAQ *mit* P, Q *Produkte von elementaren Permutationsmatrizen,* $A(1 : k, k)$ *wird durch* $R(1 : k, k)$, $k = 1 : n$ *und* $A(k + 1 : n, k)$ *durch* $L(k + 1 : n, k)$, $k = 1 : n - 1$ *überschrieben.* P_k *vertauscht Zeilen* k *und* $p(k)$, Q_k *vertauscht Spalten* k *und* $q(k)$.

```
FOR   k = 1 : n − 1
      Bestimme μ, λ, so daß |A(μ, λ)| = max {|A(i, j)| : i, j = k : n}
      A(k, 1 : n) ←→ A(μ, 1 : n)
      A(1 : n, k) ←→ A(1 : n, λ)
      p(k) = μ;
      q(k) = λ;
      IF    A(k, k) ≠ 0
            t = GAUSS(A(k : n, k));
            A(k + 1 : n, k) = t;
            A(k : n, k + 1 : n) = GAUSSAPP(A(k : n, k + 1 : n), t);
      END
END
```

Die Kosten für den Algorithmus sind $\frac{2n^3}{3}$ flops und Vergleiche. Die Kosten für die Vergleiche sind nicht vernachlässigbar.

Bemerkung 93 *Man sieht sofort, dass vollständige Pivotisierung LR–Zerlegungen für Matrizen mit* $\operatorname{rank} A = r < n$ *erlaubt, denn wenn es kein Pivot ungleich* 0 *mehr gibt so ist auch die Elimination beendet.*

In exakter Arithmetik gilt für Algorithmus 9, dass

$$\left|a_{ij}^{(k)}\right| \leq k^{\frac{1}{2}}(2 \cdot 3^{\frac{1}{2}} \cdots k^{\frac{1}{k-1}})^{\frac{1}{2}} \max |a_{ij}|.$$

Die Schranke ist sehr langsam wachsend mit k. *Mit der empirischen Tatsache, daß* $\rho \approx 10$, *kann man aussagen, dass Gauß–Elimination mit vollständiger Pivotisierung rückwärts stabil ist, d.h. die Methode löst exakt*

$$(A + E)\hat{x} = b$$

für kleines E. *Die Wilkinson'sche Vermutung, dass die Schranke* n *ist, ist nicht richtig.*

7.7 Abschätzung der Genauigkeit

Das *Residuum* r der berechneten Lösung von $Ax = b$ ist der Vektor $r = b - A\tilde{x}$. Ein kleines Residiuum bedeutet, dass $A\tilde{x}$ eine gute Näherung von b ist. Wenn man annimmt, dass $(A+E)\tilde{x} = b, \|E\|_\infty \approx \text{eps } \|A\|_\infty$, so folgt $\|b - A\tilde{x}\|_\infty \approx \text{eps } \|A\|_\infty \|\tilde{x}\|_\infty$.

Heuristik I Gauß–Elimination erzeugt eine Lösung $\tilde{x}$ mit relativ kleinem Residuum.
Kleine Residuen implizieren hohe Genauigkeit jedoch nur bei kleiner Konditionszahl.

$$\frac{\|\tilde{x} - x\|_\infty}{\|x\|_\infty} \approx \text{ eps}\, \kappa_\infty(A).$$

Heuristik II Falls eps $\approx 10^{-d}$ und $\kappa_\infty(A) \approx 10^q$, dann erzeugt Gauß–Elimination eine Lösung mit ungefähr $d - q$ korrekten Dezimalstellen.

Beispiel 94

$$\begin{bmatrix} .986 & .579 \\ .409 & .237 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} .235 \\ .107 \end{bmatrix}, \ \kappa(A)_\infty \approx 700, \ x = \begin{bmatrix} 2 \\ 3 \end{bmatrix}.$$

eps	$\tilde{x}_1$	$\tilde{x}_2$	$\frac{\|\tilde{x}-x\|}{\|x\|_\infty}$	$\frac{\|b-A\tilde{x}\|_\infty}{\|A\|_\infty \|\tilde{x}\|_\infty}$
10^{-3}	2.11	-3.17	$5 \cdot 10^{-2}$	$2.0 \cdot 10^{-3}$
10^{-4}	1.986	-2.975	$8 \cdot 10^{-3}$	$1.5 \cdot 10^{-4}$
10^{-5}	2.0019	-3.0032	$1 \cdot 10^{-3}$	$2.1 \cdot 10^{-6}$
10^{-6}	2.00025	-3.00094	$3 \cdot 10^{-4}$	$4.2 \cdot 10^{-7}$

Eine weitere Verbesserung ist durch *Skalierung* der Daten zu erreichen. Dies sollte man in der Praxis immer durchführen.

$$Ax = b \Longleftrightarrow D_1^{-1} A D_2 y = D_1^{-1} b, \ y = D_2^{-1} x$$

Falls man für D_1, D_2 Diagonalmatrizen aus Maschinenzahlen der Form

$$\text{diag}\,(p^{r_1}, p^{r_2}, \ldots, p^{r_n})$$

nimmt, so kann die Skalierung ohne Rundungsfehler in $\mathcal{O}(n^2)$ flops durchgeführt werden. Dann gilt

$$\frac{\|D_2^{-1}(\tilde{x}-x)\|_\infty}{\|D_2^{-1}x\|_\infty} = \frac{\|\tilde{y}-y\|_\infty}{\|y\|_\infty} \approx \text{ eps } \kappa_\infty(D_1^{-1}AD_2). \tag{7.40}$$

Wenn man also $\kappa_\infty(D_1^{-1}AD_2)$ gegen $\kappa_\infty(A)$ verkleinert, erwartet man eine Verbesserung des Resultats. Es gibt zwei gebräuchliche Varianten:

a) Zeilenskalierung: $D_2 = I$, D_1 so, daß alle Zeilen ungefähr gleiche ∞–Norm haben.

b) Zeilen–Spalten–Gleichgewichtung: Wähle D_1, D_2 so, dass alle Zeilen und Spalten ∞–Norm im Intervall $\left[\frac{1}{p}, 1\right]$ haben (wobei p die Basis der Maschine ist).

Beispiel 95

$$\begin{bmatrix} 10 & 100000 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 100000 \\ 2 \end{bmatrix} \iff \begin{bmatrix} 0.0001 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

Bei dreistelliger Arithmetik, $p = 10$ ergibt das erste System $\tilde{x} = \begin{bmatrix} 0.00 \\ 1.00 \end{bmatrix}$ und das zweite $\begin{bmatrix} 1.00 \\ 1.00 \end{bmatrix}$. Exakte Lösung $x = \begin{bmatrix} 1.0001... \\ 0.9999... \end{bmatrix}$.

7.8 Iterative Verbesserung

Angenommen, wir haben $Ax = b$ mittels Gauß-Elimination mit partieller Pivotisierung gelöst, $PA = LR$, und wollen jetzt die Lösungsgenauigkeit verbessern. Wir könnten folgendes ausführen:

$$\begin{cases} r = b - A\tilde{x} \quad \text{(doppelt genau)} , \\ \text{Löse } Ly = Pr, \\ \text{Löse } Rz = y, \\ \text{Setze } x = \tilde{x} + z, \end{cases} \tag{7.41}$$

so folgt bei exakter Rechnung

$$Ax = A\tilde{x} + Az = (b - r) + r = b.$$

Wenn man (7.41) allerdings einfach so ausführt, ist das Ergebnis nicht genauer als $\tilde{x}$. Dies ist zu erwarten, denn $\tilde{r} = gl(b - A\tilde{x})$ hat im allgemeinen kaum richtige signifikante Stellen (siehe Heuristik I). Also

$$\tilde{z} = gl(A^{-1}r) = A^{-1} \cdot \text{Rauschen} = \text{Rauschen}.$$

Um eine Verbesserung zu erhalten, sollte man daher $r = b - Ax$ mit erhöhter Genauigkeit berechnen (doppelt so viele Stellen wie sonst). Dieser Prozess kann iteriert werden.

Heuristik III Bei Maschinengenauigkeit $\text{eps} = 10^{-d}$ und $\kappa_\infty(A) \approx 10^q$, hat nach k Schritten von (7.41) (mit doppelter Genauigkeit für das Residuum) das berechnete x ungefähr $\min\{d, k(d-q)\}$ korrekte Stellen.

Die Kosten betragen $\mathcal{O}(n^2)$ pro Nachiteratinsschritt. Bei Rechnung von $PA = LR$ in einfacher Genauigkeit und $\text{eps} \cdot \kappa_\infty(A) \leq 1$, ist die Lösung korrekt in einfacher Genauigkeit nach einer Iteration. Also haben wir kaum eine Verteuerung und dafür eine Verbesserung der Lösung.

7.9 Der Cholesky–Algorithmus, Bandmatrizen

Ein guter numerischer Algorithmus sollte spezielle Eigenschaften des mathematischen Problems berücksichtigen. Der Gauß–Algorithmus ist auf allgemeine lineare Gleichungssysteme zugeschnitten. Nun sind aber viele der Anwendungsprobleme so, dass die Gleichungssysteme eine spezielle Struktur haben, wie zum Beispiel Symmetrie oder eine Bandstruktur. Wir betrachten nun zuerst symmetrische Systeme

$$Ax = b \ \text{ mit } \ A = A^T, \ \text{ d.h. } a_{ij} = \overline{a}_{ij}, \tag{7.42}$$

und zusätzlich den wichtigen Spezialfall der positiven Definitheit, d.h.

$$x^T Ax > 0, \ \forall x \in \mathbb{C}^n \setminus \{0\}.$$

Wir untersuchen nun , wie wir den Gauß–Algorithmus verbessern können.

Theorem 96 (Cholesky Zerlegung)
Sei $A \in \mathbb{R}^{n,n}$, symmetrisch positiv definit. Dann existiert eine untere Dreiecks-Matrix $G \in \mathbb{R}^{n,n}$ mit positiven Diagonalelementen, so dass

$$A = GG^T.$$

Beweis. Da A positiv definit ist, sind alle Hauptabschnittsmatrizen

$$A(1:k, 1:k),\ k = 1, \ldots, n$$

positiv definit, haben also Determinante $\neq 0$. Also existiert eine eindeutige LR–Zerlegung $A = LR$ mit L untere Dreiecks-Matrix mit 1–Diagonale, und R hat Diagonalelemente > 0 (gleich den Pivots). Also kann man R schreiben als $D\tilde{R}$, wobei $\tilde{R}$ 1–Diagonale hat. Sei $D = \operatorname{diag}(d_1, \ldots, d_n)$. Die Pivots sind gerade die Determinanten der Hauptabschnittsmatrizen, also reell positiv (alle Eigenwerte sind reell, positiv), also ist $d_i > 0,\ \forall i = 1, \ldots, n$. Wir haben $A = LD\tilde{R} = LD^{\frac{1}{2}}D^{\frac{1}{2}}\tilde{R}$. $D^{\frac{1}{2}} = \operatorname{diag}(d_1^{\frac{1}{2}}, \ldots, d_n^{\frac{1}{2}})$.
Also folgt

$$D^{-\frac{1}{2}}L^{-1}AL^{-*}D^{-\frac{1}{2}} = D^{\frac{1}{2}}\tilde{R}L^{-*}D^{-\frac{1}{2}}.$$

Links steht eine symmetrische Matrix, rechts eine obere Dreiecks-Matrix mit 1–Diagonale. Also muß rechts die Einheitsmatrix stehen und es gilt

$$D^{\frac{1}{2}}\tilde{R} = D^{\frac{1}{2}}L^T$$

und damit $\tilde{R} = L^T$. Wir erhalten also $G := LD^{\frac{1}{2}}$. □

Damit können wir den Gauß–Algorithmus verbessern und erhalten den Cholesky–Algorithmus.

Algorithmus 10 (Cholesky–Zerlegung)

Input: *symmetrisch positiv definite Matrix $A \in \mathbb{R}^{n,n}$.*
Output: *Untere Dreiecks-Matrix $G \in \mathbb{R}^{n,n}$ mit positiver Diagonale, so daß $A = GG^T$. Für $i \geq j$ überschreibt $G(i,j)$ jeweils $A(i,j)$.*

FOR $k = 1 : n$
  $A(k,k) = \sqrt{A(k,k)}$;
  $A(k+1:n, k) = A(k+1:n, k)/A(k,k)$;
  FOR $j = k+1 : n$
    $A(j:n, j) = A(j:n, j) - A(j:n, k) * A(j,k)$;
  END
END

Die Kosten für diesen Algorithmus betragen $n^3/3$ flops . und die Fehleranalyse kann analog wie bei der LR–Zerlegung durchgeführt werden.

Die Lösung des Gleichungssystems erhält man dann wie bei Gauß mit Vorwärts– und Rückwärtseinsetzen.

Um bessere Stabilität zu erhalten sollten man auch hier wieder pivotisieren, es ist aber auch wichtig die Symmetrie zu erhalten. Dies erfordert dann symmetrische Permutationen PAP^T verwenden, die die Diagonalelemente vertauschen. Damit können dann auch positiv semidefinite Matrizen zerlegt werden.

7.10 Bandsysteme

Eine weitere häufig vorkommende spezielle Struktur ist die Bandstruktur. Eine Matrix heißt $p - q$ *Bandmatrix* mit *unterer Bandbreite* p und *oberer Bandbreite* q, falls $a_{ij} = 0$ für $i > j + p$ und $j > i + q$.

Beispiel 97

$$A = \begin{bmatrix} * & * & * & 0 & 0 \\ * & * & * & * & 0 \\ 0 & * & * & * & * \\ 0 & 0 & * & * & * \\ 0 & 0 & 0 & * & * \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

untere Bandbreite 1
obere Bandbreite 2
1–2 Bandmatrix.

Theorem 98 *Die Matrix $A \in \mathbb{R}^{n,n}$ habe eine LR–Zerlegung $A = LR$. Falls A eine $p - q$–Bandmatrix ist, so hat L untere Bandbreite p und R obere Bandbreite q.*

Beweis. Mit vollständiger Induktion:

$$A = \begin{bmatrix} \alpha & w^T \\ v & B \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ v/\alpha & I_{n-1} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & B - vw^T/\alpha \end{bmatrix} \begin{bmatrix} \alpha & w^T \\ 0 & I_{n-1} \end{bmatrix}$$

Die matrix $B - vw^T/\alpha$ ist $p - q$ Bandmatrix, da nur die ersten q Komponenten von w und die ersten p Komponenten von v ungleich 0 sind. Sei $L_1 R_1$ die LR–Zerlegung von $B - vw^T/\alpha$. Nach Induktionsvoraussetzung und unter Ausnutzung der Nullen in v, w folgt dass

$$L = \begin{bmatrix} 1 & 0 \\ v/\alpha & L_1 \end{bmatrix} \quad \text{und} \quad R = \begin{bmatrix} \alpha & w^T \\ 0 & R_1 \end{bmatrix}$$

die gewünschten Brandbreiten haben und es gilt $A = LR$. $\square$ Es folgt also,

dass man die Bandstruktur optimal ausnutzen kann vorausgesetzt, die LR–Zerlegung existiert, und man muss keine Vertauschungen durchführen.

Algorithmus 11 (Band Gauß–Elimination)

Input: *$p - q$–Bandmatrix $A \in \mathbb{R}^{n,n}$ welche eine LR–Zerlegung besitzt.*

Output: *Zerlegung $A = LR$, $A(i,j)$ überschrieben durch $L(i,j)$ für $i > j$ und $R(i,j)$ sonst.*

```
FOR k = 1 : n - 1
    FOR i = k + 1 : min(k + p, n)
        A(i,k) = A(i,k)/A(k,k);
    END
    FOR j = k + 1 : min(k + q, n)
        FOR i = k + 1 : min(k + p, n)
            A(i,j) = A(i,j) - A(i,k) * A(k,j);
        END
    END
END
```

Die Kosten für diesen Algorithmus betragen für $n \gg p, q$ ca. $2npq$ flops und die Fehleranalyse ist analog zur LR-Zerlegung ohne Pivotisierung. Man kann auch noch spezielle Speichertechniken nutzen und auch die Dreieckslöser verbessern.

Algorithmus 12 (Band–Vorwärts–Auflösen)

Input: *$L \in \mathbb{R}^{n,n}$ untere Dreiecks-Matrix mit 1–Diagonale und Bandbreite p, $b \in \mathbb{R}^n$.*

Output: *b überschrieben mit der Lösung von $Lx = b$.*

```
FOR j = 1 : n
    FOR i = j + 1 : min(j + p, n)
        b(i) = b(i) - L(i,j) * b(j);
    END
END
```

Die Kosten betragen für $n \gg p$ ca. $2np$ flops.

Algorithmus 13 (Band–Rückwärts–Auflösen)

Input: *$R \in \mathbb{R}^{n,n}$ obere Dreiecks-Matrix mit oberer Bandbreite $q, b \in \mathbb{R}^n$.*
Output: *b überschrieben mit der Lösung von $Rx = b$.*

```
FOR  j = n : −1 : 1
     b(j) = b(j)/u(j, j);
     FOR  i = max(1, j − q) : j − 1
          b(i) = b(i) − L(i, j) ∗ b(j);
     END
END
```

7.11 Householder-Orthogonalisierung

Wir haben bei der Fehleranalyse von Gauß gesehen, dass die Größe $|\tilde{L}|, |\tilde{R}|$ der berechneten Matrizen $\tilde{L}, \tilde{R}$ in die Abschätzung des Fehlers eingeht. Nun kann natürlich $|\tilde{L}|$ sehr groß sein, wenn wir ohne Pivotisierung arbeiten. Außerdem haben wir gesehen, daß die Lösung von Gleichungssystemen mit Gauß auch bei Pivotisierung nicht numerisch rückwärts stabil ist, wegen der Wachstumsfaktoren. In diesem Kapitel betrachten wir nun Methoden, die auf Zerlegungen mit orthogonalen Matrizen beruhen und bei denen wir numerische Stabilität zeigen können. Diese Zerlegungen werden dann auch verwendet, um überbestimmte Systeme zu lösen.

Sei $v \in \mathbb{R}^n \backslash \{0\}$. Eine $n \times n$ Matrix der Form

$$P = I - \frac{2vv^T}{v^Tv} \tag{7.43}$$

heißt *Householder Matrix*.

Eine Multiplikation mit P spiegelt einen Vektor x an der Hyperebene $\text{span}(v)^\perp$. Householder Matrizen sind symmetrisch und orthogonal. Sie sind Rang 1 Modifikationen der Identität. Sie werden verwendet, um bestimmte Komponenten eines Vektors zu eliminieren, analog wie bei Gauß–Transformationen. Wir werden hier jetzt nur den reellen Fall betrachten, im komplexen werden die Formeln etwas komplizierter.

Angenommen $x \in \mathbb{R}^n \setminus \{0\}$ und wir wollen v bestimmen, so dass $Px = ce_1$ (P wie in (7.43)), d.h.

$$Px = \left(I - \frac{2vv^T}{v^Tv}\right)x = x - \left(\frac{2v^Tx}{v^Tv}\right)v$$

Da wir wollen, dass $Px \in \text{span}\{e_1\}$ so folgt dass $v \in \text{span}\{x, e_1\}$. Setze $v = x + \alpha e_1$, dann gilt

$$v^T x = x^T x + \alpha x_1, v^T v = x^T x + 2\alpha x_1 + \alpha^2.$$

Also folgt, dass

$$Px = \left(1 - 2\frac{x^T x + \alpha x_1}{x^T x + 2\alpha x_1 + \alpha^2}\right) x - 2\alpha \frac{v^T x}{v^T v} e_1$$

Wir wählen α so, dass der Koeffizient vor x die 0 ist, d.h. $\alpha = \pm\|x\|_2 = \pm(x^T x)^{\frac{1}{2}}$. Dann gilt $v = x \pm \|x\|_2 e_1$ und damit

$$Px = \left(I - 2\frac{vv^T}{v^T v}\right) x = \pm\|x\|_2 e_1.$$

Wir haben dabei aber noch die Möglichkeiten das Vorzeichen zu wählen.

Falls $x \approx \alpha e_1$, dann hat $v = x - \alpha e_1$ eine sehr kleine Norm und es kann Auslöschung auftreten und damit ein großer relativer Fehler in $\beta = 2/v^T v$. Wähle daher immer $v = x + \text{sign}(x_1)\|x\|_2 e_1$. Dann gilt $\|v\|_2 \geq \|x\|_2$ und P ist fast perfekt orthogonal.

Weiterhin normalisiert man v so, dass auf $v(1) = 1$. Dies vereinfacht eine ganze Menge von Algorithmen, die auf Householder–Vektoren beruhen.

Algorithmus 14 (Erzeugung des Householder Vektors)

Input: $x \in \mathbb{R}^n$.

Output: $v \in \mathbb{R}^n$ *mit* $v(1) = 1$, *so daß* $\left(I - \frac{2vv^T}{v^T v}\right) x = \alpha e_1$.

```
FUNCTION   v = HOUSE(x)
           n = LENGTH(x);
           μ = ||x||_2;
           v = x;
           IF    μ ≠ 0
                 β = x(1) + sign(x(1)) * μ;
                 v(2 : n) = v(2 : n)/β;
           END
           v(1) = 1;
END HOUSE
```

Die Kosten betragen ca. $3n$ flops und das berechnete $\tilde{P}$ erfüllt $\|\tilde{P} - P\|_2 = \mathcal{O}(\text{ eps })$. Bei der Multiplikation mit Householder–Matrizen kann an einigen Stellen die Struktur ausgenutzt werden.

$$PA = \left(I - \frac{2vv^T}{v^T v}\right) A = A + vw^T$$

mit $w = \beta A^T v$ und $\beta = \frac{-2}{v^T v}$. Dies is eien Matrix–Vektor–Multiplikation und eine äußere–Produkt Aufdatierung.

Algorithmus 15 (Vormultiplikation mit Householder–Matrix)

Input: $A \in \mathbb{R}^{m,n}, v \in \mathbb{R}^m, v(1) = 1.$

Output: *A überschrieben mit* $PA = \left(I - \frac{2vv^T}{v^T v}\right) A.$

FUNCTION $A = ROWHOUSE(A, v)$
$\beta = -2/v^T * v;$
$w = \beta * A^T * v;$
$A = A + v * w^T;$
END ROWHOUSE

Die Kosten betragen $4mn$ flops und für dem Fehler gilt, dass $gl(\tilde{P}A) = P(A+E)$, $\|E\|_2 = \mathcal{O}(\text{ eps } \|A\|_2)$.

Algorithmus 16 (Nachmultiplikation mit Householder–Matrix)

Input: $A \in \mathbb{R}^{m,n}, v \in \mathbb{R}^n, v(1) = 1.$

Output: *A überschrieben mit* $AP = A\left(I - \frac{2vv^T}{v^T v}\right).$

FUNCTION $A = COLHOUSE(A, v)$
$\beta = -2/v^T * v;$
$w = \beta * A * v;$
$A = A + w * v^T;$
END COLHOUSE

Die Kosten betragen $4mn$ flops und für den Fehler gilt $gl(A\tilde{P}) = (A + E)P$, $\|E\|_2 = \mathcal{O}(\text{ eps } \|A\|_2)$.

Diese 3 Algorithmen werden verwendet um bestimmte Teile einer Matrix zu eliminieren.

Beispiel 99 *Überschreibe* $A \in \mathbb{R}^{m,n} (m \geq n)$ *mit* $B = Q^T A$ *wobei* Q *orthogonal, so dass* $B(j + 1 : m, j) = 0$ *für ein* j *mit* $1 \leq j \leq n$*. Sei weiter* *angenommen, dass $A(j:m, 1:j-1) = 0$. Wir wollen den nichtrivialen Teil von v in $A(j+1:m, j)$ speichern.*

$$\begin{aligned} v(j:m) &= HOUSE(A(j:m,j)); \\ A(j:m, j:n) &= ROWHOUSE(A(j:m, j:n), v(j:m)); \\ A(j+1:m, j) &= v(j+1:m); \end{aligned}$$

Mathematisch gesehen haben wir damit mit der $m \times m$ Householder–Matrix

$$P = \begin{bmatrix} I_{j-1} & 0 \\ 0 & \tilde{P} \end{bmatrix} = I - \frac{2vv^T}{v^T v}$$

multipliziert. Die Fehleranalyse von Wilkinson besagt, dass Berechnung und Anwendung von Householder Matrizen numerisch rückwärts stabil sind.

7.12 Die QR–Zerlegung.

Die QR–Zerlegung einer Matrix $A \in \mathbb{R}^{m,n}$ $(m \geq n)$ ist gegeben durch

$$A = QR$$

mit $Q \in \mathbb{R}^{m,m}$ orthogonal, $R = \begin{bmatrix} R_1 \\ O \end{bmatrix} \in \mathbb{R}^{m,n}$ mit R_1 obere Dreiecks–Matrix.

Falls A vollen Rang hat, dann bilden die ersten n Spalten von Q eine Orthonormalbasis für Bild (A). Mit Hilfe der QR–Zerlegung kann man also Orthonormalbasen für Mengen von Vektoren bestimmen.

Zur Lösung von Gleichungssystemen $Ax = b$ mit $A \in \mathbb{R}^{n,n}$ erhalten wir mit $A = QR$, dass

$$c = Q^T b, \ Rx = c.$$

Die Anwendung von Q^T auf b ist nur eine Matrix–Vektor Multiplikation oder in faktorisierter Householder Form, mehrere innere Produkte. Die Lösung von $Rx = c$ erfolgt durch Rückwärts-Einsetzen.

Algorithmus 17 (Householder QR–Zerlegung)

Input: $A \in \mathbb{R}^{m,n}$ *mit* $m \geq n$.

Output: *Householder–Matrizen* $P_1, \ldots, P_n$, *so daß für* $Q = P_1 \cdots P_n, Q^T A = R$ *obere Dreiecks-Matrix. Der obere Dreiecks-Teil von* A *wird durch* R *überschrieben und die Komponenten* $j+1 : m$ *des* j*–ten Householdervektors werden auf* $A(j+1 : m, j)$, $j < m$ *überschrieben.*

```
FOR  j = 1 : n
     v(j : m) = HOUSE(A(j : m, j));
     A(j : m, j : n) = ROWHOUSE(A(j : m, j : n), v(j : m));
     IF    j < m
           A(j + 1 : m, j) = v(j + 1 : m);
     END
END
```

Die Kosten betragen $2n^2(m - \frac{n}{3})$ flops sowie, falls Q benötigt wird, noch einmal

$$4(m^2 n - mn^2 + \frac{n^3}{3})\,\text{flops}\,.$$

Die Fehleranalyse liefert, dass $\tilde{Q}^T(A + E) = \tilde{R}$, wobei das berechnete $\tilde{Q}$ exakt orthogonal ist und $Q^T\tilde{Q} = I + F$ mit $\|F\|_2 \approx$ eps und weiterhin $\|E\|_2 \approx \text{eps}\,\|A\|_2$.

Beispiel 100

$$\begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \end{bmatrix}, \; P_1 A = \begin{bmatrix} * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & * & * & * & * \\ 0 & * & * & * & * \\ 0 & * & * & * & * \\ 0 & * & * & * & * \end{bmatrix}, \; P_2 P_1 A = \begin{bmatrix} * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & 0 & * & * & * \\ 0 & 0 & * & * & * \\ 0 & 0 & * & * & * \\ 0 & 0 & * & * & * \end{bmatrix}, \quad \textit{usw.}$$

Die Matrix A sieht am Ende des Algorithmus wie folgt aus

$$\begin{bmatrix} r_{11} & r_{12} & \cdots & \cdots & r_{1,n} \\ v_2^{(1)} & r_{22} & & & \vdots \\ v_3^{(1)} & v_3^{(2)} & \ddots & & \vdots \\ \vdots & \vdots & \ddots & r_{n-1,n-1} & \vdots \\ \vdots & \vdots & & v_n^{(n-1)} & r_{n,n} \\ \vdots & \vdots & & \vdots & v_{n+1}^{(n)} \\ \vdots & \vdots & & \vdots & \vdots \\ v_m^{(1)} & v_m^{(2)} & \cdots & v_m^{(n-1)} & v_m^{(n)} \end{bmatrix}$$

Die j–te Komponente von $v^{(j)}$ ist jeweils 1.

Der obige Algorithmus beweist die Existenz der QR-Zerlegung. Es gelten noch folgende weitere Eigenschaften.

Theorem 101 *Ist $A = QR$ eine QR–Zerlegung einer Matrix $A = [a_1, \ldots, a_n] \in \mathbb{R}^{m,n}$ von vollem Rang und $Q = [q_1, \ldots, q_n]$, dann ist*

$$\text{span}\,\{a_1, \cdots, a_k\} = \text{span}\,\{q_1, \cdots, q_k\},\ k = 1 : n.$$

Für $Q_1 = Q(1 : m, 1 : n), Q_2 = Q(1 : m, n+1 : m)$ gilt:

$$\begin{aligned} \text{Bild}\,(A) &= \text{Bild}\,(Q_1), \\ \text{Bild}\,(A)^\perp &= \text{Bild}\,(Q_2) \end{aligned}$$

und $A = Q_1 R_1$ mit $R_1 = R(1 : n, 1 : n)$.

Beweis. Es gilt

$$a_k = \sum_{i=1}^{k} r_{ik} q_i \in \text{span}\,\{q_1, \ldots, q_k\}$$

und daher span $\{a_1, \ldots, a_k\} \subseteq$ span $\{q_1, \ldots, q_k\}$. Da rang$(A) = n$, so folgt

$$\dim(\text{span}\,\{a_1, \ldots, a_k\}) = k,\ \forall k \leqslant n$$

und damit span $\{a_1, \ldots, a_k\} =$ span $\{q_1, \ldots, q_k\}$. Der Rest ist klar. $\square$

Theorem 102 *Die Matrix $A \in \mathbb{R}^{m,n}$ habe vollen Rang. Die „dünne“ QR–Zerlegung $A = Q_1 R_1$ mit $Q_1 \in \mathbb{R}^{m,n}$ und $R \in \mathbb{R}^{n,n}$ obere Dreiecks-Matrix mit $r_{ii} \in \mathbb{R}, r_{ii} > 0$, ist eindeutig. $G = R_1^T$ ist der Cholesky Faktor von $A^T A$.*

Beweis. Es gilt $A^T A = (Q_1 R_1)^T (Q_1 R_1) = R_1^T R_1$. Der Faktor R_1 ist eindeutig und da A vollen Rang hat auch $Q_1 = A R_1^{-1}$. $\square$

7.13 Gram–Schmidt Orthogonalisierung

Wir haben gesehen, dass wir mit Hilfe der Householder QR-Zerlegung orthogonalisieren können. Die dünne QR-Zerlegung $A = Q_1 R_1$ wie in Satz 102 kann man natürlich auch über den Gram–Schmidt Prozess erhalten. Sei $\operatorname{rang}(A) = n$. Der klassische Gram–Schmidt Algorithmus ist dann gegeben durch

$$\begin{aligned} r_{ik} &= q_i^T a_k, \ i = 1 : k-1 \\ q_k &= (a_k - \sum_{i=1}^{k-1} r_{ik} q_i)/r_{kk}. \end{aligned}$$

Numerisch gesehen ist dieser Algorithmus sehr schlecht, da die Orthogonalität der q_k durch Rundungsfehler zerstört wird. Eine leichte Umsortierung ergibt den sogenannten modifizierten Gram–Schmidt (MGS), der numerisch wesentlich besser ist. Definiere $A^{(k)} \in \mathbb{R}^{m,n-k+1}$ durch

$$A - \sum_{i=1}^{k-1} q_i r_i^T = \sum_{i=k}^{n} q_i r_i^T = [0 \ A^{(k)}].$$

Mit $A^{(k)} = [\underbrace{z}_{1} \ \underbrace{B}_{n-k}]$ gilt, dass $r_{kk} = \|z\|_2, \ q_k = z/r_{kk}$ und $[r_{k,k+1}, \ldots r_{k,n}] = q_k^T B$.
Damit können wir

$$A^{(k+1)} = B - q_k [r_{k,k+1}, \ldots, r_{k,n}]$$

als äußeres Produkt bestimmen und weitermachen.

Algorithmus 18 (Modifizierter Gram–Schmidt)

Input: $A \in \mathbb{R}^{m,n}$ *Rang* $(A) = n$.

Output: *Zerlegung* $A = Q_1 R_1, Q_1 \in \mathbb{R}^{m,n}$ *mit orthonormalen Spalten,* $R_1 \in \mathbb{R}^{n,n}$ *obere Dreiecks-Matrix.*

```
FOR  k = 1 : n
     R(k,k) = ||A(1 : m,k)||_2;
     Q(1 : m,k) = A(1 : m,k)/R(k,k);
     FOR  j = k + 1 : n
          R(k,j) = Q(1 : m,k)^T * A(1 : m,j);
          A(1 : m,j) = A(1 : m,j) - Q(1 : m,k) * R(k,j);
     END
END
```

Die Kosten betragen $2mn^2$ flops und die Fehleranalyse liefert $\tilde{Q}_1^T \tilde{Q}_1 = I + E, \|E\|_2 \approx \text{eps}\,\kappa_2(A)$.

Zum Vergleich erhielten wir bei der Householder Orthogonalisierung $\|E\|_2 = \text{eps}$.

Allerdings ist zur Berechnung einer Orthonormalbasis für Bild(A) der modifizierte Gram–Schmidt schneller als QR.

7.14 Ausgleichsprobleme

Bei vielen wissenschaftlichen Versuchen geht es darum, aus Messungen die Werte von Konstanten $x_1, \ldots, x_n$ zu bestimmen. Oft kann man x_i nicht direkt messen sondern nur eine leichter zugängliche Größe y, die als Funktion von x und anderen Parametern z abhängt, $y = f(z, x_1, \ldots, x_n)$.

Beispiel 103 *Bestimmung von g. Wir machen Fallversuche $h = g\frac{t^2}{2}$, wobei h die Fallhöhe, t die Fallzeit und g die Gravitationskonstante.*

Um die x_i zu bestimmen, führen wir $m > n$ verschiedene Experimente (Messungen) durch und erhalten

$$y_k = f(z_k, x_1, \ldots, x_n),\ k = 1, \ldots, m.$$

Dies ergibt ein überbestimmtes Gleichungssystem, doch dies ist i.a. nicht lösbar. Daher machen wir eine beste Approximation möglich. Zum Beispiel minimieren wir

$$\sum_{k=1}^{m} (y_k - f(z_k, x_1, \ldots, x_n))^2 \tag{7.44}$$

Das ist die Methode der kleinsten Quadrate (least squares).

Oder wir minimieren die Maximums-Norm

$$\max_{1\le k\le m} |y_k - f(z_k, x_1, \ldots, x_n)|\,.$$

Für (7.44) ergibt sich die notwendige Bedingung für die Minimierung durch

$$\frac{\partial}{\partial x_i}\left(\sum_{k=1}^{m}(y_k - f_k(x_1, \ldots, x_n))^2\right) = 0,\ i = 1 : n, \tag{7.45}$$

wobei $f_k(x_1, \ldots, x_n) := f(z_k, x_1, \ldots, x_n)$.

Dies sind die sogenannten *Normalengleichungen.* Ein wichtiger Spezialfall, das lineare Ausgleichsproblem, liegt vor, falls $f_k(x_1, \ldots, x_n)$ linear in $x_1, \ldots, x_n$, d.h.

$$\begin{bmatrix} f_1(x_1, \ldots, x_n) \\ \vdots \\ f_m(x_1, \ldots, x_n) \end{bmatrix} = Ax,\ A \in \mathbb{R}^{m,n}. \tag{7.46}$$

Dann sind die Normalengleichungen

$$\mathrm{grad}_x\left((y - Ax)^T(y - Ax)\right) = 2A^TAx - 2A^Ty = 0. \tag{7.47}$$

Also man hat das lineare Gleichungssystem

$$A^TAx = A^Ty \tag{7.48}$$

zur Lösung von

$$\min_{x\in\mathbb{R}^n} \|Ax - y\|_2. \tag{7.49}$$

Theorem 104 *Das lineare Ausgleichsproblem (7.49) hat mindestens eine Lösung x_0. Ist x_1 eine weitere Lösung, so gilt $Ax_0 = Ax_1$. Das Residuum $r = y - Ax_0$ ist eindeutig bestimmt und erfüllt $A^Tr = 0$. Jede Lösung x_0 erfüllt auch (7.48) und umgekehrt.*

Beweis. $\mathbb{R}^m = \underbrace{\mathrm{Bild}\,(A)}_{L} \oplus \underbrace{\mathrm{Bild}\,(A)^\perp}_{L^\perp}$.

Daher gibt es eine eindeutige Zerlegung

$$y = s + r,\ \text{mit}\ s \in L, r \in L^\perp \tag{7.50}$$

und es gibt ein x_0 mit $A^T x_0 = s$. Da $A^T r = 0$, so folgt $A^T y = A^T s = A^T A x_0$. Also löst x_0 (7.48) die Normalengleichungen.
Umgekehrt sei x_1 eine Lösung von (7.48). Dann hat y die Zerlegung $y = s + r$ mit $s = Ax_1, r = y - Ax_1,\ s \in L, r \in L^\perp$. Wegen der Eindeutigkeit der Zerlegung folgt $Ax_1 = Ax_0$.
Weiter erfüllt jede Lösung x_0 von (7.48) das Ausgleichsproblem (7.49). Denn für beliebiges x setze $z = Ax - Ax_0,\ r = y - Ax_0$, dann gilt wegen $r^T z = 0$:

$$\|y - Ax\|_2^2 = \|r - z\|_2^2 = \|r\|_2^2 + \|z\|_2^2 \geq \|r\|_2^2 = \|y - Ax_0\|_2^2.$$

□

Falls A vollen Rang hat, so ist $A^T A$ positiv definit. Nun könnte man einfach folgenden Algorithmus verwenden:

Algorithmus 19 (Normalengleichung)

Input: $A \in \mathbb{R}^{m,n}$ *mit* $\operatorname{rang}(A) = n$, *und* $b \in \mathbb{R}^m$.
Output: *Lösung x des Minimierungsproblems (7.49).*

Berechne das untere Dreieck von $C = A^T A$.
$d = A^T b$.
Berechne die Cholesky Zerlegung $C = GG^T$.
Löse $Gy = d$ und $G^T x = y$.

Die Kosten betragen $(m + n/3)n^2$ flops und die Fehleranalyse liefert das folgende: Angenommen, es werden keine Fehler bei der Berechnung von $C = A^T A$ und $d = A^T b$ gemacht.

Dann folgt aus der Analyse der Cholesky Zerlegung, dass

$$(A^T A + E)\tilde{x} = A^T b,$$

mit $\|E\|_2 \approx \text{eps}\, \|A^T\|_2 \|A\|_2 \approx \text{eps}\, \|A^T A\|_2$., d.h. wir erwarten

$$\frac{\|x - \tilde{x}\|_2}{\|x\|_2} \approx \text{eps}\, \kappa_2(A^T A) = \text{eps}\, \kappa_2(A)^2,$$

d.h. das Quadrat der Konditionszahl geht ein. Das ist schlecht, denn es bedeutet, dass wir im Extremfall viel weniger korrekte Stellen haben als möglich.

Beispiel 105

$$A = \begin{bmatrix} 1 & 1 \\ 10^{-3} & 0 \\ 0 & 10^{-3} \end{bmatrix},\ b = \begin{bmatrix} 2 \\ 10^{-3} \\ 10^{-3} \end{bmatrix},\ x = \begin{bmatrix} 1 \\ 1 \end{bmatrix},\ \kappa_2(A) = 1.4 \cdot 10^3.$$

Mit 6–stelliger Arithmetik ist $A^TA = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$ *singulär. Mit 7–stelliger Arithmetik folgt*

$$\tilde{x} = \begin{bmatrix} 2.00001 \\ 0 \end{bmatrix} \Longrightarrow \frac{\|\tilde{x} - x\|_2}{\|x\|_2} \approx \underbrace{\text{eps}}_{10^{-6}} \underbrace{\kappa_2(A)^2}_{10^6}.$$

Die Normalengleichung kann also zu sehr schlechten Ergebnissen führen. Wir verwenden daher besser die QR–Zerlegung. Bilde

$$\begin{aligned} Q^TA &= R = \begin{array}{c} \begin{bmatrix} R_1 \\ 0 \end{bmatrix} \end{array} \begin{array}{l} n \\ m-n \end{array}, \quad R_1 \text{ obere Dreiecks-Matrix} \\ Q^Tb &= \begin{bmatrix} c \\ d \end{bmatrix} \begin{array}{l} n \\ m-n \end{array} \end{aligned}$$

Dann gilt, da Q orthogonal ist, dass

$$\|Ax - b\|_2^2 = \|Q^TAx - Q^Tb\|_2^2 = \|R_1x - c\|_2^2 + \|d\|_2^2,$$

für alle $x \in \mathbb{R}^n$. Falls $\operatorname{rang}(A) = \operatorname{rang}(R_1) = n$, so gilt für die Lösung $R_1x = c$. Also hat man:

Algorithmus 20 (*QR* Lösung des linearen Ausgleichsproblems)
Input: $A \in \mathbb{R}^{m,n}$ *mit* $\operatorname{rang}(A) = n$, *und* $b \in \mathbb{R}^m$.
Output: $x \in \mathbb{R}^n$, *so dass* $\|Ax - b\|_2 \stackrel{!}{=} \min$.

Verwende Algorithmus 17, um A mit seiner QR–Zerlegung zu überschreiben.

```
FOR  j = 1 : n
     v(j) = 1;
     v(j + 1 : m) = A(j + 1 : m, j);
     b(j : m) = ROWHOUSE(b(j : m), v(j : m));
END
```

Löse $R(1:n, 1:n)x = b(1:n)$ *mit Rückwärts–Einsetzen.*

Die Kosten für dieses Verfahren betragen $2n^2(m - \frac{n}{3})$ flops . Die Fehleranalyse ergibt, dass das berechnete $\tilde{x}$ das Minimierungsproblem $\min \|(A + \delta A)x - (b + \delta b)\|_2$, löst, wobei

$$\begin{aligned} \|\delta A\|_F &\leq (6m - 3n + 41)n \,\text{eps}\, \|A\|_F + \mathcal{O}(\text{eps}^2) \\ \|\delta b\|_2 &\leq (6m - 3n + 40)n \,\text{eps}\, \|b\|_2 + \mathcal{O}(\text{eps}^2) \end{aligned}$$

Es ist wieder klar, dass es Probleme gibt, wenn $\kappa_2(R) \approx \frac{1}{\text{eps}}$. Ein weiteres Problem entsteht natürlich wenn $\text{Rang}(A) < n$.

Abhilfe schafft hier die sogenannte Singulärwertzerlegung $A = U\Sigma V^T$ mit U, V orthogonal, $\Sigma = \left[\begin{array}{ccc} \sigma_1 & & 0 \\ & \ddots & \\ 0 & & \sigma_n \\ \hline & 0 & \end{array}\right]$. Dies kann zu diesem Zeitpunkt hier nicht gemacht werden.

7.15 Iterative Verfahren

Für viele der Gleichungssysteme, die in der Praxis auftauchen, gilt $n \gg 100000$ und typischerweise $a_{ij} = 0$ für $> 90\%$ der Einträge. Wenn man derartige Probleme mit Gauß oder QR löst, kann es passieren, das in den Faktoren L, Q, R alle Einträge $\neq 0$ sind. Als Extrembeispiel betrachte den Gauß-Algorithmus für die Matrix

$$\begin{bmatrix} * & * & \cdots & * \\ * & * & & 0 \\ \vdots & & \ddots & \\ * & 0 & & * \end{bmatrix}.$$

Die Matrix läuft total voll. Es gibt spezielle dafür Varianten von Gauß/Cholesky/QR. Dies ist Thema einer Spezialvorlesung.

Eine wichtige Idee, die insbesondere im Zusammenhang mit Vektorrechnern und Parallelrechnern von größter Bedeutung ist, ist die Konstruktion von Verfahren, die die Matrix an sich nicht verändern, und wenn möglich auf Matrix * Matrix oder Matrix * Vektor–Operationen aufgebaut sind. Das führt auf iterative Verfahren. Im wesentlichen betrachten wir hier 2 Ansätze.

7.15.1 Splitting–Verfahren

Zerlege A additiv

$$A = M - N, \tag{7.51}$$

wobei M^{-1} eine Approximation an A^{-1} ist, die leicht invertierbar ist, z.B. diagonal, blockdiagonal, Dreiecks-Matrix,… . Ersetze $Ax = b$ durch

$$Mx = Nx + b \Longleftrightarrow x = M^{-1}Nx + M^{-1}b. \tag{7.52}$$

Dies ist eine Fixpunktgleichung und wir können die Grundidee eines Fizpunktverfahrens anwenden, d.h.

$$x^{(k+1)} = M^{-1}Nx^{(k)} + M^{-1}b, \text{ mit gegebenen } x^{(0)}. \tag{7.53}$$

Theorem 106 *Das Fixpunktverfahren (7.53) konvergiert für alle Startwerte $x^{(0)}$ gegen die Lösung von $Ax = b$ genau dann, wenn der Spektralradius $\rho(M^{-1}N) < 1$ ist, wobei $\rho(T) = \max\{|\lambda| : \lambda \text{ Eigenwert von } T\}$.*

Beweis. Der Beweis folgt aus dem Banachschen Fixpunktsatz. □ Mögliche

Wahlen für M, N: Setze $A = D - L - U$, D Diagonale, L unteres Dreieck, U oberes Dreieck. Mit $M = D,\ N = L + U$ ergibt sich das Verfahren

$$x^{(k+1)} = D^{-1}(L + U)x^{(k)} + D^{-1}b \tag{7.54}$$

Algorithmus 21 (Jacobi oder Gesamtschrittverfahren)

Input: *$A \in \mathbb{R}^{n,n}$, $b \in \mathbb{R}^n$, $a_{ii} \neq 0, i = 1, \ldots, n$ und Startvektor $x^{(0)} = [x_1^{(0)}, \ldots, x_n^{(0)}]^T$ sowie Abbruchgrenze δ.*

Output: *Näherung $\tilde{x}$ für Lösung $Ax = b$*

FOR $k = 0, 1, 2, \ldots$
  FOR $i = 1 : n$

$$x_i^{(k+1)} = \left(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k)}\right) / a_{ii};$$

  END
  IF $\|x^{(k+1)} - x^{(k)}\| < \delta$, *STOP*
END

Die Kosten sind im wesentlichen ein Matrix–Vektor Produkt (unter Ausnutzung der Sparsität) und 1 Vektoraddition pro Iterationsschritt.

Die Fehleranalyse is die von der Matrix * Vektor, Vektor+Vektor Operation. Der Verfahrensfehler ist abhängig von $\delta, \rho(D^{-1}(L + U))$.

Mit $M = D - L,\ N = U$ ergibt sich das Verfahren

$$x^{(k+1)} = (D - L)^{-1}Ux^{(k)} + (D - L)^{-1}b \tag{7.55}$$

Algorithmus 22 (Gauß–Seidel oder Einzelschrittverfahren)

Input: $A \in \mathbb{R}^{n,n},\ b \in \mathbb{R}^n,\ a_{ii} \neq 0, i = 1, \ldots, n$ *und Startvektor* $x^{(0)} = [x_1^{(0)}, \ldots, x_n^{(0)}]^T$ *sowie Abbruchgrenze* δ.

Output: *Näherung* $\tilde{x}$ *für Lösung* $Ax = b$

FOR $k = 0, 1, 2 \ldots$

FOR $i = 1 : n$

$$x_i^{(k+1)} = \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)} \right) / a_{ii};$$

END

IF $\|x^{(k+1)} - x^{(k)}\| < \delta$, *STOP*

END

Die Kosten sind im wesentlichen die gleichen wie bei der Jacobi-Iteration, das Verfahren ist nur schwerer auf Vektor- und Parallelrechnern zu implementieren

Der Fehler ist abhängig vom Fehler in Lösung von $(D - L)x = c$ sowie von $\rho((D - L)^{-1}U), \delta$.

Beide Verfahren sind einfach, aber konvergieren i.a. sehr langsam, insbesondere bei den Problemen, die in der Praxis auftauchen.

Eine Möglichkeit, die Konvergenz zu beschleunigen, ist die sukzessive Überrelaxation (SOR).

Mit $M = D - \omega L, N = (1-\omega)D + \omega U, \omega \in (0, 2), (M - N) = \omega(D - L - U)$ ergibt sich das Verfahren

$$x^{(k+1)} = (D - \omega L)^{-1}((1 - \omega)D + \omega U)x^{(k)} + (D - \omega L)^{-1}\omega b. \tag{7.56}$$

Algorithmus 23 (SOR)

Input: $A \in \mathbb{R}^{n,n},\ b \in \mathbb{R}^n,\ a_{ii} \neq 0, i = 1, \ldots, n$ *und Startvektor* $x^{(0)} = [x_1^{(0)}, \ldots, x_n^{(0)}]^T$ *sowie Abbruchgrenze* δ.

Output: *Näherung* $\tilde{x}$ *für Lösung* $Ax = b$

FOR $k = 0, 1, 2 \ldots$

  FOR $i = 1 : n$

$$x_i^{(k+1)} = \omega \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)} \right) / a_{ii} + (1-\omega) x_i^{(k)};$$

  END

  IF $\|x^{(k+1)} - x^{(k)}\| < \delta$, *STOP*

END

Kosten und Fehler wie beim Gauß–Seidel Verfahren.

In einigen Spezialfällen von strukturierten Matrizen aus partiellen Differentialgleichungen gibt es Konvergenzbeweise, optimales ω, etc. Weitere Beschleunigungsmethoden: Tschebyschew–Beschleunigung, semiiterative Verfahren, unvollständige Dreieckszerlegungen.

7.15.2 Das Konjugierte Gradienten Verfahren

Die Idee des Verfahrens der Konjugierten Gradienten (CG) bruht auf einer einfachen Grundidee, die wiederum aus der Behandlung von Optimierungsproblemen herrührt.

Betrachte eine quadratisches Funktional

$$\Phi(x) = \frac{1}{2} x^T A x - x^T b,\ A \in \mathbb{R}^{n,n} \text{ symmetrisch, positiv definit, } b \in \mathbb{R}^n.$$

Das Minimum von Φ ist der Wert $-b^T A^{-1} b/2$. Dieser wird erreicht bei $x = A^{-1} b$. Also ist Minimierung von $\Phi(x)$ äquivalent zur Lösung von $Ax = b$.

Eine Grundmethode aus der Optimierung ist die *Methode des Steilsten Abstiegs* In jedem Schritt lege die Richtung des steilsten Abstiegs fest und gehe in diese Richtung so weit wie möglich. Beim Punkt x_c angelangt, ist die Richtung des steilsten Abstiegs die Richtung des negativen Gradienten.

$$-\nabla \Phi|_{x_c} = b - A x_c = r_c, \tag{7.57}$$

dabei ist r_c das Residiuum in x_c. Falls $r_c \neq 0$, dann gibt es ein α, so daß $\Phi(x_c + \alpha r_c) < \Phi(x_c)$. Wähle

$$\alpha = \frac{r_c^T r_c}{r_c^T A r_c}. \tag{7.58}$$

Das ergibt das Minimum von $\Phi(c_x + \alpha r_c)$ über α. Man hätte damit folgende Methode des steilsten Abstiegs.

$k = 0;\ x_0 = 0;\ r_0 = b;$
WHILE $r_k \neq 0$
  $k = k + 1;$
  $\alpha_k = r_{k-1}^T r_{k-1} / r_{k-1}^T A r_{k-1};$
  $x_k = x_{k-1} + \alpha_k r_{k-1};$
  $r_k = b - A x_k;$
END

Dieses Verfahren konvergiert i.a. sehr langsam. Besser ist es, die Suchrichtungen so zu wählen, dass die Residuen senkrecht aufeinanderstehen, so dass man bei exakter Rechnung in $\leq n$ Schritten fertig ist.

Algorithmus 24 (Konjugierte Gradienten–Verfahren)

Input: *$A \in \mathbb{R}^{n,n}$, symmetrisch, positiv definit, $b \in \mathbb{R}^n$, $k_{\max}$ (Maximum an Iterationsschritten), $\varepsilon > 0$ Toleranz.*

Output: *Näherungslösung von $Ax = b$.*

$k = 0$; $x_0 = 0$; $r_0 = b$; $\rho_0 = \|r_0\|_2^2$;
WHILE $\sqrt{\rho_k} > \varepsilon\sqrt{\rho_0}$ *AND* $k < k_{\max}$
    $k = k + 1$;
    IF $k = 1$
        $p = r$;
    ELSE
        $\beta = \rho_{k-1}/\rho_{k-2}$; % $(= r_{k-1}^T r_{k-1}/r_{k-2}^T r_{k-2})$
        $p = r + \beta p$; % $(= p_k)$
    END
    $w = Ap$;
    $\alpha = \rho_{k-1}/p^T w$; % $(= r_{k-1}^T r_{k-1}/p_k^T A p_k)$
    $x = x + \alpha p$; % $(= x_k)$
    $r = r - \alpha w$; % $(= r_k)$
    $\rho_k = \|r\|_2^2$; % $(= r_k^T r_k)$.
END

Die Kosten für diese Methode sind 1 Matrix * Vektor–Multiplikation + 2 Skalarprodukte + 2 GAXPY (a+x*b) pro Schritt.

Die Fehleranalyse liefert, dass die Rundungsfehler die Konvergenzgeschwindigkeit erniedrigen und die Orthogonalität der Residuen zerstören.

Bemerkung 107
- *Für die Vektoren $p \equiv p_k$, $r \equiv r_k$ aus Algorithmus 24 gilt: $p_k^T A p_l = 0$ und $r_k^T r_l = 0$ für alle $k > l \geqslant 0$.*
- *Für die Funktion Φ gilt mit $p_1, \ldots, p_k$ aus Algorithmus 24 gilt:*

$$\min_{\alpha_1,\ldots,\alpha_k \in \mathbb{R}} \Phi(x + \sum_{l=1}^{k} \alpha_l p_l) = \min_{\alpha_k \in \mathbb{R}} \left(\min_{\alpha_1,\ldots,\alpha_{k-1} \in \mathbb{R}} \Phi(x + \sum_{l=1}^{k} \alpha_l p_l) \right).$$

Theorem 108 *Sei $A \in \mathbb{R}^{n,n}$ symmetrisch, positiv definit und $b \in \mathbb{R}^n$. Für die Iterierten in Algorithmus 24 gilt:*

$$\begin{aligned} \|x - x_k\|_A &\equiv \sqrt{(x - x_k)^T A (x - x_k)} \\ &\leq 2\|x - x_0\|_A \left(\frac{\sqrt{\kappa_2(A)} - 1}{\sqrt{\kappa_2(A)} + 1} \right)^k . \end{aligned} \tag{7.59}$$

Beweis. Siehe Golub/Van Loan. ☐ Die Genauigkeit der Methode ist oft besser

als (7.59) vorhersagt. Je näher $\kappa_2(A)$ an 1, je schneller ist die Konvergenz. Daher verwendet man heute in der Praxis noch Tricks zur Konvergenzbeschleunigung durch sogenannte Präkonditionierung.

Anstatt des ursprünglichen Systems $Ax = b$ löst man

$$(C^{-1}AC^{-T})(C^T x) = C^{-1}b \tag{7.60}$$

wobei C die folgenden Anforderung erfüllen sollte:

- C sollte invertierbar sein;
- Gleichungsssysteme mit C sind einfach zu lösen;
- $\kappa_2(C^{-1}AC^{-T})$ sollte möglichst nahe an 1 liegen.

In vielen Anwendungen wie etwa bei Finite Elemente Methoden für Partielle Differentialgleichungen, hat man natürliche Präkonditionierer. Ein gut präkonditioniertes Kongugiertes Gradientenverfahren ist für symmetrisch, positiv definite Systeme i.a. das beste Verfahren.

Kapitel 8

Lösung nichtlinearer Gleichungssysteme

Viele Anwendungsprobleme führen auf die Aufgabe der Lösung von nichtlinearen Gleichungssystemen. Gegeben

$$
\begin{array}{ccc}
f : E & \longrightarrow & F \qquad (\text{z.B. hier } E = F = \mathbb{R}^m) \\
\begin{bmatrix} x_1 \\ \vdots \\ x_m \end{bmatrix} & \longmapsto & \begin{bmatrix} f_1(x_1, \ldots, x_m) \\ \vdots \\ f_m(x_1, \ldots, x_m) \end{bmatrix}
\end{array}
$$

Gesucht: Vektor $[x_1, \cdots, x_m]^T$ so dass $f_i(x_1, \cdots, x_m) = 0$, für alle $i = 1, \cdots, n$. E, F können auch unendlich dimensionale lineare Räume sein, etwa Räume von Funktionen und f kann auch ein Differentialoperator sein. Hier betrachten wir nur den Fall $E = F = \mathbb{R}^m$. Im allgemeinen ist die Lösung nur durch Näherungsverfahren zu bestimmen. Nullstellenprobleme sind eng verwandt mit Minimierungsproblemen wie

$$\text{Bestimme für } h : \mathbb{R}^m \longrightarrow \mathbb{R} \quad \min_{x \in \mathbb{R}^m} h(x). \tag{8.1}$$

In üblicher Weise muß zur Bestimmung des Minimums

$$\text{grad } (h(x)) = \begin{bmatrix} \frac{\partial h}{\partial x_1} \\ \vdots \\ \frac{\partial h}{\partial x_m} \end{bmatrix} = 0$$

berechnet werden, welches wiederum ein Nullstellenproblem ist.

8.1 Fixpunktverfahren

Die einfachste idee zur Lösung nichtlinearer Gleichungen sind Fixpunktverfahren.

Man wandelt dazu das Nullstellenproblem in ein Fixpunktproblem

$$x = \Phi(x) \tag{8.2}$$

um. Dies ist im allgemeinen auf viele Arten möglich, die äquivalent oder nicht äquivalent sein können.

Beispiel 109 $e^x - \sin x = 0$
Mögliche Fixpunktgleichungen, diese sind nicht alle äquivalent.

$$\begin{array}{llll} a) & x & = & e^x - \sin x + x, \\ b) & x & = & \sin x - e^x + x, \\ c) & x & = & \arcsin(e^x), \quad \textit{für } x < 0, \\ d) & x & = & \ln(\sin x), \quad \textit{für } (-2n\pi, 2n\pi + \pi), n = 1, 2, 3, \ldots. \end{array}$$

Die Idee des Fixpunktverfahrens ist es, eine Folge

$$x^{(i+1)} = \Phi(x^{(i)}), \quad i = 0, 1, 2, \ldots \tag{8.3}$$

mit gegebenem Startwert x_1 zu erzeugen. Die Konvergenz dieser Folge gegen eine Lösung erhalten wir mit dem Banachschen Fizpunktsatz.

Theorem 110 (Banachscher Fixpunktsatz) *Sei $D \subseteq D_\Phi \subseteq \mathbb{R}^m$ ein abgeschlossenes Gebiet und Φ eine kontrahierende Abbildung von D in sich, d.h.*

a) $\Phi : D \longrightarrow D$

b) Φ kontrahierend, d.h. $\exists\, \alpha < 1$ und eine Norm $\|\bullet\|$, so dass

$$\|\Phi(x) - \Phi(y)\| \leq \alpha \|x - y\|, \quad \forall x, y \in D. \tag{8.4}$$

Dann gilt:

i) Es gibt genau einen Fixpunkt $\hat{x}$ von (8.2) in D.

ii) Die Fixpunktiteration (8.3) konvergiert für jeden Startwert $x^{(0)} \in D$ gegen $\hat{x}$.

iii) Es gelten die Fehlerabschätzungen

$$\|\hat{x} - x^{(n)}\| \leq \frac{\alpha^n}{1-\alpha}\|x^{(1)} - x^{(0)}\| \quad \textit{a priori}, \tag{8.5}$$

$$\|\hat{x} - x^{(n)}\| \leq \frac{\alpha}{1-\alpha}\|x^{(n)} - x^{(n-1)}\| \quad \textit{a posteriori}\ . \tag{8.6}$$

Beweis. Wenn $x^{(0)} \in D$, so gilt $x^{(n)} = \Phi(x^{(n-1)}) \in D$, für alle $n = 1, 2, \ldots$. Wir erhalten aus der Lipschitzbedingung, dass

$$\begin{aligned}
\|x^{(n+1)} - x^{(n)}\| &= \|\Phi(x^{(n)}) - \Phi(x^{(n-1)})\| \\
&\leq \alpha\|x^{(n)} - x^{(n-1)}\| \leq \alpha^2\|x^{(n-1)} - x^{(n-2)}\| \\
&\leq \alpha^n\|x^{(1)} - x^{(0)}\|
\end{aligned}$$

und damit

$$\begin{aligned}
\|x^{(n+k)} - x^{(n)}\| &\leq \sum_{i=1}^{k} \|x^{(n+i)} - x^{(n+i-1)}\| \\
&\leq \sum_{i=1}^{k} \alpha^{n+i-1}\|x^{(1)} - x^{(0)}\| \\
&\leq \alpha^n\|x^{(1)} - x^{(0)}\| \sum_{i=1}^{k} \alpha^{i-1} \\
&\leq \frac{\alpha^n}{1-\alpha}\|x^{(1)} - x^{(0)}\|.
\end{aligned}$$

Damit ist die Folge eine Cauchy–Folge und da $\mathbb{R}^m$ vollständig ist erhalten wir Konvergenz. Da Φ stetig ist, dies folgt bereits aus der Kontraktionseigenschaft (8.4)), so ist der Grenzwert Fixpunkt von Φ, wegen

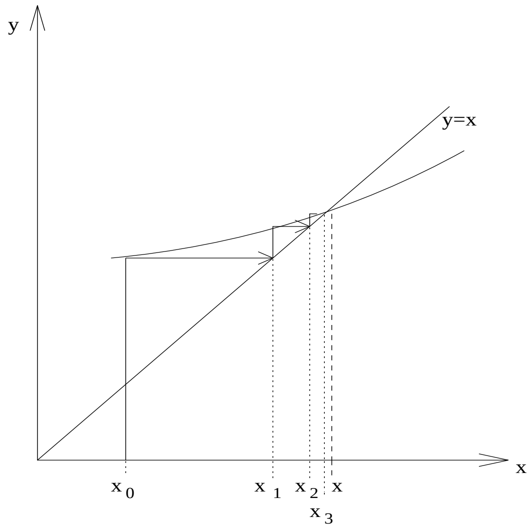
$$\hat{x} \equiv \lim_{n\mapsto\infty} x^{(n+1)} = \lim_{n\mapsto\infty} \Phi(x^{(n)}) = \Phi(\lim_{n\mapsto\infty} x^{(n)}) = \Phi(\hat{x}).$$

Für den beweis der Eindeutigkeit seien $\hat{x}, \bar{x}$ Fixpunkte von Φ. Dann gilt

$$\|\hat{x} - \bar{x}\| = \|\Phi(\hat{x}) - \Phi(\bar{x})\| \leqslant \alpha\|\hat{x} - \bar{x}\|$$

und da $\alpha < 1$, so kann nur $\hat{x} = \bar{x}$ gelten.

Graphische Veranschaulichung der Konvergenz $m = 1$.



Die Abschätzungen ergeben sich wie folgt:

$$
\begin{aligned}
\|x^{(n+k)} - x^{(n)}\| &\leq \frac{\alpha^n}{1-\alpha}\|x^{(1)} - x^{(0)}\| \\
\overset{k\to\infty}{\Longrightarrow} \|\hat{x} - x^{(n)}\| &\leq \frac{\alpha^n}{1-\alpha}\|x^{(1)} - x^{(0)}\|
\end{aligned}
$$

und dies ist (8.5). Mit $n = 1$ folgt

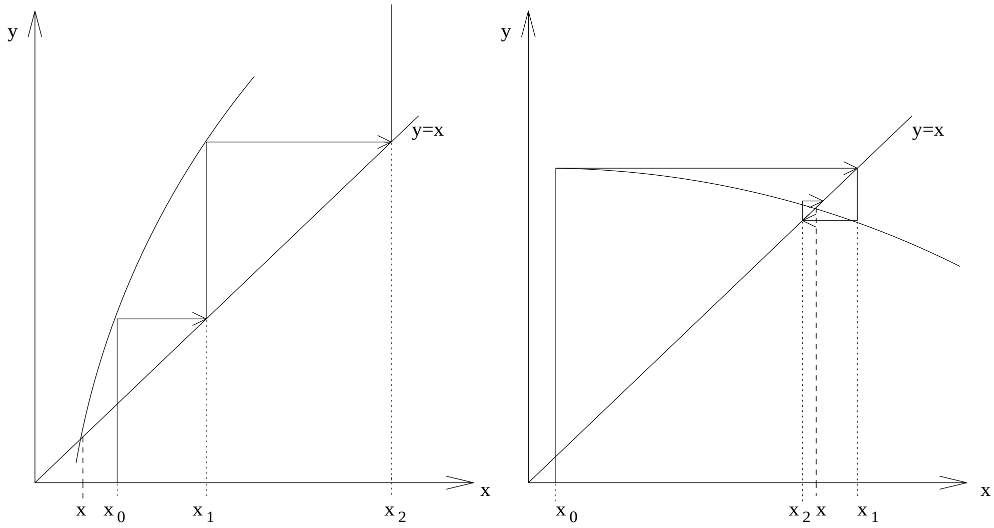
$$
\|\hat{x} - x^{(1)}\| \leq \frac{\alpha}{1-\alpha}\|x^{(1)} - x^{(0)}\|
$$

Ersetze $x^{(1)}$ durch $x^{(n)}undx^{(0)}$ durch $x^{(n-1)}$ so folgt (8.6). $\square$

Definition 111 *Ein Fixpunkt $\hat{x}$ heißt* anziehend*, wenn die Iteration (8.3) für alle genügend nahe bei $\hat{x}$ gelegenen Startwerte $x^{(0)} \neq \hat{x}$ gegen $\hat{x}$ konvergiert.*

Ein Fixpunkt $\hat{x}$ heißt abstoßend*, wenn es eine offene Kugel U_ε um $\hat{x}$ gibt, so dass für alle $x^{(0)} \in U_\varepsilon \setminus \{\hat{x}\}$ ein m existiert, so dass $x^{(m)} \notin U_\varepsilon$.*

Abstoßender Fixpunkt / Anziehender Fixpunkt.



Theorem 112 *Sei $D \subseteq D_\Phi \subseteq \mathbb{R}^m$ abgeschlossen und konvex und man nehme an, dass*
$$D\Phi = \begin{bmatrix} \frac{\partial\Phi_1(x)}{\partial x_1} & \cdots & \frac{\partial\Phi_1(x)}{\partial x_m} \\ \vdots & & \vdots \\ \frac{\partial\Phi_m(x)}{\partial x_1} & \cdots & \frac{\partial\Phi_m(x)}{\partial x_m} \end{bmatrix}$$
existiert und stetig ist. Falls es eine Norm gibt, so dass $\|D\Phi(x)\| \leq \alpha < 1$, so ist Φ in D kontrahierend mit Lipschitzkonstante α.

Beweis. Übung. □ Ist $D\Phi(\hat{x})$ stetig und $\|D\Phi(\hat{x})\| < 1$ so ist $\hat{x}$ anziehend.

Ein Fixpunkt ist im allgemeinen nicht anziehend, falls einer der Eigenwerte von $D\Phi(x)$ vom Betrag größer 1 ist.

Definition 113 *Eine Iterationsfolge $\{x^{(i)}\}_{i=0}^{\infty}$, die für $p \geq 1$*

$$\|x^{(i+1)} - \xi\| \leq c\|x^{(i)} - \xi\|^p \quad i = 0, 1, 2 \cdots \tag{8.7}$$

und $c < 1$ für $p = 1$ erfüllt, heißt Folge von mindestens p–ter Ordnung.

Corollary 114 *Das Fixpunktverfahren (8.3) sofern es konvergiert, ist konvergent von mindestens 1. Ordnung (linear konvergent).*

Die Kosten für die Iteration (8.3) sind durch eine Auswertung von $\Phi(x)$ pro Iteration gegeben. Die Anzahl der Iterationen ist aus der Fehlerschätzung ablesbar. Wir versuchen daher eine höhere Konvergenzordnung zu bekommen.

8.2 Das Newtonverfahren

Die Grundidee beim Newtonverfahren ist die Linearisierung. Man macht eine Taylorentwicklung für die Nullstelle $\hat{x}$ von $f(x)$, für $x^{(0)}$ aus einer Umgebung der Nullstelle.

$$0 = f(\hat{x}) = f(x^{(0)}) + Df(x^{(0)})(\hat{x} - x^{(0)}) + \underbrace{\mathcal{O}(\|\hat{x} - x^{(0)}\|^2)}_{\text{weglassen}}. \tag{8.8}$$

Weglassen des quadratischen Terms ergibt.

$$0 = f(x^{(0)}) + Df(x^{(0)})(x^{(1)} - x^{(0)}) \tag{8.9}$$

Falls $Df(x^{(0)})$ nichtsingulär, so folgt dass

$$Df(x^{(0)})(x^{(1)} - x^{(0)}) = -f(x^{(0)})$$

eine eindeutige Lösung $\delta^{(1)} = x^{(1)} - x^{(0)}$ hat und wir erhalten (die nächste Iterierte) $x^{(1)} = x^{(0)} + \delta^{(1)}$.

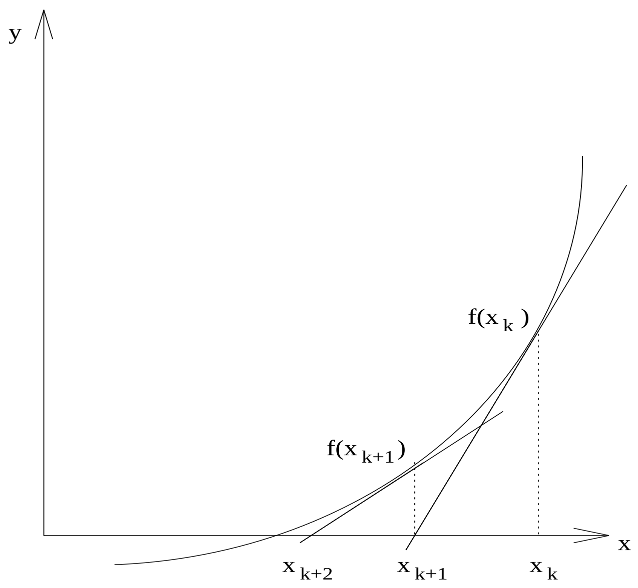
Theorem 115 *Sei $D \subseteq \mathbb{R}^m$ offen und sei D_0 konvex mit $\bar{D}_0 \subseteq D$. Die Funktion $f : D \to \mathbb{R}^m$ sei für alle $x \in D_0$ differenzierbar und für alle $x \in D$ stetig. Sei $x^{(0)} \in D_0$, so dass es Konstanten $r, \alpha, \beta, \gamma, h$ gibt mit*

$$\begin{aligned} S_r(x^{(0)}) &:= \left\{x \mid \|x - x^{(0)}\| < r\right\} \subseteq D_0 \\ h &= \alpha\beta\gamma/2 < 1 \\ r &= \alpha/(1-h), \end{aligned}$$

und $f(x)$ erfülle

$$\|Df(x) - Df(y)\| \leq \gamma\|x - y\|, \; \forall x, y \in D_0. \tag{8.10}$$

Newtonschritt



Die Ableitung

$$Df(x)^{-1} \text{ existiere } \forall x \in D_0 \text{ und es sei } \|Df(x)^{-1}\| \le \beta,\ \forall x \in D_0, \quad (8.11)$$

sowie

$$\|Df(x^{(0)})^{-1} f(x^{(0)})\| \le \alpha. \quad (8.12)$$

Dann gilt

a) *Ausgehend von $x^{(0)}$ ist jedes $x^{(i)}$, welches durch*

$$x^{(i+1)} = x^{(i)} - Df(x^{(i)})^{-1} f(x^{(i)}),\ i = 0, 1, 2, \dots$$

gegeben ist, wohldefiniert und es ist $x^{(i)} \in S_r(x^{(0)})$, $\forall\, i = 0, 1, 2, \dots$

b) $\lim\limits_{i\to\infty} x^{(i)} = \hat{x}$ *existiert und es ist $f(\hat{x}) = 0, \hat{x} \in \overline{S_r(x^{(0)})}$.*

c) *Für alle $i \ge 0$ gilt $\|x^{(i)} - \hat{x}\| \le \alpha \dfrac{h^{2^i - 1}}{1 - h^{2^i}}$*

Das Newton–Verfahren ist also mindestens quadratisch konvergent.

Beweis. a) Da $Df(x)^{-1}$ für alle $x \in D_0$ existiert, so ist $x^{(i+1)}$ für alle i dann wohldefiniert, wenn $x^{(i)} \in S_r(x^{(0)})$ für alle $i \geq 0$. Wegen Voraussetzung (8.12) gilt dies für $x^{(0)}, x^{(1)}$.
I.V. Sei nun $x^{(j)} \in S_r(x^{(0)})$ für $j = 0, 1, \ldots, k, k \geq 1$, so folgt aus (8.11)

$$\begin{aligned}
\|x^{(k+1)} - x^{(k)}\| &= \| - Df(x^{(k)})^{-1} f(x^{(k)})\| \leq \beta \|f(x^{(k)})\| \\
&= \beta \|f(x^{(k)}) - f(x^{(k-1)}) - Df(x^{(k-1)})(x^{(k)} - x^{(k-1)})\|,
\end{aligned}$$

da $f(x^{(k-1)}) + Df(x^{(k-1)})(x^{(k)} - x^{(k-1)}) = 0$.
Wende das folgende Lemma an: □

Lemma 116 *$Df(x)$ existiere für alle $x \in D_0$ mit $D_0 \subseteq \mathbb{R}^m$, D_0 konvex und es gebe ein γ, so dass*

$$\|Df(x) - Df(y)\| \leq \gamma \|x - y\|, \ \forall\, x, y \in D_0.$$

Dann gilt $\forall x, y \in D_0$ die Abschätzung

$$\|f(x) - f(y) - Df(y)(x - y)\| \leq \frac{\gamma}{2} \|x - y\|^2$$

Beweis. Sei $\varphi : [0, 1] \longrightarrow \mathbb{R}^m$ gegeben durch $\varphi(t) := f(y + t(x - y))$. φ ist differenzierbar $\forall t \in [0, 1], \forall x, y \in D_0$ und $\varphi'(t) = Df(y + t(x - y))(x - y)$. Dann gilt

$$\|\varphi'(t) - \varphi'(0)\| \leq \|Df(y + t(x - y)) - Df(y)\| \, \|x - y\| \leq \gamma t \|x - y\|^2.$$

Nun ist

$$P \equiv f(x) - f(y) - Df(y)(x - y) = \varphi(1) - \varphi(0) - \varphi'(0) = \int_0^1 (\varphi'(t) - \varphi'(0))\, dt,$$

also folgt

$$\begin{aligned}
\|P\| &\leq \int_0^1 \|\varphi'(t) - \varphi'(0)\|\, dt \\
&\leq \gamma \|x - y\|^2 \int_0^1 t\, dt \leq \frac{\gamma}{2} \|x - y\|^2
\end{aligned}$$

und damit die gewünschte Abschätzung. □

Nachdem wir das Lemma gezeigt haben, machen wir weiter im Beweis von Satz 115. Aus dem Lemmas folgt, dass

$$\|x^{(k+1)} - x^{(k)}\| \leq \frac{\beta\gamma}{2}\|x^{(k)} - x^{(k-1)}\|^2 \tag{8.13}$$

und damit

$$\|x^{(k+1)} - x^{(k)}\| \leq \alpha h^{2^k-1}. \tag{8.14}$$

Diese letzte Abschätzung folgt aus (8.12) für $k = 0$ und mit Induktion für $k+1$ aus (8.13). Aus (8.14) folgt

$$\begin{aligned} \|x^{(k+1)} - x^{(0)}\| &\leq \|x^{(k+1)} - x^{(k)}\| + \cdots + \|x^{(1)} - x^{(0)}\| \\ &\leq \alpha(1 + h + h^3 + h^7 + \cdots h^{2^k-1}) < \frac{\alpha}{1-h} = r, \end{aligned}$$

also $x^{(k+1)} \in S_r(x_0)$.

b) Aus (8.14) folgt, dass $x^{(k)}$ Cauchy–Folge ist, denn

$$\begin{aligned} \|x^{(n+k)} - x^{(n)}\| &\leq \sum_{i=1}^{k} \|x^{(n+i)} - x^{(n+i-1)}\| \\ &\leq \alpha \sum_{i=1}^{k} h^{2^{n+i-1}-1} \\ &\leq \alpha h^{2^n-1}(1 + h^{2^n} + (h^{2^n})^2 + \cdots) \\ &\leq \frac{\alpha h^{2^n-1}}{1 - h^{2^n}} < \varepsilon, \end{aligned} \tag{8.15}$$

für genügend großes $n \geq N(\varepsilon)$, weil $0 < h < 1$. Damit folgt

$$\lim_{k\to\infty} x^{(k)} = \hat{x} \in \overline{S_r(x_0).}$$

Aus (8.15) folgt:

$$\|\hat{x} - x^{(n)}\| = \lim_{m\to\infty} \|x^{(m)} - x^{(n)}\| \leq \frac{\alpha h^{2^n-1}}{1 - h^{2^n}}.$$

Noch zu zeigen ist, dass $\hat{x}$ Nullstelle von f in $\overline{S_r(x_0)}$ ist. Da $x^{(k)} \in S_r(x^{(0)})\ \forall\, k \geq 0$, folgt dass

$$\|Df(x^{(k)}) - Df(x^{(0)})\| \leq \gamma\|x^{(k)} - x^{(0)}\| < \gamma r.$$

Also gilt

$$\|Df(x^{(k)})\| \leq \gamma r + \|Df(x^{(0)})\| =: \kappa.$$

Aus der Iterationsgleichung folgt

$$\|f(x^{(k)})\| \leq \kappa \|x^{(k+1)} - x^{(k)}\|.$$

und damit

$$\lim_{k\to\infty} \|f(x^{(k)})\| = 0$$

und da f stetig ist gilt $\lim\limits_{k\to\infty} f(x^{(k)}) = f(\hat{x})$. □ Man kann unter noch stärkeren Voraussetzungen mit dem Satz von Newton–Kantorovich zeigen, dass $\hat{x}$ die einzige Nullstelle in $S_r(x^{(0)})$ ist.
Voraussetzung dafür, dass das Newton–Verfahren konvergiert, ist, dass der Startwert genügend nahe bei der gesuchten Lösung liegt. Um globale Konvergenz zu erzielen, wird das Newton–Verfahren durch Einführung eines Parameters modifiziert. Setze

$$x^{(k+1)} = x^{(k)} - \lambda_k d^{(k)}, d^{(k)} := Df(x^{(k)})^{-1} f(x^{(k)}). \tag{8.16}$$

Die λ_k werden dabei so gewählt, dass die Folge $\{h(x^{(k)})\}$ mit $h(x) = f(x)^T f(x)$ streng monoton fällt und die $x^{(k)}$ gegen ein Minimum von $h(x)$ konvergieren.

Was hat das mit Nullstelle zu tun? Da

$$\begin{array}{lll} h(x) \geq 0 & & \forall x, \\ h(\hat{x}) = 0 & \iff & f(\hat{x}) = 0 \quad (f(x)^T f(x) = \|f(x)\|_2^2), \end{array}$$

so ist jedes lokale Minimum $\hat{x}$ von h mit $h(\hat{x}) = 0$ auch globales Minimum von h und Nullstelle von f.

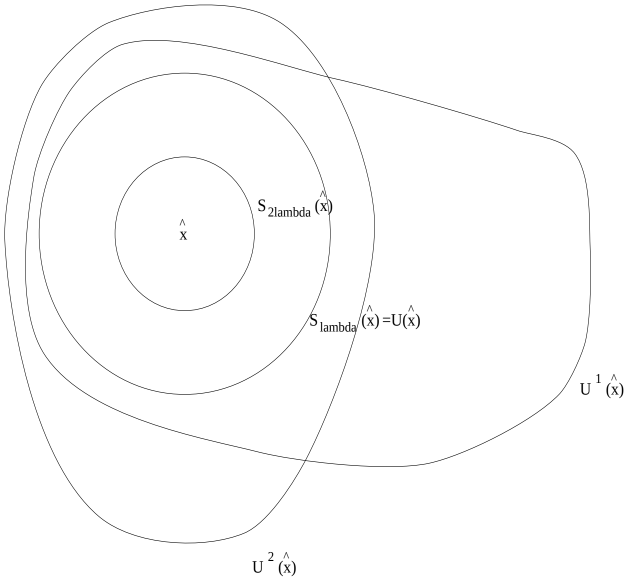
Definiere die Menge

$$D(\gamma, x) = \{s \in \mathbb{R}^m \mid \|s\| = 1 \text{ und } Dh(x)^T s \geq \gamma \|Dh(x)\|\}, \gamma > 0. \tag{8.17}$$

Lemma 117 *Sei $h : \mathbb{R}^m \to \mathbb{R}$ eine Funktion, deren Gradient $Dh(x)$ für alle $x \in V(\hat{x})$ ($V(\hat{x})$ Umgebung von $\hat{x}$) definiert und stetig ist. Sei ferner $Dh(\hat{x}) \neq 0$ und $\gamma > 0$. Dann gibt es eine Umgebung $U(\hat{x}) \subseteq V(\hat{x})$ von $\hat{x}$ und ein $\lambda > 0$, so dass*

$$h(x - \mu s) \leq h(x) - \frac{\mu\gamma}{4} \|Dh(\hat{x})\|,$$

für alle $x \in U(\hat{x}), s \in D(\gamma, x)$ und $0 \leq \mu \leq \lambda$.
(zeigt wann man $h(y) < h(x)$ finden kann.)



Beweis. Sei

$$U^1(\hat{x}) = \{x \in V(\hat{x}) | \, \|Dh(x) - Dh(\hat{x})\| < \frac{\gamma}{4} \|Dh(\hat{x})\|\}.$$

Da $Dh(\hat{x}) \neq 0$ und $Dh(x)$ stetig auf $V(\hat{x})$, so folgt $U^1(\hat{x}) \neq \emptyset$ und U^1 ist Umgebung von $\hat{x}$. Aus demselben Grund folgt auch, dass

$$U^2(\hat{x}) = \{x \in V(\hat{x}) | \, D(\gamma, x) \subseteq D(\frac{\gamma}{2}, \hat{x})\}$$

eine Umgebung von $\hat{x}$ ist. Wähle nun $\lambda > 0$, so dass

$$\overline{S_{2\lambda}(\hat{x})} := \{x | \, \|x - \hat{x}\| \leq 2\lambda\} \subseteq U^1(\hat{x}) \cap U^2(\hat{x})$$

und setze

$$U(\hat{x}) = \overline{S_\lambda(\hat{x})} = \{x | \, \|x - \hat{x}\| \leq \lambda\}.$$

Dann gilt für alle $x \in U(\hat{x})$ mit $0 \leq \mu \leq \lambda, s \in D(\gamma, x)$, dass

$$\begin{aligned} h(x) - h(x - \mu s) &= \mu Dh(x - \theta \mu s)^T s \\ &= \mu \left[\left(Dh(x - \theta \mu s) - Dh(\hat{x})\right)^T s + Dh(\hat{x})^T s \right], \end{aligned}$$

für ein θ mit $0 < \theta < 1$. Mit $x \in U(\hat{x})$ gilt auch

$$x, x - \mu s, x - \theta \mu s \in U^1 \cap U^2,$$

also folgt

$$\begin{aligned} h(x) - h(x - \mu s) &\geq -\frac{\mu\gamma}{4}\|Dh(\hat{x})\| + \mu Dh(\hat{x})^T s \\ &\geq \frac{-\mu\gamma}{4}\|Dh(\hat{x})\| + \mu\frac{\gamma}{2}\|Dh(\hat{x})\| \\ &= \frac{\mu\gamma}{4}\|Dh(\hat{x})\|. \end{aligned}$$

□

Algorithmus 25

Input: *Differenzierbare Funktion* $h(x) : \mathbb{R}^m \to \mathbb{R}$, *und Zahlen* $\sigma_k, \gamma_k,\ k = 0, 1, 2, \ldots$ *mit*

$$\inf_k(\gamma_k) > 0, \inf_k \sigma_k > 0,$$

sowie einen Startwert $x^{(0)} \in \mathbb{R}^m$.

Output: *Folge* $x^{(k)},\ k = 0, 1, 2, \ldots,$ *die gegen Minimum von* $h(x)$ *konvergiert.*

FOR $k = 0, 1, 2, \ldots$ *UNTIL SATISFIED*

Sei $s^{(k)} \in D(\gamma_k, x^{(k)})$,

$$x^{(k+1)} = x^{(k)} - \lambda_k s^{(k)}, \tag{8.18}$$

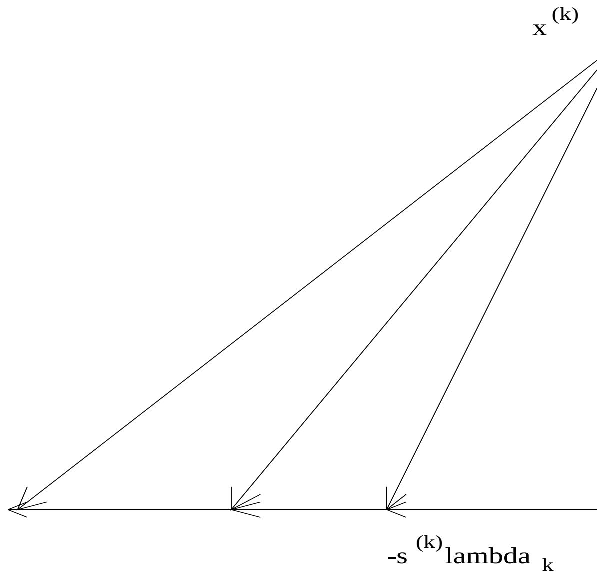
wobei $\lambda_k \in [0, \sigma_k\|Dh(x^{(k)})\|]$, *so dass*

$$h(x^{(k+1)}) = \min_\mu\{h(x^{(k)} - \mu s^{(k)})\,|\, 0 \leq \mu \leq \sigma_k\|Dh(x^{(k)})\|\}.$$

END

In jedem Schritt wird $\min \varphi(\mu) = \min h(x^{(k)} + \mu s^{(k)}),\ \mu \in [0, \sigma_k\|Dh(x^{(k)})\|]$ über λ_k gesucht. Das ist jetzt eine eindimensionale Minimierung, und damit zwar einfacher als $\min h(x)$ aber immer noch sehr aufwendig. Dies wird später noch vereinfacht.

Was können wir nun über die Konvergenz von (8.18) aussagen.



Theorem 118 *Seien $h : \mathbb{R}^m \to \mathbb{R}$ und $x^{(0)} \in \mathbb{R}^m$, so dass*

a) $K := \{x|\ h(x) \le h(x^{(0)})\}$ *kompakt.*

b) *h stetig differenzierbar in einer Umgebung von K.*

Dann gilt für jede Folge $\{x^{(k)}\}$ in (8.18)

1) $x^{(k)} \in K$ *für alle* $k = 0, 1, 2, \ldots$.

2) $\left\{x^{(k)}\right\}$ *besitzt mindestens einen Häufungspunkt $\hat{x}$ in K.*

3) Jeder Häufungspunkt $\hat{x}$ von $\left\{x^{(k)}\right\}$ erfüllt $Dh(\hat{x}) = 0$. (Stationärer Punkt)

Beweis. Aus der Definition von $x^{(k)}$ folgt $h(x^{(0)}) \ge h(x^{(1)}) \ge \cdots$ und damit $x^{(k)} \in K$ für alle k und da K kompakt ist, gibt es einen Häufungspunkt $\hat{x}$ in K. Also folgt 1), 2).

Teil 3) beweisen wir mit Widerspruch. Angenommen $Dh(\hat{x}) \neq 0$, und sei o.B.d.A. $\lim_{k\to\infty} x^{(k)} = \hat{x}$ (sonst nehme am einfach eine Teilfolge). Setze $\gamma = \inf_k \gamma_k > 0, \sigma = \inf_k \sigma_k > 0$. Nach Lemma 117 gibt es $U(\hat{x})$ und $\lambda > 0$, so dass

$$h(x-\mu s) \le h(x)-\mu\frac{\gamma}{4}\|Dh(\hat{x})\|,\ \forall x \in U(\hat{x}),\ s \in D(\gamma, x),\ 0 \le \mu \le \lambda. \quad (8.19)$$

Da $\lim_k x^{(k)} = \hat{x}$ und $Dh(x)$ stetig ist, gibt es k_0, so dass für alle $k \geq k_0$ gilt

$$x^{(k)} \in U(\hat{x}), \ \|Dh(x^{(k)})\| \geq \frac{1}{2}\|Dh(\hat{x})\|.$$

Sei nun

$$\begin{aligned} \Lambda &= \min\{\lambda, \frac{1}{2}\sigma\|Dh(\hat{x})\|\}, \\ \varepsilon &= \Lambda\frac{\gamma}{4}\|Dh(\hat{x})\| > 0 \end{aligned}$$

Wegen $\sigma_k \geq \sigma$ folgt, dass

$$[0, \Lambda] \subseteq [0, \sigma_k\|Dh(\hat{x})\|], \ \forall\, k \geq k_0,$$

also $h(x^{(k+1)}) \leq \min_\mu \{h(x^{(k)} - \mu s^{(k)})|\ 0 \leq \mu \leq \Lambda\}$. Aus (8.19) folgt dann

$$h(x^{(k+1)}) \leq h(x^{(k)}) - \frac{\Lambda\gamma}{4}\|Dh(\hat{x})\| = h(x^{(k)}) - \varepsilon, \ \forall\, k \leq k_0,$$

da $\Lambda \leq \lambda, \ x^{(k)} \in U(\hat{x}), \ s^{(k)} \in D(\gamma_k, x^{(k)}) \subseteq D(\gamma, x^{(k)})$. Damit ergibt sich

$$\lim_{k\to\infty} h(x^{(k)}) = -\infty,$$

Dies ist ein Widerspruch, denn $h(x^{(k)}) \geq h(x^{(k+1)}) \geq \cdots \geq h(\hat{x})$. □ Dies

ist ein allgmeines Minimierungsverfahren, was sehr viele Anwendungen hat, benötigt wird noch eine Methode für die eindimensionale Minimierung. Diese wird in Algorithmus 25 durch einen endlichen Suchprozess ersetzt.

a) Wähle $s^{(k)} \in D(\gamma_k, x^{(k)})$ und definiere

$$\begin{aligned} \rho_k &:= \sigma_k\|Dh(x^{(k)})\| \\ h_k(\mu) &= h(x^{(k)} - \mu s^{(k)}) \end{aligned}$$

und bestimme die kleinste Zahl $j \geq 0$ mit

$$h_k(\rho_k 2^{-j}) \leq h_k(0) - \rho_k 2^{-j}\frac{\gamma_k}{4}\|Dh(x^{(k)})\|.$$

b) Bestimme λ_k und damit

$$\begin{aligned} x^{(k+1)} &:= x^{(k)} - \lambda_k s^{(k)}, \quad \text{so dass gilt} \\ h(x^{(k+1)}) &= \min_{0\leq i\leq j} h_k(\rho_k 2^{-i}). \end{aligned}$$

Das gesuchte j existiert. Ist $x^{(k)}$ stationär, so ist $j = 0$, andernfalls wende Lemma 117 auf $\hat{x} = x^{(k)}$ an. In jedem Fall kann j und damit auch λ_k in einem endlichen Suchprozess bestimmt werden und es gilt Satz 118 auch für die so bestimmte Folge.

8.2.1 Das modifizierte Newtonverfahren

Wir wenden nun zur Lösung von $f(x) = 0$ das Minimierungsverfahren 25 auf $h(x) = f(x)^T f(x)$ an.
Als Suchrichtung $s^{(k)}$ in Punkt $x^{(k)}$ wählen wir den normierten Newtonvektor

$$s^{(k)} = \frac{d^{(k)}}{\|d^{(k)}\|} \text{ mit } d^{(k)} = Df(x^{(k)})^{-1} f(x^{(k)}),$$

dieser ist immer definiert, falls $Df(x^{(k)})^{-1}$ existiert.

Lemma 119 *Sei x so, dass $d(x) = Df(x)^{-1}f(x)$ existiert und nicht verschwindet. Dann gilt*

$$s(x) = \frac{d(x)}{\|d(x)\|} \in D(\gamma, x), \text{ für alle } 0 < \gamma < \bar{\gamma}(x),$$

wobei

$$\bar{\gamma}(x) = \frac{1}{\kappa_2(Df(x))} = \frac{1}{\|Df(x)\|_2 \|Df(x)^{-1}\|_2}.$$

Beweis. $Dh(x) = 2f^T(x)Df(x)$. Submultiplikativität von $\| \bullet \|_2$ liefert

$$\begin{aligned} \|f^T(x)Df(x)\|_2 &\leq \|Df(x)\|_2 \, \|f(x)\|_2 \\ \|Df(x)^{-1}f(x)\|_2 &\leq \|Df(x)^{-1}\|_2 \, \|f(x)\|_2 \end{aligned}$$

und daher

$$\begin{aligned} \frac{Dh(x)s}{\|Dh(x)\|} &= \frac{f(x)^T Df(x) Df(x)^{-1} f(x)}{\|Df(x)^{-1}f(x)\| \, \|f^T(x)Df(x)\|} \\ &\geq \frac{1}{\kappa_2(Df(x))} \\ &> 0. \end{aligned}$$

Für alle γ mit $0 < \gamma < \frac{1}{\kappa_2(Df(x))}$ gilt also $s \in D(\gamma, x)$. $\square$ Falls $Df(x)^{-1}$ existiert folgt also, daß $Dh(x) = 0$ genau dann wenn $f(x) = 0$. Damit erhalten wir das modifizierte Newtonverfahren.

Algorithmus 26

Input: *Differenzierbare Funktion* $f(x) : \mathbb{R}^m \to \mathbb{R}^m$ *sowie Startwert* $x^{(0)} \in \mathbb{R}^m$.

Output: *Folge* $x^{(k)}$, $k = 0, 1, 2, \ldots$ *die gegen Nullstelle von* f *konvergiert.*

FOR $k = 0, 1, 2, \ldots$ *UNTIL SATISFIED*

Berechne $d^{(k)} = Df(x^{(k)})^{-1} f(x^{(k)})$, *(lineares Gleichungssystem mit QR)*

$\gamma_k = \frac{1}{\kappa_2(Df(x^{(k)}))}$, *(fast frei aus QR).*

Setze $h_k(\tau) = h(x^{(k)} - \tau d^{(k)}) = f(x^{(k)} - \tau d^{(k)})^T f(x^{(k)} - \tau d^{(k)})$.

Bestimme $j \geq 0$, *so dass* $h_k(2^{-j}) \leq h_k(0) - 2^{-j} \frac{\gamma_k}{4} \|d^{(k)}\| \, \|Dh(x^{(k)})\|$.

Bestimme λ_k *und* $x^{(k+1)} = x^{(k)} - \lambda_k d^{(k)}$, *so daß gilt*

$$h(x^{(k+1)}) = \min_{0 \leq i \leq j} h_k(2^{-i}).$$

END

Als Kosten fallen pro Schritt an:

a) Berechnung von $Df(x^{(k)})$.

b) Lösung von $Df(x^{(k)}) d^{(k)} = f(x^{(k)})$.

c) eindimensionale Minimierung.

Die Fehleranalyse für Teil b) ist klar. Die für. a) und c) hängt von f ab.

Über dieses Verfahren hat man die folgende Konvergenzaussage:

Theorem 120 *Gegeben sei* $f : \mathbb{R}^m \to \mathbb{R}^m, x^{(0)} \in \mathbb{R}^m$ *mit*

a) $K = \{x | \ h(x) \leq h(x_0)\}$ *ist kompakt, wobei* $h(x) = f^T(x) f(x)$

b) f *ist auf Umgebung von* K *stetig differenzierbar.*

c) $Df(x)^{-1}$ *existiert für alle* $x \in K$.

Dann ist die Folge $\{x^{(k)}\}$*, die in Algorithmus 26 erzeugt wird, wohldefiniert und es gilt*

a) $x^{(k)} \in K,\ \forall k = 0, 1, 2, \ldots, \{x^{(k)}\}$ *besitzt mindestens Häufungspunkt* $\hat{x} \in K$.

b) Jeder Häufungspunkt $\hat{x}$ *von* $\{x^{(k)}\}$ *ist Nullstelle von* f.

Beweis. Den Beweis machen wir hier nicht, siehe z.B. das Buch von Stoer.
□

8.2.2 Praktische Realisierung des modifizierten Newton–Verfahrens

Ein Problem bei der Durchführung des Newtonverfahrens ist, dass in jedem Schritt $Df(x^{(k)})$ ausgerechnet werden muß. Man kann nun $Df(x^{(k)})$ durch die Differenzenquotientenmatrix $\triangle f(x^{(k)}) = (\triangle_1 f, \cdots, \triangle_m f)(x^{(k)})$, approximieren, wobei

$$\begin{aligned} \triangle_i f(x) &= \frac{f(x_1, \ldots, x_{i-1}, x_i + h_i, x_{i+1}, \ldots, x_m) - f(x_1, \ldots, x_m)}{h_i} \\ &= \frac{f(x + h_i e_i) - f(x)}{h_i}. \end{aligned}$$

Falls h_i zu groß ist so erhalten wir eine schlechte Approximation an $Df(x)$, ist h_i zu klein, dann gilt $f(x + he_i) \approx f(x)$.
Wähle h_i so, dass $f(x)$ und $f(x+he_i)$ ungefähr die $\frac{t}{2}$ ersten Stellen gemeinsam haben (t–stellige Rechnung)

$$|h_i| \approx \sqrt{eps}\|f(x)\|/\| \triangle_i f(x)\|$$

Die auftretende Auslöschung ist dann noch erträglich.

Im allgemeinen ist jedoch auch die Berechnung von $\triangle f$ zu teuer daher verwende folgende Vereinfachung mit dem Satz von Broyden.

Theorem 121 *Seien* $A, B \in \mathbb{R}^{n,n}, b \in \mathbb{R}^m, F : \mathbb{R}^m \to \mathbb{R}^m$ *gegeben durch* $F(u) = Au + b,\ x, x' \in \mathbb{R}^m$ *und* $p = x - x',\ q = F(x') - F(x) = Ap$*. Dann gilt für die Rang–1–Modifikation* $B' = B + \frac{1}{p^T p}(q - Bp)p^T$ *die Abschätzung*

$$\|B' - A\|_2 \leq \|B - A\|_2$$

und $B'p = Ap = q$.

Beweis. $(B' - A)p = Bp + (q - Bp) - Ap = 0$.
Für jedes $u \in \mathbb{R}^m$ mit $\|u\|_2 = 1$ existiert ein v mit

$$u = \alpha p + v,\ v^T p = 0,\ \|v\|_2 \leq 1.$$

Also folgt mit $\|u\|_2 = 1$, dass

$$\|(B' - A)u\|_2 = \|(B' - A)v\|_2 = \|(B - A)v\|_2 \leq \|B - A\|_2\ \|v\|_2 \leq \|B - A\|_2.$$

Also gilt dies auch für $\|B' - A\|_2$, welches das Supremum über alle u ist. □
Es folgt, dass $DF(x) = A$ von B' genau so gut approximiert wird wie von B.

B' und $DF(x)$ transformieren p in denselben Vektor. Die Idee ist nun, $f(x)$ durch eine affine Abbildung zu approximieren (in der Nähe einer Nullstelle) und diese Idee anzuwenden.
Ersetze in Algorithmus 26 den Teil in der Schleife.

Algorithmus 27 (Broyden–Rang 1–Verfahren)
Input: $f(x) : \mathbb{R}^m \to \mathbb{R}^m$ *differenzierbar und* $x^{(0)} \in \mathbb{R}^m$
Output: *Folge* $x^{(k)}, k = 0, 1, 2, \ldots$

FOR $k = 0, 1, 2, \ldots$ *UNTIL SATISFIED*
$d^{(k)} = B_k^{-1} f(x^{(k)})$
$x^{(k+1)} = x^{(k)} - \lambda_k d^{(k)}$
$p^{(k)} = x^{(k+1)} - x^{(k)}$
$q^{(k)} = f^{(x^{(k+1)})} - f(x^{(k)})$
$B^{(k+1)} = B_k + \frac{1}{p^{(k)^T} p^{(k)}}(q^{(k)} - B_k p^{(k)}) p^{(k)^T}$
END

λ_k wird durch näherungsweise Minimierung von $\|f(x^{(k+1)})\|^2 \approx \min\limits_{\lambda \geq 0} \|f(x^{(k)} - \lambda d^{(k)})\|^2$ bestimmt wie vorher.
Es gilt $B_0 \approx \triangle f(x^{(0)})$. Praktische Erfahrung zeigt, dass man diese Wahl von B_{k+1} nur für $0.5 \leq \lambda_k \leq 1$ machen sollte, sonst
wählt man besser $B_{k+1} = \triangle f(x_{k+1})$.
Die Lösung des linearen Gleichungssystems

$$B_k d^{(k)} = f(x^{(k)})$$

kann über eine Rang 1 Aufdatierungsformel sehr leicht gemacht werden.

Kapitel 9

Eigenwertprobleme

In diesem Kapitel werden wir uns mit der numerischen Lösung von Eigenwertproblemen beschäftigen. Zuerst einige Grundlagen zur Wiederholung. Eigenwerte einer Matrix $A = [a_{ij}]_{i,j=1,\ldots,n} \in \mathbb{C}^{n,n}$ sind die n Nullstellen $\lambda_1, \cdots, \lambda_n$ des *charakteristischen Polynoms* $p(z) = \det(zI - A)$. Das *Spektrum einer Matrix* ist $\sigma(A) = \{\lambda_1, \cdots, \lambda_n\}$. Es gilt $\det(A) = \lambda_1 \cdots \lambda_n$,

Eine Matrix heißt *normal* falls $A^*A = AA^*$.

Sei $\lambda \in \sigma(A)$, dann heißen Vektoren $x \in \mathbb{C}^n \setminus \{0\}$ die $Ax = \lambda x$ erfüllen, *Eigenvektoren.* Ein Unterraum $S \in \mathbb{C}^n$ mit der Eigenschaft, dass falls $x \in S$ dann auch $Ax \in S$ heißt *invariant.* Falls $AX = XB$ mit $B \in C^{k,k}$, $X \in \mathbb{C}^{n,k}$ und $\operatorname{rang} X = k$, so spannen die Spalten von X einen k dimensionalen invarianten Unterraum auf und $By = \lambda y$ impliziert, dass $AXy = \lambda Xy$.

Falls X vollen Spaltenrang hat, so gilt daher $\sigma(B) \subset \sigma(A)$. Falls X quadratisch ist, so folgt $\sigma(B) = \sigma(A)$, und $A, B = X^{-1}AX$ heißen *ähnlich.*

Lemma 122 *Seien $A \in \mathbb{C}^{n,n}$, $B \in \mathbb{C}^{k,k}$ und $X \in \mathbb{C}^{n,k}$ mit $AX = XB$ und* $\operatorname{rang}(X) = k$. *Dann gibt es eine unitäre Matrix $Q \in \mathbb{C}^{n,n}$, so dass*

$$Q^*AQ = T = \begin{array}{cc} \begin{bmatrix} T_{11} & T_{12} \\ 0 & T_{22} \end{bmatrix} & \begin{array}{c} k \\ n-k \end{array} \end{array} \tag{9.1}$$

und $\sigma(T_{11}) = \sigma(A) \cap \sigma(B)$.

Beweis. Sei $X = Q \begin{bmatrix} R \\ 0 \end{bmatrix}$ eine QR–Zerlegung von X. Dann gilt

$$AQ \begin{bmatrix} R \\ 0 \end{bmatrix} = Q \begin{bmatrix} R \\ 0 \end{bmatrix} B$$

genau dann wenn

$$(Q^*AQ)\begin{bmatrix} R \\ 0 \end{bmatrix} = \begin{bmatrix} R \\ 0 \end{bmatrix} B =: \begin{bmatrix} T_{11} & T_{12} \\ T_{21} & T_{22} \end{bmatrix} \begin{bmatrix} R \\ 0 \end{bmatrix} \begin{array}{c} p \\ n-p \end{array}.$$

Die Matrix R ist nichtsingulär, da Rang $(X) = k$. Aus $T_{21}R = 0 \cdot B = 0$ folgt, dass$T_{21} = 0$ und aus $T_{11}R = RB$ folgt, dass $\sigma(T_{11}) = \sigma(B)$. Daher gilt $\sigma(A) = \sigma(T) = \sigma(T_{11}) \cup \sigma(T_{22})$ und damit die Behauptung. □

Theorem 123 (Satz von Schur) *Sei $A \in \mathbb{C}^{n,n}$. Dann gibt es eine unitäre Matrix $Q \in \mathbb{C}^{n,n}$, so dass*

$$Q^*AQ = T = D + N, \tag{9.2}$$

mit $D = \operatorname{diag}(\lambda_1, \cdots, \lambda_n)$, und N strikte obere Dreiecks–Matrix. Q kann so gewählt werden, dass die Eigenwerte in beliebiger Folge auf der Diagonale stehen.

Beweis. Beweis durch Induktion. $n = 1$ trivial. Sei die Behauptung richtig für $n-1$. Sei $Ax = \lambda x$, für $x \neq 0$. Mit Lemma 122 für $B = \lambda$ gibt es unitäres U, so dass

$$U^*AU = \begin{bmatrix} \lambda & w^* \\ 0 & c \end{bmatrix} \begin{array}{c} 1 \\ n-1. \end{array}$$

Nach Induktionsvoraussetzung gibt es $\tilde{U}$, so dass $\tilde{U}^*C\tilde{U}$ obere Dreiecks–Matrix ist. Sei

$$Q = U \begin{bmatrix} T & \tilde{w}^* \\ 0 & \tilde{U} \end{bmatrix},$$

dann ist Q^*AQ obere Dreiecks–Matrix. □

Die Spalten q_i von $Q = [q_1, \cdots, q_n]$ heißen *Schur–Vektoren*, und spannen den invarianten Unterraum $S_k = \operatorname{span}\{q_1, \cdots, q_k\}$ auf.

Corollary 124 *$A \in \mathbb{C}^{n,n}$ ist normal genau dann wenn es eine unitäre Matrix $Q \in \mathbb{C}^{n,n}$ gibt, so dass*

$$Q^*AQ = \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{bmatrix}.$$

Beweis. Die Matrix A ist normal genau dann wenn Q^*AQ normal ist für alle unitären Matrizen Q, denn

$$(Q^*AQ)^*(Q^*AQ) = Q^*A^*AQ = Q^*AA^*Q = (Q^*AQ)(Q^*AQ)^*.$$

Ist $Q^*AQ = T$ obere Dreiecks–Matrix und normal, so ist T diagonal. □

Es gibt eine reelle Variante des Satzes von Schur:

Theorem 125 *Für jedes $A \in \mathbb{R}^{n,n}$ gibt es eine orthogonale Matrix $Q \in \mathbb{R}^{n,n}$, so dass*

$$Q^TAQ = \begin{bmatrix} R_{11} & \cdots & R_{1m} \\ & \ddots & \vdots \\ & & R_{mm} \end{bmatrix}$$

wobei jedes R_{ii} entweder 1×1 oder 2×2 mit komplex–konjugierten Eigenwerten ist.

Beweis. Übung. □

Theorem 126 (Jordansche Normalform) *Sei $A \in \mathbb{C}^{n,n}$. Dann gibt es $X \in C^{n,n}$, so dass $X^{-1}AX = diag\ (J_1, \cdots, J_t)$, wobei*

$$J_i = \begin{bmatrix} \lambda_i & 1 & & \\ & \ddots & \ddots & \\ & & \ddots & 1 \\ & & & \lambda_i \end{bmatrix} \in \mathbb{C}^{m_i, m_i}$$

und $m_1 + \cdots + m_t = n$.

Beweis. Siehe Lineare Algebra Vorlesung. □

Die Summe der Dimensionen m_i zu einem Eigenwert λ heißt *algebraische Vielfachheit*. Falls diese 1 ist, so heißt λ einfacher Eigenwert. Die Anzahl der Jordanblöcke zu λ heißt *geometrische Vielfachheit*. Es gilt immer, dass die algebraische Vielfachheit größer oder gleich der geometrischen Vielfachheit ist. Falls die algebraische Vielfachheit echt größer ist als die geometrische Vielfachheit, so heißt λ *defektiver Eigenwert* und A *defektiv*.

Corollary 127 *Die Matrix $A \in \mathbb{C}^{m,n}$ ist nicht defektiv genau, wenn es eine nichtsinguläre Matrix $X \in \mathbb{C}^{n,n}$ gibt, so dass*

$$X^{-1}AX = \; diag(\lambda_1, \ldots, \lambda_n). \tag{9.3}$$

Beweis. Klar aus Satz 126. □

Die Jordanform ist numerisch schwer zu berechnen. Kleine Störungen in der Matrix ändern die Jordanform stark, auf dem Rechner ist jede Matrix diagonalisierbar, diagonalisierbare Matrizen liegen dicht in der Menge der Matrizen. Matrizen die fast defektiv sind, können sehr schlecht konditionierte Eigenvektormatrizen haben. Und dies kann drastische Folgen haben, falls man nichtunitäre Transformationen zur Eigenwertberechnung verwendet. Denn es gilt

$$gl(X^{-1}AX) = X^{-1}AX + E \tag{9.4}$$

wobei

$$\|E\|_2 \cong \text{ eps } \kappa_2(X)\|A\|_2 \tag{9.5}$$

Falls X nicht unitär ist, so kann E sehr groß werden.

Beispiel 128 *Betrachte*

$$A = \begin{bmatrix} 5 & -1 \\ 2 & 2 \end{bmatrix}$$

und die Transformationsmatrizen auf Schur- und Jordanform

$$Q = \begin{bmatrix} 0.6 & 0.8 \\ -0.8 & 0.6 \end{bmatrix}, \quad X = \begin{bmatrix} 1.0 & 1.01 \\ 2. & 2.00 \end{bmatrix}.$$

Dann gilt $\kappa_2(X) = 167$, $\kappa_2(Q) = 1$ und $\sigma(A) = \{3, 4\}$.

Mit dreistelliger Arithmetik ergibt sich mit Rundung

$$\begin{aligned} \|gl(X^{-1}AX) - X^{-1}AX\|_2 &\cong 1.9, \\ \|gl(Q^*AQ) - Q^*AQ\|_2 &\cong 10^{-3}. \end{aligned}$$

Theorem 129 (Satz von Gerschgorin) *Sei*

$$A = D + F \text{ mit } D = diag(d_1, \cdots, d_n)$$

und F habe Nulldiagonale. Dann gilt

$$\sigma(A) \subseteq \bigcup_{i=1}^{n} D_i$$

wobei

$$D_i = \left\{ z \in \mathbb{C} : \ |z - d_i| \le \sum_{j=1}^{n} |f_{ij}| \right\}.$$

Beweis. Übung. □

Theorem 130 (Satz von Bauer–Fike.) *Sei μ ein Eigenwert von $A+E \in \mathbb{C}^{n,n}$ und $X^{-1}AX = D = diag\,(\lambda_1, \cdots, \lambda_n)$ diagonal, dann gilt*

$$\min_{\lambda \in \sigma(A)} |\lambda - \mu| \le \kappa_p(X) \|E\|_p$$

Beweis. Übung. □

Der Satz von Bauer–Fike besagt, dass $\kappa(X) \cdot \|E\|$ große Störungen in den Eigenwerten auftreten können. Falls X orthogonal ist, so ist $\kappa_2(X) = 1$.

Störungsresultate für invariante Unterräume sind im allgemeinen sehr schwer.

Theorem 131 (Satz von Stewart) *Sei Q unitär, so dass*

$$Q^* A Q = \begin{array}{cc} \begin{bmatrix} T_{11} & T_{12} \\ 0 & T_{22} \end{bmatrix} & \begin{array}{c} p \\ n-p \end{array} \end{array}$$

eine Schur–Zerlegung mit $Q = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix}$ ist. Sei

$$\operatorname{sep}(T_{11}, T_{22}) = \min_{X \neq 0} \frac{\|T_{11}X - XT_{22}\|_F}{\|X\|_F}$$

und sei $E \in \mathbb{C}^{n,n}$ eine Störung. Sei

$$Q^* E Q = \begin{array}{cc} \begin{bmatrix} E_{11} & E_{12} \\ E_{21} & E_{22} \end{bmatrix} & \begin{array}{c} p \\ n-p. \end{array} \end{array}$$

Falls $\delta = \operatorname{sep}(T_{11}, T_{22}) - \|E_{11}\|_2 - \|E_{22}\|_2 > 0$ und $\|E_{21}\|_2(\|T_{12}\|_2 + \|E_{21}\|_2) \le \frac{\delta^2}{4}$, so gibt es $P \in \mathbb{C}^{n-k,k}$ mit $\|P\|_2 \le 2\|E_{21}\|_2/\delta$ und die Spalten von

$$\hat{Q} = (Q_1 + Q_2 P)(I + P^* P)^{-\frac{1}{2}}$$

bilden eine Orthonormalbasis für den invarianten Unterraum von $A + E$.
Beweis. Siehe zum Beispiel das Das Buch von Golub/Van Loan.

9.1 Der QR–Algorithmus für unsymmetrische Matrizen

Wir betrachten hier nur den reellen Fall, der Berechnung der reellen Schur–Form.

Der beste Algorithmus zur Berechnung der Schur–Form ist der QR–Algorithmus von Francis, der im Prinzip die folgende Iterationsschrift ausführt.

Sei $H_0 = Q_0^T A Q_0$ für noch zu bestimmende unitäre Matrix Q_0.

FOR $k = 1, 2, \cdots$

Berechne eine QR–Zerlegung

$$\begin{aligned} H_{k-1} &= Q_k R_k \quad \text{und setze} \\ H_k &= R_k Q_k. \end{aligned}$$

END

Diese Iteration würde pro Schritt $\mathcal{O}(n^3)$ flops kosten (bei ungefähr $2n$ Iterationen) also $\mathcal{O}(n^4)$ flops insgesamt. Um das zu verkleinern, verwenden wir Q_0, um A zuerst auf obere Hessenbergform zu transformieren. Dies geschieht mit Hilfe von Householder–Transformationen (HOUSE, COLHOUSE, ROWHOUSE).

$$A = \begin{bmatrix} x & x & x & x & x & x \\ x & x & x & x & x & x \\ x & x & x & x & x & x \\ x & x & x & x & x & x \\ x & x & x & x & x & x \\ x & x & x & x & x & x \end{bmatrix} \to P_1^T A P_1 = \begin{bmatrix} x & x & x & x & x & x \\ x & x & x & x & x & x \\ 0 & x & x & x & x & x \\ 0 & x & x & x & x & x \\ 0 & x & x & x & x & x \\ 0 & x & x & x & x & x \end{bmatrix} \to$$

$$P_2^T P_1^T A P_1 P_2 = \begin{bmatrix} x & x & x & x & x & x \\ x & x & x & x & x & x \\ 0 & x & x & x & x & x \\ 0 & 0 & x & x & x & x \\ 0 & 0 & x & x & x & x \\ 0 & 0 & x & x & x & x \end{bmatrix} \to$$

$$P_3^T P_2^R P_1^T A P_1 P_2 P_3 = \begin{bmatrix} x & x & x & x & x & x \\ x & x & x & x & x & x \\ 0 & x & x & x & x & x \\ 0 & 0 & x & x & x & x \\ 0 & 0 & 0 & x & x & x \\ 0 & 0 & 0 & x & x & x \end{bmatrix} \to P_4^T P_3^T P_2^T P_1^T A P_1 P_2 P_3 P_4 = ...$$

Algorithmus 28 (Householder–Reduktion auf Hessenberg–Form)

Input: $A \in \mathbb{R}^{n,n}$

Output: $H \in \mathbb{R}^{n,n}$, $H = Q^T A Q$, *wobei* Q *Produkt von Householder–Transformationen ist, gespeichert auf unterem Teil von* H.

FOR $k = 1 : n - 2$
- $v(k+1:n) =$ *HOUSE* $(A(k+1:n,k))$;
- $A(k+1:n,k:n) =$ *ROWHOUSE* $(A(k+1:n,k:n), v(k+1:n))$;
- $A(1:n,k+1:n) =$ *COLHOUSE* $(A(1:n,k+1:n), v(k+1:n))$;
- $A(k+2:n,k) = v(k+2:n)$;

END

Die Kosten betragen $20n^3/3$ flops sowie weitere $\frac{4n^3}{3}$ flops, falls Q gebraucht wird.

Die Fehleranalyse liefert $\tilde{H} = Q^T(A+E)Q$ mit $Q^TQ = I$ und $\|E\|_F \leq cn^2 \text{ eps } \|A\|_F$.

Die Hessenberg–Reduktion ist nicht eindeutig. Falls die Hessenberg–Matrix *unreduziert* ist, d.h., alle Elemente in der ersten unteren Nebendiagonalen $\neq 0$ sind, so ist die Hessenreduktion fast eindeutig durch die erste Spalte von Q bestimmt.

Theorem 132 (Implizites Q–Theorem) *Sei $A \in \mathbb{R}^{n,n}$, seien $Q = [q_1, \ldots, q_n]$ und $V = [v_1, \cdots, v_n]$ orthogonale Matrizen, so dass $Q^T AQ = H, V^T AV = G$ beides Hessenbergmatrizen. Sei k der kleinste Index, so dass $h_{k+1,k} = 0$ ($k = n$, falls H unreduziert). Falls $v_1 = q_1$, so gilt $v_i = \pm q_i$ und $|h_{i,i-1}| = |q_{i,i-1}|$ für $i = 2 : k$. Falls $k < n$, so gilt $g_{k+1,k} = 0$.*

Beweis. Sei $W = V^T Q$. Dann gilt $GW = WH$. Also folgt für $i = 2 : k$, dass

$$h_{i,i-1} w_i = G w_{i-1} - \sum_{j=1}^{i-1} h_{j,i-1} w_j.$$

Da $w_1 = e_1$ so folgt, dass $[w_1, \cdots, w_l]$ obere Dreiecks–Matrix ist. Aus $w_i = v_i^T q_i$ und $h_{i,i-1} = w_i^T G w_{i-1}$ folgt, dass $v_i = \pm q_i, |h_{i,i-1}| = |g_{i,i-1}|$ für $i = 2 : k$. Falls $h_{k-1,k} = 0$, so folgt (bis auf Vorzeichen)

$$\begin{aligned} g_{k+1,k} &= e_{k+1}^T G e_k = e_{k+1}^T G W e_k = (e_{k+1}^T W)(H e_k) \\ &= e_{k+1}^T \sum_{i=1}^{k} h_{ik} W e_i = \sum_{i=1}^{k} h_{ik} e_{k+1}^T e_i = 0. \end{aligned}$$

□

Damit kann man nun eine praktische Variante des QR–Algorithmus machen. In der Iteration bleibt für eine Hessenberg Matrix $H = Q_0^T A Q_0$ die Matrix weiter Hessenbergmatrix, denn $Q = HR^{-1}$ ist obere Hessenberg–Matrix, und daher auch $H = RQ$ ebenfalls.

Im folgenden kann man annnehmen, dass $h_{i+1,i} \neq 0$ für die Hessenberg–Matrix. Sonst kann man das Problem zerlegen. Falls z. B.

$$|h_{p+1,p}| \leq c \,\mathrm{eps}\,(|h_{pp}| + |h_{p+1,p+1}|), \tag{9.6}$$

so setzt man $h_{p+1,p}$ zu 0 und erhält

$$H = \left[\begin{array}{c|c} H_{11} & H_{12} \\ \hline 0 & H_{22} \end{array}\right] \begin{array}{c} p \\ n-p \end{array}$$

Dann spaltet man das eigenwertproblem für H in zwei Teilprobleme H_{11}, H_{22} auf.

Man kann die Konvergenz des Verfahrens durch Shifts erheblich beschleunigen.

Theorem 133 *Sei μ ein Eigenwert einer unreduzierten oberen Hessenberg-Matrix H. Sei $H - \mu I = QR$ eine QR–Zerlegung und $\tilde{H} = RQ + \mu I$ so gilt $\tilde{h}_{n,n-1} = 0$ und $\tilde{h}_{nn} = \mu$.*

Beweis. Wenn H unreduziert ist, so ist auch $H - \mu I$ unreduziert. Da $U^T(H - \mu I) = R$ singulär ist und da man aus der QR-Zerlegung erhält, dass $|r_{ii}| \geq |h_{i+1,i}|$ für $i = 1, \ldots, n-1$, so folgt $r_m = 0$ und damit verschwindet die letzte Zeile von $\tilde{H}$. ☐

Die Idee ist nun mit Näherungen an Eigenwerte zu shiften, z.B. mit einem Eigenwert von

$$\begin{bmatrix} h_{n-1,n-1} & h_{n-1,n} \\ h_{n,n-1} & h_{n,n} \end{bmatrix},$$

falls der reell ist oder mit zwei komplex konjugierten Shifts $\lambda, \bar{\lambda}$, um komplexe Arithmetik zu vermeiden. Dabei kann man dann zwei Schritte zusammen ausführen, denn aus

$$\begin{aligned} A_i - \lambda I &= Q_1 R_1, \\ A_{i+1} &= R_1 Q_1 + \lambda I, \\ A_{i+1} - \bar{\lambda} I &= Q_2 R_2, \\ A_{i+2} &= R_2 Q_2 + \bar{\lambda} I \end{aligned}$$

folgt $(A_i - \lambda I)(A_i - \bar{\lambda} I) = (Q_1 Q_2)(R_2 R_1)$. Die Matrix

$$(A_i - \lambda I)(A_i - \bar{\lambda} I) = A_i^2 + 2Re(\lambda) A_i + |\lambda|^2$$

ist reell auch wenn λ nicht reell ist. Man könnte nun $A_i^2 - 2Re(\lambda)A_i + |\lambda_1|^2$ berechnen und davon eine QR–Zerlegung $Q_i R_i$ bilden und dann $A_{i+2} = Q_i^T A_i Q_i$ setzen. Dies ist aber zu teuer, man nutzt besser das implizite Q–Theorem.

Dazu berechnet man die erste Spalte $z = (A_i - \lambda_i I)(A_i - \bar{\lambda}_i I)e_1$. Dann bestimmt man eine Householder–Matrix P_0, so dass $P_0 z = \alpha e_1$ und danach Householder–Matrizen $P_1, P_2, \cdots, P_{n-2}$ so dass $P_{n-2}^T \cdots P_0^T A_i P_0 \cdots P_{n-2}$ wieder obere Hessenbergmatrix ist. Ingesamt erhalten wir den folgenden Algorithmus für einen Schritt.

Algorithmus 29 (Francis QR–Schritt)

Input: *Unreduzierte obere Hessenbergmatrix $H \in \mathbb{R}^{n,n}$, dessen untere rechte 2×2 Matrix die Eigenwerte a_1, a_2 hat.*

Output: *$Z^T HZ$, wobei $Z = P_1 \cdots P_{n-2}$ Produkt von Householder–Matrizen ist und $Z^T(H - a_1 I)(H - a_2 I)$ obere Dreiecks–Matrix.*

$m = n - 1;$
$s = H(m,m) + H(n,n);$
$t = H(m,m) * H(n,n) - H(m,n) * H(n,m);$
$x = H(1,1) * H(1,1) + H(1,2) * H(2,1) - s * H(1,1) + t;$
$y = H(2,1) * (H(1,1) + H(2,2) - s);$
$z = H(2,1) * H(3,2);$
FOR $\quad k = 0 : n - 3$
$\quad v =$ *HOUSE* $((x, y, z)^T);$
$\quad H(k+1 : k+3, k+1 : n) =$ *ROWHOUSE* $(H(k+1 : k+3, k+1 : n), v);$
$\quad r = \min\{k+4, n\};$
$\quad H(1 : r, k+1 : k+3) =$ *COLHOUSE* $(H(1 : r, k+1 : k+3), v);$
$\quad x = H(k+2, k+1);$
$\quad y = H(k+3, k+1);$
$\quad$ *IF* $\quad k < n - 3$
$\qquad z = H(k+4, k+1);$
$\quad$ *END*
END
$v =$ *HOUSE* $([x, y]^T);$
$H(n-1 : n, n-2 : n) =$ *ROWHOUSE*$(H(n-1 : n, n-2 : n), v);$
$H(1 : n, n-1 : n) =$ *COLHOUSE*$(H(1 : n, n-1 : n), v);$

Die Kosten betragen $10n^2$ flops sowie $20n^2$ flops für die Akkumulation von Q. Insgesamt erhalten wir den folgenden Algorithmus.

Algorithmus 30 (QR–Algorithmus)

Input: $A \in \mathbb{R}^{n,n}$, *Toleranz* $tol >$ eps .

Output: *Reelle Schurform* $Q^T AT = T$. *A wird mit Hessenbergform von A überschrieben. Falls Q, T gebraucht werden, so wird T in A gespeichert. Falls nur Eigenwerte gebraucht werden, so werden die Diagonalblöcke in den entsprechenden Positionen gespeichert.*

Verwende Algorithmus 9.6 zur Transformation auf Hessenbergform.

UNTIL $q = n$

Setze alle Subdiagonalelemente zu Null, die $|h_{i,i-1}| \leq tol\ (|h_{ii}| + |h_{i-1,i-1}|)$ *erfüllen.*

Finde größtes $q \geq 0$ *und kleinstes* $p \geq 0$ *so dass*

$$H = \begin{bmatrix} H_{11} & H_{12} & H_{13} \\ 0 & H_{22} & H_{23} \\ 0 & 0 & H_{33} \end{bmatrix} \begin{array}{l} p \\ n-p-q \\ q \end{array},$$

wobei H_{33} *quasi obere Dreiecks–Matrix und* H_{22} *unreduziert.*

IF $q < n$

Mache einen Francis QR*–Schritt mit* $H_{22}, H_{22} = Z^T H_{22} Z$.

IF Q *benötigt*

$Q = Q \cdot \ diag(I_p, Z, I_q)$,

$H_{12} = H_{12} Z$,

$H_{23} = Z^T H_{23}$,

END

END

END

Bringe alle 2×2 *Blöcke mit reellen Eigenwerten auf obere Dreiecks–Form und akkumuliere gegebenenfalls die Transformationen.*

Die Kosten betragen ca. $10n^3$ flops für die Berechnung der Eigenwerte sowie zusätzlich ca. $25n^3$ flops für die Berechung Q und T. Diese Zahlen beruhen darauf, dass die Konvergenzrate für diesen Algorithmus ca 2.4 Iterationen pro Eigenwert beträgt.

Die Fehleranalyse liefert, dass der Algorithmus $\tilde{T} = Q^T(A+E)Q$ berechnet mit $Q^T Q = I$, $\|E\|_2 \approx$ eps $\|A\|_2$. Die Matrix $\tilde{Q}$ ist fast orthogonal, denn $\tilde{Q}^T \tilde{Q} = I + F \quad \|F\|_2 \approx$ eps.

Konvergenzbeweise existieren nur für Spezialfälle und spezielle Wahlen der Shifts.

Beispiel 134

$$A = \begin{bmatrix} 2 & 3 & 4 & 5 & 6 \\ 4 & 4 & 5 & 6 & 7 \\ 0 & 3 & 6 & 7 & 8 \\ 0 & 0 & 2 & 8 & 9 \\ 0 & 0 & 0 & 1 & 10 \end{bmatrix}$$

Iteration	$\mathcal{O}(\lvert h_{21}\rvert)$	$\mathcal{O}(\lvert h_{32}\rvert)$	$\mathcal{O}(\lvert h_{43}\rvert)$	$\mathcal{O}(\lvert h_{34}\rvert)$
1	10^0	10^0	10^0	10^0
2	10^0	10^0	10^0	10^0
3	10^0	10^0	10^{-1}	10^0
4	10^0	10^0	10^{-3}	10^{-3}
5	10^0	10^0	10^{-6}	10^{-5}
6	10^{-1}	10^0	10^{-13}	10^{-13}
7	10^{-1}	10^0	10^{-28}	10^{-13}
8	10^{-4}	10^0	*conv.*	*conv.*
9	10^{-8}	10^0		
10	10^{-8}	10^0		
11	10^{-16}	10^0		
12	10^{-32}	10^0		
13	*conv.*	*conv.*		

Was macht man nun, wenn man nur bestimmte Eigenvektoren zu bestimmten Eigenwerten will, nicht den invarianten Unterraum. Dann gibt es verschiedene Möglichkeiten; entweder man sortiert die Eigenwerte um, denn die erste Spate von Q ist der Eigenvektor zum ersten Diagonalelement, oder man verwendet die sogenannte *Inverse Iteration.* Sei dazu μ eine Eigenwertnäherung und sei $(A - \mu I)$ nicht exakt singulär, so ist der Eigenvektor zum größten Eigenwert von $(A - \mu I)^{-1}$ gerade der gesuchte.

Algorithmus 31 (Inverse Iteration)

Input: $A \in \mathbb{R}^{n,n}$, $\mu \in \mathbb{C}$ *Eigenwertnäherung und Startvektor* $q^{(0)}$.

Output: *Verbesserte Eigenwertnäherung und zugehöriger Eigenvektor.*

FOR $k = 1, 2, \cdots$ *UNTIL SATISFIED*

Löse $(A - \mu I) z^{(k)} = q^{(k-1)}$.

Setze $q^{(k)} = z^{(k)} / \|z^{(k)}\|_2$.

Bilde $\lambda^{(k)} = (q^{(k)})^T A q^{(k)}$.

END

Kann das überhaupt klappen, denn $A - \mu I$ ist doch fast singulär? Angenommen die Matrix A hat eine Basis aus Eigenvektoren $\{x_1, \cdots, x_n\}$ und $Ax_i = \lambda x_i,\ i = 1, \cdots, n$. Falls

$$q^{(0)} = \sum_{i=1}^{n} \beta_i x_i,$$

so ist $q^{(k)}$ ein Vektor in der Richtung

$$(A - \mu I)^{-k} q^{(0)} = \sum_{i=1}^{n} \frac{\beta_i}{(\lambda_i - \mu)^k} x_i. \tag{9.7}$$

Das Hauptgewicht liegt also in der Richtung von x_i, der zu $\lambda_i \approx \mu$ gehört. In diese Richtung ist das Gleichungssystem auch lösbar, da x_i im Bild von $A - \mu I$ liegt.

Wann beenden wir die Inverse Iteration? Sei $r^{(k)} = (A - \mu I) q^{(k)}$ das Residuum. Man bricht die Iteration ab wenn

$$\|r^{(k)}\|_\infty \leq c \cdot \text{ eps } \|A\|_\infty. \tag{9.8}$$

Falls man eine Transformation auf Hessenbergform schon hat, so braucht jeder Schritt nur $\mathcal{O}(n^2)$ flops.

Für symmetrische Matrizen kann man den QR Algorithmus genau so anwenden. Man braucht nur weniger Speicher und flops, indem man eine symmetrische Speicherung verwendet. Ausserdem erhält man in diesem Fall kubische Konvergenz. Da alle Eigenwerte reell sind, sind auch Doppelshifts nicht notwendig.

9.2 Die Singulärwertzerlegung

Eines der wichtigsten Hilfsmittel in allen Bereichen der numerischen Mathematik ist heutzutage die Singulärwertzerlegung einer Matrix. Diese wird zur Rangbestimmung, Datenkompression, Bildverarbeitung und vielen anderen Anwendungen verwendet.

Theorem 135 *Sei A eine reelle $m \times n$ Matrix. Dann gibt es orthogonale Matrizen $U = [u_1, \cdots, u_m] \in \mathbb{R}^{m,m}$ und $V = [v_1, \cdots, v_n] \in \mathbb{R}^{n,n}$, so dass*

$$U^T A V = \textit{ diag } (\sigma_1, \cdots, \sigma_p) \in \mathbb{R}^{m,n},\ p = \min(m, n) \tag{9.9}$$

wobei $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_p \geq 0$.

Beweis. Seien $x \in \mathbb{R}^n, y \in \mathbb{R}^m$ Vektoren mit $\|x\|_2 = \|y\|_2 = 1$, so dass $Ax = \sigma y, \sigma = \|A\|_2$. Dies ist möglich, da $\|A\|_2 = \sup_{\|x\|_2=1} \|Ax\|_2$ angenommen wird.
Da jede Menge von orthogonalen Vektoren zu einer Basis des ganzen Raumes ergänzt werden kann, gibt es $V_1 \in \mathbb{R}^{n,(n-1)}$ und $U_1 \in \mathbb{R}^{m,(m-1)}$, so dass $V = [x, V_1] \in \mathbb{R}^{n,n}, U = [y, U_1] \in \mathbb{R}^{m,m}$ orthogonal sind. Dann gilt

$$\begin{aligned} U^T AV &= \begin{bmatrix} y^T \\ U_1^T \end{bmatrix} A \begin{bmatrix} x & V_1 \end{bmatrix} = \begin{bmatrix} y^T \\ U_1^T \end{bmatrix} \begin{bmatrix} \sigma y, & AV_1 \end{bmatrix} \\ &= \begin{bmatrix} \sigma & w^T \\ 0 & B \end{bmatrix} =: A_1. \end{aligned}$$

Da

$$\left\| A_1 \begin{bmatrix} \sigma \\ w \end{bmatrix} \right\|_2^2 \geq (\sigma^2 + w^T w)^2,$$

so folgt, dass

$$\|A_1\|_2^2 \geq \sigma^2 + w^T w.$$

Da $\sigma^2 = \|A\|_2^2 = \|A_1\|_2^2$ folgt $w = 0$. Per Induktion folgt das Resultat. □

Die Werte σ_i heißen *Singulärwerte*, die u_i, v_i *linke und rechte Singulärvektoren.* Es gilt

$$Av_i = \sigma_i u_i,\ A^T u_i = \sigma_i v_i,\ i = 1, \cdots, \min(m, n). \tag{9.10}$$

Die Singulärwerte sind die Längen der Halbachsen der Hyperellipsoide $E = \{Ax : \|x\| = 1\}$.

Sei $\sigma_1 \geq \cdots \geq \sigma_r > \sigma_{r+1} = \cdots = \sigma_p = 0$, so gilt

$$\begin{aligned} \text{rang } (A) &= r; \\ \text{Kern } (A) &= \text{ span } \{v_{r+1}, \cdots, v_n\}; \\ \text{Bild } (A) &= \text{ span } \{u_1, \cdots, u_r\}; \\ \|A\|_F^2 &= \sigma_1^2 + \cdots + \sigma_p^2 \quad p = \min\{m, n\} \\ \|A\|_2 &= \sigma_1; \\ \text{cond }_2(A) &= \frac{\sigma_1}{\sigma_p}. \end{aligned}$$

Theorem 136 *Sei* $A = U\Sigma V^T$ *die Singulärwertzerlegung von* $A \in \mathbb{R}^{m,n}$. *Sei* $k < r = \mathrm{rang}(A)$ *und*

$$A_k = \sum_{i=1}^{k} \sigma_i u_i v_i^T. \tag{9.11}$$

Dann gilt

$$\min_{\mathrm{rang}(B)=k} \|A - B\|_2 = \|A - A_k\|_2 = \sigma_{k+1}. \tag{9.12}$$

Beweis. Aus $U^T A_k V = \mathrm{diag}\ (\sigma_1, \cdots, \sigma_k, 0 \cdots, 0)$ folgt $\mathrm{rang}(A_k) = k$ und da $U^T(A - A_k)V = \mathrm{diag}(0, \cdots, 0, \sigma_{k+1}, \cdots, \sigma_p)$, so folgt $\|A - A_k\|_2 = \sigma_{k+1}$. Angenommen $\mathrm{rang}(B) = k$ für $B \in \mathbb{R}^{m,n}$. Dann gibt es orthonormale Vektoren $x_1, \cdots, x_{n-k}$, so dass $\mathrm{kern}(B) = \mathrm{span}\{x_1, \cdots, x_{n-k}\}$. Aus Dimensionsgründen gilt

$$W = \mathrm{span}\{x_1, \cdots, x_{n-k}\} \cap \mathrm{span}\{v_1, \cdots, v_{k+1}\} \neq \{0\}.$$

Sei $z \in W, \|z\|_2 = 1$. Mit $z = \sum \alpha_i v_i$ folgt $\alpha_i = v_i^T z$. Da $Bz = 0$ und $Az = \sum_{i=1}^{k+1} \sigma_i (v_i^T z) u_i$, so folgt

$$\|A - B\|_2^2 \geq \|(A - B)z\|_2^2 = \|Az\|_2^2 = \sum_{i=1}^{k+1} \sigma_i^2 (v_i^T z) \geq \sigma_{k+1}^2.$$

□

9.2.1 Die Berechnung der Singulärwertzerlegung

Man könnte die Berechnung der Singulärwertzerlegung direkt auf ein symmetrisches Eigenwertproblem zurückführen, denn falls $A = U\Sigma V^T \in \mathbb{R}^{m,n}$, $m \geq n$, so gilt

$$AA^T = U\Sigma V^T V \Sigma^T U^T = U\Sigma^2 U^T = U\,\mathrm{diag}(\sigma_1^2, \cdots, \sigma_n^2) U^T$$

und da AA^T symmetrisch ist, ist dies eine Schurform mit spezieller Anordnung der Diagonalelemente. Analog gilt

$$A^T A = V\Sigma^2 V^T = V\,\mathrm{diag}(\sigma_1^2, \cdots, \sigma_n^2, 0, \cdots, 0) V^T.$$

Man könnte also $A^T A$ und AA^T bilden und den symmetrischen QR-Algorithmus verwenden, aber wie wir schon beim Ausgleichsproblem gesehen haben, würde dieses zur Quadrierung der Konditionszahl führen.

Besser ist daher der Algorithmus von Golub/Kahan, der auf dem symmetrischen QR–Algorithmus aufbaut.

Algorithmus 32 (Householder–Bidiagonalisierung)

Input: $A \in \mathbb{R}^{m,n}, m \geq n$

Output: *Der obere Bidiagonalteil von A wird dem oberen Bidiagonalteil von $B = U^T AV$ überschrieben, B obere Bidiagonalmatrix. $U = U_1 \cdots U_{n-1}(\cdot U_n)$ und $V = V_1 \cdots V_{n-2}$ sind Produkte von Householder–Matrizen. Der wesentliche Teil von U_j ist in $A(j+1:m, j)$ und der wesentliche Teil von V_j in $A(j, j+2:n)$ gespeichert.*

```
FOR     j = 1 : n − 1
        v(j : m) = HOUSE(A(j : m, j));
        A(j : m, j : n) = ROWHOUSE(A(j : m, j : n), v(j : m));
        A(j + 1 : m, j) = v(j + 1 : m);
        IF     j ≤ n − 2
               v(j + 1 : n) = HOUSE(A(j, j + 1 : n)^T);
               A(j : m, j + 1 : n) = COLHOUSE(A(j : m, j + 1 : n), v(j + 1 : n));
               A(j, j + 2 : n) = v(j + 2 : n)^T;
        END
END
IF      m > n
        v(n : m) = HOUSE(A(n : m, n));
        A(n : m, n) = ROWHOUSE(A(n : m, n), v(n : m));
        A(n + 1 : m, n) = v(n + 1 : m);
END
```

Die Kosten betragen $4mn^2 - 4n^3/3$ flops. Falls U, V explizit benötigt werden, erhält man diese mit weiteren $4m^2n - 4n^3/3$ flops für U und weiteren $4n^3/3$ für V.

Die Fehleranalyse liefert, dass $B = U^T(A + E)V$ mit $U^T U = I$, $V^T V = I$ und $\|E\|_F \leq cn^2 \,\text{eps}\, \|A\|_F$.

Das Vorgehen ist dann wie folgt:

$$\begin{bmatrix} x & x & x & x \\ x & x & x & x \\ x & x & x & x \\ x & x & x & x \\ x & x & x & x \end{bmatrix} \xrightarrow{U_1} \begin{bmatrix} x & x & x & x \\ 0 & x & x & x \\ 0 & x & x & x \\ 0 & x & x & x \\ 0 & x & x & x \end{bmatrix} \xrightarrow{V_1}$$

$$\begin{bmatrix} x & x & 0 & 0 \\ 0 & x & x & x \\ 0 & x & x & x \\ 0 & x & x & x \\ 0 & x & x & x \end{bmatrix} \xrightarrow{U_2} \begin{bmatrix} x & x & 0 & 0 \\ 0 & x & x & x \\ 0 & 0 & x & x \\ 0 & 0 & x & x \\ 0 & 0 & x & x \end{bmatrix} \xrightarrow{V_2} \begin{bmatrix} x & x & 0 & 0 \\ 0 & x & x & 0 \\ 0 & 0 & x & x \\ 0 & 0 & x & x \\ 0 & 0 & x & x \end{bmatrix}$$

$$\xrightarrow{U_3} \begin{bmatrix} x & x & 0 & 0 \\ 0 & x & x & 0 \\ 0 & 0 & x & x \\ 0 & 0 & 0 & x \\ 0 & 0 & 0 & x \end{bmatrix} \xrightarrow{U_4} \begin{bmatrix} x & x & 0 & 0 \\ 0 & x & x & 0 \\ 0 & 0 & x & x \\ 0 & 0 & 0 & x \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

Anschliessend wendet man den symmetrischen QR–Algorithmus implizit auf $T = B^T B$ (tridiagonal) an, Mit

$$B = \left[\begin{array}{cccc} d_1 & f_1 & & \\ & \ddots & \ddots & \\ & & \ddots & f_{n-1} \\ & & & d_n \\ \hline & & 0 & \end{array}\right],$$

gilt

$$B^T B = \begin{bmatrix} d_1^2 & f_1 d_1 & & \\ f_1 d_1 & d_2^2 + f_1^2 & f_2 d_2 & \\ \ddots & \ddots & \ddots & \\ & & & f_{n-1} d_{n-1} \\ & & f_{n-1} d_{n-1} & d_n^2 + f_{n-1}^2 \end{bmatrix}.$$

Aber wir bilden das Produkt nicht, sondern arbeiten mit der Faktorisierung.

Zur Zerlegung verwendet man nun nicht Householder Spiegelungen, sondern *Givens-Rotationen.*

Bestimme $c, s \in \mathbb{R}$ mit $c^2 + s^2 = 1$, so dass

$$\begin{bmatrix} c & -s \\ s & c \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} * \\ 0 \end{bmatrix}.$$

Um die Stabilität der Berechnung von c, s zu erhöhen berechnet man nicht $c = \pm b/\sqrt{a^2+b^2}, s = \mp a/\sqrt{a^2+b^2}$, sondern erst den Quotienten $t = a/b$ bzw. $t = b/a$ und dann macht man eine Fallunterscheidung. Aus Stabilitätsgründen sollte $|t| \leqslant 1$ gelten.

Algorithmus 33 (Erzeugung der Givens–Parameter)

Input: $a, b \in \mathbb{R}$.

Output: $c, s \in \mathbb{R}$ *mit* $c^2 + s^2 = 1$, *so dass* $\begin{bmatrix} c & -s \\ s & c \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} * \\ 0 \end{bmatrix}$.

```
FUNCTION      [c, s] = GIVENS(a, b)
              IF      |b| > |a|
                      t = -a/b;
                      s = 1/sqrt(1 + t^2);
                      c = s * t;
              ELSE
                      t = -b/a;
                      c = 1/sqrt(1 + t^2);
                      s = c * t;
              END
END GIVENS
```

Die Kosten sind 5 flops und 1 Wurzel.

Algorithmus 34 (Vormultiplikation mit Givens–Rotation)

Input: $A \in \mathbb{K}^{2,n}, c, s \in \mathbb{R}, c^2 + s^2 = 1$.

Output: A *überschrieben mit* $\begin{bmatrix} c & -s \\ s & c \end{bmatrix} A$.

```
FUNCTION      A = ROWROT(A, c, s);
              A = [c, -s; s, c] * A;
END ROWROT
```

Die Kosten betragen $6n$ flops und für den Fehler gilt $gl(\tilde{G}A) = G(A+E)$ mit $\|E\|_2 = \mathcal{O}(\text{eps}\,\|A\|_2)$.

Algorithmus 35 (Nachmultiplikation mit Givens–Rotation)

Input: $A \in \mathbb{K}^{m,2}, c, s \in \mathbb{R}, c^2 + s^2 = 1.$

Output: *A überschrieben mit* $A \begin{bmatrix} c & s \\ -s & c \end{bmatrix}.$

```
FUNCTION      A = COLROT(A, c, s);
              A = A * [c, s; -s, c];
END COLROT
```

Die Kosten betragen $6n$ flops und für den Fehler gilt $gl(A\tilde{G}) = (A + E)G$, mit $\|E\|_2 = \mathcal{O}(\text{eps}\,\|A\|_2)$.

Für die Iteration in der Singulärwertzerlegung gehen wir implizit wie im QR-Algorithmus vor. Wir berechnen den Eigenwert λ von

$$T(n-1:n, n-1:n) = \begin{bmatrix} d_{n-1}^2 + f_{n-2}^2 & d_{n-1}f_{n-1} \\ d_{n-1}f_{n-1} & d_n^2 + f_{n-1}^2 \end{bmatrix}$$

der näher an $d_n^2 + f_{n-1}^2$ liegt. Dann berechnen wir die beiden Elemente von

$$(T - \lambda I)e_1 = \begin{bmatrix} d_1^2 - \lambda \\ d_1 f_1 \end{bmatrix}$$

und eine Givensrotation G, so dass

$$G \begin{bmatrix} d_1^2 - \lambda \\ d_1 f_1 \end{bmatrix} = \begin{bmatrix} * \\ 0 \end{bmatrix}.$$

Transformiere dann

$$BG_1 = \begin{bmatrix} * & * & & & \\ \circledast & * & * & & \\ & & \ddots & \ddots & \\ & & & \ddots & * \\ & & & & * \end{bmatrix}$$

wieder auf Bidiagonalform.

Algorithmus 36 (Golub/Kahan SWZ–Schritt)

Input: *Bidiagonalmatrix* $B \in \mathbb{R}^{m,n}$*, die keine Nullen auf Diagonale und Superdiagonale hat.*

Output: B *überschrieben mit der Bidiagonalmatrix* $\bar{B} = \bar{U}^T B \bar{V}$*, wobei* $\bar{U}$ *und* $\bar{V}$ *orthogonal und* $\bar{V}$ *ist im wesentlichen die orthogonale Matrix die man erhalten würde, wenn man die symmetrische Version des* QR*–Algorithmus auf* $T = B^T B$ *anwenden würde.*

Sei μ *der Eigenwert der rechten unteren* 2×2 *Matrix von* $B^T B$ *der näher am letzten Diagonalelement von* $B^T B$ *ist.*

$y = B(1,1)^2 - \mu$

$z = B(1,1) * B(1,2)$

FOR $\quad k = 1 : n-1$

$\qquad [c,s] = GIVENS(y,z);$

$\qquad l = \max\{1, k-1\}$

$\qquad B(l : k+1, k : k+1) = COLROT(B(l : k+1, k : k+1), c, s);$

$\qquad y = B(k,k);$

$\qquad z = B(k+1,k);$

$\qquad [c,s] = GIVENS(y,z);$

$\qquad l = \min\{k+2, n\}$

$\qquad B(k : k+1, k : l) = ROWROT(B(k : k+1, k : l), c, s);$

$\qquad$ *IF* $\quad k < n-1$

$\qquad\qquad y = B(k, k+1);$

$\qquad\qquad z = B(k, k+2);$

$\qquad$ *END*

END

Die Kosten sind $30n$ flops und $2n$ Quadratwurzeln. Zusätzlich benötigt man für die Akkumulation von U $6mn$ flops und für V $6n^2$ flops
Die Fehleranalyse ist analog zur QR-Zerlegung.

Algorithmus 37 (SWZ Berechnung–Golub/Reinsch)

Input: $A \in \mathbb{R}^{m,n}$, $m \geq n, \varepsilon = c \cdot \mathrm{eps}$, *c klein.*

Output: *A überschrieben mit* $U^T AV = D + E$ U, V *orthogonal,* D *diagonal,* $\|E\|_2 \approx \mathrm{eps}\, \|A\|_2$.

Bidiagonalisiere A mit Algorithmus 32. $q = 0$;

WHILE $q < n$

Setze $A(i, i+1)$ *zu Null, falls*

$|A(i,i+1)| \leq \varepsilon(|A(i,i)| + |A(i+1,i+1)|),\ i = 1 : n-1.$

Bestimme größtes q und kleinstes p, so dass in

$$B = \begin{array}{cl} \left[\begin{array}{ccc} B_{11} & 0 & 0 \\ 0 & B_{22} & 0 \\ 0 & 0 & B_{33} \end{array}\right] & \begin{array}{l} p \\ n-p-q \\ q+m-n \end{array} \\ \begin{array}{ccc} p & n-p-q & q \end{array} & \end{array}$$

B_{33} *diagonal ist, und* B_{22} *nur Nebendiagonalelemente* $\neq 0$ *hat.*

IF $q < n$

Setze in B_{22} *alle* $A(i,i)$ *zu Null, falls* $|A(i,i)| \leq \varepsilon \|B_{22}\|$.

IF *Diagonalelement von* B_{22} *Null, so mache Spezialschritt (unten)*

END

Wende Algorithmus 36 auf B_{22} *an.*

END

END

Spezialschritt: Sei k der größte Index, so dass für die Einträge in B_{22}, $A(k,k) = 0$ *ist und* $A(l,l) \neq 0, l > k$.

```
FOR  j = k + 1 : n − p − q
     [c, s] = GIVENS(A(j, j), A(k, j));
     A([j, k], j) = ROWROT(A([j, k], j), c, s);
     IF    j < n − p − q
           A([j, k], j + 1) = ROWROT(A([j, k], j + 1), c, s);
     END
END
FOR  j = k − 1 : −1 : p + 1
     [c, s] = GIVENS(A(j, j), A(j, k));
     A(j, [j, k]) = COLROT(A(j, [j, k]), c, s);
     IF    j > p + 1
           A(j − 1, [j, k]) = COLROT(A(j − 1, [j, k]), c, s);
     END
END
```

Die Kosten sind in der folgenden Tabelle zusammengefasst:

Berechnung von	flops
Σ	$4mn^2 - 4n^3/3$
Σ, V	$4mn^2 + 8n^3$
Σ, U	$4m^2n - 8mn^2$
Σ, U, V	$4m^2n + 8mn^2 + 9n^3$

Den Spezialschritt verdeutlichen wir an folgendem Beispiel. Zuerst schieben wir den Eintrag $(k, k+1)$ nach rechts heraus

$$\left[\begin{array}{cc|c|cc} \ast & \ast & & & \\ & \ast & \circledast & & \\ \hline & & 0 & \circledast & \\ \hline & & & \ast & \ast \\ & & & & \ast \end{array}\right] \mapsto \left[\begin{array}{cc|c|cc} \ast & \ast & & & \\ & \ast & \circledast & & \\ \hline & & 0 & 0 & \circledast \\ \hline & & & \circledast & \circledast \\ & & & & \ast \end{array}\right] \mapsto \left[\begin{array}{cc|c|cc} \ast & \ast & & & \\ & \ast & \ast & & \\ \hline & & 0 & 0 & 0 \\ \hline & & & \circledast & \circledast \\ & & & & \circledast \end{array}\right].$$

Dann schieben wir den Eintrag $(k-1, k)$ nach oben heraus

$$\left[\begin{array}{cc|c|cc} \ast & \ast & & & \\ & \ast & \circledast & & \\ \hline & & 0 & 0 & 0 \\ \hline & & & \circledast & \circledast \\ & & & & \circledast \end{array}\right] \mapsto \left[\begin{array}{cc|c|cc} \ast & \circledast & \circledast & & \\ & \circledast & 0 & & \\ \hline & & 0 & 0 & 0 \\ \hline & & & \circledast & \circledast \\ & & & & \circledast \end{array}\right] \mapsto \left[\begin{array}{cc|c|cc} \circledast & \circledast & 0 & & \\ & \circledast & 0 & & \\ \hline & & 0 & 0 & 0 \\ \hline & & & \circledast & \circledast \\ & & & & \circledast \end{array}\right].$$

Beispiel 137

Sei $A = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 2 & 1 & 0 \\ 0 & 0 & 3 & 1 \\ 0 & 0 & 0 & 4 \end{bmatrix}$ *dann ergibt sich:*

Iteration	$\mathcal{O}(a_{21})$	$\mathcal{O}(a_{23})$	$\mathcal{O}(a_{34})$
1	10^0	10^0	10^0
2	10^0	10^0	10^0
3	10^0	10^0	10^0
4	10^0	10^{-1}	10^{-2}
5	10^0	10^{-3}	10^{-8}
6	10^0	10^{-1}	10^{-27}
7	10^0	10^{-1}	–
8	10^0	10^{-4}	–
9	10^{-1}	10^{-14}	–
10	10^{-1}	–	–
11	10^{-4}	–	–
12	10^{-12}	–	–
13	–	–	–

Man sieht hier gut die kubische Konvergenz.

9.3 Bisektionsverfahren für symmetrische tridiagonale Matrizen

Eine andere Methode zur Berechnung der Eigenwerte symmetrischer Tridiagonalmatrizen T ergibt sich mit Hilfe der Bisektion. Sei T_r die führende Hauptabschnittsmatrix von

$$T = \begin{bmatrix} a_1 & b_1 & & & \\ \bar{b}_1 & a_2 & b_2 & & \\ & \bar{b}_2 & \ddots & \ddots & \\ & & \ddots & \ddots & b_{n-1} \\ & & & \bar{b}_{n-1} & a_n \end{bmatrix} \tag{9.13}$$

und sei $p_r(x) = \det(T_r - xI)$ für $r = 1 : n$. Dann gilt

$$p_r(x) = (a_r - x)p_{r-1}(x) - |b_{r-1}|^2 p_{r-2}(x) \tag{9.14}$$

Man kann $p_r(x)$ in $\mathcal{O}(r)$ flops auswerten.

Algorithmus 38 *Bisektion zur Berechnung der Eigenwerte von T.*
Input: *T wie in (9.13) symmetrisch, tridiagonal, Toleranzgrenze tol, $y, z \in \mathbb{R}$, $y < z$, $p_n(y)p_n(z) < 0$.*
Output: *Eigenwert x von T.*

WHILE $|y - z| > tol\,(|y| + |z|)$
  $x = \frac{y+z}{2}$
  IF $p_n(x)p_n(y) < 0$,
    $z = x$
  ELSE
    $y = x$.
  END IF
END

Die Kosten betragen $\mathcal{O}(h)$ flops pro Schritt und der Fehler halbiert sich pro Schritt, d.h., wir haben lineare Konvergenz.