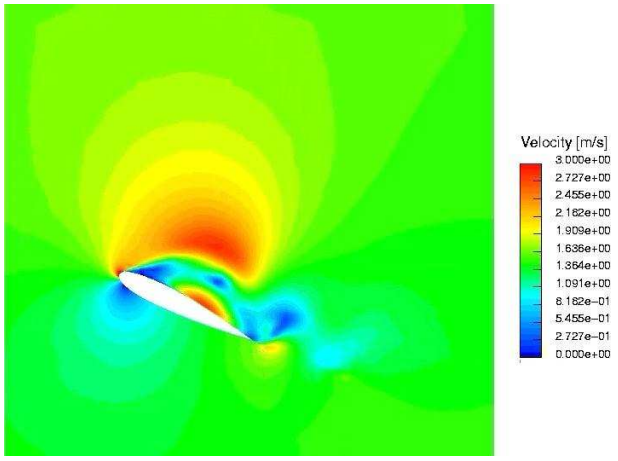
Stefan A. Funken, Dirk Lebiedz, Karsten Urban

Angewandte Numerik 2



SKRIPT, UNIVERSITÄT ULM, WINTERSEMESTER 2014/15

Vorwort. Dieses Manuskript ist entstanden aus Mitschriften und Skripten verschiedener Vorlesungen, die wir seit 2002 an der Universität Ulm gehalten haben. Es ist der Sinn des vorliegenden Dokumentes, den Studierenden unserer Vorlesungen einen einheitlichen Stoffumfang für die Vorlesung **Angewandte Numerik II** zu geben, unabhänging davon, wer von uns tatsächlich die Vorlesung hält. In diesem Sinne bietet das vorliegende Manuskript einen Rahmen für die Nachbearbeitung der Vorlesungen und der Vorbereitung auf Prüfungen. Dieses Manuskript kann keinesfalls das Studium von Lehrbüchern ersetzen. Eine entsprechende Liste von Lehrbüchern findet sich im Literaturverzeichnis und auf der Internet–Seite der Vorlesung.
Jedes Manuskript weist Fehler auf, sicher auch dieses. Wenn Sie Fehler, Druckfehler, sprachliche Unzulänglichkeiten oder inhaltliche Flüchtigkeiten finden, würden wir uns über einen entsprechenden Hinweis per Email freuen. Sie helfen damit zukünftigen Studierenden. Vielen Dank im Voraus.

Danksagung. Einige Vorgängerversionen dieses Manuskriptes wurden aus Mitteln der Studiengebühren finanziert. Eine Reihe von Personen haben bei der Erstellung geholfen. Wir danken Theresa und Julia Springer, Markus Bantle und Judith Rommel für zahlreiche Hinweise. Frau Kristin Kirchner und Herrn Moritz Reinhard sind wir für das sorgfältige Lesen des Manuskripts zu besonderem Dank verpflichtet. Ihre zahlreichen Kommentare, Vorschläge, Korrekturen und Hinweise haben die Qualität des Textes wesentlich verbessert. Weiterhin möchten wir Katharina Becker-Steinberger, Sebastian Kestler, Dr. Michael Lehn und Moritz Reinhard für die Ausarbeitung der Aufgaben und zugehöriger Lösungen danken.

Ganz besonderer Dank gebührt auch Frau Petra Hildebrand, die unsere handschriftlichen Aufzeichnungen in LaTeX umgesetzt und zahlreiche Grafiken erstellt hat. Frau Brandner, Frau Serbine und Herrn Weithmann möchten wir für das LaTeX'en der Lösungen danken.

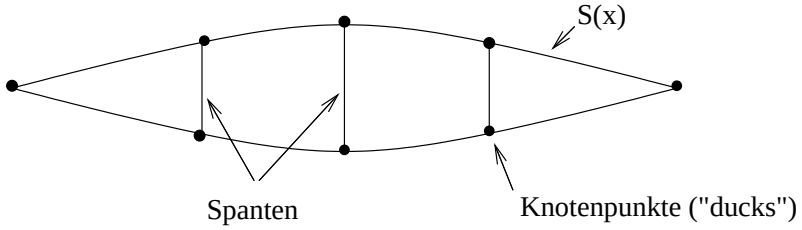
Stand. Ulm, Oktober 2014, Stefan A. Funken, Dirk Lebiedz, Karsten Urban

Inhaltsverzeichnis

1 SPLINES

In diesem Kapitel wird die Anwednung und Grundzüge der Theorie polynomialer und rationaler Splinefunktionen dargestellt. Zuerst stellt sich jedoch die Frage, warum überhaupt die Spline-Approximation eingeführt wurde. Ein kleiner Ausflug in die Geschichte fördert Interessantes zu Tage.

Bemerkung 1.0.1 *Das Wort „Spline" bedeutet etwa „dünne Holzlatte". Im Schiffsbau etwa seit dem 18. Jahrhundert wurden solche Holzlatten um die Spanten gelegt, um so einen Schiffsrumpf aus Holz zu formen.*



Die Latte biegt sich so, dass die innere Energie (Formarbeit) minimal wird, d.h.

$$E(S) := \int_a^b \frac{S''(x)}{(1+S'(x)^2)^{5/2}} dx \longrightarrow \underset{S}{Min},$$

wobei das zu minimierende Integral die mittlere quadratische Krümmung ist. Es ist also diejeige Kurve S gesucht, für die $E(S)$ minimal wird. Bei schlanken Booten ist die Biegung S' beschränkt, minimiere also

$$\tilde{E}(S) := \int_a^b S''(x) dx \longrightarrow \underset{S}{Min} \tag{1.1}$$

unter den Nebenbdingungen $S(x_i) = f_i$ für $i = 1, \ldots, n$, wobei (x_i, f_i) die gegebenen Daten sind.

Im Gegensatz zur Polynominterpolation verhindert das „Glattheitsmaß" der mininalen Energie das Oszillieren wie etwa beim Gegenbeispiel von Runge. Euler und Bernoulli haben schon erkannt, dass die Lösung S des Minimierungsproblems folgende Eigenschaften besitzt: (1) $S_{|[x_i,x_{i+1})} \in \mathcal{P}_3$ (lokal polynomial), (2) $S \in C^2([a,b])$ (global glatt). Dies wird uns zur Definition von Splines führen. Präzise formulieren kann man die Approximationseigenschaften von Spline wie folgt:

Satz 1.0.2 (von Jackson) *Es gilt die Abschätzung*

$$\forall k \in \mathbb{N}_0 \, \exists c_k > 0 \, \forall (-\infty < a < b < \infty) \, \forall f \in C^k([a,b], \mathbb{R}) \, \forall n < k$$

$$E_n(f) \leq c_k \left(\frac{b-a}{n}\right)^k \omega\left(f^{(k)}, \frac{b-a}{2(n-k)}\right)$$

für die Bestapproxomation E_n von f. (Vgl. [Rivlin], Seite 23, in anderer Form [Schönhage], Seite 182.) Dabei ist

$$E_n(f) := dist(f, \mathbb{P}_n) = \inf_{g \in \mathbb{P}_n} \|f - g\|_{\infty,[a,b]}$$

*der **Fehler der besten Approximation** und*

$$\omega(g,h) := \sup_{|\delta|<h} \|g(\cdot) - g(\cdot + \delta)\|_{\infty,[a,b]}$$

*der so genannte **Stetigkeitsmodul** von g, ein Maß für die Glattheit von g.*

Der Satz von Jackson besagt folgendes: Da der Stetigkeitsmodul einer wenig glatten Funktion groß ist, wird man für solche Funktionen keine „gute“ Näherung erwarten dürfen, da schon die Schranke für die Bestapproximation entsprechend groß wird. Ferner sind Interpolationspolynome

- abhängig von der Wahl der Knoten, wie schon das Beispiel von Runge zeigt,
- bei Änderung eines Koeffizienten eines Polynoms, tritt eine **globale** Änderung ein, und
- viele Basen sind schlecht konditioniert.

Die einfachste Form einer stetigen Funktion f, die die Bedingung $f(x_i) = y_i$ für gegebene geordnete Paare (x_i, y_i) $(i = 0, \dots, n)$ erfüllt, ist sicherlich der Streckenzug, d.h.

$$f|_{(x_{i-1},x_i)} = \frac{y_i - y_{i-1}}{x_i - x_{i-1}}(\cdot - x_{i-1}) + y_{i-1} \quad (i = 1, \dots, n)\,.$$

Auf jedem Intervall ist f also ein Polynom höchstens ersten Grades (ein Geradenstück) und insgesamt eine stetige Funktion. Allgemeiner lautet dann die Definition wie folgt:

Definition 1.0.3 (Splineraum $\mathcal{S}^k(\mathcal{T})$) *Es sei $\mathcal{T} = \{t_0, \cdots, t_{n+1}\}$ eine Knotenfolge von $n+2$ paarweise verschiedenen Knoten*

$$a = t_0 < \ldots < t_{n+1} = b\,.$$

*Ein **Spline** vom Grad k ($k \geq 0$) bezüglich der Knoten $\mathcal{T}$ ist eine Funktion $s \in C^{k-1}[a,b]$, für die auf jedem Intervall $[t_j, t_{j+1}]$, $j = 0, \dots, n$*

$$s\big|_{[t_k,t_{k+1}]} \in \mathbb{P}_k$$

gilt. Den Raum aller Splines vom Grad k zur Knotenfolge $\mathcal{T}$ bezeichnen wir mit $\mathcal{S}^k(\mathcal{T})$. Unter $C^{-1}[a,b]$ ist der Raum der stückweise stetigen Funktionen zu verstehen, d.h. unstetig nur an den Knoten x_j ($j = 0, \dots, n+1$).

Wir lernen nun wichtige Spezialfälle kennen.

1.1 Kubische Spline-Interpolation

Im Folgenden betrachten wir die Interpolation mit **kubischen Splines** (also $k = 3$). Betrachten Sie folgende Bilder:

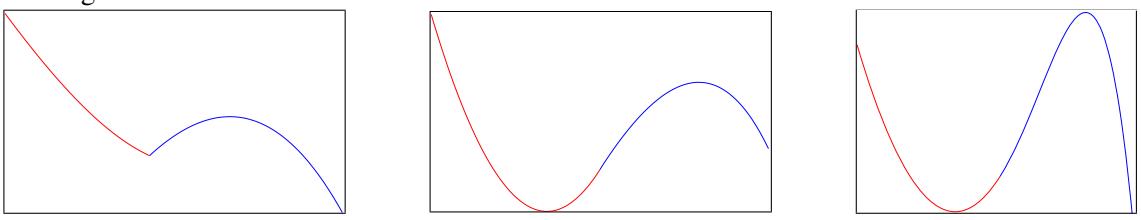


Abb. 1.1: Verschiedene Funktionen aus C^0, C^1 und C^2

Welche dieser Funktionen in der Sequenz von Grafiken empfinden Sie als glatt?
Der Knick im linken Graphen ist offensichtlich, auch im mittleren Graphen erkennt man mit etwas Geduld und Erfahrung eine Unstetigkeit in der Krümmung der Funktion, welche jedoch in der rechten Grafik nicht mehr auszumachen ist.
In vielen grafischen Anwendungen genügen diese Glattheitsanforderungen an die Interpolationsfunktion, nämlich sie vom Auge als „glatt“ zu empfinden.

Dies ist einer der Hauptgründe, weswegen wir uns zuerst auf Funktionen aus $\mathcal{S}^k(\mathcal{T}) \subset C^2$ beschränken.
Untersuchen wir zuerst die Eigenschaften der kubischen Splines, bevor wir zu ihrer Konstruktion und Berechnung kommen.

Satz 1.1.1 *Sei s ein interpolierender kubischer Spline zu der Funktion f an den Knoten $a = t_0 < \cdots < t_{n+1} = b$ und y eine beliebige interpolierende Funktion von f, sodass*

$$s''(t) \cdot \big(y'(t) - s'(t)\big) = 0 \quad (t \in [a,b]). \tag{1.2}$$

Dann gilt

$$\|s''\|_2 := \Big(\int_a^b \big(s''(t)\big)^2 \, dt \Big)^{1/2} \leq \|y''\|_2. \tag{1.3}$$

Beweis. Aus (1.3) folgt mit $y'' = s'' + (y'' - s'')$

$$\begin{aligned} \int_a^b (y''(x))^2 \, dx &= \int_a^b (s''(x))^2 + 2s''(x)(y''(x) - s''(x)) + (y''(x) - s''(x))^2 dx \\ &= \int_a^b (s''(x))^2 + (y''(x) - s''(x))^2 dx \geq \int_a^b (s''(x))^2 dx, \end{aligned}$$

falls $\int_a^b s''(y'' - s'')dx$ verschwindet. Dies läßt sich aber mit (1.2) und partieller Integration unter Berücksichtigung von $s(x)\big|_{[x_{i-1},x_i]} \in \mathbb{P}_3$ (und somit $s'''(x)\big|_{[x_{i-1},x_i]} \equiv c_i \in \mathbb{R}$) wiefolgt zeigen:

$$\begin{aligned} & \int_a^b s''(x)(y''(x) - s''(x))dx = \sum_{i=1}^n \int_{x_{i-1}}^{x_i} s''(x)(y''(x) - s''(x))dx \\ = & \sum_{i=1}^n \Big(\underbrace{s''(x)(y'(x) - s'(x))}_{=0} \Big|_{x=x_{i-1}}^{x_i} - \int_{x_{i-1}}^{x_i} s'''(x)(y'(x) - s'(x))dx \Big) \\ = & -\sum_{i=1}^n c_i \int_{x_{i-1}}^{x_i} y'(x) - s'(x)dx = -\sum_{i=1}^n c_i \big[(y(x_i) - s(x_i)) - (y(x_{i-1}) - s(x_{i-1})) \big] = 0 \end{aligned}$$

□

Die Aussage des Satzes ist so zu verstehen: Nach Voraussetzung soll y' mit s' mindestens an allen Punkten nicht-verschwindender Krümmung von s übereinstimmen. Unter all' diesen Funktionen hat s die kleinste Krümmung. Die Aussage dieses Satzes benötigen wir insbesondere zum Beweis des folgenden wichtigen Resultats über die Minimaleigenschaften der kubischen Splines.

Satz 1.1.2 (Minimaleigenschaft der kubischen Splines) *Es sei $\mathcal{T} = \{x_i\}$ eine Knotenfolge mit $a = x_0 < \cdots < x_{n+1} = b$ und $s \in \mathcal{S}^3(\mathcal{T})$ ein kubischer Spline, der neben den Interpolationsbedingungen $s(x_i) = f(x_i)$ eine der folgenden Randbedingungen erfülle:*

(i) $s'(a) = f'(a)$ *und* $s'(b) = f'(b)$ *(vollständige Randbedingung)*

(ii) $s''(a) = s''(b) = 0$ *(natürliche Randbedingung)*

(iii) $s'(a) = s'(b)$ *und* $s''(a) = s''(b)$ *(falls f periodisch mit Periode $b - a$ ist)* *(periodische Randbedingung)*

Ein solches $s \in \mathcal{S}^3(\mathcal{T})$ existiert und ist eindeutig bestimmt. Für jede interpolierende Funktion $y \in C^2[a,b]$, die dieselben Interpolations- und Randbedingungen erfüllt, gilt ferner

$$\|s''\|_2^2 = \int_a^b \big(s''(x)\big)^2 \, dx \leq \int_a^b \big(y''(x)\big)^2 \, dx = \|y''\|_2^2.$$

Nun noch eine Anmerkung zur Dimension von $\mathcal{S}^3(\mathcal{T})$ mit $\mathcal{T} = \{x_0, \ldots, x_{n+1}\}$. Für das Teilintervall $[x_i, x_{i+1}]$ der Länge $h_i := x_{i+1} - x_i$ wählen wir folgenden Ansatz:

$$s_i(x) := s(x)\big|_{[x_i,x_{i+1}]} = a_i(x - x_i)^3 + b_i(x - x_i)^2 + c_i(x - x_i) + d_i.$$

Für seinen Wert und die Ableitungen s', s'' an den Endpunkten erhalten wir

$$\begin{aligned}
s_i(x_i) &= d_i = f(x_i) && (1.4)\\
s_i(x_{i+1}) &= a_i h_i^3 + b_i h_i^2 + c_i h_i + d_i = f(x_{i+1}) && (1.5)\\
s_i'(x_i) &= c_i && (1.6)\\
s_i'(x_{i+1}) &= 3a_i h_i^2 + 2b_i h_i + c_i && (1.7)\\
s_i''(x_i) &= 2b_i && (1.8)\\
s_i''(x_{i+1}) &= 6a_i h_i + 2b_i && (1.9)
\end{aligned}$$

Gehen wir nun davon aus, dass auf dem ersten Intervall $[x_0, x_1]$ die Koeffizienten a_0, b_0, c_0, d_0 gegeben seien, sodass die Interpolationsbedingungen auf $[x_0, x_1]$ erfüllt sind. Durch die Interpolationsbedingungen und die Stetigkeit von s' und s'' an den Knoten folgt:

$$\begin{aligned}
s_1(x_1) &= d_1 = f(x_1) && (1.10)\\
s_1(x_2) &= a_1 h_1^3 + b_1 h_1^2 + c_1 h_1 + d_1 = f(x_2) && (1.11)\\
s_1'(x_1) &= c_1 = 3a_0 h_0^2 + 2b_0 h_0 + c_0 && (1.12)\\
s_1''(x_1) &= 2b_1 = 6a_0 h_0 + 2b_0 && (1.13)
\end{aligned}$$

Aus den obigen Gleichungen (1.10) - (1.13) folgt die Eindeutigkeit der a_1, b_1, c_1, d_1. Per Induktion folgt dann auch die eindeutige Bestimmung der weiteren a_k, b_k, c_k, d_k, d.h. der Raum $\mathcal{S}^3(\mathcal{T})$ mit $\mathcal{T} = \{x_0, \ldots, x_{n+1}\}$ besitzt $(n + 2) + 2 = n + 4$ Freiheitsgrade. Allgemein läßt sich zeigen:

$$\dim\ \mathcal{S}^k(\mathcal{T}) = n + k + 1.$$

Beweis von Satz 1.1.2. Die Interpolations- und Randbedingungen sind linear in s, und ihre Anzahl stimmt mit der Dimension von $\mathcal{S}_k(\mathcal{T})$ überein. Somit genügt es zu zeigen, dass für die Splinefunktion $f \equiv 0$ der triviale Spline $s \equiv 0$ einzige Lösung ist.

Da $y \equiv 0$ alle Bedingungen erfüllt, folgt mit Satz 6, dass auch $\|s''\| = 0$ gilt. Da s'' stetig, folgt somit auch $s'' = 0$, somit $s' = c_0$ und $s(x) = c_0 x + c_1$. Aus den Interpolationsbedingungen $s(x_i) = 0$ folgt sofort $s = 0$. □

Bemerkung 1.1.3 *Der Satz 1.1.2 gilt unter Verallgemeinerung der Randbedingungen für beliebige Splineräume $\mathcal{S}^k$ ($k \geq 3$). Siehe z.B. [Hämmerlin/Hoffmann], Seite 252.*

Berechnung kubischer Splines

Kommen wir nun zur Berechnung kubischer Splines. Zusätzlich zu den Daten $y_i := f(x_i)$ $(i = 0, \ldots, n+1)$ ist es zweckmäßig Variablen $y_i'' := s_i''(x_i) = s_{i-1}''(x_i)$ einzuführen und die Variablen $a_i, \ldots, d_i$ durch diese zu ersetzen. Aus den Gleichungen (1.4), (1.5) sowie (1.8) und (1.9) gewinnt man das Gleichungssystem

$$\begin{pmatrix} 0 & 0 & 0 & 1 \\ h_i^3 & h_i^2 & h_i & 1 \\ 0 & 2 & 0 & 0 \\ 6h_i & 2 & 0 & 0 \end{pmatrix} \begin{pmatrix} a_i \\ b_i \\ c_i \\ d_i \end{pmatrix} = \begin{pmatrix} s_i(x_i) \\ s_i(x_{i+1}) \\ s_i''(x_i) \\ s_i''(x_{i+1}) \end{pmatrix} = \begin{pmatrix} y_i \\ y_{i+1} \\ y_i'' \\ y_{i+1}'' \end{pmatrix}. \qquad (1.14)$$

Die Berechnung der Determinanten der Koeffizientenmatrix liefert

$$\det \begin{pmatrix} 0 & 0 & 0 & 1 \\ h_i^3 & h_i^2 & h_i & 1 \\ 0 & 2 & 0 & 0 \\ 6h_i & 2 & 0 & 0 \end{pmatrix} = -\det \begin{pmatrix} h_i^3 & h_i^2 & h_i \\ 0 & 2 & 0 \\ 6h_i & 2 & 0 \end{pmatrix} = -2 \det \begin{pmatrix} h_i^3 & h_i \\ 6h_i & 0 \end{pmatrix} = 12h_i^2$$

und damit die Eindeutigkeit der $a_i, \ldots, d_i$ und somit von s, falls s_i und s_i'' für alle $i = 0, \ldots, n+1$ bekannt sein sollten. Das Lösen von (1.14) liefert nun

$$d_i = y_i \qquad b_i = \frac{1}{2}y_i'' \tag{1.15}$$

$$a_i = \frac{1}{6h_i}(y_{i+1}'' - y_i'') \qquad c_i = \frac{1}{h_i}(y_{i+1} - y_i) - \frac{1}{6}h_i(y_{i+1}'' + 2y_i''). \tag{1.16}$$

Es lassen somit sich die kubischen Polynome $s_i(x)$ in jedem Teilintervall eindeutig bestimmen, wenn neben den Stützwerten y_k auch die Größen y_k'' bekannt sind. Damit wäre neben der Interpolationseigenschaft auch gleich die Stetigkeit der zweiten Ableitung von $s(x)$ gesichert.

Was nun zu zeigen wäre, ist, dass aus den y_i, y_i'' auch die Stetigkeit von $s(x)$ folgt. Setzen wir die Darstellung der a_i, b_i, c_i und d_i in (1.7) ein, so erhalten wir

$$\begin{aligned} s_i'(x_{i+1}) &= 3h_i^2 a_i + 2h_i b_i + c_i \\ &= 3h_i^2\big(\frac{1}{6h_i}(y_{i+1}'' - y_i'')\big) + 2h_i(\frac{1}{2}y_i'' + \frac{1}{h_i}(y_{i+1} - y_i) - \frac{1}{h_i}(y_{i+1}'' + 2y_i'') \\ &= h_i\big(\frac{1}{2}y_{i+1}'' - \frac{1}{2}y_i'' + y_i'' - \frac{1}{6}y_{i+1}'' - \frac{1}{3}y_i''\big) + \frac{1}{h_i}(y_{i+1} - y_i) \\ &= \frac{h_i}{6}(2y_{i+1}'' + y_i'') + \frac{1}{h_i}(y_{i+1} - y_i) \end{aligned}$$

bzw.

$$s_{i-1}'(x_i) = \frac{1}{h_{i-1}}(y_i - y_{i-1}) + \frac{h_{i-1}}{6}(2y_i'' + y_{i-1}'') \tag{1.17}$$

Die Stetigkeit von $s'(x)$ an den inneren Knoten liefert mit der letzten Gleichung (1.17) wegen der Darstellung von c_i aus (1.6) und (1.15) die Gleichung

$$\begin{aligned} &\frac{1}{h_{i-1}}(y_i - y_{i-1}) + \frac{1}{6}h_{i-1}(2y_i'' + y_{i-1}'') \\ = \;& s_{i-1}'(x_i) \overset{!}{=} s_i'(x_i) = c_i = \frac{1}{h_i}(y_{i+1} - y_i) - \frac{1}{6}h_i(y_{i+1}'' + 2y_i'') \end{aligned}$$

Sortieren von y_k'' nach links, y_k nach rechts und anschließende Multiplikation mit 6 liefert

$$h_{i-1}y_{i-1}'' + 2(h_{i-1} + h_i)y_i'' + h_i y_{i+1}'' = \frac{6}{h_i}(y_{i+1} - y_i) - \frac{6}{h_{i-1}}(y_i - y_{i-1}). \tag{1.18}$$

Diese Bedingung muss für alle inneren Knoten $x_1, \ldots, x_n$ erfüllt sein und liefert somit n Gleichungen für die Unbekannten $y_0'', \ldots, y_{n+1}''$.
Allerdings reichen die n Gleichungen für $n+2$ Unbekannte $y_0'', \ldots, y_{n+1}''$ nicht aus um diese eindeutig zu bestimmen. Untersuchen wir nun die unterschiedliche Behandlung der Randbedingungen.

Berücksichtigung natürlicher und vollständiger Randbedingungen

Die vollständige Randbedingung $s'(a) = f'(a)$, bzw. $s'(b) = f'(b)$ liefert aufgrund von (1.6), (1.7) und (1.15) die weitere Gleichung

$$s'(a) = s_0'(x_0) = c_0 = \frac{1}{h_0}(y_1 - y_0) - \frac{1}{6}(y_1'' + 2y_0'') \overset{!}{=} f'(a).$$

Somit folgt

$$2h_0 y_0'' + h_0 y_1'' = \frac{6}{h_0}(y_1 - y_0) - 6f'(a), \tag{1.19}$$

bzw. aus

$$\begin{aligned} s'(b) &= s'_n(x_{n+1}) = 3a_n h_n^2 - 2b_n h_n + c_n \\ &= \frac{h_n}{2}(y''_{n+1} - y''_n) + h_n y''_n + \frac{1}{h_n}(y_{n+1} - y_n) - \frac{1}{6}h_n(y''_{n+1} + 2y''_n) \\ &= h_n\big(\frac{1}{2}y''_{n+1} - \frac{1}{2}y''_n + y''_n - \frac{1}{6}y''_{n+1} - \frac{1}{3}y''_n\big) + \frac{1}{h_n}(y_{n+1} - y_n) \\ &= h_n(\frac{1}{3}y''_{n+1} + \frac{1}{6}y_n) + \frac{1}{h_n}(y_{n+1} - y_n) \stackrel{!}{=} f'(b), \end{aligned}$$

für den rechten Rand

$$h_n y''_n + 2h_n y''_{n+1} = -\frac{6}{h_n}(y_{n+1} - y_n) + 6f'(b) \tag{1.20}$$

Für die natürlichen Randbedingung $y''_0 = y''_{n+1} = 0$ ergeben sich folgende Gleichungen

$$y''_0 = 0 \tag{1.21}$$
$$y''_{n+1} = 0 \tag{1.22}$$

Natürliche und vollständige Randbedingungen können auch gemischt auftreten, d.h. an der Stelle x_0 ist neben $f(x_0)$ auch die Ableitung $f'(x_0)$ vorgegeben und an der Stelle x_{n+1} gilt $y_{n+1} = 0$.
Im Falle $n = 5$ erhalten wir daraus folgendes lineares Gleichungssystem:

$$\begin{pmatrix} \star & \star & \star & \star & \star & \star \\ h_0 & 2(h_0+h_1) & h_1 & & & \\ & h_1 & 2(h_1+h_2) & h_2 & & \\ & & h_2 & 2(h_2+h_3) & h_3 & \\ & & & h_3 & 2(h_3+h_4) & h_4 \\ \star & \star & \star & \star & \star & \star \end{pmatrix} \cdot \begin{pmatrix} y''_0 \\ \vdots \\ \\ \\ \vdots \\ y''_5 \end{pmatrix}$$

$$= \begin{pmatrix} \star \\ \frac{6}{h_1}(y_2 - y_1) - \frac{6}{h_0}(y_1 - y_0) \\ \frac{6}{h_2}(y_3 - y_2) - \frac{6}{h_1}(y_2 - y_1) \\ \frac{6}{h_3}(y_4 - y_3) - \frac{6}{h_2}(y_3 - y_2) \\ \frac{6}{h_4}(y_5 - y_4) - \frac{6}{h_3}(y_4 - y_3) \\ \star \end{pmatrix} \tag{1.23}$$

Die in diesem Schema mittels $\star$ gekennzeichnete erste und letzte Zeile ist dabei durch die Wahl der Randbedingungen (1.19), (1.8) bzw. (1.21), (1.22) gegeben.
Für vollständige Randbedingungen sind in (1.23) erste und letzte Zeile durch die Gleichungen (1.19) und (1.21) zu ersetzen, sodass wir erhalten:

$$\begin{pmatrix} 2h_0 & h_0 & & & \\ \star & \star & \star & & \\ & & \ddots & & \\ & & \star & \star & \star \\ & & & h_n & 2h_n \end{pmatrix} \begin{pmatrix} y''_0 \\ \star \\ \vdots \\ \star \\ y''_{n+1} \end{pmatrix} = \begin{pmatrix} \frac{6}{h_0}(y_1 - y_0) - 6f'(a) \\ \star \\ \vdots \\ \star \\ -\frac{6}{h_n}(y_{n+1} - y_n) + 6f'(b) \end{pmatrix}$$

Hierbei sind die mit $\star$ gekennzeichneten Zeilen 1 bis n dem System (1.23) zu entnehmen.

Analog erhalten wir für die Randbedingungen $y_0'' = f''(a)$, $y_{n+1}'' = f''(b)$

$$\begin{pmatrix} 1 & & & & & \\ \star & \star & \star & & & \\ & & \ddots & & & \\ & & \star & \star & \star & \\ & & & & 1 \end{pmatrix} \begin{pmatrix} y_0'' \\ \star \\ \vdots \\ \star \\ y_{n+1}'' \end{pmatrix} = \begin{pmatrix} f''(a) \\ \star \\ \vdots \\ \star \\ f''(b) \end{pmatrix}$$

Da im letzten Gleichungssystem y_0'' und y_{n+1}'' explizit bekannt sind, können wir es wie folgt reduzieren:

$$\begin{pmatrix} 2(h_0+h_1) & h_1 & & & \\ h_1 & 2(h_1+h_2) & h_2 & & \\ & & \ddots & & \\ & & h_{n-1} & 2(h_{n-1}+h_n) \end{pmatrix} \cdot \begin{pmatrix} y_1'' \\ \vdots \\ \vdots \\ y_n'' \end{pmatrix} = \begin{pmatrix} \star & -h_0 y_0'' \\ \star & \\ \star & \\ \star & -h_n y_{n+1}'' \end{pmatrix}$$

Bemerkung 1.1.4 *Man beachte, dass $y_0'' = c_0$ und $y_{n+1}'' = c_1$ mit $c_0, c_1 \in \mathbb{R}$ nicht die Matrix, sondern nur die rechte Seite modifiziert, damit also auch eine Verallgemeinerung der natürlichen Randbedingungen möglich ist.*

Berücksictigung periodischer Randbedingungen

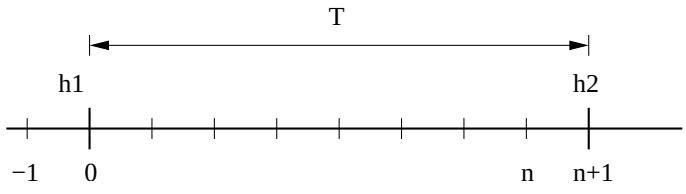
Die Stützstellen $x_0 < \cdots < x_{n+1}$ seien nun so festgelegt, dass $x_{n+1} = x_0 + T$, wobei T die Periode der gesuchten Funktion darstelle.

Die periodischen Randbedingungen sind $f(x_0) = f(x_{n+1})$, $f'(x_0) = f'(x_{n+1})$ sowie $f''(x_0) = f''(x_{n+1})$. Mit den Größen $y_0 = y_{n+1}, y_1, \ldots, y_n$ und den Unbekannten $y_0'' = y_{n+1}'', y_1'', \ldots, y_n''$ ist die Stetigkeit an den $n+1$ Stützstellen $x_0, \ldots, x_n$ zu erfüllen (beachte, dass die Stetigkeit bei x_0 der bei x_{n+1} aufgrund der Periodizität entspricht).

Für die inneren Knoten $x_1, \ldots, x_n$ sind die Gleichungen gegeben durch (1.18). Aber auch zum Knoten x_0 sind die Gleichungen durch (1.18) gegeben, wenn man bei der Formulierung

$$\begin{aligned} h_{-1} &= x_0 - x_{-1} = x_{n+1} - x_n = h_n \\ y_{-1}'' &= y_n'' \text{ und } y_{-1} = y_n \end{aligned}$$

beachtet, da zu jedem Knoten x_i nur die Informationen y_{i-1}, y_i, y_{i+1} und $y_{i-1}'', y_i'', y_{i+1}''$ benötigt werden, d.h. die Daten vom linken, rechten Nachbarn sowie vom Knoten selbst



Das System lautet dann allgemein für $n = N, x_{N+1} = x_0 + T$

$$\begin{pmatrix} 2(h_N+h_0) & h_0 & & & & h_N \\ h_0 & 2(h_0+h_1) & h_1 & & & \\ & h_1 & 2(h_1+h_2) & h_2 & & \\ & & \ddots & \ddots & & \\ & & h_{N-2} & 2(h_{N-2}+h_{N-1}) & h_{N-1} & \\ h_N & & & h_{N-1} & 2(h_{N-1}+h_N) & \end{pmatrix} \cdot \begin{pmatrix} y_0'' \\ y_1'' \\ y_2'' \\ \vdots \\ \vdots \\ y_{N-1}'' \\ y_N'' \end{pmatrix}$$

$$= \begin{pmatrix} \frac{6}{h_0}(y_1-y_0) - \frac{6}{h_N}(y_0-y_N) \\ \frac{6}{h_1}(y_2-y_1) - \frac{6}{h_0}(y_1-y_0) \\ \frac{6}{h_2}(y_3-y_2) - \frac{6}{h_1}(y_2-y_1) \\ \vdots \\ \frac{6}{h_{N-1}}(y_N-y_{N-1}) - \frac{6}{h_{N-2}}(y_{N-1}-y_{N-2}) \\ \frac{6}{h_N}(y_0-y_N) - \frac{6}{h_{N-1}}(y_N-y_{N-1}) \end{pmatrix}$$

Eine Matlab-Realisierung zur Berechnung der Koeffizienten a_i, b_i, c_i, d_i für alle Intervalls $[x_i, x_{i+1}]$ $(i = 0, \ldots, n)$ ist im Folgenden wieder gegeben.

MATLAB-Funktion: CoeffSpline.m

```
function coeff = CoeffSpline(t,y,kind,param)
% param(1,1)=f'(a) bzw. f''(a) (abhängig von der Wahl von kind)
% param(2,1)=f'(b) bzw. f''(b)
n=length(t);
dy = y(2:end)-y(1:end-1);
%% Berechnung des "Kerns" von A und b
h1 = t(2:n)-t(1:n-1);
h2 = t(3:end)-t(1:end-2);
A = sparse(n,n);
B = [h1,[2*h2;0],[h1(2:end);0]];
A(2:n-1,:)=spdiags(B,[0,1,2],n-2,n);
b = zeros(n,1);
b(2:n-1)=6*((y(3:n)-y(2:n-1))./h1(2:n-1)-...
          (y(2:n-1)-y(1:n-2))./h1(1:n-2));
%% Erweiterung von A und b für unterschiedliche Randbedingungen
switch kind
  case 'nat'
    A(1,1)=1; A(n,n)=1;
    b(1)=param(1); b(end)=param(2);
  case 'per'
    if y(1)~=y(end)
      error('This data is not applicable for periodic splines')
    end
    A(1,[1,2,n-1])=[2*(h1(1)+h1(end)),h1(1),h1(end)];
    A(n,[1,n])=[1,-1];
    b(1)=6*((y(2)-y(1))/h1(1)-(y(1)-y(end-1))/h1(end));
  case 'compl'
    A(1,[1,2])=[2*h1(1),h1(1)];
    A(n,[n-1,n])=[h1(end),2*h1(end)];
    b(1)=6*(y(2)-y(1)-param(1));
    b(n)=-6*(y(end)-y(end-1)+param(2));
```

```
32    otherwise
33      error('This kind does not exist!')
34  end
35  %% Berechnung der y''
36  y2d=A\b;
37  %% Berechnung der Koeffizienten a,b,c,d
38  coeff=[y(1:end-1),...
39           (y(2:end)-y(1:end-1))./h1(1:end)-h1(1:end).*(y2d(2:end)+2*
                y2d(1:end-1))/6,...
40           y2d(1:end-1)/2,(y2d(2:end)-y2d(1:end-1))./(6*h1(1:end))]';
```

1.2 Punktauswertung kubischer Splines

Nachdem wir nun im vorletzten Kapitel analysiert haben, wie sich aus den Daten $a = x_0 < x_1 < \cdots < x_{n+1} = b$ und $y_0, \ldots, y_{n+1}$ ein kubischer Spline bestimmen läßt, stellt sich nun die Frage, wie wir diesen auswerten können. Im Gegensatz zu einem Polynom sind die Splines nur jeweils stückweise definiert, d.h. wollen wir an einer Stelle $x \in [a, b]$ den Spline $s(x)$ auswerten, müssen wir erst k mit $x \in [x_k, x_{k+1}]$ bestimmen.

Bei der trivialen Realisierung, bei der man nacheinander $j = 0, 1, \ldots, n$ durchgeht und testet ob x in $[x_j, x_{j+1}]$ liegt, benötigt man bei einer äquidistanten Unterteilung $|x_{j+1} - x_j| =: h$ $(j = 0, \ldots, n)$ in $n+1$ Teilintervalle durchschnittlich $n/2$ Abfragen. Da x mit der gleichen Wahrscheinlichkeit $1/(n+1)$ in einem der Intervalle $[x_j, x_{j+1}]$ $(j = 0, \ldots, n)$ liegt, wird bei einem x, welches im letzten $\big((n+1)$-ten$\big)$ Intervall $[x_n, x_{n+1}]$ oder vorletzten Intervall $[x_{n-1}, x_n]$ vorher n-mal ein Test auf „x liegt im Intervall“ durchgeführt. Für ein x, welches im j-ten Intervall $[x_{j-1}, x_j]$ liegt, wird eine solche Überprüfung j-mal durchgeführt, d.h. durchschnittlich werden $\frac{1}{n+1} + \frac{2}{n+1} + \ldots + \frac{n-1}{n+1} + \frac{n}{n+1} + \frac{n}{n+1} = \frac{(n+1)n}{2(n+1)} + \frac{n}{n+1} \approx \frac{n}{2} + 1$ Tests benötigt.

Wie man schnell sieht, ist man mit einer binären Suche deutlich schneller. Vereinfachen wir die Voraussetzungen insofern, dass wir von $n = 2^s$ Intervallen ausgehen und unser x liege mit der gleichen Wahrscheinlichkeit in einem der Teilintervalle. Mit einem Test ob x in den ersten 2^{s-1} oder den letzten 2^{s-1} Intervallen liegt, reduzieren wir die Problemgröße auf die Hälfte und erhalten nach s Tests das gesuchte Intervall.

Gilt nun $2^{s-1} < n \leq 2^s$, so wähle man $2^s - n$ virtuelle Intervalle $[b, b]$ und nach s Bisektionen hat man das gesuchte Intervall gefunden. Die Anzahl der Tests ist für $2^{s-1} < n \leq 2^s$ gerade $s = \lceil \log_2 n \rceil$. Für $n = 100(10000)$ benötigt man durchschnittlich bei der sequentiellen Suche dem $501(5001)$ Tests und bei der binären Suche nur $7(14)$ Tests.

Nachfolgend führen wir die entsprechenden **Matlab**-Zeilen zur Auswertung mittels sequentieller und binärer Suche an. In beiden Fällen wird das Vorgehen durch die entsprechende **Matlab**-Ausgabe verdeutlicht:

MATLAB-Funktion: sequentiellesuche.m

```
1  function k = sequentielle_suche(x,x0)
2  % x_1 < x_2 < x_3 <  < x_n
```

```
% Finde kleinstes k, sodass  x0 in [x_k,x_(k+1)]
% und setze k = 0, wenn es kein solches k gibt
for k = 1:length(x)-1
    if x(k) <= x0 && x0 <= x(k+1)
      return
    end
end
k = 0;
```

MATLAB-Funktion: binaeresuche.m

```
function k_unten = binaeresuche(x,x0)
% x_1 < x_2 < x_3 <   < x_n
% Finde kleinstes k so dass x0 in [x_k,x_(k+1)]
% und setze k = 0, wenn es kein solches k gibt
if x0 < x(1) || x(end) < x0
    k_unten = 0;
    return
end
k_unten = 1;
k_oben = length(x);
while k_oben - k_unten > 1
    k_mitte = floor((k_unten + k_oben)/2);   %rundet -> -inf
    if x(k_unten)<= x0 && x0 <= x(k_mitte)
         k_oben = k_mitte;
    else
         k_unten = k_mitte;
    end
end
```

MATLAB-Beispiel:

Als Ausgabe erhalten wir:

```
>> N=10^6,x=[1:N];x0=N*rand(100,1);
>> tic, for k=1:100,
   sequentiellesuche(x,x0(k)); end, toc
N =
     1000000
Elapsed time is 1.844000 seconds.
>> N=10^6,x=[1:N];x0=N*rand(100,1);
>> tic, for k=1:100,
   binaeresuche(x,x0(k)); end, toc
N =
     1000000
Elapsed time is 0.016000 seconds.
```

Suche mit korrelierten Daten

Häufig kommt man bei der Auswertung des Splines noch zu einer speziellen Situation, nämlich die Auswertung von s an einer aufsteigenden Folge $(t_j \leq t_{j+1})$ von Punkten $t_j \in [a,b]$ $(j = 1,\ldots,m)$. Gilt $t_j \in [x_{k_j}, x_{k_j+1}]$ $(k_j \in \{0,\ldots,n\})$, so ist klar, dass man t_{j+1} nur noch in der Teilmenge $[x_{k_j}, b]$ suchen muss. Wäre nur noch ein Intervall zu suchen, böte die Bisektionsmethode einen effizienten Suchalgorithmus, wenn aber noch mehrere Intervalle für $t_{j+1},\ldots,t_m$ zu lokalisieren sind, wird das nächste k_{j+1} in der Nähe des zuletzt bestimmten k_j liegen. Dies muss aber keineswegs dasselbe oder auch das nächste Intervall sein. Die Idee des folgenden „Jagd"-Algorithmus ist es, das nächste k_{j+1} durch **größer** werdende Schritte einzuschachteln.

Gilt $t_{j+1} \notin [x_{k_j}, x_{k_j+1}]$ so teste man nacheinander

$$\begin{aligned} t_{j+1} &\in [x_{k_j+1}, x_{k_j+2}]\,? \\ t_{j+1} &\in [x_{k_j+2}, x_{k_j+4}]\,? \\ t_{j+1} &\in [x_{k_j+4}, x_{k_j+8}]\,? \\ &\vdots \end{aligned}$$

Hat man hierdurch ein Intervall identifiziert, wendet man bezüglich diesem Intervall die Bisektionsmethode an. Im „Worstcase" benötigt man zweimal länger als mit der Bisektionssuche, aber im besten Fall ist man um den Faktor $\log_2 n$ schneller.

Nachfolgend der oben erwähnte „Jagd-Algorithmus"in **MATLAB**-Implementierung:

MATLAB-Funktion: lokalisierejagd.m

```
function ks = lokalisierejagd(xs,x0)
% xs(1) < xs(2) < xs(3) < ... < xs(n)
% Finde für alle x0's die kleinsten k's, sodass x0 in [xs(k),xs(k
    +1)]
  ks = zeros(length(x0),1);
  n = length(xs);
  k = 1;
  for j = 1:length(x0)
    inc = 1;
    k_next = k + 1;
    while k_next <= n && x0(j) > xs(k_next) % hunting
      k = k_next;
      k_next = k_next + inc;
      inc = 2 * inc;
    end
    k_next = min(k_next,n);
    if k_next > k+1                          % bisection
      k = k + binaeresuche(xs(k:k_next),x0(j)) - 1;
    end
    ks(j) = k;
  end
end
```

MATLAB-Beispiel:

Besteht der Spline aus n Intervallen und gesucht ist die Auswertung an $m \ll n$ Stützstellen so benötigen `lokalisierejagd` und `binaeresuche` etwa gleich viel Zeit.

```
>> m = 20001; n = 80001;
>> nodes = linspace(0,1,n);
>> x0 = linspace(0,1,m);
>> tic, s = lokalisierejagd(nodes,x0);toc
Elapsed time is 0.448687 seconds.
>> t=zeros(m,1);
>> tic,for j = 1:m,t(j) = binaeresuche(
   nodes,x0(j)); end,toc
Elapsed time is 0.349985 seconds.
```

Ist jedoch die Anzahl der Auswertung deutlich größer als die Anzahl der Intervalle, auf dem der Spline stückweise definiert ist, so ist `lokalisierejagd` deutlich schneller als `binaeresuche`.

```
>> m = 80001; n = 20001;
>> nodes = linspace(0,1,n);
>> x0 = linspace(0,1,m);
Elapsed time is 0.023416 seconds.
>> tic, s = lokalisierejagd(nodes,x0);toc
>> t=zeros(m,1);
>> tic,for j = 1:m,t(j) = binaeresuche(
   nodes,x0(j)); end,toc
Elapsed time is 1.382369 seconds.
```

1.3 Parametrisierte Kurven und Flächen

Implizite und parametrisierte Darstellung

Die beiden häufigsten Methoden um Kurven oder Flächen mathematisch zu beschreiben sind die implizite Darstellung und die parametrisierte Form.
Die implizite Darstellung einer Kurve in der xy-Ebene hat die Form $f(x, y) = 0$. Zu einer gegeben Kurve ist diese Datsellung eindeutig bis auf eine multiplikative Konstante. Ein Beispiel ist der Einheitskreis, definiert durch die Gleichung $f(x, y) = x^2 + y^2 - 1 = 0$.
In der Parameterdarstellunmg wird jede Koordinate eines punkts auf der Kurve seperat durch eine explizite Funktion eines unabhängigen Parameters dargestellt,

$$\mathbf{C}(t) = (x(t), y(t)) \quad a \le t \le b\,.$$

Somit ist $\mathbf{C}(t)$ eine vektorwertige Funktion des Parameteres t. Obwohl das Intervall $[a, b]$ beliebig sein kann, wird es üblicherweise auf $[0, 1]$ normiert. Der erste Quadrant des Einheitskreises ist definiert durch die Paremeterdarstellung

$$x(t) = \cos(t),\ y(t) = \sin(t) \quad a \le t \le \pi/2\,.$$

Substituiert man $u = \tan(t/2)$ so erhällt man die alternative Darstellung

$$x(u) = \frac{1-u^2}{1+u^2},\ y(u) = \frac{2u}{1+u^2} \quad 0 \le u \le 1\,.$$

Die parametrische Darstellung ist folglich nicht eindeutig.
Beide Darstellungsformen haben Vor- und Nachteile, von denen einige hier genannt seien.

- Fügt man eine z-Koordinate hinzu, so lässt sich die gegebene Parameterdarstellung einer Kurve einfach in 3-dimensionalen Raum einbetten. Durch die implizite Form lassen sich nur Kurven in der xy (oder yz oder yz) Ebene darstellen.

- Parametrisierte Kurven haben eine natürliche Richtung (von $\mathbf{C}(a)$ zu $\mathbf{C}(b)$ für $a \leq t \leq b$. Somit lassen sich einfach geordnete Folgen von Punkten erzeugen. Implizit gegebene Kurven haben diese Eigenschaft nicht.
- In der Parameterdarstellung muss man manchmal mit „Anomalien kämpfen"', die nicht im Zusammenhang stehen mit der wirklichen Geometrie. Ein Beispiel ist die Einheitskugel. Verwendet man Kugelkoordianten, so sind die Pole algorithmisch schwierige Punkte, obwohl sie sich von den anderen Punkten nicht unterscheiden.
- Die Komplexität vieler geometrischer Operationen und Manipulationen hängt stark von der Darstellung ab. Die Berechnung eines Punktes auf einer Kurve ist schwierig in der impliziten Darstellung. Die Entscheidung, ob ein Punkt auf einer Kurve oder Fläche liegt ist jedoch in impliziten Darstellung einfacher.
- Unbeschränkte Geometrien lassen sich nur schwer mit einer Parameterdarstellung beschreiben.

Lässt man beliebige Koordinatenfunktionen $x(t)$, $y(t)$, $z(t)$ zur Beschreibung von Kurven zu, so erhällt man eine riesige Auswahl an möglichen Kurven. Möchte man dies aber mit Hilfe eines Rechners umsetzen, so gibt es einige Restriktionen zu berücksichtigen. Am besten wäre es, man beschränkt sich auf eine Klasse von Funktionen, die

- die gewünschten Kurven präzise genug darstellt, wie sie für Berechnungen oder Darstellungen benötigt werden,
- einfach, effizient und stabil sind,
- wenig Speicherplatz benötigen,
- mathematisch einfach gut verstanden sind (d.h. keine Heuristiken).

Eine naheliegende Wahl von Funktionen wären die Polynome. Obwohl sie die letzten beiden Punkte in der Wunschliste erfüllen, gibt es mehere Beispiele wichtiger Kurven und Flächen, die sich nicht durch Polynome darstellen lassen, z.B. Kreise und Kugeln.
Die Darstellung einer Kurve in monomialer Basis n-ten Grades ist gegeben durch

$$\mathbf{C}(t) = (x(t), y(t), z(t)) = \sum_{j=0}^{n} \mathbf{a}_j t^j \quad 0 \leq t \leq 1$$

mit $\mathbf{a}_j = (x_j, y_j, z_j)$. Zu einem gegebenem t_0 lässt sich der Punkt $\mathbf{C}(t_0)$ möglichst effizient mit dem Horner-Schema berechnen:

- für den Grad $= 1$: $\mathbf{C}(t_0) = \mathbf{a}_0 + t_0\mathbf{a}_1$
- Grad $= 2$: $\mathbf{C}(t_0) = \mathbf{a}_0 + t_0(\mathbf{a}_1 + t_0\mathbf{a}_2)$
- Grad $= 3$: $\mathbf{C}(t_0) = \mathbf{a}_0 + t_0(\mathbf{a}_1 + t_0(\mathbf{a}_2 + t_0\mathbf{a}_3))$
- $\vdots$
- Grad $= n$: $\mathbf{C}(t_0) = \mathbf{a}_0 + t_0(\ldots t_0(\mathbf{a}_{n-2} + \ldots t_0(\mathbf{a}_{n-1} + t_0\mathbf{a}_n)))$

In Matlab sieht der Algorithmus wie folgt aus.

MATLAB-Funktion: horner.m

```
function f = horner(a,t0)
% Compute point on a power basis curve
f = a(:,end);
for k = size(a,2)-1:-1:1
  f = f.*t0+a(:,k);
end
```

1.4 Bernstein-Polynome und Bézier-Kurven

Die monomiale Basis ist nicht die einzige, um Polynome darzustellen. In Rahmen der Interpolation wurden auch schon die Lagrange- und Newton-Basis diskutiert. Wir definieren nun zuerst eine weitere Basis, nämlich die Bernstein-Polynome. Obwohl die parametrisierten Funktionen, dargestellt in monomialer Basis oder mit Bernstein-Polynomen, mathematisch äquivalent sind, so ist die Darstellung mit Hilfe der Bernstein-Polynome für die Darstellung von Kurven und Flächen deutlich geeigneter. An entsprechender Stelle kommen wir auf diesen Punkt zurück.

Definition 1.4.1 (Bernstein-Polynom) *Das i-te* **Bernstein***-Polynom vom Grad n bezüglich des Intervalls $[0,1]$ ist das Polynom $B_i^n \in \mathbb{P}_n$ mit*

$$B_i^n(t) = \frac{n!}{i!(n-i)!} t^i (1-t)^{n-i}, \quad i = 0, \ldots, n. \tag{1.24}$$

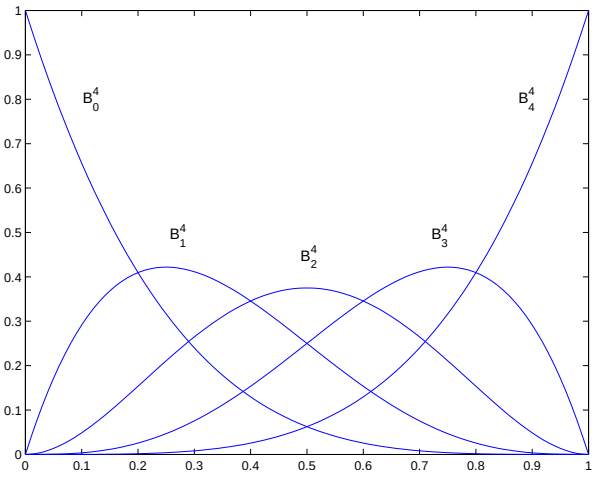


Abb. 1.2: Bernstein-Polynome $B_0^4, \ldots, B_4^4$ auf dem Intervall $[0,1]$.

Satz 1.4.2 (Eigenschaften der Bernstein-Polynome) *Die Bernstein-Polynome haben folgende Eigenschaften:*

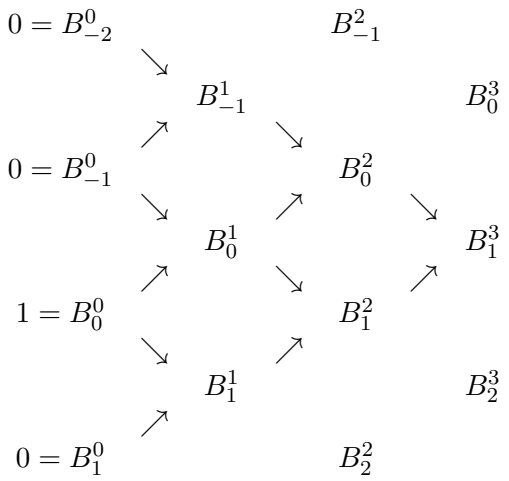
(i) $B_i^n(t) \geq 0$ *für alle* i, n *und* $0 \leq t \leq 1$ *(Positivität).*

(ii) $\sum_{i=0}^n B_i^n(t) = 1$ *für alle* $0 \leq t \leq 1$ *(Zerlegung der Eins).*

(iii) $B_0^n(0) = B_n^n(1) = 1.$

(iv) $B_i^n(t)$ *hat genau ein Maximum im Intervall* $[0,1]$*, und zwar bei* $t = i/n$.

(v) $B_i^n(t) = B_{n-i}^n(1-t)$ *für* $i = 0, \ldots, n$ *(Symmetrie).*



Tab. 1.1: Berechnung von B^3_1.

(vi) $B^n_i(t) = (1-t)B^{n-i}_i(t) + tB^{n-1}_{i-1}(t)$ *(Rekursionsformel); wir definieren* $B^n_i(t) \equiv 0$ *for* $i < 0$ *oder* $i > n$.

(vii)

$$\frac{d}{dt}B^n_i(t) = n\,\left(B^{n-1}_{i-1}(t) - B^{n-1}_i(t)\right)$$

mit $B^{n-1}_{-1}(t) \equiv B^{n-1}_n(t) \equiv 0$.

Die Gleichung (1.24) liefert $B^0_0(t) = 1$. Aus der Eigenschaft Satz 1.4.2.vi gewinnen wir die linearen und quadratischen Bernstein-Polynome

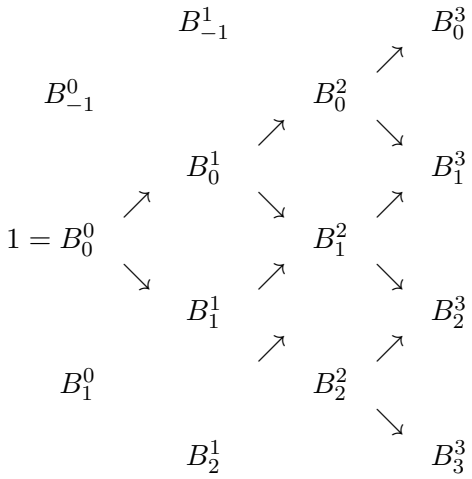
$$\begin{aligned}
B^1_0(t) &= (1-t)B^0_0(t) + tB^0_{-1}(t) = 1-t \\
B^1_1(t) &= (1-t)B^0_1(t) + tB^0_0(t) = t \\
B^2_0(t) &= (1-t)B^1_0(t) + tB^1_{-1}(t) = (1-t)^2 \\
B^2_1(t) &= (1-t)B^1_1(t) + tB^1_0(t) = 2t(1-t) \\
B^2_2(t) &= (1-t)B^1_2(t) + tB^1_1(t) = t^2
\end{aligned} \tag{1.25}$$

Die Eigenschaft Satz 1.4.2.vi liefert einen einfachen Algorihmus, um Werte der Bernstein-Polynome zu einem gegebenen t zu bestimmen. In Tabelle 1.1 ist dargestellt, welche B^0_k benötigt werden, um B^3_1 zu berechnen. Berücksichtigt man die Nulleinträge ($B^0_{-2} = B^0_{-1} = B^0_1 = 0$), d.h. man vernachlässigt die Terme von denen man weiss das sie verschwinden, so lassen sich alle kubischen Bernstein-Polynome, wie in der folgenden Tabelle 1.2 dargestellt, effizient bestimmen.

Die Funktion `AllBernstein.m` kombiniert das in Tabelle 1.2 dargestellte Vorgehen mit Gleichung (1.24) um die Bernstein-Polynome n-ten Grades an einer gegeben Stelle t zu bestimmen.

MATLAB-Funktion: AllBernstein.m

```
function B = AllBernstein(n,x)
% Compute all n-th Bernstein polynomials
B = zeros(n+1,1);
B(1) = 1;
for j=1:n
  saved = 0;
```



Tab. 1.2: Berechnung aller kubischen Bernstein-Polynome.

```
7     for k=1:j
8       temp = B(k);
9       B(k) = saved + (1-x) * temp;
10      saved = x * temp;
11    end
12    B(j+1) = saved;
13  end
```

MATLAB-Beispiel:

Die folgenden Zeilen stellen die Bernstein-Polynome $B_0^8, \ldots, B_8^8$ graphisch dar.

```
>> n = 8;  no = 100;
>> t = linspace(0,1,no);
>> B = zeros(n+1,no);
>> for k=1:no,B(:,k)=AllBernstein(n,t(k));
   end
>> hold on, for k=1:n+1, plot(t,B(k,:)),
   end
```

Nun zu Kurven im Raum, die man bequem mittels der Bernstein-Polynome definieren und effizient auswerten kann.

Definition 1.4.3 (Bézierkurve) *Gegeben seien $n+1$* **Kontrollpunkte** *$\mathbf{P}_j \in \mathbb{R}^d$ ($d \in \mathbb{N}$). Eine* **Bézierkurve** *n-ten Grades zu gegebenen $n+1$ Punkten $\mathbf{P}_j \in \mathbb{R}^d$ ($i = 0, \ldots, n$, $d \in \mathbb{N}$) ist für $t \in [0,1]$ definiert als*

$$\mathbf{C}(t) = \sum_{j=0}^{n} B_j^n(t)\, \mathbf{P}_i\,. \tag{1.26}$$

Bemerkung 1.4.4 *Da die Bernstein-Polynome eine Zerlegung der Eins bilden, ist die Summe ausgewertet für ein festes t nichts anderes als die Linearkombination der gegebenen Punkte $\mathbf{P}_j$. Diese Punkte heißen* ***Kontrollpunkte*** *zur Splinekurve s.*

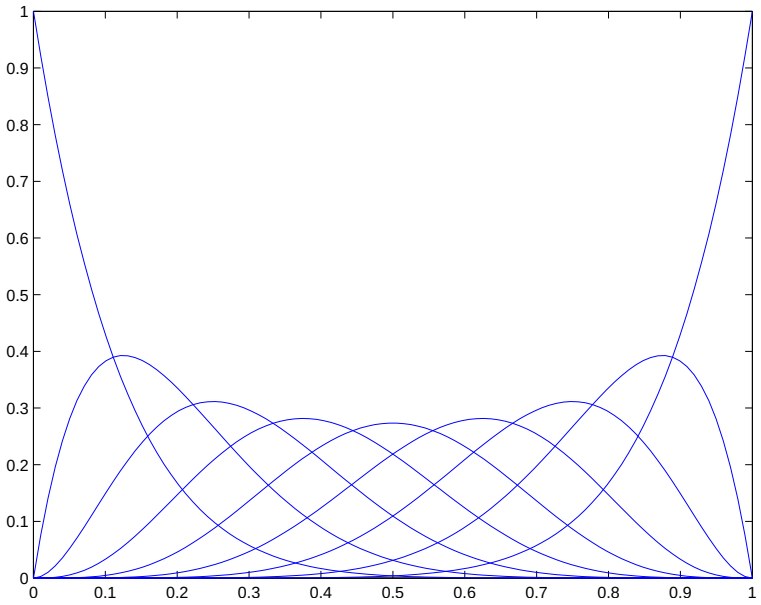


Abb. 1.3: Ergebnis der Matlab-Zeilen.

Bemerkung 1.4.5 *Ist $d = 2$, so heißt die geradlinige Verbindung der Punkte $\mathbf{P}_0, \ldots, \mathbf{P}_n$ das* ***Kontrollpolygon****.*

Für $n = 2$ und $\mathbf{C}(t) = \sum_{j=0}^{2} B_j^2(t)\mathbf{P}_j$ gilt

$$\begin{aligned}\mathbf{C}(t) &= (1-t)^2\mathbf{P}_0 + 2t(1-t)\mathbf{P}_1 + t^2\mathbf{P}_2 \\ &= (1-t)\Big(\underbrace{(1-t)\mathbf{P}_0 + t\,\mathbf{P}_1}_{\text{linear}}\Big) + t\Big(\underbrace{(1-t)\mathbf{P}_1 + t\,\mathbf{P}_2}_{\text{linear}}\Big)\,.\end{aligned}$$

Somit kann $\mathbf{C}(t)$ als Linearkombination von zwei Bézierkurven 1-ten Grades bestimmt werden. Betrachten wir dies nun allgemeiner. Bezeichnen wir eine beliege Bézierkurve n-ten Grades mit

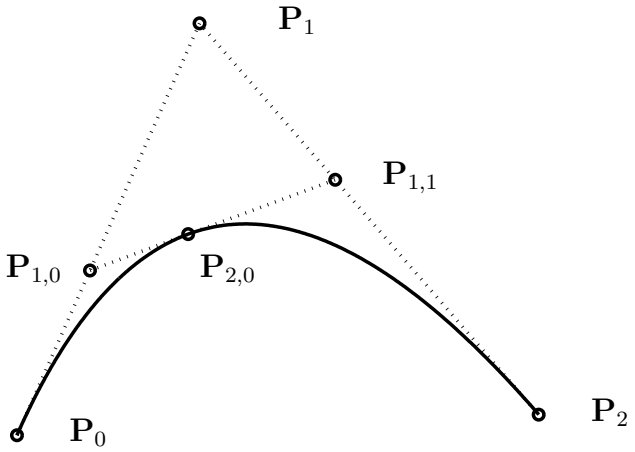


Abb. 1.4: Berechnung eines Punktes durch wiederholte lineare Interpolation für $t = 2/5$.

$\mathbf{C}_n(P_0, \ldots, P_n)$, dann liefert uns die Rekursion 1.4.2.vi die Darstellung

$$\mathbf{C}_n(t : P_0, \ldots, P_n) = (1-t)\mathbf{C}_{n-1}(t : P_0, \ldots, P_{n-1}) + t\mathbf{C}_{n-1}(t : P_1, \ldots, P_n)\,.$$

Dies liefert ein rekursives Verfahren zur Bestimmung von $\mathbf{C}(t_0) = \mathbf{P}_{n,0}(t_0)$ auf einer Bézierkurve n-ten Grades, nämlich

$$\mathbf{P}_{k,j}(t_0) = (1-t_0)\mathbf{P}_{k-1,j}(t_0) + t_0\mathbf{P}_{k-1,j+1}(t_0) \quad \text{für} \begin{cases} k &= 1,\ldots,n \\ i &= 0,\ldots,n-k \end{cases} \tag{1.27}$$

Die Gleichung wird als **de Casteljau- Algorithmus** bezeichnet.

MATLAB-Funktion: deCasteljau1.m

```
function C = deCasteljau1(P,t)
% Compute point on a bezier curve using Casteljau
for k=1:size(P,2)-1
  for i=1:size(P,2)-k
    P(:,i) = (1-t)*P(:,i) + t*P(:,i+1);
  end
end
C = P(:,1);
```

MATLAB-Beispiel:

Die folgenden Zeilen liefern das Kontrollpolygon und die Bézierkurve zu den Punkten $P_0 = (0,0)$, $P_0 = (1,-1)$, $P_0 = (3,4)$ und $P_3 = (3,3)$.

```
>> P=[0, 1, 2, 3;
>>    0 -1, 4, 3];
>> t = linspace(0,1,100);
>> for k=1:length(t)
>>   C(:,k)= deCasteljau1(P,t(k));
>> end
>> plot(C(1,:),C(2,:),'k-',
        P(1,:),P(2,:),'r*:');
```

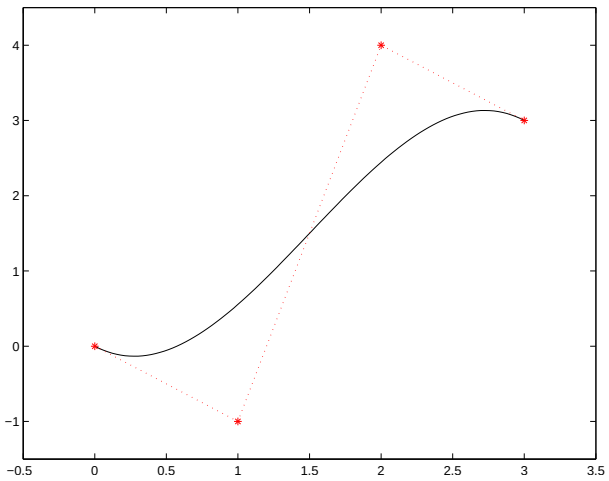


Abb. 1.5: Kontrollpolygon und Bézierkurve, Ergebnis des Matlab-Beispiels.

Kommen wir nochmals auf den Vergleich der Darstellungen zurück, d.h. die Koordinatenfunktionen $x(t)$, $y(t)$ und $z(t)$ sind Polynome in monomialer Basis oder in der Bernstein-Basis. Betrachten wir diesbezüglich einige Beispiele.

Beispiel 1.4.6 Es sei $n = 1$. Aus (1.25) erhalten wir $B_0^1(t) = 1 - t$ und $B_1^1(t) = t$. Die Darstellung (1.26) nimmt dann die Form $\mathbf{C}(t) = (1 - t)\mathbf{P}_0 + t\mathbf{P}_1$ an. Dies ist eine gerade Linie von $\mathbf{P}_0$ nach $\mathbf{P}_1$.

Beispiel 1.4.7 Es sei $n = 2$. Aus (1.25) und (1.26) erhalten wir $\mathbf{C}(t) = (1 - t)^2\mathbf{P}_0 + 2t(1 - t)\mathbf{P}_1 + t^2\mathbf{P}_2$. Dies ist eine parabolische Kurve von $\mathbf{P}_0$ nach $\mathbf{P}_2$ (siehe Abb. 1.6 rechts). Man beachte, dass der Polygonzug mit den Punkten $\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2$, sprich das Kontrollpolygon, die Form der Kurve näherungsweise gut approximiert. Für die Endpunkte gilt $\mathbf{P}_0 = \mathbf{C}(0)$ und $\mathbf{P}_2 = \mathbf{C}(1)$. Die tangentialen Richtungen an den Endpunkten sind parallel zu $\mathbf{P}_1 - \mathbf{P}_0$ und $\mathbf{P}_2 - \mathbf{P}_1$. Die Kurve liegt im Dreieck mit den Ecken $\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2$.

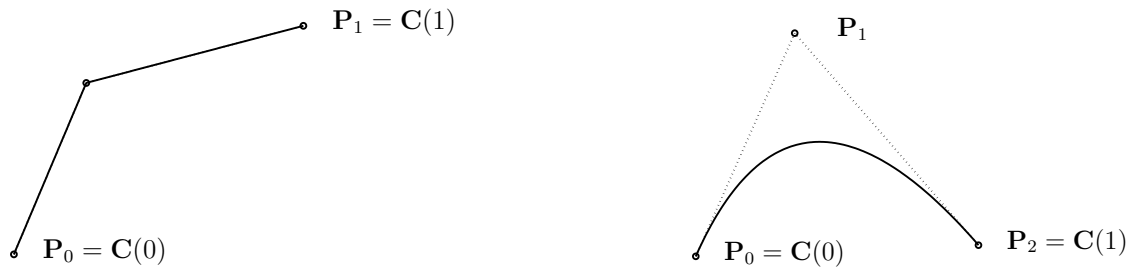


Abb. 1.6: Eine lineare (links) und quadratische (rechts) Bézierkurve.

Beispiel 1.4.8 Es sei $n = 3$. Wir erhalten $\mathbf{C}(t) = (1-t)^3\mathbf{P}_0+3t(1-t)^2\mathbf{P}_1+3t^2(1-t)\mathbf{P}_2+t^3\mathbf{P}_3$. Beispiele kubischer Bézierkurven sind in Abb. 1.7 dargestellt. Man beachte, dass das Kontrollpolygon näherungsweise die Kurve beschreibt. Für die Endpunkte gilt $\mathbf{P}_0 = \mathbf{C}(0)$ und $\mathbf{P}_3 = \mathbf{C}(1)$. Die tangentialen Richtungen an den Endpunkten sind parallel zu $\mathbf{P}_1 - \mathbf{P}_0$ und $\mathbf{P}_3 - \mathbf{P}_2$. Die Kurve liegt in der konvexen Hülle der Punkte $\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3$. Keine Gerade schneidet die Kurve häufiger als sie das Kontrollpolygon schneidet. Die Kurve krümmt sich bei $t = 0$ in die gleiche Richtung wie $\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2$ bzw. bei $t = 1$ wie $\mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3$.

Die Kurve $\mathbf{C}(t)$ ist eine vektorwertige funktion in einer Variablen. Eine Fläche ist eine vektorwertige Funktion in zwei Parametern s und t und stellt die Abbildung eines Gebiets R von der st-Ebene in den 3-dimensionalen Raum dar, nämlich $\mathbf{S}(s,t) = (x(s,t), y(s,t), z(s,t)))$, $(s,t) \in R$. Es gibt mehrere Möglichkeiten die Koordinatenfunktionen zu definieren. Der sicherlich einfachste und häufig verwendete Ansatz, ist der des Tensorprodukts, d.h.

$$\mathbf{S}(s,t) = (x(s,t), y(s,t), z(s,t))) = \sum_{i=0}^{m}\sum_{j=0}^{n} f_i(s)g_j(t)\mathbf{b}_{ij}$$

mit

$$\mathbf{b}_{ij} = (x_{ij}, y_{ij}, z_{ij}) \quad 0 \le s,t \le 1\,.$$

Man beachte, dass die Definitionsmenge dieser Abbildung das Quadrat $[0,1]^2$ ist. Verwendet man als Basisfunktionen wieder die Bernstein-Polynome so erhält man

$$\mathbf{S}(s,t) = \sum_{i=0}^{m}\sum_{j=0}^{n} B_i^m(s)B_j^n(t)\mathbf{P}_{ij} \quad 0 \le s,t \le 1\,. \tag{1.28}$$

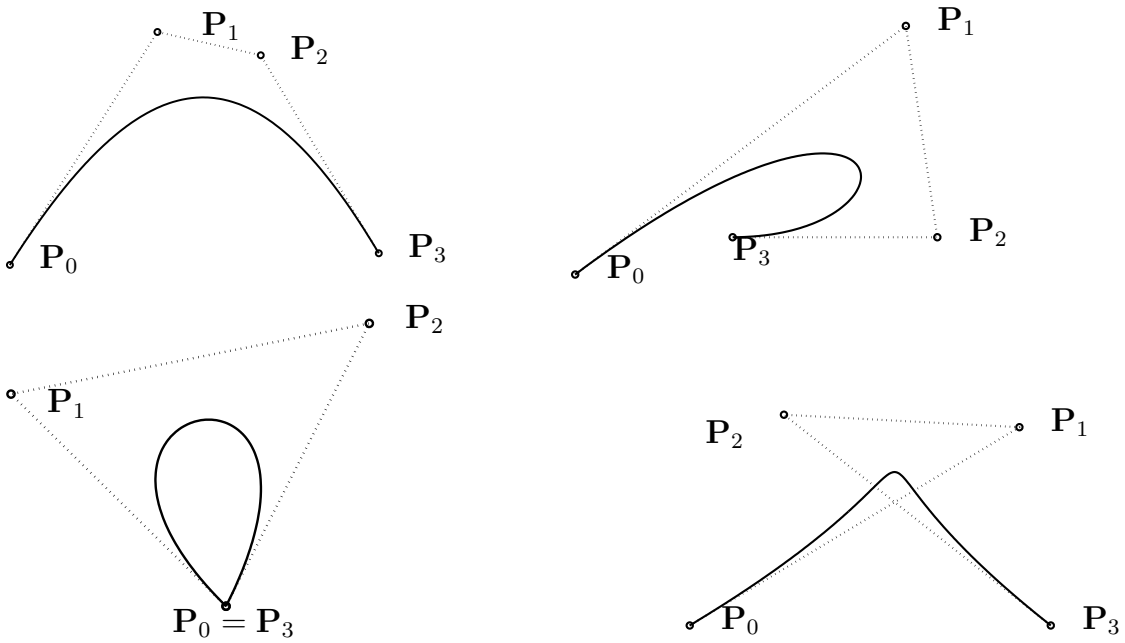


Abb. 1.7: Kubische Bézierkurve und zugehörige Kontrollpolygone.

Für ein festes s_0 gilt

$$\begin{aligned} \mathbf{C}_{s_0}(t) = \mathbf{S}(s_0,t) &= \sum_{i=0}^{m}\sum_{j=0}^{n} B_i^m(s_0)B_j^n(t)\mathbf{P}_{ij} \\ &= \sum_{j=0}^{n} B_j^n(t)\left(\sum_{i=0}^{m} B_i^m(s_0)\mathbf{P}_{ij}\right) \\ &= \sum_{j=0}^{n} B_j^n(t)\mathbf{Q}_j(s_0) \end{aligned} \tag{1.29}$$

wobei $\mathbf{Q}_j(s_0) = \sum_{i=0}^{m} B_i^m(s_0)\mathbf{P}_{ij}$, $j = 0,\ldots,n$ eine Bézierkurve ist, die auf der Fläche liegt. Mittels (1.29) kann man also (1.28) zu gegebenen (s_0,t_0) durch mehrmaliges anwenden des eindimensionalen deCasteljau-Algorithmus bestimmen. Dieses Vorgehen ist in der Routine `deCasteljau2.m` realisiert.

MATLAB-Funktion: deCasteljau2.m

```
function S = deCasteljau2(P,s,t)
% Compute a point on a Bezier surface
n = size(P,2); m = size(P,3);
if n <= m
  for j = 1:m
    Q(:,j) = deCasteljau1(squeeze(P(:,:,j)),s);
  end
  S=deCasteljau1(Q,t);
else
  for i = 1:n
    Q(:,i)=deCasteljau1(squeeze(P(:,i,:)),t);
```

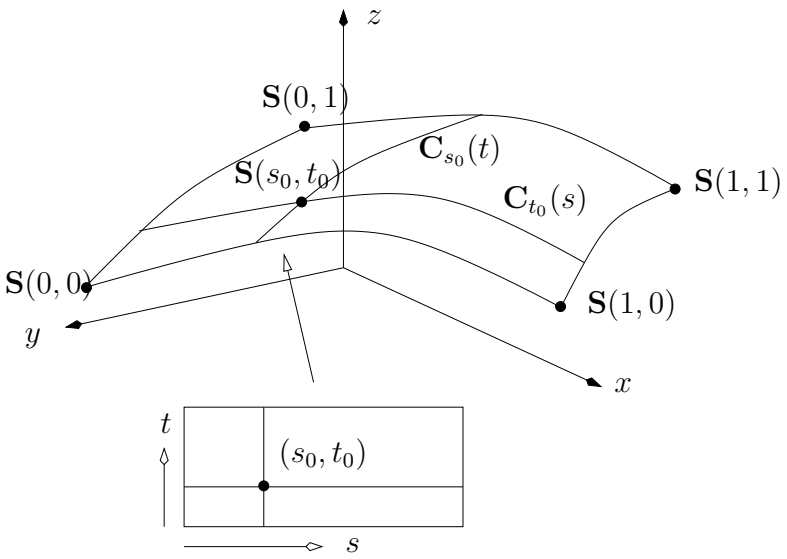
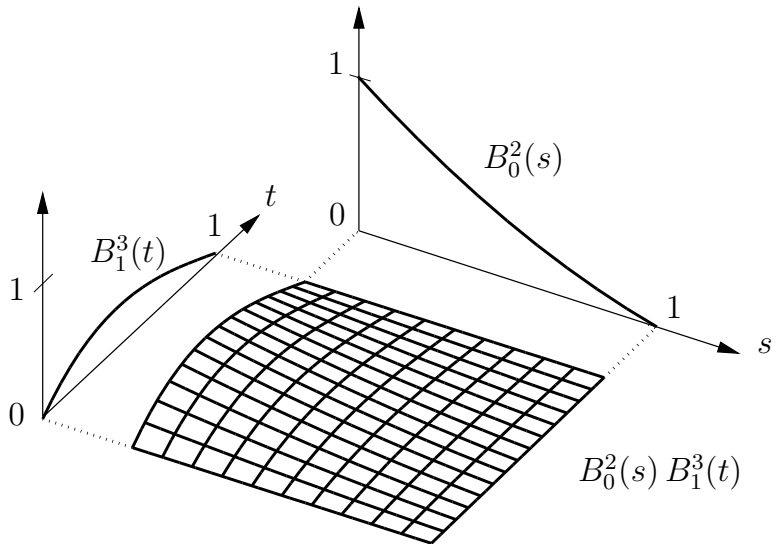


Abb. 1.8: Eine Tensorproduktfläche und isoparametrische Kurven.



```
12    end
13    S=deCasteljau1(Q,s);
14  end
```

In Abb. 1.9 ist das Kontrollnetz und die Bézierfläche beispielhaft dargestellt.

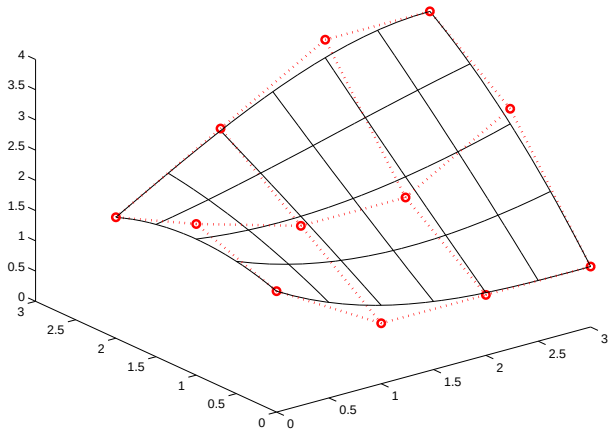


Abb. 1.9: Beispiel eines Kontrollnetz und Bézierfläche in $\mathbb{R}^2$.

1.5 B-Splines

In Kapitel 1.1 haben wir uns mit den kubischen Splines beschäftigt und diese mit einem direkten Ansatz der Form

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

hergeleitet. Bis auf eine Verschiebung um x_i ist dies ein monomialer Ansatz. Im Folgenden wollen wir uns mit einem allgemeineren[1] Ansatz beschäftigen, der durch die Anwendung in der Computergrafik motiviert ist. Wir werden eine weitere Basis einführen, deren Basisfunktionen einen Träger minimaler Länge haben (Monome haben ganz $\mathbb{R}$ als Träger) und deren Elemente sich effektiv und numerisch stabil berechnen lassen. Wir erinnern daran, dass mit dem **Träger** einer Funktion $f : \mathbb{R} \to \mathbb{R}$ die Menge $\text{supp}(f) := \overline{\{x \in \mathbb{R}, f(x) \neq 0\}}$ bezeichnet wird, wobei der Strich den Abschluß der Menge bezeichnet. Splines, die in dieser Basis dargestellt werden, nemmt man **B-Splines**.

Rekursive Definition der B-Splines-Basisfunktionen

Definition 1.5.1 (B-Splines-Basisfunktionen) *Sei $\mathcal{T} = \{t_0, t_m\}$ eine nichtfallende Knotenfolge reeller Zahlen, d.h. $t_i \leq t_{i+1}$ $(i = 0, \ldots, m-1)$. Die t_i werden als* **Knoten** *und $\mathcal{T}$ als* **Knotenvektor** *bezeichnet. Die i-te B-Spline Basisfunktion vom Grade p (Ordnung $p+1$) ist definiert für $p = 0$ als stückweise konstante Funktion der Form*

$$N_j^0(t) := \begin{cases} 1 & , \quad \text{falls } t_j \leq t < t_{j+1} \\ 0 & , \quad \text{sonst} \end{cases} \tag{1.30}$$

[1] Allgemeiner bzgl. des Polynomgrads auf jedem Teilintervall, als auch der Glattheitsanforderung an den Knoten zwischen den Teilintervallen, bzw. an den beiden Endpunkten.

und für $p > 0$ durch

$$N_j^p(t) := \frac{t - t_i}{t_{i+p} - t_i} N_i^{p-1}(t) + \frac{t_{i+p+1} - t}{t_{i+p+1} - t_{i+1}} N_i^{p-1}(t)\,. \tag{1.31}$$

Bemerkung 1.5.2 *(i) N_i^0 ist eine Treppenfunktion, die auf dem halboffenen Intervall $[t_j, t_{j+1})$ Eins ist und sonst verschwindet.*

(ii) Man beachte die rechtsseitige Stetigkeit in der Definition der N_j^0, d.h. $\lim_{t \to t_{j+1}^+} N_j^0(t) = N_{j+1}^0(t_{j+1})$.

(iii) Für $p > 0$ ist $N_i^p(t)$ eine Linearkombination von zwei Basisfunktionen vom Grade $(p-1)$.

(iv) Die Berechnung einer Menge von Basisfunktionen erfofert einen Knotenvektor $\mathcal{T}$ und einen Grad p.

(v) In (1.31) kann der Nenner im Bruch Null werden; dieser Quotient sei per Definition Null.

(vi) Die $N_i^p(t)$ sind stückweise polynomiale Funktionen auf der reellen Achse. Normalerweise ist nur das Intervall $[t_0, t_m]$ von Interesse.

(vii) Das i-te Knotenintervall $[t_i, t_{i+1})$ kann die Länge Null haben, da aufeinanderfolgende Knoten nicht verschieden sein müssen.

(viii) Die Berechnung der Basisfunktionen kann in dem bekannten Dreiecksschema erfolgen.

(ix) Die N_j^0 liefern auf dem Intervall $[t_0, t_m)$ eine Zerlegung der Eins und sind positiv.

Die rekursive Definition der B-Splines (1.31) liefert einen einfachen Algorihmus, um Werte der B-Splines zu einem gegebenen $t \in [t_j, t_{j+1}$ zu bestimmen.

Beispiel 1.5.3 Es sei $\mathcal{T} = \{t_0 = 0, t_1 = 0, t_2 = 0, t_3 = 1, t_4 = 1, t_5 = 1\}$ und $p = 2$. Für B-Spline-Basisfunktionen vom Grade 0, 1 und 2 lauten dann

$$\begin{aligned}
N_0^0 &= N_1^0 = 0 \quad -\infty < t < \infty \\
N_2^0 &= \begin{cases} 1 & 0 \le t < 1 \\ 0 & \text{sonst} \end{cases} \\
N_3^0 &= N_4^0 = 0 \quad -\infty < t < \infty
\end{aligned}$$

$$\begin{aligned}
N_0^1 &= \frac{t-0}{0-0} N_0^0 + \frac{0-t}{0-0} N_1^0 = 0 && -\infty < t < \infty \\
N_1^1 &= \frac{t-0}{0-0} N_1^0 + \frac{1-t}{1-0} N_2^0 = \begin{cases} 1-t & 0 \le t < 1 \\ 0 & \text{sonst} \end{cases} \\
N_2^1 &= \frac{t-0}{1-0} N_2^0 + \frac{1-t}{1-1} N_3^0 = \begin{cases} t & 0 \le t < 1 \\ 0 & \text{sonst} \end{cases} \\
N_3^1 &= \frac{t-1}{1-1} N_3^0 + \frac{1-t}{1-1} N_4^0 = 0 && -\infty < t < \infty
\end{aligned}$$

$$\begin{aligned}
N_0^2 &= \frac{t-0}{0-0} N_0^1 + \frac{1-t}{1-0} N_1^1 = \begin{cases} (1-t)^2 & 0 \le t < 1 \\ 0 & \text{sonst} \end{cases} \\
N_1^2 &= \frac{t-0}{1-0} N_1^1 + \frac{1-t}{1-0} N_2^1 = \begin{cases} 2t(1-t) & 0 \le t < 1 \\ 0 & \text{sonst} \end{cases} \\
N_2^2 &= \frac{t-0}{1-0} N_2^1 + \frac{1-t}{1-1} N_3^1 = \begin{cases} t^2 & 0 \le t < 1 \\ 0 & \text{sonst} \end{cases}
\end{aligned}$$

Man beachte, dass N_i^2 auf das Intervall $[0,1)$ restringiert gerade die quadratischen Bernstein-Polynome sind. Aus diesem Grunde ist die B-Spline-Darstellung mit einem Knotenvektor der Form

$$U = \{ \underbrace{0,\ldots,0}_{(p+1)-mal} , \underbrace{1,\ldots,1}_{(p+1)-mal} \}$$

eine Verallgemeinerung der Bézier-Darstellung ist.

Beispiel 1.5.4 Es sei $\mathcal{T} = \{t_0 = t_1 = t_2 = 0, t_3 = 1, t_4 = 2, t_5 = 3, t_6 = t_7 = 4, t_8 = t_9 = t_{10} = 5\}$ und $p = 2$. Für stückweise konstanten, linearen und quadratischen B-Spline-Basisfunktionen lauten dann:

$$\begin{aligned}
N_0^0 &= N_1^0 = 0 \quad -\infty < t < \infty \\
N_2^0 &= \begin{cases} 1 & 0 \le t < 1 \\ 0 & \text{sonst} \end{cases} \\
N_3^0 &= \begin{cases} 1 & 1 \le t < 2 \\ 0 & \text{sonst} \end{cases} \\
N_4^0 &= \begin{cases} 1 & 2 \le t < 3 \\ 0 & \text{sonst} \end{cases} \\
N_5^0 &= \begin{cases} 1 & 3 \le t < 4 \\ 0 & \text{sonst} \end{cases} \\
N_6^0 &= 0 \quad -\infty < t < \infty \\
N_7^0 &= \begin{cases} 1 & 4 \le t < 5 \\ 0 & \text{sonst} \end{cases} \\
N_8^0 &= N_9^0 = 0 \quad -\infty < t < \infty
\end{aligned}$$

$$\begin{aligned}
N_0^1 &= \frac{t-0}{0-0}N_0^0 + \frac{0-t}{0-0}N_1^0 = 0 && -\infty < t < \infty \\
N_1^1 &= \frac{t-0}{0-0}N_1^0 + \frac{1-t}{1-0}N_2^0 = \begin{cases} 1-t & 0 \le t < 1 \\ 0 & \text{sonst} \end{cases} \\
N_2^1 &= \frac{t-0}{1-0}N_2^0 + \frac{2-t}{2-1}N_3^0 = \begin{cases} t & 0 \le t < 1 \\ 2-t & 1 \le t < 2 \\ 0 & \text{sonst} \end{cases} \\
N_3^1 &= \frac{t-1}{2-1}N_3^0 + \frac{3-t}{3-2}N_4^0 = \begin{cases} t-1 & 1 \le t < 2 \\ 3-t & 2 \le t < 3 \\ 0 & \text{sonst} \end{cases} \\
N_4^1 &= \frac{t-2}{3-2}N_4^0 + \frac{4-t}{4-3}N_5^0 = \begin{cases} t-2 & 2 \le t < 3 \\ 4-t & 3 \le t < 4 \\ 0 & \text{sonst} \end{cases} \\
N_5^1 &= \frac{t-3}{4-3}N_5^0 + \frac{4-t}{4-4}N_6^0 = \begin{cases} t-3 & 3 \le t < 4 \\ 0 & \text{sonst} \end{cases} \\
N_6^1 &= \frac{t-4}{4-4}N_6^0 + \frac{5-t}{5-4}N_7^0 = \begin{cases} 5-t & 4 \le t < 5 \\ 0 & \text{sonst} \end{cases} \\
N_7^1 &= \frac{t-4}{5-4}N_7^0 + \frac{5-t}{5-5}N_8^0 = \begin{cases} t-4 & 4 \le t < 5 \\ 0 & \text{sonst} \end{cases} \\
N_8^1 &= \frac{t-5}{5-5}N_8^0 + \frac{5-t}{5-5}N_9^0 = 0 && -\infty < t < \infty
\end{aligned}$$

Die folgenden N_i^2 sind bis auf die angegebenen Intervalle jeweils Null.

$$
\begin{aligned}
N_0^2 &= \frac{t-0}{0-0}N_0^1 + \frac{1-t}{1-0}N_1^1 &&= (1-t)^2 && 0 \le t < 1\\
N_1^2 &= \frac{t-0}{1-0}N_1^1 + \frac{2-t}{2-0}N_2^1 &&= \begin{cases} 2t - \frac{3}{2}t^2 & 0 \le t < 1 \\ \frac{1}{2}(2-t)^2 & 1 \le t < 2 \end{cases}\\
N_2^2 &= \frac{t-0}{2-0}N_2^1 + \frac{3-t}{3-1}N_3^1 &&= \begin{cases} \frac{1}{2}t^2 & 0 \le t < 1 \\ -\frac{3}{2} + 3t - t^2 & 1 \le t < 2 \\ \frac{1}{2}(3-t)^2 & 2 \le t < 3 \end{cases}\\
N_3^2 &= \frac{t-1}{3-1}N_3^1 + \frac{4-t}{4-2}N_4^1 &&= \begin{cases} \frac{1}{2}(t-1)^2 & 1 \le t < 2 \\ -\frac{11}{2} + 5t - t^2 & 2 \le t < 3 \\ \frac{1}{2}(4-t)^2 & 3 \le t < 4 \end{cases}\\
N_4^2 &= \frac{t-2}{4-2}N_4^1 + \frac{4-t}{4-3}N_5^1 &&= \begin{cases} \frac{1}{2}(t-2)^2 & 2 \le t < 3 \\ -16 + 10t - \frac{3}{2}t^2 & 3 \le t < 4 \end{cases}\\
N_5^2 &= \frac{t-3}{4-3}N_5^1 + \frac{5-t}{5-4}N_6^1 &&= \begin{cases} (t-3)^2 & 3 \le t < 4 \\ (5-t)^2 & 4 \le t < 5 \end{cases}\\
N_6^1 &= \frac{t-4}{5-4}N_6^1 + \frac{5-t}{5-4}N_7^1 &&= 2(t-4)(5-t) && 4 \le t < 5\\
N_7^1 &= \frac{t-4}{5-4}N_7^1 + \frac{5-t}{5-5}N_8^1 &&= (t-4)^2 && 4 \le t < 5
\end{aligned}
$$

In Abbildung 1.10 ist die Zusammensetzung von N_3^2 aus den jeweiligen stückweise polynomialen Funktionen auf den einzelnen Teilintervallen grafisch dargestellt.

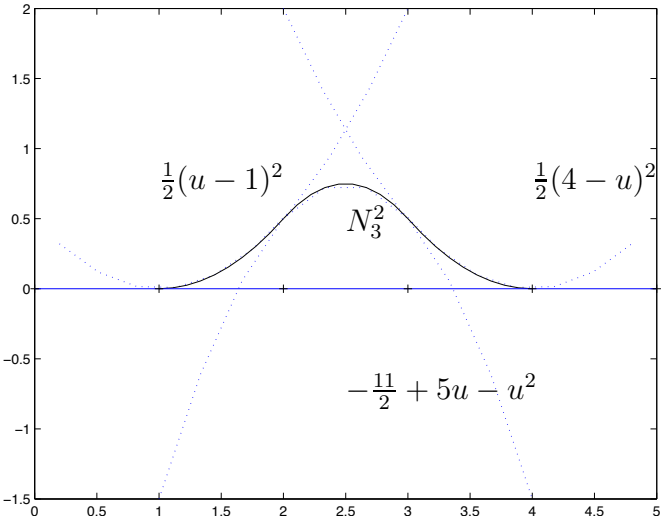


Abb. 1.10: Die Zerlegung von N_3^2 in seine stückweise polynomialen Teilfunktionen.

Es ist wichtig, den Effekt von mehrfachen Knoten zu verstehen. Man betrachte die Funktionen N_0^2, N_1^2, N_2^2, N_5^2 und N_5^2 in Abbildung 1.12. Beachtet man die rekursive Definition der Basis-

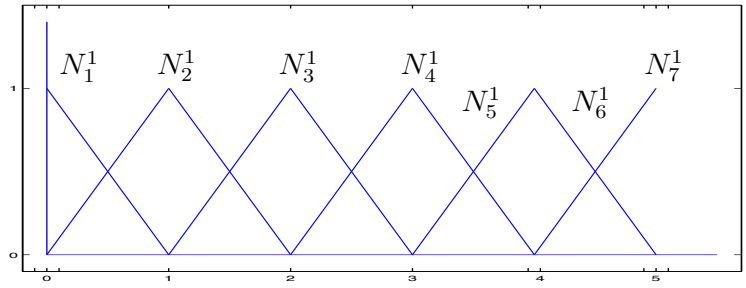


Abb. 1.11: Die stückweise linearen Basisfunktionen zu $\mathcal{T} = \{0, 0, 0, 1, 2, 3, 4, 4, 5, 5, 5\}$.

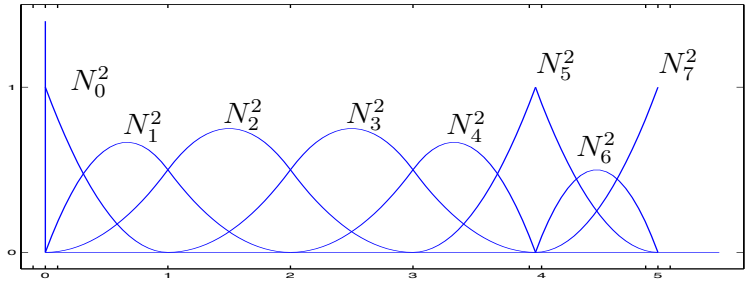


Abb. 1.12: Die stückweise quadratischen Basisfunktionen zu $\mathcal{T} = \{0, 0, 0, 1, 2, 3, 4, 4, 5, 5, 5\}$.

funktionen (1.31), so stellt man fest, dass sie jeweils nur von 4 Knoten abhängen, nämlich:

$$\begin{aligned} N_0^2 &: \{0,0,0,1\} \\ N_1^2 &: \{0,0,1,2\} \\ N_2^2 &: \{0,1,2,3\} \\ N_5^2 &: \{3,4,4,5\} \\ N_6^2 &: \{4,4,5,5\} \end{aligned}$$

Der Begriff Vielfachheit eines Knotens, kann man verstehen

- in Bezug auf eine Knoten im Knotenvektor oder
- in Bezug auf einen Knoten beüglich einer Basisfunktion.

Zum Beispiel hat $t = 0$ die Vielfachheit 3 im o.g. Knotenvektor $\mathcal{T}$, aber in Bezug auf die Basisfunktion N_1^2 ist $t = 0$ ein Knoten mit der Vielfachheit 2. Die Basisfunktionen sind stückweise polynomiale Funktionen, d.h. im Inneren der Intervallen (t_j, t_{j+1}) sind sie beliebig glatt. Unstetigkeiten können also nur an den Knoten auftreten. Für $t = 0$ stellt man fest, dass N_0^2 unstetig ist, N_1^2 C^0-stetig ist, N_2^2 C^1-stetig ist und N_5^2 und all seine Ableitungen dort Null von beiden Seiten ist. N_1^2 sieht $t = 0$ als doppelten Knoten, N_2^2 sieht $t = 0$ als einfachen Knoten und N_5^2 enthällt $t = 0$ gar nicht als Knoten.

Satz 1.5.5 *Ist t_ℓ ein m-facher Knoten, d.h.*

$$t_{\ell-1} < t_\ell = \cdots = t_{\ell+m-1} < t_{\ell+m},$$

so ist N_j^k an der Stelle t_ℓ mindestens $(k-1-m)$-mal stetig differenzierbar. Für die Ableitung von N_j^k gilt

$$\frac{d}{dt} N_j^k(t) = (k-1) \cdot \left(\frac{N_j^{k-1}(t)}{t_{j+k-1} - x_j} - \frac{N_{j+k}^{k-1}(t)}{t_{j+k} - t_{j+1}} \right)$$

Beweis. Der Beweis dieses Satzes wird an späterer Stelle erbracht □

Satz 1.5.6 (Marsdens[2] Identiät) *Bei gegebener Knotenfolge $\mathcal{T} := \{t_j\}, j \in \mathbb{Z}$, mit*

$$\lim_{j \to \pm\infty} t_j = \pm\infty$$

sei für beliebiges $y \in \mathbb{R}$

$$\psi_{j1}(y) := 1, \quad \psi_{jk}(y) := (x_{j+1} - y)(x_{j+2} - y) \cdot \cdots \cdot (x_{j+k-1} - y), \quad k > 1.$$

Dann gilt

$$(x - y)^{k-1} = \sum_{j \in \mathbb{Z}} \psi_{jk}(y) B_{jk}(x)$$

Beweis. Sei $(a_j)_{j \in \mathbb{Z}}$ eine beliebige Folge reeller Zahlen. Dann folgt aus der Rekursion (1.31)

$$\sum_{j \in \mathbb{Z}} a_j B_{jk} = \sum_{j \in \mathbb{Z}} (a_{j-1}(1 - \omega_{jk} + a_j \omega_{jk}) B_{j,k-1}$$

Setzen wir $a_j := \psi_{jk}(x)$, so folgt

$$\begin{aligned} a_{j-1}(1 - \omega_{jk}(x)) + a_k \omega_{jk}(x) &= \Big((x_j - y)(1 - \omega_{jk}(x)) + (x_{j+k-1} - y)\omega_{jk}(x)\Big)\psi_{j,k-1}(y) \\ &= (x - y)\psi_{j,k-1}(y). \end{aligned}$$

Damit gewinnen wir die Gleichung

$$\sum_{j \in \mathbb{Z}} \psi_{jk}(y) B_{jk}(x) = (x - y) \sum_{j \in \mathbb{Z}} \psi_{j,k-1}(y) B_{j,k-1}(x)$$

und somit mithilfe der Definitionen der ψ_{jk} und B_{j1}

$$\sum_{j \in \mathbb{Z}} \psi_{jk} B_{jk}(x) = (x - y)^{k-1} \sum_{j \in \mathbb{Z}} \psi_{j,1}(y) B_{j,1}(x) = (x - y)^{k-1}$$

□

Bemerkung 1.5.7 *Da y im obigen Satz beliebig war, haben wir also gezeigt, dass $\mathbb{P}_{k-1} \subset \mathcal{S}_{k,t}$.*

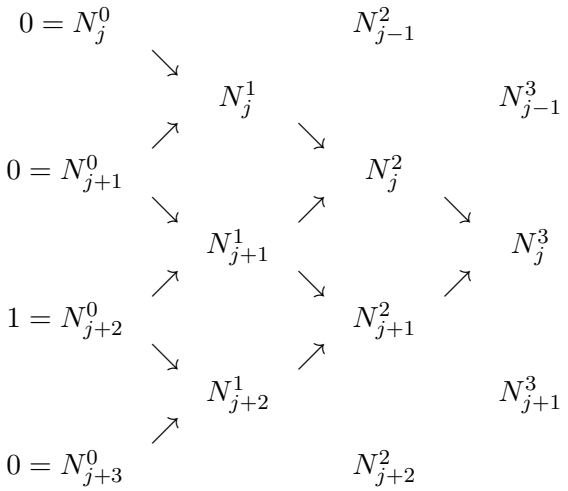
Effiziente Auswertung der B-Spline-Basisfunktionen

Die Funktion N_j^3 ist eine Linearkombination der Funktionen N_j^0, N_{j+1}^0, N_{j+2}^0 und N_{j+1}^0. Somit ist N_j^3 nur von Null verschieden für $t \in [t_j, t_{j+4})$.
In jedem Knotenintervall $[t_j, t_{j+1})$ sind maximal $p + 1$ der N_i^p von Null verschieden, nämlich $N_{j-p}^p, \ldots, N_j^p$. Auf $[t_3, t_4)$ ist z.B. N_3^0 die einzige nichtverschwindende Basisfunktion vom Grad Null. Somit sind $N_0^3, \ldots, N_3^3$ die einzigen von Null verschiedenen kubischen Funktionen auf $[t_3, t_4)$. Diese Eigenschaft ist in Tabelle 1.4 dargestellt.

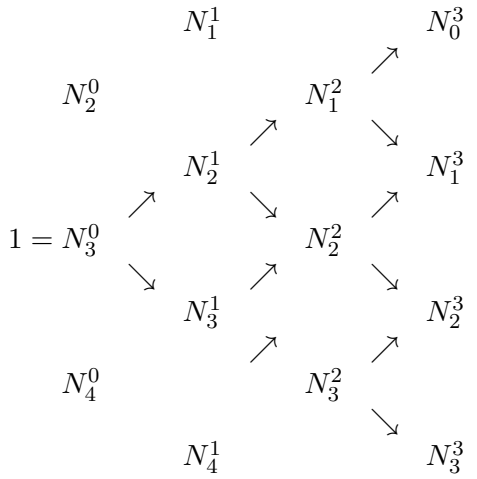
MATLAB-Funktion: BasisFunc.m

```
1  function N = BasisFunc(i,p,U,t)
2  % compute the nonvanishing basis functions
3  N = zeros(p+1,1);
4  N(1) = 1;
5  for j=1:p
6    left(j) = t - U(i+1-j);
7    right(j) = U(i+j) - t;
8    saved = 0;
```

[5] benannt nach M. J. Marsden, vlg. hierzu [Marsden]



Tab. 1.3: N_j^3 ist nur auf dem Intervall $[t_j, t_{j+4})$ von Null verschieden.



Tab. 1.4: N_3^0 ist nur auf dem Intervall $[t_3, t_4)$ von Null veschieden, Somit sind auch $N_0^3, \ldots, N_3^3$ die einzigen von Null verschiedenen kubischen Funktionen auf $[t_3, t_4)$.

```
9     for r=1:j
10      temp = N(r)/(right(r)+left(j-r+1));
11      N(r) = saved + right(r) * temp;
12      saved = left(j-r+1) * temp;
13    end
14    N(j+1) = saved;
15  end
```

MATLAB-Funktion: FindSpan.m

```
1   function mid = FindSpan(p,U,t)
2   % returns the knot span index
3   n = length(U)-p;
4   if t==U(end)           % special case
5     mid = n-1;
6     return
7   end
8   low = p;
9   high = n;
10  mid = floor((low+high)/2);
11  while t<U(mid) | t>= U(mid+1)
12    if t<U(mid)
13      high = mid;
14    else
15      low = mid;
16    end
17    mid = floor((low+high)/2);
18  end
```

MATLAB-Funktion: CurvePoint.m

```
1   function value = CurvePoint(p,U,P,t)
2   % compute point on B-spline curve
3   span = FindSpan(p,U,t);
4   B = BasisFunc(span,p,U,t);
5   value = 0*P(:,1);
6   for i=0:p
7     value = value + B(i+1) * P(:,span-p+i);
8   end
```

Ableitung der B-Splines

MATLAB-Funktion: AllBasisFunc.m

```
function N = AllBasisFunc(i,p,U,t)
% compute the nonvanishing basis functions
N = zeros(p+1,p+1);
N(1,1) = 1;
for j=1:p
  left(j) = t - U(i+1-j);
  right(j) = U(i+j) - t;
  saved = 0;
  for r=1:j
    temp = N(r,j)/(right(r)+left(j-r+1));
    N(r,j+1) = saved + right(r) * temp;
    saved = left(j-r+1) * temp;
  end
  N(j+1,j+1) = saved;
end
```

MATLAB-Funktion: CurveDerivPts.m

```
function PK = CurveDerivPts(p,U,P,d,r1,r2)
% compute control points of curve derivatives
r = r2-r1;
for i=1:r+1
  PK(:,1,i)=P(:,r1+i);
end
for k=2:d+1
  tmp = p-k+2;
  for i=1:r-k+2
    PK(:,k,i)=tmp*(PK(:,k-1,i+1)-PK(:,k-1,i))/(U(r1+i+p+1)-U(r1+i+k
        -1));
  end
end
```

MATLAB-Funktion: CurveDerivs.m

```
function CK = CurveDerivs(p,U,P,t,d)
% Compute curve derivatives
du = min(d,p);
CK(1:size(P,1),[p+2:d+1]) = 0;
span = FindSpan(p,U,t);
N = AllBasisFunc(span,p,U,t);

```

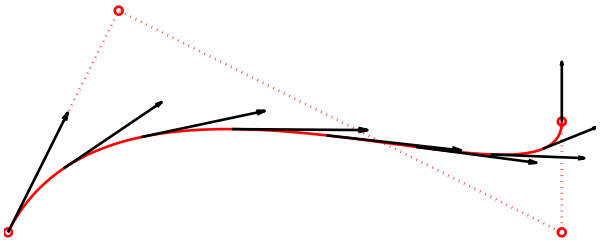


Abb. 1.13: Ergebnis der Matlab-Zeilen.

```
8  PK = CurveDerivPts(p,U,P,du,span-p-1,span-1);
9
10 for k=1:du+1
11  CK(:,k) = 0;
12  for j=1:p-k+2
13    CK(:,k) = CK(:,k) + N(j,p-k+2)*PK(:,k,j);
14  end
15 end
```

MATLAB-Beispiel:

Die folgenden Zeilen stellen die Bernstein-Polynome $B_0^8, \ldots, B_8^8$ graphisch dar.

```
>> P =[0,1,5,5;
       0,2,0,1];
>> U = [0,0,0,0,1,1,1,1];  p=3;
>> s = linspace(U(1),U(end),201);
>> for k = 1:length(s)
     C(:,k) = CurvePoint(p,U,P,s(k));
   end
>> plot(C(1,:),C(2,:),'-', ...
        P(1,:),P(2,:),'o:');
>> hold on
>> for j = 1 :25:length(s)
     V = CurveDerivs(p,U,P,s(j),1);
     quiver(V(1,1),V(2,1), ...
            0.2*V(1,2),0.2*V(2,2))
   end
```

1.6 Rationale B-Splines

MATLAB-Funktion: RatCurvePoint.m

```
1 function value = RatCurvePoint(p,U,Pw,t)
2 % compute point on B-spline curve
```

```
span = FindSpan(p,U,t);
B = BasisFunc(span,p,U,t);
value = 0*Pw(:,1);
for i=0:p
  value = value + B(i+1) * Pw(:,span-p+i);
end
value = value(1:end-1)/value(end);
```

MATLAB-Funktion: RatCurveDerivs.m

```
function CK = RatCurveDerivs(p,U,Pw,t,d)
% compute derivatives on rational B-spline curve
du = min(d,p);

ders = CurveDerivs(p,U,Pw,t,d);
Aders = ders(1:end-1,:);
wders = ders(end,:);

CK(1:size(Aders,1),p+2:d+1) = 0;

for k=1:du+1
   v = Aders(:,k);
   for i=1:k-1
     v = v - nchoosek(k-1,i)*wders(i+1)*CK(:,k-i);
   end
   CK(:,k) = v/wders(1);
end
```

MATLAB-Beispiel:

Die folgenden Zeilen stellen die Bernstein-Polynome $B_0^8, \ldots, B_8^8$ graphisch dar.

```
>> w = [1, 1,  2];
>> P = [1, 1,  0;
        0  1,  1];
>> U = [0,0,0,1,1,1];   p=2;
>> s = linspace(0,1,201);
>> Pw = [P(1,:).*w;P(2,:).*w;w];
>> for k = 1:length(s)
     Cw(:,k) = RatCurvePoint(p,U,Pw,s(k));
   end
>> plot(Cw(1,:),Cw(2,:),'-', ...
        P(1,:),P(2,:),'*:');
>> hold on
>> for j = 1 :25:length(s)
     CK = RatCurveDerivs(p,U,Pw,s(j),1);
      quiver(CK(1,1),CK(2,1),0.3*CK(1,2)
          ,0.3*CK(2,2))
   end
```

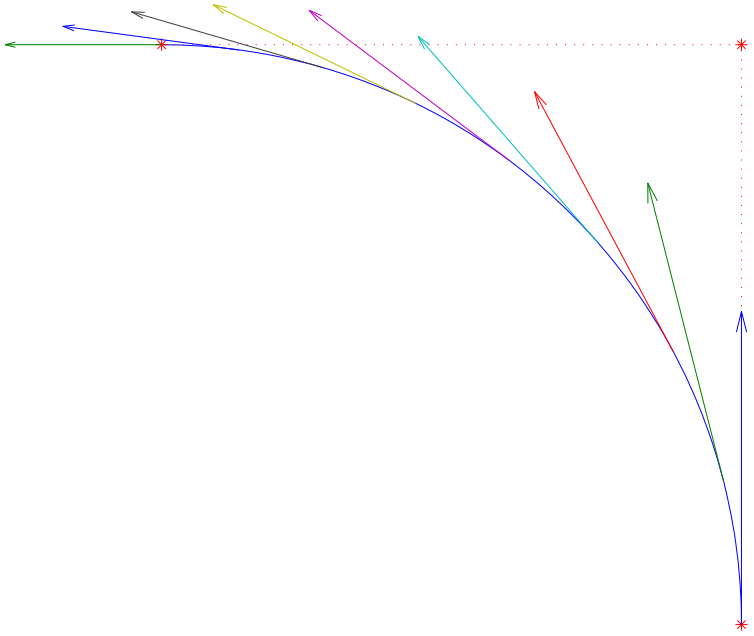


Abb. 1.14: Ergebnis der Matlab-Zeilen.

1.7 Grundlegende Algorithmen

MATLAB-Funktion: CurveKnotIns.m

```
function [U,Q] = CurveKnotIns(p,U,P,t,k,s,r)
if p < s+r, Q=P; return, end
% compute new curve from knot insertion
np = length(U)-p-1;
% unaltered control points
Q = P(:,1:k-p);
Q(:,k-s+r:np+r)=P(:,k-s:np);
R = P(:,k-p:k-s);
for j=1:r % insert new knot r times
   L=k-p+j-1;
   for i=1:p-j-s+1
     alpha = (t-U(L+i))/(U(i+k)-U(L+i));
     R(:,i) = alpha*R(:,i+1)+(1-alpha)*R(:,i);
   end
   Q(:,L+1) = R(:,1);
   Q(:,k+r-j-s)= R(:,p-j-s+1);
end
%copy remaining control points
Q(:,L+2:k-s)= R(:,2:k-s-L);
% new knot vevtor
U = [U(1:k),t*ones(1,r),U(k+1:end)];
```

MATLAB-Funktion: CurveSplit.m

```
function [U1,P1,U2,P2] = CurveSplit(p,U,P,t)
if t==U(1)
  U1=[];P1=[];U2=U;P2=P;return
elseif t==U(end)
  U1=U;P1=P;U2=[];P2=[];return
end
k = FindSpan(p,U,t);
s = sum((t==U));
if p>s
  [U,P]=CurveKnotIns(p,U,P,t,k,s,p-s);
end
U1 = U([1:k+p-s,k+p-s])-U(1);
U1 = U1/max(U1); P1 = P(:,1:k);
U2 = U([k+1-s,k+1-s:end])-U(k+1-s);
U2 = U2/max(U2); P2 = P(:,k-s:end);
```

MATLAB-Beispiel:

Die folgenden Zeilen stellen die Bernstein-Polynome $B_0^8, \ldots, B_8^8$ graphisch dar.

```
>> P =[0,1,5,5;
       0,2,0,1];
>> U = [0,0,0,0,1,1,1,1];
>> p = 3;
>> t = 1/3;
>> [U1,P1, U2, P2] = CurveSplit(p,U,P,t)
U1 =
   0   0   0   0   1   1   1   1
P1 =
         0   0.3333   1.0000   1.7407
         0   0.6667   0.8889   0.9259
U2 =
   0   0   0   0   1   1   1   1
P2 =
   1.7407   3.2222   5.0000   5.0000
   0.9259   1.0000   0.3333   1.0000
```

MATLAB-Funktion: isLine.m

```
function [flag,P0,P1] = isLine(P,tol1,tol2)
S = mean(P,2);
PmS = P-S*ones(1,size(P,2));
[j,k] = max(sum(abs(PmS)));
rot = GivensRotMat(PmS(1,k),PmS(2,k));
Q = rot'*([PmS(1,:);PmS(2,:)]);
h = max(Q,[],2)-min(Q,[],2);
if h(2) <= max(tol1,tol2*h(1))
```

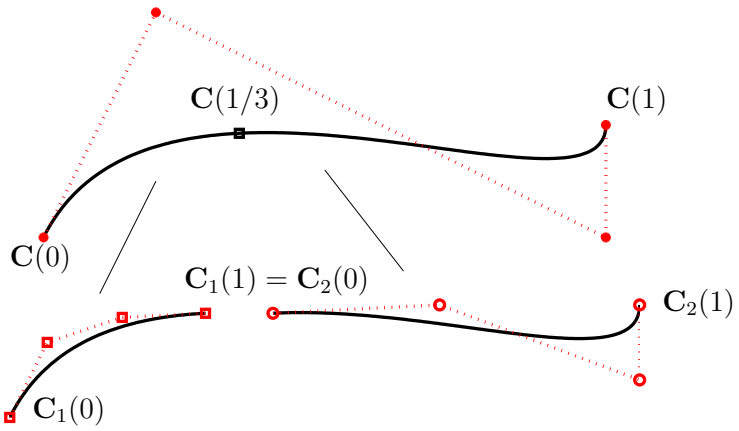


Abb. 1.15: Ergebnis der Matlab-Zeilen.

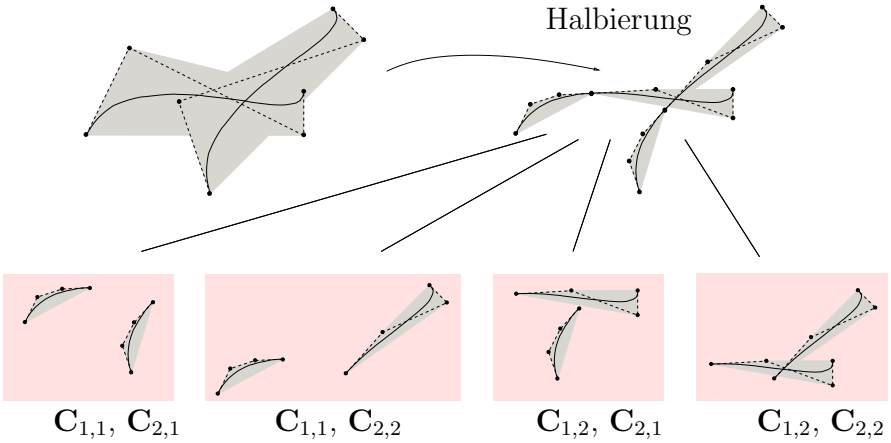


Abb. 1.16: Testen auf gemeinsamen Schnitt der einzelnen konvexen Hüllen nach Halbierung der einzelnen Bézierkurven $\mathbf{C}_1 = \mathbf{C}_{1,1} \cup \mathbf{C}_{1,2}$ und $\mathbf{C}_2 = \mathbf{C}_{2,1} \cup \mathbf{C}_{2,2}$.

```
  flag = 1;
  P0 = S + rot*[h(1);0]/2;
  P1 = S + rot*[-h(1);0]/2;
else
  flag = 0; P0=[];P1=[];
end
```

MATLAB-Funktion: LineIntersect.m

```
function [flag,s] = LineIntersect(x,y)
% check for intersection of two line segments
% 0 no intersection, 1 one or more intersection points
dx = x(:,2)-x(:,1); mx = norm(dx);
dy = y(:,2)-y(:,1); my = norm(dy);
dw = (y(:,2)+y(:,1)-x(:,2)-x(:,1)); mw = norm(dw);
flag = 1;s=[];
if abs(det([dx,dy])) >= 1e-12 *mx*my % check for non parallel
  if ~sum(abs([dx,dy]\dw)>1)
    t = [dx,dy]\dw;
    s = ((x(:,2)+x(:,1))+t(1)*dx)/2;
    return
  end
elseif det([dx,dw]) < max(1e+10*realmin,1e-12*mx*mw)  % on common
   line
  mm = [((x(:,2)+x(:,1))/2)*[1,1],((y(:,2)+y(:,1))/2)*[1,1]];
  dd = [dx,dx,dy,dy];
  xx = [dw+dy,dw-dy,-dw+dx,-dw+dx];
  for j=1:4
    t = xx(:,j)'*dd(:,j)/(dd(:,j)'*dd(:,j));
    if abs(t)<=1
      s = mm(:,j)+t/2*dd(:,j);
      return
    end
  end
end
flag = 0;
```

MATLAB-Funktion: comConvHull.m

```
function flag = comConvHull(xy,st)
if comBoundingBoxes(min(xy,[],2),max(xy,[],2), ...
                    min(st,[],2),max(st,[],2))
  flag = 1;

  if inConvHull(xy,mean(st(:,1:end-1),2)), return, end
  if inConvHull(st,mean(xy(:,1:end-1),2)), return, end

  for j = 1:size(xy,2)-1
```

```
    for k = 1:size(st,2)-1
      if LineIntersect(xy(:,j:j+1),st(:,k:k+1))
        return
      end
    end
  end
end
flag = 0;
```

MATLAB-Funktion: CurveBisection.m

```
function points = CurveBisection(p1,U1,Q1,p2,U2,Q2,tol)
[flag1,a1,b1] = isLine(Q1,tol(1),tol(2));
[flag2,a2,b2] = isLine(Q2,tol(1),tol(2));
if flag1 && flag2  % both segments are lines
  [flag,points] = LineIntersect([a1,b1],[a2,b2]);
else
  xy = myConvHull(Q1,flag1,a1,b1);
  st = myConvHull(Q2,flag2,a2,b2);
  points =[];
  if  comConvHull(xy,st) % intersection of convex hulls non empty
    [U11,Q11,U21,Q21] = CurveSplit(p1,U1,Q1,(max(U1)-min(U1))/2);
    [U12,Q12,U22,Q22] = CurveSplit(p2,U2,Q2,(max(U2)-min(U2))/2);
    points = [points,CurveBisection(p1,U11,Q11,p2,U12,Q12,tol)];
    points = [points,CurveBisection(p1,U11,Q11,p2,U22,Q22,tol)];
    points = [points,CurveBisection(p1,U21,Q21,p2,U12,Q12,tol)];
    points = [points,CurveBisection(p1,U21,Q21,p2,U22,Q22,tol)];
  end
end

function h = myConvHull(Q,flag,a,b)
% Q
% [flag,a,b] = isLine(Q,1e-7,1e-7)

if flag
  h = [a,b,a];
else
  h = Q(:,convhull(Q(1,:),Q(2,:)));
end
```

MATLAB-Beispiel:

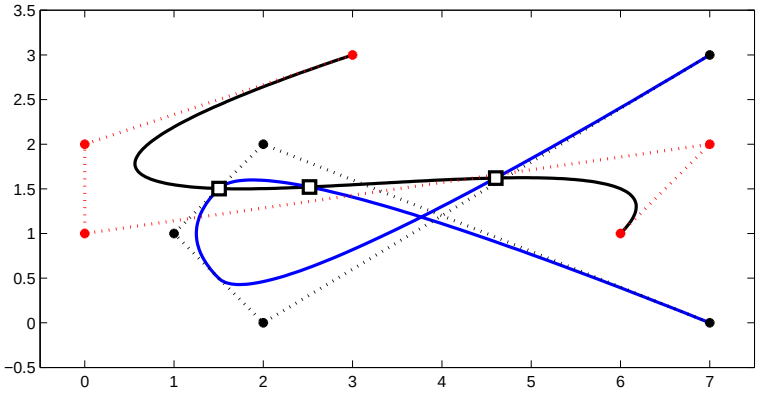


Abb. 1.17: Ergebnis der Matlab-Zeilen.

Die folgenden Zeilen stellen die Bernstein-Polynome $B_0^8, \ldots, B_8^8$ graphisch dar.

```
>> P1=[3 0,0,7,6;
       3,2,1,2,1]; p1=3;
>> U1 = [0,0,0,0,1/2,1,1,1,1];
>> P2 = [7,2,1,2,7;
         3,0,1,2,0]; p2=2;
>> U2 = [0,0,0,1/3,2/3,1,1,1];
>> CurveBisection(p1,U1,P1,p2,U2,P2, ...
                  [1e-9,1e-7])
ans =
    1.5038    4.6028    2.5178
    1.5037    1.6214    1.5214
```

2 ITERATIVE LÖSUNG LINEARER GLEICHUNGSSYSTEME

Die in Angewandter Numerik I beschriebenen direkten Verfahren gehen überwiegend von beliebigen vollbesetzten Matrizen aus. Viele praktische Probleme führen aber zu der Aufgabe, ein sehr großes lineares Gleichungssystem $Ax = b$ zu lösen, bei dem $A \in \mathbb{R}^{n\times n}$ nur schwachbesetzt (engl. sparse) ist, d.h. viele Nulleinträge besitzt (wie etwa Beispiel 2.0.1, zur Erinnerung noch einmal aus Angewandter Numerik I widerholt).

Beispiel 2.0.1 (Schwingungsgleichung aus Ang. Num. I) Gegeben sei eine elastische Saite der Länge 1, die an beiden Enden fixiert ist. Die Saite wird nun durch eine äußere Kraft f ausgelenkt (angezupft). Wir wollen die Auslenkung u der Saite aus ihrer Ruhelage als Funktion von $x \in [0,1]$ berechnen. Die gesuchte Auslenkung $u : [0,1] \to \mathbb{R}$ ist Lösung des folgenden linearen Randwertproblems zweiter Ordnung

$$-u''(x) + \lambda(x)u(x) = f(x)\ ,\ x \in (0,1), \qquad u(0) = u(1) = 0 \tag{2.1}$$

mit gegebenen $f : (0,1) \to \mathbb{R}$ und $\lambda \in \mathbb{R}$. Genaueres zur Modellierung findet man z.B. in [Arendt/Urban].
Wir wollen (2.1) näherungsweise mit Hilfe eines numerischen Verfahrens lösen. Dazu unterteilen wir $[0,1]$ in Teilintervalle gleicher Länge. Die Anzahl der Intervalle sei $N > 1, N \in \mathbb{N}$ und $h = \frac{1}{N}$ die Schrittweite. Dann setzt man $x_i := ih\ ,\ i = 0,\dots,N \quad (x_0 = 0, x_N = 1)$, die x_i werden als **Knoten** bezeichnet. Die **Schrittweite** ist $h := x_{i+1} - x_i$ für alle i, man spricht von einem **äquidistanten Gitter**. Wir wollen die Lösung an den Knoten x_i approximieren und ersetzen hierzu (wie aus der Analysis bekannt) die zweite Ableitung durch den zentralen Differenzenquotienten

$$u''(x_i) \approx \frac{1}{h^2}\big(u(x_{i-1}) - 2u(x_i) + u(x_{i+1})\big) =: D_c^2 u(x_i).$$

Es gilt bekanntlich $\|u'' - D_c^2 u\| = \mathcal{O}(h^2)$, falls $u \in C^4[0,1]$. Damit erhält man für die Näherung $u_i \approx u(x_i)$ also folgende Bedingungen ($\lambda_i = \lambda(x_i), f_i = f(x_i)$):

$$\begin{cases} \frac{1}{h^2}(-u_{i-1} + 2u_i - u_{i+1}) + \lambda_i u_i = f_i,\ 1 \le i \le N-1, \\ u_0 = u_N = 0, \end{cases}$$

also ein lineares Gleichungssystem der Form

$$\underbrace{\begin{bmatrix} (2+h^2\lambda_1) & -1 & & 0 \\ -1 & (2+h^2\lambda_2) & \ddots & \\ & \ddots & \ddots & -1 \\ 0 & & -1 & (2+h^2\lambda_{n-1}) \end{bmatrix}}_{=:A_h} \underbrace{\begin{bmatrix} u_1 \\ \vdots \\ \vdots \\ u_{n-1} \end{bmatrix}}_{=:u_h} = \underbrace{\begin{bmatrix} h^2 f_1 \\ \vdots \\ \vdots \\ h^2 f_{n-1} \end{bmatrix}}_{=:f_h},$$

d.h. $A_h u_h = f_h$. Für $N \to \infty$ ($h \to 0$) konvergiert die „**diskrete Lösung**“ u_h gegen die Lösung u von (2.1). Allerdings wächst die Dimension der Matrix A_h mit kleiner werdendem h. Bei mehrdimensionalen Problemen führt dies leicht zu sehr großen LGS. Wir nennen diese Matrix auch **Standardmatrix**.

Die bisherigen Verfahren (mit Ausnahme der speziellen Rekursion für Tridiagonalmatrizen) nutzen eine spezielle Struktur nicht aus und führen beim Lösen des LGS teilweise sogar zu vollbesetzten Zwischenmatrizen. Man betrachte dazu folgendes Beispiel.

Beispiel 2.0.2 Zu der Matrix

$$A = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & & & & \\ 1 & & 5 & & & \\ 1 & & & 10 & & \\ 1 & & & & 15 & \\ 1 & & & & & 10 \end{pmatrix}$$

lautet die LR-Zerlegung mit $A = L \cdot R$

$$L = \begin{pmatrix} 1 & & & & & \\ 1 & 1 & & & & \\ 1 & -1 & 1 & & & \\ 1 & -1 & -\frac{2}{3} & 1 & & \\ 1 & -1 & -\frac{2}{3} & -\frac{1}{2} & 1 & \\ 1 & -1 & -\frac{2}{3} & -\frac{1}{2} & -\frac{1}{2} & 1 \end{pmatrix}, \qquad R = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ & 1 & -1 & -1 & -1 & -1 \\ & & 3 & -2 & -2 & -2 \\ & & & \frac{20}{3} & -\frac{10}{3} & -\frac{10}{3} \\ & & & & 10 & -5 \\ & & & & & \frac{5}{2} \end{pmatrix}.$$

⚠ Obwohl A nur in der ersten Zeile und Spalte sowie auf der Diagonalen Nichtnulleinträge besitzt, sind L und R vollbesetzt.

Bemerkung 2.0.3 *Ist die Bandbreite einer Matrix groß und in jeder Zeile treten nur wenige Nichtnulleinträge auf, dann ist ein Bandlöser „teuer" und die folgenden Verfahren liefern gute Alternativen.*

Beispiel 2.0.4 (2D-Standardmatrix) Nun betrachten wir eine (zweidimensionale) quadratische Membran, die am Rand fest eingespannt ist und auf die eine äußere Kraft f wirkt. Die gesuchte Auslenkung $u : [0,1]^2 \to \mathbb{R}$ ergibt sich als Lösung des sogenannten **Dirichlet-Problems** auf dem Einheitsquadrat

$$\begin{aligned} -\Delta u(x,y) &:= -\frac{\partial^2}{\partial x^2}u(x,y) - \frac{\partial^2}{\partial y^2}u(x,y) = f(x,y), \quad (x,y) \in \Omega := (0,1)^2, \quad (2.2) \\ u(x,y) &= 0, \quad (x,y) \in \Gamma := \partial\Omega. \end{aligned}$$

Den Differenzialoperator Δ nennt man auch **Laplace-Operator**.
Wir verwenden die gleiche Idee wie im eindimensionalen Fall (Beispiel 2.0.1) und definieren ein äquidistantes Gitter

$$\Omega_h := \{(x,y) \in \overline{\Omega} : x = kh, \quad y = \ell h, \quad 0 \le k, \ell \le N+1\} \tag{2.3}$$

für $h := \frac{1}{N+1}$, $N = N_h \in \mathbb{N}$, wie in Beispiel 2.0.1. Der Rand besteht jetzt natürlich aus mehr als zwei Punkten, nämlich

$$\partial\Omega_h := \{(x,y) \in \Gamma : x = kh \text{ oder } y = \ell h, \quad 0 \le k, \ell \le N+1\}, \tag{2.4}$$

vgl. Abbildung 2.1. Wir definieren $\mathring{\Omega}_h := \Omega_h \setminus \partial\Omega_h$.
Die partiellen Ableitungen zweiter Ordnung approximieren wir wieder durch den zentralen Differenzenquotienten, also

$$\begin{aligned} \Delta u(x,y) = & \frac{\partial^2}{\partial x^2}u(x,y) + \frac{\partial^2}{\partial y^2}u(x,y) \\ \approx & \frac{1}{h^2}(u(x+h,y) - 2u(x,y) + u(x-h,y)) \\ & + \frac{1}{h^2}(u(x,y+h) - 2u(x,y) + u(x,y-h)) \\ = & \frac{1}{h^2}(u(x+h,y) + u(x-h,y) + u(x,y+h) + u(x,y-h) - 4u(x,y)). \quad (2.5) \end{aligned}$$

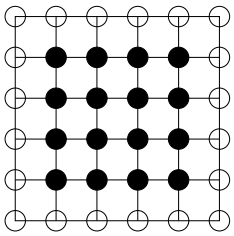


Abb. 2.1: Äquidistantes Gitter auf $\overline{\Omega} = [0,1]^2$. Die inneren Punkte ($\mathring{\Omega}_h$) sind ausgefüllt dargestellt, $\partial\Omega_h$ besteht aus den nicht ausgefüllten Punkten ($\circ$).

Dazu beschreiben wir zunächst das lineare Gleichungssystem $A_h u_h = f_h$. Die genaue Gestalt der Matrix $A_h \in \mathbb{R}^{N^2 \times N^2}$ hängt von der Nummerierung ab. Wir wählen die so genannte **lexikographische Nummerierung** (vgl. Abbildung 2.2)

$$z_k = (x_i, y_j), \quad x_i = ih, \quad y_j = jh, \quad k := (j-1)N + i.$$

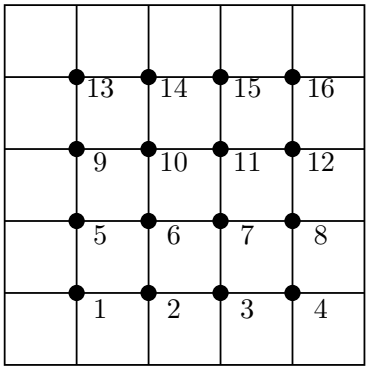


Abb. 2.2: Lexikographische Nummerierung der Gitterpunkte für $\Omega = (0,1)^2$, $N = 5$.

Dann erhält man für das lineare Gleichungssystem $A_h u_h = f_h$ eine **Block-Tridiagonalmatrix**

$$A_h = \begin{bmatrix} B_h & C_h & & & 0 \\ C_h & B_h & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & B_h & C_h \\ 0 & & & C_h & B_h \end{bmatrix} \in \mathbb{R}^{N^2 \times N^2}$$

mit den Blöcken

$$B_h = \begin{bmatrix} 4 & -1 & & & 0 \\ -1 & 4 & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & 4 & -1 \\ 0 & & & -1 & 4 \end{bmatrix} \in \mathbb{R}^{N \times N}, \quad C_h = \text{ diag } (-1) \in \mathbb{R}^{N \times N},$$

der rechten Seite $f_h = (h^2 f(z_k))_{k=1,\ldots,N^2}$, $k = (j-1)N + i$, $1 \le i, j \le N$, sowie dem Lösungsvektor $u_h = (u(z_k))_{k=1,\ldots,N^2} \in \mathbb{R}^{N^2}$. Wir nennen diese Matrix auch **2D-Standardmatrix**, sie ist die direkte Verallgemeinerung der Standardmatrix aus Beispiel 2.0.1.

Für dieses Gleichungssystem kann man keine einfache Rekursionsformel für die Cholesky-Zerlegung herleiten. Allerdings ist A_h dünnbesetzt, symmetrisch und positiv definit. Mehr zu diesem Beispiel in [Arendt/Urban]. □

Aus den oben genannten Gründen wurden schon früh iterative Verfahren zur Lösung von LGS herangezogen. Bei diesen Verfahren wird ausgehend von einem Startvektor $x^{(0)}$ eine Folge von Vektoren

$$x^{(0)} \to x^{(1)} \to x^{(2)} \to \cdots$$

mittels einer Iterationsvorschrift

$$x^{(k+1)} = \phi(x^{(k)}), \quad k = 0, 1, \ldots \tag{2.6}$$

erzeugt, die gegen die gesuchte Lösung x konvergiert. In den folgenden Abschnitten werden sowohl die klassischen Iterationsverfahren, die bereits Mitte des 19. Jahrhunderts entdeckt wurden, als auch das Gradienten–Verfahren sowie das 1952 von Hestenes und Stiefel entwickelte Verfahren der konjugierten Gradienten vorgestellt.
Allen diesen Verfahren ist gemein, dass ein einzelner Iterationsschritt $x^{(k)} \to x^{(k+1)}$ einen Rechenaufwand erfordert, welcher vergleichbar ist mit der Multiplikation von A mit einem Vektor, d.h. insbesondere mit einem geringen Aufwand, sofern A schwachbesetzt ist. Im Gegensatz zu den direkten Verfahren liefern diese Iterationsverfahren die exakte Lösung x des LGS im Allgemeinen nicht mehr nach endlich vielen Schritten. Da man aber in der Regel an der Lösung x nur bis auf eine vorgegebene Genauigkeit ϵ interessiert ist, die von der Genauigkeit der Eingabedaten abhängt (vgl. Angewandte Numerik I), scheint dieses Vorgehen sinnvoll.

2.1 Klassische Iterationsverfahren

Gegeben sei eine reguläre Matrix $A \in \mathbb{R}^{n\times n}$ und ein lineares Gleichungssystem

$$Ax = b$$

mit der exakten Lösung x. Mit Hilfe einer beliebigen regulären Matrix $B \in \mathbb{R}^{n\times n}$ erhält man Iterationsvorschriften der Form (2.6) aus der Gleichung

$$Bx + (A - B)x = b\,,$$

indem man

$$Bx^{(k+1)} + (A - B)x^{(k)} = b$$

setzt und nach $x^{(k+1)}$ auflöst

$$\begin{aligned} x^{(k+1)} &= x^{(k)} - B^{-1}(Ax^{(k)} - b) \\ &= (I - B^{-1}A)x^{(k)} + B^{-1}b\,. \end{aligned} \tag{2.7}$$

Jede Wahl einer nichtsingulären Matrix B führt zu einem möglichen **Iterationsverfahren**. Es wird umso brauchbarer, je besser B die folgenden Kriterien erfüllt

i) B ist leicht zu invertieren (**einfache Realisierbarkeit**);

ii) die Eigenwerte von $(I - B^{-1}A)$ sollen möglichst kleine Beträge haben (**Konvergenzeigenschaft**).

Wir wollen hier nun einige Beispiele angeben. Dazu verwenden wir folgende (additive) Standardzerlegung

$$A = L + D + R\,,$$

wobei $\boldsymbol{D}$ eine **Diagonalmatrix**, $\boldsymbol{L}$ eine strikte **untere Dreiecksmatrix** und $\boldsymbol{R}$ eine strikte **obere Dreiecksmatrix** seien. Die Wahl

i) $B = \gamma I$ liefert das **Richardson**-Verfahren;

ii) $B = D$ liefert das **Jacobi**-Verfahren (**Gesamtschrittverfahren**);

iii) $B = L + D$ oder $B = D + R$ liefert das **Gauß-Seidel**-Verfahren (**Einzelschrittverfahren**).

Was zeichnet nun die einzelnen Verfahren aus? Betrachten wir dazu ein Beispiel.

Beispiel 2.1.1 Zu gegebenem $n \in \mathbb{N}$ und

$$A = \begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{pmatrix} \in \mathbb{R}^{n\times n}, b = \vec{1} \in \mathbb{R}^n, x^{(0)} = \vec{0} \in \mathbb{R}^n$$

(vgl. Beispiel 2.0.1 mit $\lambda = 0$) bestimmen wir die **Konvergenzrate**

$$c := \max_k \frac{\|x - x^{(k+1)}\|_2}{\|x - x^{(k)}\|_2},$$

für das Jacobi- und das Gauß-Seidel-Verfahren ($B = D + R$), d.h. den Faktor, um den der **Fehler** in der 2-Norm in jedem Iterationsschritt **mindestens reduziert** wird.

MATLAB-Funktion: runKonvergenz.m

```
n = 10;
e = ones(n,1);
A = spdiags([e -2*e e], -1:1, n, n);
x_ex = rand(n,1);          % exakte Loesung
b = A * x_ex;              % exakte rechte Seite
x{1} = rand(n,1);          % zufaelliger Startv.
x{2}=x{1};
W{1} = triu(A);            % Gauss-Seidel
W{2} = diag(diag(A));      % Jacobi
for j = 1:length(x)
  error_old = norm(x{j}-x_ex);
  for k = 1 : 20
      x{j} = x{j} + W{j} \ (b-A*x{j});
      error_new = norm(x{j}-x_ex);
      quot{j}(k) = error_new/error_old;
      error_old = error_new;
  end
end
plot(1:20,quot{1},'m-s',1:20,quot{2},'k:*');
xlabel('Anzahl der Iterationen'), ylabel('Kontraktion')
legend({'Gauss-Seidel-Verf.','Jacobi-Verfahren'},4)
```

Dem numerischen Experiment kann man entnehmen, dass in diesem Fall beide Verfahren konvergieren, da die Konvergenzerate jeweils kleiner 1 ist, und dass das Gauß-Seidel-Verfahren schneller konvergiert, da die Konvergenzrate hier kleiner ist.

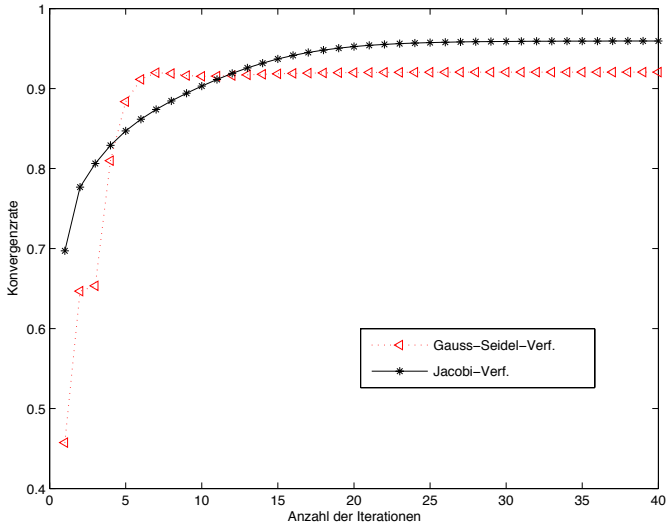


Abb. 2.3: *Kontraktionszahlen für Gesamt- und Einzelschrittverfahren.*

In der praktischen Anwendung kann man bei dem **Gauß-Seidel-Verfahren** mit $B = L + D$ folgende Formulierung verwenden, um die Anzahl der Operationen zu reduzieren

$$x^{(k+1)} = (L + D)^{-1}(b - Rx^{(k)}) ,$$

da

$$\begin{aligned} x^{(k+1)} &= x^{(k)} - B^{-1}(Ax^{(k)} - b) \\ &= x^{(k)} - B^{-1}([(A - B) + B]\, x^{(k)} - b) \\ &= x^{(k)} - B^{-1}(A - B)x^{(k)} - B^{-1}Bx^{(k)} + B^{-1}b \\ &= B^{-1}(b - (A - B)x^{(k)}) \end{aligned}$$

gilt und mit $B = L + D$ und $A - B = R$ folgt

$$x^{(k+1)} = (L + D)^{-1}(b - Rx^{(k)}) .$$

Ein Schritt des Gauß-Seidel-Verfahrens ist also etwa so aufwendig wie eine Matrix-Vektor-Multiplikation. Ähnlich kann man auch für $B = D + R$ vorgehen, d.h.

$$x^{(k+1)} = (D + R)^{-1}(b - Lx^{(k)}) .$$

Schließlich vereinfacht man das **Jacobi-Verfahren** zu

$$x^{(k+1)} = D^{-1}(b - (L + R)x^{(k)})$$

sowie das **Richardson-Verfahren** zu

$$x^{(k+1)} = x^{(k)} + \frac{1}{\gamma}(b - Ax^{(k)}) .$$

2.2 Konvergenz iterativer Verfahren

Es sei x Lösung von $Ax = b$. Mit (2.7) erhalten wir

$$\begin{aligned} x - x^{(k)} &= x - B^{-1}b - (I - B^{-1}A)x^{(k-1)} \\ &= Ix - B^{-1}Ax - (I - B^{-1}A)x^{(k-1)} \\ &= (I - B^{-1}A)(x - x^{(k-1)}) = \cdots = (I - B^{-1}A)^k(x - x^{(0)}) . \end{aligned}$$

Die Konvergenz des dargestellten Verfahrens hängt also nur von den Eigenschaften der Iterationsmatrix $I - B^{-1}A$ ab. Es sei C eine beliebige komplexwertige $(n \times n)$-Matrix, $\lambda_i := \lambda_i(C)$ $(i = 1, \ldots, n)$ seien die Eigenwerte von C. Dann bezeichnen wir mit

$$\varrho(C) := \max_{1 \le i \le n} \{|\lambda_i(C)|\}$$

den **Spektralradius** von C. Bevor wir ein Konvergenzkriterium angeben, bereiten wir noch den Begriff der Jordanschen Normalform vor.

Definition 2.2.1 (Jordan-, bzw. Elementarmatrix) *Eine Matrix $E_k(\lambda) \in \mathbb{C}^{k\times k}$ heißt **Jordanmatrix** (oder **Elementarmatrix**) zum Eigenwert λ, wenn*

$$E_k(\lambda) = \begin{pmatrix} \lambda & 1 & & 0 \\ & \ddots & \ddots & \\ & & \ddots & 1 \\ 0 & & & \lambda \end{pmatrix}. \tag{2.8}$$

Satz 2.2.2 (Jordansche Normalform (siehe z.B. [Fischer])) *Zu jeder Matrix $A \in \mathbb{C}^{n\times n}$ existiert eine reguläre Matrix $T \in \mathbb{C}^{n\times n}$, so dass*

$$A = T^{-1}JT\,,$$

*wobei J, die durch die Paare $(\lambda_1, n_1), \ldots, (\lambda_k, n_k)$ mit $\lambda_i \in \mathbb{C}$, $n_i \ge 1$ (eindeutig bis auf die Reihenfolge) bestimmte **Jordansche Normalform***

$$J = \begin{pmatrix} E_{n_1}(\lambda_1) & & 0 \\ & \ddots & \\ 0 & & E_{n_k}(\lambda_k) \end{pmatrix}$$

von A ist.

Satz 2.2.3 (Konvergenzkriterium) *Es sei $C \in \mathbb{C}^{n\times n}$. Die Folge $(C^k)_{k\in\mathbb{N}}$ ist genau dann eine Nullfolge, wenn $\varrho(C) < 1$ gilt.*

Beweis. Sei zunächst $\varrho(C) \ge 1$. Dann gibt es einen Eigenwert λ mit $|\lambda| \ge 1$ und einen Vektor $x \ne 0$ mit $Cx = \lambda x$. Wegen $C^k x = \lambda^k x$ und $\lim_{k\to\infty} \lambda^k \ne 0$ kann folglich $(C^k)_{k\in\mathbb{N}}$ keine Nullfolge sein. Die Bedingung $\varrho(C) < 1$ ist somit notwendig.
Sei nun $\varrho(C) < 1$. Weil $(TCT^{-1})^k = TC^kT^{-1}$ für jede Ähnlichkeitstransformation T gilt, reicht es, $\lim_{k\to\infty}(TCT^{-1})^k = 0$ zu zeigen. Die Matrix C lässt sich durch Ähnlichkeitstransformation auf die Jordansche Normalform J transformieren. Wir zeigen, dass $\lim_{k\to\infty} J^k = 0$ gilt, wenn alle Eigenwerte $\lambda_1, \ldots, \lambda_n$ dem Betrag nach kleiner Eins sind. Dazu sei$\mu \in \{1, \ldots, n\}$ beliebig und

$$E_\mu = E_{n_\mu}(\lambda_\mu) = \begin{pmatrix} \lambda_\mu & 1 & & 0 \\ & \ddots & \ddots & \\ & & \ddots & 1 \\ 0 & & & \lambda_\mu \end{pmatrix} \in \mathbb{C}^{n_\mu \times n_\mu}$$

eine Elementarmatrix zum Eigenwert λ_μ der Jordanschen Normalform J von C. Da offenbar

$$J^k = \begin{pmatrix} E_1^k & & & \\ & E_2^k & & \\ & & \ddots & \\ & & & E_\ell^k \end{pmatrix}$$

mit $1 \le \ell \le n$ gilt, genügt es, das Konvergenzverhalten einer Jordanmatrix E_μ zu untersuchen. Wir schreiben E_μ in der Form $E_\mu = \lambda_\mu I + S$ mit

$$S = \begin{pmatrix} 0 & 1 & & 0 \\ & \ddots & \ddots & \\ & & \ddots & 1 \\ 0 & & & 0 \end{pmatrix} \in \mathbb{C}^{n_\mu \times n_\mu}$$

und bilden $E_\mu^k = (\lambda_\mu I + S)^k$. Nach Anwendung der Binomialentwicklung und unter Beachtung von $S^{n_\mu} = 0$ erhält man die Beziehung

$$E_\mu^k = \sum_{\nu=0}^{\min\{k,\, n_\mu - 1\}} \binom{k}{\nu} \lambda_\mu^{k-\nu} S^\nu .$$

Für feste ν hat man mit

$$\binom{k}{\nu} = \frac{k!}{\nu!(k-\nu)!} = \frac{k(k-1)\cdot\ldots\cdot(k-\nu+1)}{1\cdot\ldots\cdot\nu} \le k^\nu$$

die Abschätzung

$$\left| \binom{k}{\nu} \lambda_\mu^{k-\nu} \right| \le |\lambda_\mu^{k-\nu} k^\nu| .$$

Da $|\lambda_\mu| < 1$ ist, strebt

$$k \log|\lambda_\mu| + \nu \log(k) \to -\infty \text{ für } k \to \infty$$

und mit

$$|\lambda_\mu^{k-\nu} k^\nu| \le \exp((k-\nu)\log|\lambda_\mu| + \nu \log k)$$

folgt die Konvergenz $\lim_{k\to\infty} \left| \binom{k}{\nu} \lambda_\mu^{k-\nu} \right| = 0$. Damit ist gezeigt, dass $(E_\mu^k)_{k\in\mathbb{N}}$ eine Nullfolge ist und somit auch die Behauptung. □

Um die Konvergenz der Richardson-Iteration nachzuweisen, muss der Spektralradius der Iterationsmatrix bestimmt werden. Während man diesen in dem Spezialfall einer symmetrischen, positiv definiten Matrix A exakt angeben und somit Konvergenzaussagen treffen kann, vgl. Aufgabe 2.2.4, so ist dies im allgemeinen Fall analytisch nicht möglich und numerisch sehr aufwendig. Es ist daher das Ziel der nächsten beiden Abschnitte, aus einfachen algebraischen Eigenschaften der Matrix A auf die Konvergenz der Jacobi- bzw. Gauß-Seidel-Iteration zu schließen. Anders als in Satz 2.2.3 sind die Ergebnisse in diesen beiden Abschnitten hinreichende Konvergenzkriterien, im Allgemeinen aber nicht notwendig.

Aufgabe 2.2.4 (Konvergenz des Richardson-Verfahrens für positiv definite Matrizen) Es sei $A \in \mathbb{R}^{n\times n}$ symmetrisch und positiv definit mit $\lambda_{\min}$ und $\lambda_{\max}$ als kleinstem bzw. größtem Eigenwert. Man beweise folgende Aussagen:

(a) Für die Iterationsmatrix $C_R(\gamma) = I - \gamma^{-1}A$ des Richardson-Verfahrens gilt

$$\varrho(C_R(\gamma)) = \max\left\{|1-\gamma^{-1}\lambda_{\max}|,\, |1-\gamma^{-1}\lambda_{\min}|\right\} \qquad \forall\, \gamma \in \mathbb{R}\setminus\{0\} .$$

(b) Das Richardson-Verfahren konvergiert genau dann, wenn $\gamma > \frac{\lambda_{\max}}{2}$ gilt.

(c) $\gamma^* := \frac{\lambda_{\max}+\lambda_{\min}}{2}$ minimiert den Spektralradius $\varrho(C_G(\gamma))$ für $\gamma \in \mathbb{R}\setminus\{0\}$.

(d) Es gilt $\varrho(C_G(\gamma^*)) = \frac{\lambda_{\max}-\lambda_{\min}}{\lambda_{\max}+\lambda_{\min}}$.

2.3 Konvergenz des Jacobi-Verfahrens

Bei dem **Jacobi-Verfahren** (auch Gesamtschrittverfahren genannt) werden alle Komponenten des Lösungsvektors in einem Iterationsschritt gleichzeitig korrigiert. Die Iterationsvorschrift lautet

$$x^{(k+1)} = D^{-1}(b - (L+R)x^{(k)}) ,$$

d.h. die Iterationsmatrix ist $C_J = I - D^{-1}A = -D^{-1}(L+R)$.

Satz 2.3.1 (Starkes Zeilen- und Spaltensummenkriterium) *Es sei $A \in \mathbb{C}^{n\times n}$. Das Jacobi-Verfahren konvergiert für jeden Startvektor $x^{(0)} \in \mathbb{C}^n$, wenn für die Matrix A das*

i) starke Zeilensummenkriterium:

$$|a_{ii}| > \sum_{\substack{k=1\\k\neq i}}^{n} |a_{ik}|, \quad \textit{für } i = 1, 2, \ldots, n\,,$$

d.h. A ist strikt diagonaldominant, oder das

ii) starke Spaltensummenkriterium:

$$|a_{kk}| > \sum_{\substack{i=1\\i\neq k}}^{n} |a_{ik}|, \quad \textit{für } k = 1, 2, \ldots, n\,,$$

d.h. A^T ist strikt diagonaldominant,

erfüllt ist.

Für den Beweis von Satz 2.3.1 benötigen wir das folgende Lemma.

Lemma 2.3.2 *Es sei $A \in \mathbb{C}^{n\times n}$. Dann gilt für jede natürliche p-Matrixnorm $\varrho(A) \leq \|A\|_p$.*

Beweis. Jeder Eigenwert λ von A mit zugehörigem Eigenvektor x genügt für jede natürliche p-Matrixnorm $\|\cdot\|_p$ der Beziehung

$$\frac{\|Ax\|_p}{\|x\|_p} = |\lambda|$$

und damit der Abschätzung $\|A\|_p \geq |\lambda|$. □

Beweis von Satz 2.3.1. i) Die Iterationsmatrix des Jacobi-Verfahrens ist

$$C_J = I - D^{-1}A = -D^{-1}(L + R).$$

Wenn das starke Zeilensummenkriterium erfüllt ist, gilt die Abschätzung

$$\|C_J\|_\infty = \left\| \begin{pmatrix} 0 & -\frac{a_{1,2}}{a_{1,1}} & \cdots & \cdots & -\frac{a_{1,n}}{a_{1,1}} \\ -\frac{a_{2,1}}{a_{2,2}} & 0 & -\frac{a_{2,3}}{a_{2,2}} & & -\frac{a_{2,n}}{a_{2,2}} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & \vdots \\ -\frac{a_{n,1}}{a_{n,n}} & \cdots & \cdots & -\frac{a_{n,n-1}}{a_{nn}} & 0 \end{pmatrix} \right\|_\infty = \max_{1\leq i\leq n} \sum_{\substack{j=1\\j\neq i}}^{n} \frac{|a_{ij}|}{|a_{ii}|} < 1\,.$$

Lemma 2.3.2 liefert dann die Behauptung i).
ii) Ist für A das starke Spaltensummenkriterium (ii) erfüllt, so gilt (i) für A^T. Also konvergiert das Jacobi-Verfahren für A^T und es ist daher wegen Satz 2.2.3 $\varrho(C) < 1$ für $C = I - D^{-1}A^T$. Nun hat C die gleichen Eigenwerte wie C^T und wie $D^{-1}C^T D = I - D^{-1}A = C_J$. Also ist auch $\varrho(C_J) < 1$, d.h. das Jacobi-Verfahren ist auch für die Matrix A konvergent. □

Definition 2.3.3 *Eine Matrix $A \in \mathbb{R}^{n\times n}$ heißt* **zerlegbar (reduzibel)**, *wenn es nichtleere Teilmengen N_1 und N_2 der Indexmenge $N := \{1, 2, \ldots, n\}$ gibt mit den Eigenschaften*

i) $N_1 \cap N_2 = \emptyset$;

ii) $N_1 \cup N_2 = N$;

iii) $a_{ij} = 0$ für alle $i \in N_1$ und $j \in N_2$.

Eine Matrix, die nicht zerlegbar ist, heißt unzerlegbar (irreduzibel).

Beispiel 2.3.4 i)

$$A = \begin{pmatrix} a_{11} & \dots & a_{1k} & 0 & \dots & 0 \\ \vdots & & \vdots & \vdots & & \vdots \\ a_{N1} & \dots & a_{NN} & 0 & \dots & 0 \\ a_{N+1,1} & \dots & a_{N+1,N} & a_{N+1,N+1} & \dots & a_{N+1,2N} \\ \vdots & & \vdots & \vdots & & \vdots \\ a_{2N,1} & \dots & a_{2N,N} & a_{2N,N+1} & \dots & a_{2N,2N} \end{pmatrix}$$

Die Teilmengen $N_1 = \{1, 2, \dots, N\}, N_2 = \{N+1, \dots, 2N\}$ haben alle in der Definition geforderten Eigenschaften. Somit ist A zerlegbar.

ii) Eine Tridiagonalmatrix mit nicht verschwindenden Nebendiagonal- und Diagonalelementen ist unzerlegbar.

Bemerkung 2.3.5 *Dass eine Matrix A unzerlegbar (irreduzibel) ist, kann man häufig leicht mit Hilfe des der Matrix A zugeordneten (gerichteten) Graphen $G(A)$ zeigen. Wenn A eine $n \times n$-Matrix ist, so besteht $G(A)$ aus n Knoten $K_1, \dots, K_n$ und es gibt eine gerichtete Kante $K_i \to K_j$ in $G(A)$ genau dann, wenn $a_{ij} \neq 0$. Man zeigt leicht, dass A genau dann unzerlegbar ist, falls der Graph $G(A)$ in dem Sinne zusammenhängend ist, dass es für jedes Knotenpaar (K_i, K_j) in $G(A)$ einen gerichteten Weg K_i nach K_j gibt.*

$$A = \begin{pmatrix} 2 & -1 & 0 \\ 1 & 4 & 0 \\ 0 & -1 & 3 \end{pmatrix}, \quad G(A): \quad$$

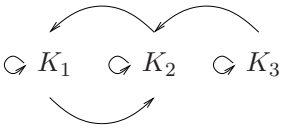


Abb. 2.4: Beispiel einer zerlegbaren Matrix A und ihres Graphen $G(A)$.

Definition 2.3.6 (Schwaches Zeilen- und Spaltensummenkriterium) *Eine Matrix $A \in \mathbb{R}^{n \times n}$ erfüllt das* **schwache Zeilensummenkriterium**, *falls*

$$\sum_{\substack{\nu=1 \\ \nu \neq \mu}}^{n} |a_{\mu\nu}| \leq |a_{\mu\mu}|$$

für alle Zeilen $\mu = 1, \dots, n$ gilt, d.h. A ist diagonaldominant, und

$$\sum_{\substack{\nu=1 \\ \nu \neq \mu_0}}^{n} |a_{\mu_0\nu}| < |a_{\mu_0\mu_0}|$$

für mindestens einen Index $\mu_0 \in \{1, \dots, n\}$ erfüllt ist.
Eine Matrix $A \in \mathbb{R}^{n \times n}$ erfüllt das **schwache Spaltensummenkriterium**, *wenn A^T das schwache Zeilensummenkriterium erfüllt.*

Satz 2.3.7 (Schwaches Zeilensummenkriterium) *Es sei $A \in \mathbb{R}^{n \times n}$ eine irreduzible Matrix, die das schwache Zeilensummenkriterium erfüllt. Dann ist das Jacobi-Verfahren konvergent.*

Zum Beweis von Satz 2.3.7 werden wir direkt den Spektralradius der Iterationsmatrix abschätzen. Die wesentliche Beobachtung dabei ist, dass jede irreduzible Matrix, die das schwache Zeilensummenkriterium erfüllt, bereits regulär ist.

Lemma 2.3.8 *Jede irreduzible Matrix $A \in \mathbb{R}^{n \times n}$, die das schwache Zeilensummenkriterium erfüllt, ist regulär und für die Diagonalelemente gilt $a_{jj} \neq 0 (j = 1, \dots, n)$.*

Beweis. Wir nehmen an, A sei nicht regulär, d.h. es existiert ein $x \in \mathbb{K}^n \setminus \{0\}$ mit $Ax = 0$. Insbesondere folgt aus der Dreiecksungleichung

$$|a_{jj}|\,|x_j| \le \underbrace{\Big|\sum_{\ell=0}^{n} a_{j\ell}\,x_\ell\Big|}_{=0} + \Big|\sum_{\substack{\ell=1\\ \ell\neq j}}^{n} a_{j\ell}\,x_\ell\Big| \le \sum_{\substack{\ell=1\\ \ell\neq j}}^{n} |a_{j\ell}|\,|x_\ell| \quad \text{für alle } j = 1,\dots,n. \tag{2.9}$$

Wir definieren die Indexmengen $J := \{j : |x_j| = \|x\|_\infty\}$ und $K := \{k : |x^{(k)}| < \|x\|_\infty\}$. Offensichtlich gilt $J \cap K = \emptyset$, $J \cup K = \{1,\dots,n\}$ und $J \neq \emptyset$. Wäre $K = \emptyset$, so könnte man in (2.9) die x_j- und x_ℓ-Terme herauskürzen und erhielte einen Widerspruch zum schwachen Zeilensummenkriterium. Also gilt $K \neq \emptyset$, und aufgrund der Irreduzibilität von M existieren Indizes $j \in J$ und $k \in K$ mit $a_{jk} \neq 0$. Mit diesem ergibt sich

$$|a_{jj}| \le \sum_{\substack{\ell=1\\ \ell\neq j}}^{n} |a_{j\ell}|\frac{|x_\ell|}{|x_j|} < \sum_{\substack{\ell=1\\ \ell\neq j}}^{n} |a_{j\ell}|\,,$$

denn der Quotient ist stets ≤ 1 wegen $|x_j| = \|x\|_\infty$ und er ist < 1 für $\ell \in K \neq \emptyset$ (also zumindest für $\ell = k$). Also erhalten wir einen Widerspruch zum schwachen Zeilensummenkriterium von A, d.h. A ist regulär. Gäbe es schließlich ein triviales Diagonalelement $a_{jj} = 0$, so folgte aus dem schwachen Zeilensummenkriterium, dass bereits die j-te Zeile die Nullzeile wäre. Da A regulär ist, folgt insbesondere $a_{jj} \neq 0$ für alle $j = 1,\dots,n$. □

Beweis von Satz 2.3.7. Wegen $a_{jj} \neq 0$ für alle $j = 1,\dots,n$ ist $C_J = -D^{-1}(A-D)$ wohldefiniert. Um $\varrho(C_J) < 1$ zu zeigen, beweisen wir, dass $M := C_J - \lambda I$ für $\lambda \in \mathbb{C}$ mit $|\lambda| \ge 1$ regulär ist. Da A irreduzibel ist, ist auch $A - D$ irreduzibel, denn es wird lediglich die Diagonale verändert. C_J entsteht durch zeilenweise Multiplikation von $A - D$ mit Werten $\neq 0$. Deshalb ist auch C_J irreduzibel. Da M und C_J sich nur auf der Diagonale unterscheiden, ist M irreduzibel. Aufgrund des schwachen Zeilensummenkriteriums von A gilt

$$\sum_{\substack{k=1\\ k\neq j}}^{n} |m_{jk}| = \sum_{\substack{k=1\\ k\neq j}}^{n} |c_{jk}^{(J)}| = \sum_{\substack{k=1\\ k\neq j}}^{n} \frac{|a_{jk}|}{|a_{jj}|} \le 1 \le |\lambda| = |m_{jj}| \text{ für alle } j = 1,\dots,n.$$

und für mindestens einen Index j gilt diese Ungleichung strikt. Also erfüllt M auch das schwache Zeilensummenkriterium und ist nach Lemma 2.3.8 insgesamt regulär. □

2.4 Konvergenz des Gauß-Seidel-Verfahrens

Die Iterationsvorschrift des **Gauß-Seidel-Verfahrens** (auch Einzelschrittverfahren genannt) für $B = L + D$ lautet

$$x^{(k+1)} = (L + D)^{-1}(b - Rx^{(k)})\,,$$

d.h. die Iterationsmatrix ist $C_{GS} := -(L + D)^{-1}R$;
die Iterationsvorschrift des Gauß-Seidel-Verfahrens für $B = D + R$ lautet

$$x^{(k+1)} = (D + R)^{-1}(b - Lx^{(k)})\,,$$

mit Iterationsmatrix $\widetilde{C}_{GS} := -(D + R)^{-1}L$.

Satz 2.4.1 (Konvergenzsatz) *Es sei $A \in \mathbb{R}^{n\times n}$ eine reguläre Matrix, die das starke Zeilensummenkriterium oder das starke Spaltensummenkriterium erfüllt. Dann sind beide Varianten des Gauß-Seidel-Verfahrens zur Lösung des linearen Gleichungssystems $Ax = b$ konvergent.*

Beweis. Sei das starke Zeilensummenkriterium erfüllt. Die Iterationsmatrizen des Gauß-Seidel-Verfahrens mit $B = L + D$ bzw. des Jacobi-Verfahrens sind $C_{GS} := -(L + D)^{-1}R$ bzw. $C_J := -D^{-1}(L + R)$. Wenn das starke Zeilensummenkriterium erfüllt ist, gilt die Abschätzung

$$\|C_J\|_\infty = \max_{1\le i\le n} \sum_{\substack{j=1\\ j\neq i}}^{n} \frac{|a_{ij}|}{|a_{ii}|} < 1\,.$$

Es sei jetzt $y \in \mathbb{R}^n$ beliebig und $z = C_{GS}\,y$. Durch vollständige Induktion beweisen wir, dass alle Komponenten z_i des Vektors z der Abschätzung

$$|z_i| \le \sum_{\substack{j=1\\ j\neq i}}^{n} \frac{|a_{ij}|}{|a_{ii}|}\|y\|_\infty$$

genügen. Dazu schreiben wir die Gleichung $z = C_{GS}\,y$ in $-(L + D)z = Ry$ um und schätzen ab:

$$|z_1| \le \sum_{j=2}^{n} \frac{|a_{1j}|}{|a_{11}|}|y_j| \le \sum_{j=2}^{n} \frac{|a_{1j}|}{|a_{11}|}\|y\|_\infty\,.$$

Schreiben wir das Gauß-Seidel-Verfahren mit $B = L + D$ in der Form

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

so folgt daraus dann mit der Induktionsvoraussetzung

$$\begin{aligned} |z_i| &\le \frac{1}{|a_{ii}|} \left(\sum_{j=1}^{i-1} |a_{ij}|\,|z_j| + \sum_{j=i+1}^{n} |a_{ij}|\,|y_j| \right) \\ &\overset{\text{IH}}{\le} \frac{1}{|a_{ii}|} \left(\sum_{j=1}^{i-1} |a_{ij}|\,\|C_J\|_\infty + \sum_{j=i+1}^{n} |a_{ij}| \right) \|y\|_\infty \le \sum_{\substack{j=1 \\ j\neq i}}^{n} \frac{|a_{ij}|}{|a_{ii}|} \|y\|_\infty \,. \end{aligned}$$

Hiermit erhält man die Abschätzung

$$\|C_{GS}\, y\|_\infty = \|z\|_\infty \le \|C_J\|_\infty \|y\|_\infty \qquad \forall\, y \in \mathbb{R}^n$$

und somit

$$\|C_{GS}\|_\infty \le \|C_J\|_\infty < 1 \,. \tag{2.10}$$

Da $\varrho(-(L+D)^{-1}R) = \varrho(C_{GS}) \le \|C_{GS}\|_\infty$, folgt daraus die Konvergenz des Gauß-Seidel-Verfahrens für $B = L + D$.
Um die Konvergenz des Gauß-Seidel-Verfahrens für $B = D + R$ mit $\widetilde{C}_{GS} = -(D+R)^{-1}L$ nachzuweisen, betrachten wir zunächst folgende Permutationsmatrix

$$P = \begin{pmatrix} 0 & & 1 \\ & \iddots & \\ 1 & & 0 \end{pmatrix} = P^T \in \mathbb{R}^{n\times n} .$$

Die Matrix $\widetilde{A} = PAP^T$ geht aus A durch simultane Zeilen- und Spaltenvertauschungen hervor, sodass die Gültigkeit des starken Zeilensummenkriteriums auch für $\widetilde{A}$ vorliegt. Mit dem oben Bewiesenen gilt somit $\varrho(-(\widetilde{L}+\widetilde{D})^{-1}\widetilde{R}) < 1$, wobei

$$\begin{aligned} \widetilde{L} &= PRP^T \\ \widetilde{D} &= PDP^T \\ \widetilde{R} &= PLP^T \end{aligned}$$

und daher

$$\begin{aligned} 1 &> \varrho(-(PRP^T + PDP^T)^{-1} PLP^T) = \varrho(-(P(R+D)P^T)^{-1} PLP^T) \\ &= \varrho(-P(R+D)^{-1}P^T PLP^T) = \varrho(-P(R+D)^{-1}LP^T) \\ &= \varrho(-(R+D)^{-1}L) \,. \end{aligned}$$

Also ist $\varrho(\widetilde{C}_{GS}) = \varrho(-(D+R)^{-1}L) < 1$ und somit das Gauß-Seidel-Verfahren für die Wahl $B = D + R$ konvergent.
Sei nun das starke Spaltensummenkriterium erfüllt. Dann erfüllt A^T das starke Zeilensummenkriterium und beide Varianten des Gauß-Seidel-Verfahrens konvergieren für A^T. Mit der Standardzerlegung von A^T

$$A^T = L_T + D_T + R_T \,,$$

wobei $L_T = R^T$, $D_T = D$ und $R_T = L^T$ ist, gilt somit

$$\begin{aligned} \varrho(-(R^T + D)^{-1} L^T) &= \varrho(-(L_T + D_T)^{-1} R_T) < 1 \,, \\ \varrho(-(L^T + D)^{-1} R^T) &= \varrho(-(R_T + D_T)^{-1} L_T) < 1 \,. \end{aligned}$$

Hieraus ergibt sich die Konvergenz des Gauß-Seidel-Verfahrens für $B = L + D$

$$\begin{aligned} \varrho(C_{GS}) &= \varrho(-(L+D)^{-1}R) = \varrho(-(L+D)(L+D)^{-1}R\,(L+D)^{-1}) \\ &= \varrho(-R\,(L+D)^{-1}) = \varrho(-(L^T+D)^{-1}R^T) < 1 \end{aligned}$$

sowie für $B = D + R$

$$\begin{aligned} \varrho(\widetilde{C}_{GS}) &= \varrho(-(D+R)^{-1}L) = \varrho(-(D+R)(D+R)^{-1}L\,(D+R)^{-1}) \\ &= \varrho(-L\,(D+R)^{-1}) = \varrho(-(D+R^T)^{-1}L^T) < 1 \,. \end{aligned}$$

Damit sind alle Aussagen des Satzes bewiesen. □

Bemerkung 2.4.2 *Häufig verleitet* (2.10) *zu der falschen Schlussfolgerung, dass das Gauß-Seidel-Verfahren schneller als das Jacobi-Verfahren konvergiert, wenn die Matrix strikt diagonaldominant ist.*

Beispiel 2.4.3 Dass bei strikter Diagonaldominanz einer regulären Matrix $A \in \mathbb{R}^{n\times n}$ aus $\|C_{GS}\|_\infty \leq \|C_J\|_\infty < 1$ nicht $\varrho(C_{GS}) \leq \varrho(C_J)$ folgen muss, sieht man, wenn man die Matrix A folgendermaßen wählt:

$$A = \begin{pmatrix} 50 & -10 & -20 \\ -20 & 49 & -20 \\ 20 & -10 & 49 \end{pmatrix}.$$

Dann besitzen die zugehörigen Iterationsmatrizen

$$C_{GS} = -(L+D)^{-1}R = \frac{1}{12005}\begin{pmatrix} 0 & 2401 & 4802 \\ 0 & 980 & 6860 \\ 0 & -780 & -560 \end{pmatrix}, \qquad C_J = \frac{1}{245}\begin{pmatrix} 0 & 49 & 98 \\ 100 & 0 & 100 \\ -100 & 50 & 0 \end{pmatrix}$$

nämlich die Spektralradien $\varrho(C_{GS}) = (4\sqrt{5})/49 \approx 0.1825$ und $\varrho(C_J) = 2/49 \approx 0.04082$ sowie die Maximumnormen $\|C_{GS}\|_\infty = 32/49 \approx 0.6531$ und $\|C_J\|_\infty = 40/49 \approx 0.8163$.

Bemerkung 2.4.4 *Mit Bezug auf das letzte Beispiel halten wir fest: Ist eine Matrix A strikt diagonaldominant, gilt für die Iterationsmatrizen $\|C_{GS}\|_\infty \leq \|C_J\|_\infty < 1$, es folgt im Allgemeinen aber nicht $\varrho(C_{GS}) \leq \varrho(C_J)$.*

Bemerkung 2.4.5 *Ebenfalls wäre die Schlussfolgerung aus Satz 2.4.1, dass eine Form des Gauß-Seidel-Verfahrens genau dann konvergent ist, wenn es die andere ist, falsch. Es gibt Beispiele regulärer Matrizen, für die $\varrho(C_{GS}) < 1$, aber $\varrho(\widetilde{C}_{GS}) \geq 1$ bzw. $\varrho(\widetilde{C}_{GS}) < 1$, aber $\varrho(C_{GS}) \geq 1$.*

Man betrachte dazu folgendes Beispiel.

Beispiel 2.4.6 Gegeben sei die reguläre Matrix

$$A = \begin{pmatrix} 2 & 0 & 2 \\ 2 & 2 & 2 \\ 0 & 2 & -1 \end{pmatrix}.$$

Die zugehörigen Iterationsmatrizen

$$C_{GS} = -(L+D)^{-1}R = \begin{pmatrix} 0 & 0 & -1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

$$\widetilde{C}_{GS} = -(D+R)^{-1}L = \begin{pmatrix} 0 & -2 & 0 \\ -1 & -2 & 0 \\ 0 & 2 & 0 \end{pmatrix}$$

besitzen die Spektralradien $\varrho(C_{GS}) = 0 < 1$ sowie $\varrho(\widetilde{C}_{GS}) = 1 + \sqrt{3} > 1$, d.h. das Gauß-Seidel-Verfahren für $B = L + D$ ist konvergent, für $B = D + R$ jedoch divergent.

Satz 2.4.7 (Schwaches Zeilensummenkriterium) *Ist $A \in \mathbb{R}^{n\times n}$ irreduzibel und erfüllt das schwache Zeilensummenkriterium, so sind beide Varianten des Gauß-Seidel-Verfahrens konvergent.*

Beweis. Die Wohldefiniertheit von $C_{GS} = -(L+D)^{-1}R$ und $\widetilde{C}_{GS} = -(D+R)^{-1}L$ ist wieder klar. Wir betrachten $W := C_{GS} - \lambda I$ sowie $\widetilde{W} := \widetilde{C}_{GS} - \lambda I$ für $\lambda \in \mathbb{C}$ mit $|\lambda| \geq 1$. Durch Multiplikation mit $-(L+D)$ sieht man, dass W genau dann regulär ist, wenn $M := R + \lambda L + \lambda D$ regulär ist. Analog folgt durch Multiplikation mit $-(D+R)$, dass $\widetilde{W}$ genau dann regulär ist, wenn $\widetilde{M} := L + \lambda D + \lambda R$ es ist. Offensichtlich erben M und $\widetilde{M}$ die Irreduzibilität von $A = D + L + R$. Ferner erfüllen M und $\widetilde{M}$ das schwache Zeilensummenkriterium, denn es gilt

$$\sum_{\substack{k=1\\k\neq j}}^{n} |m_{jk}| = |\lambda| \sum_{k=1}^{j-1} |a_{jk}| + \sum_{k=j+1}^{n} |a_{jk}| \leq |\lambda| \sum_{\substack{k=1\\k\neq j}}^{n} |a_{jk}| \leq |\lambda|\,|a_{jj}| = |m_{jj}|$$

sowie

$$\sum_{\substack{k=1\\k\neq j}}^{n} |\widetilde{m}_{jk}| = \sum_{k=1}^{j-1} |a_{jk}| + |\lambda| \sum_{k=j+1}^{n} |a_{jk}| \leq |\lambda| \sum_{\substack{k=1\\k\neq j}}^{n} |a_{jk}| \leq |\lambda|\,|a_{jj}| = |\widetilde{m}_{jj}|$$

für $j = 1, \ldots, n$ mit strikter Ungleichung jeweils für mindestens einen Index j. Nach Lemma 2.3.8 sind M und $\widetilde{M}$ regulär. Insgesamt erhalten wir wie zuvor $\varrho(C_{GS}) < 1$ und $\varrho(\widetilde{C}_{GS}) < 1$. □

Beispiel 2.4.8 Es sei $A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$ mit $a, b, c, d \in \mathbb{C}$. Die zugehörige Iterationsmatrix zum Jacobi-Verfahren lautet somit

$$C_J = -D^{-1}(L+R) = -\begin{pmatrix} a^{-1} & 0 \\ 0 & d^{-1} \end{pmatrix}\begin{pmatrix} 0 & b \\ c & 0 \end{pmatrix} = \begin{pmatrix} 0 & -\frac{b}{a} \\ -\frac{c}{d} & 0 \end{pmatrix}.$$

Das charakteristische Polynom hierzu lautet $p(\lambda) = \lambda^2 - \frac{bc}{ad}$ und hat Nullstellen $\lambda_{1,2} = \pm\sqrt{\frac{bc}{ad}}$. Entsprechend erhält man für die Gauß-Seidel-Verfahren

$$C_{GS} = -(L+D)^{-1}R = -\frac{1}{ad}\begin{pmatrix} d & 0 \\ -c & a \end{pmatrix}\begin{pmatrix} 0 & b \\ 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & -\frac{bd}{ad} \\ 0 & \frac{bc}{ad} \end{pmatrix},$$
$$\widetilde{C}_{GS} = -(D+R)^{-1}L = -\frac{1}{ad}\begin{pmatrix} d & -b \\ 0 & a \end{pmatrix}\begin{pmatrix} 0 & 0 \\ c & 0 \end{pmatrix} = \begin{pmatrix} \frac{bc}{ad} & 0 \\ -\frac{ac}{ad} & 0 \end{pmatrix}.$$

In beiden Fällen lautet das charakteristische Polynom $p(\lambda) = \lambda(\lambda - bc/ad)$ und hat Nullstellen

$$\lambda_1 = 0, \quad \lambda_2 = \frac{bc}{ad}.$$

womit

$$\varrho(C_J) = \sqrt{\frac{|bc|}{|ad|}} \quad \text{und} \quad \varrho(C_{GS}) = \varrho(\widetilde{C}_{GS}) = \frac{|bc|}{|ad|}$$

gilt. Man beachte $\|C_J\|_1 = \|C_J\|_\infty = \max\{|b/a|, |c/d|\}$ sowie

$$\|C_{GS}\|_1 = \frac{|b|(|c|+|d|)}{|ad|}, \qquad \|C_{GS}\|_\infty = \frac{|b|\max\{|c|,|d|\}}{|ad|},$$
$$\|\widetilde{C}_{GS}\|_1 = \frac{|c|(|a|+|b|)}{|ad|}, \qquad \|\widetilde{C}_{GS}\|_\infty = \frac{|c|\max\{|a|,|b|\}}{|ad|}.$$

Wir können somit für $A \in \mathbb{R}^{2\times 2}$ festhalten: Konvergiert das Jacobi- oder Gauß-Seidel-Verfahren, so konvergiert auch das jeweilige andere Verfahren. Und im Falle der Konvergenz, konvergiert das Gauß-Seidel doppelt so schnell wie das Jacobi-Verfahren. Die Frage ist nun: Gilt dies immer, bzw. kann dies ggf. einfach charakterisiert werden?

Definition 2.4.9 *Eine Matrix $A \in \mathbb{R}^{m\times n}$ heißt **nichtnegativ**, wenn alle Koeffizienten a_{ij} von A nichtnegativ sind.*

Satz 2.4.10 (von Stein und Rosenberg) *Die Iterationsmatrix $C_J \in \mathbb{R}^{n\times n}$ des Jacobi-Verfahrens sei nichtnegativ. Dann gilt genau eine der folgenden Aussagen:*

i) $\varrho(C_J) = \varrho(C_{GS}) = 0$

ii) $\varrho(C_J) = \varrho(C_{GS}) = 1$

iii) $0 < \varrho(C_{GS}) < \varrho(C_J) < 1$

iv) $1 < \varrho(C_J) < \varrho(C_{GS})$

Beweis. Siehe [Hämmerlin/Hoffmann]. □

Die Voraussetzung $C_J \geq 0$ ist insbesondere dann erfüllt, wenn die Matrix A positive Diagonalelemente und nichtpositive Nichtdiagonalelemente besitzt, d.h. $a_{ii} > 0$, $a_{ik} \leq 0$ für $i \neq k$. Dieser Fall liegt auch im folgenden Beispiel vor.

Beispiel 2.4.11

$$A = \begin{pmatrix} 2 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{pmatrix} \Rightarrow C_J = -D^{-1}(L+R) = \begin{pmatrix} 0 & \frac{1}{2} & 0 & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 \\ 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & 0 & \frac{1}{2} & 0 \end{pmatrix}$$

Die Iterationsmatrix ist nichtnegativ, $\varrho(C_J) = \frac{1+\sqrt{5}}{4} \approx 0.809 < 1$ und somit folgt, dass das Gauß-Seidel-Verfahren schneller ist als das Jacobi-Verfahren!

Bemerkung 2.4.12 *Dass das Gauß-Seidel-Verfahren nicht immer besser sein muss als das Jacobi-Verfahren oder aus der Divergenz des Jacobi-Verfahrens nicht auch die Divergenz des Gauß-Seidel-Verfahrens folgen muss, zeigen die folgenden beiden Beispiele.*

Beispiel 2.4.13 (Jacobi- immer schlechter als Gauß-Seidel-Verfahren?)

i) Die Iterationsmatrizen C_J bzw. C_{GS} zur Matrix

$$A = \begin{pmatrix} 1 & 2 & -2 & 2 \\ 1 & 1 & 1 & 0 \\ 2 & 2 & 1 & 2 \\ -1 & -2 & 1 & 1 \end{pmatrix}$$

lauten

$$C_J = \begin{pmatrix} 0 & -2 & 2 & -2 \\ -1 & 0 & -1 & 0 \\ -2 & -2 & 0 & -2 \\ 1 & 2 & -1 & 0 \end{pmatrix} \quad \text{bzw.} \quad C_{GS} = -(L+D)^{-1}R = \begin{pmatrix} 0 & -2 & 2 & -2 \\ 0 & 2 & -3 & 2 \\ 0 & 0 & 2 & -2 \\ 0 & 2 & -6 & 4 \end{pmatrix}$$

mit den Spektralradien $\varrho(C_J) = 0$ und $\varrho(C_{GS}) \approx 7.3850$.

ii) Die Iterationsmatrizen C_J bzw. C_{GS} zur Matrix

$$A = \frac{1}{3}\begin{pmatrix} 3 & -2 & -1 & 1 \\ 1 & 2 & -2 & 1 \\ 1 & -2 & 2 & 2 \\ 1 & -2 & 1 & 1 \end{pmatrix}$$

lauten

$$C_J = \frac{1}{6}\begin{pmatrix} 0 & 4 & 2 & -2 \\ -3 & 0 & 6 & -3 \\ -3 & 6 & 0 & -6 \\ -6 & 12 & -6 & 0 \end{pmatrix} \quad \text{bzw.} \quad C_{GS} = -(L+D)^{-1}R = \frac{1}{6}\begin{pmatrix} 0 & 4 & 2 & -2 \\ 0 & -2 & 5 & -2 \\ 0 & -4 & 4 & -7 \\ 0 & -4 & 4 & 5 \end{pmatrix}$$

mit den Spektralradien $\varrho(C_J) \approx 1.4527$ und $\varrho(C_{GS}) \approx 0.9287 < 1$.

Bemerkungen 2.4.14 *i) Die obigen Iterationsverfahren ließen sich in der Form*

$$x^{(k+1)} = B^{-1}(B-A)x^{(k)} + B^{-1}b = Cx^{(k)} + d$$

schreiben. Da die Iterationsmatrix C für alle k konstant ist, spricht man von ***stationären Iterationsverfahren****.*

ii) Das quantitative Verhalten solch stationärer Verfahren lässt sich durch die Einführung eines (Relaxations-) Parameters ω verbessern:

$$x^{(k+1)} = \omega(Cx^{(k)} + d) + (1-\omega)x^{(k)}\,.$$

Für $0 < \omega < 1$ spricht man von einem **Unterrelaxationsverfahren** *und für $\omega > 1$ von einem* **Überrelaxationsverfahren***. Man kann zeigen, dass der optimale Parameter für eine positiv definite Matrix A beim gedämpften Jacobi–Verfahren*

$$\omega_{opt} = \frac{2}{\lambda_{\min}(D^{-1}A) + \lambda_{\max}(D^{-1}A)}$$

*lautet und für das überrelaxierte Gauß-Seidel-Verfahren (SOR = **s**uccessive **o**ver**r**elaxation method) angewandt auf eine positiv definite Matrix $A = L+D+L^T$ ergibt sich der optimale Parameter zu*

$$\omega_{opt} = \frac{2}{1+\sqrt{\lambda_{\min}(D^{-1}A) + \lambda_{\max}((D+2L)D^{-1}(D+2L^T)A^{-1})}}\,.$$

Ergebnisse für allgemeinere Fälle findet man z.B. bei [Niethammer].

iii) Die Bedeutung der oben genannten Iterationsverfahren liegt heute weniger in deren unmittelbarem Einsatz zur Lösung von $Ax = b$, sondern auf Grund deren **„Glättungseigenschaften"** *als Beschleuniger anderer moderner Verfahren (vorkonditioniertes konjugiertes Gradienten–Verfahren, Mehrgitter).*

2.5 Abbruchkriterien

Da ein Iterationsverfahren aufeinanderfolgende Näherungen der Lösung liefert, ist ein praktischer Test notwendig, um das Verfahren zu stoppen, wenn die gewonnene Approximation genau genug ist. Da es nicht möglich ist, den **Fehler** $e^{(k)} := x - x^{(k)}$, d.h. den Abstand zur eigentlichen (gesuchten) Lösung, zu bestimmen, müssen andere Quantitäten gewonnen werden, die meist auf dem Residuum $r = b - Ax$ basieren.
Die hier vorgestellten Verfahren liefern eine Folge $(x^{(k)})$ von Vektoren, die gegen den Vektor x streben, welcher Lösung des linearen Gleichungssystems $Ax = b$ ist. Um effizient zu sein, muss die Methode wissen, wann sie abbrechen soll. Eine gute Methode sollte

i) feststellen, ob der Fehler $e^{(k)} := x - x^{(k)}$ klein genug ist,

ii) abbrechen, falls der Fehler nicht weiter kleiner wird oder nur noch sehr langsam, und

iii) den maximalen Aufwand, der zur Iteration verwendet wird, beschränken.

Das folgende **Abbruchkriterium** ist eine einfache, aber häufig genügende Variante. Man muss hierzu die Quantitäten *maxit*, $\|b\|$, *tol* und wenn möglich auch $\|A\|$ (und $\|A^{-1}\|$) zur Verfügung stellen. Dabei ist

- die natürliche Zahl *maxit* die **maximale Anzahl an möglichen Iterationen des Verfahrens**,

- die reelle Zahl $\|A\|$ eine Norm von A, (jede einigermaßen vernünftige Approximation des betragsgrößten Eintrags in A genügt schon),
- die reelle Zahl $\|b\|$ eine Norm von b (auch hier genügt eine einigermaßen vernünftige Approximation des betragsgrößten Eintrags in b),
- die reelle Zahl *tol* eine **Schranke für die Größe des Residuums bzw. des Fehlers**.

MATLAB-Beispiel: Beispiel eines vernünftigen Abbruchkriteriums

```
k = 0;
while 1
  k = k + 1;
  % Berechne die Approximation x^(k)
  % Berechne das Residuum r^(k) = b - A x^(k)
  % Berechne norm_ak = || A * x^(k) ||, norm_rk = || r^(k) ||
  % und norm_b = || b ||
  if (k >= maxit) | ( norm_rk <= tol * max( norm_ak, norm_b ) )
    break
  end
end
```

Da sich nach einigen Iterationen der Term $\|Ax^{(k)}\|$ nicht mehr groß ändert, braucht man diesen nicht immer neu zu bestimmen. Zu bestimmen ist eigentlich $\|e^{(k)}\| = \|A^{-1}r^{(k)}\| \leq \|A^{-1}\|\,\|r^{(k)}\|$. Man beachte, dass man $\|A^{-1}B\|$ bei den bisherigen Verfahren mit der Neumannschen-Reihe abschätzen kann: Es gilt $x^{(k+1)} = B^{-1}(B-A)x^{(k)} + B^{-1}b = Cx^{(k)} + d$ und

$$B^{-1}A = I - B^{-1}(B-A) = I - C \quad \text{sowie} \quad \|A^{-1}B\| = \|(I-C)^{-1}\| \leq \frac{1}{1-\|C\|}\,.$$

2.6 Gradienten–Verfahren

Im Folgenden nehmen wir stets an, dass

$$A \in \mathbb{R}^{n\times n} \quad \text{symmetrisch positiv definit (s.p.d.) ist.} \tag{2.11}$$

Wir ordnen nun dem Gleichungssystem $Ax = b$, $b \in \mathbb{R}^n$, die Funktion

$$f : \mathbb{R}^n \to \mathbb{R}\,, \quad f(x) := \frac{1}{2}x^T Ax - b^T x$$

zu. Man sieht leicht, dass die Funktion aufgrund von (2.11) strikt konvex ist. Der Gradient von f ist $f'(x) = \frac{1}{2}(A + A^T)x - b$. Da $A = A^T$ nach Voraussetzung (2.11), lautet die Ableitung

$$f'(x) = \nabla f(x) = Ax - b\,.$$

Notwendig für ein Minimum von f ist das Verschwinden des Gradienten, d.h. $Ax = b$. Da die Hesse-Matrix $f''(x) = A$ positiv definit ist, liegt für die Lösung von $Ax = b$ tatsächlich ein Minimum vor. Das Minimum ist eindeutig.

Mit $\arg\min_{y\in\mathbb{R}^n} f(y)$ bezeichnen wir denjenigen Wert aus $\mathbb{R}^n$, der den Term f minimiert, d.h.

$$f(x) = \min_{y\in\mathbb{R}^n} f(y)\,, \quad \text{falls } x := \arg\min_{y\in\mathbb{R}^n} f(y)\,.$$

Idee:

$$Ax = b \iff x = \arg\min_{y\in\mathbb{R}^n} f(y)$$

mit $f(y) := \frac{1}{2}y^TAy - b^Ty,\ f : \mathbb{R}^n \to \mathbb{R}$.

Bewiesen haben wir soeben das folgende Lemma.

Lemma 2.6.1 *Es sei $A \in \mathbb{R}^{n\times n}$ symmetrisch positiv definit, dann gilt*

$$Ax = b \iff x = \arg\min_{y\in\mathbb{R}^n} f(y)\,.$$

Beweis. Man verwandet die Darstellung

$$f(x) = f(x^*) + \frac{1}{2}(x-x^*)^TA(x-x^*) \quad \text{mit } x^* := A^{-1}b. \tag{2.12}$$

Hieraus folgt $f(x) > f(x^*)$ für $x \neq x^*$, d.h. $x^* := A^{-1}b$ ist das eindeutige Minimum von f. Man beachte dabei, dass (2.12) ein Sonderfall der folgenden Entwicklung von f um einen beliebigen Wert $\tilde{x} \in \mathbb{R}^n$ ist, welche sich durch ausmultiplizieren zeigen lässt:

$$f(x) = f(\tilde{x}) + \langle x-\tilde{x}, A\tilde{x}-b\rangle + \frac{1}{2}\langle x-\tilde{x}, A(x-\tilde{x})\rangle$$

□

Folgerung: Man kann also Verfahren zur numerischen Optimierung/Minimierung verwenden, um das lineare Gleichungssystem zu lösen.
Der **Gradient** ist die Richtung des **steilsten Anstiegs**, also kann man $-\nabla f$ als Abstiegsrichtung wählen und entlang dieser Geraden ein Minimum suchen.

Gradienten–Verfahren (allgemein):
Es sei $\Omega \subseteq \mathbb{R}^n$, $f : \Omega \to \mathbb{R}$, $x^{(0)} \in \Omega$ Startwert, für $k = 1, 2, 3, \ldots$

1) **Bestimmung der Abstiegsrichtung:** $d^{(k)} := -\nabla f(x^{(k)})$
2) **Liniensuche:** Suche auf der Geraden $\{x^{(k)} + td^{(k)} : t \geq 0\} \cap \Omega$ ein Minimum, d.h. bestimme $\lambda_k \geq 0$ mit $f(x^{(k)} + \lambda_k d^{(k)}) \leq f(x^{(k)})$ und setze

$$x^{(k+1)} = x^{(k)} + \lambda_k d^{(k)}\,.$$

Bemerkung: Daraus folgt $f(x^{(0)}) \geq f(x^{(1)}) \geq f(x^{(2)}) \geq \cdots$

Für die quadratische Funktionen $f(x) = \frac{1}{2}x^TAx - b^Tx$ und $\Omega = \mathbb{R}^n$ kann man 1) und 2) leicht berechnen: Da $\nabla f(x) = Ax - b$, ergibt sich $d^{(k)} = -\nabla f(x^{(k)}) = b - Ax^{(k)}$. Sei $p \in \mathbb{R}^n \setminus \{0\}$ und $F(\lambda) := f(x + \lambda p)$, dann gilt für die Liniensuche

$$\begin{aligned} F(\lambda) &= f(x+\lambda p) \\ &= \frac{1}{2}\langle x+\lambda p, A(x+\lambda p)\rangle - \langle b, x+\lambda p\rangle \\ &= \frac{1}{2}\langle x, Ax\rangle - \langle b, x\rangle + \lambda\langle p, Ax-b\rangle + \frac{1}{2}\lambda^2\langle p, Ap\rangle \\ &= f(x) + \lambda\langle p, Ax-b\rangle + \frac{1}{2}\lambda^2\langle p, Ap\rangle. \end{aligned} \tag{2.13}$$

Da $p \neq 0$ nach Voraussetzung, folgt $\langle p, Ap\rangle > 0$. F ist also eine quadratische Funktion mit positivem führenden Koeffizienten. Somit folgt aus

$$0 \stackrel{!}{=} F'(\lambda) = \langle p, Ax - b\rangle + \lambda\langle p, Ap\rangle\,, \tag{2.14}$$

dass der Parameter

$$\lambda_{opt}(x,p) = \frac{\langle p, b - Ax\rangle}{\langle p, Ap\rangle}\,. \tag{2.15}$$

zu gegebenem Vektor $p \in \mathbb{R}^n \setminus \{0\}$ das Funktional $F(\lambda) := f(x + \lambda p)$ minimiert.

Für allgemeine Funktionen f wird die Liniensuche angenähert, z.B. mit der **Schrittweitenregel von Armijo**.

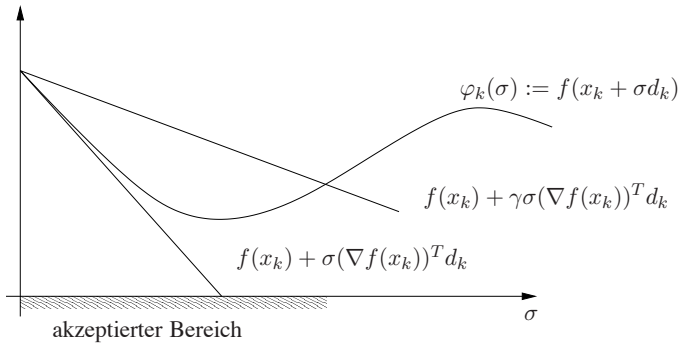


Abb. 2.5: *Optimale Wahl des Dämpfungsparameters*

Schrittweitenregel von Armijo:
Wähle $\beta \in (0,1)$ (z.B. $\beta = \frac{1}{2}$) und $\gamma \in (0,1)$ (z.B. $\gamma \in [10^{-3}, 10^{-2}]$)
Bestimme die größte Schrittweite $\sigma_k \in \{1, \beta, \beta^2, \beta^3, \ldots\}$ mit

$$f(x^{(k)}) - f(x^{(k)} + \sigma_k d^{(k)}) \geq -\gamma\sigma_k \nabla f(x^{(k)})^T d^{(k)}\,,$$

d.h.

$$f(x^{(k)} + \sigma_k d^{(k)}) \leq f(x^{(k)}) + \gamma\sigma_k \nabla f(x^{(k)})^T d^{(k)}\,.$$

Formulieren wir nun das Gradienten–Verfahren mit der optimalen Schrittweite (2.15) in folgendem Algorithmus.

Algorithmus 2.6.1: Gradienten–Verfahren: $Ax = b$

Input: `Initial guess` $x^{(0)}$

$r^{(0)} := b - Ax^{(0)}$

Iteration: $k = 0, 1, \ldots$

$a^{(k)} := Ar^{(k)}$

$\lambda_{opt} := \langle r^{(k)}, r^{(k)} \rangle \,/\, \langle r^{(k)}, a^{(k)} \rangle$

$x^{(k+1)} := x^{(k)} + \lambda_{opt}\, r^{(k)}$

$r^{(k+1)} := r^{(k)} - \lambda_{opt}\, a^{(k)}$

`Man beachte:`
$r^{(k+1)} = b - Ax^{(k+1)} = b - A(x^{(k)} + \lambda_{opt}\, r^{(k)}) = r^{(k)} - \lambda_{opt}\, Ar^{(k)}$ **.**

Wir untersuchen nun die Konvergenz des Verfahrens für quadratische Funktionen. Hierzu bietet sich die sogenannte **Energienorm** an

$$\|x\|_A := \sqrt{x^T Ax}\,, \quad (A \in \mathbb{R}^{n\times n}).$$

Man beachte: alle Normen auf dem $\mathbb{R}^n$ sind äquivalent.

Lemma 2.6.2 *Es sei $A \in \mathbb{R}^{n\times n}$ positiv definit und $x^* \in \mathbb{R}^n$ erfülle $Ax^* = b$. Dann gilt für die durch das Gradienten–Verfahren erzeugten Iterierten $x^{(k)}$ folgende Absahätzung:*

$$\|x^{(k+1)} - x^*\|_A^2 \le \|x^{(k)} - x^*\|_A^2 \left(1 - \frac{\langle r^{(k)}, r^{(k)}\rangle^2}{\langle r^{(k)}, Ar^{(k)}\rangle \langle r^{(k)}, A^{-1}r^{(k)}\rangle}\right).$$

Beweis. Es gilt

$$\begin{aligned} f(x^{(k+1)}) = f(x^{(k)} + \lambda_{opt}\, r^{(k)}) &= \frac{1}{2}\lambda_{opt}^2 \langle r^{(k)}, Ar^{(k)}\rangle - \lambda_{opt}\langle r^{(k)}, r^{(k)}\rangle + f(x^{(k)}) \\ &= f(x^{(k)}) + \frac{1}{2}\frac{\langle r^{(k)}, r^{(k)}\rangle^2}{\langle r^{(k)}, Ar^{(k)}\rangle} - \frac{\langle r^{(k)}, r^{(k)}\rangle^2}{\langle r^{(k)}, Ar^{(k)}\rangle} = f(x^{(k)}) - \frac{1}{2}\frac{\langle r^{(k)}, r^{(k)}\rangle^2}{\langle r^{(k)}, Ar^{(k)}\rangle}. \end{aligned} \tag{2.16}$$

Für die exakte Lösung x^* von $Ax = b$ gilt für $x \in \mathbb{R}^n$

$$\begin{aligned} f(x^*) + \frac{1}{2}\|x - x^*\|_A^2 &= \frac{1}{2}\langle x^*, Ax^*\rangle - \langle b, x^*\rangle + \frac{1}{2}\langle x - x^*, A(x - x^*)\rangle \\ &= \frac{1}{2}\langle x^*, Ax^*\rangle - \langle x^*, b\rangle + \frac{1}{2}\langle x, Ax\rangle - \langle x, Ax^*\rangle + \frac{1}{2}\langle x^*, Ax^*\rangle \\ &= \langle x^*, Ax^* - b\rangle + \frac{1}{2}\langle x, Ax\rangle - \langle x, b\rangle \\ &= f(x)\,, \qquad \text{d.h. } \|x - x^*\|_A^2 = 2\,(f(x) - f(x^*)) \end{aligned}$$

also mit (2.16)

$$\begin{aligned} \|x^{(k+1)} - x^*\|_A^2 = 2f(x^{(k+1)}) - 2f(x^*) &= 2f(x^{(k+1)}) - 2f(x^{(k)}) + \|x^{(k)} - x^*\|_A^2 \\ &= \|x^{(k)} - x^*\|_A^2 - \frac{\langle r^{(k)}, r^{(k)}\rangle^2}{\langle r^{(k)}, Ar^{(k)}\rangle}. \end{aligned}$$

Mit $r^{(k)} = b - Ax^{(k)} = A(x^* - x^{(k)})$ folgt wegen

$$\begin{aligned} \|x^{(k)} - x^*\|_A^2 = \|x^* - x^{(k)}\|_A^2 &= \langle x^* - x^{(k)}, A(x^* - x^{(k)})\rangle \\ &= \langle A^{-1}A(x^* - x^{(k)}), r^{(k)}\rangle = \langle r^{(k)}, A^{-1}r^{(k)}\rangle \end{aligned}$$

die Behauptung. □

Frage: Was sagt Lemma 2.6.2 bzgl. der Konvergenz und Kondition aus?

Lemma 2.6.3 (Kantorowitsch–Ungleichung) *Es sei $A \in \mathbb{R}^{n\times n}$ symmetrisch positiv definit und $\kappa := \kappa_2(A) := \|A\|_2\, \|A^{-1}\|_2$. Dann gilt für alle $x \in \mathbb{R} \setminus \{0\}$*

$$\frac{\langle x, Ax\rangle \langle x, A^{-1}x\rangle}{\langle x, x\rangle^2} \le \frac{1}{4}\left(\sqrt{\kappa} + \sqrt{\kappa^{-1}}\right)^2.$$

Beweis. Die Eigenwerte von A seien geordnet gemäß

$$0 < \lambda_1 \le \lambda_2 \le \cdots \le \lambda_n \ , \ \kappa = \frac{\lambda_n}{\lambda_1}.$$

Da A symmetrisch ist, existiert eine orthonormale Matrix Q mit $Q^T AQ = \Lambda = \text{ diag } (\lambda_i)$. Für $y = Q^T x$ gilt dann

$$x^T Ax = x^T Q\Lambda Q^T x = y^T \Lambda y = \sum_{i=1}^n \lambda_i y_i^2, \quad x^T A^{-1} x = x^T Q \Lambda^{-1} Q^T x = \sum_{i=1}^n \lambda_i^{-1} y_i^2$$

sowie $x^T x = x^T Q Q^T x = y^T y$, also

$$\frac{\langle x, Ax\rangle \langle x, A^{-1}x\rangle}{\langle x, x\rangle^2} \quad = \quad \frac{\left(\sum_i \lambda_i y_i^2\right)\left(\sum_i \lambda_i^{-1} y_i^2\right)}{\|y\|_2^4} = \left(\sum_{i=1}^n \lambda_i z_i\right)\left(\sum_{i=1}^n \lambda_i^{-1} z_i\right) \tag{2.17}$$

mit $z_i := \frac{y_i^2}{\|y\|_2^2}$. Man beachte $\sum\limits_{i=1}^n z_i = 1$.
Für $\lambda_1 \le \alpha \le \lambda_n$ folgt

$$0 \ge (\alpha - \lambda_1)(\alpha - \lambda_n) = \alpha^2 - \alpha(\lambda_1 + \lambda_n) + \lambda_1 \lambda_n$$

und somit

$$\lambda_1\lambda_n + \lambda_k^2 \le \lambda_k(\lambda_1 + \lambda_n) \quad \Rightarrow \quad \frac{\lambda_1\lambda_n}{\lambda_k} + \lambda_k \le \lambda_1 + \lambda_n \quad (k = 1, \ldots, n)\,.$$

Es gilt nun

$$\lambda_1\lambda_n \sum_{i=1}^n \lambda_i^{-1} z_i + \sum_{i=1}^n \lambda_i z_i = \left(\frac{\lambda_1\lambda_n}{\lambda_1} + \lambda_1\right) z_1 + \left(\frac{\lambda_1\lambda_n}{\lambda_2} + \lambda_2\right) z_2 + \ldots + \left(\frac{\lambda_1\lambda_n}{\lambda_n} + \lambda_n\right) z_n \le \lambda_1 + \lambda_n,$$

d.h.

$$\sum_{i=1}^n \lambda_i^{-1} z_i \le \frac{\lambda_1 + \lambda_n - \lambda}{\lambda_1\lambda_n}$$

mit $\lambda := \sum\limits_{i=1}^n \lambda_i z_i$. Somit lässt sich (2.17) abschätzen durch

$$\frac{\langle x, Ax\rangle \langle x, A^{-1}x\rangle}{\langle x, x\rangle^2} \le \underbrace{\lambda \cdot \frac{\lambda_1 + \lambda_n - \lambda}{\lambda_1\lambda_n}}_{=:h(\lambda)}.$$

Für welches λ wird nun das Polynom h maximal?

$$\begin{aligned} h'(\lambda) \quad &= \quad \frac{\lambda_1 + \lambda_n - \lambda}{\lambda_1\lambda_n} - \frac{\lambda}{\lambda_1\lambda_n} = \frac{\lambda_1 + \lambda_n}{\lambda_1\lambda_n} - \lambda\frac{2}{\lambda_1\lambda_n} \stackrel{!}{=} 0 \quad \Rightarrow \lambda^* = \frac{\lambda_1 + \lambda_n}{2} \\ h''(\lambda) \quad &= \quad -\frac{2}{\lambda_1\lambda_n} < 0 \quad \Rightarrow \lambda^* \text{ maximiert } h \end{aligned}$$

d.h.

$$\max_{\lambda\in[\lambda_1,\lambda_n]} h(\lambda) = h(\lambda^*) = \frac{(\lambda_1 + \lambda_n)^2}{4\,\lambda_1\lambda_n} = \frac{1}{4}\left(\sqrt{\frac{\lambda_1}{\lambda_n}} + \sqrt{\frac{\lambda_n}{\lambda_1}}\right)^2 = \frac{1}{4}\left(\sqrt{\kappa} + \sqrt{\kappa^{-1}}\right)^2 .$$

□

Satz 2.6.4 *Es sei $A \in \mathbb{R}^{n\times n}$ positiv definit und $x^* \in \mathbb{R}^n$ erfülle $Ax^* = b$. Dann gilt für das Gradienten–Verfahren*

$$\|x^{(k)} - x^*\|_A \le \left(\frac{\kappa - 1}{\kappa + 1}\right)^k \|x^{(0)} - x^*\|_A$$

Beweis. Lemma 2.6.2 liefert die Abschätzung

$$\|x^{(k+1)} - x^*\|_A^2 \le \|x^{(k)} - x^*\|_A^2 \left(1 - \frac{\langle r^{(k)}, r^{(k)}\rangle^2}{\langle r^{(k)}, Ar^{(k)}\rangle \langle r^{(k)}, A^{-1}r^{(k)}\rangle}\right).$$

und mit Lemma 2.6.3 ergibt sich

$$\begin{aligned} 1 - \frac{\langle r^{(k)}, r^{(k)}\rangle^2}{\langle r^{(k)}, Ar^{(k)}\rangle \langle r^{(k)}, A^{-1}r^{(k)}\rangle} &\le 1 - 4\left(\sqrt{\kappa} + \sqrt{\kappa^{-1}}\right)^{-2} = \frac{\left(\sqrt{\kappa} + \sqrt{\kappa^{-1}}\right)^2 - 4}{\left(\sqrt{\kappa} + \sqrt{\kappa^{-1}}\right)^2} \\ &= \frac{\kappa + 2 + \kappa^{-1} - 4}{\kappa + 2 + \kappa^{-1}} = \frac{\kappa^2 - 2\kappa + 1}{\kappa^2 + 2\kappa + 1} = \left(\frac{\kappa - 1}{\kappa + 1}\right)^2 . \end{aligned}$$

□

Bemerkungen 2.6.5 *i) Für große κ gilt*

$$\frac{\kappa - 1}{\kappa + 1} = \underbrace{\frac{\kappa}{\kappa + 1}}_{\approx 1} - \frac{1}{\kappa + 1} \approx 1 - \frac{1}{\kappa + 1}\,,$$

also sehr nahe bei 1, d.h. geringe Konvergenzgeschwindigkeit!

ii) Dies tritt auch schon bei einfachen Beispielen auf:

$$A = \begin{pmatrix} 1 & 0 \\ 0 & a \end{pmatrix}\ ,\ a \gg 1\ ,\ b = \begin{pmatrix} 0 \\ 0 \end{pmatrix}\ \textit{und}\ x^{(0)} = \begin{pmatrix} a \\ 1 \end{pmatrix}$$

*daraus folgt (**Übung**):*

$$\begin{pmatrix} x^{(k+1)} \\ y_{k+1} \end{pmatrix} = \rho \begin{pmatrix} x^{(k)} \\ -y^{(k)} \end{pmatrix}\ \textit{mit}\ \rho = \frac{a-1}{a+1}$$

wegen $a = \kappa_2(A)$ ist das genau die Rate aus Satz 2.6.4!

iii) Anschaulich sieht man ein „Zick–Zack–Verhalten“ der Iteration, vgl. Abbildung 2.6.

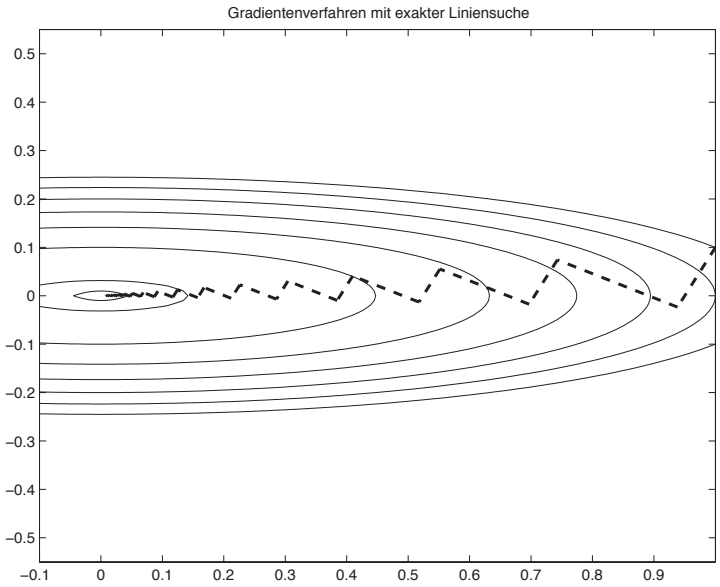


Abb. 2.6: Zick–Zack–Verhalten des Gradienten–Verfahrens mit exakter Liniensuche.

2.7 Verfahren der konjugierten Gradienten

Das Verfahren der konjugierten Gradienten (englisch: conjugate gradient, auch **cg-Verfahren** genannt) wurde 1952 von von Hestenes und Stiefel entwickelt. Man konnte zunächst zeigen, dass dieses Verfahren nach n Schritten die exakte Lösung liefert, wenn keine Rundungsfehler auftreten. In diesem Sinne ist das cg-Verfahren ein direktes Verfahren. Für große n ist diese Aussage aber wertlos. Erst 1971 gewann das cg-Verfahren durch die sogenannte **Vorkonditionierung** enorm an Bedeutung und heute gehören vorkonditionierte cg–Verfahren zu den schnellsten Verfahren, die sehr oft verwendet werden. Für das Beispiel der 2D–Standardmatrix (Beispiel 2.0.4) ist das cg-Verfahren ab einer Systemgröße von 2000–4000 Variablen deutlich besser als das Gauß–Verfahren bei zusätzlich erheblich geringerem Speicherbedarf.
Idee: Vermeide Zick–Zack–Verhalten durch Verwendung von Orthogonalität bzgl.

$$(x, y)_A := x^T A y\ ,$$

dies ergibt „konjugierte Gradienten“, daher der Name cg-Verfahren. Das Skalarprodukt bzgl. dessen man Orthogonalität misst, ist also durch die Matrix selber bestimmt.

Bemerkungen 2.7.1

i) Zwei Vektoren $x, y \in \mathbb{R}^n$ heißen **konjugiert** *oder A-***orthogonal***, falls $(x,y)_A = 0$.*

ii) Sind die Vektoren $\{x^{(1)}, \ldots, x^{(k)}\}$ ***paarweise konjugiert****, d.h.*

$$(x^{(i)}, x^{(j)})_A = \delta_{ij}\|x^{(i)}\|_A^2\ ,\ x^{(i)} \neq 0 \quad (i,j \in \{1,\ldots,k\}),$$

dann sind $\{x^{(1)}, \ldots, x^{(k)}\}$ linear unabhängig.

iii) Jeder Vektor $x \in \mathbb{R}^n$ besitzt eine eindeutige Entwicklung

$$x = \sum_{k=1}^{n} \alpha_k d^{(k)} \tag{2.18}$$

bezüglich konjugierter Richtungen $\{d^{(1)}, \ldots, d^{(n)}\}$. Aus (2.18) folgt

$$(x, d^{(i)})_A = \sum_{k=1}^{n} \alpha_k \underbrace{(d^{(k)}, d^{(i)})_A}_{=\delta_{ik}\|d^{(i)}\|_A^2} \quad = \quad \alpha_i \|d^{(i)}\|_A^2\ ,$$

also

$$\alpha_k = \frac{(d^{(k)})^T A x}{(d^{(k)})^T A d^{(k)}} \quad (k = 1, \ldots, n)\,. \tag{2.19}$$

iv) Für die Lösung x^* von $Ax = b$ gilt offenbar

$$\alpha_i = \frac{(d^{(i)})^T b}{(d^{(i)})^T A d^{(i)}}.$$

Lemma 2.7.2 *Seien $\{d^{(1)}, \ldots, d^{(n)}\}$ konjugierte Richtungen. Für jedes $x^{(0)} \in \mathbb{R}^n$ und*

$$x^{(k)} = x^{(k-1)} + \alpha_k d^{(k)}\ ,\ \alpha_k = \frac{(r^{(k-1)})^T d^{(k)}}{(d^{(k)})^T A d^{(k)}}\ ,\ r^{(k)} := b - Ax^{(k)} \quad (k \geq 1) \tag{2.20}$$

gilt nach (höchstens) n Schritten $x^{(n)} = A^{-1}b$.

Beweis. Aus (2.19), (2.20) folgt für $x^* = A^{-1}b$,

$$x^* - x^{(0)} = \sum_{k=1}^{n} \tilde{\alpha}_k d^{(k)} \text{ mit } \tilde{\alpha}_k = \frac{(d^{(k)})^T A(x^* - x^{(0)})}{(d^{(k)})^T A d^{(k)}} = \frac{(d^{(k)})^T (b - Ax^{(0)})}{(d^{(k)})^T A d^{(k)}}\,.$$

Da $d^{(k)}$ zu $d^{(i)}$, $i \neq k$, konjugiert ist, gilt

$$(d^{(k)})^T A(x^{(k-1)} - x^{(0)}) = (d^{(k)})^T A \left(\sum_{i=1}^{k-1} \alpha_i d^{(i)}\right) = \sum_{i=1}^{k-1} \alpha_i \underbrace{(d^{(k)})^T A d^{(i)}}_{=0,\ \text{da } i\neq k} = 0\,,$$

also

$$\begin{aligned}(d^{(k)})^T A(x^* - x^{(0)}) &= (d^{(k)})^T A(x^* - x^{(k-1)}) + \underbrace{(d^{(k)})^T A(x^{(k-1)} - x^{(0)})}_{=0} \\ &= (d^{(k)})^T (b - Ax^{(k-1)}) = (d^{(k)})^T r^{(k-1)} \quad \Rightarrow \tilde{\alpha}_k = \alpha_k,\end{aligned}$$

womit die Aussage bewiesen ist. □

Bemerkungen 2.7.3 *i) $r^{(k)} := b - Ax^{(k)}$ wird als* **Residuum** *von $Ax = b$ bzgl. $x^{(k)}$ bezeichnet.*

ii) Lemma 2.7.2 besagt, dass das Verfahren ein direktes Verfahren ist, welches nach n Iterationen konvergiert. Also:

$$A \text{ sparse } \Rightarrow \mathcal{O}(n^2)$$

*(dies ist **nicht** optimal).*

iii) Wie in (2.14) gilt

$$\frac{\mathrm{d}}{\mathrm{d}\lambda} f(x^{(k-1)} + \lambda\, d^{(k)}) = \lambda\, \langle d^{(k)}, Ad^{(k)}\rangle + \langle d^{(k)}, (Ax^{(k-1)} - b)\rangle\,,$$

d.h.

$$f(x^{(k)}) = f(x^{(k-1)} + \alpha_k d^{(k)}) = \min_{\lambda\in\mathbb{R}} f(x^{(k-1)} + \lambda\, d^{(k)})\,,$$

dann ist

$$\lambda_{opt} = -\frac{\langle d^{(k)}, (Ax^{(k-1)} - b)\rangle}{\langle d^{(k)}, Ad^{(k)}\rangle} = \frac{\langle r^{(k-1)}, d^{(k)}\rangle}{\langle d^{(k)}, Ad^{(k)}\rangle} = \alpha_k\,.$$

Satz 2.7.4 *Seien $\{d^{(1)}, \ldots, d^{(n)}\}$ konjugierte Richtungen und $r^{(k)}$ ($k = 0, \ldots, n-1$) die durch* (2.20) *definierten Residuen. Dann gilt*

$$(r^{(k)})^T d^{(j)} = 0 \;\; \textit{bzw.} \;\; r^{(k)} \perp U_k := \textit{span}\{d^{(1)}, \ldots, d^{(k)}\} \quad (1 \le k \le n,\; 1 \le j \le k)\,. \tag{2.21}$$

Beweis. Nach (2.20) gilt für $k \in \{1, \ldots, n\}$

$$r^{(k)} \;=\; b - Ax^{(k)} = r^{(k-1)} - \alpha_k Ad^{(k)} = r^{(k-2)} - \alpha^{(k-1)} Ad^{(k-1)} - \alpha_k Ad^{(k)} = \cdots = r^{(0)} - \sum_{j=1}^{k} \alpha_j Ad^{(j)}\,.$$

Daraus folgt nun für $1 \le j \le k$

$$\begin{aligned}(r^{(k)})^T d^{(j)} &= (r^{(0)})^T d^{(j)} - \sum_{\ell=1}^{k} \alpha_\ell\, (d^{(\ell)})^T A d^{(j)} \\ &= (r^{(0)})^T d^{(j)} - \alpha_j\, (d^{(j)})^T A d^{(j)} = (r^{(0)})^T d^{(j)} - \frac{(r^{(j-1)})^T d^{(j)}}{(d^{(j)})^T A d^{(j)}}\, (d^{(j)})^T A d^{(j)} \\ &= ((r^{(0)})^T - (r^{(j-1)})^T) d^{(j)} = \sum_{\ell=1}^{j-1} \alpha_\ell\, (d^{(\ell)})^T A d^{(j)} = 0\end{aligned}$$

womit der Satz bewiesen ist. □

Frage: Wie sind nun die $d^{(k)}$ und damit erzeugten Teilräume span$\{d^{(1)}, \ldots, d^{(k)}\}$ zu wählen?

Falls $r^{(0)} \neq 0$ gilt (sonst ist $x^{(0)}$ schon die gesuchte Lösung) setzt man $d^{(1)} = r^{(0)}$.
Für $k = 1, 2, 3, \ldots$ verfahren wir wie folgt: Falls $r^{(k)} \neq 0$ (sonst wäre ja $x^{(k)}$ schon die gesuchte Lösung), gewinnt man formal $d^{(k+1)}$ mittels des Gram-Schmidtschen-Orthogonalisierungsverfahren aus $r^{(k)}$ und den schon bestimmten konjugierten Richtungen $d^{(1)}, \ldots, d^{(k)}$, d.h.

$$d^{(k+1)} = r^{(k)} - \sum_{j=1}^{k} \frac{\langle r^{(k)}, Ad^{(j)}\rangle}{\langle d^{(j)}, Ad^{(j)}\rangle} d^{(j)}\,. \tag{2.22}$$

Damit das ganze Verfahren effizient wird, benötigen wir noch folgende Eigenschaft

$$Ad^{(k)} \in U_{k+1} := \text{span}\{d^{(1)}, \ldots, d^{(k+1)}\} = \text{span}\{r^{(0)}, \ldots, r^{(k)}\}\,,$$

wenn $r^{(k)} \neq 0$ gilt. Denn daraus ergibt sich $\langle r^{(k)}, Ad^{(j)}\rangle = 0$ für $1 \le j \le k-1$ und (2.22) verkürzt sich zu

$$d^{(k+1)} = r^{(k)} - \underbrace{\frac{\langle r^{(k)}, Ad^{(k)}\rangle}{\langle d^{(k)}, Ad^{(k)}\rangle}}_{=:\beta_{k+1}} d^{(k)}\,. \tag{2.23}$$

(Beweis: Aus der Definition von $r^{(k)} = r^{(k-1)} - \alpha_k A d^{(k)}$ folgt sofort $Ad^{(k)} \in \{r^{(0)}, \ldots, r^{(k)}\}$, da $r^{(k-1)} \in U^{(k-1)}$ und $\alpha_k \neq 0$ gilt. Die Gleichheit von U_k und $\{r^{(0)}, \ldots, r^{(k)}\}$ ergibt sich aus Aufgabe **??**.)
Für den Algorithmus schreiben wir nur noch α_k, β_k um: $\alpha_k = \frac{(r^{(k-1)})^T d^{(k)}}{(d^{(k)})^T A d^{(k)}}$ und

$$(r^{(k-1)})^T d^{(k)} = (r^{(k-1)})^T r^{(k-1)} - \beta_k \underbrace{(r^{(k-1)})^T d^{(k-1)}}_{=0} \quad \text{also}$$

$$\alpha_k = \frac{(r^{(k-1)})^T r^{(k-1)}}{(d^{(k)})^T A d^{(k)}} \tag{2.24}$$

und wegen $\alpha_k (r^{(k)})^T A d^{(k)} = (r^{(k-1)} - r^{(k)})^T r^{(k)} = -(r^{(k)})^T r^{(k)}$

$$\beta_{k+1} = \frac{(r^{(k)})^T A d^{(k)}}{(d^{(k)})^T A d^{(k)}} = -\frac{(r^{(k)})^T r^{(k)}}{\alpha_k (d^{(k)})^T A d^{(k)}} = -\frac{(r^{(k)})^T r^{(k)}}{(r^{(k-1)})^T r^{(k-1)}} \,. \tag{2.25}$$

Bemerkung 2.7.5 *Die Ausdrücke (2.24), (2.25) haben sich als numerisch stabiler und Speicherplatz-optimal herausgestellt.*

Algorithmus 2.7.1: Konjugiertes Gradienten–Verfahren (cg-Verfahren): $Ax = b$

Input: `Initial guess` $x^{(0)}$

$r^{(0)} := b - Ax^{(0)}$

$\rho_0 := \langle r^{(0)}, r^{(0)} \rangle$

$d^{(1)} := r^{(0)}$

Iteration: $k = 1, 2, \ldots$ `as long as` $k \leq n$ `and` $r^{(k)} \neq 0$

$a^{(k)} := Ad^{(k)}$

$\alpha_k := \rho^{(k-1)} \,/\, \langle d^{(k)}, a^{(k)} \rangle$

$x^{(k)} := x^{(k-1)} + \alpha_k \, d^{(k)}$

$r^{(k)} := r^{(k-1)} - \alpha_k \, a^{(k)}$

$\rho_k := \langle r^{(k)}, r^{(k)} \rangle$

$d^{(k+1)} := r^{(k)} + \frac{\rho^{(k)}}{\rho^{(k-1)}} \, d^{(k)}$

Satz 2.7.6 *Es sei A symmetrisch positiv definit. Für das cg-Verfahren und $x^* = A^{-1}b$ gilt folgende Abschätzung*

$$\|x^{(k)} - x^*\|_A \leq 2 \left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1} \right)^k \|x^{(0)} - x^*\|_A \,.$$

Beweis. Der Beweis gliedert sich in 4 Schritte. Es sei $e^{(k)} := x^* - x^{(k)}$.

(1) Zuerst zeigt man induktiv, dass Polynome $p_k \in \mathbb{P}_k$, $k = 0, \ldots, n-1$, existieren mit

$$e^{(k)} = p_k(A)e^{(0)}\,, \quad p_k(0) = 1\,. \tag{2.26}$$

$k = 0$: klar!
$k-1 \Rightarrow k$: Aus der Definition (2.20) folgt

$$\alpha_k d^{(k)} = x^{(k)} - x^{(k-1)} = x^* - x^{(k-1)} - (x^* - x^{(k)}) = e^{(k-1)} - e^{(k)}\,,$$

bzw. $e^{(k)} = e^{(k-1)} - \alpha_k d^{(k)}$. Mit (2.23) ergibt sich nun

$$\begin{aligned} d^{(k)} &= r^{(k-1)} - \beta_k d^{(k-1)} = b - Ax^{(k-1)} - \beta_k \frac{1}{\alpha_{k-1}}(e^{(k-2)} - e^{(k-1)}) \\ &= A(x^* - x^{(k-1)}) + \frac{\beta_k}{\alpha_{k-1}}(e^{(k-1)} - e^{(k-2)}) \\ &= \left(\frac{\beta_k}{\alpha_{k-1}} I + A\right) e^{(k-1)} - \frac{\beta_k}{\alpha_{k-1}} e^{(k-2)} \\ &\overset{\text{IH}}{=} \Big\{ \underbrace{\left(\frac{\beta_k}{\alpha_{k-1}} I + A\right) p^{(k-1)}(A) - \frac{\beta_k}{\alpha_{k-1}} p_{k-2}(A)}_{=:\tilde{q}_k(A)\,,\ \tilde{q}_k \in \mathbb{P}_k} \Big\} e^{(0)}\,, \end{aligned}$$

also

$$e^{(k)} = e^{(k-1)} - \alpha_k d^{(k)} \overset{\text{IH}}{=} p^{(k-1)}(A)e^{(0)} - \alpha_k d^{(k)} = \underbrace{\left[p_{k-1}(A) - \alpha_k \tilde{q}_k(A)\right]}_{=:p_k(A)} e^{(0)}.$$

(2) Nun zeigt man, dass für alle $q_k \in \mathbb{P}_k$ mit $q_k(0) = 1$ die Ungleichung $\|e^{(k)}\|_A \le \|q_k(A)e^{(0)}\|_A$ gilt. Zuerst halten wir $r^{(k)} = b - Ax^{(k)} = A(x^* - x^{(k)}) = Ae^{(k)}$ fest. Des Weiteren gilt für beliebige $\sigma^{(0)}, \ldots, \sigma^{(k-1)} \in \mathbb{R}$ (beachte: $(r^{(j)})^T r^{(k)} = \delta_{jk}$):

$$\begin{aligned} \|e^{(k)}\|_A^2 &= (e^{(k)})^T A e^{(k)} = (r^{(k)})^T e^{(k)} \\ &= (r^{(k)})^T \left(e^{(k)} + \sum_{j=0}^{k-1} \sigma_j r^{(j)}\right) = (r^{(k)})^T \left(e^{(k)} + \sum_{j=0}^{k-1} \sigma_j A e^{(j)}\right) = (Ae^{(k)})^T \left(p_k(A) + \sum_{j=0}^{k-1} \sigma_j A p_j(A)\right) e^{(0)}\,. \end{aligned}$$

Sei $q_k \in \mathbb{P}_k$ mit $q_k(0) = 1$ beliebig. Da die $\{p^{(0)}, \ldots, p^{(k-1)}\}$ linear unabhängig sind, folgt aus der letzten Umformung, dass es eindeutig bestimmte $\widetilde{\sigma}^{(0)}, \ldots, \widetilde{\sigma}^{(k-1)}$ existieren mit

$$q_k(t) = p_k(t) + \sum_{j=0}^{k-1} \widetilde{\sigma}_j t p_j(t)\,.$$

Mit $\sigma_i = \widetilde{\sigma}_i$ und der Ungleichung von Cauchy-Schwarz folgt also

$$\|e^{(k)}\|_A^2 = \langle e^{(k)}, A q_k(A) e^{(0)} \rangle = \langle A^{\frac{1}{2}} e^{(k)}, A^{\frac{1}{2}} q_k(A) e^{(0)} \rangle \le \|A^{\frac{1}{2}} e^{(k)}\| \cdot \|A^{\frac{1}{2}} q_k(A) e^{(0)}\| = \|e^{(k)}\|_A \cdot \|q_k(A) e^{(0)}\|_A\,.$$

Wir haben somit gezeigt, dass

$$\|e^{(k)}\|_A \le \|q_k(A)e^{(0)}\|_A \text{ für alle } q_k \in \mathbb{P}_k, q_k(0) = 1 \tag{2.27}$$

gilt.

(3) Die Eigenwerte von A seien $0 < \lambda_{\min} = \lambda_1 \le \cdots \le \lambda_n = \lambda_{\max}$. Es gelte $A = Q\Lambda Q^T$ mit $QQ^T = Q^T Q = I$ und $\Lambda = \operatorname{diag}(\lambda_1, \ldots, \lambda_n)$. Beachtet man weiterhin $q_k(A) = q_k(Q\Lambda Q^T) = Q q_k(\Lambda) Q^T$, so schätzt man wie folgt ab

$$\begin{aligned} \|q_k(A)e^{(0)}\|_A^2 &= (e^{(0)})^T Q\, q_k(\Lambda)\, Q^T Q \Lambda Q^T Q\, q_k(\Lambda)\, Q^T e^{(0)} = (e^{(0)})^T Q\, q_k(\Lambda)\, \Lambda\, q_k(\Lambda)\, Q^T e^{(0)} \\ &\le \max_{\lambda \in \{\lambda_1, \ldots, \lambda_n\}} \{q_k(\lambda)^2\}\, (e^{(0)})^T Q \Lambda Q^T e^{(0)} \le \left(\max_{\lambda \in [\lambda_{\min}, \lambda_{\max}]} |q_k(\lambda)|\right)^2 (e^{(0)})^T A e^{(0)}, \end{aligned} \tag{2.28}$$

d.h. es gilt

$$\|q_k(A)e^{(0)}\|_A \le \|e^{(0)}\|_A \cdot \max_{\lambda \in [\lambda_1, \lambda_n]} |q_k(\lambda)|\,.$$

(4) Man sucht nun Polynome p_k, die die rechte Seite minimieren. Man kann zeigen, dass die **Tschebyscheff–Polynome** $T_k : \mathbb{R} \to \mathbb{R}$ ($k = 0, 1, \ldots$) definiert als

$$T_k(x) := \frac{1}{2}\left[\left(x + \sqrt{x^2 - 1}\right)^k + \left(x - \sqrt{x^2 - 1}\right)^k\right]$$

auf $[-1, 1]$ folgende Eigenschaften haben

$$T_k(1) = 1\,,\ |T_k(x)| \le 1 \quad \forall\, -1 \le x \le 1$$

und minimal bzgl. $\|\cdot\|_\infty$ unter allen Polynomen $p \in \mathbb{P}_k$ mit $p(1) = 1$ sind (siehe Angweandte Numerik I). Gesucht ist nun eine Transformation, die $[\lambda_{\min}, \lambda_{\max}]$ auf $[-1, 1]$ abbildet. Somit ist $\widetilde{T}_k(z) := T_k\left(\frac{\lambda_{\max} + \lambda_{\min} - 2z}{\lambda_{\min} - \lambda_{\max}}\right)$ minimal bzgl. $\|\cdot\|_\infty$ auf $[\lambda_{\min}, \lambda_{\max}]$ unter allen Polynomen aus $p \in \mathbb{P}_k$ mit $p(\lambda_{\max}) = 1$. Man wählt also

$$q_k(z) := T_k\left(\frac{\lambda_{\max} + \lambda_{\min} - 2z}{\lambda_{\min} - \lambda_{\max}}\right) \Big/ T_k\left(\frac{\lambda_{\max} + \lambda_{\min}}{\lambda_{\min} - \lambda_{\max}}\right)$$

auf $[0, \lambda_{\max}]$, $q_k(0) = 1$.
Daraus folgt

$$\min_{q_k(0)=1} \max_{\lambda\in[\lambda_1,\lambda_n]} |q_k(\lambda)| \le \frac{1}{\left|T_k\left(\frac{\lambda_{\max}+\lambda_{\min}}{\lambda_{\min}-\lambda_{\max}}\right)\right|}$$

und

$$\left|T_k\left(\frac{\lambda_{\max}+\lambda_{\min}}{\lambda_{\min}-\lambda_{\max}}\right)\right| = \left|T_k\left(\frac{\lambda_{\max}+\lambda_{\min}}{\lambda_{\max}-\lambda_{\min}}\right)\right| = \left|T_k\left(\frac{\kappa+1}{\kappa-1}\right)\right| \ge \frac{1}{2}\left(z+\sqrt{z^2-1}\right)^k$$

mit $z = \frac{\kappa+1}{\kappa-1}$, also

$$z+\sqrt{z^2-1} = \frac{\kappa+1}{\kappa-1} + \sqrt{\frac{\kappa^2+2\kappa+1-\kappa^2+2\kappa-1}{(\kappa-1)^2}} = \frac{1}{\kappa-1}(\kappa+1+2\sqrt{\kappa}) = \frac{(\sqrt{\kappa}+1)^2}{(\sqrt{\kappa}+1)(\sqrt{\kappa}-1)} = \frac{\sqrt{\kappa}+1}{\sqrt{\kappa}-1},$$

womit der Satz bewiesen ist.

□

Bemerkung 2.7.7 *(i) Im letzten Beweis wurde unter (3) (siehe* (2.28)*) gezeigt, dass*

$$\|q_k(A)e^{(0)}\|_A \le \max_{\lambda\in\{\lambda_1,\dots,\lambda_n\}} |q(\lambda)|\, \|e^{(0)}\|_A$$

gilt. Daraus folgt sofort, dass das cg-Verfahren nach m-Schritten terminiert, wenndas Spektrum von A nur m verschiedene Eigenwerte besitzt.

(ii) Man beachte, dass sich der Fehlerreduktionsfaktor im Vergleich zum Gradienten–Verfahren (vgl. Satz 2.6.4) durch die Wurzel reduziert.

2.8 Vorkonditionierung, das pcg-Verfahren

Die Abschätzung der Konvergenzgeschwindigkeit des cg-Verfahrens hängt gemäß Satz 2.7.6 monoton von der Kondition κ der Matrix A ab. Ziel dieses Abschnittes ist es daher, das Problem $Ax = b$ so zu transformieren, dass die enstehende Matrix möglichst „gut konditioniert“ (d.h. κ möglichst klein) ist.
Idee: Betrachte anstatt

$$Ax = b \text{ mit } A \text{ s.p.d.} \tag{2.29}$$

$$\bar{A}\bar{x} = \bar{b} \text{ mit } \bar{A} = P^{-\frac{1}{2}}AP^{-\frac{1}{2}}\,, \bar{x} = P^{\frac{1}{2}}x \text{ und } \bar{b} = P^{-\frac{1}{2}} \tag{2.30}$$

mit einer symmetrischen, positiv definiten Matrix $P \in \mathbb{R}^{n\times n}$, dem so genannten **Vorkonditionierer** (manchmal auch **Präkonditionierer** genannt), und wende hierauf das cg-Verfahren an. Beachtet man, dass

$$\bar{r}^{(k)} = \bar{b} - \bar{A}\bar{x}^{(k)} = P^{-\frac{1}{2}}b - P^{-\frac{1}{2}}AP^{-\frac{1}{2}}P^{\frac{1}{2}}x^{(k)} = P^{-\frac{1}{2}}(b - Ax^{(k)}) = P^{-\frac{1}{2}}r^{(k)}$$

gilt und setzt

$$\bar{x}^{(k)} = P^{\frac{1}{2}}x^{(k)} \text{ sowie } \bar{d}^{(k)} = P^{\frac{1}{2}}d^{(k)}\,,$$

so erhält man

$$\bar{\alpha}_k = \frac{\langle \bar{r}^{(k)}, \bar{d}^{(k)}\rangle}{\langle \bar{d}^{(k)}, \bar{A}\bar{d}^{(k)}\rangle} = \frac{\langle r^{(k)}, d^{(k)}\rangle}{\langle d^{(k)}, Ad^{(k)}\rangle} = \alpha_k \qquad \Rightarrow \bar{\alpha}_k = \alpha_k$$

$$\bar{x}^{(k+1)} = \bar{x}^{(k)} + \bar{\alpha}_k\bar{d}^{(k)} = P^{\frac{1}{2}}x^{(k)} + \alpha_k P^{\frac{1}{2}}d^{(k)} \qquad \Rightarrow x^{(k+1)} = x^{(k)} + \alpha_k d^{(k)}$$

$$\bar{r}^{(k+1)} = \bar{r}^{(k)} - \bar{\alpha}_k\bar{A}\bar{d}^{(k)} = P^{-\frac{1}{2}}r^{(k)} - \alpha_k P^{-\frac{1}{2}}Ad^{(k)} \qquad \Rightarrow r^{(k+1)} = r^{(k)} - \alpha_k Ad^{(k)}$$

$$\bar{d}^{(k+1)} = \bar{r}^{(k+1)} - \frac{\langle \bar{r}^{(k+1)}, \bar{A}\bar{d}^{(k)}\rangle}{\langle \bar{d}^{(k)}, \bar{A}\bar{d}^{(k)}\rangle}\,\bar{d}^{(k)}$$

$$= P^{-\frac{1}{2}}r^{(k+1)} - \frac{\langle P^{-1}r^{(k+1)}, Ad^{(k)}\rangle}{\langle d^{(k)}, Ad^{(k)}\rangle}\,d^{(k)}$$

$$\Rightarrow d^{(k+1)} = P^{-1}r^{(k+1)} - \frac{\langle P^{-1}r^{(k+1)}, Ad^{(k)}\rangle}{\langle d^{(k)}, Ad^{(k)}\rangle}\,d^{(k)}\,.$$

Insgesamt ergibt sich folgendes **pcg-Verfahren**.

Algorithmus 2.8.1: Vorkonditionierter Konjugiertes Gradienten–Verfahren (pcg-Verfahren)

Input: Initial guess $x^{(0)}$

$r^{(0)} := b - Ax^{(0)}$

$d^{(0)} := P^{-1}r^{(0)}$

$\rho_0 := \langle d^{(0)}, r^{(0)} \rangle$

Iteration: $k = 1, 2, \ldots$ as long as $k \leq n$ and $r^{(k)} \neq 0$

$a^{(k)} := Ad^{(k)}$

$\alpha_k := \frac{\rho_k}{\langle d^{(k)}, a^{(k)} \rangle}$

$x^{(k+1)} := x^{(k)} + \alpha_k \, d^{(k)}$

$r_{k+1} := r^{(k)} - \alpha_k \, a^{(k)}$

$q^{(k+1)} := P^{-1}r^{(k+1)}$

$\rho_{k+1} := \langle q^{(k+1)}, r^{(k+1)} \rangle$

$d^{(k+1)} := q^{(k+1)} + \frac{\rho_{k+1}}{\rho_k} \, d^{(k)}$

Pro Iterationsschritt benötigt dieser Algorithmus gegenüber dem cg-Verfahren nur eine Multiplikation mit der Matrix P^{-1} mehr. Doch dieser zusätzliche Aufwand rentiert sich, sofern sich die Kondition $\kappa(\bar{A})$ der Matrix $\bar{A} = P^{-\frac{1}{2}} A P^{-\frac{1}{2}}$ gegenüber der Konditionszahl $\kappa(A)$ des nicht vorkonditionierten Systems „verbessert“, d.h. abnimmt.

Beispiel 2.8.1 Eine sehr einfache, aber häufig schon wirkungsvolle Vorkonditionierung einer s.p.d. Matrix A mit nichtverschwindenden Diagonalelementen liefert die Wahl $P^{-1} := D^{-1}$, also der Inversen der Diagonale von A. Man spricht in diesem Fall von **diagonaler Vorkonditionierung** (oder auch **Diagonalskalierung**).

Beispiel 2.8.2 Man kann auch einige (wenige) Schritte des Jacobi- oder symmetrischen Gauß–Seidel–Verfahrens als Vorkonditionierer nutzen. Dabei entspricht Jacobi der Diagonalskalierung. Symmetrisches Gauß–Seidel bedeutet folgendes: Beim klassischen Gauß–Seidel–Verfahren ist die Iterationsmatrix $Q_{\text{GS}} = D + L$ nicht symmetrisch und kann also nicht als Vorkonditionierer verwendet werden. Wähle also

$$Q_{\text{SGS}} := (D + L)D^{-1}(D + L)^T = (D + L)D^{-1}(D + R),$$

welches eine symmetrische (Iterations-)Matrix ist.
Man stellt die Iterationsmatrix **nicht** auf. Im pcg–Verfahren braucht man nur die Anwendung des Vorkonditionierers auf einen Vektor $r^{(k)}$ zu kennen. Beim symmetrischen Gauß–Seidel–Vorkonditionierer bedeutet dies, dass $r^{(k)}$ der Startwert für einige Schritte des iterativen Verfahrens ist.

Bemerkung 2.8.3 *Für Matrizen A_h, die aus der Diskretisierung der Gitterweite $h > 0$ von elliptischen partiellen Differenzialgleichungen herrühren (wie unsere Standardmatrix in den Beispielen 2.0.1 und 2.0.4) gibt es (asymptotisch)* ***optimale*** *Vorkonditionierer:*

$$\kappa(A_h B_h) = \mathcal{O}(1) \ , \ h \to 0$$

(Oswald 1988, Bramble–Pascial–Xu 1989). Dies bedeutet, dass man einen Anfangsfehler um einen Faktor reduzieren kann mit einer Anzahl von pcg-Schritten, die ***unabhänging*** *von der Gitterweite h (und damit der Matrixdimension) ist.*

Bemerkung 2.8.4 *Es ist klar, dass die Wahl des Vorkonditionierers P elementar von den Eigenschaften der Matrix A abhängt. Die Wahl eines geeigneten Vorkonditionierers ist daher oft nicht leicht. Hat man aber einen gefunden, so erhöht sich die Konvergenzgeschwindigkeit beträchtlich, was besonders für große lineare Gleichungssysteme von Bedeutung ist.*

3 ANFANGSWERTAUFGABEN BEI GEWÖHNLICHEN DIFFERENZIALGLEICHUNGEN

Theorie (Existenz, Eindeutigkeit) und einige grundlegende Lösungsmethoden für Anfangswertaufgaben gewöhnlicher Differenzialgleichungen werden in den mathematischen Grundvorlesungen behandelt. Ebenso wurde dort die Reduktion auf Systeme erster Ordnung gezeigt. Wir betrachten daher hier nur die numerische Lösung von Anfangswertaufgaben (AWA) für Systeme erster Ordnung der Form

$$y' = f(t, y), \quad y(t_0) = y^0, \tag{3.1}$$

wobei $y : \mathbb{R} \to \mathbb{R}^n$, $y(t) = (y_1(t), \ldots, y_n(t))^T$, $y^0 \in \mathbb{R}^n$, $f : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}^n$. In der Praxis (z.B. bei der Diskretisierung zeitabhängiger partieller Differenzialgleichungen) kann n jedoch **sehr** groß sein.

3.1 Theorie

Nach Hadamard (1906) heißt ein Problem **korrekt gestellt**, wenn

1.) eine Lösung existiert (Existenz),

2.) diese eindeutig ist (Eindeutigkeit),

3.) und stetig von den Daten abhängt (Stabilität).

Zu diesen Eigenschaften stellen wir einige bekannte Aussagen zusammen.

Satz 3.1.1 (Satz von Picard–Lindelöf) *Sei $f : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}^n$ stetig auf dem Gebiet*

$$R := \{(t, y) : t \in [t_0, t_0 + a], \|y - y^0\| \leq b\}$$

und genüge (in y) einer Lipschitz–Bedingung

$$\|f(t, y) - f(t, z)\| \leq L\|y - z\|, \quad \forall (t, y),\ (t, z) \in R.$$

Mit $M := \|f\|_{L^\infty(\mathbb{R})}$, $\alpha := \min\{a, \frac{b}{M}\}$ besitzt (3.1) *eine eindeutige Lösung auf $[t_0, t_0 + \alpha]$.*

Der Beweis geht über die **Fredholm'sche Integralgleichung**

$$y(t) = y^0 + \int_{t_0}^{t} f(s, y(s))ds \tag{3.2}$$

und dem Banach'schen Fixpunktsatz. Die Stabilität wird in den Grundvorlesungen aber oft nicht behandelt. Diese ist für die numerische Lösung jedoch zentral. Daher geben wir die entsprechenden Resultate hier an.

Zur Motivation, betrachte die Anfangswertaufgabe $v'(t) = u(t)\, v(t)$, $v(t_0) = c$ mit der Lösung

$$v(t) = c \cdot \exp\left\{ \int_{t_0}^{t} u(s)ds \right\}. \tag{3.3}$$

Dies sieht man leicht durch Einsetzen. Andererseits ergibt (3.2)

$$v(t) \quad = \quad v(t_0) + \int_{t_0}^{t} u(s)v(s)ds = c + \int_{t_0}^{t} u(s)v(s)ds. \tag{3.4}$$

Man könnte nun den Ansatz verfolgen, dass wenn man (3.4) durch eine Integral–**Ungleichung** ersetzt, so wird auch (3.3) zu einer Ungleichung mit einer rechten Seite, die nur von den Daten abhängt!

Satz 3.1.2 (Gronwall–Lemma) *Seien u, v auf $[t_0, t_0 + a]$ stückweise stetig, $u(t), v(t) \geq 0$ auf $[t_0, t_0 + a]$ und $c \geq 0$. Falls*

$$v(t) \leq c + \int_{t_0}^{t} u(s)v(s)ds, \quad t \in [t_0, t_0 + a], \tag{3.5}$$

dann folgt

$$v(t) \leq c \cdot \exp\left\{ \int_{t_0}^{t} u(s)ds \right\}, \quad t \in [t_0, t_0 + a]. \tag{3.6}$$

Beweis. Wir nehmen zunächst $c > 0$ an und setzen

$$W(t) := c + \int_{t_0}^{t} u(s)v(s)ds,$$

also folgt wegen (3.5)

$$v(t) \leq W(t),$$

sowie wegen $u(t), v(t) \geq 0$ auch $u(t) \cdot v(t) \geq 0$ und daher

$$0 < c \leq W(t) = c + \int_{t_0}^{t} u(s)v(s)ds, \qquad \forall t \in [t_0, t_0 + a],$$

also wieder mit (3.5)

$$W'(t) = u(t)v(t) \leq u(t)W(t),$$

sowie

$$W'(t) \geq 0.$$

Damit folgt wegen $W(t) \geq c > 0$

$$\frac{W'(t)}{W(t)} = \frac{u(t)v(t)}{W(t)} \leq \frac{u(t)W(t)}{W(t)} = u(t), \quad \forall t \in [t_0, t_0 + a]. \tag{3.7}$$

Daraus folgt

$$\log\left(\frac{W(t)}{c}\right) \quad = \quad \log(W(t)) - \log c = \log(W(t)) - \log(W(t_0)) = \int_{t_0}^{t} \frac{W'(s)}{W(s)} ds \leq \int_{t_0}^{t} u(s)ds,$$

wegen (3.7). Also folgt wegen $v(t) \leq W(t)$ die Behauptung für $c > 0$. Für $c = 0$ gilt natürlich (3.5) für jedes $\tilde{c} > 0$ und damit wie oben

$$v(t) \leq \tilde{c} \exp\left\{ \int_{t_0}^{t} u(s)ds \right\}, \quad \forall \tilde{c} > 0. \tag{3.8}$$

Da nach Voraussetzung $v(t) \geq 0$ folgt aus (3.8) $v(t) = 0$, also gilt auch (3.6). □

Aus dem Gronwall–Lemma folgt nun direkt die Stabilität der Lösung einer AWA.

Satz 3.1.3 (Stabilitätssatz) *Sei $f : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}^n$ stetig auf dem Gebiet*

$$R_D := \{(t, y) : t \in [t_0, t_0 + a], \|y - z\| \le b, z \in D\},$$

mit $D \subseteq \mathbb{R}^n$ offen und Lipschitz-stetig auf R_D bzgl. y mit Lipschitz–Konstante L. Für die nach Satz 3.1.1 eindeutige Lösung $y(t; t_0, z)$ auf $[t_0, t_0 + \alpha]$, $\alpha := \min\left\{a, \frac{b}{M}\right\}$ von $y'(t) = f(t, y(t))$, $y(t_0) = z$ gilt

$$\|y(t; t_0, z) - y(t; t_0, \tilde{z})\| \le e^{L|t-t_0|}\|z - \tilde{z}\|, \quad \forall z, \tilde{z} \in D. \tag{3.9}$$

Beweis. Wegen

$$y(t; t_0, z) = z + \int_{t_0}^{t} f(s, y(s; t_0, z))ds$$

erhält man

$$y(t; t_0, z) - y(t; t_0, \tilde{z}) = (z - \tilde{z}) + \int_{t_0}^{t} [f(s, y(s; t_0, z)) - f(s, y(s; t_0, \tilde{z}))]\, ds,$$

also wegen der Dreiecks–Ungleichung und der Lipschitz–Stetigkeit

$$\|y(t; t_0, z) - y(t, t_0, \tilde{z})\| \le \|z - \tilde{z}\| + \int_{t_0}^{t} L\|y(s; t_0, z) - y(s; t_0, \tilde{z})\|ds.$$

Mit $c := \|z - \tilde{z}\|$, $u(s) := L$, $v(s) = \|y(s; t_0, z) - y(s, t_0, \tilde{z})\|$ sind alle Voraussetzungen des Gronwall–Lemmas erfüllt, also folgt (3.9) aus (3.6). □

3.2 Einschrittverfahren

Das vielleicht einfachste Verfahren ist das **Euler'sche Polygonzugverfahren**, vgl. Abbildung 3.1. Wegen $y'(t) = f(t, y(t))$ ist die Steigung zum Zeitpunkt t von y durch $f(t, y(t))$ gegeben, die

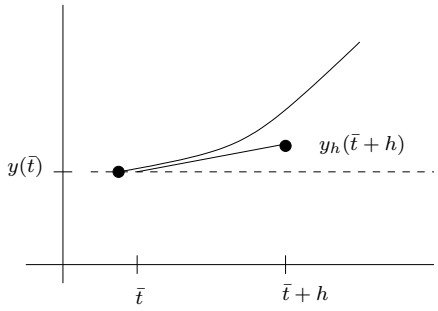


Abb. 3.1: Visualisierung des Polygonzugverfahrens (Euler-Verfahren).

jedoch die Unbekannte y enthält.

Die Idee ist nun die Steigung im Punkt $(\bar{t}, y(\bar{t}))$ (also $y'(\bar{t})$) durch $f(\bar{t}, y(\bar{t}))$ zu ersetzen und diese diese in $[\bar{t}, \bar{t} + h]$ beizubehalten

$$y_h(\bar{t} + h) = y(\bar{t}) + hf(\bar{t}, y(\bar{t})).$$

Nun brauchen wir noch eine Diskretisierung der AWA in t

$$y'(t) = f(t, y(t)), \quad y(t_0) = y^0. \tag{3.10}$$

Wähle $\{t_j\}_{j=0}^m$ und $t_m = T$ (Endzeitpunkt), $h_j := t_{j+1} - t_j$ (z.B. $t_j = t_0 + j\Delta t$) und setze

$$y^{j+1} = y^j + h_j f(t_j, y^j), \quad j = 0, \ldots, m - 1 \text{ (**Euler–Verfahren**).}$$

Dies ist das einfachste Verfahren einer ganzen Klasse von Verfahren. Allgemeiner Ansatz: ersetze in

$$y(t_{j+1}) = y(t_j) + \int_{t_j}^{t_{j+1}} f(s, y(s))ds \tag{3.11}$$

das Integral durch eine Quadraturformel.

Definition 3.2.1 *Seien $\{t_j\}_{j\geq 0}$ und y^0 gegeben. Die Folge*

$$\begin{cases} y^0 := y(t_0) \\ y^{j+1} := y^j + h_j F(f, t_j, h_j, y^j, y^{j+1}), \quad j = 0, 1, 2, \ldots \end{cases} \tag{3.12}$$

heißt ***Einschritt–Verfahren*** *(ESV) und F heißt* ***Verfahrensfunktion****. Das Verfahren heißt* ***explizit****, wenn F nicht von y^{j+1} abhängt, ansonsten* ***implizit****.*

Bemerkung 3.2.2 *(a) Bei einem expliziten Verfahren kann man y^{j+1} direkt aus y^j berechnen, bei impliziten bedeutet* (3.12) *ein (i.d.R. nichtlineares) Gleichungssystem. Dies kann man z.B. mit Fixpunkt- oder Newton–Verfahren lösen.*

(b) Beim Euler–Verfahren gilt $F(f, t, h, y, \bar{y}) \equiv F(f, t, h, y) = f(t, y(t))$, das Verfahren ist also explizit.

Beispiel 3.2.3 Ersetze $I_j := \int_{t_j}^{t_{j+1}} f(s, y(s))ds$ in (3.11) durch

(a) die linksseitige Rechteckregel: $h_j f(t_j, y(t_j))$, man erhält das Euler–Verfahren.

(b) die rechtsseitige Rechteckregel: $h_j f(t_{j+1}, y(t_{j+1}))$, also gilt $F(f, t_j, h_j, y^j, y^{j+1}) = f(t_j + h_j, y^{j+1})$. Dies ist das sogenannte implizite Euler–Verfahren.

(c) die **Trapezregel:** $F(f, t, h, y, z) = \frac{1}{2}(f(t, y) + f(t + h, z))$.

(d) die **implizite Mittelpunktregel:** $F(f, t, h, y, z) = f(t + \frac{h}{2}, \frac{1}{2}(y + z))$.

Es stellt sich die Frage, wie man solche Verfahren bewertet. Eine erste offensichtliche Idee zur lokalen Fehleranalyse besteht darin, die exakte Lösung $y(t)$ zur Zeit t (als Startwert) in das Verfahren einzusetzen und dann einen Schritt mit Schrittweite h zu machen. Der Vergleich der so erhaltenen Approximation mit $y(t + h)$ führt auf den folgenden Begriff.

Definition 3.2.4 *Sei $z(t, t^*, y^*)$ die exakte Lösung der AWA*

$$y' = f(t, y), \quad y(t^*) = y^*.$$

(a) Der Ausdruck

$$\delta_h(t, h, y) := z(t + h, t, y) - y_h(t + h, t, y) \tag{3.13}$$

heißt ***lokaler Abbruchfehler*** *des expliziten Verfahrens*

$$y_h(t + h, t, y) := y + hF(f, t, h, y).$$

(b) Die Größe

$$\tau(t,h,y) := \tau_h(t) := \frac{1}{h}\left(z(t+h,t,y) - y\right) - F(f,t,h,y) \tag{3.14}$$

heißt ***Konsistenzfehler****.*

Bemerkung 3.2.5 *Es gilt folgender Zusammenhang*

$$\delta_h(t,h,y) = z(t+h,t,y) - y(t) - hF(f,t,h,y) = h\tau(t,h,y). \tag{3.15}$$

Nun stellt sich offensichtlich die Frage, wie man dies für **implizite** Verfahren definiert. Man könnte analog zu (3.14) definieren

$$\tau(t,h,y) := \frac{1}{h}\Big(z(t+h,t,y) - y\Big) - F(f,t,h,y,z(t+h,t,y)). \tag{3.16}$$

Im letzten Term steckt jedoch die unbekannte Lösung zum Zeitpunkt $t+h$. Daher betrachtet man alternativ ausgehend von (3.13), (3.15)

$$\hat\tau(t,h,y) := \frac{1}{h}\Big(z(t+h,t,y) - y\Big) - F(f,t,h,y,y_h(t+h,t,y)).$$

Nun gilt aber mit geeigneten „Zwischenfunktionen“ ξ mit dem Satz von Taylor und $y_h(t+h,t,y) - z(t+h,t,y) = \delta_h(t,h,y)$

$$\begin{aligned}
\tau(t,h,y) - \hat\tau(t,h,y) &= F(f,t,h,y,y_h(t+h,t,y)) - F(f,t,h,y,z(t+h,t,y)) \\
&= \frac{\partial}{\partial z}F(f,t,h,y,\xi)(y_h(t+h,t,y) - z(t+h,t,y)) \\
&= \mathcal{O}(h\,\delta_h(t,h,y)),
\end{aligned}$$

wegen $\|\xi\| = \mathcal{O}(h)$, denn

$$\|\xi\| \le C_1\|y_h(t+h,t,y) - z(t+h,t,y)\| \le C_2|t+h-t| = \mathcal{O}(h).$$

Also haben beide Ausdrücke **dieselbe Fehlerordnung**, man kann also die jeweils bequemere Variante benutzen.

Definition 3.2.6 *Ein ESV heißt* ***konsistent (von der Ordnung*** p*),* *falls*

$$\|\tau(\cdot,h,\cdot)\| = \begin{cases} \mathcal{O}(h^p), & h \to 0^+, \quad p > 1, \\ o(1), & h \to 0^+, \quad p = 1, \end{cases}$$

für alle fehlenden Argumente von τ*.*

Beispiel 3.2.7 Für das explizite Euler–Verfahren gilt wegen $z(t,t,y) = y(t)$, $z'(t,t,y) = f(t,y)$ und mit $\zeta \in (0,1)$

$$\begin{aligned}
\delta_h(t,h,y) &= z(t+h,t,y) - y(t) - hf(t,y) \\
&= z(t,t,y) - y(t) + hz'(t,t,y) + \frac{h^2}{2}z''(t+\zeta h,t,y) - y(t) - hf(t,y) \\
&= \frac{h^2}{2}z''(t+\zeta h,t,y) \\
&= \frac{h^2}{2}f'(t+\zeta h, y(t+\zeta h)) \\
&= \frac{h^2}{2}\{f_t(t+\zeta h, y(t+\zeta h) + f_y(t+\zeta h, y(t+\zeta h))y'(t+\zeta h)\} \\
&= \mathcal{O}(h^2), \ \text{ falls } \ f \in C^1,
\end{aligned}$$

also $\|\tau_h\| = \mathcal{O}(h)$. Das Verfahren ist also konsistent von erster Ordnung.

Beispiel 3.2.8 Das implizite Euler–Verfahren hat ebenfalls Konsistenzordnung 1, die beiden Verfahren aus Beispiel 3.2.3 (c), (d) haben Ordnung 2.

Nun will man den Zusammenhang zur Konvergenz herstellen. Dazu benötigen wir aber zunächst eine Definition des Konvergenzbegriffes, welches wir nun einführen möchten. Ein ESV liefert Näherungswerte nur zu diskreten Zeitpunkten t_i. Dazu brauchen wir ein Fehlermaß. Dies beschreibt den globalen Fehler für alle Zeitpunkte.

Definition 3.2.9 *Sei $\Delta := \{t_j\}_{j=0}^{m_\Delta}$ ein Gitter und $y_\Delta := \Delta \to \mathbb{R}^d$ eine **Gitterfunktion** (eine Approximation der Funktion $y : \mathbb{R} \to \mathbb{R}^d$), dann heißt*

$$\varepsilon_\Delta : \Delta \to \mathbb{R}^d, \quad \varepsilon_\Delta(t) := y(t) - y_\Delta(t), \quad t \in \Delta,$$

*der **Gitterfehler** und*

$$\|\varepsilon_\Delta\|_\infty := \max_{t\in\Delta} \|\varepsilon_\Delta(t)\|$$

*der **Diskretisierungsfehler**.*

Definition 3.2.10 *Für Δ wie oben sei*

$$h_\Delta := \max_{0\le j<m_\Delta} h_j, \quad h_j := t_{j+1} - t_j.$$

*Eine Familie von Gitterfunktionen x_Δ **konvergiert gegen** x (**mit der Ordnung** p), falls*

$$\|\varepsilon_\Delta\|_\infty = \begin{cases} \mathcal{O}(h_\Delta^p), & p > 1 \\ o(1), & p = 1 \end{cases}, \quad h_\Delta \to 0 + .$$

Offenbar beschreibt die Konsistenz das lokale Fehlerverhalten des Verfahrens, die Konvergenz das globale Verhalten auf dem ganzen Gitter. Um von der Konsistenz auf die Konvergenz schließen zu können, beantworten wir die Frage, wie sich lokale Fehler akkumulieren. Dies ist eine Frage der Stabilität. Dazu benötigen wir folgendes Resultat.

Lemma 3.2.11 (Diskrete Gronwall–Ungleichung) *Seien $\{\delta_i\}_{i\in\mathbb{N}_0}$, $\{\eta_i\}_{i\in\mathbb{N}_0}$, $\{z_i\}_{i\in\mathbb{N}_0}$ Folgen mit $\delta_i \ge 0$, $\eta_i \ge 0$, $z_i \ge 0$ und*

$$z_{m+1} \le (1 + \delta_m) z_m + \eta_m, \quad m = 0, 1, 2, \ldots, \tag{3.17}$$

dann gilt

$$z_m \le \left(z_0 + \sum_{j=0}^{m-1} \eta_j \right) \exp\left(\sum_{j=0}^{m-1} \delta_j \right). \tag{3.18}$$

Falls speziell $\delta_j \equiv \delta$, $\eta_j \equiv \eta$, dann gilt

$$z_m \le z_0 e^{m\delta} + \eta \begin{cases} \frac{1}{\delta}(e^{m\delta} - 1), & \delta > 0, \\ m, & \delta = 0. \end{cases}$$

Beweis. Wir führen die Behauptung auf den kontinuierlichen Fall (Gronwall–Lemma, Satz 3.1.2) zurück. Zunächst folgt aus (3.17) rekursiv

$$\begin{aligned} z_{m+1} &\le (1+\delta_m) z_m + \eta_m = z_m + \delta_m z_m + \eta_m \le z_{m-1} + \delta_{m-1} z_{m-1} + \delta_m z_m + \eta_{m-1} + \eta_m \\ &\le \cdots \le z_0 + \sum_{j=0}^{m} \delta_j z_j + \sum_{j=0}^{m} \eta_j. \end{aligned} \tag{3.19}$$

Definiere nun

$$C := z_0 + \sum_{j=0}^{m} \eta_j, \quad v(x) := \sum_{j=0}^{m} z_j \chi_{[j,j+1]}(x), \quad u(x) := \sum_{j=0}^{m} \delta_j \chi_{[j,j+1]}(x),$$

dann gilt für $t \in [\ell, \ell+1],\ 0 \le \ell \le m-1$, wegen (3.19)

$$v(t) = z_\ell \le z_0 + \sum_{j=0}^{\ell-1} \delta_j z_j + \sum_{j=0}^{\ell-1} \eta_j \le C + \int_0^\ell u(s)v(s)ds \le C + \int_0^m u(s)v(s)ds,$$

da $\delta_j \ge 0$ und $z_j \ge 0$. Also sind die Voraussetzungen des Gronwall–Lemmas für $t_0 = 0,\ a = t$ erfüllt. Damit folgt aus Satz 3.1.2 für obige t:

$$v(t) = z_\ell \le C \exp\left\{ \int_{t_0}^t u(s)ds \right\}$$

und damit die Behauptung. Speziell für $\delta_j \equiv \delta$, $\eta_j \equiv \eta$ gilt dann wie oben

$$\begin{aligned} z_m &\le (1+\delta)z_{m-1} + \eta \le (1+\delta)[(1+\delta)z_{m-2} + \eta] + \eta = (1+\delta)^2 z_{m-2} + (1+\delta)\eta + \eta \\ &\le \cdots \le (1+\delta)^m z_0 + \eta \sum_{j=0}^{m-1} (1+\delta)^j = (1+\delta)^m z_0 + \eta \frac{(1+\delta)^m - 1}{\delta}. \end{aligned}$$

Wegen $(1+\delta)^m \le e^{m\delta}$ folgt daraus $z_m \le e^{m\delta} z_0 + \eta \frac{1}{\delta}(e^{m\delta} - 1)$ für $\delta > 0$. Mit der Regel von l'Hospital gilt schließlich

$$\lim_{\delta \to 0+} \frac{e^{m\delta} - 1}{\delta} = \lim_{\delta \to 0+} m \cdot e^{m\delta} = m,$$

und damit folgt die Behauptung. □

Dies verwenden wir nun, um eine Fehlerabschätzung zu zeigen. Wir setzen nun stets voraus, dass die AWA (3.10) eine eindeutige Lösung $y(t; t_0, y^0)$ auf einem Intervall $I = [t_0, T]$ besitzt. Definiere für ein $\gamma > 0$ die Umgebung des Lösungsgraphen

$$G := \{(t, v) : t \in I, \|y(t) - v\| \le \gamma\},$$

wobei $\|\cdot\|$ eine beliebige Norm auf dem $\mathbb{R}^n$ ist, vlg. Abbildung 3.2.

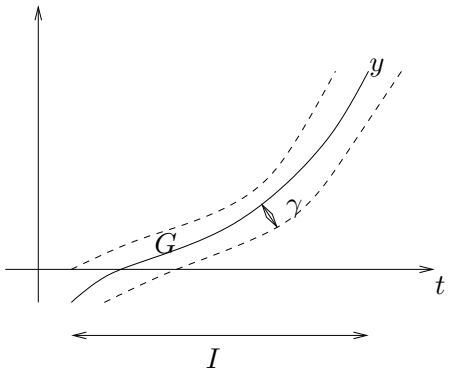


Abb. 3.2: Umgebung des Lösungsgraphen.

Analog zur Lipschitz–Bedingung an f fordern wir für die Verfahrensfunktion (wir lassen das Argument f weg)

$$\begin{cases} \|F(t,h,v) - F(t,h,w)\| \le L_F \|v - w\| & \text{(explizit)}, \\ \|F(t,h,v,w) - F(t,h,u,z)\| \le L_F(\|v-u\| + \|w-z\|) & \text{(implizit)}, \end{cases} \tag{3.20}$$

mit einer Konstanten L_F. Für den Fehler

$$\varepsilon_\Delta(t) := y(t) - y_h(t)$$

gilt dann folgende Abschätzung.

Satz 3.2.12 (Konvergenzsatz für ESV) *Die Anfangswertaufgabe* (3.1) *und das ESV erfülle die Annahme* (3.20)*, dann gilt für* $h < h_0 = (2L_F)^{-1}$

$$\|\varepsilon_\Delta\| \le \left(\|\varepsilon_0\| + 2(t-t_0) \max_{j=0,\dots,m} \|\tau(t_j,h)\| \right) e^{4L_F(t-t_0)}, \tag{3.21}$$

wobei

$$t = t_0 + \sum_{j=0}^{m-1} h_j = t_m$$

und ε_0 *eine Störung des Startwertes bedeutet.*

Bemerkung 3.2.13 *Der Satz sagt, dass für „vernünftige" ESV gilt:*

„Konsistenzordnung = Konvergenzordnung".

Beweis. Wir zeigen nur den impliziten Fall, der explizite geht analog. Zunächst gilt wegen der Definition von τ in (3.16)

$$y(t_{j+1}) = y(t_j) + h_j F(t_j, h_j, y(t_j), y(t_{j+1})) + h_j \tau(t_j, h_j)$$

und auf der anderen Seite

$$y^{j+1} = y^j + h_j F(t_j, h_j, y^j, y^{j+1}),$$

also

$$\varepsilon_{j+1} \quad := \quad \varepsilon_\Delta(t_{j+1}) = y(t_{j+1}) - y^{j+1} = \varepsilon_j + h_j\{F(t_j,h_j,y(t_j),y(t_{j+1})) - F(t_j,h_j,y^j,y^{j+1})\} + h_j\tau(t_j,h_j).$$

Mit $e_j := \|\varepsilon_j\|$ und $\tau_j := \|\tau(t_j,h_j)\|$ folgt dann mit der Dreiecks–Ungleichung

$$\begin{aligned} e_{j+1} &\le e_j + h_j \|F(t_j,h_j,y(t_j),y(t_{j+1})) - F(t_j,h_j,y^j,y^{j+1})\| + h_j\tau_j \\ &\le e_j + h_j L_F(\|y(t_j) - y^j\| + \|y(t_{j+1}) - y^{j+1}\|) + h_j\tau_j = e_j + h_j L_F(e_j + e_{j+1}) + h_j\tau_j \\ &= (1 + h_j L_F)e_j + h_j L_F e_{j+1} + h_j\tau_j, \end{aligned}$$

also

$$(1 - h_j L_F)e_{j+1} \le (1 + h_j L_F)e_j + h_j\tau_j,$$

was äquivalent ist zu

$$e_{j+1} \quad \le \quad \frac{1 + h_j L_F}{1 - h_j L_F} e_j + \frac{h_j}{1 - h_j L_F}\tau_j. \tag{3.22}$$

Nun sei h_j „hinreichend klein", d.h.

$$h_j L_F < \frac{1}{2} < 1, \text{ also } h_j < \frac{1}{2L_F} = h_0,$$

dann gilt mit der geometrischen Reihe:

$$\begin{aligned} \frac{1 + h_j L_F}{1 - h_j L_F} &= (1 + h_j L_F)(1 + h_j L_F + (h_j L_F)^2 + \dots) = 1 + 2\sum_{k=1}^{\infty} (h_j L_F)^k \\ &= 1 + 2h_j L_F \sum_{k=0}^{\infty} (h_j L_F)^k = 1 + 2h_j L_F (1 - h_j L_F)^{-1} \le 1 + 4h_j L_F, \end{aligned}$$

wegen $h_j L_F < \frac{1}{2}$ Also gilt mit (3.22): $e_{j+1} \le (1 + 4h_j L_F)e_j + 2h_j\tau_j$. Damit können wir Lemma 3.2.11 (diskrete Gronwall–Ungleichung) anwenden für

$$z_j = e_j, \quad \delta_j = 4h_j L_F, \quad \eta_j = 2h_j\tau_j,$$

und (3.18) ergibt

$$e_j \quad \le \quad \left(e_0 + \sum_{k=0}^{j-1} 2h_k\tau_k\right) \exp\left(\sum_{k=0}^{j-1} 4h_k L_F\right) \le \left\{e_0 + 2(t-t_0)\left(\max_{k=0,\dots,j-1} \|\tau_k\|\right)\right\} e^{4L_F(t-t_0)},$$

also die Behauptung. □

Praktische Bedeutung der Konvergenzordnung

Eine übliche Fragstellung ist, was es für die Praxis bedeutet, ein Verfahren niedriger oder hoher Ordnung zu verwenden. Angenommen, wir möchten eine Zielgenauigkeit ε erreichen. Wir fragen uns, wie viele Schritte mit welcher Schrittweite wir hierfür benötigen. Bei einem Verfahren der Ordnung p haben wir

$$\|\varepsilon_\Delta\|_\infty \leq C \cdot h^p,$$

also muss die Schrittweite h von der Größenordnung

$$h \sim \left(\frac{\varepsilon}{C}\right)^{1/p}$$

sein, um $\|\varepsilon_\Delta\|_\infty \leq \varepsilon$ garantieren zu können.

Beispiel 3.2.14 Sei $p = 1,\ \varepsilon = 10^{-6}$. Dies bedeutet $h \sim 10^{-6}$, wir benötigen also in der Größenordnung 10^6 Schritte.
Dann besagt (3.21) im Wesentlichen, dass der Gesamtfehler die Summe der lokalen Abbruchfehler ist. Zum Zeitpunkt t gilt bei einem äquidistantem Gitter $t - t_0 = mh$, also gilt

$$\|\varepsilon_\Delta\|_\infty = \mathcal{O}(mh\|\tau(\cdot, h)\|) \sim 10^{-12} \stackrel{!}{=} \mathcal{O}(\varepsilon),$$

der lokale Fehler muss also sehr klein sein.
Hinzu kommen aber auch noch Rundungsfehler. Bei einfacher Genauigkeit sind diese von der Ordnung $\mathcal{O}(10^{-8})$. Also ist der Fehler von der Ordnung $\mathcal{O}(10^{-12}) + \mathcal{O}(10^{-8})$.

- Bei 10^6 Schritten ($h \sim 10^{-6}$) bleibt also nur noch eine Genauigkeit von $\mathcal{O}(10^{-2})$!
- Für $h \sim 10^{-4}$ erhält man $\|h_j\tau(t_j, h_j)\| = \mathcal{O}(10^{-8})$ und 10^4 Schritte.

Bei einfacher Genauigkeit hat man also einen Fehler von $\mathcal{O}(10^{-4})$! Man erreicht also mit einer größeren Schrittweite größere Genauigkeit.
Für $p = 1$ ist dies die maximal erreichbare Genauigkeit! Man braucht also Verfahren höherer Ordnung!

3.3 Die Methode der Taylor–Entwicklung

Die naheliegende Idee ist, die Taylor-Reihe zur Konstruktion von Verfahren höherer Ordnung zu verwenden. Dazu benögien wir die Annahme, dass die Funktion f glatt in einer Umgebung von $(t, y(t)), t \in I$

$$\begin{aligned} y'(t) &= f(t, y(t)) \\ y''(t) &= f'(t, y(t)) = (Df)((t, y(t)) \\ &= \frac{\partial}{\partial t} f(t, y(t)) + \frac{\partial}{\partial y} f(t, y(t)) y'(t) \\ &= f_t(t, y(t)) + f_y(t, y(t)) f(t, y(t)) \end{aligned}$$

und allgemein

$$y^{(p+1)}(t) = (D^p f)(t, y(t)), \quad D^\ell f := D(D^{\ell-1} f).$$

Betrachte nun die Taylor–Entwicklung von y

$$\begin{aligned} \frac{1}{h}(y(t+h) - y(t)) &= y'(t) + \frac{h}{2} y''(t) + \frac{h^2}{6} y'''(t) + \cdots + \frac{h^{p-1}}{p!} y^{(p)}(t) + \mathcal{O}(h^p), \\ &= f(t, y(t)) + \frac{h}{2}(Df)(t, y, (t)) \\ &\quad + \cdots + \frac{h^{p-1}}{p!}(D^{p-1} f)(t, y(t)) + \mathcal{O}(h^p). \end{aligned}$$

Um nun ein Verfahren der Ordnung p zu erhalten, liegt folgende Definition auf der Hand

$$F(t,h,v) := f(t,v) + \frac{h}{2}(Df)(t,v) + \cdots + \frac{h^{p-1}}{p!}(D^{p-1}f)(t,v). \tag{3.23}$$

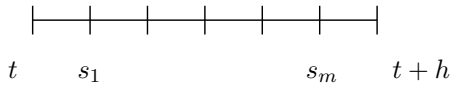
Lemma 3.3.1 *Das ESV mit Verfahrensfunktion* (3.23) *hat die Ordnung p.*

Beweis. Klar nach Konstruktion. □

Bemerkung 3.3.2 *Der gravierende Nachteil dieser Verfahren ist offensichtlich. Man benötigt die Terme $D^\ell f$ und muss dies mittels symbolischer, numerischer oder automatischer Differenziation ggf. mit großem Aufwand bestimmen. Daher sucht man nach Alternativen.*

3.4 Runge–Kutta–Verfahren

Dies ist eine wichtige Verfahrensklasse, die sehr häufig verwendet wird. Die Herangehensweise ist die, dass man Quadraturformeln hoher Ordnung in der Integralgleichung verwendet, also



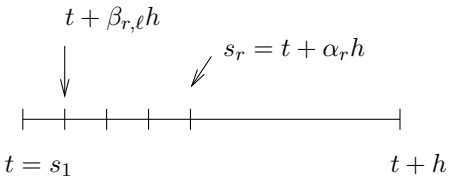
$$\int_t^{t+h} f(s,y(s))ds \approx h\sum_{r=1}^{m}\gamma_r f(s_r,y(s_r)), \qquad t \le s_1 \le \cdots \le s_m \le t+h,$$

wobei γ_r entsprechende Gewichte sind. Da $y(s_r)$ unbekannt ist, approximiere also $f(s_r,y(s_r)) \approx k_r(t,y)$, wobei letztere noch zu bestimmen sind. Dazu sei $s_1 = t$ sowie $s_r = t + \alpha_r h$ (also $\alpha_1 = 0$). Dann gilt

$$y(s_r) = y(t) + \int_t^{t+\alpha_r h} f(s,y(s))ds,$$

wobei das Integral über ein Intervall der Länge $\alpha_r h$ gebildet wird ($\alpha_r \le 1$)

$$s_r = t + \alpha_r h, \qquad f(s_r,y(s_r)) \approx k_r(t,y) \text{ auf } [t,t+\alpha_r h].$$



Man unterteilt $[t,t+\alpha_r h]$ in $t + \beta_{r,\ell}h$, $\ell = 1,\ldots,r-1$. Dort bildet man wiederum mittels $\beta_{r,1},\ldots,\beta_{r,r-1}$ eine Partition mit der Eigenschaft

$$\alpha_r = \sum_{\ell=1}^{r-1}\beta_{r,\ell}.$$

Dies führt auf den Ansatz:

$$\left\{\begin{array}{ll} k_1(t,y) & := f(t,y(t)) = f(s_1,y(s_1)) \\ k_2(t,y) & := f(t+\alpha_2 h, y(t)+h\beta_{2,1}k_1(t,y)) \qquad (Rechteckregel) \\ k_3(t,y) & := f(t+\alpha_3 h, y(t)+h(\beta_{3,1}k_1(t,y)+\beta_{3,2}k_2(t,y))) \\ & \vdots \\ k_m(t,y) & := f\left(t+\alpha_m h, y(t)+h\sum\limits_{\ell=1}^{m-1}\beta_{m,\ell}k_\ell(t,y)\right) \end{array}\right. \tag{3.24}$$

und dann

$$F(t,h,v) := \sum_{\ell=1}^{m}\gamma_\ell k_\ell(t,v),$$

also

$$y^{j+1} = y^j + h_j\sum_{\ell=1}^{m}\gamma_\ell k_\ell(t_j,y^j).$$

Man nennt dies ein m**–stufiges Runge–Kutta (RK)–Verfahren**. Typischerweise wird es durch das so genannte **Butcher–Tableau** dargestellt:

$$\begin{array}{r|cccc} 0=\alpha_1 & & & & \\ \alpha_2 & \beta_{21} & & & \\ \alpha_3 & \beta_{31} & \beta_{32} & & \\ \vdots & \vdots & \vdots & \ddots & \\ \alpha_m & \beta_{m1} & \beta_{m2} & \dots & \beta_{m,m-1} \\ \hline & \gamma_1 & \gamma_2 & \dots & \gamma_{m-1} \quad \gamma_m \end{array}$$

Für $m=1,\ \gamma_1=1$, also

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

ergibt sich das Euler–Verfahren.

Es stellt sich sofort die Frage, wie man die Parameter $\alpha_i, \gamma_i, \beta_{ij}$ bestimmt.

Betrachte zunächst den **Spezialfall** $m=2$**:**

$$F(t,h,v) = \gamma_1 f(t,v) + \gamma_2 f(t+\alpha_2 h, v+h\beta_{21}f(t,v)). \tag{3.25}$$

Es wäre von Vorteil, die Parameter $\alpha_2, \gamma_1, \gamma_2, \beta_{21}$ so zu bestimmen, dass ein Verfahren möglichst hoher Ordnung resultiert.

Lemma 3.4.1 *Mittels* (3.25) *ergibt sich ein ESV der Ordnung 2, falls*

(i) $\gamma_1=0,\ \gamma_2=1,\ \alpha_2=1/2,\ \beta_{21}=1/2$, *also*

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

d.h.

$$\left\{\begin{array}{l} F(t,h,y) = f(t+\frac{h}{2}, y+\frac{h}{2}f(t,y)), \\ y^{j+1} = y^j + h_j\, f(t_j+\frac{h_j}{2}, y^j+\frac{h_j}{2}f(t_j,y^j)), \end{array}\right.$$

das sogenannte ***verbesserte Euler–Verfahren****, vgl. Abbildung 3.3.*

(ii) $\gamma_1 = \gamma_2 = 1/2,\ \alpha_2 = \beta_{21} = 1$ *also*

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

das sogenannte ***Euler–Cauchy–Verfahren****, d.h.*

$$\begin{cases} F(t,h,y) = \frac{1}{2}f(t,y) + \frac{1}{2}f(t+h, y+hf(t,y)) \\ y^{j+1} = y^j + \frac{h_j}{2}(f(t_j,y^j) + f(t_j+h_j, y^j + h_j f(t_j,y^j))). \end{cases}$$

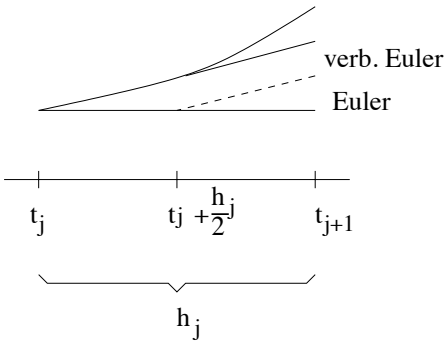


Abb. 3.3: Euler-Verfahren gegenüber verbessertem Euler-Verfahren.

Bemerkung 3.4.2 *Es kann leicht gezeigt werden, dass Ordnung 2 genau dann erreicht wird, wenn*

$$\gamma_1 + \gamma_2 = 1, \quad \gamma_2\alpha_2 = \gamma_2\beta_{21} = \frac{1}{2}.$$

Man kann analog zeigen, dass $p = 3$ *für* (3.25) *nicht möglich ist.*

Es wird nun untersucht, welche (Konsistenz- und) Konvergenzordnung Runge–Kutta–Verfahren allgemein haben
Überprüfe zunächst die Voraussetzung (3.20) (Lipschitz–Bedingung für F).

Lemma 3.4.3 *Erfüllt* f *eine Lipschitz–Bedingung, so auch die Verfahrensfunktion des Runge–Kutta–Verfahrens.*

Beweis. Per Induktion über i bzgl. k_i:

$$\begin{aligned} i=1: \quad \|k_1(t,y) - k_1(t,z)\| &= \|f(t,y) - f(t,z)\| \le L\|y-z\|, \\ i>1: \quad \|k_i(t,y) - k_i(t,z)\| &= \Big\| f\Big(t+\alpha_i h, y + h\sum_{\ell=1}^{i-1}\beta_{i,\ell}k_\ell(t,y)\Big) - f\Big(t+\alpha_i h, z + h\sum_{\ell=1}^{i-1}\beta_{i,\ell}k_\ell(t,z)\Big)\Big\| \\ &\le L\Big\{\|y-z\| + h\underbrace{\sum_{\ell=1}^{i-1}|\beta_{i,\ell}|}_{=\alpha_i}\underbrace{\|k_\ell(t,y) - k_\ell(t,z)\|}_{\le \tilde{L}\|y-z\|}\Big\} \le \tilde{\tilde{L}}\|y-z\|, \end{aligned}$$

per Induktionsvoraussetzung. □

Bemerkung 3.4.4 *Unter obigen Voraussetzungen folgt also aus Satz 3.2.12, dass aus der Konsistenz–Ordnung auch die Konvergenz–Ordnung folgt.*

Wir können uns also auf die leichter zu untersuchende Konsistenzordnung beschränken. Zunächst betrachten wir die Frage, welche Ordnung erreichbar ist.

Zunächst kann man für ein m-stufiges Runge–Kutta–Verfahren mit $f \in C^\infty$ für die Konsistenz–Ordnung $p \in \mathbb{N}$ zeigen, dass $p \le m$ gilt. Dann kann man mit etwas technischem Aufwand folgende Aussage zeigen.

Satz 3.4.5 *Ein Runge–Kutta–Verfahren besitzt für jedes $f \in C^p$ genau dann die Konsistenz–Ordnung p falls gilt*

(i) für $p = 1$: $\sum_{i=1}^{m} \gamma_i = 1$,

(ii) für $p = 2$: zusätzlich $\sum_{i=1}^{m} \gamma_i \alpha_i = \frac{1}{2}$,

(iii) für $p = 3$: zusätzlich $\sum_{i=1}^{m} \gamma_i \alpha_i^2 = \frac{1}{3}$, $\sum_{i,j=1}^{m} \gamma_i \beta_{ij} \alpha_j = \frac{1}{6}$,

(iv) für $p = 4$: zusätzlich $\sum_{i=1}^{m} \gamma_i \alpha_i^3 = \frac{1}{4}$, $\sum_{i,j=1}^{m} \gamma_i \alpha_i \beta_{ij} \alpha_j = \frac{1}{8}$,
$\sum_{i,j=1}^{m} \gamma_i \beta_{ij} \alpha_j^2 = \frac{1}{12}$, $\sum_{i,j,k=1}^{m} \gamma_i \beta_{ij} \beta_{jk} \alpha_k = \frac{1}{24}$.

Beweis. Mittels Taylor–Entwicklung und Koeffizienten–Vergleich. □

Bemerkung 3.4.6 *Viele Beweise für numerische Verfahren zur Lösung gewöhnlicher Differenzialgleichungen gehen nach dem gleichen Schema. Verfahrensfunktion oder der jeweilige Fehler werden in eine Taylor–Reihe entwickelt. Um dann eine bestimmte Ordnung zu garantieren, müssen die Terme niederer Ordnung verschwinden. Durch entsprechenden Koeffizienten–Vergleich erhält man Ordnungsbedingungen.*
Die Ausführung dieser Beweis–Idee für den jeweiligen Fall ist technisch aufwändig, zeigt aber keine neuen Ideen. Daher lassen wir viele dieser Beweise weg und verweisen auf die Literatur.

Beispiel 3.4.7 Will man ein Verfahren der Ordnung $p = 4$ konstruieren, so hat man nach Satz 3.4.5 acht Gleichungen zu erfüllen. Wir wissen, dass für Stufenanzahl m gilt $m \geq p$. Für $m = 4$ hat man

0				
α_2	β_{21}			
α_3	β_{31}	β_{32}		
α_4	β_{41}	β_{42}	β_{43}	
	γ_1	γ_2	γ_3	γ_4

mit den zusätzlichen Bedingungen $\alpha_i = \sum_{\ell=1}^{i-1} \beta_{i,\ell}$.
Man hat also 10 Freiheitsgrade und somit ein unterbestimmtes (nichtlineares) Gleichungssystem zu lösen! Man kann also davon ausgehen, gewisse Freiheiten zu haben.

Gibt man $\alpha_4 = 1$ vor, erhält man im Wesentlichen 2 Lösungen:

0				
$1/2$	$1/2$			
$1/2$	0	$1/2$		
1	0	0	1	
	$1/6$	$1/3$	$1/3$	$1/6$

Klassisches Runge–Kutta–Verfahren (Simpson–Regel)

0				
$1/3$	$1/3$			
$2/3$	$-1/3$	1		
1	1	-1	1	
	$1/8$	$3/8$	$3/8$	$1/8$

Kutta'sche $3/8$–Regel

Hier treten jedoch negative $\beta_{i,\ell}$ auf, was zu Auslöschungseffekten führen kann.

Beispiel 3.4.8 Bekannte Verfahren der Ordnung $p = 3$ sind:

Verfahren von Heun

0			
$1/3$	$1/3$		
$2/3$	0	$2/3$	
	$1/4$	0	$3/4$

Verfahren von Kutta

0			
$1/2$	$1/2$		
1	-1	2	
	$1/6$	$2/3$	$1/6$

Wiederum treten wie bei der $3/8$–Regel negative Werte auf.

Bemerkung 3.4.9 *Es zeigt sich, dass $m \geq p$ für Ordnung p notwendig, aber **nicht** hinreichend ist. Sei $p(m)$ die maximale Ordnung eines m-stufigen Runge–Kutta–Verfahrens; dann erhält man*

m	1	2	3	4	5	6	7	8	9	≥ 9
$p(m)$	1	2	3	4	4	5	6	6	7	$\leq m-2$

Man kann also festhalten, dass man für höhre Ordnung eine wachsende Anzahl von Stufen benötigt.

3.5 Schrittweitensteuerung

Ein naheliegendes Ziel ist das Erreichen einer gewünschten Genauigkeit mit möglichst geringem Aufwand, d.h. mit möglichst wenig (Zeit-)Schritten. Die Steuerung der Schrittweite soll während des Algorithmus („**adaptiv**“) ohne Kenntnis der Lösung erfolgen.

Wir gehen dazu wie folgt vor

1.) Festlegung einer **Zielgenauigkeit**: der globale Fehler soll maximal

$$\varepsilon |t - t_0|$$

für ein vorgegebenes $\varepsilon > 0$ sein.

2.) Der globale Diskretisierungsfehler ergibt sich aus der Summe der lokalen Abbruchfehler. Daraus resultiert die **Forderung:** lokaler Abbruchfehler pro Schritt ist maximal

$$h_j \varepsilon.$$

3.) Um dies zu gewährleisten, braucht man einen **berechenbaren Schätzer** (SCH) für den lokalen Abbruchfehler $c \cdot$ SCH $\leq$ Fehler $\leq C \cdot$ SCH . Wir zeigen hier 2 Möglichkeiten:

- Ausgehend von t_j berechne parallel 2 Schritte mit der Schrittweite $\frac{1}{2}h_j$.
  Die Differenz dient als Schätzer für $h\tau(t, h, y)$.
- Differenz von Verfahren verschiedener Ordnung (sogenannte eingebettete Runge–Kutta–Verfahren).

Algorithmus bei einem Verfahren der Ordnung p: Der grobe Ablauf sieht wie in Abbildung 3.4 aus. Die Abfrage SCH$< 2^{-(p+1)}h\varepsilon$ ergibt sich aus den Überlegungen in Beispiel 3.2.14. Wenn der Schätzer kleiner als $\frac{1}{2}h^{-p}\varepsilon$ (also der Hälfte des für Ordnung p zu erwartenden Fehlers) ist, dann verdoppelt man die Schrittweite.
Nun zur **Fehlerschätzung**, also zur Konstruktion und Analyse von berechenbaren Schätzern.

Definition 3.5.1 *Sei $x(t)$ Lösung der AWA*

$$x'(t) = f(t, x), \quad x(t_0) = x_0. \tag{3.26}$$

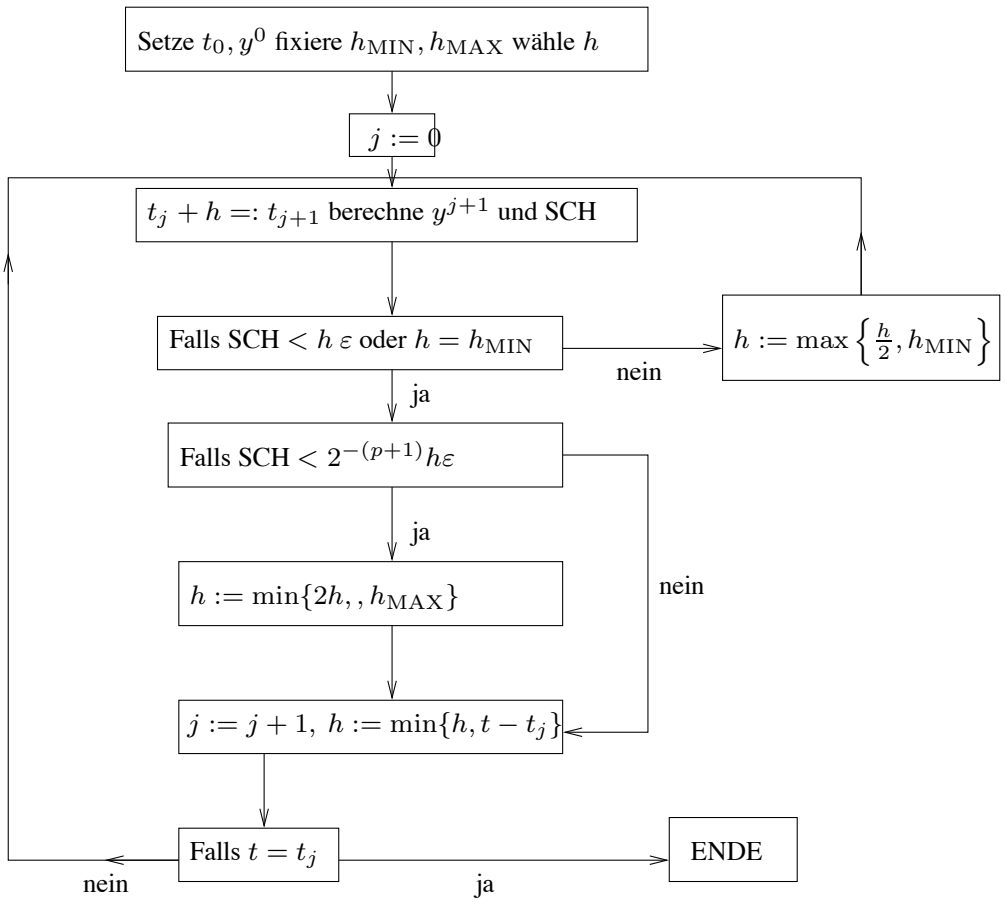


Abb. 3.4: Algorithmus zur Schrittweitensteuerung bei Ordnung p.

*(a) Die Abbildung Φ^{t,t_0} definiert durch $x(t) = \Phi^{t,t_0} x_0$ heißt **Evolution** von* (3.26).

(b) Sei $\Delta = \{t_0, t_1, \ldots, t_n\}$ eine Diskretisierung von (3.26) *auf dem Intervall $[t_0, T]$ und x_Δ sei ein ESV, dann heißt die Abbildung $\Psi^{t,s}$ definiert durch*

$$\Psi^{t_0,t_0} x_\Delta(t_0) = x_0, \quad x_\Delta(t_{j+1}) = \Psi^{t_{j+1},t_j} x_\Delta(t_j), \quad j = 0, \ldots, n-1, \tag{3.27}$$

***diskrete Evolution**.*

Bemerkung 3.5.2 *Mit obiger Definition gilt für den lokalen Abbruchfehler*

$$\delta_h(t, h, y) = \Phi^{t+h,t} y - \Psi^{t+h,t} y.$$

Eine sich daraus entwickelnde Idee ist es nun, ein zweites Verfahren zur Fehlerschätzung (mit anderer Schrittweite oder anderer Ordnung) zu verwenden und dies durch zwei diskrete Evolutionen auszudrücken, also

$$\hat{\delta}_h(t, h, y) = \Phi^{t+h,t} y - \hat{\Psi}^{t+h,t} y.$$

Zu beachten ist dabei, dass diese Größe unbekannt ist. Wir nehmen o.B.d.A. an, dass Ψ genauer ist, d.h.

$$\theta := \frac{|\delta_h(t, h, y)|}{|\hat{\delta}_h(t, h, y)|} < 1. \tag{3.28}$$

Definiere nun den **Fehler–Schätzer**

$$\begin{aligned}[\hat{\delta}_h] &:= \Psi^{t+h,t}y - \hat{\Psi}^{t+h,t}y = \Psi^{t+h,t}y - \Phi^{t+h,t}y + \Phi^{t+h,t}y - \hat{\Psi}^{t+h,t}y \\ &= \hat{\delta}_h(t,h,y) - \delta_h(t,h,y).\end{aligned} \tag{3.29}$$

Lemma 3.5.3 *Der (berechenbare) Term $[\hat{\delta}_h]$ ist ein **effizienter** und **zuverlässiger** Fehlerschätzer für $\hat{\delta}_h$, d.h.*

$$\frac{1}{1+\theta}|[\hat{\delta}_h]| \le |\hat{\delta}_h| \le \frac{1}{1-\theta}|[\hat{\delta}_h]|. \tag{3.30}$$

Beweis. Zunächst gilt mit (3.28) und (3.29)

$$\frac{|\hat{\delta}_h - [\delta_h]|}{|\hat{\delta}_h|} = \frac{|\delta_h|}{|\hat{\delta}_h|} = \theta < 1$$

und mit der Dreiecks–Ungleichung

$$|\hat{\delta}_h| \le |\hat{\delta}_h - [\hat{\delta}_h]| + |[\hat{\delta}_h]| = \theta|\hat{\delta}_h| + |[\hat{\delta}_h]|, \text{ also } |\hat{\delta}_h| \le \frac{|[\hat{\delta}_h]|}{1-\theta},$$

sowie

$$|[\hat{\delta}_h]| \le |[\hat{\delta}_h] - \hat{\delta}_h| + |\hat{\delta}_h| = (1+\theta)|\hat{\delta}_h|.$$

□

Für $\theta \to 0+$ wird aus den Ungleichungen in (3.30) eine Gleichungskette, der Fehlerschätzer ist also **asymptotisch exakt**. Man kann daher mit zwei **beliebigen** Verfahren Fehlerschätzer konstruieren. Eine einfache Möglichkeit ist, dasselbe Verfahren mit $\frac{h_j}{2}$ zu verwenden. Wir wollen nun eine besonders effiziente Möglichkeit vorstellen.
Wir möchten nun einen effizient berechenbaren Fehlerschätzer konstruieren (also mit wenigen Funktionsauswertungen)

- mit halber Schrittweite p zusätzliche Auswertungen,
- mit drittel Schrittweite $2p$ zusätzliche Auswertungen,
- mit viertel Schrittweite p zusätzliche Auswertungen.

Eingebettete Runge–Kutta–Verfahren

Wir betrachten nun zwei Verfahren unterschiedlicher Ordnung und benutzen die Differenz als Fehlerschätzer. Dabei sollen die Berechnungen so effizient wie möglich sein. Sei

- $\Psi^{t+h,t}$ ein Verfahren der Ordnung p
- $\hat{\Psi}^{t+h,t}$ ein Verfahren der Ordnung $p-1$, wobei α, β bei beiden identisch sind, nur die Gewichte $\gamma, \hat{\gamma}$ variieren

α	β
	γ^T
	$\hat{\gamma}^T$

  insbesondere werden keine zusätzlichen Funktionsauswertungen benötigt. Die Schreibweise ist $RK(\gamma,\beta)$ für $\Psi^{t+h,t}$ und $RK(\hat{\gamma},\beta)$ für $\hat{\Psi}^{t+h,t}$. Das zweite Verfahren ist in das erste eingebettet, man schreibt $RKp(p-1)$.

Beispiel 3.5.4 Man kann zunächst zeigen, dass es **kein** RK 4(3) mit $m = 4$ gibt. Das Beispiel $RK4(3)$ mit $m = 5$ lautet

0					
1/2	1/2				
1/2	0	1/2			
1	0	0	1		
1	1/6	1/3	1/3	1/6	
	1/6	1/3	1/3	1/6	0
	1/6	1/3	1/3	0	1/6

Andere Beispiele sind unter anderem das Verfahren von Dorman/Prince (1980-81) DoPri 54, DoPri 87 (vgl. [Deuflhard/Bornemann] 223-225).

Beispiel 3.5.5 Vergleiche die Anzahl der Stützstellen m_Δ und der Funktionsauswertungen $\#f$ beim Dreikörperproblem

	m_Δ	$\#f$
RK 4	117.000	468.000
DoPri 54	346	2.329
DoPri 87	101	1.757

3.6 Steife Anfangswertaufgaben

Wir beschäftigen uns nun mit einer großen Klasse "besonders schwieriger" Anfangswertaufgaben, die in vielen Anwendungen vorkommen. Wir wollen zunächst beschreiben, was "besonders schwierig" heißt.

3.6.1 Kondition einer Anfangswertaufgabe

Die Konvergenzaussagen sind typischerweise asymptotisch

$$\|\varepsilon_\Delta\|_\infty = C \cdot h_\Delta^p$$

(ohne Adaptivität), wobei C in der Regel unbekannt ist. Man hat also folgende Situation

- Man weiss oft nicht, wie man h_Δ wählen muss, um eine gewünschte Genauigkeit tol sicher zu erreichen.
- Gute Abschätzungen für C fehlen, typisch sind
  - worst case–Szenario (oft zu pessimistisch),
  - nur für kleine Funktionenklassen (unrealistisch).

Wir wollen verstehen, wann C groß bzw. klein ist. Dies hat offenbar etwas mit der Kondition einer Anfangswertaufgabe

$$y' = f(t, y), \quad y(t_0) = y^0,$$

bzw. deren numerischer Approximation mittels der Parameter

$$y_h,\ h,\ F$$

zu tun. Sei wieder $\bar{y}_h$ die numerische Lösung zum Anfangswert $y(t_0) = \bar{y}^0$.

Definition 3.6.1 *Die kleinste Zahl κ_h mit*

$$\max_{t\in\Delta} \|y_h(t) - \bar{y}_h(t)\| \le \kappa_h \|y^0 - \bar{y}^0\| + \text{ Terme höherer Ordnung}$$

für $\bar{y}^0 \to y^0$ heißt ***diskrete Kondition*** *des Verfahrens F.*

Definition 3.6.2 *Seien y, z Lösungen der Differenzialgleichung $y' = f(t,y)$ zu Anfangswerten $y(t_0) = y^0$ bzw. $y(t_0) = z^0$. Die kleinste Zahl $\kappa[t_0,t]$ mit*

$$\max_{s\in[t_0,t]} \|y(s) - z(s)\| \le \kappa[t_0,t]\, \|y^0 - z^0\| + \text{ Terme höherer Ordnung} \tag{3.31}$$

heißt ***intervallweise Kondition*** *der Differenzialgleichung.*

Falls F „vernünftig“ ist, sollte

$$\kappa_h \approx \kappa[t_0,T] \tag{3.32}$$

für den Endzeitpunkt T gelten. Falls jedoch

$$\kappa_h \gg \kappa[t_0,T] \tag{3.33}$$

reagiert offenbar Ψ sensibler auf Störungen als Φ. Dies bedeutet, dass h zu groß ist, denn aus der Konvergenz des Verfahrens folgt ja

$$\kappa_h \to \kappa[t_0,T] \text{ für } h \to 0+ .$$

Definition 3.6.3 *Gilt* (3.33)*, so nennt man das AWP (etwas vage)* ***steif****, für* (3.32) ***nicht steif****. Steife AWA sind also Probleme, die mit einem vorgebenen Verfahrenstyp nicht effizient gelöst werden können (da* (3.33) *sehr kleine h, also großen Aufwand, erzwingt).*

Beispiel 3.6.4 Betrachte das skalare Modellproblem

$$y' = \lambda y, \quad y(0) = 1$$

mit der Lösung $y(t) = e^{\lambda t}$ und verwende das Euler–Verfahren auf dem Gitter

$$\Delta = \{t_0 = 0 < t_1 < t_2 < \cdots\}, \quad h_j := t_{j+1} - t_j,$$

also

$$y_h(t_{j+1}) = (1 + h_j\lambda) y_j(t_j)$$

und damit

$$y_h(t_{j+1}) = \prod_{k=0}^{j} (1 + h_k\lambda) y_h(t_0).$$

Also haben wir

$$\max_{t\in\Delta} \|y_h(t) - \bar{y}_h(t)\| \le \underbrace{\left(\max_{0\le j\le n_\Delta - 1} \prod_{k=0}^{j} |1 + h_k\lambda| \right)}_{=\kappa_h} \|y^0 - \bar{y}^0\|$$

Fall 1: $\lambda \geq 0$: Hier gilt in (3.31) für die exakte Lösung

$$\max_{s\in[0,T]} |y^0 e^{\lambda s} - z^0 e^{\lambda s}| = \|y^0 - z^0\| e^{\lambda T} = \|y^0 - z^0\| \cdot \kappa[0,T],$$

andererseits (mit $1 + \lambda h_j \leq e^{\lambda h_j}$)

$$\kappa_h \leq \prod_{k=0}^{n_\Delta - 1} (1 + \lambda h_k) \leq \exp\left\{ \sum_{k=0}^{n_\Delta - 1} \lambda h_k \right\} = e^{\lambda T},$$

also $\kappa_h \leq \kappa[0,T]$, das Probem ist also **nicht-steif**.

Fall 2: $\lambda < 0$: Wie oben sieht man $\kappa[0,T] = 1$ wegen $\lambda < 0$. Für das äquidistante Gitter

$$\Delta = \{t_j = jh : j = 0, \ldots, n_\Delta - 1\}, \quad h_j \equiv h,$$

gilt wegen $\lambda < 0$

$$\kappa_h = \max_{0\leq j\leq n_\Delta - 1} |1 + h\lambda|^{j+1} = \max_{0\leq j\leq n_\Delta - 1} |1 - h|\lambda||^{j+1}$$

- falls $h \leq \frac{2}{|\lambda|}$ gilt $\kappa_h \leq 1 = \kappa[0,T]$
- für $h \gg \frac{2}{|\lambda|}$ gilt hingegen

$$\kappa_h \geq |1 - h|\lambda||^{n_\Delta} \gg 1 = \kappa[0,T].$$

Also bedeutet (3.32) eine (unrealistische) Beschränkung der Schrittweite (besonders für den Fall $|\lambda| \to \infty$), das Problem ist also **steif**!

Bemerkung 3.6.5 *Steife Differenzialgleichungen treten u.a. häufig bei chemischen Reaktionen und auch in der Biologie sowie Medizin auf.*

3.6.2 Stabilität

Falls die diskrete Kondition deutlich größer als die intervallweise Kondition ist, kann dies also nur daran liegen, dass das Verfahren ungeeignet ist.
Die Frage, welche Verfahren für steife AWA geeignet sind, führt auf die Untersuchung der Stabilitätseigenschaften. Zunächst wieder einige Wiederholungen.

Definition 3.6.6 *Die lineare Iteration $x_{n+1} = Ax_n$, $A \in \mathbb{C}^{n\times n}$, heißt **stabil**, falls*

$$\sup_{n\geq 1} \|A^n\| < \infty,$$

*sie heißt **asymptotisch stabil**, falls*

$$\lim_{n\to\infty} \|A^n\| = 0.$$

Satz 3.6.7 *Die lineare Iteration $x_{n+1} = Ax_n$ ist genau dann stabil, wenn für den Spektralradius gilt*

$$\rho(A) = \max_{\lambda\in\sigma(A)} |\lambda| \leq 1$$

und alle Eigenwerte $\lambda \in \sigma(A)$ mit $|\lambda| = 1$ einfach sind. Sie ist genau dann asymptotisch stabil, wenn für den Spektralradius $\rho(A) < 1$ gilt.

Beweis. [Deuflhard/Bornemann, S. 118-119]. □

Nun zum numerischen Verfahren beschrieben durch die diskrete Evolution

$$\Psi^h \equiv \Psi^{t+h,t}$$

wie in (3.27). Dann heißt der Ausdruck

$$h_c := \sup\{\bar{h} > 0 : \text{ die Rekursion } x_{n+1} = \Psi^h x_n \text{ ist stabil für } 0 < h \leq \bar{h}\}$$

charakteristische Schrittweite (die Größe entscheidet über die Steifheit).

Nochmal zurück zu **Beispiel 3.6.4:** Wir brauchen offenbar eine Verfahrensfunktion, die in $|h\lambda|$ fällt. Idee (wenn $h|\lambda|$ groß wird): Entwickle F aus dem Euler–Verfahren mittels geometrischer Reihe:

$$1 + h\lambda = \frac{1}{1 - h\lambda} + \mathcal{O}(|h\lambda|^2), \quad |h\lambda| < 1.$$

Für das Verfahren

$$y_h(t_{j+1}) = \frac{1}{1 - h_j\lambda} y_h(t_j)$$

gilt für äquidistante Stützstellen

$$y_h(t_{j+1}) = \left(\frac{1}{1 - h\lambda}\right)^{j+1} y_h(t_0)$$

und damit für **alle** $h > 0$: $\kappa_h = \kappa[0, T] = 1$.

Offenbar macht es also Sinn, als Verfahrensfunktionen rationale Funktionen $R : \mathbb{C} \to \bar{\mathbb{C}} = \mathbb{C} \cup \{\infty\}$ der Form

$$R(z) = \frac{P(z)}{Q(z)}$$

mit zwei teilerfremden Polynomen P, Q zuzulassen.

Bemerkung 3.6.8 *Die Lösung des Modellproblems $y' = Ay$ lautet ja $y(t) = e^{At}y^0$ mit der Matrixexponentialfunktion: $\Phi^t = \exp(tA)$. Nach obigem betrachten wir nun rationale Approximationen:*

$$\Psi^t = R(tA).$$

Bemerkung 3.6.9 *(a) In Erweiterung von Satz 3.6.7 ist Ψ^t genau dann stabil, wenn $\rho(\Psi^t) \leq 1$ gilt und die Eigenwerte λ mit $|\lambda| = 1$ einfach sind.*

(b) Man kann weiter zeigen, dass

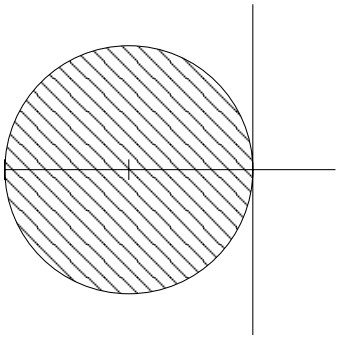
$$\rho(\Psi^t) = \max_{\lambda \in \sigma(A)} |R(t\lambda)|.$$

Definition 3.6.10 *Die Menge*

$$\mathcal{S} := \{z \in \mathbb{C} : |R(z)| \leq 1\}$$

heißt ***Stabilitätsgebiet*** *von R.*

Ziel ist es nun also, $t\lambda \in \mathcal{S}$ zu erreichen.



Beispiel 3.6.11 Für das explizite Euler–Verfahren gilt $R(z) = 1 + z$ und

$$\mathcal{S} = \{z \in \mathbb{C} : |1 + z| \leq 1\} = \overline{B_1(-1)}$$

Andere Beispiele sind in Abbildung 3.5 dargestellt.

Wir können nun entsprechende Stabilitätsbegriffe definieren. Die erste Idee ist, dass $\mathcal{S}$ zumindest das Stabilitätsgebiet der Exponentialfunktion

$$\{z \in \mathbb{C} : |\exp(z)| \leq 1\} = \{z \in \mathbb{C} : Re(z) \leq 0\} = \mathbb{C}_-$$

enthalten sollte, (Dahlquist 1925 - 2005).

Definition 3.6.12 (Dahlquist, 1963) *Ein Verfahren heißt A-**stabil**, falls*

$$\mathbb{C}_- \subset \mathcal{S}.$$

Satz 3.6.13 (Dahlquist, 1925–2005) *Das Verfahren $\Psi^t = R(tA)$ ist genau dann für alle $t > 0$ und für alle (asymptotisch) stabilen AWA $y' = Ay$, $y(0) = y^0$ (asymptotisch) stabil, wenn das Verfahren A-stabil ist.*

Beweis. [Deuflhard/Bornemann, S. 294] □

Beispiel 3.6.14 Für $R(z) := \frac{1}{1-z}$ wie oben gilt

$$\mathcal{S} = \{z \in \mathbb{C} : \frac{1}{|1 - z|} \leq 1\} = \{z \in \mathbb{C} : |1 - z| \geq 1\} = \mathbb{C}\backslash B_1(1),$$

also ist $R(\cdot)$ A-stabil.

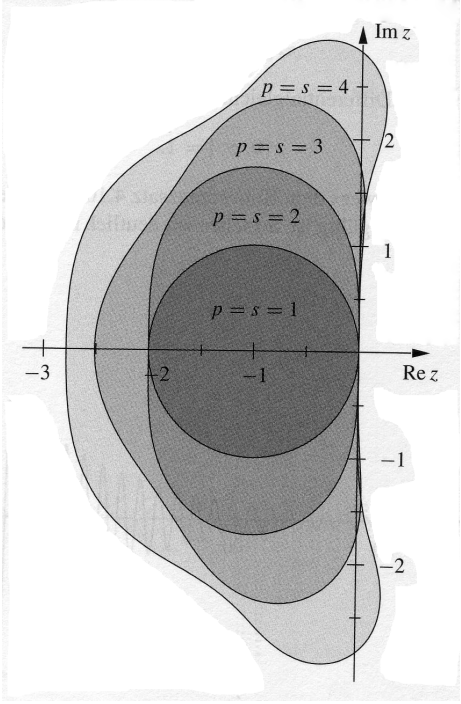
$$y' = \lambda y, y'(0) = 1,\ y^{j+1} = \frac{1}{1 - h\lambda} y^j \quad \Longleftrightarrow \quad \begin{aligned} y^{j+1} - h\lambda y^{j+1} &= y^j \\ y^{j+1} &= h\lambda y^{j+1} + y^j \\ &= y^j + hf(y^{j+1}) \rightsquigarrow \text{ impl. Euler.} \end{aligned}$$

Dies entspricht offenbar dem **impliziten** Euler–Verfahren.

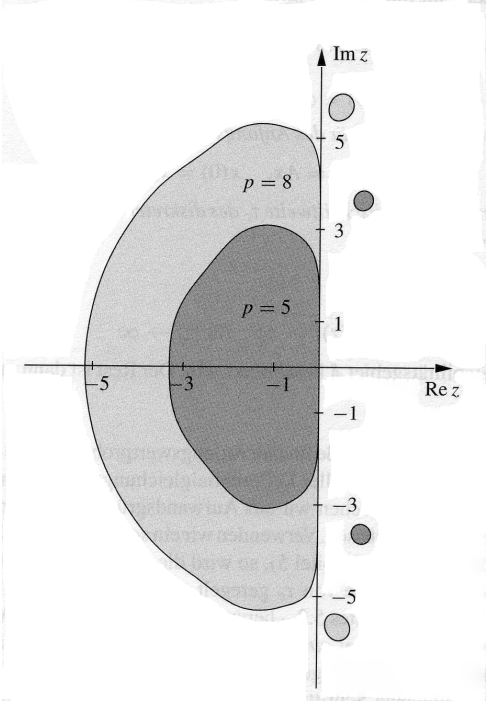
Satz 3.6.15 *(Dahlquist) Ein lineares A-stabiles Verfahren ist stets implizit.*

Beweis. [Dahl] □

Also scheinen implizite Verfahren für steife Probleme geeignet zu sein.



(a) Abbildung aus [Deuflhard/Bornemann]: Stabilitätsgebiete der einfachen expliziten Runge–Kutta–Verfahren.



(b) Abbildung aus [Deuflhard/Bornemann]: Stabilitätsgebiete der Verfahren von J.R. Dormand und P.J. Prince.

Abb. 3.5: Weitere Beispiele für Stabilitätsgebiete.

3.6.3 Implizite Runge–Kutta–Verfahren

Wir haben nun hinreichend motiviert, warum man implizite Verfahren benötigt. Die Runge–Kutta–Verfahren hatten wir als wichtige Klasse bei expliziten Verfahren kennengelernt. Nun also zu deren impliziter Variante. Eine Idee wäre, (3.24) durch

$$k_j(t,h,y) := f\left(t+\alpha_j h, y + h\sum_{l=1}^{m}\beta_{jl}k_l(t,h,y)\right), \quad j=1,\dots,m, \tag{3.34}$$

zu ersetzen, wobei die innere Summe nun also bis m läuft. Also sind die $k_j(t,h,y)$ Lösungen des nichtlinearen Gleichungssystems (3.34). Man muss sich nun zunächst die Frage stellen, wann Lösungen von (3.34) existieren.

Satz 3.6.16 *Es gelte*

$$\|f(t,y)-f(t,z)\| \le L\|y-z\|, \quad \forall (t,y),(t,z) \in I\times\mathbb{R}^n. \tag{3.35}$$

Falls für alle Schrittweiten h die Bedingung

$$q := Lh\left(\max_{j=1,\dots,m}\sum_{l=1}^{m}|\beta_{j,l}|\right) < 1 \tag{3.36}$$

gilt, dann existieren für alle $(t,y)\in I\times\mathbb{R}^n$ eindeutige Lösungen $k_i(t,h,y)$, $i=1,\dots,m$.

Beweis. Mit dem Banach'schen Fixpunktsatz aus (3.34). □

Bemerkung 3.6.17 *(a) Es genügt, die Lipschitz–Bedingung* (3.35) *nur lokal in einer Umgebung des Lösungsgraphen zu fordern.*

(b) Die Bedingung (3.36) *besagt, dass h „hinreichend klein“ sein muss.*

(c) In der Praxis löst man (3.34) *typischerweise mit Newton.*

Es stellen sich nun folgende zwei Fragen, welche Konsistenzordung diese Verfahren haben und wie groß kann man die Ordnung (maximal) machen. Diese Fragen möchten wir nun klären.

Bemerkung 3.6.18 *Für die Konsistenz (expliziter und impliziter) RK–Verfahren ist*

$$\gamma_1 + \cdots + \gamma_m = 1$$

notwendig.

Satz 3.6.19 *Es gelte die Lipschitz–Bedingung* (3.35) *für f. Wählt man zu paarweise verschiedenen $\alpha_j \in [0,1]$, $j = 1, \ldots, m$, die Parameter γ_j, $j = 1, \ldots, m$ und β_{il}, $i, l = 1, \ldots, m$, so dass für ein r gilt*

$$\sum_{k=1}^{m} \beta_{jk} \alpha_k^l = \frac{\alpha_j^{l+1}}{l+1}, \quad l = 0, \ldots, r-1, \quad j = 1, \ldots, m, \tag{3.37}$$

$$\sum_{k=1}^{m} \gamma_k \alpha_k^l = \frac{1}{l+1}, \quad l = 0, \ldots, r-1, \tag{3.38}$$

so ist das entsprechende RK–Verfahren konsistent von der Ordnung $p = r$.

Beweis. Einsetzen der Exaktheitsbedingungen und Nachrechnen. □

Bemerkung 3.6.20 *Die Bedingungen* (3.37), (3.38) *besagen gerade, dass die Quadraturformeln mit den Gewichten $\gamma_1, \ldots, \gamma_m$ auf $[0,1]$ und den Gewichten β_{jk} auf $[0, \alpha_j]$ exakt vom Grade $r-1$ sind (dies entspricht Exaktheitsbedingungen für Newton–Cotes–Formeln).*

Nun zur zweiten Frage, wie groß man die Ordnung (maximal) machen kann.

Bemerkung 3.6.21 *Man kann die Ordnung $p = 2m$ erreichen, indem man die α_i als die **Gauß–Knoten** wählt (vgl. Grigorieff, 1977). Diese Ordnung ist maximal.*

Beispiel 3.6.22 (a) Gauß–Form: Für $m = 1$, $p = 2$ ergibt sich $\begin{array}{c|c} 1/2 & 1/2 \\ \hline & 1 \end{array}$, also

$$k_1(t,h,y) = f\left(t + \tfrac{h}{2}, y + \tfrac{h}{2}, k_1(t,h,y)\right),$$
$$y^{j+1} = y^j + h_j k_1.$$

(b) Gauß–Form: Für $m = 2$, $p = 4$ erhält man

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

(c) Weitere Beispiele: Radon-Form, Lobatto-Form.

3.7 Mehrschrittverfahren für Anfangswertaufgaben

Wie schon in den bereits vorhergehenden Kapiteln beschrieben, sind die wesentlichen Vorteile der Einschrittverfahren zum einen die einfache Umsetzung und zum anderen ist eine einfache Schrittweitensteuerung möglich. Ein wesentlicher Nachteil ist aber, dass sehr viele Funktionsauswertung notwendig sind, um ein Verfahren hoher Ordnung realisieren zu können. Eine Alternative bieten sogenannte Mehrschrittverfahren (MSV), die wir nun beschreiben.

3.7.1 Konsistenz bei Mehrschrittverfahren

Seien zunächst $t_0, \ldots, t_k$ paarweise verschieden. Wir verfolgen nun den Ansatz, die Lösung der AWA durch ein Interpolationspolynom anzunähern. Für das Interpolationspolynom $P(y|t_0, \ldots, t_k)$ an y bzgl. $t_0, \ldots, t_k$ gilt dann näherungweise

$$P'(y|t_0, \ldots, t_k)(\bar{t}) \approx y'(\bar{t}) \text{ (numerische Diff.)}$$

und auf der anderen Seite

$$P'(y|t_0, \ldots, t_k)(\bar{t}) = \sum_{i=0}^{k} a_i(\bar{t}) y(t_i)$$

mit der Lagrange–Darstellung

$$a_i(\bar{t}) = \frac{d}{dt} \left(\prod_{\substack{\ell=0 \\ \ell \neq i}}^{k} \frac{t - t_\ell}{t_i - t_\ell} \right)_{|t=\bar{t}}.$$

Setze dies nun in die AWA ein

$$y' = f(t, y), \quad y(t_0) = y^0, \tag{3.39}$$

dann erhält man die folgende Form.

Definition 3.7.1 *Ein Verfahren der Form*

$$\begin{cases} \sum\limits_{\ell=0}^{k} a_\ell y^{j+\ell} = hF(t_j, \ldots, t_{j+k}, h, y^j, \ldots, y^{j+k}) \\ t_j = t_0 + jh, \quad j = 0, 1, 2, \ldots \end{cases} \tag{3.40}$$

heißt ***Mehrschrittverfahren (k-Schrittverfahren)*** *und F heißt* ***Verfahrensfunktion****. Das MSV* (3.40) *heißt* ***explizit****, falls F nicht von y^{j+k} abhängt, sonst* ***implizit****.*

Bemerkung 3.7.2 *(a) Offenbar benötigt man für* (3.40) *k Startwerte $y^0, y^1, \ldots, y^{k-1}$ („Anlaufstück“). Diese berechnet man z.B. über ein ESV.*

(b) ***Lineare k-Schrittverfahren*** *haben die Form*

$$F(t_j, \ldots, t_{j+k}, h, y^j, \ldots, y^{j+k}) = \sum_{\ell=0}^{k} b_\ell f^{j+\ell} \tag{3.41}$$

mit $f^\ell := f(t_\ell, y^\ell)$, d.h.

$$\sum_{\ell=0}^{k} a_\ell y^{j+\ell} = h \sum_{\ell=0}^{k} b_\ell f^{j+\ell}. \tag{3.42}$$

(c) *Wie bei ESV (RK) kann man im impliziten Fall (für kleine h) mit einem Fixpunkt-Argument die Lösbarkeit der entsprechenden Gleichungssysteme zeigen.*

Definition 3.7.3 *Der Ausdruck*

$$\tau(t,h,y) := \frac{1}{h}\sum_{\ell=0}^{k} a_\ell y(t-(k-\ell)h) - F(t-kh,\dots,t,h,y(t-kh),\dots,y(t))$$

*für $t-kh \geq t_0$ heißt **Konsistenzfehler (lokaler Diskretisierungsfehler)** und $h\tau(t,h,y)$ heißt **lokaler Abbruchfehler**.*

Definition 3.7.4 *Ein MSV heißt **konsistent (von der Ordnung p)** mit der AWA* (3.39)*, falls für jede exakte Lösung gilt*

$$\begin{cases} \|y(t_j)-y^j\| = o(1)\ (\mathcal{O}(h^p)), & j=0,\dots,k-1, \\ \max\limits_j \|\tau(t_j,h_j,y)\| = o(1)\ (\mathcal{O}(h^p)), & h\to 0+, \end{cases}$$

für alle genügend glatten f.

Beispiel 3.7.5 Betrachte die Taylor–Entwicklungen

$$y(t\pm h) = y(t) \pm hy'(t) + \frac{h^2}{2}y''(t) \pm \frac{h^3}{6}y'''(t) + \mathcal{O}(h^4),$$

also wegen $y'(t) = f(t,y)$

$$y(t+h) - y(t-h) - 2hf(t,y) = \frac{h^3}{3}y'''(t) + \mathcal{O}(h^4),$$

d.h.,

$$\frac{1}{h}(y(t+h)-y(t-h)) - 2f(t,y) = \frac{h^2}{3}y'''(t) + \mathcal{O}(h^3).$$

Das Verfahren

$$y^{j+1} - y^{j-1} = 2hf(t_j,y^j) \tag{3.43}$$

hat also die Ordnung 2. Offenbar entspricht (3.43) der Mittelpunktregel.

Bei MSV impliziert die Konsistenz **nicht** die Konvergenz (mehr dazu später). Nun zunächst einige Beispiele.

Eine nun offensichtliche Idee ist, in der folgenden Gleichung $(0 \leq r \leq k, 0 \leq \ell \leq r)$

$$y(t_{j+k}) - y(t_{j+r-\ell}) = \int_{t_{j+r-\ell}}^{t_{j+k}} f(s,y(s))ds$$

den Integranden f durch den Interpolanden zu ersetzen

$$P_{r,j}(s) = P(f|t_j,\dots,t_{j+r})(s) \in \mathcal{P}_r.$$

Übliche Werte sind: $r = k, k-1$, $\ell = 0,1,2$. Man erhält also

- Integrationsbereich: $[t_{j+r-\ell}, t_{j+k}]$ ($k+\ell-r$ Stützstellen)
- Interpolationsbereich: $[t_j,\dots,t_{j+r}]$ (r Stützstellen)

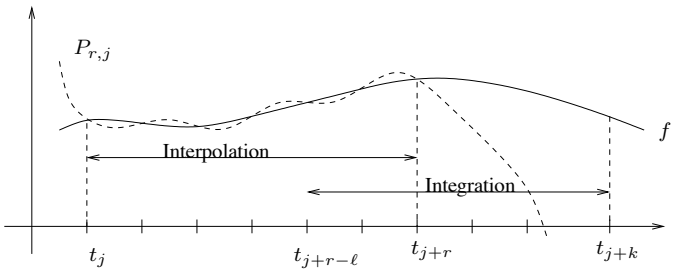


Abb. 3.6: Visualisierung des Interpolations- und Integrationsbereich.

Wir wollen diese Verfahren ab sofort mit MSV (r, ℓ) bezeichnen.

Typische Verfahrensklassen werden im folgenden Beispiel aufgelistet.

Beispiel 3.7.6 (a) **Adams Bashforth:** ($\ell = 0$, MSV($k-1$,0)):

$$y^{j+k} = y^{j+k-1} + h \sum_{i=0}^{k-1} \beta_i^{(k-1,0)} f^{j+i}, \text{ speziell}$$

- $r = k-1 = 0$ $(k = 1)$: $y^{j+1} = y^j + hf^j$ (Euler)
- $r = k-1 = 1$ $(k = 2)$: $y^{j+2} = y^{j+1} + \frac{h}{2}(3f^{j+1} - f^j)$

(b) **Nyström:** ($\ell = 1$, MSV $(k-1$,1):

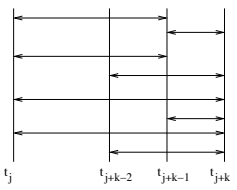
$$y^{j+k} = y^{j+k-2} + h \sum_{i=0}^{k-1} \beta_i^{(k-1,1)} f^{j+i} \text{ , speziell}$$

- $r = k-1 = 0$ $(k = 1)$: $y^{j+1} = y^{j-1} + 2hf^j$ (Mittelpunktregel)

(c) **Adams–Moulton:** (MSV(k,1)): $y^{j+k} = y^{j+k-1} + h \left\{ \sum_{i=0}^{k-1} \beta_i^{(k,1)} f^{j+i} + \beta_k^{(k,1)} f^{j+k} \right\}$

- $r = k = 0$: $y^j = y^{j-1} + hf^j$ (implizites Euler)
- $r = k = 1$: $y^{j+1} = y^j + \frac{h}{2}(f^{j+1} - f^j)$ (implizite Trapezregel)
- $r = k = 2$: $y^{j+2} = y^{j+1} + \frac{h}{12}\{5f^{j+2} + 8f^{j+1} - f^j\}$

r	ℓ	Name	Art
$k-1$	0	Adams Bashforth	Extrapolation explizit
$k-1$	1	Nyström	Extrapolation explizit
k	1	Adams–Moulton	Interpolation implizit
k	2	Milne–Simpson	Interpolation implizit



Satz 3.7.7 *Die MSV vom Typ (r, ℓ) sind konsistent für alle $f \in C^{r+1}(\mathcal{U})$ (wobei $\mathcal{U}$ eine Umgebung der Lösung ist) von der Ordnung*

$$p = r + 1.$$

Beweis. Mit der Restglied–Darstellung der Polynom–Interpolation. □

Bemerkung 3.7.8 *Für Adams–Bashforth und Nyström gilt also $p = k$, für Adams–Moulton und Milne–Simpson gilt $p = k + 1$.*

Man sucht natürlich allgemeine Konsistenz-Kriterien.

Definition 3.7.9 *Zum MSV* (3.40) *heißt*

$$\rho(z) := \sum_{\ell=0}^{k} a_\ell z^\ell$$

das ***1. charakteristische Polynom****.*

Lemma 3.7.10 *Sei $y \in C^1$ Lösung von* (3.39)*. Dann ist das MSV* (3.40) *genau dann konsistent ($p = 1$), wenn*

$$y(t)\rho(1) = 0, \tag{3.44}$$

$$y_h^j \to y^0 = y(t_0), \quad j = 0, \ldots, k-1, \quad h \to 0+, \tag{3.45}$$

$$\lim_{h\to 0} \left\{ \max_{k\le j\le m} |F(t_j, \ldots, t_{j+k}, h, y(t_j), \ldots, y(t_{j+k})) - \rho'(1) f(t_j, y(t_j))| \right\} = 0. \tag{3.46}$$

Beweis. Taylor–Entwicklung von F und Koeffizienten–Vergleich. □

Die Gleichungen (3.44) - (3.46) sind etwas unhandlich. Für lineare MSV (3.41) kann man einige Aussagen mehr treffen.

Definition 3.7.11 *Die Funktion*

$$\sigma(z) := \sum_{\ell=0}^{k} b_\ell z^\ell$$

heißt ***2. charakteristisches Polynom*** *des linearen MSV* (3.41)*.*

Lemma 3.7.12 *Ein lineares MSV* (3.41) *ist genau dann konsistent (also $p = 1$) für alle glatten f, wenn*

$$\begin{aligned} &y_h^j \to y^0 = y(t_0), \quad j = 0, \ldots, k-1, \quad h \to 0+, \\ &\rho(1) = 0, \quad \rho'(1) = \sigma(1). \end{aligned} \tag{3.47}$$

Beweis. Taylor–Entwicklung von F in (3.46). □

Satz 3.7.13 *Für das lineare MSV* (3.41) *sind folgende Aussagen äquivalent:*

(a) Es gilt

$$\sum_{i=0}^{k} (i^j a_i - j i^{j-1} b_i) = 0 \quad \forall j = 0, \ldots, p$$

vgl. (3.47)*, dort für $p = 1$.*

(b) Das MSV ist für jedes $f \in C^p(\mathcal{U})$ konsistent von der Ordnung p.

(c) Die Konsistenzordnung beträgt p für die AWA

$$y' = y, \quad y(t_0) = 1.$$

(d) Die Konsistenzordnung ist p für eine Klasse von AWA, deren Lösungen den $\mathcal{P}_p$ aufspannen.

Bemerkung 3.7.14 *Die maximale Ordnung ist $p = 2k$ und man kann zeigen, dass diese auch erreichbar ist.*

3.7.2 Asymptotische Stabilität

Eine unbeantwortete Frage bisher ist, ob es Konvergenzaussagen gibt.

Als Vorüberlegung dazu, betrachte die spezielle AWA

$$y' = 0, \quad y(0) = 1 \quad (\text{d.h. } f \equiv 0,\ y \equiv 1).$$

Dann ergibt sich für die numerische Lösung aus (3.42)

$$\sum_{\ell=0}^{k} a_\ell\, y^{j+\ell} = 0, \quad j = 0, 1, 2, \ldots, \tag{3.48}$$

die sogenannte **lineare homogene Differenzengleichung k-ter Ordnung**. Offenbar muss die Folge $\{y^j\}$ zumindest beschränkt sein, damit das MSV konvergiert.

Bemerkung 3.7.15 *Die Lösungen von* (3.48) *sind alle genau dann beschränkt, wenn ρ die* ***Wurzel-Bedingung*** *(WZB) erfüllt*

$$\begin{cases} \rho(z) = 0 \Rightarrow |z| \le 1 \\ \rho(z) = 0,\ |z| = 1 \Rightarrow z \ \textit{ist} \ \textbf{\textit{einfache}} \ \textit{Nullstelle} \end{cases} \tag{3.49}$$

Beweis. Differenzialgleichungen, Wronski-Detederminante. □

Nun zur Konvergenz. Dazu schreiben wir das allgemeine MSV in der Form

$$\begin{cases} \sum\limits_{i=0}^{k} a_i y(t_{j+i}) = hF(t_j, \ldots, t_{j+k}, h, y^j, \ldots, y^{j+k}) + h\varepsilon^{j+k}(h), \\ y^j = y(t_j) + \varepsilon^j(h), \qquad j = 0, \ldots, k-1, \\ t_j = t_0 + jh, \qquad t = t_m = t_0 + mh. \end{cases} \tag{3.50}$$

Linearität wird hier **nicht** vorausgesetzt.

Definition 3.7.16 *Das MSV* (3.50) *heißt konvergent, falls*

$$\lim_{\substack{m\to\infty \\ mh=t}} \max_{0\le j\le m} \|y^j - y(t_j)\| = 0$$

für alle Startwerte $j = 0, \ldots, k-1$, mit

$$\|\varepsilon^j(h)\| = o(1), \quad h \to 0+, \quad j \le m \to \infty.$$

Mit einigem Aufwand (Störungstheorie) zeigt man folgende Aussage.

Satz 3.7.17 *Angenommen, es gilt: Falls $f(t,u) = 0\ \forall (t,u) \in \mathcal{U}$, dann verschwinde dort auch F. Dann gilt*

(a) Ist das MSV (3.50) *konvergent, dann ist die WZB* (3.49) *erfüllt.*

(b) Das MSV sei konsistent und F sei zusätzlich stetig. Dann ist das MSV für alle $f \in C^1$ konvergent genau dann, wenn die WZB gilt.

Bemerkung 3.7.18 *(a) Man kann kurz sagen „Konsistenz + Stabilität ⇒ Konvergenz“.*

(b) Für lineare MSV gilt auch die Umkehrung.

(c) Unter der Voraussetzung der Stabilität (WZB) folgt aus der Konsistenzordnung auch die entsprechende Konvergenzordnung.

Bemerkung 3.7.19 *In Abschnitt 3.7.1 hatten wir Verfahren über Quadratur konstruiert. Man kann alternativ numerische Differenziation verwenden. Dies führt auf die sogenannten* ***BDF–Verfahren****. Eine weitere große Klasse wäre das sogenannte Prädiktor-Korrektor-Verfahren.*

4 RANDWERTPROBLEME

Wir kennen bereits ein Modellproblem für Randwertprobleme (RWP), die schwingende Saite

$$-u''(x) = f(x), \quad x \in (0,1), \quad u(0) = u(1) = 0.$$

Dies ist ein so genanntes raumartiges Problem für $x \in (0,1)$. Wir betrachten zunächst Randwertprobleme in der Zeit. Wir betrachten hier Probleme der Form

$$\begin{cases} y'(t) = f(t, y(t)), & y = (y_1, \ldots, y_n)^T \\ & f(t,y) = (f_i(t, y_1, \ldots, y_n))_{i=1,\ldots,n}^T \end{cases} \tag{4.1}$$

mit den allgemeinen Randbedingungen

$$r(y(a), y(b)) = 0 \text{ und } r(u,v) = (r_i(u_1, \ldots, u_n, v_1, \ldots, v_n))_{i=1,\ldots,n}^T. \tag{4.2}$$

4.1 Typen von Randwertproblemen

Die Formulierung (4.2) ist sehr allgemein. Aus diesem Grund möchten wir nun einige wichtige Spezialfälle aufführen

Lineare Randbedingungen:

$$Ay(a) + By(b) = c \text{ mit } A, B \in \mathbb{R}^{n \times n},\ c \in \mathbb{R}^n. \tag{4.3}$$

Separierte Randbedingungen:

$$A_1 y(a) = c_1, \quad B_2 y(b) = c_2,$$

d.h. (4.3) mit $A = \binom{A_1}{0}$, $B = \binom{0}{B_2}$, $c = \binom{c_1}{c_2}$.

Bemerkung 4.1.1 *Mit $B_2 \equiv 0$ erhält man eine AWA, also sind AWA spezielle Randwertprobleme.*

Mehrpunkt-Bedingungen:
Können beispielsweise in folgender Form auftreten

$$y\left(\frac{a+b}{2}\right) = \frac{1}{2}(y(a) + y(b)).$$

Integral-Bedingungen:
Können beispielsweise in folgender Form auftreten

$$\int_a^b y(x)dx = 1.$$

Periodische Randbedingungen:

$$y(a) = y(b), \quad y'(a) = y'(b),$$

oder

$$y(a) = -y(b), \quad y'(a) = -y'(b).$$

Definition 4.1.2 *Ist f in* (4.1) *affin-linear und die Randbedingungen* (4.2) *linear in $y(a)$ und $y(b)$, dann nennt man das Randwertproblem* ***linear****.*

Wir möchten nun einige Beispiele geben, welche uns für das Verständnis und anschließende genauere Betrachtung von RWP helfen.

Beispiel 4.1.3 Die Differenzialgleichung $\omega'' + \omega = 0$ für $\omega : \mathbb{R} \to \mathbb{R}$ lässt sich wie üblich transformieren:

$$\left.\begin{array}{l} y_1(t) := \omega(t) \\ y_2(t) := \omega'(t) \end{array}\right\} \Rightarrow \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}' = \begin{bmatrix} y_2 \\ -y_1 \end{bmatrix}, \quad y' = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} y, \qquad y = (y_1, y_2)^T.$$

Die allgemeine Lösung lautet: $\omega(t) = c_1 \sin t + c_2 \cos t$, $c_1, c_2 \in \mathbb{R}$.

- $\omega(0) = 0,\ \omega\left(\frac{\pi}{2}\right) = 1 \rightsquigarrow \omega(t) = \sin t$ ist eindeutige Lösung.
- $\omega(0) = 0,\ \omega(\pi) = 0 \rightsquigarrow \omega(t) = c_1 \sin t$ ist Lösung $\forall c_1 \in \mathbb{R}$.
- $\omega(0) = 0,\ \omega(\pi) = 1 \rightsquigarrow$ es existiert **keine** Lösung.

Eine offensichtliche Konsequenz aus Beispiel 4.1.3 ist, dass es keinen allgemeinen Existenz- und Eindeutigkeits–Satz geben kann.

Beispiel 4.1.4 Betrachte das **Differenzialgleichungs–Eigenwertproblem**

$$y' = f(t, y, \lambda), \quad r(y(a), y(b), \lambda) = 0 \tag{4.4}$$

(Bsp.: $\omega'' + \lambda\omega = 0$), wobei die Eigenwerte λ so zu bestimmen sind, dass die jeweilige Lösung eindeutig ist.
Ein erster Ansatz wäre $y_{n+1}(t) := \lambda$, d.h., $y'_{n+1}(t) = 0$ zu setzen. Dann ist (4.4) äquivalent zu

$$\bar{y}' = \bar{f}(t, \bar{y}), \quad \bar{r}(\bar{y}(a), \bar{y}(b)) = 0$$

mit

$$\bar{y} := \begin{bmatrix} y \\ y_{n+1} \end{bmatrix}, \quad \bar{f}(t, \bar{y}) := \begin{bmatrix} f(t, y, y_{n+1}) \\ 0 \end{bmatrix}$$

und

$$\bar{r}(u_1, \ldots, u_n, u_{n+1}, v_1, \ldots, v_n v_{n+1}) := r(u_1, \ldots, u_n, v_1, \ldots, v_n, v_{n+1}).$$

Beispiel 4.1.5 (Randwertprobleme mit freiem Rand) Zu $y' = f(t, y)$ und festem a ist eine Funktion gesucht, die $n + 1$ Randbedingungen

$$r(y(a), y(b)) = 0, \quad r(u, v) := (r_1(u, v), \ldots, r_{n+1}(u, v))^T$$

erfüllt, wobei b hier **unbekannt** ist.
Solche Probleme tauchen z.B. bei Schmelzvorgängen, Simulation von Lawinen, Diffusion durch poröse Medien oder Bewertung Amerikanischer Optionen auf. Wie oben wählt man als Ansatz

$$y_{n+1} := b - a,$$

d.h., man führt eine neue Variable x ein durch die Definition

$$x := \frac{t - a}{y_{n+1}}, \quad 0 \le x \le 1,$$

ein, also für $t = b$ folgt $x = 1$. Zunächst gilt

$$y'_{n+1} = \frac{d}{dx} y_{n+1} = \frac{d}{dx}(b - a) = 0.$$

Setze nun $z(x) := y(a + xy_{n+1})$, $0 \le x \le 1$, dann gilt

$$\begin{aligned} z'(x) = \frac{d}{dx} z(x) &= \left(\frac{d}{dt} y(a + x \cdot y_{n+1}) \right) y_{n+1} \\ &= y_{n+1} f(a + xy_{n+1}, z(x)) =: \tilde{f}(x; z(x)) \end{aligned}$$

aufgrund der Differenzialgleichung. Für die Randbedingungen gilt

$$z(0) = y(a), \quad z(1) = y(b),$$

also erhält man ein Randwertproblem mit festem Rand, d.h. auf $[0, 1]$. Man kann also freie Randwertprobleme in 1D (!) in gewöhnlich Randwertprobleme umschreiben. Dies gilt **nicht** in höheren Dimensionen!

4.2 Das einfache Schieß–Verfahren

Das einfache Schieß–Verfahren basiert auf der intuitiven Idee, das Randwertproblem auf eine Folge von AWA zurück zu führen.

Betrachte das Randwertproblem

$$y''(t) = f(t, y, y'), \quad y(a) = \alpha,\ y(b) = \beta \tag{4.5}$$

und die AWA

$$y''(t) = f(t, y, y'), \quad y(a) = \alpha,\ y'(a) = s. \tag{4.6}$$

Wir wollen nun $\hat{s}$ so bestimmen, dass für die Lösung $y(t; s)$ von (4.6) gilt

$$y(b; \hat{s}) = \beta,$$

d.h., bestimme die Nullstelle $\hat{s}$ von

$$F(s) := y(b; s) - \beta.$$

Dies ist ein nichtlineares Gleichungssystem. Hier bedeutet s also die Steigung am linken Intervallrand. Man zielt also so lange, bis man rechts trifft. Daher der etwas martialische Name des Verfahrens, siehe auch Abbildung 4.1.

Bemerkung 4.2.1 *Für jedes feste $s \in \mathbb{R}$ kann $y(t; s)$ mit den Verfahren aus Kapitel 1 bis 3 berechnet werden. Dabei entspricht eine Funktionsauswertung von F, also der Lösung einer AWA.*

Bemerkung 4.2.2 *Zur Lösung des nicht-linearen Systems $F(s) = 0$ kann man die Verfahren aus Numerik I verwenden.*

Beispiel 4.2.3 Falls $F \in C^2$ (dies ist im Falle von (4.5), (4.6) gegeben) verwende das Newton–Verfahren

$$\begin{cases} s^{(0)} \in \mathbb{R}^n \text{ Starwert, für } i = 0, 1, 2, \dots \\ F'(s^{(i)}) \Delta s^{(i)} = -F(s^{(i)}) \\ s^{(i+1)} = s^{(i)} + \Delta s^{(i)} \end{cases} \tag{4.7}$$

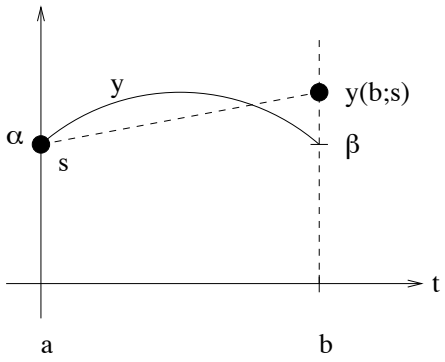


Abb. 4.1: Visualisierung des Schieß-Verfahrens

Man braucht offenbar $F'(s) = \frac{\partial}{\partial s} y(b;s)$. Für die Berechnung betrachte

$$v(t) := v(t,s) := \frac{\partial}{\partial s} y(t;s).$$

Für diese Funktion gilt aufgrund der Glattheit un der Anfangsbedingungen:

$$\begin{aligned}
v(a) &= \frac{\partial}{\partial s} y(a;s) = \frac{\partial}{\partial s}\alpha = 0 && (4.8)\\
v'(a) &= \frac{\partial}{\partial t}\frac{\partial}{\partial s} y(t;s)_{|t=a} = \frac{\partial}{\partial s}\frac{\partial}{\partial t} y(a,s) = \frac{\partial}{\partial s} y'(a) = \frac{\partial}{\partial s} s = 1 && (4.9)\\
v''(t) &= \frac{\partial^2}{\partial t^2}\frac{\partial}{\partial s} y(t;s) = \frac{\partial}{\partial s}\frac{\partial^2}{\partial t^2} y(t;s) = \frac{\partial}{\partial s} y''(t;s)\\
&= \frac{\partial}{\partial s} f(s;y,y')_{|s=t} && (4.10)\\
&= f_y(t;y,y')\frac{\partial}{\partial s} y(t;s) + f_{y'}(t;y,y')\frac{\partial}{\partial s} y'(t,s)\\
&= f_y(t;y,y')v(t) + f_{y'}(t;y,y')v'(t),
\end{aligned}$$

also ist v (für beliebiges s) Lösung einer AWA (4.8)-(4.10). Diese ist jedoch deutlich komplizierter als (4.6)! Man ersetzt daher in (4.7) F' durch den Differenzenquotienten

$$\Delta F(s) := \frac{1}{\Delta s}(F(s+\Delta s) - F(s))$$

mit „genügend kleinem“ Δs, also numerischer Differenziation. Man erhält also ein Quasi-Newton-Verfahren.

Es treten dabei die folgenden naheliegenden Probleme auf

- Wählt man Δs zu groß, ist ΔF eine schlechte Approximation von F', man erhält also langsame Konvergenz von (4.7).
- Wählt man Δs zu klein, dann gilt $F(s+\Delta s) \approx F(s)$, es kommt also zu Auslöschung!

Wir sehen also, dass die Berechnung von F sehr genau sein muss. Dafür verwendet man Extrapolations–Verfahren für F (bzw. $y(b;s)$).

Bemerkung 4.2.4 *Bei allgemeinen Randbedingungen*

$$y' = f(t,y); \quad r(y(a),y(b)) = 0, \quad y = (y_1,\ldots,y_n)^T$$

geht man so vor, dass zunächst ein Startvektor $s^{(0)} \in \mathbb{R}^n$ für das AWP gewählt wird

$$y' = f(t, y); \quad y(a) = s \tag{4.11}$$

und suche $\hat{s} \in \mathbb{R}^n$ so, dass die Lösung $y(t; s)$ von (4.11)

$$r(y(a; \hat{s}), y(b; \hat{s})) = r(\hat{s}, y(b; \hat{s})) = 0$$

erfüllt. Also löse $F(s) = 0$ für

$$F(s) := r(s; y(b; s))$$

wiederum mit obigem Quasi–Newton–Verfahren.

Bemerkung 4.2.5 *Bei **linearen** Randwertproblemen stimmen Newton- und Quasi–Newton–Verfahren überein, d.h. man hat lokal quadratische Konvergenz (sonst lineare). Man kann dann sogar eine explizite Formel herleiten, so dass das Schieß–Verfahren die Lösung in **einem** Schritt liefert.*

Bemerkung 4.2.6 *Ein Teil der oben beschriebenen Probleme löst die **Mehrziel–Methode** (Mehrfach–Schieß–Verfahren, multiple shooting) (Morrison, Riley, Zancanaro 1962).*

Die Idee ist hierbei, das Intervall $[a, b]$ gemäß

$$\Delta = \{a = t_1 < t_2 < \cdots < t_m = b\}, \quad m > 2$$

zu unterteilen. Betrachte nun lokale, unabhängige AWA, vlg. Abbildung 4.2

$$\begin{cases} y_j' = f(t, y_j), & t \in [t_j, t_{j+1}], \quad j = 1, \ldots, m-1, \\ y_j(t_j) = \xi_j. \end{cases}$$

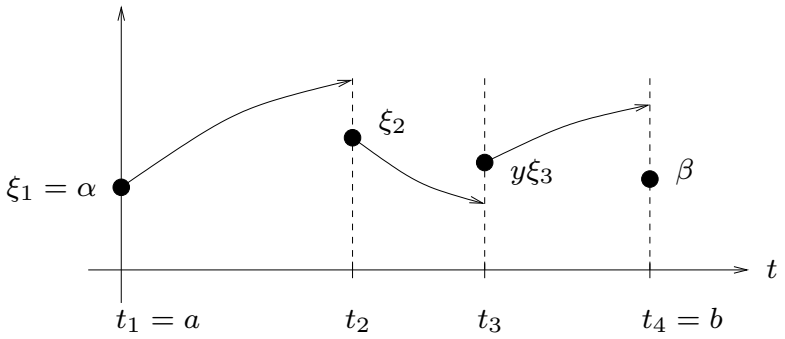


Abb. 4.2: Visualisierung des Mehrfach-Schieß-Verfahrens

Man hat also Stetigkeitsbedingungen

$$F_j(\xi_j, \xi_{j+1}) := y_j(t_{j+1}) - \xi_{j+1} = 0, \quad j = 1, \ldots, m-1,$$

und Randbedingungen

$$F_m(\xi_1, \xi_m) := r(\xi_1, \xi_m) = 0,$$

also ein Gleichungssystem

$$F(\boldsymbol{\xi}) = 0, \quad F = (F_1, \ldots, F_m)^T, \quad \boldsymbol{\xi} = (\xi_1, \ldots, \xi_m)^T.$$

Zu beachten ist hierbei, dass F und F' **zyklische Struktur** besitzen, da F_j nur von zwei ξ_i abhängt

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

Für derartige Systeme kann man spezielle Newton–Verfahren konstruieren [DH, 430-439].

Multiple shooting wurde erfolgreich in einer Reihe komplexer Optimalsteuerungsproblemen verwendet. Typischerweise sind Schieß–Verfahren nur für „**zeit-artige**“ Randwertprobleme verwendbar, für „**raum-artige**“ verwendet man andere Methoden.

4.3 Modellproblem eines raumartigen Zweipunkt RWP

Wir betrachten ein Modellproblem — die schwingende Saite — eines Randwertproblems, das auch als Modellproblem für partielle Differenzialgleichungen verwendet werden kann

$$\begin{aligned} -u''(x) &= f(x), \quad x \in (0,1), && (4.12)\\ u(0) &= u(1) = 0. && (4.13) \end{aligned}$$

Aus dem Hauptsatz der Differenzial- und Integralrechung folgt die Existenz einer Lösung $u \in C^2(0,1)$ von (4.12):

$$u(x) = c_1 + c_2 x - \int_0^x F(s)\, ds$$

mit beliebigen Konstanten $c_1, c_2 \in \mathbb{R}$ und $F(x) := \int_0^x f(t)\, dt$.

Die Idee ist es nun, die Konstanten c_1, c_2 so zu bestimmen, dass (4.13) erfüllt ist. Dazu führen wir eine partielle Integration durch

$$\begin{aligned} \int_0^x F(s)\, ds &= sF(s)\big|_{s=0}^{s=x} - \int_0^x sF'(s)\, ds \\ &= xF(x) - \int_0^x sf(s)\, ds = \int_0^x (x-s)f(s)\, ds. && (4.14) \end{aligned}$$

Setze nun die Randbedingungen ein:

- $0 = u(0) = c_1$
- $0 = u(1) = c_2 - \int_0^1 F(s)\, ds$ also folgt mit (4.14) die Bedingung: $c_2 = \int_0^1 (1-s)f(s)\, ds$,

also

$$\begin{aligned} u(x) &= x\int_0^1 (1-s)f(s)\,ds - \int_0^x (x-s)f(s)\,ds \\ &= \int_0^x [x(1-s)-(x-s)]f(s)\,ds + x\int_x^1 (1-s)f(s)\,ds, \\ &= \int_0^x s(1-x)f(s)\,ds + x\int_x^1 (1-s)f(s)\,ds, \end{aligned}$$

oder kompakter

$$u(x) = \int_0^1 G(x,s)f(s)\,ds \tag{4.15}$$

mit der sogenannten **Green'schen Funktion** des Randwertproblems (4.12), (4.13)

$$G(x,s) := \begin{cases} s(1-x), & 0 \le s \le x, \\ x(1-s), & x \le s \le 1. \end{cases} \tag{4.16}$$

Lemma 4.3.1 *Falls $f \in C^0([0,1])$ exisitiert genau eine Lösung $u \in C^2([0,1])$ von* (4.12)*,* (4.13)*. Falls $f \in C^m([0,1])$, $m \ge 0$, gilt $u \in C^{m+2}([0,1])$ („shift theorem“).*

Beweis. Der erste Teil folgt aus (4.15) und den Eigenschaften von G — der zweite Teil ist nicht so einfach. □

Natürlich kann man die Theorie der Green'schen Funktion auf allgemeine Randwertprobleme erweitern. Dies wird in der Vorlesung „Partielle Differenzialgleichungen“ gemacht. Wir verwenden G hier dazu, um die für die Numerik von Randwertproblemen notwendige Analysis zu vereinfachen.

Satz 4.3.2 *(a) **Monotonie:** Ist $f \ge 0,\ f \in C^0$, dann folgt $u \ge 0$.*

*(b) **Maximumprinzip:** Für $f \in C^0$ gilt*

$$\|u\|_\infty \le \frac{1}{8}\|f\|_\infty. \tag{4.17}$$

Beweis. (a) folgt aus (4.15) wegen $G(x,s) \ge 0, \forall x, s \in [0,1]$. Weiter gilt wegen (4.15)

$$\begin{aligned} |u(x)| &\le \int_0^1 G(x,s)|f(s)|ds \le \|f\|_\infty \int_0^1 G(x,s)ds = \|f\|_\infty \left\{ \int_0^x s(1-x)ds + \int_x^1 x(1-s)ds \right\} \\ &= \|f\|_\infty \left\{ (1-x)\frac{1}{2}x^2 + \frac{1}{2}x(1-x)^2 \right\} = \|f\|_\infty \left\{ \frac{1}{2}x^2 - \frac{1}{2}x^3 + \frac{1}{2}x - x^2 + \frac{1}{2}x^3 \right\} = \frac{1}{2}x(1-x)\|f\|_\infty \end{aligned}$$

woraus sich Behauptung b) ergibt. □

Bemerkung 4.3.3 *Die beiden Eigenschaften aus Satz 4.3.2 sollten auch von jeder numerischen Approximation erfüllt werden.*

4.4 Finite Differenzen–Methode (FDM)

Dieses Verfahren kennen wir teilweise aus Beispielen in der Numerik I.

Die grundlegende Idee fassen wir wie folgt zusammen

- Führe auf $[0,1]$ ein äquidistantes Gitter ein

$$x_j = jh, \quad h = \frac{1}{m}, \quad j = 0, \dots, m.$$

- Bestimme eine Approximation von u durch $\{u_j\}_{j=0}^m$ mit

$$u_j \approx u(x_j).$$

- Ersetze u'' durch den **zentralen Differenzenquotienten**

$$u''(x_j) \approx \frac{1}{h^2}(u_{j+1} - 2u_j + u_{j-1})$$

und löse das Gleichungssystem

$$\begin{cases} -u_{j+1} + 2u_j - u_{j-1} = h^2 f(x_j), & j = 1, \dots, m-1, \\ u_0 = u_m = 0, \end{cases} \tag{4.18}$$

also $A^{(m)} u^{(m)} = f^{(m)}$ mit einer tridiagonalen Matrix $A^{(m)} \in \mathbb{R}^{(m-1)\times(m-1)}$. Dies kann man z.B. mit dem vorkonditionierten cg-Verfahren oder einer speziellen LR-Zerlegung mit optimalem Aufwand in $\mathcal{O}(m)$ lösen.

Lemma 4.4.1 (Diskretes Maximumprinzip) *Falls $f(x_j) \geq 0 \ \forall j = 1, \dots, m-1$, dann gilt $u_j \geq 0 \ \forall j$.*

Beweis. Wir geben hier nur eine Idee. Zeige, dass A eine M–Matrix ist und M–Matrizen Monotonie-erhaltend sind. □

Wir untersuchen nun Stabilitäts- und Konvergenz-Eigenschaften. Dazu betrachten wir wiederum **Gitterfunktionen**

$$\omega_h := \Delta_h \to \mathbb{R}, \quad \Delta_h := \{x_j = jh : j = 0, \dots, m; \ h = \frac{1}{m}\}.$$

Definition 4.4.2 *(a) Sei V_h die Menge aller Gitterfunktionen und setze*

$$V_h^0 := \{\omega_h \in V_h : \omega_h(0) = \omega_h(1) = 0\}.$$

*(b) Wir definieren den **diskreten Operator** $L_h : V_h^0 \to V_h^0$ durch*

$$(L_h \omega_h)(x_j) = -\frac{1}{h^2}(\omega_{j+1} - 2\omega_j + \omega_{j-1}), \quad j = 1, \dots, m-1,$$

sowie $\omega_j := \omega_h(x_j)$.

Damit wird (4.18) zu folgender Aufgabe:

$$\text{Finde } \ u_h \in V_h^0 \ \text{ so dass } \ L_h u_h = f_h$$

mit $f_h = (f(x_j))_{j=1,\dots,m-1}$.

Definition 4.4.3 *Für $v_h, w_h \in V_h$ definiere das **diskrete Skalarprodukt***

$$(w_h, v_h)_h := h \sum_{k=0}^{m} c_k w_k v_k \tag{4.19}$$

*mit $c_0 := c_m := \frac{1}{2}$, $c_k := 1$, $k = 1, \ldots, m-1$, sowie die **diskrete Norm***

$$\|v_h\|_h := \sqrt{(v_h, v_h)_h}.$$

Bemerkung 4.4.4 *Offenbar enstpricht* (4.19) *der zusammengesetzten Trapezregel zur Berechnung des exakten Skalarproduktes*

$$(w, v) = \int_0^1 w(x)v(x)dx,$$

*d.h., $(w_h, v_h)_h = T_h(wv)$ wobei w_h, v_h die Funktionen w, v auf Δ_h interpoliert. Man nennt $\|\cdot\|_h$ auch **Energie–Norm** und die entsprechende Stabilitätsanalyse **Energie–Methode**.*

Wir können die Gleichung $L_h u_h = f_h$ auch äquivalent schreiben als $(L_h u_h, v_h)_h = (f, v_h)_h$ für alle $v_h \in V_h^0$, was man leicht sieht. Mit der Hölder-Ungleichung sieht man ebenso leicht, dass gilt $(w_h, w_h)_h \leq \|w_h\|_h \|v_h\|_h$, also eine Cauchy-Schwarz-Ungleichung. Beides ist für die Analysis des Verfahrens sehr nützlich.

Lemma 4.4.5 *Der Operator L_h ist symmetrisch und positiv definit.*

Beweis. Wir hatten in Numerik I bereits gezeigt, dass $A^{(m)}$ s.p.d. ist. Für die weitere Untersuchung benötigen wir jedoch Teile eines alternativen Beweises: Wegen

$$w_{j+1}v_{j+1} - w_j v_j = (w_{j+1} - w_j)v_j + (v_{j+1} - v_j)w_{j+1}$$

folgt

$$\begin{aligned}\sum_{j=0}^{m-1}(w_{j+1} - w_j)v_j &= \sum_{j=0}^{m-1} w_{j+1}v_{j+1} - \sum_{j=0}^{m-1} w_j v_j - \sum_{j=0}^{m-1}(v_{j+1} - v_j)w_{j+1} \\ &= w_m v_m - w_0 v_0 - \sum_{j=0}^{m-1}(v_{j+1} - v_j)w_{j+1},\end{aligned} \tag{4.20}$$

was auch **partielle Summation** genannt wird. Setze $w_{-1} := v_{-1} := 0$ und verwende (4.20) für $w_h, v_h \in V_h^0$, sowie $v_j = 0$ für $j = 0, m$ und $c_0 = c_m = \frac{1}{2}$, $c_k = 1\ \forall k = 1, \ldots, m-1$

$$\begin{aligned}(L_h w_h, v_h)_h &= h\sum_{j=0}^{m} c_j (L_h w_h)_j v_j = h\sum_{j=1}^{m-1}\left(-\frac{1}{h^2}\right)(w_{j+1} - 2w_j + w_{j-1})v_j \\ &= -h^{-1}\left\{\sum_{j=1}^{m-1}(w_{j+1} - w_j)v_j - \sum_{j=1}^{m-1}(w_j - w_{j-1})v_j\right\} \\ &= -h^{-1}\left\{\sum_{j=0}^{m-1}(w_{j+1} - w_j)v_j - \sum_{j=0}^{m-1}(w_j - w_{j-1})v_j\right\}\end{aligned}$$

wegen $v_0 = 0$. Auf den ersten Term wenden wir partielle Summation an

$$\sum_{j=0}^{m-1}(w_{j+1} - w_j)v_j = -\sum_{j=0}^{m-1}(v_{j+1} - v_j)w_{j+1},$$

da die Randterme verschwinden. Für den zweiten Term gilt

$$\begin{aligned}\sum_{j=0}^{m-1}(w_j - w_{j-1})v_j &= \sum_{j=0}^{m-1} w_j v_j - \sum_{j=0}^{m-1} w_{j-1}v_j = \sum_{j=0}^{m-1} w_j v_j - \sum_{j=0}^{m-2} w_j v_{j+1} \\ &= \sum_{j=0}^{m-1} w_j v_j - \sum_{j=0}^{m-1} w_j v_{j+1} = \sum_{j=0}^{m-1} w_j(v_j - v_{j+1}),\end{aligned}$$

wobei wir $v_m = 0$ benutzt haben. Damit erhalten wir

$$(L_h w_h, v_h)_h = \frac{1}{h}\sum_{j=0}^{m-1}\{(v_{j+1} - v_j)w_{j+1} + w_j(v_j - v_{j+1})\} = \frac{1}{h}\sum_{j=0}^{m-1}(w_{j+1} - w_j)(v_{j+1} - v_j).$$

Daraus folgt sofort $(L_h w_h, v_h)_h = (w_h, L_h v_h)_h$. Setze nun $w_h = v_h$

$$(L_h v_h, v_h)_h = \frac{1}{h}\sum_{j=0}^{m-1}(v_{j+1} - v_j)^2, \tag{4.21}$$

woraus die Positivität folgt. □

Definition 4.4.6 *Für $v_h \in V_h^0$ definiere die Norm*

$$|||v_h|||_h := \left\{ h \sum_{j=0}^{m-1} \left(\frac{v_{j+1} - v_j}{h} \right)^2 \right\}^{1/2}.$$

Nun wird (4.21) zu folgender Aussage: Es gilt

$$(L_h v_h, v_h)_h = |||v_h|||_h^2 \quad \text{für alle } v_h \in V_h^0 \tag{4.22}$$

Lemma 4.4.7 *Es gilt*

$$\|v_h\|_h \leq \frac{1}{\sqrt{2}} |||v_h|||_h \tag{4.23}$$

für alle $v \in V_h^0$.

Beweis. Mit der Minkowski–Ungleichung, vgl. auch [?, S. 222]. □

Lemma 4.4.8 (A priori–Abschätzung) *Es gilt*

$$\|u_h\|_h \leq \frac{1}{2} ||f_h||_h \tag{4.24}$$

Beweis. Aus $L_h u_h = f_h$ folgt komponentenweise wegen $u_h \in V_h^0$

$$(L_h u_h, u_h)_h = h \sum_{j=1}^{m-1} (L_h u_h)(x_j) u_j = h \sum_{j=1}^{m-1} f(x_j) u_j = (f, u_h)_h$$

und mit (4.22), sowie (4.23)

$$\|u_h\|_h^2 \quad \leq \quad \frac{1}{2} |||u_h|||_h^2 = \frac{1}{2} (L_h u_h, u_h)_h = \frac{1}{2} (f, u_h)_h \leq \frac{1}{2} \|f_h\|_h \cdot \|u_h\|_h.$$

□

Bemerkung 4.4.9 *(a) Aus (4.24) ergibt sich auch die eindeutige Lösbarkeit.*

(b) Lemma 4.4.8 besagt, dass u_h stetig von f_h abhängt, also ist dies ein Stabilitätsresultat!

Definition 4.4.10 *Für $f \in C^0([0,1])$ und $u \in C^2([0,1])$ definiert man den **lokalen Abbruchfehler** $\tau_h \in V_h$ durch*

$$\tau_h(x_j) := (L_h u)(x_j) - f(x_j), \quad j = 1, \ldots, m-1.$$

*Mittels der **diskreten Maximumnorm***

$$\|v_h\|_{h,\infty} := \max_{0 \leq j \leq m} |v_h(x_j)|, \quad v_h \in V_h,$$

definiert man die Konsistenz(ordnung) analog zu Kapitel 3.

Lemma 4.4.11 *L_h hat Konsistenzordnung 2, genauer*

$$\|\tau_h\|_{h,\infty} \leq \frac{h^2}{12} \|f''\|_\infty, \tag{4.25}$$

falls $f \in C^2([0,1])$.

Beweis. Taylor–Entwicklung und Vergleich mit den Differenzenquotienten. □

Bemerkung 4.4.12 *(a) Für $f \in C^2$ gilt $u \in C^4$. Man kann Ordnung 2 sogar für $u \in C^{3,1}(0,1) := \{u \in C^3(0,1) : u^{(3)}$ ist Lipschitz-stetig$\}$ zeigen.*

*(b) Den Ausdruck $e_h := u - u_h$ nennt man **Diskretisierungsfehler**. Es gilt*

$$L_h e_h = L_h u - L_h u_h = L_h u - f_h = \tau_h. \tag{4.26}$$

Man kann zeigen (vgl. [Quarteroni et. al., S. 266]), dass

$$\|\tau_h\|_h^2 \leq 3(\|f\|_h^2 + \|f\|_{L_2(0,1)}^2),$$

also ist wegen $\tau_h = L_h e_h$ der Ausdruck e_h'' beschränkt für $f \in C([0,1]) \cap L_2(0,1)$.

Satz 4.4.13 (Stabilität) *Falls $f \in C^2([0,1])$, dann gilt*

$$\|u_h\|_{h,\infty} \leq \frac{1}{8}\|f\|_{h,\infty}.$$

Beweis. Definiere zunächst die **diskreten Green–Funktionen** $G^k \in V_h^0$ durch: mit $\delta^k \in V_h^0$, $\delta^k(x_j) := \delta_{k,j}$, $j = 1, \ldots, m-1$, definiere G^k als Lösung von

$$L_h G^k = \delta^k.$$

Dann gilt

$$G^k(x_j) = hG(x_j, x_k) \tag{4.27}$$

mit der Green'schen Funktion G aus (4.16), vgl. Abbildung 4.3, denn sei x_k fest, dann gilt für alle $j \neq k$

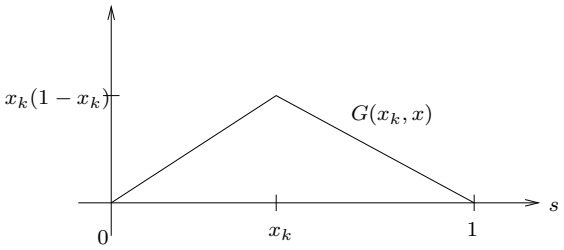


Abb. 4.3: Green'sche Funktion.

$$L_h G(x_j, x_k) = G''(x_j, x_k) = 0$$

sowie aufgrund der Definition von G

$$\begin{aligned} L_h G(x_k, x_k) &= \left(-\frac{1}{h^2}\right)\{G(x_{k+1}, x_k) - 2G(x_k, x_k) + G(x_{k-1}, x_k)\} \\ &= \left(\frac{1}{h^2}\right)\{x_{k+1}(1-x_k) - 2x_k(1-x_k) + x_k(1-x_{k-1})\} \\ &= \frac{1}{h^2}\left((k+1)h(m-k)h - 2kh(m-k)h + kh(m-k+1)h\right) \\ &= (k+1-2k)(m-k) + k(m-k) + k = m - k + k = m = \frac{1}{h}, \end{aligned}$$

also

$$hL_h G(x_j, x_k) = \delta_{j,k} = (L_h G^k)_j,$$

und daraus folgt (4.27). Definiere nun den Opertor $\tilde{T}_h : V_h^0 \to V_h^0$ durch

$$w_h = \tilde{T}_h(g_h), \quad w_h := \sum_{j=1}^{m-1} g_j G^j, \quad g_h \in V_h^0,$$

dann gilt mit $L_h G^j = \delta^j$

$$L_h w_h = \sum_{j=1}^{m-1} g_j L_h G^j = \sum_{j=1}^{m-1} g_j e^j = g_h,$$

also

$$L_h u_h \quad = \quad f_h = L_h(\tilde{T}_h f),$$

woraus $u_h = \tilde{T}_h f$ folgt. Dies bedeutet mit Hilfe von (4.27), dass

$$u_h(x_j) = \sum_{k=1}^{m-1} f(x_k) G^k(x_j) = h \sum_{k=1}^{m-1} f(x_k) G(x_j, x_k).$$

Weiter gilt:

$$|u_h(x_j)| \quad \leq \quad h \sum_{k=1}^{m-1} |f(x_k)| G(x_j, x_k) \leq \|f\|_{h,\infty} h \sum_{k=1}^{m-1} G(x_j, x_k) = \|f\|_{h,\infty} \frac{1}{2} x_j (1 - x_j),$$

denn die Funktion $u(x) = \frac{1}{2} x(1-x)$ löst das RWP $-u''(x) = 1,\ u(0) = u(1) = 0$, woraus die Form der Green'schen Funktion folgt. Bildet man nun das Maximum über j, so ergibt sich

$$\|u_h\|_{h,\infty} \quad \leq \quad \frac{1}{8} \|f\|_{h,\infty},$$

also ein diskretes Analogon von (4.17). □

Nun können wir alles zusammenfügen und erhalten eine Fehlerabschätzung, die die Konvergenz des Verfahrens sichert. Wegen (4.26) $L_h e_h = \tau_h$ folgt mit der Stabilität und der Konsistenz (4.25)

$$\|u - u_h\|_{h,\infty} = \|e_h\|_{h,\infty} \leq \frac{1}{8} \|\tau_h\|_{h,\infty} \leq \frac{1}{8} \cdot \frac{1}{12} h^2 \|f''\|_\infty = \frac{h^2}{96} \|f''\|_\infty, \quad \text{falls } f \in C^2([0,1]).$$

4.5 Variable Koeffizienten

Wir betrachten nun etwas allgemeinere Randwertprobleme der Art

$$\begin{cases} Lu(x) := -(\alpha(x)u'(x))' + \gamma(x)u(x) = f(x), & x \in (0,1), \\ u(0) = d_0, u(1) = d_1, \end{cases}$$

wobei $\alpha, \gamma \in C^0([0,1])$ sogenannte **variable Koeffizienten** sind. Typischerweise setzt man voraus

$$\gamma(x) \geq 0, \quad \forall x \in [0,1], \quad \alpha(x) \geq \alpha_0 > 0, \quad \alpha_0 \text{ fest.}$$

Oft führt man eine Hilfsvariable, den sogenannten **Fluss**

$$J(u)(x) := \alpha(x) u'(x)$$

ein, der oft eine physikalische Bedeutung besitzt und daher genau approximiert werden muss.

Daher führt man oft ein zweites Gitter ein, das sogenannte **verschobene Gitter**

$$x_{j+\frac{1}{2}} := \frac{1}{2}(x_j + x_{j+1})$$

(„**staggered grid**“) und verwendet die zentrale Differenz

$$L_h w(x_j) := -\frac{1}{h}\Big(J_{j+\frac{1}{2}}(w_h) - J_{j-\frac{1}{2}}(w_h)\Big) + \gamma_j w_j \tag{4.28}$$

mit $\gamma_j := \gamma(x_j)$ und dem **approximierten Fluss**

$$J_{j+\frac{1}{2}}(w_h) := \alpha_{j+\frac{1}{2}} \frac{1}{h}(w_{j+1} - w_j). \tag{4.29}$$

Der Vorteil dieser Approximation ist klar, da die System–Matrix symmetrisch ist

$$A_h = D_h + R_h \in \mathbb{R}^{(m-1)\times(m-1)},$$

mit

$$D_h = \frac{1}{h^2}\begin{bmatrix} \alpha_{1/2}+\alpha_{3/2} & -\alpha_{3/2} & & & 0 \\ -\alpha_{3/2} & \alpha_{3/2}+\alpha_{5/2} & -\alpha_{5/2} & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & -\alpha_{m-3/2} \\ 0 & & & -\alpha_{m-3/2} & \alpha_{m-3/2}+\alpha_{m-1/2} \end{bmatrix}$$

und $R_h = \operatorname{diag}(\gamma_1, \ldots, \gamma_{m-1})$. Falls $\gamma > 0$, ist A_h s.p.d. und streng diagonaldominant. Dann führt man das Verfahren analog durch.

4.6 Randbedingungen

Bislang haben wir Randbedingungen der Form

$$u(0) = d_0, \quad u(1) = d_1 \tag{4.30}$$

kennengelernt. Diese heißen **Dirichlet-Randbedingungen**. Zunächst genügt es, sich bei linearen Operatoren L auf **homogene** Randbedingungen ($d_0 = d_1 = 0$) zu beschränken, denn sei u_0 eine glatte, beliebige Funktion, die (4.30) erfüllt. Löse dann

$$Lu^* = f - Lu_0, \quad u^*(0) = u^*(1) = 0,$$

also ein Randwertproblem mit homogenen Randbedingungen, dann löst $u := u^* + u_0$ das ursprüngliche Randwertproblem:

$$\begin{aligned} Lu &= Lu^* + Lu_0 = f - Lu_0 + Lu_0 = f, \\ u(0) &= u^*(0) + u_0(0) = d_0, \\ u(1) &= u^*(1) + u_0(1) = d_1. \end{aligned}$$

Dieser Vorgang heißt **Homogenisierung** (Reduktion auf homogene Randbedingungen).

Oft treten auch andere Randbedingungen als (4.30) auf, z.B. Bedingungen an Ableitungen etwa

$$J(u)(1) = \alpha(1)u'(1), \tag{4.31}$$

sogenannte **Neumann-Randbedingungen**. Zur Diskretisierung von (4.31) verwendet man das **Spiegelungsprinzip**: für $\psi \in C^2$ gilt mit Taylor

$$\psi(x_{m\pm\frac{1}{2}}) = \psi(x_m) \pm \frac{h}{2}\psi'(x_m) + \frac{h^2}{8}\psi''(\xi_m^{\pm})$$

mit Zwischenstellen $\xi_m^+ \in (x_m, x_{m+\frac{1}{2}})$, $\xi_m^- \in (x_{m-\frac{1}{2}}, x_m)$, also

$$\psi(x_m) = \frac{1}{2}(\psi(x_{m-\frac{1}{2}}) + \psi(x_{m+\frac{1}{2}}) - \frac{h^2}{16}(\psi''(\xi_m^+) + \psi''(\xi_m^-)).$$

Für $\psi \equiv J(u)$ ergibt sich eine h^2–Approximation durch

$$g_1 = J(u_m) = \frac{1}{2}(J(u_{m-\frac{1}{2}}) + J(u_{m+\frac{1}{2}})) + \mathcal{O}(h^2),$$

also

$$J(u_{m+\frac{1}{2}}) = 2g_1 - J(u_{m-\frac{1}{2}}), \tag{4.32}$$

wobei $J(u_{m+\frac{1}{2}})$ der Fluss am „fiktiven“ Punkt $x_{m+\frac{1}{2}}$ ist. Für den Punkt x_m ergibt sich also mit (4.28), (4.29) und (4.32)

$$\begin{aligned} f_m &= -\frac{1}{h}(J_{m+\frac{1}{2}}(w_h) - J_{m-\frac{1}{2}}(w_h)) + \gamma_m w_m \\ &= \frac{2}{h}J_{m-\frac{1}{2}}(w_h) + \gamma_m w_m - \frac{2}{h}g_1, \\ &= \frac{2}{h}\alpha_{m-\frac{1}{2}}\frac{1}{h}(w_m - w_{m-1}) + \gamma_m w_m - \frac{2}{h}g_1 \end{aligned}$$

also

$$-\alpha_{m-\frac{1}{2}}\frac{1}{h^2}w_{m-1} + \left(\alpha_{m-\frac{1}{2}}\frac{1}{h^2} + \frac{\gamma_m}{2}\right)w_m = \frac{1}{2}f_m + \frac{g_1}{h}.$$

Dies bedeutet, dass die letzte Zeile im linearen Gleichungssystem modifiziert wird.

Eine weitere Form der Randbedingungen sind

$$\lambda u + u' = g \text{ für } \lambda \neq 0,$$

sogenannte **Robin-Randbedingungen**.

4.7 Galerkin–Verfahren

Diese Klasse von Verfahren sind grundlegender Bestandteil Finiter Elemente Methoden (FEM) und Finite Volumen–Verfahren (FV). Die Analyse für höherdimensionale Probleme (2d, 3d) basiert auf der Funktionalanalysis. Galerkin-Verfahren geben den geeigneten Rahmen zur „korrekten“ Formulierung von Partiellen Differenzialgleichungen (PDEs).

Beispiel 4.7.1 Für $-u''(x) = f(x)$, $x \in (0,1)$, $u(0) = u(1) = 0$, hat man oft nur $f \in L_2([0,1])$, d.h., eine äußere Kraft mit endlicher Energie. Wie ist dann u'' zu verstehen?

Beispiel 4.7.2 Für den **Laplace–Operator**

$$\Delta u(x) := \frac{\partial^2}{\partial x_1}u(x) + \frac{\partial^2}{\partial x_2}u(x)$$

betrachte das Randwertproblem der PDE

$$-\Delta u(x) = f(x), \quad x \in \Omega, \tag{4.33}$$

für $\Omega := (-1,1)^2 \backslash [0,1]^2$ (das so genannte „L-Gebiet“). Selbst für $f \in C^\infty(\Omega)$ existiert **kein** $u \in C^2(\Omega) \cap C(\bar{\Omega})$ mit (4.33).

Eine Schlussfolgerung ist, dass die punktweise Interpretation von Differenzialgleichungen oft keinen Sinn macht. Ein Ausweg wäre beispielsweise

- Physikalische Gesetze, die auf Differenzialgleichungen führen, gelten oft nur **im Mittel** über Kontrollvolumina(z.B. physikalische Erhaltungssätze wie etwa das Prinzip der Masse-Erhaltung)
- Suche einen schwächeren Ableitungsbegriff.

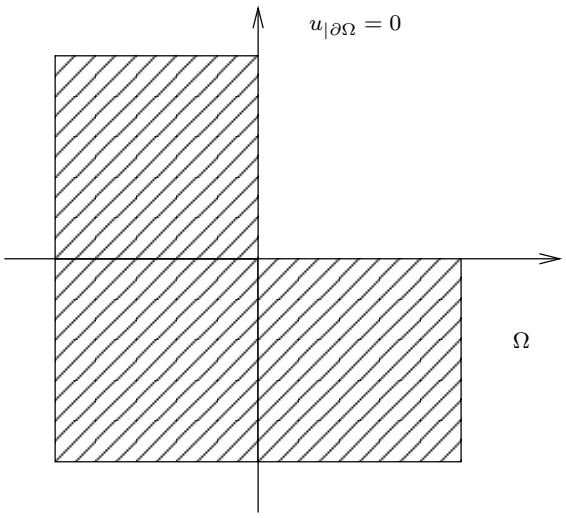


Abb. 4.4: Laplace-Problem auf dem L-Gebiet.

Wir betrachten nun die allgemeine Form linearer Randwertprobleme zweiter Ordnung in 1D

$$\begin{cases} -(\alpha(x)u'(x))' + \beta(x)u'(x) + \gamma(x)u(x) = f(x), & x \in (0,1) \\ u(0) = u(1) = 0. \end{cases} \tag{4.34}$$

mit $\alpha, \beta, \gamma \in C^0([0,1])$, $\alpha(x) \geq \alpha_0 > 0$. Nun sei $v \in C_0^1([0,1]) = \{w \in C^1([0,1]) : w(0) = w(1) = 0\}$, dann folgt aus (4.34)

$$\begin{aligned} \int_0^1 f(x)v(x)dx &= -\int_0^1 (\alpha(x)u'(x))'v(x)dx \\ &\quad + \int_0^1 \beta(x)u'(x)v(x)dx + \int_0^1 \gamma(x)u(x)v(x)dx \\ &= \underbrace{[\alpha(x)u'(x)v(x)]_{x=0}^{x=1}}_{=0\,(v\in C_0^1)} + \int_0^1 \alpha(x)u'(x)v'(x)dx \\ &\quad + \int_0^1 \beta(x)u'(x)v(x)dx + \int_0^1 \gamma(x)u(x)v(x)dx. \end{aligned} \tag{4.35}$$

Offenbar ist (4.35) für alle Funktionen u, v sinnvoll, für die beide Seiten endlich sind:

$$H_0^1(0,1) := \{v \in L_2(0,1) : v' \in L_2(0,1), v(0) = v(1) = 0\},$$

wobei v' hier wieder **stückweise** zu verstehen ist. Dieser Raum heißt wie in Abschnitt **?? Sobolev–Raum**, hier mit Randbedingungen.

Mit dem **Test-** und **Ansatzraum** $V := H_0^1(0,1)$ lautet dann die **schwache Formulierung** von (4.34)

$$\text{finde } u \in V : a(u,v) = (f,v), \qquad \forall v \in V, \tag{4.36}$$

mit der Bilinearform

$$a(u,v) := \int_0^1 \alpha(x)u'(x)v'(x)dx + \int_0^1 \beta(x)u'(x)v(x)dx + \int_0^1 \gamma(x)u(x)v(x)dx$$

und der Linearform („Funktional“)

$$F(v) := (f,v) := \int_0^1 f(x)v(x)dx \quad \text{(Skalarprodukt in } L_2\text{)}.$$

Bemerkung 4.7.3 *Falls $u \in C^2([0,1])$ das Randwertproblem* (4.34) *löst, dann ist u auch Lösung von* (4.36). *Umgekehrt, falls $u \in V$* (4.36) *löst und zusätzlich $u \in C^2([0,1])$ gilt, dann ist u auch Lösung von* (4.34).

Definition 4.7.4 *Sei $u \in L_2(0,1)$. Eine Funktion $v \in L_2(0,1)$ heißt* ***schwache Ableitung*** *von u ($v = u'$), falls*

$$\int_0^1 v(x)\varphi(x)dx = -\int_0^1 u(x)\varphi'(x)dx$$

für alle Testfunktionen $\varphi \in C_0^\infty(0,1) = \{w \in C^\infty(0,1) : w(0) = w(1) = 0\}$.

Beispiel 4.7.5 Sei die Hutfunktion $u(x)$ gegeben durch, vgl. Abbildung 4.5

$$u(x) := \begin{cases} x, & 0 \le x < \frac{1}{2}, \\ 1-x, & \frac{1}{2} \le x \le 1, \end{cases}$$

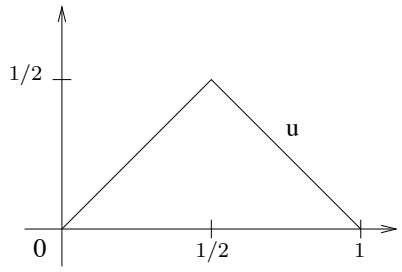


Abb. 4.5: Bsp 4.7.5: Visualisierung der Hutunktion $u(x)$.

Dann gilt $u'(x) = h(x) = \chi_{[0,1/2]}(x) - \chi_{[1/2,1]}(x)$, vgl. Abbildung 4.6

Man definiert dann die Sobolev–Norm

$$\|u\|_{H^1}^2 := \|u\|_{L_2}^2 + \|u'\|_{L_2}^2$$

und zeigt dann

$$H_0^1(0,1) = \overline{C_0^\infty(0,1)}^{\|\cdot\|_{H^1}}.$$

Letzteres nutzt man als Definition in höheren Dimensionen: $H_0^1(\Omega) := \overline{C_0^\infty(\Omega))}^{\|\cdot\|_{H^1}}$ für $\Omega \subset \mathbb{R}^n$.

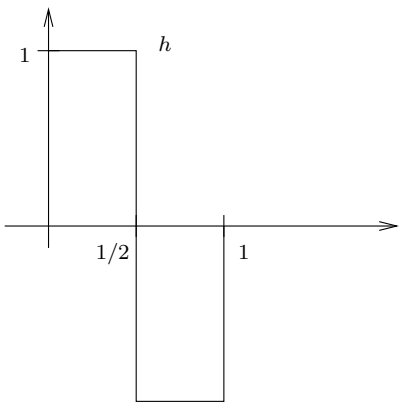


Abb. 4.6: Bsp 4.7.5: Visualisierung der Funktion $u'(x)$.

Bemerkung 4.7.6 *(a) Mit Hilfsmitteln aus der Funktionalanalysis sichert der Satz von Lax–Milgram, dass die schwache Formulierung* (4.36) *(auch Variationsformulierung genannt) für alle $f \in L_2(0,1)$ eine* ***eindeutige Lösung*** *besitzt mit*

$$\|u\|_{H^1} \lesssim \|f\|_{L_2}.$$

vgl. [Arendt/Urban].

(b) Neumann- und Robin–Randbedingungen können ähnlich behandelt werden. Die partielle Integration liefert hier wieder wie in Abschnitt 4.6 Randterme auf der rechten Seite.

Nun ist (4.36) so numerisch unbrauchbar, da $H_0^1(0,1)$ ein ∞-dimensionaler Raum ist. Die Idee des Galerkin–Verfahrens ist es, V durch einen endlich-dimensionalen Teilraum

$$V_h \subset V, \quad \dim V_h =: N_h =: N < \infty$$

zu approximieren:

$$\text{finde } u_h \in V_h : a(u_h, v_h) = (f, v_h), \quad \forall v_h \in V_h. \tag{4.37}$$

Falls eine Basis $\{\varphi_1, \ldots, \varphi_N\}$ von V_h bekannt ist, genügt es, (4.37) für alle φ_i zu testen und man sucht

$$u_h = \sum_{j=1}^{N} u_j \varphi_j$$

mit

$$\begin{aligned} (f, \varphi_i) &= a(u_h, \varphi_i) = a\left(\sum_{j=1}^{N} u_j \varphi_j, \varphi_i\right) \\ &= \sum_{j=1}^{N} a(\varphi_i, \varphi_j) u_j, \quad \forall i = 1, \ldots, N, \end{aligned}$$

also

$$A_N u_N = f_N$$

mit der sogenannten **Steifigkeitsmatrix**

$$A_N = (a(\varphi_i, \varphi_j))_{i,j=1,\ldots,N} \in \mathbb{R}^{N \times N}$$

und den Vektoren

$$u_N = (u_i)_{i=1,\ldots,N}, \quad f_N = (f, \varphi_i))_{i=1,\ldots,N} \in \mathbb{R}^N.$$

Beispiel 4.7.7 Für $m > 1$ sei $x_i = ih$, $h = \frac{1}{m}$, $\Delta = \{x_i : i = 0, \ldots, m\}$. Sei $V_h = S^0_{2,\Delta}$ (Raum der linearen Splines bzgl. der Knoten x_i, $i = 1, \ldots, m-1$, und homogenen Randbedingungen)

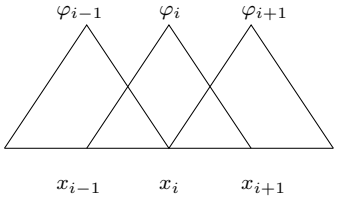


Abb. 4.7: Hut–Funktionen.

Also gilt $\dim V_h = N = m - 1$. Dann gilt im Beispiel $-u''(x) = f(x)$ für die Einträge der Steifigkeitsmatrix wegen $h = x_i - x_{i-1} = x_{i+t} - x_i$

$$\begin{aligned} a(\varphi_i, \varphi_i) &= \int_{x_{i-1}}^{x_{i+1}} (\varphi_i'(x))^2 dx \\ &= \int_{(i-1)h}^{ih} \Big[\underbrace{\frac{d}{dx}\left(\frac{x - x_{i-1}}{x_i - x_{i-1}}\right)}_{=\frac{1}{h}} \Big]^2 dx + \int_{ih}^{(i+1)h} \Big[\underbrace{\frac{d}{dx}\left(\frac{x_{i+1} - x}{x_{i+1} - x_i}\right)}_{=-\frac{1}{h}} \Big]^2 dx \\ &= \frac{2}{h} \end{aligned}$$

für die Diagonaleinträge und

$$\begin{aligned} a(\varphi_{i-1}, \varphi_i) &= \int_{x_{i-1}}^{x_i} \varphi_{i-1}'(x) \varphi_i'(x) dx \\ &= \int_{(i-1)h}^{ih} \left(-\frac{1}{h}\right)\left(\frac{1}{h}\right) dx = -\frac{1}{h} = a(\varphi_i, \varphi_{i-1}) \end{aligned}$$

sowie $a(\varphi_i, \varphi_j) = 0$ für $|i - j| \geq 2$, also

$$A_N = \frac{1}{h} \begin{bmatrix} 2 & -1 & & 0 \\ -1 & \ddots & \ddots & \\ & \ddots & \ddots & -1 \\ 0 & & -1 & 2 \end{bmatrix}$$

wie bei FDM. Trotzdem kommt i.A. eine andere Näherungslösung heraus, denn bei der FDM lautet die rechte Seite

$$f_i = f(x_i),$$

bei Galerkin hingegen $f_i = (f, \varphi_i)$.

Bemerkung 4.7.8 *Die Spektral–Kollokations–Methode* (**??**) *lautete*

$$\textit{finde } u_m \in \mathcal{P}_m^0(I) \textit{ mit } (L_m u_m, v_m)_m = (f, v_m)_m \quad \forall v_m \in \mathcal{P}_m^0,$$

also eine spezielle Form eines Galerkin–Verfahrens mit

$$a_m(u_m, v_m) := (L_m u_m, v_m)_m$$

und dem Funktional $F(v_m) := (f, v_m)_m$ auf der rechten Seite. Wegen der besonderen Form von $a_m(\cdot,\cdot)$ kann man beide Verfahren jedoch nicht völlig analog analysieren.

Nun zur Analyse. Wir definieren die **Semi–Norm** (Halb–Norm)

$$|v|_{H^1} := \|v'\|_{L_2}.$$

Damit lautet die Poincaré–Friedrichs–Ungleichung (**??**) etwas allgemeiner

$$\|v\|_{L_2} \le C_I |v|_{H^1}, \quad v \in H_0^1(I). \tag{4.38}$$

Lemma 4.7.9 (Stabilität) *Für die Koeffizienten in* (4.34) *gelte zusätzlich*

$$\beta \in C^1([0,1]), \quad -\frac{1}{2}\beta'(x) + \gamma(x) \ge 0, \qquad \forall x \in [0,1], \tag{4.39}$$

dann gilt

$$|u_h|_{H^1} \le \frac{C_I}{\alpha_0} \|f\|_{L_2}.$$

Beweis. Wähle als Testfunktion $v_h = u_h$, dann gilt

$$\begin{aligned}(f, u_h) &= a(u_h, u_h) \\ &= \int_0^1 \alpha(x) u_h'(x) u_h'(x) dx + \int_0^1 \beta(x) u_h'(x) u_h(x) dx + \int_0^1 \gamma(x) u_h^2(x) dx.\end{aligned}$$

Mit partieller Integration gilt für alle $V_h \in H_0^1(I)$

$$\int_0^1 \beta(x) v_h(x) v_h'(x) dx = -\int_0^1 (\beta(x) v_h(x))' v_h(x) dx = -\int_0^1 \beta'(x) v_h^2(x) dx - \int_0^1 \beta(x) v_h'(x) v_h(x) dx,$$

also

$$\int_0^1 \beta(x) v_h(x) v_h'(x) dx = -\frac{1}{2}\int_0^1 \beta'(x) v_h^2(x) dx$$

und mit $\left(\gamma(x) - \frac{1}{2}\beta'(x)\right) \ge 0$ nach (4.39)

$$a(v_h, v_h) = \int_0^1 \alpha(x)(v_h'(x))^2 dx + \int_0^1 \left(\gamma(x) - \frac{1}{2}\beta'(x)\right) v_h^2(x) dx \ge \alpha_0 \int_0^1 (v_h'(x))^2 dx = \alpha_0 |v_h(x)|_{H_1}^2, \tag{4.40}$$

d.h. $a(\cdot,\cdot)$ ist koerziv auf $H_0^1(I)$. Mit Cauchy–Schwarz und (4.38) gilt schließlich für die Lösung $u_h \in H_0^1(I)$ von (4.37)

$$\alpha_0 |u_h|_{H_1}^2 \le a(u_h, u_h) = (f, u_h) \le \|f\|_{L_2} \|u_h\|_{L_2} \le C_I \|f\|_{L_2} |u_h|_{H^1}.$$

□

Satz 4.7.10 (Céa–Lemma) *Mit $C := (\|\alpha\|_\infty + C_I\|\beta\|_\infty + C_I^2\|\gamma\|_\infty^2)$ gilt $a(u,v) \le C\,|u|_{H^1}\,|v|_{H^1}$ und*

$$|u - u_h|_{H^1} \le \frac{C}{\alpha_0} \cdot \min_{w_h \in V_h} |u - w_h|_{H^1}.$$

Bemerkung 4.7.11 *Die Galerkin–Lösung u_h ist also — bis auf eine Konstante — so gut wie die* ***beste Approximation*** *an u aus V_h. Damit wird die Bestimmung der Konvergenzrate zu einem Problem der Approximationstheorie.*

Beweis. Betrachte noch einmal (4.36) und (4.37)

$$\begin{array}{ll} (4.36) & \text{finde } u \in V: \quad a(u,v) = (f,v), \qquad \forall v \in V \\ (4.37) & \text{finde } u_h \in V_h: \quad a(u_h,v_h) = (f,v_h), \quad \forall v_h \in V_h \subset V \end{array}$$

Teste nun (4.36) mit $v_h \in V_h \subset V$ und subtrahiere beide

$$a(u-u_h, v_h) = (f,v_h) - (f,v_h) = 0, \qquad \forall v_h \in V,$$

die sogenannte **Galerkin–Orthogonalität**. Damit gilt für den Fehler $e_h := u - u_h$ für beliebiges $w_h \in V_h$ mit (4.40), (4.38) und $w_h - u_h \in V_h$

$$\begin{aligned} \alpha_0 |e_h|_{H^1}^2 &\leq a(e_h,e_h) \\ &= a(e_h, u-w_h) + a(e_h, w_h - u_h) = a(e_h, u-w_h) \\ &= \int_0^1 \alpha e_h'(u-w_h)' dx + \int_0^1 \beta e_h'(u-w_h) dx + \int_0^1 \gamma e_h (u-w_h) dx \\ &\leq \|\alpha\|_\infty |e_h|_{H^1} |u-w_h|_{H^1} + \|\beta\|_\infty |e_h|_{H^1} \|u-w_h\|_{L_2} \\ &\quad + \|\gamma\|_\infty \|e_h\|_{L_2} \|u-w_h\|_{L_2} \\ &\leq |e_h|_{H^1} |u-w_h|_{H^1} \{\|\alpha\|_\infty + C_I \|\beta\|_\infty + C_I^2 \|\gamma\|_\infty\}. \end{aligned}$$

□

Bemerkung 4.7.12 *Offensichtlich gelten Lemma 4.7.9 und Satz 4.7.10 in einem viel allgemeineren Rahmen, der auch für partielle Differenzialgleichungen notwendig ist: Sei V ein Hilbert–Raum mit Norm $\|\cdot\|_V$ und $a: V \times V \to \mathbb{R}$ sei eine Bilinearform mit*

$$\begin{array}{ll} \exists \alpha_0 > 0 : a(v,v) \geq \alpha_0 \|v\|_V^2, & \forall v \in V, \quad \textbf{(Koerzivität)}, \\ \exists C > 0 : |a(u,v)| \leq C \|u\|_V \|v\|_V, & \forall u,v \in V, \quad \textbf{(Stetigkeit, Beschränktheit)}. \end{array}$$

Für die rechte Seite gelte außerdem

$$\exists\, K > 0 : |(f,v)| \leq K ||v||_V, \quad \forall v \in V$$

dann gilt

(a) ***Satz von Lax–Migram***: (4.36), (4.37) *besitzen eindeutige Lösungen $u \in V$ bzw. $u_h \in V_h$ mit*

$$\|u\|_V \leq \frac{K}{\alpha_0}, \quad \|u_h\|_V \leq \frac{K}{\alpha_0}.$$

(b) ***Céa–Lemma***: $\|u-u_h\|_V \leq \frac{C}{\alpha_0} \min\limits_{w_h \in V_h} \|u-w_h\|_V$.

Lemma 4.7.13 *Für $\beta = 0$ und $\gamma \geq 0$ ist die Steifigkeitsmatrix A_N s.p.d.*

Beweis. Die Symmetrie folgt aus der Symmetrie von $a(\cdot,\cdot)$. Nun sei

$$\mathbf{v}_N = (v_j)_{j=1,\ldots,N} \in \mathbb{R}^n$$

der Koeffizienten–Vektor eines $v_h = \sum\limits_{j=1}^N v_j \varphi_j \in V_h$, dann gilt

$$\mathbf{v}_N^T A_N \mathbf{v}_N \quad = \quad \sum_{i,j=1}^N v_i a_{ij} v_j = \sum_{i,j=1}^N v_i a(\varphi_i, \varphi_j) v_j = a\left(\sum_{i=1}^N v_i \varphi_i, \sum_{j=1}^N v_j \varphi_j \right) = a(v_h, v_h),$$

also $v_N^T A_N v_N \geq \alpha_0 \|v_h\|_V^2 > 0$ falls $v_h \neq 0$ und vershwindet genau dann, wenn $v_h = 0$. □

Bemerkung 4.7.14 *Für $\beta \neq 0$ (nicht-verschwindende Konvektion) ist A_N unsymmetrisch.*

4.8 Finite Elemente Methode (FEM)

In Beispiel 4.7.7 hatten wir bereits eine erste FEM gesehen (Hutfunktion). Der allgemeine Ansatz lautet: Zu $\Omega := (0,1)$ wählt man eine Zerlegung

$$\mathcal{T}_h = \{T_i\}_{i=1}^{N_h}, \quad N = N_h,$$

in Intervalle (sogenannte **Elemente**) $T_i = [x_i, x_{i+1}]$, so dass

$$\bar{\Omega} = \bigcup_{i=1}^{N_h} T_i,$$

also o.B.d.A. $0 = x_0 < x_1 < \cdots < x_{N-1} < x_N = 1$,

$$h_i := x_{i+1} - x_i, \quad i = 0, \ldots, N-1, \quad h := \max_i h_i.$$

Als Ansatzfunktionen wählt man stückweise Polynome

$$X_h^k := \{v_h \in C^0(\bar{\Omega}) : v_{h|T_i} \in \mathcal{P}_k, \ \forall i = 1, \ldots, N_h\},$$

bzw.

$$X_h^{k,0} := \{v_h \in X_h^k : v_h(0) = v_h(1) = 0\}.$$

Satz 4.8.1 *Für die exakte Lösung u von* (4.36) *gelte*

$$u \in H_0^1(\Omega) \cap H^s(\Omega)$$

für ein $s \geq 2$. Dann gilt für die Lösung $u_h \in X_h^{k,0}$ von (4.37)

$$\|u - u_h\|_{H^1(\Omega)} \lesssim h^\ell \|u\|_{H^{\ell+1}(\Omega)}, \quad \ell = \min\{k, s-1\},$$

und

$$\|u - u_h\|_{L_2(\Omega)} \lesssim h^{\ell+1} \|u\|_{H^{\ell+1}(\Omega)}, \quad \ell = \min\{k, s-1\}. \tag{4.41}$$

Bemerkung 4.8.2 *(a) Der Beweis erfolgt mit Mitteln der Funktionalanalysis und Approximationstheorie.*

(b) Die Abschätzung (4.41) *(mit $\ell \to \ell + 1$, $H_0^1 \to L_2$) heißt auch* ***Aubin–Nitsche–Trick****.*

*(c) Die „**Regularitätsschranke**“*

$$\ell = \min\{k, s-1\}$$

besagt auch, dass es wenig Sinn macht, Elemente mit mehr als $k = s - 1$ zu verwenden. Allerdings ist s in der Regel nicht bekannt! Ausweg: Fehlerschätzer, adaptive Methoden, hp-Methoden.

(d) Für $s = 1$ gilt nur noch Konvergenz, keine Ordnung, für $s = 2$, $\ell = k = 1$ folgt

$$\|u - u_h\|_{H_0^1(\Omega)} = \mathcal{O}(h), \quad \|u - u_h\|_{L_2(\Omega)} = \mathcal{O}(h^2), \quad h \to 0+.$$

(e) Es gilt

$$\dim X_h^{k,0} = N_h k - 1$$

also für die Hutfunktionen $(k = 1)$ genau die Anzahl der ***inneren Knoten****.*

Für die numerische Lösung ist

$$\kappa_2(A_N), \quad A_N = A_h$$

eine entscheidende Größe. Zur Ermittlung zunächst eine Vorbereitung. Wir betrachten nur den linearen Fall $k = 1$.

Lemma 4.8.3 (Inverse Abschätzung) *Es gilt*

$$|v_h|_{H^1} \leq \frac{C}{h}\|v_h\|_{L_2}, \quad \forall v_h \in V_h = X_h^{1,0}. \tag{4.42}$$

Lemma 4.8.4 (Norm–Äquivalenz) *Für* $v_h = \sum_{j=1}^{N-1} v_j\varphi_j \in X_h^{k,0}$, $\mathbf{v} := (v_j)_{j=1}^{N-1} \in \mathbb{R}^{N-1}$, $\|v\|^2 := \sum_{j=1}^{N-1} |v_j|^2$ *existieren Konstanten* $0 < \tilde{c} < \tilde{C} < \infty$ *mit*

$$\tilde{c}\|v_h\|_{L_2} \leq h\|\mathbf{v}\| \leq \tilde{C}\|v_h\|_{L_2}. \tag{4.43}$$

Satz 4.8.5 *Es gilt*

$$\kappa_2(A_N) = \mathcal{O}\left(\frac{1}{h^2}\right), \quad h \to 0+.$$

Bemerkung 4.8.6 *Dies besagt also, dass die Kondition schlechter wird, wenn die Diskretisierung besser wird! Daher ist eine Vorkonditionierung dringend notwendig! Dies ist unabhängig vom speziellen Beispiel* $-u'' = f$ *und* $V_h = X_h^{1,D}$.

Beweis. Wir schätzen die Rayleigh–Quotienten ab mit Hilfe von (4.42) und (4.43)

$$\frac{\mathbf{v}^T A_h \mathbf{v}}{\|\mathbf{v}\|^2} = \frac{a(v_h, v_h)}{\|\mathbf{v}\|^2} \lesssim \frac{|v_h|_{H^1}^2}{\|\mathbf{v}\|^2} \lesssim \frac{1}{h^2}\frac{\|v_h\|_{L_2}^2}{\|\mathbf{v}\|^2} \lesssim 1,$$

also $\lambda_{\max}(A_h) \leq C$. Andererseits gilt mit (4.43)

$$\frac{\mathbf{v}^T A_h \mathbf{v}}{\|\mathbf{v}\|^2} = \frac{a(v_h, v_h)}{\|\mathbf{v}\|^2} \geq \frac{\alpha_0\|v_h\|_{H^1}^2}{\|\mathbf{v}\|^2} \geq \alpha_0\frac{\|v_h\|_{L_2}^2}{\|\mathbf{v}\|^2} \geq \frac{\alpha_0}{\tilde{C}}h^2,$$

also $\lambda_{\min}(A_h) \gtrsim h^2$. □

Zur Vereinfachung der Darstellung (und der Implementierung) reduziert man typischerweise alle Berechungen auf ein **Referenzelement**

$$\hat{T} = [0, 1]$$

mit Hilfe der affin-linearen Transformation

$$\phi : [0, 1] \to T_i, \quad x = \phi(\xi) := x_i + \xi(x_{i+1} - x_i), \quad i = 0, 1, \ldots, n-1.$$

Auf dem Referenzelement definiert man dann sogenannte **Formfunktionen**, z.B. im linearen Fall

$$\hat{\varphi}_0(\xi) = 1 - \xi, \quad \hat{\varphi}_1(\xi) = \xi$$

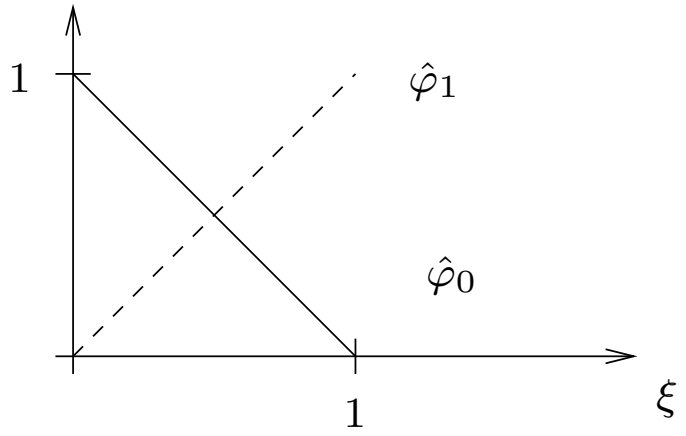


Abb. 4.8: Lineare Formfunktionen

und damit ergeben sich die Ansatzfunktionen φ_i, φ_{i+1} auf T_i durch

$$\varphi_i(x) := \hat{\varphi}_0(\xi(x)), \quad \varphi_{i+1}(x) := \hat{\varphi}_1(\xi(x))$$

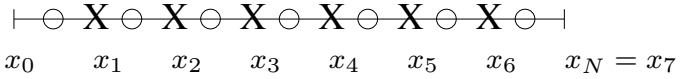
mit

$$\xi(x) = \phi^{-1}(x) = \frac{x - x_i}{x_{i+1} - x_i}, \quad \xi : T_i \to \hat{T}.$$

Allgemein definiert man $k+1$ Formfunktionen $\hat{\varphi}_0, \ldots, \hat{\varphi}_k$ auf $\hat{T}$ und transformiert diese auf T_i.

Beispiel 4.8.7 (Quadratische FE) Wir wollen Formfunktionen $\hat{\varphi}_0, \hat{\varphi}_1, \hat{\varphi}_2$ für X_h^2 konstruieren.

- $V_{h|_{T_i}} \in \mathcal{P}_2$, d.h. also 3 Punkte legen $v_{h|_{T_i}}$ fest
- Stetigkeit: Wähle Funktionswerte in den Knoten $x_1, \ldots, x_{N-1}$ (x_0, x_N sind durch die Randbedingungen festgelegt), woraus folgt, dass $N-1$ Freiheitsgrade (d.o.f. – degrees of freedom) existieren. Nach Bemerkung 4.8.2 gilt $\dim X_h^{2,0} = 2N - 1$, es fehlen also N d.o.f.



wähle also die Mittelpunkte

$$\hat{\varphi}_0(\xi) = (1-\xi)(1-2\xi), \qquad \hat{\varphi}_1(\xi) = 4(1-\xi)\xi, \qquad \hat{\varphi}_2(\xi) = \xi(2\xi - 1),$$

vgl. Abbildung 4.9

Offensichtlich sind $\hat{\varphi}_i$ interpolatorisch, deswegen heißt diese Basis auch vom **Lagrange–Typ**.

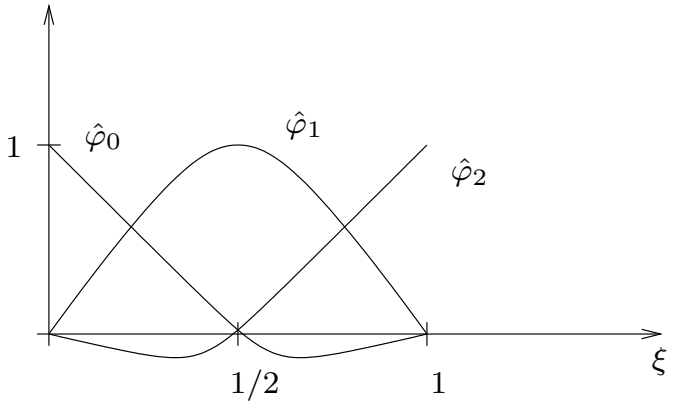
Beispiel 4.8.8 (Hierarchische Basen) Die Idee ist eine Verfeinerung $h \to \frac{h}{2}$ oder $k \to k+1$. Diese sollte unter **Hinzunahme** neuer Funktionen geschehen (Aufdatierung). Bei $X_h^{1,0}$ ist dies kein Problem bzgl. h, vgl. Abbildung 4.10.
Diese Basis führt zu besseren Konditionszahlen (Yserentant, 1986)

$$\kappa(A_N) = \begin{cases} \mathcal{O}(1), & d = 1 \quad \text{(optimal!)}, \\ \mathcal{O}(N+1)^2, & d = 2 \quad (N \to \infty), \\ \mathcal{O}(2^{(d-2)N}), & d \geq 3. \end{cases}$$

Für $X_h^{2,0}$ kann man zeigen, dass folgende Formfunktionen auf eine Basis führen (vgl. Abb. 4.11):

$$\hat{\varphi}_0(\xi) := 1 - \xi, \qquad \hat{\varphi}_1(\xi) := 4(1-\xi)\xi, \qquad \hat{\varphi}_2(\xi) := \xi.$$

Dies ist besonders günstig für hp-Methoden!



ξ	0	1/2	1
$\hat{\varphi}_0(\xi)$	1	0	0
$\hat{\varphi}_1(\xi)$	0	1	0
$\hat{\varphi}_2(\xi)$	0	0	1

Abb. 4.9: Quadratische Formfunktionen

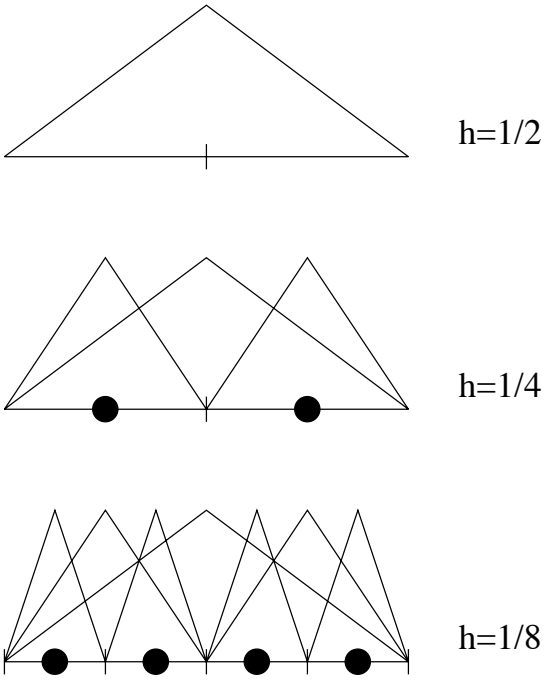


Abb. 4.10: Hierarchische Basen

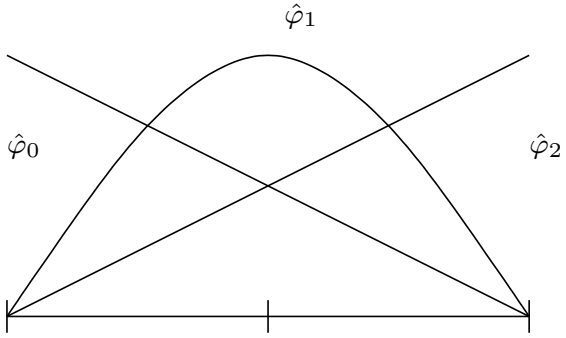


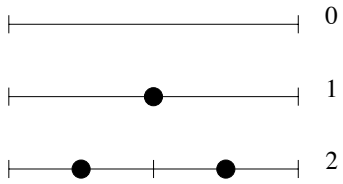
Abb. 4.11: Hierarchische quadratische Formfunktionen

4.9 Multilevel–Verfahren

Dies sind moderne und äußerst effiziente Verfahren. Man betrachtet hier nicht einen einzelnen Raum V_h, sondern eine Folge geschachtelter Räume

$$V_0 \subset V_1 \subset \cdots \subset V_m \subset \ldots \subset V (\text{z.B. } H_0^1(\Omega))$$

(z.B. $V_m = V_{2^{-m}}$, d.h. $h = 2^{-m}$), ähnlich zur MRA bei Wavelets.



Mit $\mathcal{V} := \bigcup_{i=0}^{\infty} V_i \subseteq V$ (hier muss **nicht** „=“ gelten!) assoziiert man zu jedem V_j einen **Projektor** P_j

$$P_j : \mathcal{V} \to V_j$$

mit

$$P_m^2 = P_m \text{ (Projektion)}, P_m V_k = V_m, \quad k \geq m. \tag{4.44}$$

Beispiel 4.9.1 Sei $V = L_2(\Omega)$, $V_j = X_h^1$ bezüglich des Gitters Δ_h, $h = 2^{-j}$. Dann kann man z.B.

$$P_j = P_{\Delta_h},$$

also den Interpolations–Operator bzgl. Δ_h wählen.

Zu beachten ist hierbei, dass wir **nicht** verlangen, dass $P_m u \longrightarrow u$, $(m \to \infty)$, $u \in V$, selbst wenn P_m auf ganz V definiert ist! Diese Familie $\mathcal{P} := \{P_j\}_{j=0}^{\infty}$ erzeugt eine **Multilevel–Zerlegung** des Ansatzraums V_m. Sei $m \geq j$ und $u_j \in V_j$, dann gilt

$$u_j = P_{j-1} u_j + (1 - P_{j-1}) u_j$$

wegen (4.44) gilt $P_{j-1} u_j \in V_{j-1}$, also (ähnlich wie bei Wavelets)

$$V_j = V_{j-1} \oplus N_{j-1} \tag{4.45}$$

mit

$$N_{j-1} := \{u_j \in V_j : P_{j-1}u_j = 0\} = V_j \cap \text{ Kern } (P_{j-1}).$$

für $j \geq 1$.

Bemerkung 4.9.2 *Die Zerlegung* (4.45) *ist nur dann orthogonal, ($V_{j-1} \perp N_{j-1}$), wenn die P_j orthogonale Projektionen sind!*

Iteriert man (4.45), so erhält man

$$V_j = \bigoplus_{m=0}^{j-1} N_m, \quad N_0 := V_0.$$

Andererseits gilt mit einer Teleskopsumme

$$V_j \ni u_j = P_0 u_j + \sum_{m=1}^{j} (P_m - P_{m-1})u_j$$

und damit erhalten wir eine weitere Zerlegung $V_j = V_{j-1} \oplus W_{j-1}$ mittels

$$\begin{cases} V_j = \bigoplus_{m=0}^{j-1} W_m, & W_m := \text{ Bild } (P_m - P_{m-1}) \\ & W_0 := V_0. \end{cases} \tag{4.46}$$

Lemma 4.9.3 *(a) Für die Projektoren gilt stets*

$$P_j P_{j-1} = P_{j-1}.$$

(b) Es gilt

$$N_j = W_j$$

***genau** dann, wenn*

$$P_{j-1}P_j = P_{j-1}. \tag{4.47}$$

In diesem Fall ist $Q_j := P_j - P_{j-1}$ eine Projektion mit $Q_j Q_k = 0$, $j \neq k$.

Beweis. (a) ist trivial, (b) benötigt (etwas) Funktionalanalysis, weswegen wir den Beweis hier weglassen. □

Bemerkung 4.9.4 *Die Bedingung* (4.47) *ist z.B. für den Interpolationsoperator I_j bzgl. $\Delta_{j-1} \subset \Delta_j$ oder auch den orthogonalen Projektor P_j erfüllt.*

Nun zur konkreteren Form des sogenannten **BPX–Vorkonditionierers** (Bramble, Pasciak, Xu, 1990), der optimal ($\mathcal{O}(1)$) ist. Dazu sei

$$V_j = \text{ span } \{\varphi_1^{(j)}, \ldots, \varphi_{n_j}^{(j)}\}$$

(z.B. Finite Elemente bezüglich geschachtelter Gitter). Wir formulieren zunächst das diskrete Problem leicht um: zu

$$u \in V : \ a(u,v) = (f,v), \quad \forall v \in V, \quad f \text{ gegeben },$$

definiere $f_j \in V_j$ durch

$$(f_j, v_j) = (f, v_j), \quad \forall v_j \in V_j, \tag{4.48}$$

d.h. $f_j = P_j f$ mit der orthogonalen Projektion $P_j : V \to V_j$, denn $P_j f$ ist eindeutig bestimmt durch

$$(f - P_j f, v_j) = 0, \quad \forall v_j \in V_j,$$

also

$$(P_j f, v_j) = (f, v_j)$$

und damit $f_j = P_j f$. Damit ist das diskrete Problem

$$u_j \in V_j : a(u_j, v_j) = (f, v_j), \quad \forall v_j \in V_j,$$

äquivalent zu

$$\mathbf{u}_j \in \mathbb{R}^{n_j} : \mathbf{A}_j \mathbf{u}_j = \mathbf{\Phi}_j \mathbf{f}_j$$

mit der Steifigkeitsmatrix $A_j = (a(\varphi_i^{(j)}, \varphi_k^{(j)}))_{i,k=1,\ldots,n_j} \in \mathbb{R}^{n_j \times n_j}$, der Gram–Matrix

$$\mathbf{\Phi}_j = ((\varphi_i^{(j)}, \varphi_k^{(j)}))_{i,k=1,\ldots,n_j}$$

und $\mathbf{f}_j = (\tilde{f}_i)_{i=1,\ldots,n_j}$ mit $f_j = \sum_{i=1}^{n_j} \tilde{f}_i \varphi_i^{(j)}$, denn wegen $\varphi_i^{(j)} \in V_j$ gilt mit (4.48)

$$\begin{aligned} a(u_j, \varphi_i^{(j)}) &= (f, \varphi_i^{(j)}) = (f_j, \varphi_i^{(j)}) \\ &= \sum_{k=1}^{n_j} \tilde{f}_k (\varphi_k^{(j)}, \varphi_i^{(j)}) = \mathbf{\Phi}_j \mathbf{f}_j. \end{aligned}$$

Definition 4.9.5 *Der **BPX–Vorkonditionierer** C_j ist für $v_j \in V_j$ definiert durch*

$$C_j^{-1} v_j = \sum_{k=0}^{j} 4^{-k} \sum_{i=1}^{n_k} \frac{(v_j, \varphi_i^{(k)})}{\|\varphi_i^{(k)}\|_{H^1(\Omega)}} \varphi_i^{(k)}. \tag{4.49}$$

Bemerkung 4.9.6 *Man beachte, dass C_j **nicht** als Matrix gegeben ist, sondern nur durch die Anwendung von C_j^{-1} auf ein Element aus V_j. Dies genügt aber für den Algorithmus vollkommen.*

Algorithmus 4.9.1: BPX-pcg

Sei $\mathbf{u}_j^{(0)} \in \mathbb{R}^{n_j}$ ein beliebiger Startwert.
$\mathbf{z}^{(0)} := C_j^{-1}(f_j - A_j u_j^{(0)})$
$\mathbf{p}^{(0)} := \mathbf{z}^{(0)}$

Für $i = 0, 1, 2, \ldots$ bestimme

1.) $\mathbf{r}^{(i)} := \mathbf{\Phi}_j \mathbf{f}_j - \mathbf{A}_j \mathbf{u}_j^{(0)}$ (Residuum)
2.) $\alpha_i := \frac{(\mathbf{r}^{(i)}, \mathbf{z}^{(i)})}{(\mathbf{A}_j \mathbf{p}^{(i)}, p^{(i)})}$
3.) $\mathbf{u}^{(i+1)} := \mathbf{u}^{(i)} + \alpha_i \mathbf{p}^{(i)}$

4.) $\mathbf{r}^{(i+1)} := \mathbf{r}^{(i)} - \alpha_i \mathbf{A}_j \mathbf{p}^{(i)}$
5.) $\mathbf{z}^{(i+1)} := C_j^{-1}(f_j - A_j u_j^{(i)})$
6.) $\beta_i := \frac{(\mathbf{r}^{(i+1)}, \mathbf{z}^{(i+1)})}{(\mathbf{r}^{(i)}, \mathbf{z}^{(i)})}$
7.) $\mathbf{p}^{(i+1)} := \mathbf{z}^{(i+1)} + \beta_i \mathbf{p}^{(i)}$.

Bemerkung 4.9.7 *Die beiden Anwendungen von C_j^{-1} beziehen sich auf $f_j - A_j u_j^{(i)} \in V_j$ (nicht auf die entsprechenden Koeffizienten–Vektoren!), wobei*

$$A_j \varphi_i^{(j)} := \sum_{k=1}^{n_j} a(\varphi_i^{(j)}, \varphi_k^{(j)}) \varphi_k^{(j)}$$

der diskrete Opterator $A_j : V_j \to V_j$ ist.

Satz 4.9.8 *Für den BPX–Vorkonditionierer gilt*

$$\kappa(C_j^{-1} A_j) = \mathcal{O}(1), \quad j \to \infty, \tag{4.50}$$

*der Vorkonditionierer ist **asymptotisch optimal**.*

Der Beweis folgt später.

Bemerkung 4.9.9 *Die Aussage* (4.50) *bedeutet Folgendes: um einen Anfangsfehler ε_0 auf eine Genauigkeit $\varepsilon < \varepsilon_0$ zu reduzieren, benötigt Algorithmus 4.9.1 gleich viele Schritte, egal wie fein das Randwertproblem diskretisiert ist, also unabhängig von j!!*

Nun zur **rekursiven Berechnung von** $C_j^{-1} v_j$. Wegen $V_{j-1} \subset V_j$ gilt

$$\varphi_i^{(j-1)} = \sum_{k=1}^{n_j} \alpha_k^j \varphi_k^{(j)}, \quad 1 \le i \le n_{j-1}, \tag{4.51}$$

mit Koeffizienten α_k^j, die von der Diskretisierung abhängen (und von denen die meisten verschwinden). Dies ist analog zur Verfeinerungsgleichung bei Skalierungsfunktionen.

Beispiel 4.9.10 Sei $V_j = X_{h_j}^1$, dann erhalten wir

$$\varphi_i^{(j-1)} = \frac{1}{2}\varphi_{2i-1}^{(j)} + \varphi_{2i}^{(j)} + \frac{1}{2}\varphi_{2i+1}^{(j)}, \qquad 1 \le i \le n_{j-1} = 2^{j-1}, \ (n_j = 2 \cdot 2^{j-1} = 2^j),$$

vgl. Abbildung 4.12.

Für die Effizienz des Verfahrens ist es absolut essentiell, (4.51) nur auf die $\alpha_k^j \neq 0$ zu beschränken!

Nun sei angenommen, dass $(v_j, \varphi_i^{(k)})$ bekannt seien für $1 \le i \le n_k$, dann gilt

$$(v_j, \varphi_i^{(k-1)}) = \sum_{m=1}^{n_k} \alpha_m^k (v_j, \varphi_m^{(k)}),$$

also eine Rekursion. Also brauchen wir nur noch

$$(f_j - A_j u_j^{(i)}, \varphi_k^{(j)}), \quad k = 1, \ldots, n_j,$$

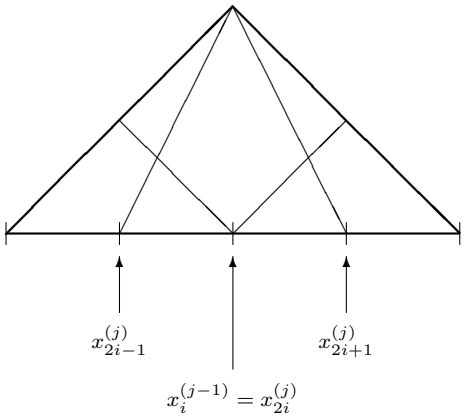


Abb. 4.12: Geschachtelte Räume erzeugt durch lineare Elemente für den BPX-Vorkonditionierer

denn nur für $v_j = f_j - A_j u_j^{(i)}$ wird $C_j^{-1} v_j$ in Algorithmus 4.9.1 benötigt. Nun gilt aber

$$\begin{aligned}(f_j - A_j u_j^{(i)}, \varphi_k^{(k)}) &= (f_j, \varphi_k^{(k)}) - (A_j u_j^{(i)}, \varphi_k^{(k)}) \\ &= (\mathbf{\Phi}_j \mathbf{f}_j)_{|_k} - (\mathbf{A}_j \mathbf{u}_j^{(i)})_{|_k} \\ &= (\mathbf{\Phi}_j \mathbf{f}_j - \mathbf{A}_j \mathbf{u}_j^{(i)})_{|_k} = \mathbf{r}_{|_k}^{(i)}\end{aligned}$$

und diese Zahlen werden im Algorithmus berechnet!

Nun zum Beweis von Satz 4.9.8, für den wir einige Vorbereitungen brauchen.

Definition 4.9.11 *Sei X endlich-dimensional mit Skalarprodukt $(\cdot,\cdot)$. Zwei s.p.d. Operatoren $A, B : X \to X$ heißen **spektral äquivalent**, falls es Konstanten $c_0, c_1 > 0$ gibt mit*

$$c_0(Bu, u) \le (Au, u) \le c_1(Bu, u), \quad \forall u \in X. \tag{4.52}$$

Lemma 4.9.12 *Seien $A, B : X \to X$ s.p.d., dann gilt*

(a) $\sigma(AB) = \sigma(BA)$.

(b) Seien zusätzlich A und B spektral äquivalent, dann gilt

$$\kappa(B^{-1}A) \le \frac{c_1}{c_0}$$

mit den Konstanten aus (4.52).

Beweis. (a) Siehe Numerik I (pcg-Verfahren).

(b) Mit dem Skalarprodukt

$$\langle u, v\rangle := (Bu, v)$$

gilt $(Au, u) = (BB^{-1}Au, u) = \langle B^{-1}Au, u\rangle$ und $(Bu, u) = \langle u, u\rangle$, also aus (4.52)

$$c_0\langle u, u\rangle \le \langle B^{-1}Au, u\rangle \le c_1\langle u, u\rangle$$

und mit dem Rayleigh–Quotienten bzgl. $\langle\cdot,\cdot\rangle$ gilt $\lambda_{\min}(B^{-1}A) \ge c_0$, $\lambda_{\max}(B^{-1}A) \le c_1$. □

Bemerkung 4.9.13 *Wir wissen bislang noch nicht, dass C_j s.p.d. ist!*

Der BPX–Vorkonditionierer entsteht aus der Multilevel–Zerlegung (4.46), wenn man P_j als **orthogonale Projektion** wählt

$$(P_j u, v_j) = (u, v_j), \quad \forall v_j \in V_j.$$

Nach Lemma 4.9.3 sind $Q_j := P_j - P_{j-1}$ auch Projektoren mit

$$Q_j Q_k = Q_k \delta_{kj}.$$

Dann zeigt man (mit reichlich F.A. und Approximationstheorie), dass

$$\sum_{k=0}^{j} 4^k \|Q_k u_j\|^2_{L_2(\Omega)} \sim \|u_j\|^2_{H^1(\Omega)}, \quad \forall u_j \in V_j \tag{4.53}$$

(man braucht u.a. direkte und inverse Abschätzungen) und dann liefert

$$a(u,u) \sim \|u\|^2_{H^1(\Omega)}$$

bereits, dass A_j und die linke Seite von (4.53) spektral äquivalent sind. Also sei

$$\tilde{C}_j u_j := \sum_{k=0}^{j} 4^k Q_k u_j, \quad u_j \in V_j,$$

dann gilt

$$\tilde{C}_j^{-1} u_j := \sum_{k=0}^{j} 4^{-k} Q_k u_j,$$

denn

$$\begin{aligned} \tilde{C}_j \tilde{C}_j^{-1} &= \sum_{k=0}^{j} 4^k Q_k \left\{ \sum_{m=0}^{j} 4^{-m} Q_m \right\} = \sum_{k=0}^{j} \sum_{m=0}^{j} 4^k 4^{-m} \underbrace{Q_k Q_m}_{=\delta_{k,m} Q_m} \\ &= \sum_{k=0}^{j} Q_k = \sum_{k=0}^{j} (P_k - P_{k-1}) = P_j, \end{aligned}$$

also auf V_j die Identität.

Nun ist die orthogonale Projektion P_j numerisch aufwändig zu realisieren (man müsste dazu $\{\varphi_1^{(j)}, \ldots, \varphi_{n_j}^{(j)}\}$ orthonormalisieren). Also ersetzt man P_j durch einen spektral äquivalenten Operator M_j, der dann auf die Form (4.49) führt. Dieser sei definiert als

$$M_j v_j := \sum_{i=1}^{n_j} \frac{1}{\alpha_i^{(j)}} (v_j, \varphi_i^{(j)}) \varphi_i^{(j)}$$

mit

$$\alpha_i^{(j)} := \|\varphi_i^{(k)}\|,$$

vgl. (4.49), die Definition von BPX.

Lemma 4.9.14 *Folgende Eigenschaften gelten*

(a) $M_k u_j = M_k P_k u_j, \quad u_j \in V_j, \quad k \le j.$

(b) $(u_j, M_k u_j) = (P_k u_j, M_k P_k u_j), \quad u_j \in V_j, \quad k \le j.$

Beweis. Es gilt $(P_k u_j, \varphi_i^{(k)}) = (u_j, \varphi_i^{(k)}), 1 \le i \le n_k, k \le j$ also

$$M_k(P_k u_j) = \sum_{i=1}^{n_j} \frac{1}{\alpha_i^{(j)}} (P_k u_j, \varphi_i^{(k)}) \varphi_i^{(k)} = M_k u_j$$

woraus sich wegen $(P_k u_j, \varphi_i^{(k)}) = (u_k, \varphi_i^{(k)})$ gerade die Behauptung a) ergbit. Weiter gilt wegen $M_k P_k u_j \in V_k$

$$(u_j, M_k u_j) = (u_j, M_k P_k u_j) = (P_k u_j, M_k P_K u_j).$$

□

Satz 4.9.15 *Es gilt* $(u_j, M_j u_j) \sim (u_j, P_j u_j) \sim (u_j, u_j)$. *Für* $u_j \in V_j$, *also sind* C_j *und* M_j *spektral äquivalent.*

Bemerkung 4.9.16 *Andere asymptotische optimale Verfahren sind (a) Mehrgitter–Verfahren (Multigrid); (b) MG-pcg (Mehrgitter vorkonditioniertes cg); (c) Wavelet–Verfahren.*

Literaturverzeichnis

[AS] M. ABRAMOWITZ, I.A. STEGUN Pocketbook of Mathematical Functions with Formulas, Verlag Harri Deutsch, Frankfurt/Main (1984).

[A] R.A. ADAMS, "Sobolev Spaces", Pure Appl. Math. 65, Academic Press, New York, 1975.

[Alt1] W. Alt Nichtlineare Optimierung — Eine Einführung in Theorie, Verfahren und Anwendung, Vieweg, Braunschweig, 2002.

[Alt2] W. Alt Numerische Verfahren der konvexen, nichtglatten Optimierung, Teubner Stuttgart, 2004.

[Analysis I] S. FUNKEN, Skript zur Vorlesung „Analysis I", gehalten im Wintersemester 07/08 an der Universität Ulm.

[Analysis II] S. FUNKEN, Skript zur Vorlesung „Analysis II" gehalten im Sommersemester 2008 an der Universität Ulm.

[Angewandte Numerik I] S. FUNKEN, D. LEBIEDZ, K. URBAN, Skript zur Vorlesung „Angewandte Numerik I", gehalten im Sommersemester 14 an der Universität Ulm.

[Arendt/Urban] W. ARENDT, K. URBAN, Partielle Differenzialgleichungen: Eine Einführung in analytische und numerische Methoden, Spektrum Akademischer Verlag 2010.

[Braess] D. Braess Finite Elemente, 2. Auflage Springer, 1997.

[Canuto et. al.] C. Canuto, m.Y. Hussaini, A. Quarteroni, T.A. Zhang: Spectral Methods, 2. Auflage Springer, 2007.

[Calvetti] D. CALVETTI, G.H. GOLUB, W.B. GRAGG UND L. REICHEL, Computation of Gauss-Kronrod rules. Math. Comp. 69, 1035–1052 (2000).

[Cull] P. CULL, M. FLAHIVE UND R. ROBSON, Difference equations, Undergraduate Texts in Mathematics. Springer, New York (2005).

[Cuyt/Wuytack] A. CUYT, L. WUYTACK, Nonlinear Methods in Numerical Analysis, North-Holland (Amsterdam).

[Dahl] G. DAHLQUIST, A special stability problem for linear multistep methods, BIT 3, 27-4,3 1963.

[Deuflhard/Hohmann] P. Deuflhard, A. Hohmann, Numerische Mathematik I, de Gruyter Berlin, 4. Auflage 2008.

[Deuflhard/Bornemann] P. Deuflhard, F. Bornemann, Numerische Mathematik II, de Gruyter Berlin, 2002.

[Discroll/Maki] T. A. DISCROLL, K. L. MAKI: Searching for Rare Growth Factors Using Multicanonical Monte Carlo Methods. SIAM Review. Vol. 49, No. 4, pp. 673-692. 2008.

[Elayadi] S. ELAYADI An introduction to difference equations, Undergraduate Texts in Mathematics. Springer, New York (2005).

[Fischer] G. FISCHER, Linare Algebra, Vieweg-Verlag, Wiesbaden, 1989.

[Forst/Hoffmann] W. FORST, D. HOFFMANN, Gewöhnliche Differenzialgleichungen, Springer, 2005.

[Großmann/Roos] C. Großmann, H.-G. Roos Numerische Behandlung partieller Differentialgleichungen, Teubner Wiesbaden, 2005.

[Golub/Loan] G. H. GOLUB, C. F. VAN LOAN, Matrix Computations, 3. ed., Hopkins Univ. Press, 1996.

[Hackbusch] W. HACKBUSCH, Iterative Lösung großer schwachbesetzter Gleichungssysteme, 2. Auflage, Teubner-Verlag, Stuttgart, 1993.

[Hämmerlin/Hoffmann] G. HÄMMERLIN, K.-H. HOFFMANN, Numerische Mathematik, 4. Auflage, Springer-Verlag, Berlin, 1994.

[Hairer et. al.] E. Hairer, S.P. Nørsett, G. Wanner, Solving Ordinary Differential Equations I, Springer, 1987.

[Hairer/Wanner] E. Hairer, G. Wanner, Solving Ordinary Differential Equations II, Springer, 1991.

[Hanke] M. HANKE-BOURGEOIS, Grundlagen der Numerischen Mathematik und des wissenschaftlichen Rechnens , 1. Auflage, Teubner-Verlag, Stuttgart, 2002.

[Heuser] H. Heuser: Funktionalanalysis. Teubner, 3. Auflage, 1992.

[Higham] N. J. HIGHAM, How accurate is Gaussian elimination? Numerical Analysis 1989, Proceedings of the 13th Dundee Conference, Vol. 228 of Pitman Research Notes in Mathematics, 137–154.

[Kiefer] J. KIEFER, Optimum sequential search and approximation methods under minimum regularity assumptions. J. Soc. Ind. Appl. Math. 5, 105-136 (1957).

[Knuth] D.E. KNUTH The Art of Computer Programming, Band 2, Addison-Wesley, 3. Auflage, 1998.

[Kronrod] A.S. KRONROD, Nodes and weights of quadrature formulas. Sixteen-place tables. New York: Consultants Bureau. Authorized translation from the Russian (1965).

[Knabner/Angermann] P. Knabner, L. Angermann Numerik partieller Differentialgleichungen, Springer Berlin, 2000.

[Larsson/Thomée] S. Larsson, V. Thomée Partielle Differentialgleichungen und numerische Methoden, Springer Berlin, 2005.

[Laurie] D.P. LAURIE, Calculation of Gauss-Kronrod quadratur rules. Math. Comp. 1133-1145 (66) 1997.

[Lebed] G. K. LEBED, Quadrature formulas with minimum error for certain classes of functions, Mathematical Notes 3, 368-373 (1968).

[Marsden] M. J. MARSDEN, An identity for spline functions with applications to variation-dimishing spline approximation. J. Approx. Theory 3 (1970), 7-49.

[Meyberg/Vachenauer] K. MEYBERG, P. VACHENAUER, Höhere Mathematik 1. Springer, Berlin (1999)

[Numerik I] S. FUNKEN, D. LEBIEDZ, K. URBAN, Skript zur Vorlesung „Numerik I“, gehalten im Wintersemester 12/13 an der Universität Ulm.

[Numerik II] S. FUNKEN, D. LEBIEDZ, K. URBAN, Skript zur Vorlesung „Numerik II“, gehalten im Sommersemester 13 an der Universität Ulm.

[Numerik III] S. FUNKEN, D. LEBIEDZ, K. URBAN, Skript zur Vorlesung „Numerik III “, gehalten im Wintersemester 13/14 an der Universität Ulm.

[Niethammer] W. NIETHAMMER, Relaxation bei nichtsymmetrischen Matrizen. Math. Zeitschr. **85** 319-327, 1964.

[Plato] R. PLATO, Numerische Mathematik kompakt, Vieweg-Verlag.

[Quarteroni et. al.] A. QUARTERONI, R. SACCO, F. SALERI, Numerische Mathematik, Band 1 & 2, Springer-Verlag, Berlin, 2002.

[QSS1] A. QUARTERONI, R. SACCO, F. SALERI, Numerische Mathematik 1, Springer 2002.

[QSS2] A. QUARTERONI, R. SACCO, F. SALERI, Numerische Mathematik 2, Springer 2002.

[Rivlin] T.J. RIVLIN, An Introduction to the Approximation of Functions, Blaisdell Publ., Waltham, MA, 1969.

[Schönhage] A. SCHÖNHAGE, Approximationstheorie, de Gruyter, Berlin, 1971.

[Schwarz] H. R. SCHWARZ, Numerische Mathematik, 4. Auflage, Teubner-Verlag, Stuttgart, 1997.

[Stör/Bulirsch] J. STÖR, R. BULIRSCH, Numerische Mathematik, Band 1 & 2, Springer-Verlag, Berlin, 1994.

[Stoer] J. STOER, R. BULIRSCH, Numerische Mathematik, Band 2, 5. Auflage, Springer-Verlag, Berlin, 2005.

[SW] K. STREHMEL, R. WEINER, Numerik gewöhnlicher Differentialgleichungen, Teubner-Verlag, Stuttgart, 1995.

[St] A. H STROUD, Gaussian quadrature formulas, Prentice-Hall, Englewood Cliff (1966).

[Sz] G. SZEGÖ, Orthogonal Polynomials, AMS 3. Auflage, 1967.

[Törnig/Spellucci] W. TÖRNIG, P. SPELLUCCI, Numerische Mathematik für Ingenieure und Physiker, Band 1 & 2, Springer-Verlag, Berlin, 1988.

[W] W. WALTER, Gewöhnliche Differentialgleichungen, 6. Auflage, Springer-Verlag, Berlin u.a., 1996.

[Wilkinson65] J. H. WILKINSON, The Algebraic Eigenvalue Problem, Oxford University Press, 1965.

[Wilkinson69] J. H. WILKINSON, Rundungsfehler, Springer-Verlag, Berlin, Heidelberg, New York, 1969.

[Wille] D. WILLE, Repetitorium der Linearen Algebra, Teil 1, 1. Auflage, Feldmann-Verlag, Springer, 1989.

Index