Skriptum zur Vorlesung

Numerische Mathematik I und II

1990/1991

C. Reinsch

3.Auflage

Mathematisches Institut der Technischen Universität München

Bücher zur Vertiefung

Für die theoretische Seite und Analyse:

[1] Stoer, J., *Numerische Mathematik 1*, Springer-Lehrbuch, 1989 (5. Auflage)

[2] Stoer, J., Bulirsch, R., *Numerische Mathematik 2*, Springer-Lehrbuch, 1990 (3. Auflage)

Für Algorithmen und fertige Computer-Programme:

[1] Press, W.H., Flannery, B.P., Teukolsky, S.A., Vetterling, W.T., *Numerical Recipes (Fortran Version)*, Cambridge Press, 1990

[2] Press, W.H., Flannery, B.P., Teukolsky, S.A., Vetterling, W.T., *Numerical Recipes in C*, Cambridge Press, 1990

[3] Press, W.H., Flannery, B.P., Teukolsky, S.A., Vetterling, W.T., *Numerical Recipes in Pascal*, Cambridge Press, 1990

Zu den letzten drei Büchern gibt es eine Floppy Disk im MS DOS-Format.

Software

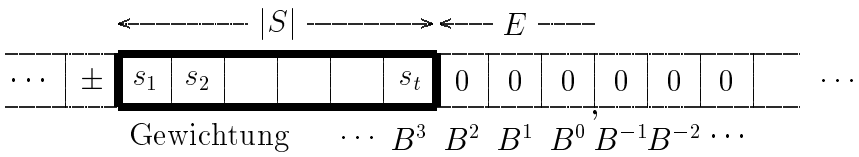
für Arbeitsplatz-Rechner vom Typ IBM PC (compatible), Apple Macintosh, Sun, Apollo, HP, DEC gibt es ein weit verbreitetes Software-System MATLAB, vertrieben von

The Math Works, Inc.	E-mail: info@mathworks.com
24 Prime Park Way	FAX: (508) 653-2997
Natick, MA	Telex: 910-240-5521
U.S.A.	Phone: (508) 653-1415
	http://www.mathworks.com

Bezeichnungen

$:=$		Definitions-Zeichen in Formeln, Zuweisungs-Zeichen in Algorithmen
$\Rightarrow$		daraus folgt
$\Leftrightarrow$		zwischen äquivalenten Aussagen
$\mapsto$		für Zuordnungen und Übergänge
$\forall$		für alle, für jedes
$\exists$		existiert mindestens ein
$\exists_0$		für kein, existiert kein
$\exists_1$		existiert genau ein
$:$		so, daß
$\ni$		so, daß
$\equiv$		für verschiedene Bezeichnungen desselben Begriffs, zum Beispiel $df/dx \equiv f'(x) \equiv Df(x) \equiv \mathrm{grad} f \equiv \nabla f$
$\mathbb{N}$	$:=$	$\{1, 2, 3, \ldots\}$
$\mathbb{N}_0$	$:=$	$\{0, 1, 2, \ldots\}$
$\mathbb{Z}$	$:=$	$\{0, \pm 1, \pm 2, \ldots\}$
$\mathbb{R}$	$:=$	Menge der reellen Zahlen
$\mathbb{R}^n$	$:=$	Menge der reellen n-Tupel und der Vektorraum dazu
$\mathbb{R}^{m,n}$	$:=$	Menge der reellen $m \times n$ Matrizen
$\mathbb{C}$	$:=$	Menge der komplexen Zahlen
$\mathbb{P}_n$	$:=$	Menge der Polynome vom Grad $\leq n$ in einer reellen oder komplexen Variablen
$\mathbb{P}$	$:=$	$\mathbb{P}_0 \cup \mathbb{P}_1 \cup \mathbb{P}_2 \cup \ldots$
$[a, b]$	$:=$	$\{x \in \mathbb{R} : a \leq x \leq b\}$
$]a, b[$	$:=$	$\{x \in \mathbb{R} : a < x < b\}$
$C^m[a, b]$	$:=$	Menge der auf $[a, b]$ m-mal stetig differenzierbaren reellwertigen Funktionen
$y = O(x)$		$\exists \gamma \ \ni \ \lvert y\rvert \leq \gamma \cdot \lvert x\rvert$ für $x \to 0$ oder $x \to \infty$
$y = o(x)$		$\lvert y\rvert / \lvert x\rvert \ \to 0$ für $x \to 0$ oder $x \to \infty$ (Landau-Symbol, „Ordnung“)
A^T		transponiert
A^H		adjungiert (transponiert und konjungiert komplex)
A^{-T}	$:=$	$(A^T)^{-1} = (A^{-1})^T$
A^{-H}	$:=$	$(A^H)^{-1} = (A^{-1})^H$
I oder I_n		$n \times n$ Einheitsmatrix (Identität)
$e^1, \ldots, e^n$		ihre Spalten
$\delta_{i,k}$		ihr i-k-Element (Kronecker-Symbol)

Kodierung, deren Details für die Numerik völlig belanglos sind. Zur Veranschaulichung kann man sich g vorstellen mit einem Vorzeichen und t Ziffern in einem Radix-B-System, wo E den Abstand zum Komma angibt:



Zwei aufeinander folgende positive Gleitpunktzahlen haben die Werte $g = S \cdot B^E$ und $g' = (S+1) \cdot B^E$, also den relativen Abstand $(g'-g)/g = 1/S$. Gleiches gilt für zwei negative Nachbarn, etwa $g = -S \cdot B^E$ und $g' = -(S+1) \cdot B^E$. Zusammen:

Auflösung (engl. *resolution*) in $\mathbb{G}_{B,t}$:

$$\varrho := \max\{|g-g'|/|g| : g, g' \text{ Nachbarn in } \mathbb{G}_{B,t}\} = 1/B^{t-1}.$$

B und t bestimmen zusammen die Feinheit der semi-logarithmischen Abstufung in der Menge der positiven und negativen Gleitpunktzahlen. Weitere aus B, t, α und β abgeleitete Parameter sind:

$$\begin{aligned} \text{die kleinste positive Maschinenzahl:} \quad & \sigma := B^{t-1} \cdot B^{\alpha}, \\ \text{die größte Maschinenzahl:} \quad & \lambda := (B^t - 1) \cdot B^{\beta}. \end{aligned}$$

Als Beispiel für die Wahl der vier Parameter sei das sogenannte IEEE-Format (Aussprache: [*ai triple i:*]) angeführt aus ANSI/IEEE-Std-754-1985, der weitgehend akzeptierten Normung für Mikroprozessoren (empfehlenswerte Literatur dazu ist das August-1984-Heft der Zeitschrift *IEEE Mikro*, S. 86–100):

Genauigkeitsstufe	B	t	α	β	ϱ	σ	$\lambda \doteq$
single precision	2	24	-149	104	2^{-23}	2^{-126}	2^{+128}
double precision	2	53	-1074	971	2^{-52}	2^{-1022}	2^{+1024}
extended precision	2	64	-16445	16320	2^{-63}	2^{-16382}	2^{+16384}

Man beachte die leichte Unsymmetrie im Exponenten-Bereich ($\sigma\lambda \doteq 4$ statt 1). In diesem Standard gibt es übrigens noch eine Reihe von Sonder-Operanden:

$+\infty$:	größer als jede reelle Zahl,
$-\infty$:	kleiner als jede reelle Zahl,
quiet NaN (**n**ot-**a**-**n**umber):	unbestimmter Wert, vererbt sich bei Operationen,
signalizing NaN:	löst als Operand einen Alarm aus,
denormalisierte Zahlen:	$0 < \|S\| < B^{t-1}$ und $E = \alpha$.

für $\mathbb{G}_{B,t}$ kann man konkret ansetzen:

$$x > 0 \implies x =: (S+\vartheta)\cdot B^E, \quad g_L := S\cdot B^E, \quad g_R := \begin{cases} S\cdot B^E & \text{falls } \vartheta = 0, \\ (S+1)\cdot B^E & \text{falls } 0 < \vartheta < 1. \end{cases}$$

$$x < 0 \implies x =: -(S+\vartheta)\cdot B^E, \quad g_L := \begin{cases} -S\cdot B^E & \text{falls } \vartheta = 0, \\ -(S+1)\cdot B^E & \text{falls } 0 < \vartheta < 1, \end{cases} \quad g_R := -S\cdot B^E.$$

Damit lassen sich verschiedene Arten des Rundens definieren als Zuordnung von x zu seinem Nachbarn links oder rechts.

Definition 1.3 *Rundung in* $\mathbb{G}$

$$\begin{aligned}
\text{Abrunden:} \quad & \mathrm{rd}_-\colon \mathbb{R}\to\mathbb{G}, x\mapsto g_L, \\
\text{Aufrunden:} \quad & \mathrm{rd}_+\colon \mathbb{R}\to\mathbb{G}, x\mapsto g_R, \\
\text{(Korrektes) Runden:} \quad & \mathrm{rd}_*\colon \mathbb{R}\to\mathbb{G}, x\mapsto \begin{cases} g_L & \text{falls } x \le (g_L+g_R)/2, \\ g_R & \text{falls } x \ge (g_L+g_R)/2, \end{cases} \\
\text{Abhacken:} \quad & \mathrm{rd}_0\colon \mathbb{R}\to\mathbb{G}, x\mapsto \begin{cases} g_L & \text{falls } x \ge 0, \\ g_R & \text{falls } x \le 0. \end{cases}
\end{aligned}$$

Bemerkungen:

- rd_-, rd_+, rd_0 werden auch als **gerichtetes Runden** bezeichnet.
- Im IEEE-System der Maschinenzahlen $\mathbb{M}$ sind die beiden Nachbarn g_L, g_R von x ebenfalls *immer* definiert (notfalls $\pm\infty$), somit auch diese Rundungen.
- Zusatz-Vorschrift dort: $\mathrm{rd}_*((g_L+g_R)/2)$ hat einen *geraden* Signifikand S. Das legt den unentschiedenen Fall nach einer altbewährten Rechenregel fest. Dieser Fall ist bei den Gleitpunkt-Operationen des nächsten Abschnittes durchaus nicht selten, so daß die Regel spürbare Konsequenzen hat.
- Alle vier Rundungen erfüllen als Abbildungen $\mathrm{rd}\colon \mathbb{R}\to\mathbb{G}$

$$x \in \mathbb{G} \implies \mathrm{rd}(x) = x, \qquad \text{(idempotent und surjektiv)}, \tag{1.1}$$

$$x \le y \implies \mathrm{rd}(x) \le \mathrm{rd}(y), \qquad \text{(monoton)}. \tag{1.2}$$

  Diese beiden Eigenschaften kann man benutzen, um die Rundung $\mathrm{rd}\colon A\to B$ für geordnete Mengen A und $B\subset A$ anstelle von $\mathbb{R}$ und $\mathbb{G}_{B,t}$ zu definieren.

Satz 1.4 *Schranke für den relativen Rundungsfehler in* $\mathbb{G}_{B,t}$

$$\forall x\in\mathbb{R} \text{ gilt } \mathrm{rd}(x) = x\cdot(1+\varepsilon_x) \text{ mit } |\varepsilon_x| \le \begin{cases} \frac{1}{2}B^{1-t} & \text{für korrektes Runden,} \\ B^{1-t} & \text{für gerichtetes Runden.} \end{cases}$$

$$\varepsilon_x \text{ ist der } \textbf{relative Fehler}\text{: } \varepsilon_x = \begin{cases} (\mathrm{rd}(x)-x)/x & \text{falls } x\neq 0, \\ 0 & \text{falls } x = 0. \end{cases}$$

Standard für die binäre Gleitpunkt-Arithmetik von Mikroprozessoren genau dieses Ergebnis. Und zwar:

- Für alle arithmetischen Grundoperationen einschließlich der Quadratwurzel,
- für alle Genauigkeitsstufen (*single, double, extended precision*),
- wahlweise für alle vier Rundungsvorschriften rd_+, rd_-, rd_*, rd_0.

Eine Kombination mit Satz 1.4 ergibt folgenden

Satz 1.5 *Rundungsfehler-Schranken für die ideale Arithmetik*

Falls $a \overset{\bullet}{*} b = \mathrm{rd}(a * b)$, dann gilt $\forall a, b \in \mathbb{G}_{B,t}$

$$\begin{aligned} a \overset{\bullet}{+} b &= (a+b)\cdot(1+\alpha), \\ a \overset{\bullet}{\times} b &= (a\times b)\cdot(1+\mu), \\ a \overset{\bullet}{-} b &= (a-b)\cdot(1+\sigma), \\ a \overset{\bullet}{/} b &= (a/b)\cdot(1+\delta), \quad b\neq 0. \end{aligned}$$

α, σ, μ, δ hängen von a, b ab und sind im Betrag beschränkt:

$$|\alpha|, |\sigma|, |\mu|, |\delta| \leq \varepsilon_{\mathrm{mach}} = \begin{cases} \frac{1}{2}B^{1-t} & \text{für } \mathrm{rd}_* \\ B^{1-t} & \text{für } \mathrm{rd}_+, \mathrm{rd}_-, \mathrm{rd}_0. \end{cases}$$

$\varepsilon_{\mathrm{mach}}$ heißt **Maschinengenauigkeit**.

Beispiel: $B = 10$, $t = 4$, rd_*, $\varepsilon_{\mathrm{mach}} = 0.0005$.

a	b	$a \times b$	$a \overset{\bullet}{\times} b$	μ
20.29	49.31	1000.4999	1000	$-0.4999/1000.4999$
28.75	34.80	1000.5000	1000 oder 1001	$\pm 0.5/1000.5$
20.71	48.31	1000.5001	1001	$0.4999/1000.5001$

Die Quadratwurzel kann mit einem Divisions-ähnlichen Prozeß von der Hardware berechnet werden. Das geht so schnell und genau wie bei der Division:

$$\overset{\bullet}{\sqrt{a}} = \sqrt{a} \cdot (1 + \omega) \qquad \text{mit} \quad |\omega| \leq \varepsilon_{\text{mach}}.$$

Bei einer Software-Berechnung ist die Fehlerschranke meist geringfügig größer.

Zu den Grundoperationen gehören auch die arithmetischen Vergleiche $<$, $\leq$, $=$, $\neq$, $\geq$, $>: \mathbb{G} \times \mathbb{G} \to \{\text{ja, nein}\}$. Sie können und sollten immer fehlerfrei implementiert sein, sie dürfen auch nie zu einer Bereichs-Überschreitung führen. Das verbietet eine Implementierung, welche $a - b$ berechnet und dies mit 0 vergleicht.

Eine erstklassige Maschinen-Arithmetik verwendet im Rechenwerk und in einigen Hilfsregistern Signifikanden und Exponenten, die um etliche Schutzziffern verlängert sind. Dafür gedacht ist das IEEE-Format *extended precision*, wie implementiert in den Koprozessoren Intel 80387 und MC 68882. Auf diese Weise können zusammengesetzte arithmetische Ausdrücke wie zum Beispiel $\sqrt{a^2 + b^2}$ ohne Gefahr von Bereichs-Überschreitung und mit praktisch nur einem Rundungsfehler berechnet werden. Das gibt höchste Genauigkeit bei den Ein/Ausgabe-Konvertierungen, den Grundoperationen für komplexe Zahlen und bei den transzendenten Standardfunktionen.

Die Gleitpunkt-Arithmetik läßt sich ergänzen durch eine **Intervall-Arithmetik**. In der einfachsten Implementierung sind alle Operanden kompakte Intervalle:

$A := [a_1, a_2]$ mit $-\infty < a_1 \leq a_2 < \infty$ und $B := [b_1, b_2]$ mit $-\infty < b_1 \leq b_2 < \infty$, die nach dem „Jeder-Wert-Prinzip“ miteinander verknüpft werden:

$$\begin{aligned} A * B &:= \{a * b : a \in A,\ b \in B\} \qquad \text{für} \quad * \in \{+, -, \times, /\}, \\ A^2 &:= \{a^2 : a \in A\} \qquad \neq A \times A, \\ \sqrt{A} &:= \{\sqrt{a} : a \in A\}, \qquad \text{usw.} \end{aligned}$$

Wegen der Monotonie der Grundoperationen $+$, $-$, $\times$, $/$ entstehen dabei wieder Intervalle, deren Endpunkte unmittelbar aus den Endpunkten der Operanden berechnet werden können ($0 \notin B$ für A/B). Bei der maschinellen Ausführung ist gerichtet zu runden: rd_- für den linken Endpunkt und rd_+ für den rechten Endpunkt. Die arithmetischen Vergleiche kennen neben der gewöhnlichen Trichotomie von ‚$<$‘, ‚$=$‘ und ‚$>$‘ noch als vierte Möglichkeit ‚?‘ („unvergleichbar“), wenn Intervalle teilweise überlappen.

Mittels der Intervall-Arithmetik kann man **Einschließungs-Algorithmen** erstellen. Hier sind die Eingabe-Daten Intervalle, die alle zulässigen oder möglichen Daten einschließen. Im Laufe des Algorithmus werden daraus nach obiger Regel neue Intervalle berechnet, die nach ihrer Einrichtung dann Einschließungs-Intervalle sind für die möglichen Zwischen- und Endresultate. Rundungsfehler werden dabei rigoros miterfaßt. Wenn die Eingabe-Intervalle zu Punkten degeneriert sind, werden sogar ausschließlich Rundungsfehler verfolgt.

Die Standard-Algorithmen der Numerik sind wenig geeignet für die Ausführung in Intervall-Arithmetik, weil sie ihre hohe Genauigkeit dadurch erzielen, daß sie

sen sich besonders leicht bewerten, indem man sie vergleicht mit den fast stets vorhandenen Fehlern in den Eingabe-Werten.

Die schwache Hypothese 1.7 erlaubt nur die Rückwärts-Interpretation. Dagegen erlaubt die starke Hypothese 1.6, die Rundungsfehler in beiden Richtungen eines Rechenganges zu interpretieren:

- **Vorwärts-Interpretation:** Das berechnete Resultat hat immer einen kleinen relativen Fehler ε mit $|\varepsilon| \leq \varepsilon_{\text{mach}}$
- **Rückwärts-Interpretation:** Das berechnete Resultat ist exakt für relativ wenig geänderte Operanden, nämlich

$a \cdot (1+\alpha)$ und $b \cdot (1+\alpha)$	bei der Addition,
$a \cdot (1+\sigma)$ und $b \cdot (1+\sigma)$	bei der Subtraktion,
$a \cdot \sqrt{1+\mu}$ und $b \cdot \sqrt{1+\mu}$	bei der Multiplikation (oder eine andere Aufteilung),
$a \cdot \sqrt{1+\delta}$ und $b/\sqrt{1+\delta}$	bei der Division (oder eine andere Aufteilung).

Fast immer wird die starke Hypothese 1.6 als Axiom für die *a priori* Rundungsfehler-Analyse gewählt, denn sie umfaßt sowohl die ideale Maschinen-Arithmetik als auch die meisten real existierenden und ist außerdem leicht zu handhaben. Bei der Rückwärts-Analyse kommt man oft sogar mit der schwachen Hypothese 1.7 aus, so daß praktisch alle Maschinen erfaßt werden.

Man darf daraus keinesfalls folgern, daß es auf die Qualität der Gleitpunkt-Arithmetik gar nicht ankommt. Vielmehr ist die ideale Gleitpunkt-Arithmetik aus mehreren Gründen wichtig:

- Sie erfüllt außer der starken Hypothese 1.6 eine Reihe von Gesetzmäßigkeiten und bietet dadurch dem Programmierer Möglichkeiten zur Minimierung der Rundungsfehler.
- Rundungsfehler treten bei ihr nur auf, wo sie unvermeidbar sind und deshalb erwartet werden, so daß es keine unvorhersehbaren Effekte geben kann.
- Maßnahmen zur Bekämpfung der Absonderlichkeiten einer Pseudo-Rundung müssen für jeden Maschinentyp individuell erfunden werden. Diese ruinieren die Verständlichkeit eines Programmes für Außenstehende.
- Wegen der guten Ausmittelung vieler Rundungsfehler sind bei der idealen Gleitpunkt-Arithmetik die aktuellen Fehler meist viel kleiner als die Fehlerschranken der Rundungsfehler-Analyse, die ja überall den ungünstigsten Fall unterstellt. Bei einer Pseudo-Rundung liegen Ist-Fehler und Fehlerschranke oft viel näher beisammen.

1.4 Fehlerfortpflanzung, Kondition

Die in der Numerik behandelten Probleme haben folgende abstrakte Beschreibung. Sie ist notwendig, um daran anschließend den Begriff der Kondition eines Problems allgemein gültig formulieren zu können.

Definition 1.9

Zulässige Eingabe-Daten:	Ein n-Tupel x reeller Zahlen: $x \in \mathbb{D} \subseteq \mathbb{R}^n$.
Resultat:	Ein m-Tupel y reeller Zahlen: $y \in \mathbb{R}^m$.
Problem:	Eine Funktion (Abbildung) $p: \mathbb{D} \to \mathbb{R}^m$, die allen zulässigen Eingabe-Daten das eindeutige Resultat zuordnet: $y = p(x)$. p ist eine vektorwertige Funktion mit $y_i = p_i(x_1, \ldots, x_n), \quad i = 1, \ldots, m.$

Komplexe Eingabe-Daten und Resultate können dabei als Paare reeller Größen angesehen werden. Die endliche Dimension bringt zum Ausdruck, daß Maschinen immer nur finite Datenmengen verarbeiten können. Drei Beispiele für solche Probleme:

Problem 1: Auflösen eines linearen Gleichungssystems $Ax = b$ nach x.

Eingabe-Daten:	$A \in \mathbb{R}^{n,n},\ b \in \mathbb{R}^n,$ (oder $\mathbb{C}$)
Resultat:	$x \in \mathbb{R}^n$.

Problem 2: Nullstellen eines Polynoms P in der üblichen Koeffizienten-Darstellung (Lösen einer algebraischen Gleichung).

Eingabe-Daten:	Koeffizienten $c = (c_0, \ldots, c_n) \in \mathbb{C}^{n+1},\ c_n \neq 0,$ so daß $P(z) = c_0 + c_1 z + \ldots + c_n z^n$.
Resultat:	Eine oder alle Nullstellen : $z,$ so daß $P(z) = 0$.

Problem 3: Eigenwerte λ und eventuell zugehörige Eigenvektoren x der Matrix A.

Eingabe-Daten:	$A \in \mathbb{C}^{n,n}$
Resultat:	$\lambda \in \mathbb{C}$ und evtl. $x \in \mathbb{C}^n$, so daß $Ax = \lambda x$ und $x \neq 0$.

Definition 1.10

Kondition des Problems:	Empfindlichkeit des Resultats gegenüber Änderungen der Eingabe-Daten.
Gut konditioniert:	„kleine“ δx geben „kleine“ δy.
Schlecht konditioniert:	„kleine“ δx geben „große“ δy.
Konditions-Zahlen:	$\left\lvert \frac{\partial p_i}{\partial x_k} \right\rvert$, $\left\lVert \frac{\partial p}{\partial x} \right\rVert$, $\left\lvert \frac{x_k}{y_i} \frac{\partial p_i}{\partial x_k} \right\rvert, \ldots$

Die Kondition eines Problems ist belanglos, wenn die Eingabe-Daten keinen Fehler haben, wie das etwas Praxis-fern in der reinen Mathematik meist unterstellt wird. Umso wichtiger ist dieser Begriff und eine Auseinandersetzung damit für jeden Anwender, weil er praktisch nie fehlerfreie Eingabe-Daten zur Verfügung stellen kann. (Nur wenn die Menge der Eingabe-Daten leer ist, kann man sie mit gutem Gewissen als fehlerfrei ansehen, aber solche Probleme — z.B. die Berechnung mathematischer Konstanten — sind ein für alle mal lösbar und aus diesem Grund für den Anwender sowieso kein lohnendes Problem.)

Auch die vier arithmetischen Grundoperationen sind nach Definition ein Problem und haben somit eine Kondition ($\varrho x := \delta x / x$):

$$
\begin{array}{rclrcl}
\delta(a \pm b) & = & \delta a \pm \delta b, & \varrho(a \pm b) & = & \varrho a \cdot \dfrac{a}{a \pm b} + \varrho b \cdot \dfrac{b}{a \pm b}, \\
\delta(a \cdot b) & = & b \cdot \delta a + a \cdot \delta b, & \varrho(a \cdot b) & = & \varrho a + \varrho b, \\
\delta(a/b) & = & \delta a/b - a \cdot \delta b/b^2, & \varrho(a/b) & = & \varrho a - \varrho b, \\
\delta(\sqrt{a}) & = & \delta a/(2\sqrt{a}), & \varrho(\sqrt{a}) & = & \frac{1}{2} \varrho a,
\end{array}
$$

Insbesondere sieht man, daß die relativen Konditionszahlen für Multiplikation, Division und Quadratwurzel 1 bzw. $\frac{1}{2}$ sind, so daß diese Operationen als gut konditioniert bezeichnet werden. Bei der Addition und Subtraktion sind die absoluten Konditionszahlen klein, aber die relativen Konditionszahlen sind unbegrenzt. Kleine relative Fehler in den Operanden geben also im allgemeinen keinen kleinen relativen Fehler in der Summe bzw. Differenz. In diesem (und *nur* in diesem) Sinne ist die sogenannte **Auslöschung** (führender Ziffern) eine gefährliche Situation.

Die Kondition linearer Gleichungssysteme (Problem 1) wird in Kapitel 3 behandelt. Die Kondition der Nullstellen eines Polynoms (Problem 2) wird als Übungsaufgabe empfohlen, wobei man sich auf einfache Nullstellen beschränken kann. Dazu sei an das implizite Funktionstheorem erinnert: Wenn das Resultat y eine Gleichung $f(x, y) = 0$ erfüllt, ist die Funktion p implizit definiert durch $f(x, p(x)) = 0 \quad \forall x \in \mathbb{D}$. Unter bekannten Voraussetzungen gilt dann $f_x + f_y \cdot \partial p / \partial x = 0$, also $\partial p / \partial x = -f_y^{-1} f_x$.

Schlecht konditionierte Probleme sind meist auch numerisch nur ungenau zu lösen, denn die Rundungsfehler verfälschen das Resultat dann ebenfalls in starkem Maße. Das ist aber nicht die echte Schwierigkeit, denn dagegen könnte man sich wehren,

(Analog für relative Fehler ρx, ρz, ρy, ...) Die Dinge wären unproblematisch, wenn ein gut konditioniertes Gesamtproblem nur Zerlegungen in gut konditionierte Teilprobleme besäße. Leider ist das nicht der Fall, es ist vielmehr durchaus möglich, daß $p' = r'q'$ klein ist, aber r' sehr groß. In einem solchen Fall ist die Zerlegung unbrauchbar.

Das obige Problem 3 mit seiner naheliegenden Zerlegung wurde eingeführt, um diese Schwierigkeit zu demonstrieren. Für eine symmetrische Matrix sind alle Eigenwerte perfekt konditioniert, d.h. kleine Unsicherheiten in den Matrix-Elementen geben ebenso kleine Unsicherheiten in den Eigenwerten (vergleiche Abschnitt 7.2). Die Nullstellen eines Polynoms in der Koeffizienten-Darstellung sind dagegen notorisch schlecht konditioniert.

> Die Berechnung der Eigenwerte als Nullstellen des charakteristischen Polynoms in der Koeffizienten-Darstellung ist deshalb ein schwerer Kunstfehler.

Zahlenbeispiel: Sei A die symmetrische 12×12 Matrix mit $1, 2, \ldots, 12$ in der Hauptdiagonalen und 1 in den beiden Nebendiagonalen, alle übrigen Elemente 0. Hier sind die Koeffizienten ihres charakteristischen Polynoms: (a) exakt und (b) gerundet auf 24 Bit (also Fehler maximal eine halbe Einheit im letzten Bit). In den letzten beiden Spalten finden sich die zugehörigen exakten Nullstellen.

k	(a)	(b)	zu (a)	zu (b)
0	116 601 199	116 601 200		
1	−711 234 043	−711 234 048	0.253 805 817 ...	0.253 805 802 ...
2	1 227 435 449	1 227 435 392	1.789 321 352 ...	1.789 342 507 ...
3	−1 053 129 168	−1 053 129 152	2.961 058 880 ...	2.960 859 724 ...
4	540 869 521	540 869 504	3.996 047 997 ...	3.988 928 696 ...
5	−181 301 679	−181 301 680	4.999 774 323 ...	5.132 691 281 ...
6	41 449 694	41 449 696	5.999 992 046 ...	5.615 392 728 ...
7	−6 590 064	−6 590 064	7.000 007 953 ...	7.349 137 838 ... $\pm$
8	729 048	729 048	8.000 225 676 ...	0.827 589 263 ... i
9	−55 055	−55 055	9.003 952 002 ...	9.696 317 177 ... $\pm$
10	2 706	2 706	10.038 941 119 ...	0.821 801 891 ... i
11	−78	−78	11.210 678 647 ...	11.453 800 408 ...
12	1	1	12.746 194 182 ...	12.714 268 818 ...

Der theoretisch kleinst-mögliche Rundungsfehler bei der Ausführung von Teilschritt 1 in IEEE *single precision* wird von Teilschritt 2 so verstärkt, daß insgesamt nur noch Schrott resultiert!

1.5 Akzeptables Resultat, numerisch stabiler Algorithmus

Die Eingabe-Daten $x \in \mathbb{D} \subseteq \mathbb{R}^n$ für das Problem $p\colon \mathbb{D} \to \mathbb{R}^m$ sollen eine Toleranz haben, beschrieben durch eine Umgebung $U_x \subset \mathbb{R}^n$. Zum Beispiel

$$U_x = \{\tilde{x} \in \mathbb{D} : |\tilde{x} - x| \leq \varepsilon |x|\} \qquad \text{oder} \qquad U_x = \{\tilde{x} \in \mathbb{D} : \|\tilde{x} - x\| \leq \varepsilon \|x\|\}.$$

Damit liegt auch das Resultat $y = p(x)$ nur fest bis auf eine bestimmte Toleranz, nämlich

$$V_y := p(U_x) = \{p(\tilde{x}) : \tilde{x} \in U_x\}.$$

Es ist dann jedes $\tilde{y} \in V_y$ gleichberechtigt mit y. F.L. Bauer (1965) nennt solche gleichberechtigten $\tilde{y}$ akzeptabel. In der Praxis hat sich eine geringfügige Abschwächung als notwendig erwiesen:

Definition 1.11

Eine Näherung $\tilde{y}$ für das exakte Resultat $y = p(x)$ heißt **akzeptabel** bezüglich der Eingabe U_x, wenn

$$\begin{aligned} &\text{streng:} && \tilde{y} \in V_y, && (1.3)\\ &\text{schwächer:} && \exists y' \in V_y : \ |\tilde{y} - y'| \leq O(\varepsilon_{\text{mach}})|\tilde{y}|, && (1.4)\\ &\text{oder:} && \exists y' \in V_y : \ \|\tilde{y} - y'\| \leq O(\varepsilon_{\text{mach}})\|\tilde{y}\|. \end{aligned}$$

Eine akzeptable Näherung muß sich also interpretieren lassen als exakt zu Eingabe-Daten, die im Rahmen der gegebenen Toleranz geeignet abgeändert worden sind, oder etwas schwächer: sie darf von einer solchen Näherung y' höchstens um wenige Einheiten in der letzten Stelle abweichen.

Sehr häufig ist es mit wenig Aufwand möglich, durch eine Kontrollrechnung zu entscheiden, ob eine vorgelegte Näherung $\tilde{y}$ akzeptabel ist. Zum Beispiel für obiges Problem 1: Gegeben $\Delta A \geq 0$, $\Delta b \geq 0$, A, b und $\tilde{x}$. Dieses $\tilde{x}$ ist also im strengen Sinn akzeptabel, wenn es ein $\tilde{A}$ und $\tilde{b}$ gibt, so daß

$$\tilde{A}\tilde{x} = \tilde{b} \quad \text{und} \quad |\tilde{A} - A| \leq \Delta A \quad \text{und} \quad |\tilde{b} - b| \leq \Delta b.$$

Das ist äquivalent zu der unmittelbar überprüfbaren Aussage (Prager und Oettli)

$$|A\tilde{x} - b| \leq \Delta A|\tilde{x}| + \Delta b.$$

Die Richtung „$\Rightarrow$“ sollte klar sein.

Für „$\Leftarrow$“ betrachte man die Wahl
$\vartheta_i := (A\tilde{x} - b)_i / (\Delta A|\tilde{x}| + \Delta b)_i$, $\tilde{a}_{ij} := a_{ij} - \vartheta_i \cdot \Delta a_{ij} \cdot \operatorname{sign}(x_j)$, $\tilde{b}_i := b_i + \vartheta_i \cdot \Delta b_i$.

Ähnlich kann man vorgehen bei Problem 2 (einzelne oder alle Nullstellen) und bei Problem 3 (Eigenwert und Eigenvektor).

- Um zu zeigen, daß ein Algorithmus instabil ist, genügt ein einziges Gegenbeispiel. Um zu zeigen, daß ein Algorithmus numerisch stabil ist, muß man neben den Verfahrensfehlern auch die Rundungsfehler abschätzen. Dazu dient meistens die Rückwärts-Analyse.

Übungsaufgabe 1.1: Die positive Wurzel von $x^2 + 2px - q$, $p > 0$, $q > 0$, ist $\lambda = \sqrt{p^2 + q} - p$. Der Algorithmus sei: $w := \sqrt{p^2 + q}$; $\lambda := w - p$. Man zeige, daß jeder Schritt für sich numerisch stabil ist, jedoch beide Schritte zusammen keinen numerisch stabilen Algorithmus ergeben. (Das berechnete $\tilde{\lambda}$ ist nicht nahe an einem λ zu leicht geänderten Daten $\tilde{p}$ und $\tilde{q}$, wenn $q \ll p$.) Was gilt bei abgeändertem zweiten Schritt $\lambda := q/(w + p)$?

heißt die Matrix **quadratisch** und n ist ihre **Ordnung**.

Es folgt eine Liste von Bezeichnungen.

Zur Numerierung von Vektoren dienen Hochindizes, weil Verwechselungen mit Exponenten nicht eintreten dürften und eine Unterscheidung von Komponenten (Tiefindizes) zweckmäßig ist. $\operatorname{span}\{x^1, \ldots, x^m\}$ ist der von $x^1, \ldots, x^m$ aufgespannte Untervektorraum (Menge aller Linearkombinationen, lineare Hülle).

$e^1, \ldots, e^n$ bezeichnet die Spalten der $n \times n$ Einheits-Matrix (Identität) I_n, oft kurz I statt I_n.

Eine Matrix heißt **diagonal**, wenn alle Außerdiagonal-Elemente null sind, die platzsparende Schreibweise ist $\operatorname{diag}(d_1, \ldots, d_n)$. T heißt **tridiagonal** wenn $t_{i,j} = 0$ für $|i - j| > 1$.

Eine **untere (linke) Dreiecksmatrix** L hat Nullen oberhalb der Diagonale:

$$l_{ij} = 0 \qquad \text{für} \quad i < j.$$

Falls $l_{11} = \ldots = l_{nn} = 0$: **strikt untere** Dreiecksmatrix,
falls $l_{11} = \ldots = l_{nn} = 1$: **normalisierte** untere Dreiecksmatrix.

Eine **obere (rechte) Dreiecksmatrix** R hat Nullen unterhalb der Diagonalen:

$$r_{ij} = 0 \qquad \text{für} \quad i > j.$$

$\det(A)$ bezeichnet die Determinante einer quadratischen Matrix:

$$\det(A) := \sum_P (-)^P a_{1,P1} \cdots a_{n,Pn}.$$

Die Summe geht über alle $n!$ Permutationen $P1, \ldots, Pn$ der Zahlen $1, \ldots, n$ und $(-)^P$ ist ± 1, je nachdem ob die Permutation gerade oder ungerade ist, also durch eine gerade oder ungerade Anzahl von paarweisen Tauschs in die natürliche Reihenfolge $1, \ldots, n$ zurückgebracht werden kann. Wichtigste Regel ist

$$A, B \in \mathbb{C}^{n,n} \qquad \Longrightarrow \qquad \det(AB) = \det(BA) = \det(A)\det(B).$$

Falls $\det(A) \neq 0$, heißt A **nicht-singulär** und es gibt genau ein B, so daß $AB = BA = I$. B heißt die **Inverse** von A und wird mit A^{-1} bezeichnet.

$$\begin{aligned} D \text{ diagonal} \quad &\Longrightarrow \quad \det(D) = d_1 \ldots d_n. \\ L \text{ untere Dreiecksmatrix} \quad &\Longrightarrow \quad \det(L) = l_{11} \ldots l_{nn}. \\ R \text{ obere Dreiecksmatrix} \quad &\Longrightarrow \quad \det(R) = r_{11} \ldots r_{nn}. \end{aligned}$$

Durch Spiegelung der Matrix $A \in \mathbb{C}^{m,n}$ an ihrer Diagonale (die i-i-Positionen) entsteht die **transponierte** Matrix $A^T \in \mathbb{C}^{n,m}$, also

$$(A^T)_{ij} := a_{ji}, \qquad i = 1, \ldots, n \text{ und } j = 1, \ldots, m.$$

Das sind jeweils Abbildungen $\|\cdot\|: \mathbb{C}^n \to \mathbb{R}$ mit den drei **Eigenschaften**:

$$\textbf{definit, d.h.} \qquad x \neq 0 \quad \Longrightarrow \quad \|x\| > 0, \tag{2.1}$$

$$\textbf{homogen, d.h.} \qquad \|\lambda x\| = |\lambda| \cdot \|x\| \quad \forall \lambda \in \mathbb{C}, \tag{2.2}$$

$$\textbf{sub-additiv, d.h.} \qquad \|x + y\| \leq \|x\| + \|y\|, \tag{2.3}$$

(Dreiecks-Ungleichung)

Der Nachweis ist elementar mit Ausnahme von (2.3) für die Euklidsche Norm. Hier benötigt man als Hilfsmittel die

$$\textbf{Cauchy-Schwarzsche Ungleichung} \qquad |x^H y| \leq \|x\|_2 \cdot \|y\|_2,$$

die wegen der Homogenität nur für $\|x\|_2 = \|y\|_2 = 1$ und $x^H y \geq 0$ bewiesen werden muß: benutze $\|x - y\|_2^2 \geq 0$.

Die Eigenschaften (2.1), (2.2), (2.3) können benutzt werden, um Normen $\|\cdot\|: \mathbb{C}^n \to \mathbb{R}$ ganz allgemein zu definieren.

Die Wahl $\lambda = 0$ in (2.2) gibt eine Ergänzung zu (2.1):

$$\|0\| = 0 \qquad \text{oder} \qquad x \neq 0 \quad \Longleftrightarrow \quad \|x\| > 0.$$

Wegen $\|x\| = \|(x \pm y) \mp y\| \leq \|x \pm y\| + \|y\|$ läßt sich die Dreiecks-Ungleichung (2.3) auch in die Form bringen

$$\|x\| - \|y\| \leq \|x \pm y\| \leq \|x\| + \|y\|.$$

Insbesondere

$$\|y\| \leq \varepsilon \quad \Longrightarrow \quad \Big|\|x + y\| - \|x\|\Big| \leq \varepsilon.$$

Normen erlauben, den Abstand zwischen zwei Punkten des Vektorraums zu definieren, also eine Metrik einzuführen bzw. den Raum zu topologisieren, d.h. Umgebungen zu definieren. Die Menge

$$\{x \in \mathbb{C}^n : \|x\| \leq 1\}$$

heißt **Normkugel**, ohne im allgemeinen besonders rund zu sein. Skizze für $\|\cdot\|_1$ und $\|\cdot\|_\infty$ im $\mathbb{R}^2$ und $\mathbb{R}^3$!

In endlich-dimensionalen Vektorräumen sind alle **Normen äquivalent**, d.h. zu jeder Norm $\|\cdot\|$ im $\mathbb{C}^n$ gehören zwei *positive* Konstanten α_n, β_n, so daß

$$\alpha_n \|x\|_\infty \leq \|x\| \leq \beta_n \|x\|_\infty \quad \forall x \in \mathbb{C}^n.$$

Übungsaufgabe 2.1: Zeige

$$1 \leq \frac{\|x\|_1}{\|x\|_2} \leq \sqrt{n} \qquad \text{und} \qquad 1 \leq \frac{\|x\|_2}{\|x\|_\infty} \leq \sqrt{n} \quad \forall x \in \mathbb{C}^n \backslash \{0\}.$$

Die Eigenschaften (2.7) und (2.8) sind der große Vorzug der Operatornorm gegenüber einer Vektornorm, angewendet auf die Matrix-Elemente, und sie wird dadurch der Funktion einer Matrix als Abbildung viel besser gerecht. Zum Beispiel ist $\|I\| = 1$ ein naheliegender Wunsch.

Satz 2.3 *Drei Operatornormen für $m \times n$ Matrizen*

$$
\begin{array}{lcll}
\|\cdot\|_1 & \Longrightarrow & \|A\|_1 & = \max_j \sum_i |a_{ij}| \\
 & & & \text{größte Spaltenbetragssumme,} \\
\|\cdot\|_\infty & \Longrightarrow & \|A\|_\infty & = \max_i \sum_j |a_{ij}| \\
 & & & \text{größte Zeilenbetragssumme,} \\
\|\cdot\|_2 & \Longrightarrow & \|A\|_2 & = \sqrt{\lambda_1}, \\
 & & & \text{wobei } \lambda_1 \geq \lambda_2 \geq \ldots \geq 0 \text{ die} \\
 & & & \text{absteigend geordneten Eigenwerte der} \\
 & & & \text{positiv-(semi) definiten Matrix } A^H A \text{ sind.} \\
 & & & \|\cdot\|_2 \text{ heißt } \textbf{Spektralnorm}.
\end{array}
$$

Der einfache Beweis sei als Übungsaufgabe empfohlen. (Für $\|A\|_2$ sind geringe Vorkenntnisse zum symmetrischen Eigenwertproblem erforderlich, vgl. Kapitel 7.)

Während $\|A\|_1$ und $\|A\|_\infty$ billig zu berechnen sind, kommt $\|A\|_2$ für den praktischen Gebrauch nicht in Frage, weil zu teuer.

Definition 2.4 *Konditionszahl*

$$\kappa(A) := \frac{\max\limits_{\|x\|=1} \|Ax\|}{\min\limits_{\|x\|=1} \|Ax\|}.$$

Sie gibt an, wie weit die Normkugel unter der Abbildung A verzerrt wird. Im Idealfall ist $\kappa = 1$. Dazu gehören zum Beispiel die unitären Matrizen bei Verwendung der Euklidschen Norm. Umgekehrt ist $\kappa(A) = \infty$, wenn $Ax = 0$ für ein $x \neq 0$. Dann ist die Abbildung sicherlich nicht umkehrbar. Wenn dagegen A^{-1} existiert, dann ist

$$1/\min_{\|x\|=1} \|Ax\| = \max_{x \neq 0} \|x\|/\|Ax\| = \max_{y \neq 0} \|A^{-1}y\|/\|y\| = \|A^{-1}\| \ ,$$

also Konditionszahl

$$\kappa(A) := \|A\| \cdot \|A^{-1}\|$$

(falls A^{-1} existiert).

Übungsaufgabe 2.2: Die **Frobenius-Norm** ist $\|A\|_F := \left(\sum\limits_{i=1}^{m} \sum\limits_{j=1}^{n} |a_{ij}|^2\right)^{\frac{1}{2}}$, also die Euklidsche Länge, wenn $A \in \mathbb{C}^{m,n}$ als $m \cdot n$ Vektor aufgefaßt wird. Man zeige

Skalierung D

Definition:	$D := \mathrm{diag}(d_1, \ldots, d_n)$, alle $d_i \neq 0$.
Anwendung:	$(Dx)_i = d_i x_i, \quad i = 1, \ldots, n.$ $(DA)_{ij} = d_i a_{ij}, \quad i$-te Zeile mal $d_i, \quad i = 1, \ldots, n.$ $(AD)_{ij} = a_{ij} d_j, \quad j$-te Spalte mal $d_j \quad j = 1, \ldots, n.$
Inverse:	$D^{-1} = \mathrm{diag}(1/d_1, \ldots, 1/d_n), \quad \det(D) = d_1 \ldots d_n.$
ÄT:	$A \mapsto DAD^{-1}$ mit $a_{ij} \mapsto (d_i/d_j)a_{ij}, \quad i, j = 1, \ldots, n.$

Die nächsten drei Transformations-Matrizen stimmen mit der Einheits-Matrix $I = \mathrm{diag}(1, \ldots, 1)$ überein bis auf die **Extra-Elemente** in den vier Positionen (i, i), (i, j), (j, i), (j, j), $i < j$ oder $i > j$.

Vertauschung P_{ij}

Definition:	P_{ij} hat Extra-Elemente $\begin{cases} 0 & 1 \\ 1 & 0 \end{cases}$
Anwendung:	$P_{ij}x$ vertauscht Komponenten x_i und x_j, $P_{ij}A$ vertauscht Zeilen i und j, AP_{ij} vertauscht Spalten i und j.
Inverse:	$P_{ij}^{-1} = P_{ij}$, $\det(P_{ij}) = -1$.
ÄT:	$A \mapsto P_{ij}AP_{ij}$ vertauscht erst Zeilen, dann Spalten (oder umgekehrt), insbesondere $\begin{matrix} a_{ii} & a_{ij} \\ a_{ji} & a_{jj} \end{matrix} \mapsto \begin{matrix} a_{jj} & a_{ji} \\ a_{ij} & a_{ii} \end{matrix}$ (kreuzweise!).

P_{ij} ist reell orthogonal, also auch unitär, in der Euklidschen Norm hat es die ideale Konditionszahl 1.

Satz 2.5

Die N_{ij} zu einem festen j sind miteinander vertauschbar und es gilt

$$\prod_{\substack{i=1\\ i\neq j}}^{n} N_{ij}(\alpha_i) = \begin{pmatrix} 1 & & \alpha_1 & & 0 \\ & \ddots & \vdots & & \\ & & 1 & & \\ & & \vdots & \ddots & \\ 0 & & \alpha_n & & 1 \end{pmatrix},$$

$$\left(\prod_{\substack{i=1\\ i\neq j}}^{n} N_{ij}(\alpha_i) \right)^{-1} = \begin{pmatrix} 1 & & -\alpha_1 & & 0 \\ & \ddots & \vdots & & \\ & & 1 & & \\ & & \vdots & \ddots & \\ 0 & & -\alpha_n & & 1 \end{pmatrix}.$$

Ferner gilt

$$\prod_{i>1} N_{i1}(\alpha_{i1}) \prod_{i>2} N_{i2}(\alpha_{i2}) \cdots \prod_{i>n-1} N_{i,n-1}(\alpha_{i,n-1})$$

$$= \begin{pmatrix} 1 & & & & 0 \\ \alpha_{21} & 1 & & & \\ \vdots & \vdots & \ddots & & \\ \vdots & \vdots & & 1 & \\ \alpha_{n1} & \alpha_{n2} & & \alpha_{n,n-1} & 1 \end{pmatrix}.$$

Dabei ist die Reihenfolge der $n-1$ Teilprodukte $\prod_{i>j} \cdots$ wichtig.

Spiegelung

Definition:	$T := I - 2vv^H$, wobei $v \in \mathbb{C}^n$ und $\|v\|_2 = 1$, $T = I - uu^H/\kappa$, wobei $u \in \mathbb{C}^n \backslash \{0\}$ und $\kappa := \frac{1}{2}u^H u$. Die Transformations-Matrix T ist also voll: $t_{ij} = \delta_{ij} - 2v_i\bar{v}_j = \delta_{ij} - u_i\bar{u}_j/\kappa, \quad i,j = 1,\ldots,n,$ und nicht so dünn besiedelt wie in den vorangegangenen Elementar-Operationen. Es wäre jedoch ein grober Kunstfehler, wenn man alle t_{ij} explizit formen und im Computer speichern würde. Vielmehr wird nur das normalisierte v oder das unnormalisierte u mitsamt dem reellen κ oder $1/\kappa$ gespeichert. T ist selbst-adjungiert: $T^H = T$. Der Name *Spiegelung* wird im Anschluß an das gewählte Besprechungsschema erläutert.
Anwendung:	$y = Tx = x - u \cdot (u^H x)/\kappa,$ $\begin{cases} \text{Schritt 1:} & \sigma := u^H x/\kappa \in \mathbb{C}, \\ \text{Schritt 2:} & y := x - u \cdot \sigma. \end{cases}$ $TA = A - u \cdot (u^H A/\kappa)$, jede Spalte von TA wie zuvor y, $AT = A - (Au/\kappa) \cdot u^H$, jede Zeile von AT wie zuvor. (Die Matrix A muß nicht quadratisch sein.)
Inverse:	$T^{-1} = T$, denn $T^2 = I - 2vv^H - 2vv^H + 4v(v^H v)v^H = I$ wegen $v^H v = 1$. $T = T^H = T^{-1}$ ist also involutorisch, selbstadjungiert, unitär. In der Euklidschen Norm hat es die ideale Konditionszahl 1. Determinante: $\det(T) = -1$. (Beweis-Tip: $T(v, v^1, \ldots, v^{n-1}) = (-v, v^1, \ldots, v^{n-1})$, wobei die v^i alle senkrecht zu v sind: $v^H v^i = 0$. Also $\det(T) \cdot \det(v, v^1, \ldots, v^{n-1}) = -\det(v, v^1, \ldots, v^{n-1})$.)
ÄT:	$TAT^{-1} = A - u \cdot (u^H A/\kappa) - (Au/\kappa) \cdot u^H + u \cdot (u^H Au/\kappa^2)u^H$. Bei einer selbst-adjungierten Matrix A sind wichtige Einsparungen möglich: $A = A^H \quad \Rightarrow \quad TAT^{-1} = A - uw^H - wu^H$. Dazu: Schritt 1: $w := Au$ und $\sigma := u^H w/(2\kappa)$, Schritt 2: $w := (w - u\sigma)/\kappa$, Schritt 3: $a_{ij} - u_i\bar{w}_j - w_i\bar{u}_j$, wird nur für $i \le j$ oder nur für $i \ge j$ ausgeführt wegen der Symmetrie.

Satz 2.6 *Householder Reduktion*

Für $x \neq 0$ setze

$$\begin{aligned} \zeta &:= \|x\|_2 \cdot x_1/|x_1|, \\ u &:= x + \zeta e^1 = (x_1 + \zeta, x_2, \ldots, x_n)^T, \\ \kappa &:= x^H x + \|x\|_2 \cdot |x_1|, \\ T &:= I - uu^H/\kappa. \end{aligned}$$

Dann ist $Tx = -\zeta e^1 = (-\zeta, 0, \ldots, 0)^T$ die Spiegelung von x.

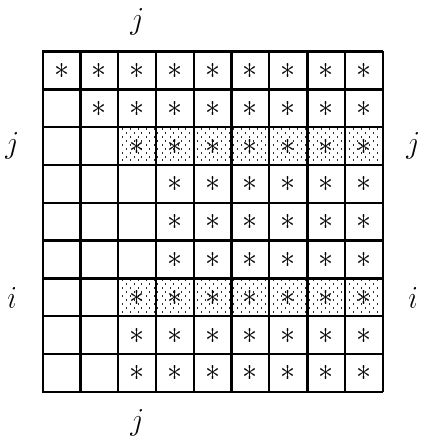
2.4 Anwendungs-Beispiele

Erstes Beispiel: Dreieckszerlegung von $A \in \mathbb{C}^{n,n}$

Sei $a_{11} \neq 0$ und $l_{2,1} := a_{2,1}/a_{1,1}$. Dann besitzt $N_{2,1}(-l_{2,1})A$ eine Null in 2-1-Position. Weitere solcher Elementar-Operationen mit geeigneten Parametern erzeugen weitere Nullen in der ersten Spalte abwärts. Danach reduziert man die zweite Spalte, usw. Beim Arbeiten in situ:

`für` $j := 1$ `bis` $n-1$:
`für` $i := j+1$ `bis` n:
    $l_{ij} := a_{ij}/a_{jj}$;
    $A := N_{ij}(-l_{ij})A$.

Gestalt von A beim i-j-Schritt:



Man speichert l_{ij} anstelle von a_{ij}.

Die End-Matrix mit lauter Nullen unterhalb der Diagonalen heiße R:

$$R = N_{n,n-1}(-l_{n,n-1}) \cdots N_{2,1}(-l_{2,1})A,$$

Drittes Beispiel: Givens' Rotation

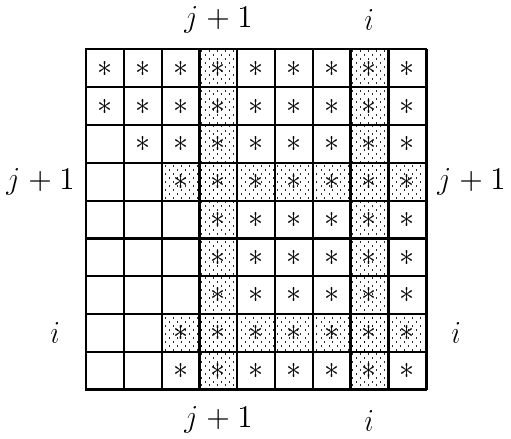
Berechnungen von Eigenwerten einer Matrix $A \in \mathbb{R}^{n,n}$ arbeiten mit Ähnlichkeits-Transformationen, weil diese die Eigenwerte nicht verändern. Eine große Vereinfachung wird erzielt, wenn man die Matrix A auf obere Fast-Dreiecksform H (obere **Hessenberg-Form**) gebracht hat. Dabei ist alles null unterhalb der ersten Subdiagonalen:

$$h_{ij} = 0 \quad \forall i > j+1.$$

Das läßt sich mit $(n-2)(n-1)/2$ ebenen Rotationen erreichen, wobei die Strategie etwas komplizierter ist wegen der beidseitigen Transformation. Zur Auslöschung von a_{ij} dient hier die Anwendung von links der Matrix $Q_{i,j+1}^T$ anstelle von Q_{ij}^T:

`für` $j := 1$ `bis` $n-2$:
`für` $i := j+2$ `bis` n:
    `Berechne` c `und` s `in` $Q_{i,j+1}$ `aus` $-sa_{j+1,j} + ca_{ij} = 0$;
    $A := Q_{i,j+1}^T A Q_{i,j+1}$.

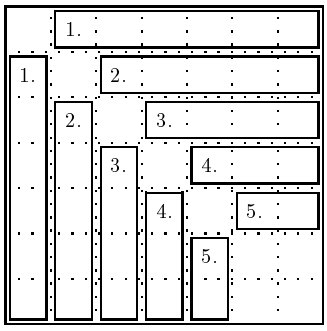
Zu Beginn des i-j-Schrittes ist die Gestalt von A:



Man überlege sich, daß eine symmetrische Ausgangs-Matrix ihre Symmetrie behält und somit (a) nur *eine* Hälfte der Matrix umgerechnet werden muß, (b) die Endmatrix tridiagonale Form hat.

Viertes Beispiel: Householder-Reduktion

Wenn man in der Reflexions-Matrix $T = I - uu^H/\kappa$ ein u wählt, dessen erste k Komponenten null sind, dann läßt die Transformation $A \mapsto TA$ die ersten k Zeilen von A unberührt: man kann sie ignorieren. Beim Übergang $A \mapsto AT$ gilt das Entsprechende für die ersten k Spalten.



Die Parkettierung bei der zweiseitigen
Householder-Reduktion
$T_{n-2} \cdots T_1 A T_1 \cdots T_{n-2} =: H$
für die unitäre Ähnlichkeits-Transformation
von A auf obere Hessenberg-Form H (hier $n = 7$).
Bei selbst-adjungiertem A ist H tridiagonal.

2.5 Orthogonalisierungs-Prozeß von Gram-Schmidt

QR-Zerlegung hat sich als kurzer Name eingebürgert für die Aufspaltung einer Matrix $A \in \mathbb{C}^{m,n}$, $m \geq n$ in einen unitären Faktor $Q \in \mathbb{C}^{m,m}$ und einen oberen Dreiecksfaktor $R \in \mathbb{C}^{m,n}$:

$$A = QR \qquad \text{bzw.} \qquad Q^H A = R \text{ mit } r_{i,k} = 0 \text{ für } i > k.$$

Zwei effiziente und numerisch stabile Algorithmen zur Gewinnung von Q und R wurden im letzten Abschnitt als Beispiele 2 und 4 vorgeführt. Dabei ergab sich Q als Produkt entweder von ebenen Rotationen $Q_{1,2}Q_{1,3}\ldots$ oder von Reflexionen $T_1 \ldots T_{n-1}$. (Nur in wenigen Situationen ist es vorteilhaft, durch Ausmultiplizieren Q *explizit* zu formen.) Diese beiden Algorithmen beweisen auch, daß die QR-Zerlegung immer möglich ist und für jeden Rang von A.

Im Falle $m > n$ kann man wahlweise auf die letzten $m - n$ Spalten von Q und die trivialen letzten $m - n$ Zeilen von R verzichten, so daß dann $A = QR$ gilt mit $Q \in \mathbb{C}^{m,n}$ und $R \in \mathbb{C}^{n,n}$. In Spaltenschreibweise, $A = (a^1, \ldots, a^n)$ und $Q = (q^1, \ldots, q^n)$, erhält man

$$a^k = q^1 r_{1,k} + \ldots + q^k r_{k,k} \qquad \text{mit} \qquad q^{iH} q^k = \delta_{i,k}.$$

Das bedeutet die gestaffelte Rekombination der Vektoren $a^1, \ldots, a^n \in \mathbb{C}^m$ zu neuen, orthonormierten Vektoren $q^1, \ldots q^n \in \mathbb{C}^m$ und zwar so, daß $\text{span}\{a^1, \ldots, a^k\} \subseteq \text{span}\{q^1, \ldots, q^k\}$ für jedes k: die Spalten von Q sind eine orthonormierte Basis für die geschachtelte Folge von Untervektorräumen, aufgespannt von der Folge $a^1, a^2, \ldots$

Die letzte Formel, umgestellt zu $q^k r_{k,k} = a^k - q^1 r_{1,k} - \ldots - q^{k-1} r_{k-1,k}$, führt direkt auf einen Algorithmus zur Produktion der q^k und der $r_{1,k}, \ldots, r_{k,k}$, da offensichtlich $r_{i,k} = q^{iH} a^k$ notwendig und hinreichend ist für $q^k \perp q^i$. Die Konvention $r_{k,k} > 0$ macht die QR-Zerlegung übrigens eindeutig.

Übrigens haben A und R als Abbildung $\mathbb{C}^n \to \mathbb{C}^m$ die gleiche Konditionszahl κ in der Euklidschen Norm:

$$\begin{aligned} \max \|Ax\|_2 / \min \|Ax\|_2 &= \max \|Rx\|_2 / \min \|Rx\|_2, \qquad (\text{überall } \|x\|_2 = 1) \\ &= \|R\|_2 \cdot \|R^{-1}\|_2 \leq \|R\|_F \cdot \|R^{-1}\|_F. \end{aligned}$$

Es ist ein weit verbreiteter Irrtum zu glauben, daß R^{-1} nur groß sein kann, wenn R mindestens *ein* kleines Diagonal-Element hat. Dafür ein Gegenbeispiel:

$$\begin{aligned} r_{k,k} &= 1, r_{i,k} = -1 \quad (i < k) \qquad \Longrightarrow \\ \|R\|_F &= \sqrt{n(n+1)/2}, \qquad \|R^{-1}\|_F = \frac{1}{3}\sqrt{4^n + 6n - 1}. \end{aligned}$$

Grundsatz 1: Die numerische Auflösung von $Ax = b$ nach x niemals mit Hilfe der Cramerschen Regel (3.5)!

Grundsatz 2: Niemals eine Matrix numerisch invertieren — immer stattdessen ein Gleichungssystem auflösen!

Beispiel: $y := BA^{-1}Cd$ für gegebenes A, B, $C \in \mathbb{R}^{n,n}$ und $d \in \mathbb{R}^n$.

Richtig: $b := Cd$; löse $Ax = b$; $y := Bx$.

Falsch: und neunmal teurer:
$X := A^{-1}$, $Y := BX$, $Z := YC$, $y := Zd$.

3.1 Kondition linearer Gleichungssysteme

Wir vergleichen x in $Ax = b$ mit $x + \delta x$ in $(A + \delta A)(x + \delta x) = b + \delta b$. Dabei soll A nicht-singulär sein und δA klein genug, daß $A + \delta A$ immer noch nicht-singulär ist. Das ist sicher der Fall, wenn

$$\|\delta A\| < 1/\|A^{-1}\|.$$

Beweis: $(A + \delta A)x = 0 \Rightarrow x = -A^{-1}\delta Ax \Rightarrow \|x\| \leq \|A^{-1}\| \cdot \|\delta A\| \cdot \|x\| \Rightarrow (1 - \|A^{-1}\| \cdot \|\delta A\|) \cdot \|x\| \leq 0.$

Das ist nur möglich für $x = 0$, denn sonst wären hier beide Faktoren > 0. Nach (3.4) ist also $A + \delta A$ nicht-singulär. (Demonstriert gleichzeitig die Überlegenheit des Kriteriums (3.4) über (3.3)!) ■

Sei also $Ax = b$ und $(A + \delta A)(x + \delta x) = b + \delta b$. Somit

$$\begin{aligned} A\delta x &= b + \delta b - (Ax + \delta A \cdot x + \delta A \cdot \delta x) = \delta b - \delta A \cdot x - \delta A \cdot \delta x, \\ \delta x &= A^{-1}(\delta b - \delta A \cdot x - \delta A \cdot \delta x), \\ \|\delta x\| &\leq \frac{\|A^{-1}\|}{1 - \|A^{-1}\| \cdot \|\delta A\|} \cdot (\|\delta b\| + \|\delta A\| \cdot \|x\|). \end{aligned}$$

Wir haben somit eine Schranke für die Änderung δx der Lösung zu gegebenen Änderungen δA, δb der Daten. Insbesondere mit der Konditionszahl
$\kappa := \|A^{-1}\| \cdot \|A\|$:

$$\begin{aligned} \frac{\|\delta A\|}{\|A\|} &\leq \varepsilon \quad \text{und} \quad \frac{\|\delta b\|}{\|b\|} \leq \varepsilon \qquad \Longrightarrow \\ \frac{\|\delta x\|}{\|x\|} &\leq \frac{\varepsilon\kappa}{1 - \varepsilon\kappa} \cdot \left(\frac{\|b\|}{\|A\| \cdot \|x\|} + 1 \right) \\ &\qquad \text{(Beachte: } \|b\| = \|Ax\| \leq \|A\| \cdot \|x\|). \end{aligned}$$

Die Konditionszahl $\kappa \geq 1$ der Matrix gibt also den Verstärkungsfaktor, mit dem beliebige Änderungen in den Daten sich auf die Änderung der Lösung auswirken.

(Der Unterschied in den beiden Residuen kann bis zu einem Faktor κ betragen.) Bei $x^{(1)}$ liegen dann *ganz starke Korrelationen in den Fehlern* der n Komponenten vor, während die Fehler von $x^{(2)}$ zwar gleich groß, aber völlig unkorreliert sind. In allen Anwendungen mit nicht exakt bekannten rechten Seite b ist $x^{(1)}$ dann wesentlich besser als $x^{(2)}$, obwohl deren absolute Fehler durchaus vergleichbar sein mögen.

Dieser Sachverhalt wird deshalb so betont, weil das numerische Standard-Verfahren, die Gauß-Elimination mit Pivotsuche, gerade eine Näherung $\tilde{x}$ vom Typ $x^{(1)}$ beschafft mit $\|r\| = O(\varepsilon_{\text{mach}} \cdot \|A\| \cdot \|\tilde{x}\|)$, also ein sehr gutes $\tilde{x}$ im wohlverstandenen Sinn. Bei der Cramerschen Regel dagegen wird jede Komponente für sich berechnet, so daß alle Fehler zufällig und unkorreliert sind: das gibt eine Näherung vom Typ $x^{(2)}$, also ein schlechtes $\tilde{x}$, wieder im wohlverstandenen Sinn. Die Cramersche Regel ist also nicht nur vom Aufwand her sehr ungeeignet, sondern auch von der Genauigkeit her ungenügend, wenn auch auf die diskutierte subtile Weise. Gleiches gilt für eine Näherung $x^{(2)}$, die gemäß (3.2) berechnet wurde, auch wenn eine sehr gute Näherung für A^{-1} oder sogar das exakte A^{-1} gegeben ist. Denn auch in (3.2) werden die Komponenten von x unabhängig berechnet und haben deshalb völlig unkorrelierte Rundungsfehler von der Größenordnung $\|A^{-1}\| \cdot \|b\| \cdot \varepsilon_{\text{mach}}$.

Weil es vielleicht unglaubwürdig klingt, sei es nochmals betont: Selbst bei bekanntem A^{-1} kann die numerische Berechnung der Lösung x von $Ax = b$ über $A^{-1}b$ zu nicht akzeptablen Resultaten führen und schlechter sein als die folgende Standard-Methode.

3.2 Die Gauß-Elimination und die Dreieckszerlegung als äquivalente Algorithmen

Jeder kennt die Gauß-Elimination oder sollte sie kennen: Man löst *eine* der n Gleichungen (3.1) nach *einer* der Unbekannten auf. Diese **explizite** Gleichung wird einerseits zur späteren Verwendung beiseite gestellt, zum anderen in die übrigen $n-1$ Gleichungen eingesetzt. So entsteht ein neues Gleichungssystem mit nur noch $n-1$ Gleichungen für nur noch $n-1$ Unbekannte. Man nennt das **Elimination** (Entfernen) der einen Unbekannten. Nach weiteren $n-2$ solchen Schritten ist man bei einem 1×1 System angekommen, das aufgelöst wird. Dann geht man durch alle zuvor beiseite gestellten expliziten Gleichungen in Rückwärts-Reihenfolge und bestimmt so nacheinander alle Unbekannten. Dieser zweite Teil heißt **Rückwärts-Substitution**.

Übrigens ist es nur eine Frage der Numerierung von Gleichungen und Unbekannten, wenn man im j-ten Eliminations-Schritt dieses Prozesses die j-te Gleichung nach der j-ten Unbekannten auflöst, $j = 1, \ldots, n-1$. Das ist die **natürliche Reihenfolge**.

Im Schema der Koeffizienten A, b beschreibt sich die Elimination von x_1 als Multiplikation der Zeile 1 mit dem Faktor $l_{ij} = a_{i1}/a_{11}$ und anschließender Subtraktion

Man beachte, daß die Elemente a_{ik} und b_i im Laufe der Eliminations-Schritte immer wieder umgerechnet werden, ohne daß dies in der Bezeichnung zum Ausdruck gekommen wäre. Die Historie des Elementes in Position (i,k) ist:

$$\begin{aligned} \text{falls } i \le k\text{:} \qquad r_{ik} &= a_{ik} - \sum_{j=1}^{i-1} l_{ij} r_{jk}, \\ \text{falls } i > k\text{:} \qquad l_{ik} &= \Big(a_{ik} - \sum_{j=1}^{k-1} l_{ij} r_{jk}\Big) / r_{kk}, \\ \text{ferner:} \qquad y_i &= b_i - \sum_{j=1}^{i-1} l_{ij} y_j. \end{aligned} \tag{3.7}$$

Man zähle die arithmetischen Operationen in der Gauß-Elimination und bestätige

je	$(n-1)n(n+1)/3 + (n-1)n/2$	Subtraktionen und Multiplikationen,
sowie	$n(n+1)/2$	Divisionen

Obige Zeilen-Operationen wurden schon als erstes Beispiel in Abschnitt 2.4 betrachtet. Es sind die sogenannten elementaren Zeilen-Operationen und sie erlauben eine dritte Beschreibung der Gauß-Elimination:

$$\begin{aligned} N_{n,n-1}(-l_{n,n-1}) \cdots N_{21}(-l_{21}) A &= R \\ N_{n,n-1}(-l_{n,n-1}) \cdots N_{21}(-l_{21}) b &= y \end{aligned}$$

Nach dem dort Gesagten kann man die Matrizen $N_{ik}(-l_{ik})$ nach rechts hinüber wälzen, so daß

$$A = LR \qquad \text{und} \qquad b = Ly, \tag{3.8}$$

mit Dreiecks-Matrizen

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

In der Tat, (3.8) lautet elementweise

$$\begin{aligned} a_{ik} &= \sum_{j=1}^{\min(i,k)} l_{ij} r_{jk} = \begin{cases} \displaystyle\sum_{j=1}^{i-1} l_{ij} r_{jk} + 1 \cdot r_{ik}, & \text{falls } i \le k, \\ \displaystyle\sum_{j=1}^{k-1} l_{ij} r_{jk} + l_{ik} r_{kk}, & \text{falls } i > k, \end{cases} \\ b_i &= \sum_{j=1}^{i-1} l_{ij} y_i + 1 \cdot y_i. \end{aligned} \tag{3.9}$$

mit $B \in \mathbb{C}^{n-1,n-1}$ und $y \in \mathbb{C}^{n-1}$ ist eine Partitionierung. Wählt man $y \neq 0$, aber sonst ganz beliebig, und dazu $\eta = -s^H y/a_{11}$, dann ist auch $x \neq 0$ und die erste Komponente von Ax verschwindet, so daß

$$0 < x^H Ax = y^H(s\eta + By) = y^H(B - s \cdot s^H/a_{11})y = y^H A' y$$

mit der Restmatrix $A' = B - ss^H/a_{11}$ nach (3.6).

Der nächste Eliminations-Schritt findet somit gleich günstige Verhältnisse vor und alle weiteren ebenso. Weil die Symmetrie erhalten bleibt, berechnet man von der Restmatrix immer nur die Elemente $i \leq k$ (oder $i \geq k$) und spart so fast 50% der Arbeit.

Die inhärente Symmetrie bringt man auch in den Dreiecks-Matrizen L und R zum Ausdruck, indem man aus R den Faktor $D = \operatorname{diag}(r_{11}, \ldots, r_{nn})$ herauszieht:

$$\textbf{Rationale Cholesky-Zerlegung:} \qquad A = LDL^H. \tag{3.10}$$

Die Diagonal-Matrix D enthält also alle Pivotelemente und diese sind positiv. Deshalb ist $D^{1/2} = \operatorname{diag}(\sqrt{r_{11}}, \ldots, \sqrt{r_{nn}})$ wohldefiniert und man kann auch schreiben

$$\textbf{Cholesky-Zerlegung:} \qquad A = \hat{L}\hat{L}^H = \hat{R}^H\hat{R}$$

mit $\hat{L} := LD^{\frac{1}{2}}$ und $\hat{R} := D^{-\frac{1}{2}}R$ zueinander adjungiert. Die Existenz der symmetrischen Faktorisierung einer positiv-definiten Matrix ist eines der nützlichsten Hilfsmittel der linearen Algebra.

Die rationale Variante (3.10) wurde früher bevorzugt, weil das Ziehen der Quadratwurzel als teuer und ungenau galt. Das Argument ist hinfällig, wenn die Quadratwurzel zu den Grund-Operationen gehört, wie z.B. bei dem IEEE-Standard 754.

3.4 Rundungsfehler-Analyse (nach W. Sautter)

In Kapitel 1 wurden zwei komplette Rundungsfehler-Analysen angekündigt. Die erste davon ist jetzt an der Reihe und betrifft die Gauß-Elimination als den wichtigsten Algorithmus der numerischen Mathematik. Nach Abschnitt 3.2 genügt es, den Einfluß der Rundungsfehler zu diskutieren bei Auswertung der Formeln:

`für` $i := 1$ `bis` n:
`für` $k := 1$ `bis` n:
$l_{ik} := \Big(a_{ik} - \sum_{j=1}^{k-1} l_{ij} r_{jk}\Big)/r_{kk}$, `falls` $i > k$;
$r_{ik} := a_{ik} - \sum_{j=1}^{i-1} l_{ij} r_{jk}$, `falls` $i \leq k$;

In Matrix-Schreibweise lautet das kompakt

$$\begin{array}{rclcrcl} A & = & \tilde{L}\tilde{R} + \delta A & \text{mit} & |\delta A| & \leq & |\tilde{L}| \cdot |\tilde{R}| \cdot \varepsilon, \\ b & = & (\tilde{L} + \delta L)\tilde{y} & \text{mit} & |\delta L| & \leq & |\tilde{L}| \cdot \varepsilon, \\ \tilde{y} & = & (\tilde{R} + \delta R)\tilde{x} & \text{mit} & |\delta R| & \leq & |\tilde{R}| \cdot \varepsilon, \end{array}$$

wobei

$$\begin{aligned} \delta a_{ik} &:= \sum_{j=1}^{\min(i,k)} \tilde{l}_{ij}\tilde{r}_{jk}\varepsilon_{ijk}, \\ \delta l_{ij} &:= \tilde{l}_{ij}\varepsilon'_{ij}, \\ \delta r_{ij} &:= \tilde{r}_{ij}\varepsilon''_{ij}. \end{aligned}$$

Die Dreieckszerlegung liefert also exakte Faktoren zu einer Nachbar-Matrix $A + \delta A$ anstelle A mit einer rigorosen Schranke für δA, deren Zutaten bei Ausführung der Rechnung zur Verfügung stehen. Entsprechendes gilt für die Vorwärts- und Rückwärts-Substitution. Schließlich

$$b = (\tilde{L} + \delta L)\tilde{y} = (\tilde{L} + \delta L) \cdot (\tilde{R} + \delta R)\tilde{x} =: \tilde{A}\tilde{x},$$

wobei

$$\begin{aligned} |\tilde{A} - A| &= |\tilde{L}\tilde{R} + \delta L \cdot \tilde{R} + \tilde{L} \cdot \delta R + \delta L \cdot \delta R - A| \\ &\leq |\delta A| + |\delta L| \cdot |\tilde{R}| + |\tilde{L}| \cdot |\delta R| + |\delta L| \cdot |\delta R| \\ &\leq |\tilde{L}| \cdot |\tilde{R}| \cdot (3\varepsilon + \varepsilon^2) =: \Delta A \end{aligned}$$

Wieder gilt: Das berechnete $\tilde{x}$ ist die exakte Lösung zu einer geänderten Koeffizienten-Matrix $\tilde{A}$ und für $|\tilde{A} - A|$ gibt es eine rigorose Schranke ΔA mit lauter bekannten Zutaten.

Schließlich gilt für das Residuum

$$|r| = |b - A\tilde{x}| = |\tilde{A}\tilde{x} - A\tilde{x}| \leq |\tilde{A} - A| \cdot |\tilde{x}| \leq \Delta A \cdot |\tilde{x}|.$$

Insgesamt liegt also eine typische Rückwärts-Analyse der Rundungsfehler vor, also eine Abschätzung des Einflusses der Rundungsfehler auf der Eingabe-Seite. Man muß die hergeleitete Schranke ΔA oder $\Delta A \cdot |\tilde{x}|$ mit Angaben über die Unsicherheit von A oder b vergleichen und kann dann entscheiden, ob die berechnete Lösung $\tilde{x}$ akzeptabel ist.

Die bewiesenen Schranken reichen nicht aus, um der Gauß-Elimination a priori das Prädikat „numerisch stabil“ zu verleihen. Dazu müßte obiges ΔA die Größenordnung von $\varepsilon_{\text{mach}}|A|$ haben und das trifft nicht bedingungslos zu, weil $|\tilde{L}| \cdot |\tilde{R}| \gg |\tilde{L}\tilde{R}| \approx |LR| = |A|$ möglich ist. Im nächsten Abschnitt wird diskutiert, was man tun kann, um diese Situation zu vermeiden.

In der vorliegenden Analyse wurden keine Normen zum Abschätzen benutzt und keine Annahme über die Skalierung der Matrix oder ihrer Zeilen oder ihrer Spalten

Zur Demonstration betrachten wir die 2×2 Matrix

$$A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}, \quad L = \begin{pmatrix} 1 & 0 \\ c/a & 1 \end{pmatrix}, \quad R = \begin{pmatrix} a & b \\ 0 & d - bc/a \end{pmatrix},$$

$$|LR| = \begin{pmatrix} |a| & |b| \\ |c| & |d| \end{pmatrix}, \qquad |L| \cdot |R| = \begin{pmatrix} |a| & |b| \\ |c| & \gamma|d| \end{pmatrix},$$

$$\text{wobei} \qquad \gamma = \left|1 - \frac{bc}{ad}\right| + \left|\frac{bc}{ad}\right|.$$

Für $|bc| \leq |ad|$ ist $\gamma \leq 3$, während es für $|ad| < |bc|$ beliebig groß werden kann. Das richtige Kriterium für die Pivotwahl ist also, die beiden Gleichungen zu vertauschen, wenn $|bc| > |ad|$ ist. Dieses Kriterium ist invariant unter Skalierungen, d.h. gleich für A und $D_1 A D_2$ mit beliebigen Diagonal-Matrizen D_1, D_2.

Im Gegensatz dazu ist die Suche nach dem betragsgrößten Element als Pivot sehr abhängig von der Skalierung und man kann sich leicht überlegen, daß durch geschickte Skalierung $A \mapsto DA$ bei der Spalten-Pivotsuche bzw. $A \mapsto D_1 A D_2$ bei der vollständigen Pivotsuche jede beliebige Pivotfolge erzwungen werden kann. Daran sieht man, daß die Suche nach dem betragsgrößten Element eine vernünftige Skalierung der Zeilen und Spalten der Koeffizienten-Matrix zur Voraussetzung hat. Bis heute ist noch keine Methode bekannt, die eine solche vernünftige Skalierung automatisch und preiswert durchführt. Die ideale Skalierung in Verbindung mit der Suche nach dem maximalen Pivot würde dafür sorgen, daß $|L| \cdot |R| = O(|LR|)$ ist und die Gauß-Elimination numerisch stabil ist.

Solange man nicht automatisch skalieren kann, ist der Anwender zuständig. Insbesondere, wenn die Gleichungen keine einheitliche physikalische Dimension haben, ist dieser Problematik Aufmerksamkeit zu schenken.

3.6 Ergänzungen

Iterative Nachverbesserung

Sei $\tilde{x}$ irgendeine Näherung für die exakte Lösung $x = A^{-1}b$. Dieses $\tilde{x}$ kann in drei Schritten verbessert werden:

`Schritt` (1): $r := b - A\tilde{x}$, `(doppelt genau)`
`Schritt` (2): `Löse` $A\Delta x = r$ `nach` Δx,
`Schritt` (3): $x := \tilde{x} + \Delta x$.

Bei exakter Rechnung ist $\Delta x = A^{-1}r = A^{-1}(b - A\tilde{x}) = x - \tilde{x}$, also tatsächlich die richtige Korrektur von $\tilde{x}$. Rundungsfehler geben ein verfälschtes $\Delta\tilde{x}$ und damit nicht das exakte x. Die Verbesserung kann wiederholt werden und solange Δx nur mindestens eine korrekte Stelle hat, ist $\tilde{x} + \Delta x$ besser als $\tilde{x}$.

Block-Elimination

Matrizen lassen sich partitionieren (unterteilen) in Blöcke, indem man Zeilen und Spalten in Gruppen zusammenfaßt. Dann entsteht eine Blockmatrix, deren Elemente selbst wieder Matrizen sind. Dreieckszerlegung und Gauß-Elimination als rein algebraische Operationen lassen sich sofort übertragen, zum Beispiel

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} = \begin{pmatrix} I & 0 \\ L_{21} & I \end{pmatrix} \begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix}$$

mit $R_{11} = A_{11}$, $R_{12} = A_{12}$, $L_{21} = A_{21}A_{11}^{-1}$, $R_{22} = A_{22} - A_{21}A_{11}^{-1}A_{12}$. Man vergleiche R_{22} mit (3.6). Voraussetzung ist nur, daß Zeilen und Spalten auf die gleiche Weise gruppiert wurden, so daß die Diagonal-Blöcke quadratisch sind. Auch die Pivots sind jetzt Matrizen und sie müssen nicht-singulär sein. Eine Pivotsuche kommt allerdings aus praktischen Gründen nicht in Frage.

Rang-Bestimmung

Macht man Gauß-Elimination mit vollständiger Pivotsuche an $A \in \mathbb{C}^{m,n}$ und bricht das Verfahren nach r Schritten ab, weil kein Pivot $\neq 0$ mehr vorhanden ist, so gilt

$$P_1 A P_2 = LR$$

mit P_1 und P_2 Permutations-Matrizen und $L \in \mathbb{C}^{m,r}$, $R \in \mathbb{C}^{r,n}$, beide vom Rang r wegen der gestaffelten Spalten bzw. Zeilen. Also ist $\mathrm{Rang}(A) = \mathrm{Rang}(P_1 A P_2) = \mathrm{Rang}(LR) \leq r$. Andererseits sind die Zeilen von R Linearkombinationen von irgendwelchen r Zeilen von $P_1 A P_2$ und analog die Spalten von L Linearkombinationen irgendwelcher r Spalten von $P_1 A P_2$. Dieses hat somit mindestens r linear unabhängige Zeilen und Spalten, so daß $\mathrm{Rang}(A) \geq r$. Zusammen also $\mathrm{Rang}(A) = r =$ Anzahl der Eliminations-Schritte mit vollständiger Pivotwahl.

Für die Praxis ist das unbrauchbar, weil der Rang eine unstetige Funktion der Matrix-Elemente ist und kleinste Ungenauigkeiten/Rundungsfehler zum trivialen Rang $r = \min(m, n)$ führen. Äußerlich ist das daran zu erkennen, daß die Restmatrix nicht Null wird, oft und verblüffenderweise noch nicht einmal irgendwie „klein“.

In Wahrheit ist die Frage nach dem Rang einer vorgelegten Matrix töricht, außer wenn die Matrix aus einem Schullehrbuch zur Algebra entsprungen ist (also kleines m und n, ganzzahliges a_{ij}). Die richtige Frage — und mit numerischen Methoden auch gut zu entscheiden — lautet: Wie groß ist der Abstand d_r einer vorgelegten Matrix $A \in \mathbb{C}^{m,n}$ zur Menge der Rang-r-Matrizen in $\mathbb{C}^{m,n}$, $r = 1, \ldots, \min(m, n)$? Anders ausgedrückt, wie klein kann δA gewählt werden, so daß $\mathrm{Rang}(A + \delta A) \leq r$? Diese d_r sind

a) viel aufschlußreicher für den Anwender als $\mathrm{Rang}(A)$,

Gauß-Jordan-Elimination

Die Gauß-Elimination annulliert im j-ten Hauptschritt alle a_{ij} für $i > j$. Man kann auch zusätzlich die Elemente a_{ij} für $i < j$ annullieren. Das nennt man Gauß-Jordan-Elimination. Sie ist teurer ($n^3/2$ statt $n^3/3$ Subtraktionen und Multiplikationen) und weniger genau, weil die Eliminations-Faktoren l_{ij} durch die Pivotwahl nur für $i > j$ betragsmäßig ≤ 1 gemacht werden können, nicht dagegen für $i < j$. Die Gauß-Jordan-Elimination erlaubt mehr Parallel-Arbeit und führt auf ein besonders kompaktes Programm zur Matrizen-Invertierung.

Zur Herleitung betrachtet man die Matrix A als Abbildung $x \mapsto y$:

$$\begin{aligned} a_{11}x_1 + \sum_{j=2}^{n} a_{1j}x_j &= y_1, \\ a_{i1}x_1 + \sum_{j=2}^{n} a_{ij}x_j &= y_i, \quad i = 2, \ldots, n. \end{aligned}$$

Jetzt bringe man x_1 nach rechts zu $y_2, \ldots, y_n$ und dafür y_1 nach links zu $x_2, \ldots, x_n$. Das bedeutet eine neue Koeffizienten-Matrix A' mit

$$\begin{aligned} a'_{11} &:= 1/a_{11} \\ a'_{1j} &:= -a_{1j}/a_{11}, \quad j = 2, \ldots, n, \\ a'_{i1} &:= a_{i1}/a_{11}, \quad i = 2, \ldots, n, \\ a'_{ij} &:= a_{ij} - a_{i1}a_{1j}/a_{11}, \quad i = 2, \ldots, n \text{ und } j = 2, \ldots, n. \end{aligned}$$

$n-1$ weitere solche **Austausch-Schritte** bringen $y_2, \ldots, y_n$ nach links und dafür $x_2, \ldots, x_n$ nach rechts, so daß aus $Ax = y$ zum Schluß $A^{-1}y = x$ geworden ist.

3.7 Überbestimmte Gleichungssysteme, lineare Ausgleichsrechnung

Ein lineares Gleichungssytem kann mehr Gleichungen als Unbekannte umfassen, also **überbestimmt** sein:

$$Ax = b \qquad \text{mit} \quad A \in \mathbb{R}^{m,n},\ x \in \mathbb{R}^n,\ b \in \mathbb{R}^m \text{ und } m > n. \tag{3.11}$$

Ungenauigkeiten in den Eingabe-Daten A und b geben in der Regel mehr oder weniger stark ausgeprägte Widersprüche, so daß dann kein x existiert, das (3.11) exakt erfüllt: Das Residuum

$$r(x) := b - Ax$$

ist für kein x zum Verschwinden zu bringen. Es ist naheliegend, dann als Ersatz für die exakte Lösung nach einem x zu suchen, das das Residuum möglichst klein macht, z.B. im Sinne der Euklidschen Norm:

$$\text{Finde } \hat{x} \text{ so, daß } \|r(\hat{x})\|_2 \leq \|r(x)\|_2 \quad \forall x \in \mathbb{R}^n. \tag{3.12}$$

Man kann die Minimierungs-Aufgabe (3.12) auch für andere Normen anstelle von $\|\cdot\|_2$ betrachten, z.B.:

$$\begin{aligned} \|r\| &= \|r\|_1, && (3.14)\\ \|r\| &= \|r\|_\infty, && (3.15)\\ \|r\|^2 &= g_1 r_1^2 + \ldots + g_m r_m^2, \qquad \text{alle } g_i > 0, && (3.16)\\ \|r\|^2 &= r^T M r, \qquad M \text{ positiv definit.} && (3.17) \end{aligned}$$

(3.14) und (3.15) führen auf nicht-quadratische Minimierungsaufgaben, die sich mit Techniken des sogenannten linearen Programmierens lösen lassen. Das ist deutlich teurer. Außerdem ergeben sich schon bei den allereinfachsten Fällen Lösungen, die für den typischen Anwender völlig inakzeptabel sind. Die Lösung ist dann nämlich durch Zufälle oder durch die maximalen Inkonsistenzen (im Laborjargon die *Ausreißer* genannt) bestimmt.

(3.16) und (3.17) sind für den Anwender sehr wichtig und haben eine statistische Theorie im Hintergrund. Sie können leicht auf (3.12) zurückgeführt werden: Für (3.16) skaliert man die Zeilen von (3.11) mit $\sqrt{g_1}, \ldots, \sqrt{g_m}$, für (3.17) benützt man die Cholesky-Zerlegung $M =: R^T R$ und transformiert $A \mapsto RA$, $b \mapsto Rb$, $r(x) \mapsto Rr(x)$, so daß $\|Rr\|_2^2 = r^T M r$.

Nachdem die Bedingungen für die Existenz und Eindeutigkeit von $\hat{x}$ und $\hat{r}$ geklärt sind, ist als nächstes unbedingt die Kondition des Ausgleichsproblems zu untersuchen, also die Änderung δx und δr abzuschätzen für gegebene, infinitesimale Änderungen δA und δb. In der Vorlesung *Numerische Mathematik III* wird gezeigt:

$$\begin{aligned} \|\delta x\| &\leq \kappa \cdot (\alpha \|\hat{x}\|_2 + \beta) + \kappa^2 \alpha \|\hat{r}\|_2 / \|A\|_2 + O(\alpha^2),\\ \|\delta r\|_2 &\leq \|\delta A\| \cdot \|\hat{x}\|_2 + \|\delta b\| + \kappa\alpha \cdot \|\hat{r}\|_2 + O(\alpha^2), \end{aligned}$$

wobei

$$\begin{aligned} \alpha &:= \|\delta A\|_2 / \|A\|_2,\\ \beta &:= \|\delta b\|_2 / \|A\|_2,\\ \kappa &:= \max_{\|x\|_2=1} \|Ax\|_2 / \min_{\|x\|_2=1} \|Ax\|_2, \quad \text{Konditionszahl von } A. \end{aligned}$$

Entscheidend ist der Term mit $\kappa^2 \|\hat{r}\|_2$ in der Abschätzung für $\|\delta x\|_2$. Er besagt, daß das Ausgleichsproblem viel empfindlicher gegenüber Änderungen von A ist als das lineare Gleichungssystem. Dabei kommt es auf das Residuum an, also auf die Konsistenz der auszugleichenden linearen Gleichungen. Sind diese in guter Übereinstimmung, dann ist $\|\hat{r}\|_2$ klein. Widersprechen sie sich dagegen, dann ist $\|\hat{r}\|_2$ groß.

Der billigste Weg zur Beschaffung von $\hat{x}$ ist über die Normalgleichung (3.13). Dabei läßt sich das Formen von $A^T A$ mit dessen Cholesky-Zerlegung kombinieren, ebenso das Formen von $A^T b$ mit der Vorwärts-Substitution. Dieser Weg galt früher als

sparse), wenn die meisten Matrixelemente Null sind. Beispiele für das Auftreten linearer Gleichungen mit solchen Koeffizienten-Matrizen:

a) Diskretisierung linearer Randwert-Aufgaben bei gewöhnlichen und partiellen Differentialgleichungen (Ersetzen von Differentialquotienten durch Differenzen-Quotienten über einem Gitter).

b) Gleichgewichts-Bedingungen in mechanischen oder elektrischen Netzwerken.

c) Matrix-Gleichungen in der Steuerungstheorie, z.B. zu $A \in \mathbb{R}^{m,m}$, $B \in \mathbb{R}^{n,n}$, $C \in \mathbb{R}^{m,n}$ finde $X \in \mathbb{R}^{m,n}$, so daß $AX + XB = C$.

d) Algebraische Methoden der Computer-Tomographie.

Die Gauß-Elimination kann dann nur noch in den besonders günstigen Fällen benutzt werden, wo es gelingt, durch geschickte Wahl der Pivotfolge das Auffüllen der Matrix mit Elementen ungleich Null gering zu halten.

In den übrigen Fällen kann man eine Iteration versuchen, bei der eine Näherung x für die exakte Lösung $\bar{x} = A^{-1}b$ des Gleichungssystems $Ax = b$ in vielen billigen Schritten verändert und allmählich verbessert wird. Dazu wird in jedem Schritt ein Indexpaar (i, k) gewählt mit $a_{ik} \neq 0$ und dann die i-te Gleichung erfüllt durch Wahl von x_k:

Komponenten-Relaxation

`Wähle` i `und` k `so, daß` $a_{ik} \neq 0$.
`Wähle` x_k `so, daß` $\sum_j a_{ij}x_j = b_i$.
`Also` $x_k := \left(b_i - \sum_{j \neq k} a_{ij}x_j\right)/a_{ik}$
`oder` $r_i := b_i - \sum_j a_{ij}x_j$, $x_k := x_k + r_i/a_{ik}$.

Für die Durchführung gut geeignet ist eine Speicher-Organisation, bei der man für jede Zeile eine Liste der Elemente ungleich 0 hat:

$$j_1, a_{ij_1}, \qquad j_2, a_{ij_2}, \qquad \ldots$$

Noch einfacher geht es bei der Diskretisierung von partiellen Differentialgleichungen, zum Beispiel der Laplace-Gleichung $u_{xx} + u_{yy} = 0$:

$$\begin{aligned} 0 &= u_{i+1,j} - 2u_{ij} + u_{i-1,j} + u_{i,j+1} - 2u_{ij} + u_{i,j-1} \qquad \Longrightarrow \\ u_{ij} &= (u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1})/4 \end{aligned}$$

Jacobi- oder Gesamtschritt-Verfahren (J)

für $m := 1, 2, \ldots$
    für alle i: $x_i' := x_i + \Big(b_i - \sum_j a_{ij} x_j\Big)/a_{ii}$;
    für alle i: $x_i := x_i'$.

Hier können n Prozessoren simultan jeweils eine neue Komponente berechnen und am Ende des Gesamtschrittes (Zyklus) ihr Resultat abliefern. Mischformen von Einzel- und Gesamtschritt-Verfahren sind möglich, wenn eine begrenzte Zahl von Prozessoren an einem größeren Gleichungssystem arbeiten soll. Man spricht dann auch von **Block-Relaxation**.

Neben der algorithmischen Formulierung gibt es eine algebraische Formulierung dieser Prozesse. Dazu bezeichne

D	den Diagonalteil von A,
E	den strikt unteren Dreiecksteil von A,
F	den strikt oberen Dreiecksteil von A,
	also (additive) Zerlegung $A = E + D + F$,
$x^{(m)}$	den Näherungsvektor am Ende des m-ten Zyklus ($m = 0, 1, 2, \ldots$).

Satz 3.2

Für alle m ist

$$A_1 x^{(m+1)} + A_2 x^{(m)} = b, \tag{3.18}$$

wobei

$$\begin{array}{lllllll} \text{für (J):} & A_1 & = & D, & A_2 & = & E + F, \\ \text{für (GS):} & A_1 & = & D + E, & A_2 & = & F, \\ \text{für (ÜR):} & A_1 & = & \omega^{-1} D + E, & A_2 & = & (1 - \omega^{-1}) D + F, \end{array} \tag{3.19}$$

und immer ist $A_1 + A_2 = A$.

Beweis: Bei (J) war $x' = x + D^{-1}(b - Ax)$; Multiplikation mit D von links und Umbenennung $x \mapsto x^{(m)}$, $x' \mapsto x^{(m+1)}$ gibt $Dx^{(m+1)} = Dx^{(m)} + b - (E + D + F)x^{(m)}$.

Bei (GS) liegt der Spezialfall $\omega = 1$ der (ÜR) vor.

Bei (ÜR) sind in der Berechnungsformel für das neue x_i rechts für $j = 1$ bis $i - 1$ schon die neuen Komponenten und für $j = i$ bis n noch die alten Komponenten einzusetzen, also

$$\begin{aligned} x_i^{(m+1)} &= x_i^{(m)} + \omega\Big(b_i - \sum_{j<i} a_{ij} x_j^{(m+1)} - a_{ii} x_i^{(m)} - \sum_{j>i} a_{ij} x_j^{(m)}\Big)/a_{ii}, \\ x^{(m+1)} &= x^{(m)} + \omega D^{-1}\left(b - E x^{(m+1)} - D x^{(m)} - F x^{(m)}\right). \end{aligned}$$

Spezial-Literatur für die Relaxations-Verfahren

[1] Varga, R.S., *Matrix Iterative Analysis*, Prentice Hall, 1962

[2] Young, D.M., *Iterative Solution of Large Linear Systems*, Academic Press, 1971

Als Einführung in die neuen Methoden:

[1] Briggs, W.L., *A Multigrid Tutorial*, SIAM, 1987

4.1 Notation und Aufgabenstellung

$f\colon [a,b] \to \mathbb{R}$ sei eine für das Folgende genügend glatte Funktion, d.h. alle auftretenden Ableitungen sollen existieren.

Definition 4.1

> $p \in \mathbb{P}_{n-1}$ heißt der **Polynom-Interpolant** von f zu den gegebenen **Stützstellen** $x_1, \ldots, x_n \in [a,b]$, wenn das **Restglied** $p - f$ die $x_1, \ldots, x_n$ als Nullstellen besitzt. Also $p - f$ ein Vielfaches von
>
> $$\omega(x) := (x - x_1) \cdots (x - x_n). \tag{4.1}$$

Die Bestimmung eines solchen p ist die **Interpolations-Aufgabe**. Die Reihenfolge der Stützstellen spielt in dieser Definition keine Rolle, so daß man sie o.B.d.A. aufsteigend geordnet annehmen kann:

$$a \le x_1 \le \ldots \le x_n \le b.$$

Die Stützstellen heißen **einfach**, wenn $x_i < x_j$ für $i < j$. Ist dagegen

$$x_{i-1} < x_i = \ldots = x_{i+m-1} < x_{i+m}, \tag{4.2}$$

dann bilden $x_i, \ldots, x_{i+m-1}$ zusammen eine m**-fache** Stützstelle und obiges $\omega(x)$ enthält den Faktor $(x - x_i)^m$. Die Definition 4.1 ist so zu verstehen, daß in diesem Fall das Restglied bei x_i eine m-fache Nullstelle haben muß, d.h.

$$D^k(p - f)(x_i) = 0 \qquad \text{für} \quad k = 0, \ldots, m-1.$$

In die Spezifikation des Polynom-Interpolanten gehen also nur die diskreten Werte $y_1, \ldots, y_n$ der Funktion f und gegebenenfalls die ihrer Ableitungen ein, präzis:

$$y_{i+k} := D^k f(x_i), \quad k = 0, \ldots, m-1$$

ist die Bedeutung der **Stützwerte** zu einer m-fachen Stützstelle (4.2).

Nach Definition 4.1 erfüllt der Interpolant $p \in \mathbb{P}_{n-1}$ somit die insgesamt n **Interpolations-Bedingungen** (IB)

$$D^k p(x_i) = y_{i+k}, \quad k = 0, \ldots, m-1, \text{ wo } x_{i-1} < x_i = \ldots = x_{i+m-1} < x_{i+m}.$$

Die Frage nach Existenz und Eindeutigkeit des Interpolanten beantwortet sich fast von selbst:

Der Ansatz $p(x) = c_0 + c_1 x + \ldots + c_{n-1} x^{n-1}$ in die n IB eingesetzt, führt auf ein lineares Gleichungssystem der Ordnung n für die Koeffizienten $c_0, \ldots, c_{n-1}$ mit den Stützwerten als rechten Seiten. Im homogenen Fall ($f(x) = 0$ bzw. alle $y_i = 0$) muß $p \in \mathbb{P}_{n-1}$ die n Nullstellen $x_1, \ldots, x_n$ haben und somit identisch verschwinden: $y_1 = \ldots = y_n = 0$, also $c_0 = \ldots = c_{n-1} = 0$. Das ist bekanntlich die notwendige und hinreichende Bedingung für die Existenz und Eindeutigkeit der Lösung im inhomogenen Fall:

4.2 Der rekursive Aufbau des Polynom-Interpolanten (Schema von Aitken und Neville)

Beim Hinweis auf Existenz und Eindeutigkeit des Interpolanten ist eine Möglichkeit zu seiner Beschaffung aufgetreten: Ein Ansatz mit n Koeffizienten und Auflösen des linearen Gleichungssystems der IB zum Beispiel mittels Gauß-Elimination. Bequemer und auch ökonomischer ist dagegen der systematische Aufbau der Gesamtlösung in Form eines Baums der Teillösungen

$$p_{i,k} \in \mathbb{P}_k \qquad \text{zu Stützstellen } x_i, \ldots, x_{i+k},\ 1 \le i \le i+k \le n.$$

Dabei handelt es sich um eine Rekursion für *Funktionen*, also Identitäten der unabhängigen Variablen x. **Alle solche Identitäten dürfen beliebig oft differenziert werden und geben dann Rekursionsformeln für die Ableitungen jeder Ordnung.**

Als Demonstration für diese grundsätzliche Technik sei das **Hornerschema** (engl. *nested multiplications*) erwähnt.

Zu $H_0(x) := c_0 + c_1 x + \ldots + c_n x^n$ gehört als (Funktions-) Rekursion das **einzeilige Hornerschema**

`Start:` $H_n(x) := c_n$;
`für` $k := n-1$ `abwärts bis 0:`
$\quad H_k(x) := H_{k+1}(x) \cdot x + c_k$.

Einmal differenzieren liefert das **zweizeilige Hornerschema**

`Start:` $H'_n(x) := 0$; $H_n(x) := c_n$;
`für` $k := n-1$ `abwärts bis 0:`
$\quad H'_k(x) := H'_{k+1}(x) \cdot x + H_{k+1}(x)$;
$\quad H_k(x) := H_{k+1}(x) \cdot x + c_k$.

Usw. für das Schema mit mehr Zeilen. Wenn man wie angegeben H'_k *vor* H_k berechnet, kann man auf den Index k verzichten (alle $H'_k(x)$ und alle $H_k(x)$ *in situ*[1] speichern).

Zu einer einfachen Stützstelle x_i gehört $p_{i,0}(x) = y_i$ (Polynom vom Grad 0 bzw. der Ordnung 1).

Zu einer m-fachen Stützstelle $x_i = \ldots = x_{i+m-1}$ gehört nach (4.5)

$$p_{i,k}(x) = y_i + \ldots + y_{i+k} \cdot (x - x_i)^k / k!, \quad k = 0, \ldots, m-1.$$

[1] *In situ* (deutsch „auf der Stelle“) bedeutet, daß in einem Computer-Programm die neue Größe an die Stelle der alten tritt, so daß die logisch zu unterscheidenden Größen den Speicherplatz miteinander teilen.

Lemma 4.5

Für $k \in \{1, \ldots, n-1\}$ gilt

$$p_{1,k}(x) - p_{1,k-1}(x) = a_{1,k} \cdot (x - x_1) \cdots (x - x_k) \quad (4.8)$$
$$p_{n-k,k}(x) - p_{n-k+1,k-1}(x) = a_{n-k,k} \cdot (x - x_{n-k+1}) \cdots (x - x_n) \quad (4.9)$$

Dies verknüpft also die Polynome in der vorderen bzw. hinteren Kante des dreieckigen Aitken-Neville-Schemas. (Selbstverständlich gilt das auch für jedes Teilschema mit Indizes i bis $i+k$ statt 1 bis n.)

Beweis: Beweis von (4.8):

$$p_{1,k} - p_{1,k-1} = (p_{1,k} - f) - (p_{1,k-1} - f).$$

Links steht ein Polynom vom Grad k, rechts stehen zwei Restglieder, die null sind bei x_1 bis x_{k+1} bzw. bis x_k. Somit für ein bestimmtes C

$$p_{1,k}(x) - p_{1,k-1}(x) = C \cdot (x - x_1) \cdots (x - x_k).$$

Der Vergleich der Koeffizienten von x^k gibt $a_{1,k} = C$.

Beweis von (4.9) analog. ■

Addiert man alle in (4.8) bzw. (4.9) angegebenen Differenzen, so kommt man zu den zwei wichtigsten Formeln für den Interpolanten:

$$p(x) = a_{1,0} + a_{1,1} \cdot (x - x_1) + \ldots + a_{1,n-1} \cdot (x - x_1) \cdots (x - x_{n-1}), \quad (4.10)$$
$$p(x) = a_{n,0} + a_{n-1,1}(x - x_n) + \ldots + a_{1,n-1} \cdot (x - x_2) \cdots (x - x_n). \quad (4.11)$$

Als Übungsaufgabe sollte der Interpolant $p_{i,k}(x)$ zu den Stützstellen $x_i, \ldots, x_{i+k}$ notiert werden.

Man sieht also, daß die $a_{i,k}$ eine Schlüsselrolle spielen bei der Konstruktion des Interpolanten p. Diese jeweils höchsten Koeffizienten sind so wichtig, daß sie noch viele andere Bezeichnungen haben. Die heute gebräuchlichste ist

$$a_{i,k} \equiv [x_i, \ldots, x_{i+k}]f,$$

die zum Ausdruck bringt, daß die $a_{i,k}$ so wie das ganze $p_{i,k}$ eine lineare Funktion der Stützwerte zu den Stützstellen $x_i, \ldots, x_{i+k}$ sind (Operator-Schreibweise). Also in der neuen Notation:

Definition 4.6 *Dividierte Differenzen*

Der höchste Koeffizient des Interpolanten $p_{i,k}$ zu den Stützstellen $x_i, \ldots, x_{i+k}$, heißt **dividierte Differenz der Ordnung k, gestützt auf**

(L) $[x_1, \ldots, x_n]f$ ist eine Linearkombination der Stützwerte
$f(x_i)$, $f'(x_i)$ wo $x_i = x_{i+1}$, $f''(x_i)$ wo $x_i = x_{i+1} = x_{i+2}$ usw.

(S) $[x_1, \ldots, x_n]f$ ist symmetrisch in allen Stützstellen, die Reihenfolge der Stützstellen in der Liste $[\cdots]$ spielt keine Rolle
(zum Beispiel $[x_i, x_{i+1}]f = [x_{i+1}, x_i]f$, vgl. oben).

(P) $f \in \mathbb{P}_{n-2} \Rightarrow [x_1, \ldots, x_n]f = 0$ und $f \in \mathbb{P}_{n-1} \Rightarrow [x_1, \ldots, x_n]f = D^{n-1}f/(n-1)!$
(weil dann $p = f$).

Übungsaufgabe 4.1: (Die Lösung wird später benötigt bei den B-Splines.)

Bestimme das Gewicht g von $D^m f(x_i)$ an einer $(m+1)$-fachen Stützstelle, $\ldots < x_i = \ldots = x_{i+m} < \ldots$:

$$[x_1 \ldots, x_n]f = g \cdot D^m f(x_i) + \text{Beiträge aller übrigen Stützwerte.}$$

Zeige insbesondere für lauter einfache Stützstellen

$$[x_1, \ldots, x_n]f = \sum_{i=1}^{n} f(x_i)/\omega'(x_i), \qquad \omega(x) = (x - x_1) \cdots (x - x_n).$$

Tip: Die Wahl $f(x) = (x - x_i)^m \prod_{x_j \neq x_i} (x - x_j) \in \mathbb{P}_{n-1}$ ist günstig und ausreichend.

Übungsaufgabe 4.2: *Produkt-Regel:*

Sei $x_1 \leq \ldots \leq x_n$ und

$$\begin{aligned}
P(x) &:= \sum_{i=1}^{n} p_i \cdot (x - x_1) \cdots (x - x_{i-1}), \qquad \text{alle } p_i \in \mathbb{R}, \\
Q(x) &:= \sum_{j=1}^{n} q_j \cdot (x - x_{j+1}) \cdots (x - x_n), \qquad \text{alle } q_j \in \mathbb{R}, \\
R(x) &:= \sum_{i=1}^{n} \sum_{j=i}^{n} p_i q_j \cdot (x - x_1) \cdots (x - x_{i-1}) \cdot (x - x_{j+1}) \cdots (x - x_n),
\end{aligned}$$

mit der Konvention: leeres Produkt $= 1$. Man zeige:

(a) Der i-j-Term in $R(x)$ hat $(i-1) + (n-j) \leq n-1$ Faktoren $x - x_k$,
$R \in \mathbb{P}_{n-1}$.

(b) Der Koeffizient von x^{n-1} in $R(x)$ ist $\sum_{k=1}^{n} p_k q_k$.

(c) $P(x)Q(x) - R(x)$ ist ein Vielfaches von $\omega(x) := (x - x_1) \cdots (x - x_n)$.

(d) $p_i = [x_1, \ldots, x_i]f$, falls P der Polynom-Interpolant von einem f,
$q_j = [x_j, \ldots, x_n]g$, falls Q der Polynom-Interpolant von einem g.

Zu (e) und (f):

$$\begin{aligned} & \Big[D^{n-1} f(x_0 + \sum_{i=1}^{n-1} \sigma_i (x_i - x_0) + \sigma_n (x_n - x_0))\Big]_{\sigma_n = 0}^{\sigma_n = 1 - \sigma_1 - \cdots - \sigma_{n-1}} \\ = \quad & D^{n-1} f(x_n + \sum_{i=1}^{n-1} \sigma_i (x_i - x_n)) - D^{n-1} f(x_0 + \sum_{i=1}^{n-1} \sigma_i (x_i - x_0)) \end{aligned}$$

Zu (g): Induktion bezüglich n.

Zu (h): Statt σ_i neue Integrations-Variablen $\tau_i := (1 - \sigma_1 - \cdots - \sigma_{i-1}) - \sigma_i = \tau_{i-1} - \sigma_i$ in $\langle x_n, x_0, \ldots, x_{n-1} \rangle f \quad (\tau_0 := 1)$.

4.4 Erweiterter Mittelwertsatz und Restgliedformel

Zunächst sei $f \in C^{n-1}[x_1, x_n]$, Stützstellen $x_1 \leq \ldots \leq x_n$. Also hat auch das Restglied $f - p$ stetige Ableitungen bis einschließlich der Ordnung $n - 1$, auf die der Satz von Rolle angewendet werden kann:

$f - p$	hat mindestens n Nullstellen (einfache oder mehrfache),
$D(f - p)$	hat mindestens $n - 1$ Nullstellen,
$\cdots$	
$D^{n-1}(f - p)$	hat mindestens eine Nullstelle, genannt ξ.

$D^{n-1} f(\xi) = D^{n-1} p$ und der höchste Koeffizient von p ist $D^{n-1} p/(n-1)! = D^{n-1} f(\xi)/(n-1)!$. Kombiniert mit der Definition der dividierten Differenzen gilt also

$$[x_1, \ldots, x_n] f = D^{n-1} f(\xi)/(n-1)! \; .$$

Selbstverständlich gilt das auch für jede Teilmenge der Stützstellen. Das beweist

Satz 4.9 *Erweiterter Mittelwertsatz*

$$f \in C^k[x_1, x_n] \Rightarrow \exists \xi \in [x_i, x_{i+k}] \ni$$

$$[x_i, \ldots, x_{i+k}] f = D^k f(\xi)/k!, \qquad 1 \leq i \leq i + k \leq n,$$

($k = 1$ ist der Standardfall, ξ hängt von i, k ab).

Jetzt sei f sogar n-mal stetig differenzierbar und $\bar{x}$ eine beliebige zusätzliche Stützstelle. $P \in \mathbb{P}_n$ interpoliere $x_1, \ldots, x_n$ und $\bar{x}$. Nach der ersten Newtonschen Interpolationsformel in Satz 4.8 ist

$$P(x) = p(x) + [x_1, \ldots, x_n, \bar{x}] f \cdot (x - x_1) \cdots (x - x_n).$$

Bringt man $p(x)$ nach links und beachtet, daß $P(\bar{x}) = f(\bar{x})$, so erhält man die Identität

$$f(\bar{x}) - p(\bar{x}) = [x_1, \ldots, x_n, \bar{x}] f \cdot (\bar{x} - x_1) \cdots (\bar{x} - x_n).$$

Randzonen um diesen riesigen Faktor verstärkt. Solche Konditionszahlen machen die Polynom-Interpolation für große n praktisch unbrauchbar.

Wenn jeder Stützpunkt einen Fehler $\delta y_i \in [-\varepsilon, \varepsilon]$ besitzt, dann hat der interpolierte Zwischenwert $y = p(x)$ einen Fehler $\delta y \in [-\varepsilon\kappa(x), \varepsilon\kappa(x)]$ mit

$$\kappa(x) := \sum_{i=1}^{n} |L_i(x)| \geq \sum_{i=1}^{n} L_i(x) = 1.$$

Das ergibt die Wahl $\delta y_i = \varepsilon \cdot \mathrm{sign}(L_i(x))$. $\kappa(x)$ ist die Konditionszahl der Interpolations-Aufgabe (Lebesgue-Funktion) und der Zwischenwert y ist schlecht konditioniert, wo $\kappa(x) \gg 1$. Zwei Beispiele mit $n = 13$ sind im nebenstehenden Diagramm dargestellt:

(a) Bei äquidistanten Stützstellen sind die beiden Randzonen gefährlich, wo $\kappa(x)$ als Funktion von n proportional zu $2^n/n \log n$ anwächst. Man sollte deshalb einen Interpolanten zu *äquidistanten* Stützstellen immer nur für Approximationszwecke in der Nähe des Zentrums einsetzen und niemals für solche auf dem *ganzen* Intervall $[x_1, x_n]$.

(b) Durch eine geeignete Verdichtung der Stützstellen in den Randzonen kann man die Situation entscheidend verändern und eine nahezu gleichmäßige Konditionszahl $\kappa(x)$ erzielen.

Auf diese beiden unterschiedlichen Situationen wird zurückgekommen bei der Quadratur in Kapitel 8. Auch der proportional zu n bzw. n^2 anwachsenden Aufwand macht die Polynom-Interpolation unpraktisch für große Datenmengen.

4.6 Interpolation mit trigonometrischen Polynomen, diskrete Fourier-Transformation

Die Polynom-Interpolation ist auch für komplexe Stützstellen und -werte definiert, und dann genauso lösbar: Das rekursive Schema von Aitken-Neville und die Newtonsche Interpolationsformel mit den dividierten Differenzen als Koeffizienten bleiben gültig. Ein Spezialfall verdient gründliche Diskussion wegen seiner Wichtigkeit und den enormen Einsparungsmöglichkeiten:

Äquidistante Stützpunkte auf dem komplexen Einheitskreis

$$\begin{array}{lll} \text{Stützstellen (einfach)} & z_j := \omega^j, \qquad j = 0, \ldots, n-1, \ \omega = e^{2\pi i/n}, \\ \text{Stützwerte} & w_j \in \mathbb{C}, \qquad j = 0, \ldots, n-1, \end{array} \tag{4.12}$$

Ansatz für Interpolant $p(z) = c_0 + c_1 \cdot z + \ldots + c_{n-1} \cdot z^{n-1}$, also

$$\omega^n = 1, \quad i := \sqrt{-1}, \quad w = (w_0, \ldots, w_{n-1}) \in \mathbb{C}^n, \quad c = (c_0, \ldots, c_{n-1}) \in \mathbb{C}^n.$$

Die Transformation der unabhängigen Variablen

$$z = e^{it} \qquad \Longleftrightarrow \qquad t = -i \ln z$$

gibt eine bijektive Abbildung des Einheitskreises in $\mathbb{C}$ auf das halbseitig offene t-Intervall $[0, 2\pi[$.

$$\begin{array}{lrcl} \text{Neue Stützstellen:} & t_j & := & 2\pi j/n, \qquad j = 0, \ldots, n-1, \\ \text{Neuer Interpolant:} & T(t) & := & p(e^{it}) = \displaystyle\sum_{k=0}^{n-1} c_k e^{ikt}. \end{array} \tag{4.13}$$

Die e^{ikt}, $k = 0, 1, \ldots, n-1$ sind eine Basis für die hier betrachteten **trigonometrischen Polynome** $\mathbb{T}_{n-1}$.

Im Spezialfall (4.12) lautet also die Interpolations-Aufgabe in kompakter, algebraischer Form

$$\text{Finde } c \ni \qquad w_j = \sum_{k=0}^{n-1} c_k \omega^{jk}, \quad j = 0, \ldots, n-1. \tag{4.14}$$

Existenz und Eindeutigkeit der Lösung c ist schon bewiesen. Im vorliegenden Spezialfall kann man sogar eine explizite Formel für c angeben:

$$c_k = \frac{1}{n} \sum_{l=0}^{n-1} w_l \omega^{-kl}. \tag{4.15}$$

Für gerades $n =: 2m$**:**

$$1, e^{it}, \ldots, e^{i(m-1)t}, \frac{1}{2}(e^{imt} + e^{-imt}), e^{i(1-m)t}, \ldots, e^{-it},$$

$$\hat{T}(t) = \frac{a_0}{2} + \sum_{k=1}^{m-1}(a_k \cos kt + b_k \sin kt) + \frac{a_m}{2}\cos mt.$$

In beiden Fällen sind die *komplexen* Koeffizienten

$$\begin{aligned} a_k &:= c_k + c_{n-k}, \\ b_k &:= i(c_k - c_{n-k}), \end{aligned} \qquad k = 0, \ldots, m, \quad \text{hier } c_n := c_0, \text{ vgl. (4.18).}$$

Die Kombination mit (4.17) führt auf

$$a_k = \frac{2}{n}\sum_{j=0}^{n-1} w_j \cos 2\pi jk/n, \qquad b_k = \frac{2}{n}\sum_{j=0}^{n-1} w_j \sin 2\pi jk/n. \tag{4.19}$$

Die neuen Basisfunktionen unterscheiden sich von den entsprechenden alten Basisfunktionen höchstens um den Faktor $\exp(-int)$, der an allen Stützstellen wegen $nt_j = 2\pi j$ natürlich gleich 1 ist, so daß dort alte und neue Basisfunktionen übereinstimmen und

$$w_j = T(t_j) = \hat{T}(t_j) \qquad \forall j \in \mathbb{Z}.$$

Selbstverständlich ist jedoch im allgemeinen $T \neq \hat{T}$. Der neue Interpolant $\hat{T}$ hat zwei große Vorzüge:

1. Für **reelle Stützwerte** sind nach (4.17) die c_k **hermitesch**, d.h. $c_{n-k} = \bar{c}_k$ (konjugiert komplex). Nach (4.19) sind dann die a_k und b_k reell, also auch $\hat{T}$ durchwegs reell:

   $$w \in \mathbb{R}^n \qquad \Longrightarrow \qquad \text{alle } a_k, b_k \in \mathbb{R} \qquad \Longrightarrow \qquad \hat{T}\colon \mathbb{R} \to \mathbb{R}.$$

   (T hat diese Eigenschaft nicht und ist somit prohibitiv für reelle Stützwerte!)

2. $\hat{T}$ hat viel geringere Welligkeit als T, verläuft also viel glatter, denn die obere Hälfte der Frequenzen ist nach unten geschoben worden. Typisch für die größere Glattheit ist

   $$\|D^k T\|_2 = O(n^k) \qquad \text{und} \qquad \|D^k \hat{T}\|_2 = O(m^k) \quad \text{für} \quad k \to \infty.$$

Die diskrete Cosinus-Transformation (DCT)

Für die digitale Verarbeitung von aperiodischen Signalen (z.B. Compact Disk, digitale Fernseher) ist die Periodizität (4.18) hinderlich, wie erkennbar an der zwangsweisen Gleichheit von Anfangs- und Endwert, $w_0 = w_n$. Davon kann man sich

In der *Numerik*-Vorlesung ist zu besprechen, wie man die beiden Richtungen der Fourier-Transformation (4.20): $f \mapsto \{C_k\}$ und (4.21): $\{C_k\} \mapsto f$ näherungsweise durchführen kann, wenn die exakte Berechnung unmöglich oder zu mühsam ist. Zwei sich ergänzende Hilfsmittel stehen dafür zur Verfügung:

1. die Abminderungsfaktoren,
2. die schnelle Fourier-Transformation (FFT).

In diesem Abschnitt werden die Abminderungsfaktoren besprochen und im nächsten die FFT.

Für die numerische Berechnung der Integrale in (4.20) kommen die gebräuchlichen Quadratur-Verfahren (zum Beispiel das Richardson-Verfahren der Extrapolation von Trapezsummen oder die Gauß-Quadratur, vgl. Abschnitte 8.3, 8.4) aus zwei Gründen nicht in Frage. Erstens könnte man damit jeweils nur *einen* Fourier-Koeffizienten berechnen, während *unendlich viele* benötigt werden. Zweitens werden diese üblichen Verfahren ungenau und/oder ineffizient, wenn der Integrand stark oszilliert. Genau das ist in (4.20) mit wachsendem $|k|$ der Fall.

Sehr bewährt hat sich dagegen ein Verfahren, bei dem nur das $f(t)$ im Integranden von (4.20) angenähert wird, während der Faktor $e^{-2\pi ikt}$ und das Integral unverändert bleiben, so daß

$$\tilde{C}_k(f) := \int_0^1 \tilde{f}(t) e^{-2\pi ikt} dt, \qquad k \in \mathbb{Z} \tag{4.22}$$

die gewählte Näherung für den exakten Fourier-Koeffizienten (4.20) ist. Für $\tilde{f}$ macht man den Ansatz

$$\begin{aligned} \tilde{f}(t) &= \sum_{j=0}^{n-1} w_j \phi(t - jh), \\ & n \in \mathbb{N},\ h := 1/n,\ w_j := f(jh),\ \phi \text{ periodisch, stetig.} \end{aligned} \tag{4.23}$$

Zum Beispiel kann ϕ die periodische Dachfunktion sein:

$$\phi(t) := \begin{cases} 1 - t/h & \text{falls} \quad 0 \le t \le h, \\ 0 & \text{falls} \quad h \le t \le 1 - h, \\ 1 + (t-1)/h & \text{falls} \quad 1 - h \le t \le 1. \end{cases} \tag{4.24}$$

Dann ist $\tilde{f}$ der Polygon-Interpolant von f auf dem h-Gitter. Man kann auch die im nächsten Kapitel zu besprechende

$$\text{periodische, kubische Splinefunktion } \phi \text{ mit } \phi(hj) = \delta_{0j} \tag{4.25}$$

für ϕ wählen. Dann ist $\tilde{f}$ der periodische Spline-Interpolant von f mit n Knoten/Stützstellen im Abstand h. Viele weitere Möglichkeiten für die Wahl von ϕ

4.8 Die schnelle Fourier-Transformation für $n = 2^p$

Dieser Abschnitt gilt der Durchführung der DFT im $\mathbb{C}^n$,

$$c_k = \frac{1}{n}\sum_{j=0}^{n-1} w_j e^{-2\pi ijk/n}, \qquad k = 0, \ldots, n-1, \tag{4.28}$$

$$w_j = \sum_{k=0}^{n-1} c_k e^{+2\pi ijk/n}, \qquad j = 0, \ldots, n-1. \tag{4.29}$$

Insbesondere wird (4.28) benötigt für die trigonometrische Interpolation mit $T(t)$ und $\hat{T}(t)$ nach Abschnitt 4.6 und in Verbindung mit den Abminderungsfaktoren τ_k für die numerische Berechnung der Fourier-Koeffizienten einer Funktion f. Wegen der Symmetrie von (4.28), (4.29) genügt es, im folgenden nur (4.29) zu diskutieren.

Die naive Durchführung von (4.29) erfordert $O(n^2)$ komplexe Additionen, Multiplikationen und trigonometrische Koeffizienten $\exp(2\pi ijk/n)$. In den Anwendungen kann n so groß sein, daß dieser Aufwand nicht mehr tragbar ist. Cooley und Tukey haben 1965 eine Möglichkeit angegeben, um den Aufwand entscheidend zu reduzieren, wenn n keine Primzahl sondern aus Faktoren zusammengesetzt ist: für $n = n_1 \cdots n_p$ reduziert sich der Arbeitsaufwand von $O(n^2)$ auf $O(n \cdot (n_1 + \ldots + n_p))$. Insbesondere für $n = 2^p$ braucht man nur je $np/2$ komplexe Additionen, Subtraktionen, Multiplikationen sowie nur n trigonometrische Koeffizienten. Übrigens wird dabei auch der Einfluß der Rundungsfehler deutlich verringert. Im folgenden wird nur dieser Fall $n = 2^p$ besprochen, an dem sich alles Wesentliche erläutern läßt.

Die simple mathematische Grundlage

Zunächst sei n gerade, $n =: 2m$. Man zerlegt (4.29) auf zweifache Weise, nämlich bezüglich k in **gerade** und **ungerade** Terme ($k \to 2k$ und $2k+1$) und bezüglich j in **untere Hälfte** $0, \ldots, m-1$ und **obere Hälfte** $m, \ldots, 2m-1$. (Man könnte es übrigens auch umgekehrt machen.) Das gibt

$$\begin{aligned} w_j &= \sum_{k=0}^{m-1} c_{2k} e^{2\pi ijk/m} + e^{i\pi j/m} \sum_{k=0}^{m-1} c_{2k+1} e^{2\pi ijk/m}, \\ w_{j+m} &= \sum_{k=0}^{m-1} c_{2k} e^{2\pi ijk/m} - e^{i\pi j/m} \sum_{k=0}^{m-1} c_{2k+1} e^{2\pi ijk/m}, \\ & \qquad j = 0, \ldots, m-1. \end{aligned} \tag{4.30}$$

Man beachte, daß die zweite Zeile von (4.30) aus der ersten Zeile folgt, indem man überall j durch $j+m$ ersetzt.

Diese Aufspaltung reduziert den Arbeitsaufwand um 50%, denn die beiden Teilsummen in w_j können wiederverwendet werden in w_{j+m}. Was oben als naive

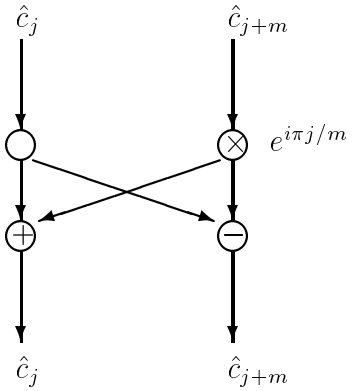
Jede zweite Komponente, nämlich $0, 2, 4, \ldots$, wird also herausgenommen und nach links geschoben, alle zurückgebliebenen, nämlich $1, 3, 5, \ldots$, gehen nach rechts. Dadurch wird aus (4.30) die Vorschrift zur Kombination

$$K: \begin{cases} w_j & = \hat{c}_j + e^{i\pi j/m}\hat{c}_{j+m}, \\ w_{j+m} & = \hat{c}_j - e^{i\pi j/m}\hat{c}_{j+m}, \end{cases} \qquad j = 0, \ldots, m-1,$$

wobei analog zu T, doch mit m statt n

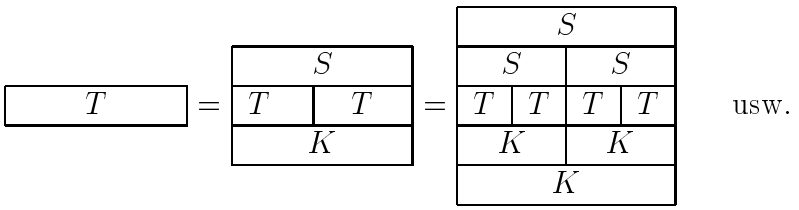
$$\begin{aligned} \hat{c}_j &= \sum_{k=0}^{m-1} c_k e^{2\pi i j k/m}, \\ \hat{c}_{j+m} &= \sum_{k=0}^{m-1} c_{k+m} e^{2\pi i j k/m}, \end{aligned} \qquad j = 0, \ldots, m-1.$$

Das Umsortieren S bringt zwei Vorteile: Zum einen haben die beiden Teilsummen $\hat{c}_j$ und $\hat{c}_{j+m}$ auch äußerlich wieder die alte Form mit konsekutiven Komponenten, zum anderen läßt sich K dann *in-situ* ausführen, wo also w_j auf $\hat{c}_j$ und w_{j+m} auf $\hat{c}_{j+m}$ abgespeichert wird. Techniker nennen K das **„butterfly diagram“**.



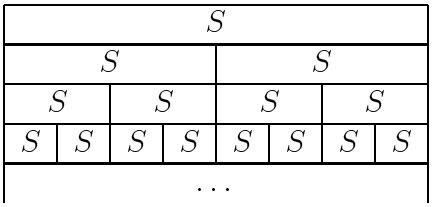
$j = 0, \ldots, m-1$
Abstand $m = n/2$

Die symbolische Darstellung des Gesamt-Prozesses ist



usw.

Insgesamt hat man einen oberen Teil mit lauter Sortierschritten abnehmender Blocklänge



```
m := n/2;
k := k + 1;
solange k' ≥ m: {k' := k' − m; m := m/2};
k' := k' + m.
```

Unzulässig wie auch überflüssig ist der Schritt von $2^p - 1$ nach 2^p. Die `solange`-Anweisung ist in 50% der Fälle abweisend, in 25% der Fälle einmal auszuführen, usw., im Mittel einmal.

Aus obiger Regel folgt $(k')' = k$, so daß das Sortieren zerfällt in lauter **paarweises Vertauschen**. Deshalb kann man das Einsortieren der Eingabe-Werte $c[0 : n-1]$ in den Arbeitsspeicher $w[0 : n-1]$, wo am Ende die Resultate stehen sollen, so organisieren, daß beide Zahlsätze nach Belieben *gleich* oder *verschieden* gewählt werden können. Insgesamt lautet dann das Programm für die Verschmelzung aller S-Zeilen:

```
w[0] := c[0]; k' := 0; m2 := n/2;
für k := 1 bis n − 1:
    m := m2
    solange k' ≥ m: {k' := k' − m; m := m/2}
    k' := k' + m
    falls k' ≥ k: {f := c[k']; w[k'] := c[k]; w[k] := f}
```

Programm für alle Kombinationsschritte

Im folgenden zählt α die Kombinationsstufen 1 bis p, $m = 2^{\alpha-1}$ ist der Partner-Abstand im „butterfly diagram“, $m2 = 2^\alpha$ ist die Blocklänge, β zählt im Block, γ zählt die Blöcke:

```
m2 := 1;
für α := 1 bis p:
    m := m2;
    m2 := m + m;
    für β := 0 bis m − 1:
        f := cos(πβ/m) + i sin(πβ/m);
        für γ := 0 in Schritten zu m2 bis n − m2:
            g := w[β + γ];
            h := w[β + γ + m] · f;
            w[β + γ] := g + h;
            w[β + γ + m] := g − h.
```

In der innersten Schleife über γ ist es ökonomischer, $\delta := \beta + \gamma$ als Laufvariable einzuführen:

also $a_k - ib_k = f_k - g_k$.

Man faßt also die reellen y_j paarweise zu komplexen Zahlen $w_j = y_{2j} + iy_{2j+1}$ zusammen und macht damit eine FFT der Länge m. Die resultierenden c_k, ergänzt durch $c_m = c_0$, werden umgerechnet in die f_k und g_k. Die gesuchten a_k und b_k sind Real- und Imaginärteil von $f_k - g_k$.

Noch ein bißchen Arbeit läßt sich einsparen, indem man von den f_k, g_k wiederum nur die untere Hälfte berechnet und ausnutzt, daß nach Definition

$$\bar{f}_{m-k} = f_k \qquad \text{und} \qquad \bar{g}_{m-k} = -g_k,$$

also

$$a_{m-k} + ib_{m-k} = f_k + g_k.$$

Beachte, daß wegen $c_m = c_0$ gilt

$$\begin{aligned} f_0 &= (c_0 + \bar{c}_0)/2 &&= \mathrm{Re}(c_0), \\ g_0 &= (c_0 - \bar{c}_0)i/2 &&= -\mathrm{Im}(c_0). \end{aligned}$$

Insgesamt hat man den Algorithmus:

`Schritt 1:`	w_j	$:= y_{2j} + iy_{2j+1}, \quad j = 0, \ldots, m-1$
`Schritt 2:`	w	$\mapsto c$ `mittels FFT der Länge` m
`Schritt 3:`	a_0	$:= \mathrm{Re}(c_0) + \mathrm{Im}(c_0)$
	a_m	$:= \mathrm{Re}(c_0) - \mathrm{Im}(c_0)$
`Schritt 4:`	`für`	$k = 1$ `bis` $m \div 2$
		$f := (c_k + \bar{c}_{m-k})/2 \in \mathbb{C}$
		$h := (c_k - \bar{c}_{m-k})/2 \in \mathbb{C}$
		$g := h \cdot (\sin \pi k/m + i \cos \pi k/m) \in \mathbb{C}$
		$a_k - ib_k := f - g$
		$a_{m-k} + ib_{m-k} := f + g$

$\div$ ist das Zeichen für Integer-Division, also Abhacken zur ganzen Zahl.

Für die Rücktransformation hat man diese Schritte in umgekehrter Richtung zu vollziehen:

`Schritt 1':`	`für`	$k = 1, \ldots, m \div 2$
		$f := ((a_k - ib_k) + (a_{m-k} + ib_{m-k}))/2$
		$g := (-(a_k - ib_k) + (a_{m-k} + ib_{m-k}))/2$
		$h := g \cdot (\sin \pi k/m - i \cos \pi k/m)$
		$c_k := f + h$
		$c_{m-k} := \bar{f} - \bar{h}$
`Schritt 2':`	c_0	$:= (a_0 + a_m)/2 + i(a_0 - a_m)/2$
`Schritt 3':`	c	$\mapsto w$ `mittels FFT der Länge` m
`Schritt 4':`	y_{2j}	$:= \mathrm{Re}(w_j),\ y_{2j+1} := \mathrm{Im}(w_j),\ j = 0, \ldots m-1.$

Datenmenge mit vollständig entkoppelten Polynom-Segmenten überlegen. Die so entstehendenen **Polynom-Splines** haben sich als das perfekte Material für viele Interpolations- und Approximations-Aufgaben erwiesen. Der als Motto angeführte Schoenberg-Ausspruch kann also mit Berechtigung variiert werden zu

> *Polynome sind wunderbar, aber erst nachdem sie ein bißchen eingekerbt worden sind.*

Der englische Name „spline“ ist ins Deutsche ohne Übersetzung übernommen worden, „der Spline“ oder „die Spline“ ist noch unentschieden.

Eine quadratische Spline-Funktion und drei ihrer Parabel-Segmente.

Dazu beachtet man, daß links und rechts von Knoten x_i die Polynome p_{i-1} und p_i zuständig sind und daß die Forderung (5.2) in der Definition 5.1 verlangt

$$D^k p_{i-1}(x_i - 0) = D^k p_i(x_i + 0) \qquad \text{für} \quad k = 0, \ldots, m-2,$$

also

$$p_i(x) = p_{i-1}(x) + \gamma_i \cdot (x - x_i)^{m-1}$$

oder

$$s(x) = \begin{cases} p_{i-1}(x) & \text{für } x < x_i, \\ p_{i-1}(x) + \gamma_i \cdot (x - x_i)^{m-1} & \text{für } x \geq x_i. \end{cases}$$

Beides preßt man in eine einzige Formel durch folgende

Definition 5.2 *Abgeschnittene Potenzfunktion*

$$(x - x_i)_+^{m-1} := \begin{cases} 0 & \text{falls } x \leq x_i, \\ (x - x_i)^{m-1} & \text{falls } x > x_i. \end{cases}$$

Abgeschnittene Potenzfunktionen sind also Splines mit *einem* Knoten. $m = 1$ ist die Sprungfunktion. Man beachte, daß „$_+$“ **kein linearer Operator** ist, denn

$$(x_i - x)_+^{m-1} \neq (-1)^{m-1}(x - x_i)_+^{m-1}.$$

Es gilt vielmehr

$$(x_i - x)_+^{m-1} + (-1)^{m-1}(x - x_i)_+^{m-1} = (x_i - x)^{m-1},$$

wie man durch Betrachtung der linken Seite für $x < x_i$ und $x > x_i$ leicht nachweist, (außer bei $m = 1$ und für $x = x_i$).

Ebenfalls zu beachten ist, daß abgeschnittene Potenzfunktionen genau wie die normalen Potenzen $(x - x_i)^{m-1}$ differenziert und integriert werden dürfen mit etwas Vorsicht im Falle $m = 2$:

$$\begin{aligned} D(x - x_i)_+^{m-1} &= (m-1)(x - x_i)_+^{m-2}, \\ &\quad \text{(nur einseitig bei } m = 2 \text{ und } x = x_i), \\ \int_{-\infty}^{x} (t - x_i)_+^{m-1} dt &= \frac{1}{m}(x - x_i)_+^m. \end{aligned}$$

Mit dieser Vorarbeit ergibt sich

Satz 5.3 *Eine Basis für Polynom-Splines*

$$\mathcal{S}_m(x_1, \ldots, x_n) = \operatorname{span}\{1, x, \ldots, x^{m-1}, (x_2 - x)_+^{m-1}, \ldots, (x_{n-1} - x)_+^{m-1}\}.$$

7. Man überlege sich, daß zwei einfache Knoten x_i und $x_i + \varepsilon$ für $\varepsilon \to 0$ in einen doppelten Knoten übergehen: $(x_i - x)_+^{m-1}$ und $(x_i + \varepsilon - x)_+^{m-1}$ spannen den gleichen Raum auf wie $(x_i - x)_+^{m-1}$ und $[(x_i + \varepsilon - x)_+^{m-1} - (x_i - x)_+^{m-1}]/\varepsilon$. Diese zweite Funktion diskutiere man für $x < x_i$, für $x_i < x < x_i + \varepsilon$ und für $x > x_i + \varepsilon$.

Man beachte **die Sonderrolle der Endpunkte** x_1 und x_n, die übrigens in der Literatur häufig nicht zu den Knoten gerechnet werden, weil dort keine Stetigkeits-Bedingung (5.2) gilt und sie auch nicht in der Basis auftreten. Sie besitzen daher auch keine Multiplizität. Nach dem Motto „aus den Augen aus dem Sinn“ ist die Wahl $x_1 = -\infty$ und $x_n = +\infty$ zu empfehlen, zumal dann die Splines auf der gesamten reellen Achse definiert sind, $s\colon \mathbb{R} \to \mathbb{R}$. Man kann aber auch die beiden Endpunkte x_1 und x_n als m-fache Knoten ansehen und $s(x) := 0$ für $x < x_1$ und $x \geq x_n$ festlegen gemäß der totalen Entkoppelung an einem m-fachen Knoten. Dem entspricht die Ersetzung von $1, x, \ldots, x^{m-1}$ bzw. $1, (x_n - x), \ldots, (x_n - x)^{m-1}$ durch $(x_n - x)_+^k$ für $k = 0, \ldots, m-1$ in der Basis 5.3. Bei dieser Konzeption hat man $2(m-1)$ Knoten mehr, also dann $N := n+2m-2$ statt n. Die Dimension des Spline-Raumes hat sich dabei natürlich nicht verändert, bleibt also $n + m - 2 = N - m$. Dieser Standpunkt ist üblich im CAD-Bereich.

5.2 Spline-Interpolation

Die Polynom-Splines mit einfachen oder mehrfachen Knoten sind eine sehr vielseitige und leicht handhabbare, auf Computer zugeschnittene Funktionsklasse. Sie bilden ein ideales Instrumentarium für viele Interpolations- und Approximations-Aufgaben. Dabei wird stets m sehr klein gewählt, meist $m = 3$ oder 4, während n häufig sehr groß ist. Die Wahl der Knoten bleibt in der Regel dem Anwender überlassen, weil eine automatische Positionierung bisher kaum möglich ist.

Wegen $\dim \mathcal{S}_m(x_1, \ldots, x_n) = n + m - 2$ ist zu erwarten, daß $n + m - 2$ Interpolationsbedingungen erfüllbar sind und für die Eindeutigkeit auch gebraucht werden. Man sieht auch sofort, daß — anders als bei der Polynom-Interpolation — die Stützstellen sicherlich nicht ganz beliebig gewählt werden dürfen, zum Beispiel höchstens m Stützstellen in einem Intervall $[x_i, x_{i+1}]$.

Ohne Beweis sei die **Verschränkungs-Bedingung von Schoenberg und Whitney** (1953) als notwendige und hinreichende Bedingung für Existenz und Eindeutigkeit des Spline-Interpolanten angegeben.

Für $m \geq 2$, einfache Knoten und Stützstellen, lautet sie

$$\left.\begin{array}{l} \text{mindestens } i \text{ Stützstellen} < x_{i+1} \\ \text{mindestens } i \text{ Stützstellen} > x_{n-i} \end{array}\right\} \qquad i = 1, \ldots, n + m - 2, \tag{5.3}$$

für alle $x_0, x_{-1}, \ldots < x_1$ und für alle $x_{n+1}, x_{n+2}, \ldots > x_n$.

Wenn eine j-fache Stützstelle auf einen k-fachen Knoten fällt, dann sollte $j+k \leq m$

Das zeigt auch, daß $s \in C^1[x_1, x_n]$. Die zweite Ableitung hat dagegen am (doppelten) Knoten x_i im allgemeinen einen Sprung, nämlich nach (5.5)

$$\begin{aligned} s''(x_i+0) - s''(x_i-0) &= (6z_i - 4y'_i - 2y'_{i+1})/h_i \\ &\quad -(-6z_{i-1} + 2y'_{i-1} + 4y'_i)/h_{i-1}. \end{aligned} \tag{5.6}$$

Damit wenden wir uns dem Fall zu, daß $x_2 \ldots, x_{n-1}$ *einfache* Knoten und *einfache* Stützstellen sind. Den kubischen Ansatz (5.3), (5.4) übernehmen wir, wobei jetzt die $y'_1, \ldots, y'_n$ unbekannt sind. Damit auch die Forderung (5.2) von Definition 5.1 erfüllt ist, muß s'' stetig sein, also (5.6) verschwinden für $i = 2, \ldots, n-1$. Das gibt die notwendige und hinreichende Bedingung für den kubischen Spline-Interpolanten mit einfachen Knoten/Stützstellen:

$$\begin{aligned} y'_{i-1}/h_{i-1} + (2/h_{i-1} + 2/h_i)y'_i + y'_{i+1}/h_i &= 3(z_{i-1}/h_{i-1} + z_i/h_i), \\ &\quad i = 2, \ldots, n-1. \end{aligned} \tag{5.7}$$

Das sind $n-2$ lineare Gleichungen für die n Unbekannten $y'_1, \ldots, y'_n$. Das Gleichungssystem ist unterbestimmt und das war zu erwarten: Wir wissen ja, daß der Raum der kubischen Splinefunktionen die Dimension $n+2$ hat! Man kann also noch zwei weitere Stützpunkte interpolieren, $s(x_0) = y_0$ und $s(x_{n+1}) = y_{n+1}$ für ein $x_0 < x_1$ und ein $x_{n+1} > x_n$ oder zusätzlich noch zwei Randbedingungen erfüllen. Drei Möglichkeiten folgen.

Typ 1: x_1 und x_n doppelte Stützstelle, also y_1 *und* y'_1 bzw. y_n *und* y'_n gegeben. Damit ist (5.7) ein **tridiagonales, positiv definites** Gleichungssystem der Ordnung $n-2$ für $y'_2, \ldots, y'_{n-1}$.

Gauß-Elimination ohne Pivotsuche ist möglich, wobei der Aufwand $O(n)$ und nicht etwa $O(n^3/3)$ ist, weil man immer nur *ein* Subdiagonal-Element zu eliminieren hat. Man zeige durch Induktion, daß die Eliminations-Faktoren immer $< 1/2$ und das i-te Pivotelement $> 1.5/h_i + 2/h_{i+1}$ ist, $i = 1, \ldots, n-2$.

Dieser sogenannte **komplette kubische Spline-Interpolant** wird also konstruiert mit einer festen Anzahl von arithmetischen Operationen pro Datenpunkt (Stützpunkt). Das ist viel besser als bei Polynomen.

Typ 2: Man kann auch als Randbedingungen $s''(x_1)$ und $s''(x_n)$ vorschreiben anstelle von $s'(x_1)$ und $s'(x_n)$. Nach (5.5) für $i = 1$, $t = 0$ und für $i = n-1$, $t = 1$ verlangt das

$$\begin{aligned} 2y'_1/h_1 + y'_2/h_1 &= 3z_1/h_1 - y''_1/2 \\ y'_{n-1}/h_{n-1} + 2y'_n/h_{n-1} &= 3z_{n-1}/h_{n-1} + y''_n/2 \end{aligned} \tag{5.8}$$

zusätzlich zu (5.7). y''_1 und y''_n sind die für s'' bei x_1 und x_n vorgeschriebenen Werte. Wieder hat man ein tridiagonales, positiv-definites Gleichungssystem, jetzt der Ordnung n für $y'_1, \ldots, y'_n$. Das für Typ 1 Gesagte gilt auch hier.

Ohne Beweis sei angegeben

Satz 5.5 *Restglied*

Die kubische Splinefunktion s mit Knoten $x_1 < \ldots < x_n$ interpoliere dort $f \in C^4[x_1, x_n]$ mit *einer* der Randbedingungen:

$$\begin{array}{lrcl} \text{entw. Typ 1:} & s'(x_1) & = & f'(x_1) \text{ und } s'(x_n) = f'(x_n), \\ \text{oder Typ 2:} & s''(x_1) & = & f''(x_1) \text{ und } s''(x_n) = f''(x_n), \\ \text{oder Typ 3:} & s'(x_1) & = & s'(x_n) \text{ und } s''(x_1) = s''(x_n), \\ & & & \text{falls auch } f \text{ periodisch.} \end{array}$$

Dann gilt im Intervall $[x_i, x_{i+1}]$, $i = 1, \ldots, n-1$:

$$\begin{aligned} |f - s| &\leq \frac{3}{64} L h_i^2 H^2, \\ |f' - s'| &\leq \frac{3}{16} L h_i H^2, \\ |f'' - s''| &\leq \frac{3}{8} L H^2, \\ |f''' - s'''| &\leq \frac{1}{2} L (H^2/h_i + h_i). \end{aligned}$$

$L := \max|f''''(x)|$, $h_i := x_{i+1} - x_i$, $H := \max(h_1, \ldots, h_{n-1})$.

Der Beweis ist nicht schwer mit technischen Hilfsmitteln, die in der Vorlesung *Numerische Mathematik III* eingeführt werden. Diese Schranken sind jedoch noch nicht optimal.

Man beachte, daß die natürlichen Randbedingungen (Typ 2°) in Satz 5.5 nicht zugelassen sind!

5.4 Die erste Integral-Relation (für $m = 4$)

Es sei $x_1 < \ldots < x_n$ eine Folge einfacher Knoten und s der kubische Spline-Interpolant eines beliebigen $f \in C^2[x_1, x_n]$ mit Randbedingungen wie diskutiert in Abschnitt 5.3. Also ist

$$(f - s)(x_i) = 0 \qquad \text{für} \quad i = 1, \ldots, n \tag{5.9}$$

und bei

$$\begin{array}{ll} \text{Typ 1:} & (f' - s')(x_1) = (f' - s')(x_n) = 0 \\ \text{Typ 2°:} & s''(x_1) = s''(x_n) = 0 \\ \text{Typ 3:} & s(x_1) = s(x_n), \quad s'(x_1) = s'(x_n) \end{array} \tag{5.10}$$

Existenz und Eindeutigkeit wurden schon nachgewiesen. (Es sei $n \geq 2$ und bei Typ 3 auch f periodisch, also $f(x_1) = f(x_n)$, $f'(x_1) = f'(x_n)$.)

Im folgenden Kreis-Diagramm stelle man sich alle möglichen Interpolanten f mit stückweise stetigem f'' nach wachsendem $\int f''^2$ irgendwie radial angeordnet. Je ein kleiner Kreis markiert die drei kubischen Spline-Interpolanten.

Alle Funktionen f mit $f(x_i) = y_i$ für $i = 1, \ldots, n$ radial angeordnet nach wachsendem $\int f''(x)^2\, dx$.

Die natürlichen Randbedingungen $f''(x_1) = f''(x_n) = 0$ sind also das, was sich von selbst einstellt, wenn man $\int f''^2$ ganz ohne Randbedingungen minimiert, allein unter den Interpolations-Bedingungen $y_i = f(x_i)$, $i = 1, \ldots, n$. **Trotzdem sind natürliche Randbedingungen keine guten Randbedingungen und selbst grobe Schätzungen für $s'(x_1), s'(x_n)$ sind besser als die Festlegung** $s''(x_1) = s''(x_n) = 0$.

Übrigens gelten alle Aussagen auch in Anwesenheit doppelter Stützstellen, weil dann wie diskutiert die Interpolations-Aufgabe in entkoppelte Teil-Probleme mit nur einfachen Stützstellen und Randbedingungen vom Typ 1 zerfällt, für die die Aussagen richtig sind.

Zum Namen der Splinefunktion

„Spline“ bezeichnet im Englischen die biegsame Leiste aus Stahl oder astfreiem Holz, mit der Zimmerleute und Schiffsbauer geschwungene Kurven erzeugen, indem sie diese mit Stiften oder Gewichten an bestimmten Stellen fixieren. Die Spline (deutsch Straaklatte) nimmt dabei eine Gestalt an, bei der die Verformungsenergie E minimal ist:

$$E = \int \kappa(\sigma)^2 d\sigma = \min!$$

Dabei ist σ die Bogenlänge der Kurve $x(\sigma), y(\sigma)$ und $\kappa(\sigma) = y''x' - x''y'$ ihre Krümmung. Für hinreichend flach verlaufende Kurven (nur eine Frage des Maßstabs) ist $x(\sigma) \approx \sigma$ und $\kappa(\sigma) \approx y''$, also näherungsweise

$$\int y''^2 dx = \min!$$

Man sieht, daß eine wenig gekrümmte Straaklatte den Verlauf einer natürlichen Splinefunktion annimmt, die dieses Integral ebenfalls minimiert.

x ist also eine im Ausdruck gebundene Variable (wie eine Integrations-Variable oder ein Summations-Index), und kommt im Gegensatz zu t außen nicht mehr vor. Die dividierten Differenzen einer Funktion wurden in Abschnitt 4.3 eingeführt und es wurde gezeigt, daß sie Linearkombinationen sind der Funktionswerte für $x = x_k, \ldots, x_{k+m}$ und der Ableitungen bis zur Ordnung $\mu - 1$ an einer μ-fachen Stützstelle, hier also Linearkombination von

$$(x_i - t)_+^{m-1}, \ldots, (x_i - t)_+^{m-\mu} \qquad \text{falls} \quad x_i \ \mu\text{-fach.} \tag{5.13}$$

Genau diese Funktionen treten auch in (5.11) auf. Jetzt zählt zwar i ab 1 statt ab $m+1$, aber die zugehörigen Terme verschwinden im Grundgebiet $[x_m, x_{n+1}]$.

Man sollte sich unbedingt die B-Splines der Ordnung $m = 1$, 2 und 3 skizzieren. $m = 1$ sind die Treppenfunktionen, und erlauben nur einfache Knoten. $m = 2$ sind die Dachfunktionen, die auch doppelte Knoten zulassen. Es gilt:

$M_{k,m}(t) = 0$ für $t \geq x_{k+m}$. Denn dann sind alle in (5.12) auftretenden $x_k - t, \ldots, x_{k+m} - t \leq 0$ und werden in $(\ldots)_+$ durch 0 ersetzt.

$M_{k,m}(t) = 0$ für $t < x_k$. Denn dann sind umgekehrt alle $x_k - t, \ldots, x_{k+m} - t > 0$ und $(\ldots)_+ = (\ldots)$. Für solche t ist also in (5.12) nicht die dividierte Differenz der *abgeschnittenen* Potenzfunktion zu bilden, sondern der gewöhnlichen Potenz $(x-t)^{m-1}$. Das ist ein Polynom vom Grad $m-1$ in x und seine dividierte Differenz der Ordnung m verschwindet nach Aussage (P) am Ende von Abschnitt 4.3. Somit ist gezeigt, daß $M_{k,m}$ nur auf $[x_k, x_{k+m}[$ von Null verschieden sein kann. Keine Splinefunktion $s \neq 0$ der Ordnung m kann kleineren Support haben (Übungsaufgabe).

Im Grundgebiet $[x_m, x_{n+1}]$ verschwinden alle Terme in (5.13) mit $i \leq m$. Zu $M_{1,m}(t)$ trägt im Grundgebiet also nur $(x_{m+1} - t)_+^{m-1}$ bei. So überlegt man sich, daß die neue Basis (5.12) tatsächlich eine Staffelung der alten Basis (5.11) ist: Die beiden Basen sind ineinander umrechenbar und spannen somit den gleichen Raum auf.

Die Basis (5.12) ist nicht nur im Grundgebiet sondern auf der ganzen reellen Achse definiert, wobei jede Linearkombination $s(t)$ für $t < x_1 = \ldots = x_m$ und für $t \geq x_{n+1} = \ldots = x_{n+m}$ abrupt auf 0 abfällt.

$M_{k,m}$ ist leicht anzugeben falls $x_k = \ldots = x_{k+m-1} < x_{k+m}$. Links liegt ein m-facher Knoten vor und rechts ein einfacher Knoten; das erfordert

$$M_{k,m}(t) = \begin{cases} \gamma \cdot (x_{k+m} - t)^{m-1} & \text{für } x_k \leq t < x_{k+m}, \\ 0 & \text{sonst.} \end{cases}$$

Die Konstante ist nach Übungsaufgabe 4.1 in Abschnitt 4.3 $\gamma = 1/(x_{k+m} - x_k)^m$, aber das ist unerheblich. Analog ergibt sich für $x_k < x_{k+1} = \ldots = x_{k+m}$

$$M_{k,m}(t) = \begin{cases} (t - x_k)^{m-1}/(x_{k+m} - x_k)^m & \text{für } x_k \leq t < x_{k+m}, \\ 0 & \text{sonst.} \end{cases}$$

Insbesondere ist

$$M_{k,1}(t) = \begin{cases} 1/(x_{k+1} - x_k) & \text{falls } x_k \leq t < x_{k+1}, \\ 0 & \text{sonst.} \end{cases}$$

Übungsaufgabe 5.1: Zeige

$$[x_0, \ldots, x_m]f = \frac{1}{(m-1)!} \int_{x_0}^{x_m} M_{0,m}(t) D^m f(t)\, dt.$$

Tip:

$$f(x) = \sum_{k=0}^{m-1} \frac{(x-x_0)^k}{k!} D^k f(x_0) + \frac{1}{(m-1)!} \int_{x_0}^{x_m} (x-t)_+^{m-1} D^m f(t)\, dt$$

(Taylorformel mit Restglied) und (5.12) verwenden.

Übungsaufgabe 5.2: *Spline-Verbindungsstück*

Man zeige: Für jedes p und $q \in \mathbb{P}_{m-1}$ gibt es genau eine Splinefunktion $s\colon \mathbb{R} \to \mathbb{R}$ der Ordnung m mit Knoten $x_1 \leq \ldots \leq x_m$, so daß

$$s(x) = \begin{cases} p(x) & \text{für } x < x_1, \\ q(x) & \text{für } x \geq x_m. \end{cases}$$

Insbesondere gilt $p = q = 0 \Rightarrow s = 0$.

Tip: Es ist zu zeigen, daß die m Funktionen $(x_i - x)_+^{m-1-}$ auf $]-\infty, x_1[$ linear unabhängig sind. Dort sind alle $x_i - x > 0$. Man sollte unbedingt mit dem Fall lauter einfacher Knoten anfangen: wieso sind die $(x_1 - x)^{m-1}, \ldots, (x_m - x)^{m-1}$ linear unabhängig, wenn $x_1 < \ldots < x_m$?

Regel auf die explizite, exakte Bildung von F'' und kommen damit dem Anwender sehr entgegen. Mehr dazu in *Numerischer Mathematik III*.

Die geschlossene Auflösung von $f(x) = 0$ ist in der Praxis fast immer unmöglich und in den wenigen Ausnahmefällen äußerst unpraktisch (z.B. die Cardanoschen Formeln für die Wurzeln einer kubischen Gleichung). Deshalb werden iterative Methoden eingesetzt. Sie produzieren eine Folge von Näherungen

$$x^0, x^1, x^2, \ldots \quad \in \mathbb{R}^n,$$

die im günstigen Fall gegen eine Nullstelle $\bar{x}$ von f konvergiert. Meistens trifft das zu, wenn x^0 „hinreichend nahe“ an $\bar{x}$ liegt, und es ist Sache des Anwenders, ein solches x^0 zu beschaffen.

ϕ sei die **Verfahrens-** oder **Iterations-Funktion**, also die Rechenvorschrift, nach der x^i aus seinen Vorgängern ermittelt wird. Mit ϕ_f statt ϕ würde man den selbstverständlichen Umstand besser zum Ausdruck bringen, daß die Verfahrensfunktion von f abhängt. Mit f sei auch ϕ hinreichend glatt. Man spricht von

1-**Schritt-Verfahren**, wenn $x^i = \phi(x^{i-1})$:	benötigt den **Startwert** x^0,
2-**Schritt-Verfahren**, wenn $x^i = \phi(x^{i-2}, x^{i-1})$:	benötigt **Startwerte** x^0, x^1,

usw.

Formal kann man übrigens auf diese Unterscheidung verzichten und jedes m-Schritt-Verfahren im $\mathbb{R}^n$ interpretieren als ein 1-Schritt-Verfahren im $\mathbb{R}^{mn}$, indem man die Vektoren $x^{i-m}, \ldots, x^{i-1}$ untereinander schreibt und zu einem Blockvektor X^{i-1} zusammenfaßt. ϕ schiebt dann die Komponenten in diesem Blockvektor um je n Plätze nach oben und füllt unten n neue Komponenten ein. Deshalb kann man ohne Einbuße an Allgemeinheit theoretische Konzepte nur für 1-Schritt-Verfahren formulieren. Ein Beispiel dafür ist der Fixpunkt im nächsten Absatz.

Die Iteration konvergiert, wenn $x^i \to \bar{x}$ für $i \to \infty$, d.h. $\lim\limits_{i\to\infty} x^i$ muß existieren *und* Nullstelle von f sein. Bei 1-Schritt-Verfahren kommen also nur solche in Frage mit

$$f(\bar{x}) = 0 \qquad \Longleftrightarrow \qquad \bar{x} = \phi(\bar{x}),$$

d.h., die Nullstellen von f sind die sogenannten **Fixpunkte** von ϕ. Nullstellen-Bestimmung und Fixpunkt-Bestimmung sind äquivalent. Die Menge der Startpunkte, für die Iteration konvergiert, ist das **Einzugsgebiet** des Verfahrens. Wenn eine solche Menge a priori angegeben werden kann, heißt das Verfahren dafür **global konvergent**. Ist dagegen die Konvergenz nur gesichert für Startwerte in einer hinreichend kleinen Umgebung eines Fixpunktes, so ist das Verfahren nur **lokal konvergent**. Als letztes dieser Auflistung von Bezeichnungen noch

Definition 6.1 *Konvergenz-Ordnung und -Faktor*

> Falls $x^i \to \bar{x}$ für $i \to \infty$ und $\|x^{i+1} - \bar{x}\| \le c \cdot \|x^i - \bar{x}\|^p$ für ein $c < \infty$, dann ist das größte solche p bzw. das Supremum die **Konvergenz-Ordnung** des Verfahrens.

Dies ist das wohl-bekannte **Newton-Verfahren** im $\mathbb{R}^n$:

```
Wähle x^0 ∈ R^n möglichst gut;
für i := 0, 1, 2, ...:
    berechne b := f(x^i) ∈ R^n;
    berechne A := f'(x^i) ∈ R^{n,n};
    löse Av = b nach v ∈ R^n;
    x^{i+1} := x^i − v.
```

Es wird ausdrücklich empfohlen, die Elemente von $f'(x)$ stets auf *zwei* Weisen zu ermitteln:

a) durch direkte Bildung der partiellen Ableitungen $\partial f_j/\partial x_k$,

b) durch unmittelbare Linearisierung von $f(x+\delta x) = b + A\delta x + \ldots$, also Vernachlässigung aller Produkte $\delta x_j \cdot \delta x_k$.

Noch sicherer ist der Einsatz eines Formel-Manipulations-Programmes mit Fortran- oder Pascal-Ausgabe.

Im univariaten Fall hat man die Vorschrift $x^i \mapsto x^{i+1} = x^i - f(x^i)/f'(x^i)$. Die Iterations-Funktion ist also für das univariate Newton-Verfahren

$$\phi(x) = x - f(x)/f'(x), \quad \phi' = ff''/f'^2, \quad \phi'' = f''/f' + f \cdot (f'''/f'^2 - 2f''^2/f'^3)$$

Für eine einfache Nullstelle ist $f(\bar{x}) = 0$, $f'(\bar{x}) \neq 0$, also $\phi(\bar{x}) = \bar{x}$, $\phi'(\bar{x}) = 0$, $\phi''(\bar{x}) = f''(\bar{x})/f'(\bar{x})$. In der Umgebung einer einfachen Nullstelle ist das Newton-Verfahren quadratisch konvergent. Im folgenden Abschnitt wird das auch auf andere Weise gezeigt.

6.2 Die Grund-Verfahren im univariaten Fall $f\colon \mathbb{R} \to \mathbb{R}$

Die **Bisektion** ist das einfachste und vielleicht robusteste Verfahren. Es startet mit zwei Punkten $a < b \in \mathbb{R}$, an denen f entgegengesetztes Vorzeichen hat:

$$f(a) \cdot f(b) < 0,$$

so daß das stetige f dazwischen mindestens eine Nullstelle hat. Man nennt deshalb $[a, b]$ ein Einschließungs-Intervall. Berechnet man f in der Intervall-Mitte (oder an einer anderen Zwischenstelle), so kann man anhand des dortigen Vorzeichens die eine Hälfte des Intervalls ausscheiden und auf diese Weise $b - a$ halbieren.

Wieder kann man das asymptotische Verhalten der Iteration bei einer einfachen Nullstelle gültig analysieren anhand der Funktion $f(x) = x + cx^2$ im Intervall $[-\varepsilon, +\varepsilon]$, $|c|\varepsilon \ll 1$. Man findet leicht, daß jeweils drei Schritte einen Zyklus bilden, bei dem ε übergeht in $c^2\varepsilon^3$. Das ist äquivalent zu einer Konvergenz-Ordnung $3^{1/3} \doteq 1.442$. (Wenn die Iterationsfunktion ϕ wie hier Fall-Unterscheidungen enthält, dann kann die Fehler-Abnahme stufenweise passieren und Definition 6.1 muß durch eine Mittelbildung ergänzt werden.)

Die drei Funktionsauswertungen in einem Zyklus vergleichen sich günstig mit der Berechnung von f, f', f'', die andere kubisch konvergente Verfahren pro Schritt benötigen.

Eines der schnellsten Verfahren ist die **Sekanten-Methode**, also das reine 2-Schritt-Verfahren $x^{i-1}, x^i \Rightarrow x^{i+1}$, bei dem x^{i+1} die Nullstelle der Sekante

$$y = y^i + \frac{y^i - y^{i-1}}{x^i - x^{i-1}}(x - x^i), \qquad y^j := f(x^j)$$

ist:

$$0 = y^i + \frac{y^i - y^{i-1}}{x^i - x^{i-1}}(x^{i+1} - x^i). \tag{6.2}$$

Auflösen nach x^{i+1} gibt den Algorithmus

```
wähle x^0;  y^0 := f(x^0);
wähle x^1;  y^1 := f(x^1);
für i := 1, 2, 3, ...
    x^{i+1} := x^i − (x^i − x^{i−1}) · y^i/(y^i − y^{i−1});
    y^{i+1} := f(x^{i+1});
    falls |y^{i+1}| ≤ Toleranz: EXIT
```

Weil keine Fall-Unterscheidungen vorkommen, ist dieses Verfahren besonders leicht zu analysieren. Nach der Restglied-Formel (Satz 4.10 für $n = 2$) gilt

$$0 = f(\bar{x}) = y^i + \frac{y^i - y^{i-1}}{x^i - x^{i-1}}(\bar{x} - x^i) + (\bar{x} - x^i)(\bar{x} - x^{i-1})f''(\xi)/2.$$

Dabei liegt ξ zwischen x^{i-1}, x^i und $\bar{x}$. Zieht man davon (6.2) ab, so ergibt sich

$$\frac{y^i - y^{i-1}}{x^i - x^{i-1}}(x^{i+1} - \bar{x}) = \frac{1}{2}f''(\xi) \cdot (x^i - \bar{x})(x^{i-1} - \bar{x})$$

oder

$$x^{i+1} - \bar{x} = (x^i - \bar{x})(x^{i-1} - \bar{x}) \cdot f''(\xi)/2f'(\xi'),$$

ξ' zwischen x^{i-1} und x^i. Falls f'' stetig ist und die Iteration gegen eine einfache Nullstelle konvergiert, ist $|f''/2f'| \le c$ und es gilt das Fehlergesetz (6.1) mit der Konvergenz-Ordnung $(1 + \sqrt{5})/2 \doteq 1.618$. Das gibt bis zu 24% Arbeitsersparnis

Zu letzterem sei sofort daran erinnert, daß es sich dabei *nicht* um Schwierigkeiten der Numerik, der Computer oder der Rundungsfehler handelt, sondern um eine Schwierigkeit, für die allein der Anwender die Verantwortung trägt: Winzige Fehler in den Eingabe-Daten, z.B. obige Taylor-Koeffizienten $c_0, \ldots, c_n$, können die Nullstellen unter Umständen soweit verfälschen, daß ihre Berechnung sinnlos wird.

Das **Laguerre-Verfahren** ist vielleicht die beste der speziell auf Polynome $P(x)$ zugeschnittenen Methoden. Wie die Newton-Iteration ist es ein 1-Schritt-Verfahren, $x^i \mapsto x^{i+1} = \phi(x^i)$ mit

$$\phi(x) = x - nP(x)/\left(P'(x) \pm W(x)\right),$$

wobei $W(x) = \sqrt{(n-1)^2 P'(x)^2 - (n-1)nP(x)P''(x)}$.

Hier ist n der Grad des Polynoms P und sein Auftreten zeigt, daß das Verfahren auf Polynome zugeschnitten ist. Das Vorzeichen von W sollte so gewählt werden, daß der Nenner von ϕ den *größeren* Betrag hat. x und P dürfen komplex sein. Wie in Abschnitt 4.2 besprochen, kann man P, P', P'' mit dem dreizeiligen Hornerschema parallel berechnen:

$r := 0;\ q := 0;\ p := c_n;$
`für` $k := n-1$ `abwärts bis` 0: { $r := rx + q;$
$q := qx + p;$
$p := px + c_k;$ }
$p' := q;\ p'' := r + r;$

Die Rekursion für P ergibt sich aus $P(x) = (\ldots(c_n x + c_{n-1})x + \ldots + c_1)x + c_0$ und die für P' und $P''/2$ durch Differenzieren dieser Identitäten in x.

Das Laguerre-Verfahren hat folgende Vorteile:

1. In der Umgebung einer einfachen Nullstelle ist es kubisch konvergent: aus $x = \bar{x} + \varepsilon$ folgt

$$\begin{aligned} P(x) &= A\varepsilon + B\varepsilon^2 + O(\varepsilon^3) \\ P'(x) &= A + 2B\varepsilon + O(\varepsilon^2), \qquad A \neq 0 \\ P''(x) &= 2B + O(\varepsilon) \\ W(x) &= (n-1)A + (n-2)B\varepsilon + O(\varepsilon^2) \\ \phi(x) &= \bar{x} + O(\varepsilon^3) \end{aligned}$$

   Zwei Laguerre-Schritte $\varepsilon \mapsto \varepsilon^3 \mapsto \varepsilon^9$ sind im Aufwand vergleichbar mit 3 Newton-Schritten $\varepsilon \mapsto \varepsilon^2 \mapsto \varepsilon^4 \mapsto \varepsilon^8$. Wieder schneidet das Newton-Verfahren nicht besonders gut ab.

2. Das Verfahren ist sehr viel robuster als das Newton-Verfahren, weil der Nenner zufällig klein wird nur für P' *und* $P'' \approx 0$, was sehr viel seltener ist als

Andererseits ist $Q(0) = n > 0$. Bei $\lambda_1 \leq \ldots \leq \lambda_i < x < \lambda_{i+1} \leq \ldots \leq \lambda_n$ liegen also $\phi_\pm(x)$ zwischen x und λ_i bzw. λ_{i+1}. Das gilt auch für $i = 0$ und $i = n$, wenn man $-\infty$ und $+\infty$ identifiziert und $\mathbb{R}$ ringförmig interpretiert. ■

Echte Schwierigkeiten mit Polynomen beginnen erst, wenn man schon eine Nullstelle λ berechnet hat und nun eine zweite, dritte, ... berechnen möchte. Dann kann man die gefundene Nullstelle λ **abdividieren** (**Deflation**) durch Koeffizienten-Vergleich in

$$P(x) \equiv (x - \lambda)Q(x).$$

Das sich daraus ergebende Rekursionschema für die Koeffizienten von $Q \in \mathbb{P}_{n-1}$

$$p_k = q_{k-1} - \lambda q_k, \qquad q_{-1} := q_n := 0,$$

kann man für $k = n, \ldots, 0$ oder für $k = 0, \ldots, n$ stricken. Ersteres stimmt übrigens mit dem einzeiligen Hornerschema überein. Rundungsfehler in diesem Schritt vererben sich auf die Berechnung aller folgenden Nullstellen. Eine Analyse zeigt, daß die Richtung $k = n, \ldots, 0$ nur gefährlich ist für alle betragskleineren Nullstellen (analog die Richtung $k = 0, \ldots, n$ für alle betragsgrößeren Nullstellen). Es kommt also darauf an, daß man die Nullstellen $\lambda_1, \ldots, \lambda_n$ von P in der Reihenfolge (ungefähr) wachsender bzw. fallender Beträge bestimmt, zum Beispiel, indem man die kubisch konvergente Laguerre-Iteration immer am Nullpunkt startet, so daß sie in der Regel zur betragskleinsten Nullstelle konvergiert (sofern eine solche klar ausgeprägt ist).

Ein **oft gegebener** aber **gänzlich falscher** Ratschlag ist, am Schluß für jede gefundene Nullstelle nochmal einen Laguerre-Schritt mit dem Orginal-Polynom zu machen. Das zerstört alle Korrelationen in den Fehlern von mehrfachen Nullstellen und damit ihre Akzeptabilität.

Ebenfalls wenig bewährt hat sich die Empfehlung, Nullstellen nur zu **lähmen** statt explizit abzudividieren. Das heißt, man arbeitet für die zweite Nullstelle formal mit der Funktion $P(x)/(x - \lambda_1)$. Zum Beispiel ist dann

$$(P(x)/(x - \lambda_1))' = P'(x)/(x - \lambda_1) - P(x)/(x - \lambda_1)^2$$

und das Newton-Verfahren als Beispiel wird damit

$$x \mapsto \phi(x) = x - P(x)/(P'(x) - P(x)/(x - \lambda_1)).$$

Diese Iterationsfunktion hat zwar eine *hebbare* Singularität bei $x = \lambda_1$, ist aber in der Umgebung doch numerisch ziemlich wackelig und das macht das Vorgehen bei mehrfachen Nullstellen unzuverlässig.

6.4 Kontraktion und Fixpunktsatz

Iterationen lassen sich auch im Kopf statt im Computer ausführen! Das dient dann nur dazu, die Existenz und eventuell die Eindeutigkeit der Lösung eines Problems

Immer wird ϕ als stetig vorausgesetzt. Dann gibt es zum Beispiel ein simples Kriterium für die Existenz von Fixpunkten: wenn ϕ ein abgeschlossenes, beschränktes Intervall $[a, b]$ in sich selbst abbildet, muß es dort mindestens einen Fixpunkt haben. In der Tat, $\phi(x) - x$ ist ≥ 0 für $x = a$ und ≤ 0 für $x = b$, es muß also dazwischen mindestens eine Nullstelle geben.

Ein solches Intervall bildet einen Käfig für die Iterationsfolge, denn mit $x^i \in [a, b]$ sind auch alle Nachfolger $\in [a, b]$. Alles das reicht jedoch nicht aus um die Konvergenz des Verfahrens zu garantieren: der Fixpunkt und das Einschließungs-Intervall sind zwar notwendig für die Konvergenz, jedoch nicht hinreichend. Zum Beweis genügt ein Beispiel, wie etwa die Iteration mit $\phi(x) := x^3/6 - 2x$ im Intervall $[-4, 4]$, welche nur für die Startwerte 0 und $\pm 2\sqrt{3}$ konvergiert, und für alle anderen $x^0 \in [-4, +4]$ in einen Zyklus um $\pm\sqrt{6}$ mündet. Insbesondere ist der einzige Fixpunkt im Intervall, $x = 0$, **abstoßend**, denn $\phi'(0) = -2$. Für die Konvergenz benötigt man jedoch offensichtlich einen **anziehenden** Fixpunkt $\bar{x}$, d.h. $\bar{x} = \phi(\bar{x})$ und $|\phi'(\bar{x})| < 1$. Wir erhöhen deshalb unsere Anforderungen an die Verfahrensfunktion ϕ durch die Einführung einer neuen Eigenschaft.

Definition 6.2

$\phi\colon [a, b] \to \mathbb{R}$ heißt **kontrahierend**, falls

$$|\phi(x) - \phi(y)| \leq \kappa \cdot |x - y| \qquad \forall x, y \in [a, b]$$

mit einer Lipschitz-Konstanten $\kappa < 1$.

Alle Sekanten müssen also eine Steigung zwischen $\pm\kappa$ besitzen. Insbesondere folgt aus dem Grenzfall $y \to x$, daß $|\phi'(x)| \leq \kappa$ sein muß, sofern ϕ differenzierbar ist. Diese Bedingung ist dann auch hinreichend:

$$|\phi'| \leq \kappa \quad \Longrightarrow \quad |\phi(x) - \phi(y)| \leq \int_y^x |\phi'| \leq \kappa \cdot (x - y) \quad \text{bei } x \geq y \text{ (o.B.d.A.).}$$

Eine kontrahierende Verfahrensfunktion ist die entscheidende Zutat zu einer Iteration mit a priori gesicherter Konvergenz, also genau das, was in diesem Abschnitt wichtig ist. Der folgende Satz belegt dies. Im erwähnten allgemeinen Szenario heißt er **Fixpunktsatz von Banach, Weissinger**.

Satz 6.3

Sei $\phi\colon [a, b] \to \mathbb{R}$ und $x^i := \phi(x^{i-1})$, $i = 1, 2, \ldots$. Falls

$$|\phi(x) - \phi(y)| \leq \kappa \cdot |x - y|, \qquad \forall x, y \in [a, b],\ \kappa < 1, \tag{6.3}$$

$$x^i \in [a, b], \qquad i = 0, 1, 2, \ldots, \tag{6.4}$$

Bemerkung: (6.9) ist eine a priori Fehlerschranke (also angebbar schon *vor* der Rechnung). Eine meist viel schärfere Fehlerschranke wird daraus, wenn man $i = 1$ wählt und ein beliebiges x^{j-1} als Startwert ansieht:

$$|x^j - \bar{x}| \leq \frac{\kappa}{1-\kappa}|x^j - x^{j-1}|.$$

Dies ist eine a posteriori Schranke, denn man muß zuvor x^{j-1} und x^j berechnet haben.

Solche Fehlerschranken spielen natürlich keine Rolle bei den eingangs ins Auge gefaßten Gedanken-Iterationen, wo man sich j schließlich beliebig groß denken kann.

Die Voraussetzung (6.4) im Fixpunktsatz kann a priori nicht überprüft werden. Im folgenden Korollar wird an dessen Stelle die leicht überprüfbare Voraussetzung gesetzt, daß neben x^0 und x^1 auch eine bestimmte Umgebung von x^1 in $[a, b]$ ist.

Korollar 6.4

Sei $x^0 \in [a, b]$ und $[x^1 - \rho, x^1 + \rho] \subseteq [a, b]$, wobei

$$x^1 := \phi(x^0) \qquad \text{und} \qquad \rho := \frac{\kappa}{1-\kappa} \cdot |x^1 - x^0|.$$

Dann sind alle $x^j \in [a, b]$, $j \in \mathbb{N}$.

Beweis: x^0 und $x^1 \in [a, b]$ nach Voraussetzung. Für $j \geq 2$ Induktion. Sei schon gezeigt, daß $x^j \in [a, b]$. Dann ist (6.11) schon anwendbar und liefert für $i = 1$:

$$|x^{j+1} - x^1| \leq \frac{\kappa}{1-\kappa}|x^1 - x^0|,$$

also $x^{j+1} \in [a, b]$ nach der jetzigen Voraussetzung. ■

6.5 Gauß-Newton-Verfahren für nicht-lineare Ausgleichsprobleme

Wie im Fall der linearen Ausgleichsrechnung (Abschnitt 3.7) stehen den n Variablen mehr als n Gleichungen gegenüber, jetzt jedoch im allgemeinen nicht linear. Bei Konsistenz hätte man also formal das System

$$f(\bar{x}) = 0 \qquad \text{mit} \quad f\colon \mathbb{R}^n \to \mathbb{R}^m,\ m > n$$

zu lösen. Wieder ist damit zu rechnen, daß es kein $\bar{x} \in \mathbb{R}^n$ gibt, welches alle m Gleichungen erfüllt, so daß es sich anbietet, stattdessen nach einem $\hat{x}$ zu suchen, welches f möglichst klein macht im Sinne der Euklidschen Norm:

$$\text{finde } \hat{x} \in \mathbb{R}^n, \quad \text{so daß } Z(x) := f(x)^T f(x) = \sum_{i=1}^{m} f_i(x)^2 \text{ minimal.}$$

ebener Rotationen oder Reflexionen:

$$Q^T A =: R = \begin{pmatrix} \bar{R} \\ 0 \end{pmatrix}, \qquad Q^T b =: c = \begin{pmatrix} \bar{c} \\ * \end{pmatrix},$$
$$v \text{ aus } \bar{R}v = \bar{c} \text{ durch Rückwärts-Substitution.}$$

Eine Besonderheit des Verfahrens ist, daß die Korrektur immer „ein Schritt in der richtigen Richtung“ ist, d.h. ein Schritt in einer Richtung, wo die zu minimierende Funktion $Z(x)$ kleiner wird:

$$Z'(x^i) \cdot (\lambda v) = b^T A \cdot (\lambda v) = -\lambda b^T A \cdot (A^T A)^{-1} A^T b < 0,$$

weil $A^T A$ und damit $(A^T A)^{-1}$ positiv-definit sind.

Das heißt aber keineswegs, daß bei obiger Iteration Z monoton abnimmt. Tatsächlich kann so ein Gauß-Newton-Schritt zwar in die richtige Richtung gehen, doch dabei weit über das Ziel hinausschießen und auf der gegenüberliegenden Talflanke höher hinaufklettern. Das wird in der Praxis häufig beobachtet und führt in kritischen Fällen zu einer Iteration mit extrem engem Einzugsgebiet.

So ein Verfahren wird robust und das Einzugsgebiet für die Praxis ausreichend groß, wenn man eine dynamische Begrenzung einbaut, also Teilschritt (c) ersetzt durch

`Schritt` (c'): $x^{i+1} := x^i + \lambda v$, $\lambda \in]0,1]$ `und` $\lambda \to 1$ `für` $i \to \infty$.

Noch viel besser bewährt hat sich eine raffiniertere Art des Bremsen, bei dem v nicht nur verkürzt wird, sondern gleichzeitig mehr in die Fallinie (Gradienten-Richtung $b^T A$) gedreht wird. Das geschieht, indem man nicht (c) modifiziert, sondern (b):

`Schritt` (b'): `Löse` $A^T b + (A^T A + D^2)v = 0$, $D = \mathrm{diag}(d_1, \ldots, d_n)$ `und`
$D \to 0$ `für` $i \to \infty$.

Diese **Levenberg-Marquardt-Technik** ist eine der wichtigsten Lösungsmethoden des nicht-linearen Ausgleichsproblems. Die richtige Wahl von D^2 (skaleninvariant und nicht zu groß, nicht zu klein) ist ziemlich technisch. In einer Labor-Umgebung hat das Ausgleichs-Problem meist die folgende Einkleidung:

Gegeben sind m Meßpunkte x_i, y_i, $i = 1, \ldots, m$, wobei die Einstell-Größen $x_i \in \mathbb{R}^l$ fehlerfrei sein sollen und die zugehörigen Meßwerte $y_i \in \mathbb{R}$ unkorrelierte, zufällige Fehler mit mittlerer Streuung $\delta y_i > 0$ haben sollen.

Diesen Meßpunkten gegenüber steht eine Theorie, die zum Ansatz

$$y = f(x, p)$$

Kapitel 7

Das symmetrische Eigenwertproblem

7.1 Der theoretische Hintergrund und einige Schlagworte

Unter dem algebraischen Eigenwertproblem versteht man die Aufgabe, Eigenwerte und Eigenvektoren einer quadratischen Matrix zu berechnen. Dabei gilt

Definition 7.1

$\lambda \in \mathbb{C}$ heißt **Eigenwert** von $A \in \mathbb{C}^{n,n}$ und $x \in \mathbb{C}^n$ ist ein **zugehöriger Eigenvektor** genau dann, wenn

$$Ax = \lambda x \qquad \text{und} \quad x \neq 0.$$

Die Eigenvektoren sind also die Richtungen, welche unter der Abbildung $x \mapsto Ax$ unverändert bleiben.

$(\lambda I - A)x = 0$ soll also nicht-triviale Lösungen haben, was bekanntlich genau dann der Fall ist, wenn die Matrix $\lambda I - A$ singulär ist, also

$$\lambda \text{ ist Eigenwert} \qquad \Longleftrightarrow \qquad \det(\lambda I - A) = 0.$$

Nach der allgemeinen Definition einer Determinante als Summe von Produkten zu je n Matrix-Elementen ist $\det(\lambda I - A)$ ein Polynom vom Grad n in λ, bekannt als das **charakteristische Polynom** der Matrix A:

$$\det(\lambda I - A) = \lambda^n + c_{n-1}\lambda^{n-1} + \ldots + c_0.$$

Seine Nullstellen sind also die Eigenwerte der Matrix und nach dem Fundamentalsatz der Algebra gibt es davon genau n, sofern man nur mehrfache Nullstellen entsprechend oft zählt. Dies ist die **algebraische Multiplizität** eines Eigenwertes. Die Gesamtheit aller Eigenwerte wird auch als das **Spektrum** bezeichnet. Wie

eine Darstellung als umso einfacher gilt, je mehr Elemente verschwinden in der zugehörigen Matrix. Zum Beispiel kann man mittels unitären Ähnlichkeits-Transformationen jede Matrix auf obere Dreiecksform bringen:

Satz 7.2 *Schursches Lemma*

Zu jeder Matrix $A \in \mathbb{C}^{n,n}$ und jeder gewünschten Reihenfolge $\lambda_1, \ldots, \lambda_n$ ihrer n Eigenwerte gibt es eine unitäre Matrix U so, daß

$$U^H A U = R = \begin{pmatrix} \lambda_1 & * & \ldots & * \\ & \ddots & & \vdots \\ & & \ddots & * \\ 0 & & & \lambda_n \end{pmatrix}.$$

Beweis: Wähle einen Eigenwert und nenne ihn λ_1. Löse das zugehörige singuläre, homogene Gleichungssystem

$$(A - \lambda_1 I)x^1 = 0$$

nach einem $x^1 \neq 0$ auf und normiere x^1 so, daß seine Länge 1 und seine erste Komponente ≥ 0 ist. In Satz 2.6 wurde eine Reflexionsmatrix T_1 explizit angegeben, so daß

$$T_1 = T_1^H = T_1^{-1} \qquad \text{und} \quad T_1 x^1 = e^1 = (1, 0, \ldots, 0)^T.$$

Dann hat die Matrix $T_1 A T_1^{-1}$ die Form

$$T_1 A T_1^{-1} = \begin{pmatrix} \lambda_1 & * & \ldots & * \\ 0 & & & \\ \vdots & & A_1 & \\ 0 & & & \end{pmatrix},$$

denn ihre erste Spalte ist $T_1 A T_1^{-1} e^1 = T_1 A x^1 = T_1 \lambda_1 x^1 = \lambda_1 e^1$. Die Matrix A_1 hat alle übrigen Eigenwerte von A, weil

$$\det(A - \lambda I) = \det(T(A - \lambda I)T^{-1}) = (\lambda_1 - \lambda)\det(A_1 - \lambda I).$$

Man kann jetzt diesen Prozeß mit A_1 fortsetzen, so wie in Abschnitt 2.4 an ähnlichen Beispielen geübt. Nach insgesamt $n-1$ Schritten sind alle Elemente unterhalb der Diagonale zu Null gemacht. Die Gesamt-Transformation ist

$$U = T_1 \begin{pmatrix} 1 & 0 \\ 0 & T_2 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & T_3 \end{pmatrix} \ldots \qquad (T_i = T_i^{-1}),$$

also ebenfalls unitär. ■

Für einen Eigenwert λ_i der algebraischen Multiplizität m sind in $R - \lambda_i I$ genau $n-m$ Diagonal-Elemente ungleich 0, so daß $\text{Rang}(R - \lambda_i I) \geq n - m$. Damit gibt es

- A ist mit A^H vertauschbar,
- A ist mittels unitärer Ähnlichkeits-Transformation diagonalisierbar,
- A besitzt ein vollständiges System von orthonormierten Eigenvektoren (geometrische Multiplizität = algebraischer Multiplizität)

Weiter zeigt das Korollar 7.3, daß die selbst-adjungierten Matrizen (insbesondere die reell-symmetrischen Matrizen) lauter reelle Eigenwerte haben und daß

$$A \text{ positiv-definit} \quad \Longleftrightarrow \quad x^H A x > 0 \ \forall x \neq 0 \quad \Longleftrightarrow \quad \text{alle Eigenwerte positiv.}$$

Analog für negativ-definit und semidefinit.

Alle Anwendungen von Eigenwerten/vektoren in den Natur- und Ingenieur-Wissenschaften haben ihre Wurzel und ihren Grund in der unitären Diagonalisierung (7.2) einer Matrix A. Es sind deshalb fast immer normale Matrizen, vor allem reell-symmetrische bzw. komplex selbst-adjungierte Matrizen, für die das Eigenwertproblem zu lösen ist.

Übungsaufgabe 7.2: Man zeige, daß eine reelle $n \times n$ Matrix durch eine reell-orthogonale Ähnlichkeits-Transformation auf obere Block-Dreiecksform gebracht werden kann mit reellen 1×1 und 2×2 Blöcken der Form

$$(\lambda_k) \quad \text{falls} \quad \lambda_k \in \mathbb{R}, \qquad \begin{pmatrix} \alpha_k & -\gamma_k \\ \beta_k & \alpha_k \end{pmatrix} \quad \text{falls} \quad \lambda_k = \alpha_k \pm i\sqrt{\beta_k \gamma_k}\,.$$

Die Reihenfolge dieser Blöcke ist frei wählbar (Kanonische Form von Wintner und Murnaghan, 1931).

7.2 Der Satz von Gershgorin und die Kondition einfacher Eigenwerte

Zu den theoretischen Grundlagen gehört auch der Satz von Gershgorin (1931). Zu seiner Formulierung wird jeder Zeile einer Matrix $A \in \mathbb{C}^{n,n}$ eine abgeschlossene Kreisscheibe in der komplexen Zahlenebene zugeordnet mit dem Diagonalelement als Mittelpunkt und der Betragssumme der übrigen Elemente dieser Zeile als Radius:

$$K_j := \{z \in \mathbb{C} : |z - a_{jj}| \leq \sum_{\substack{k=1 \\ k \neq j}}^{n} |a_{jk}|\}, \qquad j = 1, \ldots, n.$$

Dann gilt

Falls λ_j ein einfacher Eigenwert von A ist, dann ist der j-te Gershgorin-Kreis für hinreichend kleines ε disjunkt zu den übrigen. Das bleibt so, wenn man die j-te Zeile von $\Lambda + C$ mit einem geeigneten Vielfachen von ε multipliziert und die j-te Spalte entsprechend dividiert. Der j-te Radius schrumpft dabei auf $O(\varepsilon^2)$, während die übrigen anwachsen auf $O(1)$, aber noch disjunkt bleiben. Also ist

$$\begin{aligned} |\lambda_j(A+B) - \lambda_j - c_{jj}| &\leq O(\varepsilon^2), \\ \lambda_j(A+B) - \lambda_j &= e^{jT} X^{-1} B X e^j + O(\varepsilon^2), \end{aligned}$$

daher folgt aus $\|B\|_2 \leq \varepsilon$ dann die scharfe Ungleichung

$$|\lambda_j(A+B) - \lambda_j| \leq \varepsilon \|e^{jT} X^{-1}\|_2 \cdot \|X e^j\|_2 + O(\varepsilon^2).$$

Man benutzt deshalb $\kappa_j := \|e^{jT} X^{-1}\|_2 \cdot \|X e^j\|_2$ als die Konditionszahl für den einfachen Eigenwert λ_j einer diagonalisierbaren Matrix A. $\kappa_j \varepsilon$ ist die maximale Änderung des Eigenwertes λ_j gegenüber Störungen B mit $\|B\|_2 = \varepsilon \to 0$ (erste Ordnung der Störungstheorie).

Zu erläutern ist noch die geometrische Bedeutung von κ_j. Es ist nämlich $1/\kappa_j$ der Abstand des normierten Eigenvektors $Xe^j/\|Xe^j\|_2$ zu der Hyperebene, aufgespannt von den übrigen Eigenvektoren Xe^k, $k \neq j$. Dieser Abstand ist nämlich $\nu^T X e^j / \|X e^j\|_2$, wobei ν^T die normierte Normale auf dieser Hyperebene ist:

$$\|\nu^T\|_2 = 1 \qquad \text{und} \qquad \nu^T X e^k = 0 \quad \forall k \neq j$$

Das ist erfüllt nur von $\nu^T = e^j X^{-1} / \|e^j X^{-1}\|_2$.

Der Eigenwert λ_j ist also schlecht konditioniert, wenn der zugehörige Eigenvektor x^j nahezu linear abhängig ist von den übrigen Eigenvektoren x^k, $k \neq j$ (kleiner Abstand, großes κ_j). Die bestmögliche Situation ist ein System von gegenseitig orthogonalen Eigenvektoren:

> Alle normalen Matrizen haben bestmöglich konditionierte Eigenwerte.
> Die Konditionszahlen κ_j sind in diesem Fall alle 1.

Bei normalen Matrizen darf man auch nur *unitäre* Ähnlichkeits-Transformationen machen, sonst geht diese Eigenschaft und die optimale Kondition der Eigenwerte verloren. Erst recht gilt das für die selbst-adjungierten Matrizen, wo man natürlich die Symmetrie $\bar{a}_{ij} = a_{ji}$ unbedingt erhalten möchte.

Wie schon in Kapitel 1 diskutiert wurde, darf man auf keinen Fall die Eigenwerte berechnen, indem man die Koeffizienten des charakteristischen Polynoms irgendwie ermittelt und daraus dann irgendwie dessen Nullstellen (z.B. Newton- oder Laguerre-Verfahren). Diese Nullstellen können nämlich beliebig schlecht konditioniert sein, so daß kleinste Rundungsfehler in den Polynom-Koeffizienten intolerabel wären.

Alle übrigen Elemente bleiben unverändert. Somit ist klar, daß sich auch die Quadratsumme aller Elemente außerhalb der vier Kreuzungspunkte (p,p), (p,q), (q,p), (q,q) nicht ändert:

$$\sigma(A_i) - 2{a'_{pq}}^2 = \sigma(A_{i-1}) - 2a_{pq}^2.$$

Damit ist zu sehen, wie die drei Parameter p, q, φ der ebenen Rotation zu wählen sind, um bei σ eine möglichst große Abnahme zu erzielen:

$$\begin{array}{ll} \varphi \text{ so, daß} & a'_{pq} = 0 \\ p \text{ und } q \text{ so, daß} & |a_{pq}| \text{ möglichst groß.} \end{array}$$

Das verringert σ um $2a_{pq}^2$ und keine andere Wahl würde mehr erbringen. Nachzurechnen ist:

$$\begin{array}{lll} a'_{pq} = 0 & \Rightarrow & (a_{pp} - a_{qq})sc + a_{pq} \cdot (c^2 - s^2) = 0. \\ t = s/c & \Rightarrow & t^2 + 2\theta t - 1 = 0 \text{ mit } \theta := (a_{qq} - a_{pp})/2a_{pq} \\ \Rightarrow & t = -\theta \pm \sqrt{\theta^2 + 1} = 1/(\theta \pm \sqrt{\theta^2 + 1}), & + \text{ falls } \theta \geq 0, \quad - \text{ falls } \theta < 0. \end{array}$$

Die beiden Lösungen für $t = s/c$ unterscheiden sich im zugehörigen Drehwinkel φ nur um $90°$, was unwesentlich ist. Die angegebene Vorzeichenregel gibt ein genaues $t \in [-1,+1]$, $\varphi \in [-45°, 45°]$. Dazu gehört und ebenfalls nachzurechnen ist

$$\begin{array}{ll} c := 1/\sqrt{1+t^2}, & s := t/\sqrt{1+t^2}, \\ a'_{pp} := a_{pp} - ta_{pq}, & a'_{pq} := 0, \\ a'_{qp} := 0, & a'_{qq} := a_{qq} + ta_{pq}. \end{array}$$

Klassisches Jacobi-Verfahren

für $i = 1, 2, \ldots$
    Schritt (a): suche betragsgrößtes Außerdiagonal-Element a_{pq}
    Schritt (b): falls $|a_{pq}| \leq \varepsilon$: STOP
    Schritt (c): berechne θ, t, c, s, a'_{pp}, a'_{qq}, $a'_{pq} := 0$
    Schritt (d): berechne die neuen Elemente in Zeile p, q und
                 Spalte p, q
                 (nur auf einer Seite der Diagonalen)

Nach dem Abbrechen sind alle Außerdiagonal-Elemente zwischen $\pm\varepsilon$ und der Satz von Gershgorin beweist, daß die Diagonal-Elemente von den Eigenwerten höchstens $(n-1)\varepsilon$ abweichen.

Die Konvergenz des Verfahrens ist leicht einzusehen, denn jeder Iterations-Schritt verkleinert σ um mindestens den Faktor $1-2/n(n-1)$. Tatsächlich ist das Verfahren quadratisch konvergent, aber das beweist sich nicht ganz so einfach.

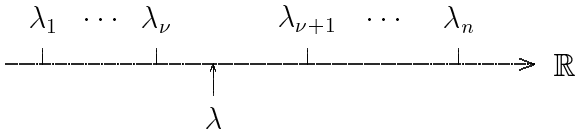
Die Suche nach dem jeweils betragsgrößten Außerdiagonal-Element wird bei sehr großen Matrizen zu teuer im Vergleich zum sonstigen Aufwand. Deshalb hat sich

weniger als n homogene Bedingungen sind und es somit immer eine Lösung $x \neq 0$ gibt. Für dieses x wäre $x^T Ax > 0$ und $(Sx)^T B(Sx) \leq 0$. ■

Damit haben wir ein sehr preiswertes Mittel an der Hand, um die Anzahl ν der Eigenwerte $\lambda_1 \leq \ldots \leq \lambda_n$ von T zu bestimmen, die kleiner sind als ein beliebiges $\lambda \in \mathbb{R}$:

Eigenwerte von T:

beliebiger Bezugspunkt:



Zu dem Zweck brauchen wir nämlich nur die sehr billige Gauß-Elimination von $T - \lambda I$ zu machen und die negativen Pivots zählen:

$$\begin{pmatrix} a_1-\lambda & b_2 & & \\ b_2 & a_2-\lambda & b_3 & \\ & b_3 & a_3-\lambda & \ddots \\ & & \ddots & \ddots \end{pmatrix} \longrightarrow \begin{pmatrix} d_1 & b_2 & & \\ 0 & d_2 & b_3 & \\ & 0 & d_3 & \ddots \\ & & \ddots & \ddots \end{pmatrix}$$

Gauß-Elimination an $T - \lambda I$:

$d_1 := a_1 - \lambda$;
`für` $i := 2$ `bis` n:
    $e_i := b_i / d_{i-1}$;
    $d_i := a_i - \lambda - e_i b_i$.

Anzahl $\nu(\lambda)$ der negativen Pivots:

$\nu := 0$;
$d_1 := a_1 - \lambda$;
`falls` $d_1 < 0$: $\nu := 1$;
`für` $i := 2$ `bis` n:
    $d_i := a_i - b_i^2 / d_{i-1} - \lambda$;
    `falls` $d_i < 0$: $\nu := \nu + 1$.

Weil $T - \lambda I$ indefinit ist (außer für $\lambda < \lambda_1$ oder $\lambda > \lambda_n$), können einige d_i verschwinden. In diesem Fall macht man einen „mutwilligen Rundungsfehler“ und ersetzt ein solches d_i durch ein geeignetes ε. Das entspricht einer Modifikation $a_i \mapsto a_i + \varepsilon$ und der Einfluß auf die Eigenwerte kann beliebig klein gemacht werden.

Wie wirken sich Rundungsfehler auf diesen Prozeß aus (einschließlich des oben erwähnten „mutwilligen“)? Offensichtlich erhalten wir beim Rechnen in endlicher Genauigkeit verfälschte Werte $\tilde{d}_i$ anstelle der exakten Werte d_i. Dabei können sogar falsche Vorzeichen auftreten, die dann ein falsches ν und eventuell eine falsche Entscheidung bei der Bisektion bewirken: es kann also eine Intervall-Hälfte ausgeschieden werden, die in Wahrheit den gesuchten Eigenwert λ_k enthält. Höchst bedenklich ist dabei, daß ein solche Fehlentscheidung nie mehr korrigiert werden kann und daß sie *in der ersten Bisektion genauso auftreten kann wie in der letzten!* Die Situation ist so bedrohlich, daß eine genaue Analyse unumgänglich ist. (Die zweite der in Kapitel 1 angekündigten Fallstudien.)

Nach der Rückwärts-Technik von Kapitel 1 können wir sagen

$$\begin{aligned}\tilde{d}_1 &= (a_1 - \lambda)\cdot(1+\sigma_1'),\\ \tilde{d}_i &= \Big(\big(a_i - b_i^2\cdot(1+\mu_i)/\tilde{d}_{i-1}\cdot(1+\delta_i)\big)(1+\sigma_i) - \lambda\Big)\cdot(1+\sigma_i'),\end{aligned}$$

wobei die Herkunft der μ_i, δ_i, σ_i, σ_i' klar ist und alle diese Größen betragsmäßig beschränkt sind durch die Maschinen-Genauigkeit $\varepsilon_{\text{mach}}$.

Wenn wir die Hilfsgrößen $\hat{d}_i := \tilde{d}_i/(1+\sigma_i')$, $i = 1,\ldots,n$ definieren, dann gilt:

$$\begin{aligned}\hat{d}_1 &= a_1 - \lambda,\\ \hat{d}_i &= a_i\cdot(1+\sigma_i) - b_i^2\cdot(1+\mu_i)\cdot(1+\delta_i)\cdot(1+\sigma_i)/((1+\sigma_{i-1}')\hat{d}_{i-1}) - \lambda\\ &= \hat{a}_i - \hat{b}_i^2/\hat{d}_{i-1} - \lambda,\end{aligned}$$

wobei $\hat{a}_1 := a_1$ und $\hat{a}_i = a_i\cdot(1+\sigma_i)$, $\hat{b}_i = b_i\cdot((1+\mu_i)(1+\delta_i)(1+\sigma_i)/(1+\sigma_{i-1}'))^{1/2}$ für $i = 2,\ldots,n$.

Wir halten fest:

a) Die Folge $\{\hat{d}_i\}$ ist exakt zur Matrix $\hat{T}$ mit den Elementen $\hat{a}_i$, $\hat{b}_i$ statt a_i, b_i.

b) Die Folge $\{\hat{d}_i\}$ hat die gleichen Vorzeichen wie die Folge der berechneten Werte $\tilde{d}_i$.

Somit kann das berechnete $\tilde{\nu}$ zwar einen *völlig falschen Wert* haben, jedoch ist es immer *korrekt für eine Nachbar-Matrix* $\hat{T}$ anstelle von T mit $|\hat{T} - T| \overset{\bullet}{\leq} 2\varepsilon_{\text{mach}}|T|$. Die Bisektion ist somit korrekt für $\hat{T}$ anstelle von T, wobei allerdings $\hat{T}$ bei jeder Bisektion ein bißchen anders sein wird.

Betrachten wir den k-ten Eigenwert aller möglichen Matrizen $\hat{T}$ und seine Variationsbreite:

$$\Delta_k := \max_{\hat{T}}\{|\lambda_k(\hat{T}) - \lambda_k(T)| : |\hat{T} - T| \leq 2\varepsilon_{\text{mach}}|T|\}.$$

Wie wir wissen, ist das im wesentlichen die Kondition von λ_k, d.h. die natürliche Unsicherheit des k-ten Eigenwertes auf Grund von Daten-Unsicherheit. Wir studieren nun zwei Fälle für die Position der Bisektionsstelle λ:

($b_1 = 0, b_2, \ldots, b_n \neq 0$). Falls λ exakt ist, muß die n-te Gleichung von selbst erfüllt sein, weil dann die letzte Zeile linear abhängig von den übrigen ist. Andernfalls gilt

$$(T - \lambda I)x = (0, \ldots, 0, \Delta(\lambda))^T \tag{7.3}$$

und aus der Cramerschen Regel folgt $x_1 = (-1)^{n-1}\Delta(\lambda)b_2 \ldots b_n / \det(T - \lambda I)$, so daß

$$\Delta(\lambda) = (-1)^{n-1}(\lambda_1 - \lambda) \cdots (\lambda_n - \lambda)/(b_2 \cdots b_n).$$

Beispiel:

$$T = \begin{pmatrix} 6 & 1 & 0 & 0 & 0 \\ 1 & 3 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & -3 & 1 \\ 0 & 0 & 0 & 1 & -6 \end{pmatrix}, \qquad \lambda_5 = 6.316\,876\ldots$$

λ	6.31	6.32	6.316 876 ...
x_1	1	1	1
x_2	0.31	0.32	0.316 876 ...
x_3	0.02 61	0.06 24	0.051 038 ...
x_4	−0.14 53 09	0.07 43 68	0.005 526 ...
x_5	−1.37 89 26 79	0.63 07 09 76	0.000 449 ...
$\Delta(\lambda)$	16.82 92 79 78 49	−7.69 59 76 24 32	0 (exakt!)
$\Delta(\lambda)/x_5$	−12.20 ...	−12.20 ...	

Bei einer Rechengenauigkeit mit drei Dezimalen sind 6.31 und 6.32 die beiden besten Näherungen für den Eigenwert λ_5 und doch kann man damit den zugehörigen Eigenvektor *so nicht berechenen*.

Achtung: Obige Zahlenwerte sind exakt und das Versagen der Methode beruht somit nicht auf Rundungsfehlern, sondern auf der extrem schlechten Kondition von $\Delta(\lambda)$.

Das Paar λ, x ist exakt für eine Tridiagonal-Matrix $\tilde{T}$ mit $\tilde{a}_n := a_n - \Delta(\lambda)/x_n$ statt a_n. Nur wenn $|\Delta(\lambda)/x_n| = O(\varepsilon_{\text{mach}}\|T\|)$, ist das so berechnete x akzeptabel. Nur in solchen Fällen wäre die simple Methode von Hyman brauchbar.

Zuverlässiger und nicht viel teurer ist die **inverse Iteration** oder **Wielandt-Iteration**. Das ist die übliche **Vektor-Iteration** von Frank und Mises, jedoch nicht angewendet auf die Matrix T mit dem Eigenwerten $\lambda_1, \ldots, \lambda_n$ sondern auf $(T - \lambda I)^{-1}$ mit den Eigenwerten $(\lambda_1 - \lambda)^{-1}, \ldots, (\lambda_n - \lambda)^{-1}$, und den gleichen zugehörigen Eigenvektoren:

$$\begin{aligned} &\text{Wähle } x^0 \in \mathbb{R}^n, \\ &x^i \text{ aus } (T - \lambda I)x^i = x^{i-1}, \qquad i = 1, 2, \ldots \end{aligned} \tag{7.4}$$

Kapitel 8

Numerische Quadratur

Unter Quadratur versteht man die Berechnung eines bestimmten Integrals,

$$J(f) := \int_a^b f(x)\,dx.$$

Wenn möglich, dann ist die geschlossene Integration stets vorzuziehen. Die dabei notwendigen Zerlegungen, Variablen-Substitutionen, partiellen Integrationen usw. sollte man als Teilschritte für sich programmieren und niemals selbst zu einer Riesenformel zusammenfassen, die man dem Computer zur Auswertung übergibt.

Wenn die geschlossene Integration nicht möglich ist, dann sollte man gute Software aus einer Programm-Bibliothek verwenden. Diese arbeitet in der Regel adaptiv, d.h. mit dynamischer Intervall-Zerlegung und Näherungsformeln für die Teil-Intervalle. Hier werden nur letztere besprochen.

Die numerischen Verfahren verlangen Integranden f, die hinreichend oft differenzierbar (glatt) sind. Deshalb ist eine Aufbereitung nötig, wenn diese Bedingung nicht erfüllt ist. Zwei Beispiele:

Substitution $\quad x = t^2$ in $\int_0^\pi \frac{\sin x}{\sqrt{x}}\,dx$

Zerlegung $\quad \int_a^b f = \int_a^c f + \int_c^b f, \qquad$ wenn f oder $D^k f$ unstetig bei $x = c$.

In einer Näherungsformel

$$\int_a^b f(x)\,dx \approx \sum_{i=1}^n g_i f(x_i) =: \tilde{J}(f)$$

sind $g_1, \ldots, g_n$ die **Gewichte** und $x_1 < \ldots < x_n$ die **Stützstellen**. Die Differenz der beiden Seiten ist das **Restglied** (Fehler):

$$R(f) = \sum_{i=1}^n g_i f(x_i) - \int_a^b f(x)\,dx.$$

Hier ist $\tilde{f}$ der Polynom-Interpolant vom Grad 1 mit den beiden Endpunkten als Stützstellen.

$$R(f) = \frac{1}{2}\int_0^h x(h-x)f''(x)\,dx = +h^3 f''(\xi)/12,$$

also exakt für $\mathbb{P}_1$.

Faßregel

(J. Kepler: *Nova stereometrica doliorum vinariorum*[1], Linz 1615)

$$\tilde{J}(f) = \frac{h}{6}(f(0) + 4f(h/2) + f(h)).$$

Hier ist $\tilde{f}$ der Polynom-Interpolant vom Grad 2 zu den Stützstellen 0, $h/2$, h.

$$R(f) = \int_0^h r(x)f''''(x)\,dx = h^5 f''''(\xi)/2880,$$

mit $r(x) = r(h-x) = hx^3/36 - x^4/24$ für $0 \le x \le h/2$, also exakt für $\mathbb{P}_3$.

Eine andere Herleitung betrachtet die Faßregel als $2:1$ Mischung von Rechtecks- und Trapezregel, um die Fehler in erster Ordnung auszumitteln $(-1:2)$. Eine dritte Herleitung betrachtet die Faßregel als $-1:4$ Mischung der Trapezregel zur Schrittweite h und $h/2$ mit Fehlerverhältnis $h^3 : 2(h/2)^3 = 4:1$.

Newton-Cotes-Formeln

$\tilde{f}$ ist der Polynom-Interpolant zu $n+1$ äquidistanten Stützstellen $x_k = kh/n$ für $k = 0, \ldots, n$. Für $n = 8$ und ab $n = 10$ negative Gewichte und wenig empfehlenswert.

Clenshaw-Curtis (1960)

Wie zuvor, jedoch nicht-äquidistante Stützstellen bei $x_k = (1 - \cos k\pi/n)h/2$ für $k = 0, \ldots, n$. Eine sehr gute Methode mit immer positiven Gewichten. Bei einem Übergang von n nach $2n$ kann man die bisherigen Funktionswerte wieder verwenden. Die Durchführung erfolgt über die diskrete Cosinus-Transformation. Für den Unterschied der Newton-Cotes- und Clenshaw-Curtis-Quadratur vergleiche die beiden Figuren in Abschnitt 4.5.

[1] Neue Volumensformel für Weinfässer.

$$x_i := a + ih \qquad \text{und} \qquad f_i := f(x_i) \text{ für } i = 0, \ldots, n,$$
$$\sum{}'' f_i \quad := \quad \frac{1}{2} f_0 + f_1 + \ldots + f_{n-1} + \frac{1}{2} f_n.$$

Bernoullische Zahlen und Polynome

Technisches Hilfsmittel für das Folgende sind die rekursiv definierten Polynome

$$\begin{aligned} b_0(t) &:= 1 \\ b_k'(t) &:= b_{k-1}(t), \qquad \text{für } k = 1, 2, 3, \ldots \qquad (8.1) \\ \int_0^1 b_k(t)\, dt &= 0, \qquad \text{für } k = 0, 1, 2, \ldots \qquad (8.2) \end{aligned}$$

(8.2) legt die in (8.1) noch offene Integrations-Konstante fest.

Übungsaufgabe 8.1:

$$b_1(t) = t - 1/2, \quad b_2(t) = t^2/2 - t/2 + 1/12, \quad b_3(t) = t^3/6 - t^2/4 + t/12.$$

Mittels Induktion sieht man sofort, daß $b_k(t)$ ein Polynom vom Grad genau k ist (mit $k!$ multipliziert heißt es das **k-te Bernoulli-Polynom**). Man sieht auch fast genauso leicht, daß die $b_k(t)$ für gerades k symmetrisch und für ungerades k schief-symmetrisch bezüglich $t = 1/2$ sind:

$$b_k(1 - t) = (-1)^k b_k(t), \quad k = 0, 1, 2, \ldots$$

In der Tat: $b_0(t)$ *ist* symmetrisch und wenn $b_{2k}(t)$ symmetrisch ist, dann erfüllt

$$b_{2k+1}(t) = \int_{0.5}^{t} b_{2k}(s)\, ds$$

(8.1) und (8.2) und *ist* schief-symmetrisch. Wenn $b_{2k+1}(t)$ schief-symmetrisch ist, dann *ist*

$$b_{2k+2}(t) = \int_{0.5}^{t} b_{2k+1}(s) ds + C$$

symmetrisch für *jedes* C. Es sei noch festgehalten, daß

$$b_k(0) = b_k(1) \qquad \text{für} \quad k = 0 \text{ und } 2, 3, 4, \ldots \qquad (8.3)$$

denn nach (8.1) und (8.2) ist ab $k = 2$

$$b_k(1) - b_k(0) = \int_0^1 b_k'(t)\, dt = \int_0^1 b_{k-1}(t)\, dt = 0.$$

Wegen der Schief-Symmetrie folgt insbesondere

$$b_k(0) = b_k(1) = 0 \qquad \text{für } k = 3, 5, 7, \ldots$$

h^2-Entwicklung der Trapezsumme

Für die Numerik sind die Formeln (8.5) und (8.6) sehr wichtig, weil sie die h-Abhängigkeit der Trapezsumme $T(h)$ sehr präzise wiedergeben. Für

$$T(h) := h{\sum}'' f_i$$

läßt sich nach (8.5), (8.6) schreiben

$$T(h) = \tau_0 + \tau_1 h^2 + \ldots + \tau_{m-1} h^{2m-2} + \tau_m(h) \cdot h^{2m}, \tag{8.7}$$

wobei

$$\begin{aligned} \tau_0 &:= \int_a^b f(x)\,dx, && (8.8)\\ \tau_k &:= b_{2k}(0) \cdot \left(D^{2k-1}f(b) - D^{2k-1}f(a)\right), \quad k = 1, \ldots, m-1, && (8.9)\\ \tau_m(h) &:= \int_a^b (b_{2m}(0) - b^*_{2m}(x)) \cdot D^{2m} f(x)\,dx. \end{aligned}$$

Letzteres hängt wie angezeigt noch von h ab, weil ja in obiger Definition von $b^*_{2m}(x)$ der Parameter h vorkommt. Es gibt aber eine *von h unabhängige Schranke*:

$$|\tau_m(h)| \le \max_{0 \le t \le 1} |b_{2m}(0) - b_{2m}(t)| \cdot \int_a^b |D^{2m} f(x)|\,dx. \tag{8.10}$$

(8.7) und (8.10) zusammen sind ein wichtiges Resultat: Die sogenannte **asymptotische Reihe** für die Trapezsumme. Eine asymptotische Reihe unterscheidet sich von einer gewöhnlichen Reihe (hier Potenzen von h^2) dadurch, daß bei letzterer das Restglied beliebig klein gemacht werden kann durch die Wahl eines hinreichend großen m, also $\tau_m(h) \cdot h^{2m} \to 0$ für $m \to \infty$. Bei asymptotischen Reihen dagegen braucht das Restglied für $m \to \infty$ weder zu existieren noch gegen 0 zu streben: Der Grenzübergang von der endlichen Summe zur unendlichen Reihe ist bei asymptotischen Entwicklungen im allgemeinen nicht möglich. Beispielsweise braucht im vorliegenden Fall f nicht unbegrenzt oft differenzierbar zu sein. In der Numerik wird die Formel (8.7) stets für *festes* m angewendet und für $h = (b-a)/n$ *variabel*, insbesondere für $h \to 0$. Man sollte übrigens nicht übersehen, daß die eigentliche Aussage *nicht* in (8.7) steckt, sondern vielmehr in (8.10). In der Tat kann man (8.7) als Definition von $\tau_m(h)$ ansehen:

$$\tau_m(h) = \left(T(h) - \tau_0 - \tau_1 h^2 - \ldots - \tau_{m-1} h^{2m-2}\right) / h^{2m}, \quad \forall h \neq 0.$$

Dabei sind die Koeffizienten $\tau_0, \ldots, \tau_{m-1}$ noch ganz beliebig. Allerdings ist das so definierte $\tau_m(h)$ in h beschränkt eben nur für die Wahl (8.8), (8.9).

Übungsaufgabe 8.2: Man benütze die bekannte Fourier-Reihe

$$t - \frac{1}{2} = -\sum_{n=1}^{\infty} \frac{\sin(2\pi n t)}{\pi n} \quad \text{für } 0 < t < 1$$

Die linke Seite können wir berechnen. Der erste Term rechts, das exakte Integral, ist gesucht.

$\tau_1 h^2$ wird für genügend kleine Schrittweiten h der dominante Term im Restglied. Man kann diesen Term eliminieren, indem man die Trapezsumme für zwei verschiedene h ausrechnet und beide Werte kombiniert:

$$\begin{aligned} T(h_1) &= \tau_0 + \tau_1 h_1^2 + \tau_2 h_1^4 + \ldots \quad \Big| \cdot -h_2^2/(h_1^2 - h_2^2). \\ T(h_2) &= \tau_0 + \tau_1 h_2^2 + \tau_2 h_2^4 + \ldots \quad \Big| \cdot h_1^2/(h_1^2 - h_2^2). \\ \frac{h_1^2 T(h_2) - h_2^2 T(h_1)}{h_1^2 - h_2^2} &= \tau_0 + 0 - \tau_2 \cdot h_1^2 h_2^2 + \ldots \end{aligned}$$

Analog kann man aus drei Trapezsummen $\tau_1 h^2$ und $\tau_2 h^4$ eliminieren und erhält für die Kombination

$$\tau_0 + 0 + 0 + \hat{\tau}_3 h_1^2 h_2^2 h_3^2 + \ldots$$

Aus m Trapezsummen kann man durch geschicktes Kombinieren alle Terme bis auf den letzten eliminieren:

$$\begin{aligned} T(h_1) &= \tau_0 + \tau_1 h_1^2 + \ldots + \tau_m(h_1) h_1^{2m} \\ &\vdots \\ T(h_m) &= \tau_0 + \tau_1 h_m^2 + \ldots + \tau_m(h_m) h_m^{2m} \\ \text{Kombination} &= \tau_0 + 0 + \ldots + 0 + \hat{\tau}_m(h_1, \ldots, h_m) \cdot h_1^2 \cdots h_m^2 \end{aligned}$$

Diese Idee stammt von Richardson. Ihre Ausführung ist nach dem Gauß-Algorithmus möglich. Eine bequemere Durchführung stammt von Romberg.

Wir führen das Polynom P vom Grad $m-1$ in h^2 ein, das die für $h = h_1, \ldots, h_m$ berechneten Trapezsummen interpoliert, also

$$T(h) = P(h^2) \text{ an den Stellen } h = h_1, \ldots, h_m.$$

Mit dem Ansatz $P(h^2) = \gamma_0 + \ldots + \gamma_{m-1} h^{2m-2}$ gilt also

$$\tau_0 + \tau_1 h_k^2 + \ldots + \tau_m(h_k) h_k^{2m} = \gamma_0 + \ldots + \gamma_{m-1} h_k^{2m-2}, \qquad k = 1, \ldots, m$$

Richardsons Vorschlag auf *beide* Seiten angewendet, eliminiert auf *beiden* Seiten alle geraden Potenzen von h, so daß übrig bleibt

$$\tau_0 + 0 + \ldots + \hat{\tau}_m(h_1, \ldots, h_m) \cdot h_1^2 \cdots h_m^2 = \gamma_0 + 0 + \ldots + 0.$$

Die linke Seite ist die Kombination, die Richardson als verbesserte Näherung für das Integral einführte. Die rechte Seite ist $P(0)$. Mithin haben wir folgendes Rezept:

Romberg-Quadratur:
für $k := 1$ bis m:
    wähle n_k;
    $h_k := (b-a)/n_k$;
    $T_k :=$ Trapezsumme zur Schrittweite h_k;
    für $i := k-1$ abwärts bis 1:
        $T_i := T_{i+1} + (T_{i+1} - T_i)/(h_i^2/h_k^2 - 1)$;
$P(0) := T_1$.

Die Speicher-Organisation im dreieckigen Schema ist hier

$$\begin{array}{ccccc} h_1^2 & h_2^2 & h_3^2 & \bullet \\ T_1 & T_2 & T_3 & \bullet \\ & T_1 & T_2 & \bullet \\ & & T_1 & \bullet \\ & & & \bullet \end{array}$$

Durch Punkte angedeutet ist die kommende Schrägzeile.

Ursprünglich benutzte man die Folge $n_k = 2^k$ und die Rekursion

$$T(h_k) = \frac{1}{2}T(h_{k-1}) + h_k(f(a+h_k) + f(a+3h_k) + \ldots + f(b-h_k)).$$

Heute bevorzugt man die nicht so rasch anwachsende Folge von R. Bulirsch

$$n_k = 2, 3, \;\; 4, 6, \;\; 8, 12, \;\; \ldots$$

wobei es eine Rekursion für die $T(h_k)$ gibt, die ebenfalls jeden benötigten Funktionswert nur einmal berechnet (Programmier-Aufgabe).

8.4 Quadratur nach Gauß-Legendre

Ausgangspunkt ist die Standard-Diskretisierungsformel

$$\int_{-1}^{+1} f(x)\,dx \approx \sum_{i=1}^{n} g_i f(x_i).$$

Die Wahl von $[-1, +1]$ als Integrations-Intervalle ist keine Beschränkung der Allgemeinheit und sehr gebräuchlich. Die Methode ist definiert durch die Forderung:

Stützstellen $x_1, \ldots, x_n$ und Gewichte $g_1, \ldots, g_n$ so, daß

$$\int_{-1}^{+1} p(x)\,dx = \sum_{i=1}^{n} g_i p(x_i) \tag{8.11}$$

für alle Polynome $p(x)$ **möglichst hohen Grades**!

Etwas anderes wäre sehr enttäuschend und äußerst bedenklich gewesen. Eine Konseqenz ist, daß die zu (8.15) gehörenden Lagrange-Polynome wohldefiniert sind:

$$L_j(x) = \prod_{\substack{i=1 \\ i \neq j}}^{n} \frac{x - x_i}{x_j - x_i}, \qquad \text{für } j = 1, \ldots, n.$$

Notwendige Bedingung für die Gewichte $g_1, \ldots, g_n$: (8.12) muß insbesondere erfüllt sein für alle L_j:

$$\int_{-1}^{+1} L_j(x)\,dx = \sum_{i=1}^{n} g_i L_j(x_i) = g_j, \qquad j = 1, \ldots, n, \tag{8.16}$$

denn $L_j \in \mathbb{P}_{n-1} \subset \mathbb{P}_{2n-1}$. Damit ist auch die Wahl der Gewichte auf einen einzigen möglichen Satz eingeengt.

Hinreichende Bedingungen: Es wird jetzt gezeigt, daß die einzig mögliche Lösung, nämlich (8.14) und (8.16), tatsächlich (8.12) erfüllt:

Betrachte ein beliebiges $p \in \mathbb{P}_{2n-1}$ und seinen Interpolanten in $\mathbb{P}_{n-1}$ (Lagrangesche Interpolationsformel), so daß

$$p(x) - \sum_{j=1}^{n} p(x_j) L_j(x) = q(x) \cdot (x - x_1) \cdots (x - x_n).$$

Die linke Seite verschwindet für $x \in \{x_1, \ldots, x_n\}$ und das begründet die Produktform auf der rechten Seite mit einem geeigneten $q \in \mathbb{P}_{n-1}$. Integration ergibt

$$\begin{aligned} \int_{-1}^{+1} p(x)\,dx &= \sum_{j=1}^{n} p(x_j) \int_{-1}^{+1} L_j(x)\,dx + \int_{-1}^{+1} q(x) \cdot (x - x_1) \cdots (x - x_n)\,dx \\ &= \sum_{j=1}^{n} p(x_j) g_j + 0 \qquad \text{nach (8.16) und (8.13).} \end{aligned}$$

Also ist (8.12) erfüllt für jedes beliebige $p \in \mathbb{P}_{2n-1}$.

Schließlich ist noch erwähnenswert, daß $L_j(x)^2 \in \mathbb{P}_{2n-2}$ in (8.12) zugelassen ist:

$$\int_{-1}^{+1} L_j(x)^2\,dx = g_j \cdot 1^2 \quad \forall j$$

und das zeigt, daß jedes Gewicht g_j positiv ist. Etwas anderes wäre sehr enttäuschend und bedenklich gewesen. Der Gaußsche Ansatz führt auf eine „schöne“ Lösung!

Wir leiten zum Schluß noch schnell eine Fehlerformel her (Restglied). Sei $f \in \mathbb{C}^{2n}[-1, +1]$ und $P \in \mathbb{P}_{2n-1}$ sein Interpolant zu den *doppelten* Stützstellen

Kapitel 9

Anfangswertprobleme bei gewöhnlichen Differentialgleichungen

Sei $f\colon [a,b] \times \mathbb{R}^m \to \mathbb{R}^m$ genügend glatt und $y\colon [a,b] \to \mathbb{R}^m$ die entsprechend glatte Lösung des **Anfangswertproblemes** (AWP)

$$y'(x) = f(x, y(x)) \quad \forall x, \tag{9.1}$$
$$y(x_0) = y_0 \qquad (\text{festes } x_0 \in [a,b]). \tag{9.2}$$

(9.1) heißt **Differentialgleichung** (DGL) und f ist ihre **rechte Seite**, (9.2) heißt **Anfangsbedingung** (AB). DGL und AB lassen sich durch eine formale Integration zusammenfassen in der äquivalenten Gleichung

$$y(x) = y_0 + \int_{x_0}^{x} f(t, y(t))\, dt. \tag{9.3}$$

Aus der Vorlesung *Analysis III* ist bekannt:

a) Der Satz von Picard und Lindelöf: Die **Lipschitz-Bedingung**

$$\|f(x,y) - f(x,z)\| \le L \cdot \|y - z\| \tag{9.4}$$

garantiert Existenz und Eindeutigkeit der Lösung. (Die **Lipschitz-Konstante** L dürfte sogar von x abhängen.)

b) DGL'en höherer Ordnung lassen sich durch Einführung von Hilfsvariablen auf DGL'en erster Ordnung zurückführen.

c) Durch $y_{m+1}(x) := x$ und $f_{m+1}(x,y) := 1$ läßt sich das x-Argument in f formal entfernen: die DGL **autonom** schreiben.

d) Viele Aufgaben aus den Natur- und Ingenieurwissenschaften führen auf Anfangswertprobleme.

9.1 Diskretisierung als Grundlage der numerischen Verfahren

In Ausnahmefällen mag der Reihenansatz

$$y(x) = y_0 + y_0' \cdot (x - x_0) + y_0'' \cdot (x - x_0)^2/2 + \dots$$

eine brauchbare Näherung liefern. Koeffizienten-Vergleich oder Nachdifferenzieren liefert zum Beispiel

$$\begin{aligned} y_0' &= f(x_0, y_0), \\ y_0'' &= f_x(x_0, y_0) + f_y(x_0, y_0) \cdot y_0', \quad (f_y \in \mathbb{R}^{m,m}) \\ &\text{usw.} \end{aligned}$$

Die universell einsetzbaren numerischen Rechenverfahren benutzen dagegen ausschließlich die **Diskretisierung**, d.h. es wird das Kontinuum $[a, b]$ für die unabhängige Variable ersetzt durch ein **Gitter**

$$x_0 < x_1 < x_2 < \dots$$

mit **Schrittweiten** $h_k := x_k - x_{k-1}$. Dann ersetzt man entweder den Differentialquotienten in (9.1) oder das Integral in (9.3) durch eine Näherung, die sich auf die Gitterpunkte stützt. Im Folgenden werden die so produzierten Näherungswerte für die wahre Lösung an den Gitterpunkten mit $\{u_k\}$ bezeichnet:

$$\begin{aligned} u_k &\approx y(x_k), \\ u_k' \equiv f_k &:= f(x_k, u_k) \approx y'(x_k). \end{aligned}$$

Je nach Kontext ist die Benennung u_k' oder f_k die suggestivere!

Die folgenden naheliegenden Beispiele dienen nur der Einführung von Namen für Verfahren. Die angegebene Klassifikation wird weiter unten erklärt. h ist die hier konstante Schrittweite.

- **Euler-Cauchy-Verfahren** (E1):

$$u_k := u_{k-1} + h f_{k-1}.$$

- **Inverses Euler-Verfahren** (I1):

$$u_k := u_{k-1} + h f_k.$$

- **Trapez-Regel** (I1):

$$u_k := u_{k-1} + \frac{h}{2}(f_{k-1} + f_k).$$

Nur bei $s = 2$ spricht ein gewichtiges Symmetrie-Argument für diese Modifikation.

Es wird nachdrücklich angeraten, diese Verfahren für die Fälle $s = 1$, $s = 2$ und konstante Schrittweite h in konkrete Formeln umzusetzen. Dabei verwende $k = s$ und

$$\begin{array}{lrcl} s=1: & L_0(x) & = & (h-x)/h, \qquad L_1(x) = x/h, \\ s=2: & L_0(x) & = & x(x-h)/2h^2, \qquad L_1(x) = (h^2-x^2)/h^2, \\ & L_2(x) & = & x(x+h)/2h^2. \end{array}$$

usw.

- Die **Runge-Kutta-Verfahren** (E1):

  Die klassische Form ist

$$\begin{array}{rcl} \Delta_1 & := & hf(x_{k-1}, u_{k-1}), \\ \Delta_2 & := & hf(x_{k-1} + h/2, u_{k-1} + \Delta_1/2), \\ \Delta_3 & := & hf(x_{k-1} + h/2, u_{k-1} + \Delta_2/2), \\ \Delta_4 & := & hf(x_{k-1} + h, u_{k-1} + \Delta_3), \\ u_k & := & u_{k-1} + (\Delta_1 + 2\Delta_2 + 2\Delta_3 + \Delta_4)/6. \end{array}$$

  (Bei DGL-Systemen sind Δ_1, Δ_2, Δ_3, Δ_4 Vektoren!)

  Allgemein berechnet man bei einem **s-stufigen** Runge-Kutta-Verfahren erst s gestaffelte Inkremente $\Delta_1, \ldots, \Delta_s$ und daraus dann mit bestimmten Gewichten gemittelt das Gesamt-Inkrement. Der Fall $s = 1$ ist das Euler-Verfahren, zu $s = 2$ gehören das Heun- und das modifizierte Euler-Verfahren. Die dahinter stehende Idee wird erst im nächsten Abschnitt klar.

9.2 Verfahrensfunktion, globaler und lokaler Diskretisierungsfehler

Mit einem Vorrat von konkreten Verfahren vor Augen können wir wieder allgemein werden und die Berechnungsvorschrift für u_k ansetzen als

$$\frac{u_k - u_{k-1}}{h} = \phi_f(h, u), \quad k = 1, 2, \ldots \tag{9.7}$$

in Analogie zur DGL (9.1). Hier ist ϕ_f die **Verfahrensfunktion**, in die die rechte Seite f der DGL irgendwie eingeht. Die Schrittweite h ist der Einfachheit halber konstant gewählt. Die weiteren Argumente, in 9.7 nur durch ein u angedeutet, sind

- bei einem **expliziten 1-Schritt-Verfahren** (E1): u_{k-1},
- bei einem **expliziten s-Schritt-Verfahren** (Es): $u_{k-s}, \ldots, u_{k-1}$,

gleichmäßig in x für alle $x \in [a,b]$. Das erfordert also, daß bei dieser Gitterverfeinerung die Punktmenge $\{x_k, u_k, k = 0, \ldots, n\}$ im Intervall $[x_0, x]$ immer dichter wird und dort gleichmäßig gegen die Lösungskurve $y(\cdot)$ strebt. Ferner ist die **Konvergenz-Ordnung** das größte p, für welches

$$\|e(h,x)\| \le s(x) \cdot h^p \qquad \text{mit einem beschränkten} \quad s\colon [a,b] \to \mathbb{R}.$$

Trotz des sehr ähnlich klingenden Namens ist der **lokale Diskretisierungsfehler** ein ganz anderer Begriff, definiert für einen *beliebigen* Bezugspunkt ξ, η durch

$$\begin{aligned} \tau(h,\xi,\eta) \;:=\; & (z(\xi+h) - z(\xi))/h - \phi_f(h,z) \\ & \text{mit} \quad z(\cdot) = \text{Lösung des lokalen AWP} \\ & z' = f(\cdot, z),\ z(\xi) = \eta. \end{aligned} \tag{9.8}$$

Dieser Begriff hat für den Anwender keine unmittelbare Bedeutung, sondern ist ein technisches Hilfsmittel, das die Analyse der Verfahren erleichtert. Es ist das **Residuum**, welches sich ergibt, wenn man die lokale Lösung z der Differentialgleichung einsetzt in die Näherungsgleichung (9.7), nach der das Verfahren rechnet: Also $u_j = z(x_j)$ und $f_j = f(x_j, z(x_j)) = z'(x_j)$ für alle in (9.7) vorkommenden j.

Bei den expliziten s-Schritt-Verfahren gibt es noch eine andere Interpretation von τ. Hier sind alle Argumente u in $\phi_f(h,u)$ frei wählbar, weil das abhängige u_k dort nicht vorkommt. Somit ist $\phi_f(h,z)$ genau der Zuwachs $(u_k - z(\xi))/h$, den das Verfahren für s exakte Eingabewerte liefert und (9.8) vereinfacht sich zu $\tau(h,\xi,\eta) = (z(\xi+h) - u_k)/h$. Also ist $\tau(h,\xi,\eta)$ der Fehler ohne „Altlasten“, begangen allein im Schritt von ξ nach $\xi + h$ und geteilt durch die Schrittweite h.

Insbesondere bei den expliziten 1-Schritt-Verfahren ist $h_k \cdot \tau(h_k, x_{k-1}, u_{k-1})$ der **zusätzliche Integrationsfehler** im Integrations-Schritt über $h_k := x_k - x_{k-1}$. (Bei den 1-Schritt-Verfahren ist jeder Integrations-Schritt unabhängig von den anderen und die Annahme einer konstanten Schrittweite vereinfacht nichts.)

Bei den impliziten Verfahren ist diese Zweit-Interpretation nicht gültig bzw. nicht ganz exakt. Der Unterschied ist aber so gering, daß man ohne wesentliche Konsequenz genauso gut die zweite Interpretation als Definition benützen kann.

Einer der großen Unterschiede zwischen globalem und lokalem Diskretisierungsfehler ist der Definitionsbereich für h. Bei $\tau(h,\xi,\eta)$ sind alle drei Argumente gewöhnliche kontinuierliche Variablen und das ist eine große Vereinfachung, denn jetzt gibt es im allgemeinen eine Taylor-Entwicklung in h (die Argumente ξ und η spielen immer eine Parameter-Rolle). Zum Beispiel kann man für das klassische Runge-Kutta-Verfahren mit etwas Geschick und Fleiß zeigen, daß $\tau(h,\xi,\eta) = O(h^4)$. Tip: setze $d := h/2$ und lege den Nullpunkt nach $(x_{k-1} + x_k)/2$.

Das Verfahren ϕ_f heißt **konsistent**, wenn der lokale Diskretisierungsfehler τ gleichmäßig in ξ, η gegen 0 strebt für $h \to 0$:

$$\|\tau(h,\xi,\eta)\| \le \sigma(h) \qquad \text{und} \qquad h \to 0 \quad \Longrightarrow \quad \sigma(h) \to 0.$$

Damit haben wir folgende Abschätzungskette

$$\begin{aligned}
\|y(x_n) - u_n\| &= \|z_0(x_n) - z_n(x_n)\| \le \sum_{k=1}^{n} \|z_{k-1}(x_n) - z_k(x_n)\| \\
&\le \sum_{k=1}^{n} \|z_{k-1}(x_k) - z_k(x_k)\| \cdot \exp(L \cdot |x_n - x_k|) \qquad \text{(nach (9.10))} \\
&\le \sigma(h_{\max}) \sum_{k=1}^{n} h_k \exp(L \cdot |x_n - x_k|) \qquad \text{(nach (9.9))} \\
&\le \sigma(h_{\max}) \sum_{k=1}^{n} \int_{x_{k-1}}^{x_k} e^{L\cdot(x_n - x)}\, dx = \sigma(h_{\max}) \int_{x_0}^{x_n} e^{L\cdot(x_n - x)}\, dx.
\end{aligned}$$

■

Zu betonen ist, daß dieser Beweis beliebige Schrittweiten h_k zwischen x_0 und x_n zuläßt und keine Lipschitz-Bedingung für die Verfahrensfunktion ϕ_f unterstellt.

Das Wesentliche von Satz 9.2 ist:

Man hat Proportionalität zwischen dem *globalen* Diskretisierungsfehler auf der linken Seite und dem *lokalen* Diskretisierungsfehler auf der rechten Seite. Die Proportionalitätskonstante $\big(\exp(L\cdot|x_n - x_0|) - 1\big)/L$ hängt *nicht* von den Schrittweiten h_k der Diskretisierung ab. Die Genauigkeits-Analyse eines jeden expliziten 1-Schritt-Verfahrens ist somit zurückgeführt auf eine kleine Fleißarbeit: Bestimmung von $\tau(h, \xi, \eta) = \tau \cdot h^p + \ldots$ durch Taylor-Entwicklung.

Beispiel:

$$\begin{array}{lllll}
\text{Euler-Verfahren:} & p = 1, & \tau(h,\xi,\eta) = O(h), & e(h,x) = O(h), \\
\text{Heun-Verfahren:} & p = 2, & \tau(h,\xi,\eta) = O(h^2), & e(h,x) = O(h^2), \\
\text{Klass. R.-K.-V.:} & p = 4, & \tau(h,\xi,\eta) = O(h^4), & e(h,x) = O(h^4).
\end{array}$$

Zur Demonstration wird das AWP $y' = +\sqrt{x^2 + y^2}$, $y(0) = 0$, mit Lipschitz-Konstante $L = 1$ über das endliche Intervall $[0, 2]$ integriert. $y(2) = 2.5886\,0752\,6700\ldots$

Die folgende Tabelle zeigt den globalen Diskretisierungsfehler $e(h, x)$ an der Stelle $x = 2$.

Information von f im zurückliegenden Integrations-Bereich stützt. Es ist klar, daß solche Information Prognose-Charakter hat und genauso unzuverlässig ist wie der Wetterbericht. Die s-Schritt-Verfahren sind deshalb längst nicht so robust wie die 1-Schritt-Verfahren und gleichen nervösen Rennpferden.

Die Konsistenz-Ordnung wird erläutert an den beiden wichtigsten s-Schritt-Verfahren. O.B.d.A. $k = s$.

BDF:

u_s aus $P'(x_s) = f(x_s, u_s)$ mit $P \in \mathbb{P}_s$ der Interpolant von x_i, u_i für $i = 0, \ldots, s$. Also mit der Lagrange-Formel

$$\sum_{i=0}^{s} u_i L_i'(x_s) = u_s'. \tag{9.11}$$

Adams und Moulton:

$u_s = u_{s-1} + \int_{x_{s-1}}^{x_s} Q(x)\,dx$ mit $Q \in \mathbb{P}_s$ der Interpolant von $x_k, u_k' := f(x_k, u_k)$ für $k = 0, \ldots, s$. Also mit der Lagrange-Formel

$$u_s - u_{s-1} = \sum_{i=0}^{s} u_i' \cdot \int_{x_{s-1}}^{x_s} L_i(x)\,dx. \tag{9.12}$$

Der lokale Diskretisierungsfehler τ war vereinbart als das Residuum, welches man erhält beim Einsetzen einer lokalen Lösung z der DGL: $u_i = z(x_i)$, $u_i' = f(x_i, z(x_i)) = z'(x_i)$. Dabei muß man (9.12) zuvor noch normieren mittels Division durch die Intervall-Länge $x_s - x_{s-1}$.

Die Durchführung ist leicht möglich mit Hilfe folgender zwei Restglied-Formeln, zu beweisen ähnlich wie in Abschnitt 4.4,

$$\begin{aligned} z'(x_s) - P'(x_s) &= D^{s+1}z(\xi_1) \cdot (x_s - x_0) \cdots (x_s - x_{s-1})/(s+1)! \\ \int_{x_{s-1}}^{x_s} [z'(x) - Q(x)]\,dx &= D^{s+2}z(\xi_2) \cdot \int_{x_{s-1}}^{x_s} (x - x_0) \cdots (x - x_s)\,dx/(s+1)! \end{aligned}$$

Läßt man nun alle Schrittweiten gleichmäßig gegen 0 streben, dann findet man damit

$$\tau_{\mathrm{BDF}} = O(h^s) \quad \text{und} \quad \tau_{\mathrm{A\&M}} = O(h^{s+1}), \qquad h := x_s - x_{s-1}.$$

Anders ausgedrückt, die Konsistenz-Ordnung dieser beiden impliziten s-Schritt-Verfahren ist s bzw. $s+1$.

Für s-Schritt-Verfahren ist der Zusammenhang zwischen lokalem Diskretisierungsfehler τ und globalem Diskretisierungsfehler $e(h, x)$ an der Stelle x längst nicht

Satz 9.4 *Dahlquist-Barriere (1956)*

Ein lineares s-Schritt-Verfahren, das die Stabilitäts-Bedingung 9.3 erfüllt, hat eine Konsistenz-Ordnung $p \leq s+1$ (falls s ungerade) und $\leq s+2$ (falls s gerade). Die Ordnung $s+2$ für gerades s kann nur erzielt werden, wenn alle Nullstellen von ψ auf dem Rande der Einheits-Kreisscheibe liegen: $\psi(\zeta) = 0 \Rightarrow |\zeta| = 1$.

Zum Beispiel wird für $s = 2$ die Barriere $p = 4$ erreicht von dem in Abschnitt 9.1 erwähnten Milne-Verfahren, wo

$$\begin{aligned} u_{k+1} - u_{k-1} &= \int_{x_{k-1}}^{x_{k+1}} Q(x)\,dx \qquad \text{mit} \quad Q(x_i) = u_i' \text{ für } i = k-1, k, k+1 \\ &= \frac{h}{3}(u'_{k-1} + 4u'_k + u'_{k+1}). \end{aligned}$$

Hier ist $\psi(\zeta) = \zeta^2 - 1$ mit Nullstellen nur auf dem Einheitskreis.

9.5 Die Extrapolations-Methode von Bulirsch und Stoer

Man betrachte zunächst eine numerische Integration des AWP über das Intervall $[x_0, x_1]$ mit Hilfe von einem Euler-Schritt (Anlauf-Rechnung) und $n-1$ Schritten der Mittelpunktsregel:

$$\begin{aligned} h &:= (x_1 - x_0)/n \\ u_0 &:= y_0 \\ u_1 &:= u_0 + hf(x_0, u_0) \\ &\text{für } i = 1 \text{ bis } n\text{:} \quad u_{i+1} := u_{i-1} + 2hf(x_0 + ih, u_i). \end{aligned} \tag{9.15}$$

Es ist vorteilhaft, jedoch logisch nicht erforderlich, noch einen Schlußschritt anzufügen:

$$\hat{u}_n(h) := \frac{1}{4}(u_{n-1} + 2u_n + u_{n+1}).$$

W. Gragg hat gezeigt, daß diese Berechnungsvorschrift für hinreichend glattes f einen globalen Diskretisierungsfehler hat mit einer h^2-Entwicklung wie in Abschnitt 8.2 für die Trapezsummen bewiesen:

$$\begin{aligned} \hat{u}_n(h) &:= y(x_1) + \gamma_1 h^2 + \ldots + \gamma_{m-1} h^{2m-2} + \gamma_m(h) \cdot h^{2m}, \\ n \cdot h &= x_1 - x_0, \quad n \text{ gerade}, \quad |\gamma_m(h)| \leq \Gamma. \end{aligned}$$

Man kann dann dieses Schema (9.15) für die Bulirsch-Folge $n = 4, 6, 8, 12, \ldots$ berechnen und die erhaltenen Rohwerte $\hat{u}_n(h)$ durch Extrapolation auf 0 mittels des Aitken-Neville-Schemas veredeln. Wie bei der Trapezsumme besprochen, eliminiert man dabei die Fehlerterme proportional zu $h^2, h^4, \ldots$ durch geschickte Linearkombination der Rohwerte $\hat{u}_n(h)$. Das Ergebnis ist eine Näherung für y am Endpunkt

Erster Fall: Anfangsbedingung $y(0) = 1$

$n =$	4	6	8	12
	−5.15E−01	−2.52E−01	−1.47E−01	−6.70E−02
		−4.15E−02	−1.19E−02	−3.17E−03
			−2.06E−03	−2.50E−04
				−2.40E−05

16	24	32	48	64
−3.81E−02	−1.70E−02	−9.60E−03	−4.27E−03	−2.41E−03
−8.18E−04	−2.07E−04	−5.22E−05	−1.31E−05	−3.28E−06
−3.49E−05	−3.78E−06	−5.25E−07	−5.80E−08	−8.14E−09
3.24E−07	1.07E−07	7.83E−09	4.17E−10	2.13E−11
1.95E−06	9.22E−08	1.25E−09	−7.78E−11	−5.01E−12
	3.92E−08	−2.07E−09	−1.16E−10	−2.36E−12
		−2.72E−09	−8.48E−11	−5.58E−13
			−6.63E−11	1.89E−13
				4.50E−13

Zweiter Fall: Anfangsbedingung $y(0) = 0$

$n =$	4	6	8	12	16
	−6.87E−02	−3.49E−02	−2.07E−02	−9.60E−03	−5.49E−03
		−7.83E−03	−2.45E−03	−7.18E−04	−2.06E−04
			−6.61E−04	−1.39E−04	−3.52E−05
				−7.36E−05	−1.82E−05
					−1.45E−05

24	32	48	64	96	128
−2.47E−03	−1.40E−03	−6.24E−04	−3.51E−04	−1.56E−04	−8.79E−05
−5.73E−05	−1.59E−05	−4.31E−06	−1.17E−06	−3.15E−07	−8.44E−08
−7.77E−06	−2.05E−06	−4.66E−07	−1.25E−07	−2.88E−08	−7.75E−08
−4.34E−06	−1.11E−06	−2.68E−07	−6.91E−08	−1.67E−08	−4.31E−09
−3.41E−06	−8.94E−07	−2.12E−07	−5.58E−08	−1.33E−08	−3.48E−09
−3.10E−06	−8.02E−07	−1.93E−07	−5.01E−08	−1.20E−08	−3.13E−09
	−7.66E−07	−1.83E−07	−4.78E−08	−1.14E−08	−2.99E−09
		−1.79E−07	−4.66E−08	−1.12E−08	−2.91E−09
			−4.61E−08	−1.10E−08	−2.88E−09
				−1.10E−08	−2.86E−09
					−2.85E−09