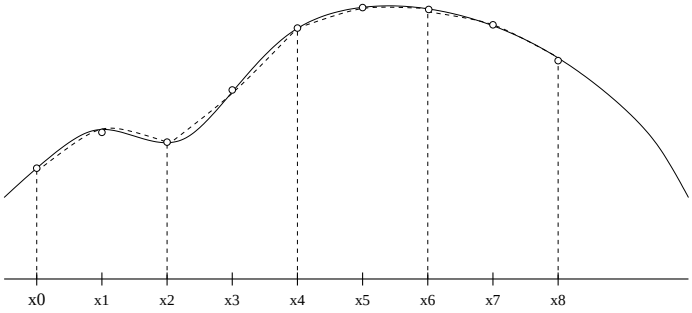
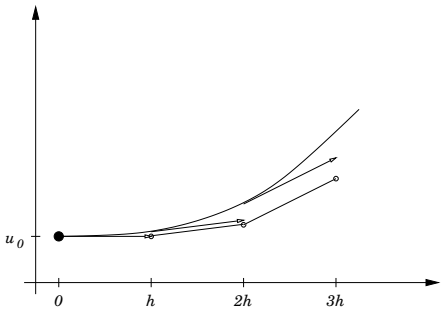


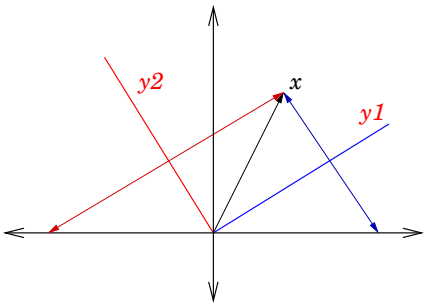


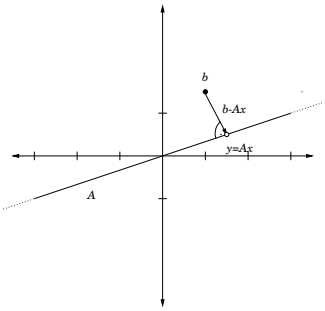
Numerische Mathematik II

Zuerst gehalten im Sommersemester 2000

Tomas Sauer

Version 3.2
Letzte Änderung: 3.8.09





Statt einer Leerseite . . . 9

Wie hoch von nöten sey Arithmetic / und die ganze Mathematische Kunst / kan man hierauß leichtlich ermessen / daß nichts bestehen mag / so nicht mit gewisser zahl und maß vereinet ist / daß auch kein freye Kunst ohn gewisse Mensuren und Proportion der zahlen seyn mag / Derohalben billich Plato / als ein haupt der Philosophen / keinen in sein Schul oder andern Künsten zugelassen / der der zahl nicht erfahren were / als dem nicht möglich / irgend in einer Kunst zuzunehmen / disputiert und bestendiglich beschleußt / daß ohn Arithmeticam / Musicam / unnd Geometricam / welche in der zahl gegründet / niemand weise mög genannt werden.

Adam Riese, Vorrede zum Rechenbuch [33]

9 Inhaltsverzeichnis

10 Ausgleichsrechnung **3**
10.1 Inverse Probleme – ein kurzer Exkurs 3
10.2 Nun aber überbestimmt 6
10.3 Ein typisches Beispiel 7
10.4 Least–squares–Lösungen und die Normalengleichungen 7
10.5 Singuläre–Werte–Zerlegung und die Pseudoinverse 11
10.6 Orthogonaltransformationen 15
10.7 Orthogonaltransformationen II – Givens–Rotationen 28
10.8 Smoothing Splines und glättende Approximation 31

11 Eigenwertprobleme **34**
11.1 Eigenwerte und Nullstellen von Polynomen 34
11.2 Terminologie und die Schur–Zerlegungen 36
11.3 Vektoriteration 40
11.4 Das QR–Verfahren I – Theorie 43
11.5 Das QR–Verfahren II – Praxis 47
11.6 Das Jacobi–Verfahren für symmetrische Matrizen 68

12 Nochmals Nullstellen von Polynomen **71**
12.1 Abdivieren, jetzt aber richtig 71
12.2 Sturmsche Ketten 80

13 Numerische Integration und orthogonale Polynome **88**
13.1 Quadraturformeln 89
13.2 Klassische Formeln 93
13.3 Zusammengesetzte Quadraturformeln 95
13.4 Gauß–Quadratur 99
13.5 Orthogonale Polynome 104
13.6 Jacobipolynome 110
13.7 Quadratur nach Art des Gaußes 113

14 Numerik gewöhlicher Differentialgleichungen **118**
14.1 Woher kommen die eigentlich? 118
14.2 Einschrittverfahren I – Euler und Taylor 121
14.3 Einschrittverfahren II – Extrapolation 124
14.4 Quadraturformeln 126
14.5 Mehrschrittverfahren 129

14.6 Instabilität und Stabilität 131
14.7 Systeme und Steifigkeit 133
14.8 Galerkin–Verfahren 134
14.9 Differential- und Differenzengleichungen 138
14.10 Lineare Systeme und die FFT 142

A Anhang: Gut zu wissen **152**
A.1 Orthogonalisierung und Gram–Schmidt 152
A.2 Wie schnell kann man 153

Für den Ausgleichsanspruch nach §1587g Abs. 1 Satz 1 gelten die §§ 1580, 1585 Abs. 1 Satz 2,3 und §1585b Abs. 2,3 entsprechend.

BGB, §1587k. Abs. 1

10 Ausgleichsrechnung

In der numerischen Linearen Algebra in Teil I [36] haben wir uns mit *quadratischen* linearen Gleichungssystemen der Form $Ax = b$, $A \in \mathbb{R}^{n\times n}$ und deren numerischer Lösung beschäftigt, das heißt, die Anzahl der Gleichung und die Anzahl der Unbekannten stimmen überein – eine notwendige Voraussetzung wenn man für *alle* rechten Seiten b eine *eindeutige* Lösung erhalten will. Diese Theorie läßt sich relativ einfach auf *unterbestimmte* Gleichungssysteme erweitern, bei denen man weniger Gleichungen als Unbekannte hat. Man verliert zwar die Eindeutigkeit der Lösung, aber solange die Matrix vollen Rang hat, kann man sich (zumindest formal) auf ein eindeutig lösbares *Teilsystem* zurückziehen[1]. Besser macht man es aber auf andere Art und Weise

10.1 Inverse Probleme – ein kurzer Exkurs

Nehmen wir jetzt also an, wir hätten es mit einem *unterbestimmten* linearen Gleichungssystem der Form

$$Ax = b, \qquad A \in \mathbb{R}^{m\times n}, \quad m < n, \qquad b \in \mathbb{R}^m, \tag{10.1}$$

zu tun, dessen Matrix A, der Einfachheit halber, aber den *maximalen Rang* m hat[2]. Anstatt nun aber den Lösungsraum einfach einzuschränken, suchen wir uns in unserer Lösungsmenge eine *beste* Lösung, wobei natürlich noch klarstellen müssen, in welchem Sinne die Lösung optimal zu sein hat.

Bemerkung 10.1 *Unterbestimmte Gleichungssysteme kommen in der Anwendungsrealität oft vor und zwar im Kontext der sogenannten* inversen Systeme*. Dabei kennt man eine (hoffentlich oder näherungsweise) lineare Beziehung der Form $y = Ax$ zwischen den (gesuchten) Regelgrößen x und den gemessenen Werten y, nur sind es oftmals viel zu viele Regelgrößen und viel*

[1] Die Auswahl der "besten" Lösung aus dem affinen Lösungsraum ist hingegen ein durchaus nichttriviales Problem.

[2] Das ist ohnehin der "generische" Fall, die Menge der Matrizen mit Rangdefekt bildet einen Nullmenge in der Menger aller $m \times n$–Matrizen.

zu wenige Messungen[3]*. Das führt dann zu einem unterbestimmten System, aus dem x nun in irgendeiner Form bestimmt werden muss.*

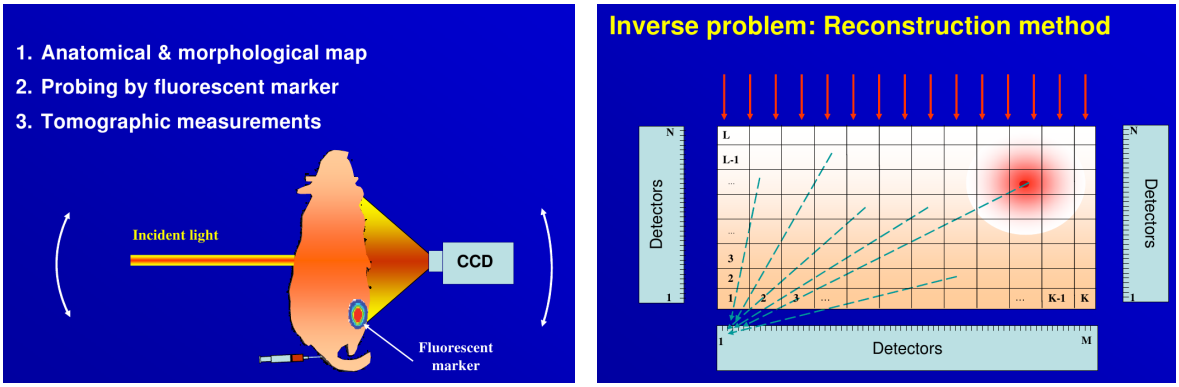


Abbildung 10.1: Optische Tomographie *(links)*. Ein fluoreszierender Marker *innerhalb* einens Lebewesens (hier eine Maus) wird durch Laserbestrahlung aktiviert und das Streulicht am anderen Ende aufgesammelt. Aus diesen Informationen soll die Lage des Markers rekonstruiert werden.

Im physikalischen Modell *(rechts)* wird der Lichtaustausch zwischen benachbarten Zellen durch entsprechende Beugungs-, Reflektions- und Absorptionsgesetze ermittelt, woraus sich der Lichteinfall auf den Detektoren ergibt.

Beispiel 10.2 (Optische Tomographie) *In der* optischen Tomographie *werden Objekte, normalerweise Lebewesen*[4] *ausschließlich mit Hilfe von sichtbarem Licht durchleuchtet, wobei im Inneren ein fluoreszierender Marker plaziert wird, siehe Abb. 10.1, der durch das einfallende (Laser-) Licht aktiviert wird. Es klingt ein wenig abwegig mit Hilfe von Licht durch eine Maus "hindurchschauen" zu wollen, aber tatsächlich kommt wirklich einiges an Licht auf der anderen Seite des Objekts an, allerdings durch die Moleküle*[5] *gestreut und gebrochen.*

Zur Modellierung dieses Problems teilt man jeden Schnitt durch das Objekt und seine Umgebung in kleine Quadrate ein und modelliert den Lichtaustausch zwischen diesen Quadraten[6] *sowie den Randquadraten und den Detektoren. Bezeichnet $x \in \mathbb{R}^{MN}$ die Lichtintesität an den Quadraten des $M \times N$–Gitters und $y \in \mathbb{R}^K$ die gemessenen Werte an den K Detektoren, die um das Objekt herum oder hinter dem Objekt angebracht sind, dann erhalten wir ein Gleichunssystem der Form*

$$y = Ax, \qquad A \in \mathbb{R}^{K \times MN}.$$

[3]Die Anzahl der Messungen kann aus vielerlei Gründen beschränkt sein: Kosten, technische Randbedingungen, Zeitbegrenzung ...

[4]Und in diesem Sinne eigentlich eher Subjekte als Objekte.

[5]Bekanntlicherweise bei Lebewesen zumeist Wasser.

[6]Der Einfachheit halber werden hier normalerweise QUadrate und deren Nachbarn betrachtet.

Ab einer bestimmten Rekonstruktionsgenauigkeit hat man dann aber mit Sicherheit wesentlich mehr Quadrate als Detektoren und das Problem wird massiv unterbestimmt!

Jetzt aber genug der Realität und zurück zur Theorie. Da unser unterbestimmtes Gleichungssytem $Ax = b$ aus (10.1) normalerweise jede Menge Lösungen hat, genauer gesagt, einen $n - m$–dimensionalen Lösungsraum, müssen wir uns aus diesen Lösungen eine gute Lösung aussuchen – warum also nicht gleich die Beste? Dazu lösen wir das Optimierungsproblem

$$\min_x \gamma(x), \qquad Ax = b, \tag{10.2}$$

mit einem vorgegeben *Gütefunktional* $\gamma : \mathbb{R}^n \to \mathbb{R}$, das zumindest nach unten beschränkt sein sollte[7]; diese untere Schranke können wir dann auch gleich auf Null setzen und so $\gamma : \mathbb{R}^n \to \mathbb{R}_+$ annehmen. Beliebte Werte für γ sind $\gamma(x) = \|x\|_p$, $1 \le p \le \infty$, oder

$$\gamma(x) = \frac{1}{2}\, x^T B x, \qquad B \text{ positiv definit.} \tag{10.3}$$

Die Lösung von (10.3) wollen wir uns noch schnell etwas genauer ansehen. Dazu erinnern wir uns, daß ein *notwendiges* Kriterium für die Existenz eines Extremums von $f : \mathbb{R}^N \to \mathbb{R}$ unter der Nebenbedinung $g(x) = 0$, $g : \mathbb{R}^n \to \mathbb{R}^m$, durch die Existenz *Lagrange–Multiplikatoren* $\lambda \in \mathbb{R}^m$ gegeben ist, daß also

$$\nabla f(x) - \nabla g(x)\, \lambda = 0, \qquad \nabla g := \left[\frac{\partial}{\partial x_j} g_k \;:\; \begin{array}{c} j = 1, \ldots, n \\ k = 1, \ldots, m \end{array} \right], \tag{10.4}$$

gelten muss. Dieses Resultat lernt man entweder in der Analysis unter dem Stichwort "Extrema unter Nebenbedingungen", siehe z.B. [21], oder in der Optimierung, siehe z.B. [38, 42]. Mit $f(x) = \frac{1}{2} x^T B x$ und $g(x) = Ax - b$ ist nun ganz einfach $\nabla f(x) = Bx$ sowie $\nabla g(x) = A^T$, so daß unsere gesuchte Optimallösung als Lösung von

$$\begin{array}{rcl} Bx - A^T\lambda &=& 0 \\ Ax &=& b \end{array} \qquad \Leftrightarrow \qquad \left[\begin{array}{cc} B & A^T \\ A & 0 \end{array} \right] \left[\begin{array}{c} x \\ -\lambda \end{array} \right] = \left[\begin{array}{c} 0 \\ b \end{array} \right]$$

gegeben ist. Daß dieses Extremum ein Minimum sein muss, liegt an der Tatsache, daß $x \mapsto x^T B x$ ein nach oben geöffnetes Paraboloid darstellt und deswegen nur ein Minimum haben kann, siehe auch den Beweis von Proposition 10.7. Anders gesagt: Die Minimallösung bestimmt man durch Lösen des *symmetrischen* quadratischen $m + n \times m + n$–Systems

$$Cx = d, \qquad C = \left[\begin{array}{cc} B & A^T \\ A & 0 \end{array} \right], \qquad d = \left[\begin{array}{c} 0 \\ b \end{array} \right], \tag{10.5}$$

wobei man sich nur um die ersten n Variablen zu kümmern braucht, die Werte von λ interessieren ja herzlich wenig.

Übung 10.1 Zeigen Sie: Hat A den maximalen Rang m, dann ist C aus (10.5) invertierbar. $\diamondsuit$

[7] Sonst gibt es normalerweise kein Minimum!

10.2 Nun aber überbestimmt

Jetzt geht es aber um *überbestimmte* lineare Systeme der Form

$$Ax \simeq b, \qquad A \in \mathbb{R}^{m\times n}, \qquad m > n, \quad b \in \mathbb{R}^m, \tag{10.6}$$

bei denen $x \in \mathbb{R}^n$ gesucht wird. Im allgemeinen[8] wird das "Gleichungssystem" (10.6) nicht mehr lösbar sein, weswegen man das folgende Ziel verfolgt:

Man finde $x \in \mathbb{R}^n$, so daß $Ax - b$ möglichst klein wird.

"Möglichst klein" kann beispielsweise bedeuten, daß man x so bestimmt, daß $\|Ax - b\|$ für eine vorgegebene Vektornorm minimiert wird – dabei wird das optimale x im allgemeinen sehr wohl von der gewählten Vektornorm abhängen.

Beispiel 10.3 *Für*

$$A = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, \qquad b = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}, \quad b_1 \le b_2 \le b_3$$

haben die Minimallösungen $x \in \mathbb{R}$ für $\|Ax - b\|_p$ die folgende Gestalt:

$p = 1$**:** *Man sieht leicht, daß $b_1 \le x \le b_3$ gelten muß, und dann ist*

$$\|Ax - b\|_1 = (x - b_1) + |x - b_2| + (b_3 - x) = b_3 - b_1 + |x - b_2|,$$

was für $x = b_2$ minimal wird.

$p = 2$**:** *Differenzieren von*

$$\|Ax - b\|_2^2 = 3x^2 - 2(b_1 + b_2 + b_3)x + b_1^2 + b_2^2 + b_3^2$$

liefert sofort, daß $x = (b_1 + b_2 + b_3)/3$.

$p = \infty$**:** *Da das Maximum von $|x - b_j|$ immer entweder für $j = 1$ oder $j = 3$ angenommen wird, ist die optimale Wahl, x von beiden gleich weit entfernt sein zu lassen, also $x = (b_3 + b_1)/2$.*

Im Spezialfall $b_1 = b_2 = b_3 =: b$ ergibt sich natürlich immer $x = b$, die Lösung des Gleichungssystems.

Übung 10.2 Bestimmen Sie die Optimallösung in Beispiel 10.3 für beliebiges $1 < p < \infty$.

In diesem Kapitel soll aber nur die Norm

$$\|\cdot\|_2 \ : \ x \mapsto \sqrt{x^T x}$$

betrachtet werden.

[8] Das hängt natürlich von der rechten Seite b ab.

10.3 Ein typisches Beispiel

In [36, Beispiel 7.37] haben wir gesehen, daß polynomiale Interpolation von Punkten auf einer Parabel nach leichter Störung normalerweise ein katastrophales Ergebnis liefert. Dies lag daran, daß die zur *eindeutigen* Interpolation nötigen Polynome einfach numerisch zu instabil sind. Auf der anderen Seite könnte man nun, in dem Wissen, daß alle Interpolationspunkte ja "fast" auf einer Parabel liegen, diejenige Parabel wählen, die zwar nicht durch alle Punkte geht, ihnen aber am nächsten kommt.

Präzisieren wir das Ganze ein wenig: Es seien $x_1, \ldots, x_m \in \mathbb{R}$ und $y_1, \ldots, y_m \in \mathbb{R}$ die zugehörigen Ordinaten. Das daraus entstehende Minimierungsproblem ist also

$$\min_{z \in \mathbb{R}^3} \|Az - y\|,$$

wobei

$$A = \begin{bmatrix} 1 & x_1 & x_1^2 \\ \vdots & \vdots & \vdots \\ 1 & x_n & x_n^2 \end{bmatrix} \qquad \text{und} \qquad y = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}.$$

Unter Vorgriff auf Proposition 10.7 ergibt sich das `Matlab`–Programm 10.1 zur Bestimmung der optimalen Parabel zu den gestörten Daten.

Beispiel 10.4 *Wir wählen die $n+1$ gleichverteilten Punkte $x_j = \frac{j}{n}$, $j = 0, \ldots, n$, in $[0,1]$ und wollen die Parabel $x(1-x)$ approximieren, wobei die Daten jeweils um einen festen Prozentsatz gestört werden. Die numerischen Werte des Maximalfehlers der "rekonstruierten" Koeffizienten sind für einige Werte von n und einige Störungen in der folgenden Tabelle angegeben.*

n	10%	5%	1%	0.5%	0%
10	0.012071	0.017943	0.0057670	0.0017770	1.9984×10^{-15}
20	0.010238	0.016348	0.0018193	0.00098497	3.9968×10^{-15}
50	0.0058672	0.010225	0.00061026	0.00097319	1.2879×10^{-14}
100	0.015457	0.0062513	0.00076983	0.0015011	4.8850×10^{-15}
200	0.015770	0.0042976	0.00076559	0.00039072	4.2855×10^{-14}

Zugegeben – als Ergebnisse einmaliger *Tests haben diese Zahlen keine große Aussagekraft, aber dennoch belegen sie, daß die Rekonstruktion der Parabel auch für große Werte von n noch halbwegs genau ist.*

10.4 Least–squares–Lösungen und die Normalengleichungen

Definition 10.5 *Ein Vektor $x \in \mathbb{R}^n$ heißt* Least squares*–Lösung von (10.6), wenn*

$$\|Ax - b\|_2 \leq \|Ay - b\|_2, \qquad y \in \mathbb{R}^n.$$

Bemerkung 10.6 *Existiert eine* Lösung *von (10.6), dann ist diese natürlich auch Least–squares–Lösung.*

```
%#
%# LSInt.m
%# Least Squares Interpolation (mit Polynomen)
%# Daten:
%#   x  Abszissenwerte
%#   y  Ordinatenwerte
%#   n  Polynomgrad (n < length( x ))

function z = LSInt( x,y,n )
  m = length(x);
  A = zeros( m,n+1 );

  for j = 1:m
    for k = 0:n
      A( j,k+1 ) = x(j)^k;
    end
  end

  % Gleichungssystem

  B = A' * A;
  b = A' * y';

  z = B \ b;
%endfunction
```

Programm 10.1 `LSInt.m`: Least–squares–Interpolation mit Polynomen.

Das schöne an der $\|\cdot\|_2$–Norm ist die Tatsache, daß wir die Least–squares–Lösung einfach bestimmen können.

Proposition 10.7 *Sei $A \in \mathbb{R}^{m\times n}$, $m \geq n$. Dann ist jede Lösung x des linearen Gleichungssystems*[9]

$$\underbrace{A^T A}_{\in\mathbb{R}^{n\times n}} x = \underbrace{A^T b}_{\in\mathbb{R}^n} \tag{10.7}$$

eine Lösung des Least–squares–Problems (10.6) und umgekehrt.

Korollar 10.8 *Hat die Matrix A vollen Rang, dann ist die Lösung des Least–squares–Problems eindeutig.*

Beweis: Hat A vollen Rang n, dann hat auch $A^T A \in \mathbb{R}^{n\times n}$ den Rang n und ist damit invertierbar. Damit gibt es aber auch (nur) genau eine Lösung der Normalengleichungen. □

Beweis von Proposition 10.7: Wir betrachten das (quadratische) Funktional

$$\begin{aligned} \|Ax-b\|_2^2 &= (Ax-b)^T (Ax-b) = x^T A^T Ax - x^T A^T b - b^T Ax + b^T b \\ &= x^T A^T Ax - 2x^T A^T b + b^T b, \end{aligned}$$

das genau dann durch x minimiert wird, wenn die Norm durch x minimiert wird. Ist nun x ein Minimalstelle, so muss

$$0 = \nabla \|Ax-b\|_2^2 = 2A^T Ax - 2A^T b$$

sein, siehe Übung 10.3, also hat jede Least–Squares–Lösung x sich an die Normalengleichungen (10.7) zu halten.
Erfüllt umgekehrt x (10.7), dann erhalten wir für jedes $y \in \mathbb{R}^n$, daß

$$\begin{aligned} &\|Ay-b\|_2^2 \\ &= \|Ax-b\|_2^2 + \underbrace{(y-x)^T A^T A (y-x)}_{\geq 0} - 2x^T A^T Ax + 2y^T A^T Ax + 2(x-y)^T A^T b \\ &\geq \|Ax-b\|_2^2 - 2(x-y)^T \underbrace{A^T Ax}_{=A^T b} + 2(x-y)^T A^T b = \|Ax-b\|_2^2, \end{aligned}$$

also $\|Ax-b\|_2 \leq \|Ay-b\|_2$ und somit ist x auch eine Least–Squares–Lösung.

□

Übung 10.3 Zeigen Sie: Ist $B \in \mathbb{R}^{n\times n}$ eine symmetrische Matrix und $F(x) := x^T Bx$, $x \in \mathbb{R}^n$, dann ist $\nabla F(x) = 2Bx$. ♢

[9] Oft auch als *Normalengleichungen* bezeichnet.

Bemerkung 10.9 *Die notwendige und hinreichende Bedingung (10.7) bedeutet, daß die Least–squares–Lösung so zu wählen ist, daß der "Residuenvektor" $r := Ax - b$ die Bedingung $A^T r = 0$ erfüllt, oder eben, daß $r \in \ker A^T$. Diese Bedingung hat eine geometrische Interpretation, siehe Abb. 10.2: Es sei*

$$V := \mathscr{R}(A) := A\mathbb{R}^n := \{Ax \ : \ x \in \mathbb{R}^n\}$$

der von (den Spalten von) A erzeugte Vektorraum[10]*. Dann ist das orthogonale Komplement*

$$\mathbb{R}^n \ominus \mathscr{R}(A) = \left\{x \in \mathbb{R}^n \ : \ x^T \mathscr{R}(A) = 0\right\}$$

gerade $\ker A^T$. Somit sind die Least–squares–Lösungen diejenigen Vektoren, bei denen der Residuenvektor senkrecht auf V steht.

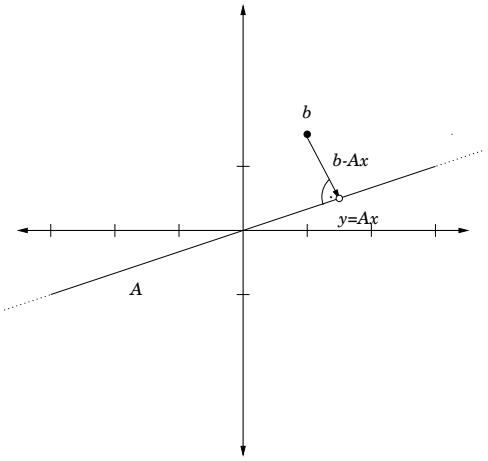


Abbildung 10.2: Geometrische Interpretation der Least–squares–Lösung: Der Fehler ist orthogonal zum Vektor Ax.

Damit hätten wir bereits einen (naiven) Ansatz, um überbestimmte Gleichungssysteme zu lösen: wir verwenden das Cholesky–Verfahren, um die Lösung von $A^T Ax = A^T b$ zu bestimmen. Allerdings führt der Übergang zu einem Gleichungssystem bezüglich $A^T A$, also die "Quadrierung" der Matrix auch zu einer Quadrierung des Fehlers.

Beispiel 10.10 *Es sei $A \in \mathbb{R}^{n+1 \times n}$ gegeben durch*

$$A = \begin{bmatrix} 1 & 1 & \dots & 1 \\ \varepsilon & & & \\ & \varepsilon & & \\ & & \ddots & \\ & & & \varepsilon \end{bmatrix},$$

[10] $\mathscr{R}(A)$ steht für "Range" von A.

dann ist

$$A^TA = \begin{bmatrix} 1 & \varepsilon & & & \\ 1 & & \varepsilon & & \\ \vdots & & & \ddots & \\ 1 & & & & \varepsilon \end{bmatrix} \begin{bmatrix} 1 & 1 & \dots & 1 \\ \varepsilon & & & \\ & \varepsilon & & \\ & & \ddots & \\ & & & \varepsilon \end{bmatrix} = \begin{bmatrix} 1+\varepsilon^2 & 1 & \dots & 1 \\ 1 & 1+\varepsilon^2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 1 \\ 1 & \dots & 1 & 1+\varepsilon^2 \end{bmatrix}.$$

Diese Matrix ist quadratisch *schlechter konditioniert als die Ausgangsmatrix! Denn da* $B := A^TA = 1_n 1_n^T + \varepsilon^2 I$*, ist jeder Vektor* $y \in \mathbb{R}^n$ *mit* $1_n^T y = 0$ *ein Eigenvektor von* B *zum Eigenwert* ε^2*. Außerdem ist*

$$B\ 1_n = 1_n \underbrace{\left(1_n^T 1_n\right)}_{=n} + \varepsilon^2 1_n = \left(n+\varepsilon^2\right)\ 1_n.$$

Also ist, unter Verwendung des Spektralradius*, siehe [36, Definition 6.3, Proposition 6.4],*

$$\kappa_2\left(A^TA\right) = \left\|A^TA\right\|_2\ \left\|\left(A^TA\right)^{-1}\right\|_2 \geq \rho\left(A^TA\right)\ \rho\left(\left(A^TA\right)^{-1}\right) = \frac{\lambda_{\max}\left(A^TA\right)}{\lambda_{\min}\left(A^TA\right)} = 1 + \frac{n}{\varepsilon^2}.$$

Da jede *Norm größer als der Spektralradius ist, ist die obige negative Aussage nicht etwa spezifisch für die 2–Norm, sondern gilt für jede beliebige Matrixnorm und damit auch für jede beliebige Konditionszahl.*

In [23, S. 398, (19.14)] kann man diese quadratisch verschlechterte Situation auch konkreter finden: Für die berechnete Lösung $\hat{x}$ des Least–squares–Problems ergibt sich ein relativer Vorwärtsfehler von

$$\frac{\|x-\hat{x}\|_2}{\|x\|_2} \leq C_{m,n}\ \kappa_2^2(A)\ \hat{u}, \qquad x \in \mathbb{R}^n \setminus \{0\}.$$

Die Notation ist wie in [36].

10.5 Singuläre–Werte–Zerlegung und die Pseudoinverse

Wir wollen uns nun der Beschreibung von Least–squares–Lösungen etwas theoretischer nähern.

Definition 10.11 *Eine quadratische Matrix* $U \in \mathbb{R}^{n\times n}$ *bzw.* $U \in \mathbb{C}^{n\times n}$ *heißt* orthogonal *bzw.* unitär*, wenn* $U^TU = I$ *bzw.* $U^HU = I$*.*
Eine Matrix $D \in \mathbb{R}^{m\times n}$ *heißt* diagonal*, wenn* $a_{jk} = 0$ *falls* $j \neq k$*,* $j = 1,\dots,m$*,* $k = 1,\dots,n$*.*

Bemerkung 10.12 *Die orthogonalen Matrizen bilden eine multiplikative Gruppe: Jede orthogonale Matrix* U *hat eine orthogonale Inverse (nämlich* U^T*) und das Produkt zweier orthogonaler Matrizen* U, V *ist wieder orthogonal (da* $(UV)^TUV = V^TU^TUV = V^TV = I$*).*

Satz 10.13 *Zu jeder Matrix* $A \in \mathbb{R}^{m\times n}$*,* $m, n \in \mathbb{N}$*, gibt es orthogonale Matrizen* $U \in \mathbb{R}^{m\times m}$*,* $V \in \mathbb{R}^{n\times n}$ *und eine Diagonalmatrix* $\Sigma \in \mathbb{R}^{m\times n}$ *so daß*

$$A = U\Sigma V^T, \qquad \sigma_{11} \geq \sigma_{22} \geq \dots \geq \sigma_{\ell,\ell} \geq 0, \qquad \ell := \min(m,n). \tag{10.8}$$

Definition 10.14 *Die Darstellung (10.8) heißt* Singuläre–Werte–Zerlegung *bzw.* singular value decomposition *(SVD), die Diagonalwerte von* Σ, $\sigma_j = \sigma_{jj}$, $j = 1, \ldots, \ell$, *heißen* singuläre Werte *von* A.

Beweis: Induktion über ℓ. Im Fall $\ell = 1$ nehmen wir ohne Einschränkung[11] an, daß $m \leq n$. Dann hat A die Form $A = [a_{11} \cdots a_{1n}] =: a^T$. Ist $a = 0$, dann bilden $U = 1$, $\Sigma = 0$ und $V = I$ die SVD von A. Andernfalls setzen wir $v_1 = \|a\|_2^{-1}\, a$, also als normierte Version von a und vervollständigen diesen Vektor zu einer Orthonormalbasis des $\mathbb{R}^n$, das heißt, wir wählen $v_2, \ldots, v_n$ so, daß

$$\mathbb{R}^n = \text{span}\,\{v_1, \ldots, v_n\}\,, \qquad v_j^T v_k = \delta_{jk}, \quad j, k = 1, \ldots, n,$$

und erhalten die Komponenten der SVD als

$$U = 1, \qquad V = [v_j \;:\; j = 1, \ldots, n]\,, \qquad \Sigma = U^T A V = [\|a\|_2\, 0 \cdots 0]\,.$$

Sei also Satz 10.13 für ein $\ell > 0$ bewiesen und seien $m, n \in \mathbb{N}$ dergestalt, daß $\min(m, n) = \ell + 1$. Wir setzen

$$\tau_1 := \max_{x \in \mathbb{R}^n} \frac{\|Ax\|_2^2}{\|x\|_2^2} = \max_{\|x\|_2 = 1} x^T \left(A^T A\right) x. \tag{10.9}$$

Ist $\tau_1 = 0$, dann ist $A = 0$ und die SVD ergibt sich als[12] $U = I$, $V = I$, $\Sigma = 0$. Andernfalls ist τ_1 der größte Eigenwert der symmetrischen positiv definiten Matrix $A^T A$; sei außerdem $v_1 \in \mathbb{R}^n$ ein zugehöriger normierter Eigenvektor. Diesen kann man wieder mit $v_2, \ldots, v_n$ zu einer Orthonormalbasis des $\mathbb{R}^n$ ergänzen und damit ist die Matrix $V = [v_j \;:\; j = 1, \ldots, n]$ orthogonal. Wegen (10.9) ist $\|Av_1\|_2 = \sigma_1$, $\sigma_1 = \sqrt{\tau_1}$, und damit ist

$$0 \neq u_1 := \frac{1}{\sigma_1} A v_1 \quad \in \mathbb{R}^m, \qquad \|u_1\|_2 = 1.$$

Wie vorher vervollständigen wir diesen Vektor zu einer Orthonormalbasis, diesmal mit $u_2, \ldots, u_m$ und setzen $U = [u_j \;:\; j = 1, \ldots, m]$. Damit ist

$$U^T A V = \begin{bmatrix} \sigma_1 & 0 \\ 0 & B \end{bmatrix} \qquad \Longleftrightarrow \qquad A = U \begin{bmatrix} \sigma_1 & 0 \\ 0 & B \end{bmatrix} V^T, \qquad B \in \mathbb{R}^{m-1 \times n-1}, \tag{10.10}$$

wobei der größte Eigenwert von $B^T B$ höchstens σ_1 ist. Nach Induktionsannahme gibt es Matrizen $\widehat{U} \in \mathbb{R}^{m-1 \times m-1}$, $\widehat{V} \in \mathbb{R}^{n-1 \times n-1}$ und $\widehat{\Sigma} \in \mathbb{R}^{m-1 \times n-1}$, so daß

$$B = \widehat{U} \widehat{\Sigma} \widehat{V}^T, \qquad \sigma_1 \geq \hat{\sigma}_1 \geq \cdots \geq \hat{\sigma}_{\ell-1} \geq 0.$$

Setzt man das in (10.10) ein, so ergibt sich, daß

$$A = \underbrace{U \begin{bmatrix} 1 & 0 \\ 0 & \widehat{U} \end{bmatrix}}_{=:U} \underbrace{\begin{bmatrix} \sigma_1 & 0 \\ 0 & \widehat{\Sigma} \end{bmatrix}}_{=:\Sigma} \underbrace{\begin{bmatrix} 1 & 0 \\ 0 & \widehat{V}^T \end{bmatrix} V^T}_{=:V^T},$$

womit der Beweis fertig ist. □

[11] Ansonsten gehen wir zu A^T über.

[12] Achtung: U und V können Einheitsmatrizen *unterschiedlicher* Größe sein.

Bemerkung 10.15 *(Beobachtungen zur SVD)*

1. *Der singuläre Wert τ_1 aus (10.9) ist gleichzeitig der größte Eigenwert der symmetrischen positiv semidefiniten Matrix A^TA: Seien $\lambda_1 \geq \cdots \geq \lambda_n \geq 0$ die Eigenwerte von $B = A^TA$ und $x_1, \ldots, x_n$ zugehörige orthonormale Eigenvektoren, dann ist*

$$x^TBx = \left(\sum_{j=1}^n \alpha_j x_j\right)^T B \left(\sum_{j=1}^n \alpha_j x_j\right) = \sum_{j=1}^n \alpha_j^2 \lambda_j \leq \lambda_1 \sum_{j=1}^n \alpha_j^2 = \lambda_1 \|x\|_2^2.$$

2. *Generell sind die singulären Werte einer Matrix A die (immer noch nichtnegativen) Wurzeln der Eigenwerte von A^TA.*

3. *Die Idee der SVD wird aus dem Beweis ersichtlich: man bestimmt orthogonale Vektoren $u_1, \ldots, u_m \in \mathbb{R}^m$, $v_1, \ldots, v_n \in \mathbb{R}^n$, so daß die von ihnen gebildeten Matrizen U und V die Ausgangsmatrix A dergestalt orthogonal transformieren, daß*

$$\left(U^TAV\right)\ e_j = \sigma_j\ e_j, \qquad j = 1, \ldots, \ell,$$

   also die kanonischen Einheitsvektoren in sich selbst abbilden.

4. *Die SVD ist nicht immer eindeutig! Das einfachste Gegenbeispiel ist $A = 0$, dann ist $\Sigma = 0$ und man kann U und V sogar frei wählen. Was natürlich eindeutig ist*[13] *ist die Matrix Σ.*

Definition 10.16 *Die* Moore–Penrose–Inverse *oder* Pseudoinverse *einer $m \times n$–Matrix A mit SVD $A = U\Sigma V^T$ ist definiert als*

$$A^+ = V\Sigma^+U^T, \qquad \Sigma^+ \in \mathbb{R}^{n\times m}$$

wobei

$$\left(\Sigma^+\right)_{jk} = \begin{cases} \sigma_{jk}^{-1}, & \sigma_{jk} \neq 0, \\ 0 & \sigma_{jk} = 0, \end{cases} \qquad j = 1, \ldots, n,\ k = 1, \ldots, m.$$

Der Name "Pseudoinverse" ist leicht erklärt. Ist nämlich $A \in \mathbb{R}^{n\times n}$ eine *invertierbare* Matrix, dann sind wegen

$$0 < |\det A| = \underbrace{|\det U|}_{=1}\ |\det \Sigma|\ \underbrace{|\det V|}_{=1} = \prod_{j=1}^n \sigma_j$$

alle singulären Werte von A strikt positiv und damit ist $\Sigma^+ = \Sigma^{-1}$. Daher ist

$$A\ A^+ = U\Sigma V^TV\Sigma^{-1}U^T = I = V\Sigma^{-1}U^TU\Sigma V^T = A^+\ A$$

und damit ist $A^+ = A^{-1}$. Im Falle einer invertierbaren Matrix kann man das "Pseudo–" also getrost vergessen.

Die wichtigsten Eigenschaften der Pseudoinversen beschreibt der folgende Satz.

[13] Wegen der absteigenden Anordnung der singulären Werte.

Satz 10.17 *Sei $A \in \mathbb{R}^{m\times n}$, $m > n$.*

1. *Die Pseudoinverse A^+ erfüllt die* Moore–Penrose–Bedingungen
   - *(a)* $A^+A = (A^+A)^T$,
   - *(b)* $AA^+ = (AA^+)^T$,
   - *(c)* $AA^+A = A$,
   - *(d)* $A^+AA^+ = A^+$.
2. *Der Vektor $x^* = A^+b$ erfüllt*

$$\|Ax^* - b\|_2 = \min_{x\in\mathbb{R}^n} \|Ax - b\|_2 ,$$

x^* ist also eine Least–squares–Lösung.

Beweis: Sei $A = U\Sigma V^T$. Zum Beweis von 1. bemerkt man, daß die Matrizen

$$A^+A = V\Sigma^+U^T\ U\Sigma V^T = V\Sigma^+\Sigma V^T \qquad \text{und} \qquad AA^+ = U\Sigma V^TV\Sigma^+U^T = U\Sigma\Sigma^+U^T$$

beide symmetrisch sind und daß

$$AA^+A = U\Sigma V^TV\Sigma^+\Sigma V^T = U\Sigma\Sigma^+\Sigma V^T$$

beziehungsweise

$$A^+AA^+ = V\Sigma^+U^TU\Sigma\Sigma^+U^T = V\Sigma^+\Sigma\Sigma^+U^T.$$

Außerdem gibt es ein $1 \le r \le \ell$ so daß $\sigma_1 \ge \cdots \ge \sigma_r > \sigma_{r+1} = \cdots = \sigma_\ell = 0$ und damit ist

$$\Sigma\Sigma^+\Sigma = \begin{bmatrix} \sigma_1\sigma_1^{-1}\sigma_1 & & & & & \\ & \ddots & & & & \\ & & \sigma_r\sigma_r^{-1}\sigma_r & & & \\ & & & 0 & & \\ & & & & \ddots & \\ & & & & & 0 \end{bmatrix} = \Sigma$$

und entsprechend $\Sigma^+\Sigma\Sigma^+ = \Sigma^+$.
Um 2. nachzuweisen berechnen wir einfach

$$A^TA\ x^* = A^TAA^+b = V\Sigma^TU^T\ U\Sigma V^T\ V\Sigma^+U^T\ b = V\Sigma^TU^T\ b = A^Tb$$

und nach Proposition 10.7 ist $x^* = A^+y$ eine Least–squares–Lösung. □

Übung 10.4 Zeigen Sie: Die Moore–Penrose–Bedingungen bestimmen die Pseudoinverse A^+ eindeutig.

Mit Hilfe der Pseudoinversen kann man die Definition der *Konditionszahl* auch auf singuläre und nichtquadratische Matrizen erweitern, indem man

$$\kappa_2(A) := \|A\|_2 \ \left\|A^+\right\|_2 \tag{10.11}$$

setzt.
So schön die SVD und die Pseudoinverse theoretisch auch sein mögen – zur *Berechnung* unserer Least–squares–Lösung helfen sie uns wenig, denn die Berechnung der SVD oder der Pseudoinversen sind möglicherweise sogar noch komplizierter.

Bemerkung 10.18 *In gewissem Sinne ist die Pseudoinverse ein Lieblingskind der Ingenieure, die schon vor langer Zeit herausgefunden haben, daß der Befehle* `pinv` *in* `Matlab/Octave` *immer eine "Inverse" liefert, aber nie eine dieser nervigen Fehlermeldungen oder Warnungen. Daß dafür die Ergebnisse nicht sonderlich sinnvoll sein können*[14] *nimmt man oftmals gerne in Kauf. Es braucht durchaus einiges an Überzeugungsarbeit, um dafür zu sorgen, daß die Pseudoinverse nicht in unangebrachter Weise verwendet*[15] *wird!*

10.6 Orthogonaltransformationen

In diesem Kapitel wollen wir Transformationsmethoden (eine Art Elimination) unter Verwendung *orthogonaler* Matrizen betrachten, um so die Lösung des Ausgleichsproblems *direkt* aus der Matrix A zu ermitteln. Warum man gerade orthogonale Matrizen verwendet, ist eigentlich ziemlich naheliegend: weil für jede orthogonale Matrix $U \in \mathbb{R}^{n\times n}$

$$\|Ux\|_2 = \sqrt{(Ux)^T (Ux)} = \sqrt{x^T U^T U x} = \|x\|_2 , \qquad x \in \mathbb{R}^n,$$

gilt, die 2–Norm also *orthogonal invariant* ist, ist auch die Matrixnorm $\|\cdot\|$ orthogonal invariant[16] und damit ist auch

$$\kappa_2 (UA) = \kappa_2 (AV) = \kappa_2 (A) , \qquad A \in \mathbb{R}^{m\times n},\ U \in \mathbb{R}^{m\times m},\ V \in \mathbb{R}^{n\times n},$$

Multiplikation mit orthogonalen Matrizen verschlechtert also nicht die Kondition des Problems[17]. Was wir also brauchen, sind einfache orthogonale Matrizen, mit denen man "was anfangen kann".

Definition 10.19 *Zu* $0 \neq y \in \mathbb{R}^m$ *ist die* Householder–Matrix $H = H(y)$ *definiert als*

$$H(y) = I - 2\frac{yy^T}{\|y\|_2^2}.$$

[14] Bei der Bildung der Pseudoinversen muss entschieden werden, ab wann ein singulärer Wert als $= 0$ angesehen werden soll – da steckt dann auch das gesamte numerische Teufelswerk!

[15] Genauso unangebracht wie das grammatisch inkorrekte "verwandt" an dieser Stelle.

[16] Entsprechendes gilt auch immer für komplexe Matrizen und *unitäre* Invarianz

[17] Und den Traum, daß sich während der Rechnung ein Problem in ein besser konditioniertes umwandelt, sollte man schon in der Numerik I aufgegeben haben.

Die Idee, Householder–Matrizen zur Lösung von Least–squares–Problemen zu verwenden geht[18] auf Householder [24] zurück, die praktischen Details wurden in [17] ausgearbeitet.

Lemma 10.20 *Householder–Matrizen sind symmetrisch und orthogonal:* $H^T = H$ *und* $H^T H = I$.

Beweis: Symmetrie ist trivial und die Orthogonalität folgt aus

$$H^T H = H^2 = \left(I - \frac{2}{\|y\|_2^2} y y^T \right)^2 = I - \frac{4}{\|y\|_2^2} y y^T + \frac{4}{\|y\|_2^2} y \underbrace{\frac{y^T y}{\|y\|_2^2}}_{=1} y^T = I.$$

□

Der Einfachheit halber wollen wir nun immer annehmen, daß der *Householder–Vektor* y für die Bildung der Householder–Matrix $H(y)$ *normiert* ist, das heißt $\|y\|_2 = 1$, denn dann ist $H(y) = I - 2yy^T$.

Bemerkung 10.21 *Die Householder–Transformation* $x \mapsto H(y)x$ *kann man auch geometrisch interpretieren, und zwar als Spiegelung an der Hyperebene*

$$E := \left\{ x \in \mathbb{R}^m \ : \ x^T y = 0 \right\},$$

siehe Abb. 10.3. Schreibt man nämlich x *als* $x = \alpha y + y^\perp$, $y^T y^\perp = 0$, *dann ist*[19]

$$H(y)\, x = \left(I - 2yy^T \right) \left(\alpha y + y^\perp \right) = \alpha y + y^\perp - 2\alpha y \underbrace{y^T y}_{=1} - 2y \underbrace{y^T y^\perp}_{=0} = -\alpha y + y^\perp,$$

das heißt, die Projektion von x *in die Hyperebene* E *bleibt unverändert, während das Vorzeichen des Anteils in Richtung* y *umgedreht wird.*

Sei also $A \in \mathbb{R}^{m \times n}$, $m > n$. Die "Strategie" ist nun naheliegend: Wir wollen mit Hilfe von Householder–Matrizen wie bei der Gauß–Elimination A auf eine obere Dreiecksgestalt bringen, d.h. *normierte* Vektoren $y_1, \ldots, y_k$ finden, so daß

$$H(y_k) \cdots H(y_1)\, A = \left[\begin{array}{ccc|c|ccc} * & \ldots & * & * & * & \ldots & * \\ & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & & * & * & * & \ldots & * \\ \hline & & & a_{k+1}^{(k+1)} & * & \ldots & * \\ & & & \vdots & \vdots & \ddots & \vdots \\ & & & a_m^{(k+1)} & * & \ldots & * \end{array}\right], \qquad k = 0, \ldots, n-1. \quad (10.12)$$

[18] Welch ein Wunder ...

[19] Nicht vergessen: $\|y\|_2 = 1$!

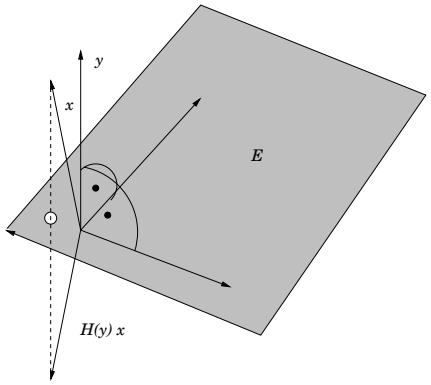


Abbildung 10.3: Die Householder–Transformation als Spiegelung an der Hyperebene $E = y^{\perp}$. Im phyiskalischen Sinn ist das also eher eine Brechung als eine Spiegelung: “Einfallswinkel gleich Ausfallswinkel”.

Für $k = 0$ ist (10.12) trivialerweise erfüllt. Um von (10.12) einen Schritt weiterzukommen, müssen wir also einen Vektor $\hat{y}_{k+1} \in \mathbb{R}^{m-k}$ und eine Zahl $\alpha_{k+1} \in \mathbb{R}$ bestimmen, so daß[20]

$$H\left(\hat{y}_{k+1}\right)\ a^{(k+1)} = \alpha_{k+1} e_1 \in \mathbb{R}^{m-k}, \qquad a^{(k+1)} = \begin{bmatrix} a_{k+1}^{(k+1)} \\ \vdots \\ a_m^{(k+1)} \end{bmatrix} \in \mathbb{R}^{m-k}, \tag{10.13}$$

denn ergibt sich unter Verwendung von

$$y_{k+1} := \begin{bmatrix} 0 \\ \hat{y}_{k+1} \end{bmatrix} \in \mathbb{R}^m \qquad \Longrightarrow \qquad H\left(y_{k+1}\right) = \begin{bmatrix} I_k & 0 \\ 0 & H\left(\hat{y}_{k+1}\right) \end{bmatrix} \in \mathbb{R}^{m \times m},$$

daß

$$H\left(y_{k+1}\right) \cdots H\left(y_1\right) A = \left[\begin{array}{ccc|c|ccc} * & \dots & * & * & * & \dots & * \\ & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & & * & * & * & \dots & * \\ \hline & & & \alpha_{k+1} & * & \dots & * \\ & & & 0 & * & \dots & * \\ & & & \vdots & \vdots & \ddots & \vdots \\ & & & 0 & * & \dots & * \end{array}\right].$$

Das entscheidende Problem besteht also darin, (10.13) zu lösen. Daß und wie das geht, sagt uns das nächste Lemma.

[20] Hierbei bezeichnet e_j wieder den j–ten kanonischen Einheitsvektor.

Lemma 10.22 *Sei $x \in \mathbb{R}^m$. Setzt man*[21] $\gamma = 2\,\|x\|_2$, $\epsilon = -\operatorname{sgn} x_1$ *und*

$$y_1 = \sqrt{\frac{1}{2} + \frac{|x_1|}{\gamma}}, \qquad y_j = -\epsilon\,\frac{x_j}{\gamma\,y_1}, \quad j = 2, \ldots, m, \tag{10.14}$$

dann ist $\|y\|_2 = 1$ und $H(y)\,x = \alpha\,e_1$, wobei $\alpha = -\epsilon\|x\|_2$.

Dieser Berechnungsschritt ist im `HousehStep.m` zu sehen.

Übung 10.5 Folgern Sie aus (10.14), daß $\|y\|_2 = 1$.

Beweis: Natürlich könnte man "einfach" nachrechnen, daß die Bedingungen aus (10.14) die gewünschten Eigenschaften liefern. Nachdem aber diese Voraussetzungen ja normalerweise nicht vom Himmel fallen, wollen wir sie explizit herleiten – das ist zwar etwas aufwendiger, aber auch "realistischer" und zeigt uns, wo man numerisch aufpassen muss.
Da $H(y)$ orthogonal ist, muß zuerst einmal

$$|\alpha| = \|H(y)\,x\|_2 = \|x\|_2$$

gelten, das heißt $\alpha = \epsilon\|x\|_2$, $\epsilon \in \{\pm 1\}$. Nach der Definition von $H(y)$ ist $H(y)\,x = \alpha\,e_1$ äquivalent zu

$$x_1 - 2y_1\left(y^T x\right) = \alpha, \qquad x_j - 2y_j\left(y^T x\right) = 0, \quad j = 2, \ldots, m. \tag{10.15}$$

Aus diesen Bedingungen ergibt sich mit $\beta := y^T x$ und der Zusatzforderung $\|y\|_2 = 1$, daß

$$\begin{aligned} \alpha\,y_1 &= y_1\underbrace{(x_1 - 2y_1\,\beta)}_{=\alpha} + \sum_{j=2}^{m} y_j\underbrace{(x_j - 2y_j\,\beta)}_{=0} = \sum_{j=1}^{m} x_j y_j - 2\beta\sum_{j=1}^{m} y_j^2 \\ &= \beta - 2\,\beta\,\underbrace{\|y\|_2^2}_{=1} = -\beta, \end{aligned}$$

also $\beta = -\alpha y_1$. Eingesetzt in die erste Gleichung von (10.15) liefert dies, daß

$$x_1 + 2\alpha y_1^2 = \alpha \qquad \Longrightarrow \qquad y_1 = \sqrt{\frac{1 - x_1/\alpha}{2}}.$$

Wählen wir nun $\epsilon \in \{-1, 1\}$ so, daß

$$\epsilon x_1 < 0, \qquad \Rightarrow \qquad x_1 = -\epsilon\,|x_1| \tag{10.16}$$

dann ergibt sich

$$y_1 = \sqrt{\frac{1}{2} - \frac{-\epsilon\,|x_1|}{\epsilon\,\|x_2\|}} = \sqrt{\frac{1}{2} + \frac{|x_1|}{2\;\|x\|_2}} \qquad \Longrightarrow \qquad \beta = -\epsilon\,\|x\|_2\;y_1$$

und schließlich

$$y_j = \frac{x_j}{2\beta} = -\epsilon\,\frac{x_j}{2y_1\;\|x\|_2}, \qquad j = 2, \ldots, m.$$

□

[21] Hierbei können wir das Vorzeichen $\operatorname{sgn} 0$ der Zahl Null beliebig als ± 1 wählen.

```
%#
%# HousehStep.m
%# Berechne y-Vektor fuer Householder-Matrix
%# Daten:
%#   x  Zu transformierender Vektor

function y = HousehStep( x )
  m = length( x );
  y = zeros( m,1 );

  g = 2 * sqrt( x' * x );

  if g < eps
    return;
  endif

  s = sign( x(1) );
  if s = 0              % Sonderfall !!!
    s = 1;
  end

  y( 1 ) = sqrt( 0.5 + abs( x(1) )/g );
  gg = g * y( 1 );
  for j = 2:m
    y( j ) = s * x( j ) / gg;
  end
%endfunction
```

Programm 10.2 `HousehStep.m`: Bestimmung des Vektors y für die Householder–Matrix. Die Behandlung des Sonderfalls $x = 0$ wird noch gerechtfertigt werden.

```
%#
%# HousehStep.m
%# Berechne y-Vektor fuer Householder-Matrix
%# Daten:
%#   x  Zu transformierender Vektor

function y = HousehStep2( x )
  m = length( x );
  y = zeros( m,1 );

  g = 2 * sqrt( x' * x );

  if g < eps
    return;
  endif

  if x(1) == 0            % Sonderfall !!!
    s = 1;
  else
    s = sign( x(1) );
  end

  y( 1 ) = sqrt( 0.5 - abs(x(1))/g );
  gg = g * y( 1 );
  for j = 2:m
    y( j ) = -s * x( j ) / gg;
  end
%endfunction
```

Programm 10.3 `HousehStep2.m`: Die "dumme" Art, den Vektor y zu bestimmen, mit der numerisch falschen Wahl von ϵ.

```
%#
%# HousehMat.m
%# Berechne Householder-Matrix H(y) = I - 2 y y^T/y^T y
%# Daten:
%#   y  Vektor

function H = HousehMat( y )
  H = eye( length(y) ) - 2*y*y'/(y'*y);
```

Programm 10.4 `HousehMat.m`: Berechnung der Householder–Matrix zu `y`

Bemerkung 10.23 *Die Wahl von ϵ in (10.16) war beliebig, aber absolut sinnvoll, insbesondere in numerischer Hinsicht, denn nun werden unter der Wurzel bei der Bestimmung von y_1 zwei* positive *Zahlen addiert, so daß* Auslöschung *vermieden wird, siehe [36]. Und da in (10.14) durch y_1 dividiert wird, sollten wir ohnehin dafür sorgen, daß diese Zahl so groß wie möglich wird, einfach um die Fehlerfortpflanzung möglichst klein zu halten.*

Die maximale Auslöschung, siehe [36], bei "falschen" ϵ träte übrigens genau dann auf, wenn $|x_1| \sim \|x_2\|$ wäre, also genau dann, wenn der Vektor x sowieso schon den Wert hätte, den er haben sollte. Und das kann man tatsächlich auch deutlich *sehen wie in Beispiel 10.24 gezeigt werden wird. an kann die Bestimmung des Householder–Vektors aber auch wie in Abb. 10.4 geometrisch interpretieren. Im $\mathbb{R}^2$ sind die Spiegelungsebenen die beiden Winkelhalbierenden der von x und der x_1–Achse erzeugten Winkel; diese Winkel summieren sich natürlich zu 2π. Wird nun einer der Winkel immer kleiner, dann ist zwar die Spiegelung am spitzen Winkel immer unproblematischer, dafür wird aber die Spiegelung an der anderen Geraden eine immer kritischere Operation, und am schlimmsten ist das ganze, wenn einer der Winkel gegen 0 geht. Keinen Unterschied gibt es, wenn $x_1 = 0$ ist, also in dem Fall, in dem wir ϵ nach Gusto wählen können, weil wir* $\operatorname{sgn} 0$ *beliebig in ± 1 festlegen dürfen.*

Beispiel 10.24 *Vergleichen wir einmal für $n = 2$ die beiden Routinen aus den Programmen 10.2 und 10.3, und zwar zuerst für eine eher braven Vektor:*

```
octave> x = [ 1 1 ]';
octave> y1 = HousehStep (x); y2 = HousehStep2 (x);
octave> HousehMat (y1)*x
ans =

   -1.4142e+00
   -1.1102e-16

octave> HousehMat (y2)*x
ans =
```

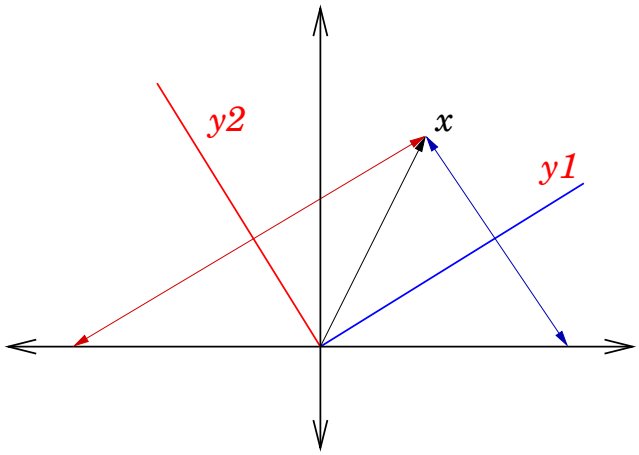


Abbildung 10.4: Die geometrische Interpretation der Householder–Transformationen: x wird an den beiden Winkelhalbierenden gespiegelt, und das "gute" ϵ entspricht dem spitzen, das schlechte dem stumpfen Winkel.

```
   1.4142e+00
   4.4409e-16
```

und in beiden Fällen wir x praktisch auf ein Vielfaches des Einheitvektors abgebildet[22]*. Das wird aber anders, wenn wir den Fall*

```
octave> x = [ 1 10^(-5) ]';
octave> y1 = HousehStep (x); y2 = HousehStep2 (x);
```

betrachten, denn nun erhalten wir das schon signifikant schlechtere Ergebnis

```
octave> HousehMat( y1 ) * x
ans =

   -1.0000e+00
    1.1911e-22

octave> HousehMat( y2 ) * x
ans =

   1.0000e+00
   8.2815e-13
```

Noch interessanter wird es für

[22] Daß im zweiten Fall der Fehler ein wenig größer ist hat noch nicht wirklich etwas zu sagen.

```
octave> x = [ 1 .2*10^(-7) ]';y1 = HousehStep (x); y2 = HousehStep2 (x);
octave> HousehMat( y1 ) * x
ans =

   -1.0000e+00
   -1.1325e-24

octave> HousehMat( y2 ) * x
ans =

   1.0000e+00
   2.2045e-09
```

und richtig übel im Fall

```
octave> x = [ 1 .18*10^(-7) ]'; y2 = HousehStep2 (x);
warning: in /.../HousehStep2.m near line 26, column 12:

>>> y (j) = -s * x (j) / gg;


warning: division by zero
octave> HousehMat( y2 ) * x
ans =

           NaN
           NaN
```

wo die Auslöschung absolut gnadenlos zuschlägt.

Unabhängig von den numerischen Fußangeln haben wir auch schon wieder eine neue Zerlegung für Matrizen kennengelernt, die sogenannte *QR–Zerlegung*.

Satz 10.25 *Sei $A \in \mathbb{R}^{m\times n}$, $m \geq n$. Dann gibt es eine orthogonale Matrix $Q \in \mathbb{R}^{m\times m}$ und eine* Rechtsdreiecksmatrix[23]

$$R = \left[\begin{array}{ccc} * & \dots & * \\ & \ddots & \vdots \\ 0 & & * \\ \hline 0 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 0 \end{array}\right] \in \mathbb{R}^{m\times n},$$

so daß $A = QR$.

```
%#
%# HousehQR.m
%# Berechne QR-Zerlegung fuer Matrix A
%# Daten:
%#   A  Zu zerlegende Matrix, ueberschreibe mit R

function [ Q,R ] = HousehQR( A )
  [m,n] = size( A );
  Q = II = eye( m );

  for j = 1:n
    x = A( j:m,j );
    y = HousehStep( x );            % Berechne orthog. Vektor
    if j > 1
      y = [ zeros( j-1,1 ); y ];    % Auffuellen
    end
    H = II - 2*y*y';                % Householder-Matrix

    A = H * A;
    Q = Q * H;
  end

  R = A;
%endfunction
```

Programm 10.5 `HousehQR.m`: QR–Zerlegung mit Householder–Matrizen.

Realisiert ist die QR–Zerlegung mittels Householder–Transformationen in `HousehQR.m`.
Beweis: Wie man die Vektoren $y_1, \ldots, y_{n-1}$ bestimmt, so daß

$$H(y_{n-1}) \cdots H(y_1) A = R$$

wissen wir ja bereits aus Lemma 10.22 – solange der "eingegebene" Vektor x nicht der Nullvektor ist. Finden wir aber bei unserer Elimination eine "Nullspalte" vor, dann ist das auch nicht schlimm, denn die obere Dreiecksstruktur ist ja dann insbesondere vorhanden und wir können in diesem Fall einfach $y = 0$ setzen. Bleibt nur noch zu bemerken, daß

$$Q = H(y_1)^T \cdots H(y_{n-1})^T = H(y_1) \cdots H(y_{n-1}).$$

□

Ein Wort zum "`flop`–Count"[24]: Der Aufwand bei der Bestimmung von y_k ist $O(m)$, die Multiplikation einer Householder–Matrix mit A von der Größenordnung[25] $O(mn)$, also hat das Gesamtverfahren, weil man ja $(n-1)$–mal Householdermatrizen berechnen und damit multiplizieren muß, einen Aufwand von $O(mn^2)$. Speichert man Q in faktorisierter Form (also die Vektoren $y_1, \ldots, y_{n-1}$, dann läßt sich die Multiplikation

$$Q^T b = H(y_{n-1}) \cdots H(y_1)\ b$$

mit $O(mn)$ Operationen bewerkstelligen, da

$$H(y)b = b - 2\left(y^T b\right) y$$

ist.

Was bringt uns nun die QR–Zerlegung? Nun, mit $A = QR$ wird das Normalengleichungssystem zu

$$R^T Q^T b = A^T b = A^T A\ x = R^T Q^T Q R\ x = R^T R\ x.$$

Schreiben wir

$$R = \begin{bmatrix} \widehat{R} \\ 0 \end{bmatrix}, \qquad \widehat{R} \in \mathbb{R}^{n \times n},$$

dann ist also

$$\left[\widehat{R}^T\ 0\right] Q^T b = \left[\widehat{R}^T\ 0\right] \begin{bmatrix} \widehat{R} \\ 0 \end{bmatrix}\ x = \widehat{R}^T \widehat{R}\ x.$$

[23] Nachdem U für unitäre Matrizen reserviert sein wird, ist R jetzt der Ersatz für unser gutes altes U aus der LU–Zerlegung von [36].

[24] Was in Teil I über die Bedeutung des Zählens von Fließkommaoperationen gesagt wurde, gilt natürlich auch hier.

[25] Das nutzt die Struktur aus und ist wesentlich besser als das $O\left(m^2 n\right)$ bei der Multiplikation einer $m \times m$- mit einer $m \times n$–Matrix.

```
%#
%# HousehLS.m
%# Loese LS-Problem
%#    Ax - b = min
%# ueber QR-Zerlegung mit Householder-Matrizen
%# Daten:
%#   A  Matrix, voller Rang!
%#   b  rechte Seite

function x = HousehLS( A,b )
  [ m,n ] = size( A );
  [ Q,R ] = HousehQR( A );

  R = R( 1:n, 1:n );      % LS-relevanter Teil
  b = Q' * b;

  x = RueckSubs( R , b( 1:n ) );
%endfunction
```

Programm 10.6 `HousehLS.m`: Lösung des Least–squares–Problems nach (10.17).

Hat nun A (und damit auch R und $\widehat{R}$) vollen Rang, dann können wir von links mit $\widehat{R}^{-T}$ multiplizieren und erhalten das einfache Gleichungssystem

$$\widehat{R}x = \widehat{R}^{-T}\left[\widehat{R}^T\ 0\right] Q^T b = [I_n\ 0]\, Q^T b = \begin{bmatrix} \left(Q^T b\right)_1 \\ \vdots \\ \left(Q^T b\right)_n \end{bmatrix},$$

das wir direkt durch Rücksubstitution lösen können. Das fassen wir noch einmal zusammen.

Korollar 10.26 *Die Matrix $A \in \mathbb{R}^{m\times n}$, $m \geq n$, habe vollen Rang und die QR–Zerlegung $A = QR$, $R = \begin{bmatrix} \widehat{R} \\ 0 \end{bmatrix}$. Dann erhält man die Least–squares–Lösung von (10.6) als Lösung des Gleichungssystems*[26]

$$\widehat{R}x = KQ^T b, \qquad K = \sum_{j=1}^{n} e_j\, e_j^T \in \mathbb{R}^{n\times m}. \tag{10.17}$$

Hat die Matrix A nicht vollen Rang, dann hat man aber, auch mit der QR–Zerlegung, ein Problem! Denn um die Gleichung

$$\widehat{R}^T Q^T b = \widehat{R}^T \widehat{R}x$$

[26] Die Matrix K hier ist die *kanonische Koordinatenprojektion* vom $\mathbb{R}^m$ in den $\mathbb{R}^n$.

durch Vorwärtselimination und Rücksubstitution lösen zu können, müssen ja $\widehat{R}$ und damit auch $\widehat{R}^T$ *vollen Rang* haben, insbesondere müssen die Diagonalelemenente von $\widehat{R}$ von Null verschieden sein. Ein "Diagonalwert" 0 in der Matrix R tritt aber nach Lemma 10.22 genau dann auf, wenn während der Householder–Elimination eine *Nullspalte* unter der Diagonalen erzeugt wurde, siehe auch die Bemerkung im Beweis von Satz 10.25. Was man mit Nullspalten macht, wissen wir aber aus der Gauß–Elimination, Stichwort *Pivotsuche*: Nach Anwendung einer Spaltenpermutation können wir entweder mit einer von Null verschiedenen Spalte weitermachen, oder aber die eliminierte Matrix hat die Form

$$\begin{bmatrix} R & B \\ 0 & 0 \end{bmatrix}, \qquad R \in \mathbb{R}^{r\times r},\ B \in \mathbb{R}^{r\times n-r}, \qquad r = \text{rang}\, A.$$

Da für Spaltenvertauschungen eine Permutationsmatrix von rechts multipliziert werden muß, hat also die QR–Zerlegung mit Spaltenpivotsuche die Form

$$AP = QR, \qquad P \in \Pi_n,$$

wobei wieder [36, Definition 4.28] $\Pi_n \subset \mathbb{R}^{n\times n}$ die Gruppe der Permutationsmatrizen bezeichnet. Die Details zur QR–Zerlegung mit Pivotsuche finden sich beispielsweise in [16, Sec. 6.4].

Zum Schluß noch ein Wort zur *Stabilität* der QR–Zerlegung, die so ziemlich genauso ist, wie man es von der Gauß–Elimination erwarten würde. Unter Verwendung der *Frobenius–Norm*

$$\|A\|_F = \left(\sum_{j=1}^{m}\sum_{k=1}^{n} a_{jk}^2\right)^{1/2}, \qquad A \in \mathbb{R}^{m\times n}$$

findet man in [23, Theorem 18.4] das folgende Resultat.

Satz 10.27 *Es sei $A \in \mathbb{R}^{m\times n}$, $m \geq n$, und $\widehat{R} \in \mathbb{R}^{m\times n}$ die mit dem Householder–Verfahren berechnete Rechtsdreiecksmatrix*[27] *der QR–Zerlegung von A. Dann gibt es eine orthogonale Matrix $\widehat{Q} \in \mathbb{R}^{n\times n}$ und eine Matrix $\widehat{A} \in \mathbb{R}^{m\times n}$, so daß*

$$\widehat{A} = \widehat{Q}\widehat{R}$$

und

$$\left\|A - \widehat{A}\right\|_F \leq C\ nm\hat{u}\ \|A\|_F + O\left(nm\hat{u}^2\right)$$

wobei $C > 0$ eine moderate Konstante ist.

Den Beweis schenken wir uns ...

[27] Achtung: hier hat $\widehat{R}$ eine andere Bedeutung als weiter oben in diesem Abschnitt

10.7 Orthogonaltransformationen II – Givens–Rotationen

Eine zweite Methode zur Orthogonaltransformation bilden die sogenannten *Givens–Rotationen* [15], wobei die Idee der Rotationsmatrizen sogar schon auf Jacobi [26] zurückgeht[28]. Um diesen Zugang zu motivieren, sehen wir uns einmal an, wie orthogonale Matrizen eigentlich im 2×2–Fall aussehen. Sei also

$$Q = \begin{bmatrix} q & r \\ s & t \end{bmatrix}$$

eine orthogonale Matrix, das heißt,

$$I = Q^T Q = \begin{bmatrix} q & s \\ r & t \end{bmatrix} \begin{bmatrix} q & r \\ s & t \end{bmatrix} = \begin{bmatrix} q^2 + s^2 & qr + st \\ qr + st & r^2 + t^2 \end{bmatrix} \tag{10.18}$$

Damit folgen aus (10.18) die Beziehungen

$$1 = q^2 + s^2 = r^2 + t^2, \qquad 0 = qr + st.$$

Die erste Identität sagt uns daß wir q, r, s, t parametrisieren können[29], und zwar bis auf Vorzeichen als

$$q = \cos\phi, \quad s = \sin\phi, \qquad r = -\sin\psi, \quad t = \cos\psi, \qquad \phi, \psi \in [0, \pi).$$

Diese Vorzeichen, die genaugenommen beispielsweise $q = \pm\cos\phi$ liefern würden, repräsentieren den *Spiegelungsanteil* der orthogonalen Transformation. Aus der zweiten Beziehung folgt unter Verwendung der Additionstheoreme, daß

$$0 = \cos\phi \, \sin\psi - \sin\phi \, \cos\psi = \sin(\phi - \psi),$$

also $\phi - \psi \in \pi\mathbb{Z}$. Wählen[30] wir $\psi \in [-\pi, \pi]$, dann ist entweder $\phi = \psi$ oder $\phi = \pi + \psi$, was natürlich zu $\phi \in [0, 2\pi]$ führt. Im ersten Fall, $\phi = \psi$, haben die orthogonalen Matrizen $Q \in \mathbb{R}^{2\times 2}$ die Form

$$Q = J(\phi) = \begin{bmatrix} \cos\phi & \sin\phi \\ -\sin\phi & \cos\phi \end{bmatrix}$$

und sind damit *Drehungen* um den Winkel ϕ. Diese Drehungen können wir nun ganz einfach in den $\mathbb{R}^n$ fortsetzen.

[28] Woran man wieder einmal sieht, daß die "Alten" sich auch für Numerik interessiert haben.

[29] Die Einschränkung von ϕ und ψ kommt daher daß Q trivialerweise genau dann orthogonal ist, wenn $-Q$ orthogonal ist.

[30] Winkel sind ja nur modulo 2π definiert.

Definition 10.28 *Zu* $1 \leq j < k \leq n$ *und* $\phi \in [0, 2\pi)$ *definieren wir die* Jacobi–Matrix $J(j, k\,;\,\phi) \in \mathbb{R}^{n\times n}$ *als*

$$J(j,k\,;\,\phi) = \begin{bmatrix}
1 & & & & & & & & \\
& \ddots & & & & & & & \\
& & 1 & & & & & & \\
& & & \cos\phi & & & & \sin\phi & \\
& & & & 1 & & & & \\
& & & & & \ddots & & & \\
& & & & & & 1 & & \\
& & & -\sin\phi & & & & \cos\phi & \\
& & & & & & & & 1 \\
& & & & & & & & & \ddots \\
& & & & & & & & & & 1
\end{bmatrix}$$

(Zeile j und Zeile k sind markiert; Spalte j und Spalte k sind markiert.)

Übung 10.6 Wie sehen die orthogonalen Matrizen in der Situation $\phi = \psi + \pi$ aus? ♢

Bemerkung 10.29 *Die Anwendung der Jacobi–Matrix* $J(j, k\,;\,\phi)$ *entspricht einer* Drehung *um den Winkel* ϕ *in der Ebene, die von den Einheitsvektoren* e_j *und* e_k *aufgespannt wird.*

Die Auswirkung der Multiplikation einer Matrix A von links mit $J(j, k\,;\,\phi)$ ergibt sich somit als

$$\left(J(j,k\,;\,\phi)\,A\right)_{rs} = \begin{cases} a_{rs} & r \neq j,k, \\ \cos\phi\; a_{js} + \sin\phi\; a_{ks} & r = j, \\ -\sin\phi\; a_{js} + \cos\phi\; a_{ks} & r = k, \end{cases} \qquad r,s = 1,\ldots,n,$$

es ändern sich also nur die beiden Zeilen $a_{j\cdot}$ und $a_{k\cdot}$. Auch Jacobimatrizen können wir zur Elimination für $A \in \mathbb{R}^{m\times n}$, $m \geq n$, verwenden: Wir bestimmen zuerst einen Drehwinkel ϕ_{12}, so daß

$$J(1,2\,;\,\phi_{12})\;A = \begin{bmatrix} * & * & \ldots & * \\ 0 & * & \ldots & * \\ * & * & \ldots & * \\ \vdots & \vdots & \ddots & \vdots \\ * & * & \ldots & * \end{bmatrix},$$

dann einen Drehwinkel ϕ_{13}, so daß

$$J(1,3\,;\,\phi_{13})\,J(1,2\,;\,\phi_{12})\;A = \begin{bmatrix} * & * & \ldots & * \\ 0 & * & \ldots & * \\ 0 & * & \ldots & * \\ * & * & \ldots & * \\ \vdots & \vdots & \ddots & \vdots \\ * & * & \ldots & * \end{bmatrix},$$

und so weiter, bis schließlich[31]

$$\left(\prod_{s=m}^{2} J\left(1,s\ ;\ \phi_{1s}\right)\right)\ A = \begin{bmatrix} * & * & \dots & * \\ 0 & * & \dots & * \\ \vdots & \vdots & \ddots & \vdots \\ 0 & * & \dots & * \end{bmatrix}$$

Als nächstes eliminieren wir in der zweiten Spalte mit Hilfe von Drehwinkeln $\phi_{23},\dots,\phi_{2m}$ die Subdiagonalelemente. Da außerdem, für jedes $B \in \mathbb{R}^{m\times n}$ mit $b_{21} = \dots = b_{m1} = 0$ und $r = 2,\dots,m$,

$$\left(J\left(2,k\ ;\ \phi_{2k}\right)B\right)_{r1} = \left\{ \begin{array}{cc} b_{r1} & r \neq 2,k \\ \cos\phi_{2k}\ b_{21} + \sin\phi_{2k}\ b_{k1} & r = 2 \\ -\sin\phi_{2k}\ b_{21} + \cos\phi_{2k}\ b_{k1} & r = k \end{array} \right\} = 0, \qquad k = 3,\dots,m,$$

die Nullen in der ersten Spalte bleiben also unangetastet. Damit ist die Vorgehensweise klar: Man bestimmt Drehwinkel

$$\phi_{12},\dots,\phi_{1m},\phi_{23},\dots,\phi_{2m},\dots,\phi_{n,n+1},\dots,\phi_{nm},$$

so daß

$$\left(\prod_{r=1}^{n}\prod_{s=m}^{r+1} J\left(r,s\ ;\ \phi_{rs}\right)\right)\ A = \left[\begin{array}{ccc} * & \dots & * \\ & \ddots & \vdots \\ 0 & & * \\ \hline 0 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 0 \end{array}\right], \tag{10.19}$$

was wieder nichts anderes als eine QR–Zerlegung ist. Was bleibt ist die Bestimmung eines Drehwinkels ϕ_{rs}, $s = r+1,\dots,m$, $r = 1,\dots,n$, so daß im jeweiligen Schritt

$$\begin{bmatrix} \cos\phi_{rs} & \sin\phi_{rs} \\ -\sin\phi_{rs} & \cos\phi_{rs} \end{bmatrix} \begin{bmatrix} a_{rr} \\ a_{rs} \end{bmatrix} = \begin{bmatrix} \alpha \\ 0 \end{bmatrix}, \qquad \alpha^2 = a_{rr}^2 + a_{rs}^2,$$

was sich immer lösen läßt, wie man sich leicht geometrisch vorstellt. Allerdings brauchen wir nicht die Winkel selbst zu bestimmen, es genügen die Werte von deren Sinus und Cosinus! Die bekommt man aber ganz einfach. Setzen wir

$$c_{rs} = \cos\phi_{rs} \qquad \text{und} \qquad s_{rs} = \sin\phi_{rs},$$

dann erhalten wir aus der zweiten Zeile, daß

$$s_{rs}\ a_{rr} = c_{rs}\ a_{rs} \quad \Longrightarrow \quad \left\{ \begin{array}{l} s_{rs} = \dfrac{(\text{sign}\ a_{rr})\,|a_{rs}|}{\sqrt{a_{rr}^2 + a_{rs}^2}}, \\ c_{rs} = \dfrac{(\text{sign}\ a_{rs})\,|a_{rr}|}{\sqrt{a_{rr}^2 + a_{rs}^2}}, \end{array} \right.$$

[31] Das etwas seltsame Produkt soll die Reihenfolge der Multiplikation andeuten!

wobei sign $0 = 1$ gesetzt wird[32]. Bei dieser Wahl von c_{rs} und s_{rs} ergibt sich übrigens

$$\alpha = (\text{sign } a_{rr})\,(\text{sign } a_{rs})\,\sqrt{a_{rr}^2 + a_{rs}^2}.$$

Übung 10.7 Implementieren Sie die QR–Zerlegung mittels Givens–Rotationen in `Matlab` oder `Octave`.

10.8 Smoothing Splines und glättende Approximation

In Abschnitt 10.3 haben wir uns mit einem, vielleicht dem typischen Beispiel für Least–Squares–Probleme beschäftigt, nämlich mit verallgemeinerten, oftmals überbestimmten Interpolationsproblemen. Dabei sind Punkte $x_j \in \mathbb{R}$, $j = 1, \ldots, N$, Daten y_j, $j = 1, \ldots, N$, und ein endlichdimensionaler Funktionenraum $\mathscr{F}$ mit Basis f_k, $k = 1, \ldots, M$, gegeben, wobei durchaus $M \neq N$ sein kann und wir uns weder auf $M \leq N$ noch auf $M \geq N$ festlegen wollen.

Wir erinnern uns, daß wir wie in [36] das Interpolationsproblem

$$f(x_j) = y_j, \qquad j = 1, \ldots, N,$$

mittels des Ansatzes $f = \sum c_j f_j$ als

$$Fc = y, \tag{10.20}$$

mit

$$F = \left[f_k(x_j) \;;\; \begin{array}{c} j = 1, \ldots, N \\ k = 1, \ldots, M \end{array} \right], \quad c = \begin{bmatrix} c_1 \\ \vdots \\ c_M \end{bmatrix}, \quad y = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix} \tag{10.21}$$

schreiben können. Da $F \in \mathbb{R}^{N \times M}$ nicht unbedingt eine quadratische Matrix ist, können wir das Gleichungssystem nicht unbedingt exakt lösen[33], aber es gibt ja immer mindestens eine Lösung im Sinne der kleinsten Quadrate[34], nach der wir uns umsehen wollen.

Ein typisches Beispiel, das sowohl in der Statistik [4] als auch beim "Fitting" von Kurven und Flächen und in der Lerntheorie eine bedeutende Rolle spielt [2, 3], sind die sogenannten *Smoothing splines*. Hierbei handelt es sich um einen Least–Squares–Approximanten mit einem zusätzlichen Glättungsterm. Aber immer schön der Reihe nach! Als Basisfunktionen verwenden wir die aus dem ersten Teil [36] der Vorlesung hoffentlich noch bekannten B–Splines $N_j^m(\cdot \,|\, T)$ der Ordnung m zu einer Knotenfolge T, verwenden aber ansonsten den Ansatz aus (10.20). Nun hat bei gestörten Daten[35] der Least–Squares–Approximant einen Hang zu unerwünschter Oszillation. Deswegen gibt man lieber etwas von der Approximationsgüte auf und versucht, statt dessen eine "glattere" Kurve zu erhalten, wobei wir Glattheit wieder mit einer *Energienorm*[36] messen, beispielsweise mit dem ebenfalls aus [36] bekannten

$$|f|_2 = \int |f''(x)|^2 \, dx. \tag{10.22}$$

[32] Aber nur der Vollständigkeit wegen, der Wert -1 wäre genau so gut.

[33] Ob wir es exakt lösen können, hängt in vielen Fällen ja auch und gerade von der rechten Seite y ab.

[34] Für irgendwas muss dieses Kapitel ja gut sein.

[35] Messfehler und dergleichen Dinge, die in der Realität oft unvermeidbar sind.

[36] Um ganz genau zu sein: Es handelt sich hierbei natürlich wieder einmal um eine Energie**halb**norm!

Unter Verwendung eines *Regularisierungsparameters* $\lambda \in \mathbb{R}_+$ minimiert man dann den Ausdruck[37]

$$\|Fc - y\|_2^2 + \lambda \, |f|_2^2 = \sum_{j=1}^{N} \left(f\left(x_j\right) - y_j\right)^2 + \lambda \int \left|f''(x)\right|^2 \, dx.$$

Verwenden wir wieder die generelle lineare Darstellung $f = \sum c_j f_j$, dann ist

$$\begin{aligned} |f|_2^2 &= \int \left(\sum_{j=1}^{M} c_j f_j''(x) \right)^2 dx = \int \sum_{j,k=1}^{N} c_j c_k f_j''(x) f_k''(x) \, dx \\ &= \sum_{j,k=1}^{N} c_j \left(\int f_j''(x) f_k''(x) \, dx \right) c_k = c^T G c, \end{aligned}$$

wobei

$$G = \left[\int f_j''(x) f_k''(x) \, dx \;:\; \begin{array}{l} j = 1, \ldots, N \\ k = 1, \ldots, N \end{array} \right]$$

eine symmetrische, positiv semidefinite Matrix ist.

Übung 10.8 Zeigen Sie die positive Semindefinitheit von G und geben Sie an, unter welchen Voraussetzungen G einen nichttrivialen Kern hat. ♢

Da außerdem wie gehabt

$$\|Fc - y\|_2^2 = (Fc - y)^T (Fc - y) = c^T F^T F c - 2 y^T F c + y^T y$$

ist, können wir also unser Minimierungsproblem in

$$\min_{c \in \mathbb{R}^M} c^T \left(F^T F + \lambda G \right) c - 2 y^T F c + y^T y \tag{10.23}$$

umschreiben, was wir nur noch nach c differenzieren und gleich Null setzen müssen, um zu den entsprechenden Normalengleichungen

$$\left(F^T F + \lambda G \right) c = F^T y \tag{10.24}$$

zu kommen.

Sehen wir uns zum Abschluss[38] wenigstens einmal ein Beispiel an, und zwar in Abb. 10.5. Das Ergebnis zeigt recht deutlich das Verhalten dieser Approximanten: Für $\lambda \to 0$ versuchen sie, so nahe wie möglich an den Daten zu bleiben, wenn möglich zu interpolieren, für $\lambda \to \infty$ hingegen wird die Funktion so glatt wie möglich und findet nun die "Datentreue" nicht mehr so wichtig.

Mehr Informationen über solche Smoothing Splines und auch einige algorithmische Aspekte von deren Berechnung finden sich beispielsweise in [40].

[37] Warum da wohl schon wieder diese Quadrate stehen?

[38] Eigentlich beginnt die Geschichte jetzt erst richtig, aber (leider) nicht im Rahmen dieser Vorlesung – *And now I must stop saying what I am not writing about, because there's nothing so special about that; every story one chooses to tell is a kind of censorship, it prevents the telling of other tales* ... [34]

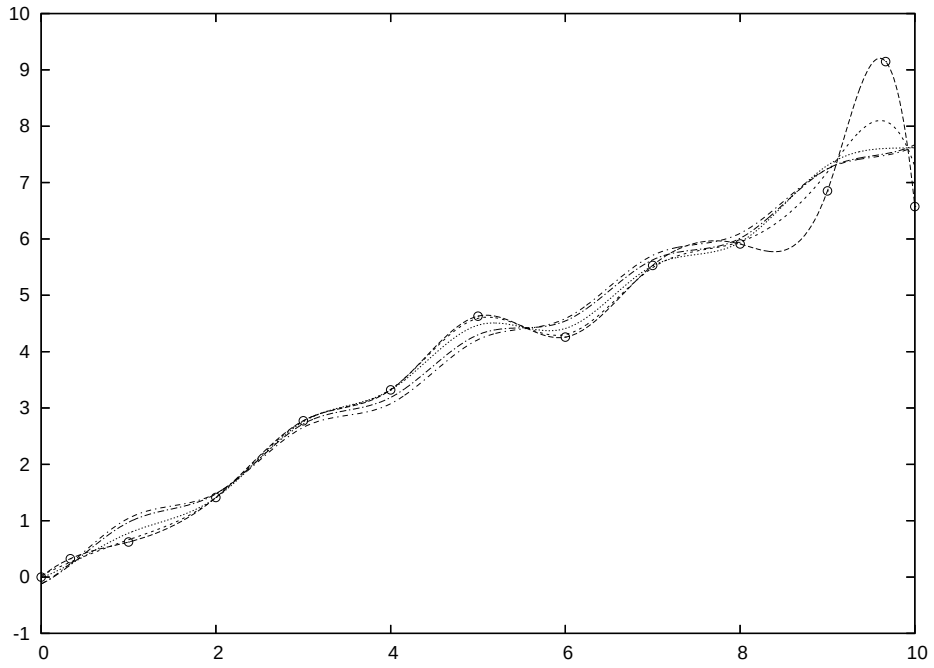


Abbildung 10.5: Smoothing splines für die Parameterwerte 0.01, 0.1, 1 und 10. Man sieht sehr schön, wie es die Funktion mehr und mehr aufgibt, den "Ausreißern" zu folgen und sich mehr und mehr einer Geraden annähert, was ja auch der Grenzwert für $\lambda \to \infty$ ist.

Die Wissenschaft fängt eigentlich erst da an, interessant zu werden, wo sie aufhört

Justus von Liebig

11 Eigenwertprobleme

Im letzten Kapitel über numerische Lineare Algebra befassen wir uns mit der "Königsdisziplin", der Bestimmung der Eigenwerte (und vielleicht auch Eigenvektoren) einer *quadratischen* Matrix $A \in \mathbb{R}^{n\times n}$, das heißt von Werten $\lambda \in \mathbb{C}$ und Vektoren $x \in \mathbb{C}^n$, so daß

$$Ax = \lambda x. \tag{11.1}$$

Definition 11.1 *Eine Zahl $\lambda \in \mathbb{C}$ heißt* Eigenwert *einer Matrix $A \in \mathbb{R}^{n\times n}$, wenn es einen* Eigenvektor $x \in \mathbb{C}^n \setminus \{0\}$ *gibt, so daß $Ax = \lambda x$. Die Menge aller Eigenwerte von A, in Zeichen $\lambda(A)$, bezeichnet man als das* Spektrum[39] *von$A$.*.

Bemerkung 11.2 *(Eigenwerte & Eigenvektoren)*

1. *Die Forderung $x \neq 0$ ist entscheidend, denn der Nullvektor ist ja ein "Eigenvektor" zu jedem "Eigenwert".*

2. *Ist die Matrix A reell, so können[40] einige der Eigenwerte komplex sein. Diese treten aber immer in konjugiert komplexen Paaren auf: Ist also $\lambda = \alpha + i\beta$ ein Eigenwert von A, so ist auch $\overline{\lambda} = \alpha - i\beta$ ein Eigenwert von A.*

11.1 Eigenwerte und Nullstellen von Polynomen

Daß man jedes Eigenwertproblem auf die Nullstellensuche bei Polynomen zurückführen kann, ist wohlbekannt: Die Eigenwerte einer Matrix $A \in \mathbb{R}^{n\times n}$ sind gerade die Nullstellen des *charakteristischen Polynoms*

$$p_A(\lambda) = \det(A - \lambda I), \qquad \lambda \in \mathbb{C},$$

denn die Determinante ist genau dann Null, wenn es eine nichttriviale Lösung des homogenen Gleichungssystems $(A - \lambda I)\, x = 0$ gibt – und dann ist λ ein Eigenwert.

Übung 11.1 Zeigen Sie, daß das charakteristische Polynom einer reellen $n \times n$ Matrix immer ein Polynom vom Grad n ist.

[39] Diese Terminologie erklärt letztendlich auch den Begriff des *Spektralradius*.

[40] Leider, denn das ist ein Problem.

Eine Methode zur Lösung des Eigenwertproblems könnte also darin bestehen, das charakteristische Polynom zu berechnen und dann dessen Nullstellen zu bestimmen – für letztere Aufgabe werden wir noch Verfahren kennenlernen. Natürlich ist die Berechnung einer Determinante *im allgemeinen* eine aufwendige und instabile Angelegenheit, es sei denn, man transformiert die Matrix passend.

Was weniger bekannt ist, ist die Tatsache, daß man auch das Auffinden der Nullstellen eines Polynoms als Eigenwertproblem formulieren kann – dieser Ansatz wird neuerdings auch zur Berechnung *gemeinsamer* Nullstellen multivariater Polynome verwendet [31].

Sei $p \in \Pi_n$ ein reelles Polynom, das wir mittels seiner (möglicherweise komplexen) Nullstellen $z_1, \ldots, z_n$ schreiben können als

$$p(x) = a_n x^n + \cdots + a_0 = a_n \, (x - z_1) \cdots (x - z_n) \,, \qquad 0 \neq a_n \in \mathbb{R}.$$

Auf dem Vektorraum Π_{n-1} ist die "Multiplikation modulo p"

$$\Pi_{n-1} \ni q \mapsto r, \qquad x \, q(x) = \alpha p(x) + r(x), \quad r \in \Pi_n, \tag{11.2}$$

eine *lineare Abbildung* und wird, da

$$x^n = \frac{1}{a_n} p(x) - \sum_{j=0}^{n-1} \frac{a_j}{a_n} \, x^j, \qquad x \in \mathbb{R},$$

bezüglich der Basis $1, x, \ldots, x^{n-1}$ durch die $n \times n$–Matrix

$$M = \begin{bmatrix} 0 & & & & -\frac{a_0}{a_n} \\ 1 & 0 & & & -\frac{a_1}{a_n} \\ & \ddots & \ddots & & \vdots \\ & & 1 & 0 & -\frac{a_{n-2}}{a_n} \\ & & & 1 & -\frac{a_{n-1}}{a_n} \end{bmatrix}, \tag{11.3}$$

die sogenannte *Frobenius–Begleitmatrix* zu p dargestellt. Seien nun die Vektoren

$$v_j = (v_{jk} \ : \ k = 1, \ldots, n) \in \mathbb{C}^n, \qquad j = 1, \ldots, n,$$

so gewählt, daß

$$\ell_j(x) = \frac{p(x)}{x - z_j} = a_n \prod_{k \neq j} (x - z_k) = \sum_{k=1}^{n} v_{jk} \, x^{k-1}, \qquad j = 1, \ldots, n,$$

dann ist

$$\sum_{k=1}^{n} \underbrace{(Mv_j - z_j v_j)_k}_{\in \mathbb{C}} x^{k-1} = x\ell_j(x) - z_j \ell_j(x) = (x - z_j) \, \ell_j(x) = p(x) \simeq 0,$$

und damit ist $Mv_j = z_j v_j$, $j = 1, \ldots, n$.

Die Eigenwerte von M sind also gerade die Nullstellen von p.

Die Frobenius–Matrix aus (11.3) ist aber nur *eine* Möglichkeit von vielen, die "Multiplikation" aus (11.2) darzustellen; jede andere Basis von Π_{n-1} liefert eine andere Matrix M, deren Eigenwerte aber trotzdem wieder die Nullstellen von p sind. Das einzige Hilfsmittel zur Manipulation von Polynomen, das wir brauchen, ist die Durchführung einer "Division mit Rest"!

11.2 Terminologie und die Schur–Zerlegungen

Wie das Beispiel

$$A = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}, \qquad p_A(\lambda) = \lambda^2 + 1 = (\lambda + i)\,(\lambda - i)\,,$$

zeigt, können reelle Matrizen ohne weiteres auch komplexe Eigenwerte haben. Daher ist es (zumindest für die Theorie) vorteilhaft, sich gleich mit *komplexen* Matrizen $A \in \mathbb{C}^{n\times n}$ zu beschäftigen.

Definition 11.3 *Zwei Matrizen $A, B \in \mathbb{C}^{n\times n}$ heißen* ähnlich*, wenn es eine invertierbare Matrix $T \in \mathbb{C}^{n\times n}$ gibt, so daß*

$$A = TBT^{-1}.$$

Lemma 11.4 *Sind $A, B \in \mathbb{C}^{n\times n}$ zueinander ähnlich, so haben sie dieselben Eigenwerte.*

Beweis: Sei $\lambda \in \mathbb{C}$ ein Eigenwert von $A = TBT^{-1}$ und $x \in \mathbb{C}^n$ ein zugehöriger Eigenvektor. Dann ist

$$B\left(T^{-1}x\right) = T^{-1}ATT^{-1}x = T^{-1}Ax = \lambda T^{-1}x$$

und damit ist λ auch ein Eigenwert von B. □

Aus der Linearen Algebra sollte man den folgenden, absolut nichttrivialen Satz kennen.

Satz 11.5 *(Jordansche Normalform)*
Jede Matrix $A \in \mathbb{C}^{n\times n}$ ist ähnlich zu einer Matrix

$$J = \begin{bmatrix} J_1 & & \\ & \ddots & \\ & & J_k \end{bmatrix}, \qquad J_\ell = \begin{bmatrix} \lambda_\ell & 1 & & \\ & \ddots & \ddots & \\ & & \ddots & 1 \\ & & & \lambda_\ell \end{bmatrix} \in \mathbb{C}^{n_\ell \times n_\ell}, \qquad \sum_{\ell=1}^{k} n_\ell = n,$$

die man die Jordan–Normalform *von A nennt.*

Bemerkung 11.6 *Ein Jordanblock entspricht einem* invarianten Teilraum $\mathscr{J}_\ell \subseteq \mathbb{R}^n$ *der Matrix A, also einem Raum, der $A\,\mathscr{J}_\ell = \mathscr{J}_\ell$ erfüllt. Aber es ist sogar noch ein wenig besser: Betrachten wir nur den Jordanblock, dann ist der Einheitsvektor e_{n_ℓ} ein Eigenvektor zum Eigenwert λ und alle anderen Einheitsvektoren sind "fast" Eigenvektoren, sie erfüllen*

$$J_\ell e_k = \lambda e_k + e_{k+1}, \qquad k = 1, \ldots, n_\ell - 1.$$

Genaueres zur Jordan–Normalform findet sich in den "einschlägigen Werken" zur linearen Algebra.

Definition 11.7 *Eine Matrix heißt* diagonalisierbar*, wenn alle Jordanblöcke J_ℓ die Größe* 1 *haben, das heißt, wenn $n_\ell = 1$, $\ell = 1, \ldots, n$. Eine Matrix heißt* nichtderogatorisch*, wenn es zu jedem Eigenwert λ_ℓ genau einen Jordanblock J_ℓ gibt, in dem er als Diagonalwert vorkommt. Offensichtlich nennt man eine Matrix dann auch* derogatorisch*, wenn sie nicht nichtderogatorisch*[41] *ist.*

Bemerkung 11.8 *Hat eine Matrix $A \in \mathbb{R}^{n\times n}$ "nur" n einfache Eigenwerte, dann ist sie sowohl diagonalisierbar als auch nichtderogatorisch und numerisch am besten zu behandeln.*

Eine derogatorische Matrix hingegen hat mindestens einen Eigenwert mit Vielfachheit > 1 und zwar sowohl mit geometrischer[42] *als auch algebraischer*[43] *Vielfachheit. Einen Jordanblock und den zugehörigen invarianten Raum könnte man ja vielleicht nocht ertragen und tolerieren* ...

Und jetzt zur Abwechslung mal wieder eine Matrixzerlegung, die *Schur–Zerlegung*.

Satz 11.9 *Zu jeder Matrix $A \in \mathbb{C}^{n\times n}$ gibt es eine* unitäre *Matrix $U \in \mathbb{C}^{n\times n}$ und eine Rechtsdreiecksmatrix*

$$R = \begin{bmatrix} \lambda_1 & * & \ldots & * \\ & \ddots & \ddots & \vdots \\ & & \ddots & * \\ & & & \lambda_n \end{bmatrix} \in \mathbb{C}^{n\times n},$$

so daß $A = URU^H$.

Bemerkung 11.10 *(Schur–Zerlegung)*

1. *Die Diagonalelemente von R sind natürlich die Eigenwerte von A. Da die beiden Matrizen ähnlich sind, haben sie schließlich dieselben Eigenwerte.*
2. *Zwischen A und R besteht eine "verschärfte" Form der Ähnlichkeit: Die beiden Matrizen sind* unitär *ähnlich.*

Bevor wir die Existenz der Schur–Zerlegung beweisen, zuerst ein paar unmittelbare Folgerungen daraus.

Korollar 11.11 *Zu jeder* hermiteschen[44] *Matrix $A \in \mathbb{C}^{n\times n}$ gibt es eine orthogonale Matrix $U \in \mathbb{C}^{n\times n}$, so daß*

$$A = U \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{bmatrix} U^H, \qquad \lambda_j \in \mathbb{R}, \quad j = 1, \ldots, n.$$

[41] Also eine doppelt negative Definition.
[42] Also Dimension des Eigenraums.
[43] Nullstelle des charakteristischen Polynoms.
[44] Das heißt $A^H = A$.

Beweis: Die Matrix R aus Satz 11.9 erfüllt ja $R = U^H A U$. Nun ist aber

$$R^H = \left(U^H A U\right)^H = U^H A^H U = U^H A U = R,$$

also muß R eine Diagonalmatrix mit reellen Diagonalelementen sein. □

Anders gesagt bedeutet Korollar 11.11, daß jede hermitesche Matrix unitär diagonalisierbar ist, womit sie auch eine Orthonormalbasis von Eigenvektoren besitzt. Als "Extra" sind bei hermiteschen Matrizen obendrein auch noch alle Eigenwerte reell. Soviel kann man zwar im allgemeinen nicht erwarten, aber die Frage, welche Matrizen unitär diagonalisierbar sind, ist doch interessant, zumindest theoretisch. Hier ist die Antwort.

Satz 11.12 *Eine Matrix $A \in \mathbb{C}^{n\times n}$ ist genau dann unitär diagonalisierbar, das heißt, es es gibt eine unitäre Matrix $U \in \mathbb{C}^{n\times n}$ so daß*

$$A = U \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{bmatrix} U^H, \qquad \lambda_j \in \mathbb{C}, \quad j = 1, \ldots, n, \tag{11.4}$$

wenn die Matrix A normal *ist, das heißt, wenn*

$$AA^H = A^H A. \tag{11.5}$$

Bemerkung 11.13 *Normalität einer Matrix bedeutet also nichts anderes als daß die Matrix mit ihrer Hermiteschen kommutiert, und das ist, wie jede Kommutativitätsforderung bei Matrixprodukten, eine sehr* starke *Forderung! Im "Normalfall" weigern sich Matrizen standhaft, zu kommutieren und $AB = BA$ tritt nur unter in sehr besonderen Fällen ein, was dann auch zu weitreichenden Konsequenzen führt. Beispielsweise haben kommutierende Matrizen dieselben Eigenvektoren und wenn man die eine diagonalisieren kann, dann kann man auch die andere diagonalisieren und zwar mit* derselben *Matrix, siehe z.B. [9, 30].*

Übung 11.2 Zeigen Sie: Das Produkt zweier symmetrischer bzw. hermitescher $n \times n$ Matrizen ist genau dann symmetrisch, wenn die beiden Matrizen kommutieren. ♢

Beweis von Satz 11.12: Wir setzen $\Lambda = \text{diag}\,(\lambda_1, \ldots, \lambda_n)$. Hat, nach (11.4), A die Form $A = U\Lambda U^H$, so ist

$$AA^H = U\Lambda U^H U\Lambda^H U^H = U\left|\Lambda\right|^2 U^H \qquad \text{und} \qquad A^H A = U\Lambda^H U^H U\Lambda U^H = U\left|\Lambda\right|^2 U^H,$$

also gilt (11.5). Für die Umkehrung verwenden wir wieder die Schur–Zerlegung von A, sei also $R = U^H A U$. Dann ist, unter der Voraussetzung (11.5)

$$R^H R = U^H A^H U U^H A U = U^H A^H A U = U^H A A^H U = U^H A U U^H A^H U = R R^H,$$

also ist auch R normal. Dann ist aber

$$\left|\lambda_1\right|^2 = \left(R^H R\right)_{11} = \left(R R^H\right)_{11} = \left|\lambda_1\right|^2 + \sum_{k=2}^{n} \left|r_{1k}\right|^2,$$

also ist $r_{12} = \cdots = r_{1n} = 0$. Induktiv sieht man außerdem, daß, für $j = 2, \ldots, n$,

$$\left(R^H R\right)_{jj} = |\lambda_j|^2 + \underbrace{\sum_{k=1}^{j-1} |r_{kj}|^2}_{=0} = \left(RR^H\right)_{jj} = |\lambda_j|^2 + \sum_{k=j+1}^{n} |r_{jk}|^2 ,$$

weswegen R eine Diagonalmatrix sein muß. □

Fehlt noch was? Genau, der Beweis für die Schur–Zerlegung!

Beweis von Satz 11.9: Die Grundidee des Beweises ist ganz ähnlich wie beim Beweis der SVD und verwendet Induktion über n, wobei der Fall $n = 1$ trivial ist. Nehmen wir also an, Satz 11.9 gelte für ein $n \in \mathbb{N}$ und sei $A \in \mathbb{C}^{n+1\times n+1}$. Sei $\lambda \in \mathbb{C}$ irgendein Eigenwert von A und $x \in \mathbb{C}^{n+1}$, $\|x\|_2 = 1$, ein dazugehöriger Eigenvektor. Wir setzen $u_1 = x$ und wählen[45] $u_2, \ldots, u_{n+1}$ so, daß $u_1, \ldots, u_{n+1}$ eine Orthonormalbasis des $\mathbb{C}^{n+1}$ bilden, oder, äquivalent, daß die Matrix $U = [u_1 \ldots u_{n+1}]$ unitär ist. Nun ist aber

$$U^H A U e_1 = U^H A u_1 = U^H A x = \lambda U^H x = \lambda e_1,$$

das heißt, es ist

$$U^H A U = \begin{bmatrix} \lambda_1 & * \\ 0 & B \end{bmatrix}, \qquad B = \mathbb{C}^{n\times n}.$$

Nach Induktionsannahme gibt es eine unitäre Matrix $V \in \mathbb{C}^{n\times n}$, so daß $B = VSV^H$ mit einer Rechtsdreiecksmatrix $S \in \mathbb{C}^{n\times n}$ ist. Daher ist

$$A = U \begin{bmatrix} \lambda_1 & * \\ 0 & VSV^H \end{bmatrix} U^H = \underbrace{U \begin{bmatrix} 1 & 0 \\ 0 & V \end{bmatrix}}_{=:U} \underbrace{\begin{bmatrix} \lambda_1 & * \\ 0 & S \end{bmatrix}}_{=:R} \underbrace{\begin{bmatrix} 1 & 0 \\ 0 & V^H \end{bmatrix} U^H}_{=U^H}$$

und der Beweis ist fertig. □

Für reelle Matrizen sieht die *reelle* Schur–Zerlegung ein bißchen komplizierter aus.

Satz 11.14 *Zu jeder Matrix $A \in \mathbb{R}^{n\times n}$ gibt es eine* orthogonale *Matrix $Q \in \mathbb{R}^{n\times n}$, so daß*

$$A = Q \begin{bmatrix} R_1 & * & \cdots & * \\ & \ddots & \ddots & \vdots \\ & & \ddots & * \\ & & & R_k \end{bmatrix} Q^T, \tag{11.6}$$

wobei entweder $R_j \in \mathbb{R}^{1\times 1}$ oder $R_j \in \mathbb{R}^{2\times 2}$ mit zwei konjugiert komplexen Eigenwerten, $j = 1, \ldots, k$.

[45] Hier ist der Bezug zum Beweis von Satz 10.13.

Die reelle Schur–Zerlegung transformiert also A in eine *obere Hessenbergmatrix*[46]

$$U^T A U = \begin{bmatrix} * & \dots & \dots & * \\ * & \ddots & & \vdots \\ & \ddots & \ddots & \vdots \\ & & * & * \end{bmatrix}$$

Beweis: Hat die Matrix A nur reelle Eigenwerte, dann kann man den Beweis der komplexen Schur–Zerlegung wortwörtlich kopieren. Sei, andernfalls $\lambda = \alpha + i\beta$, $\beta \neq 0$, ein komplexer Eigenwert von A und $x + iy$ der zugehörige Eigenvektor. Dann ist

$$A\ (x + iy) = \lambda\,(x + iy) = (\alpha + i\beta)\,(x + iy) = (\alpha x - \beta y) + i\,(\beta x + \alpha y)\,,$$

oder, in Matrixschreibweise,

$$A \underbrace{[x\ y]}_{\in \mathbb{R}^{n\times 2}} = [x\ y] \underbrace{\begin{bmatrix} \alpha & \beta \\ -\beta & \alpha \end{bmatrix}}_{=:R}.$$

Da $\det R = \alpha^2 + \beta^2 > 0$ weil $\beta \neq 0$, ist span $\{x, y\}$ ein zweidimensionaler A–invarianter Teilraum des $\mathbb{R}^n$, zumindest wenn x und y linear unabhängig sind. Dann wählen wir eben u_1, u_2 als Orthonormalbasis für diesen Teilraum, ergänzen mit $u_3, \dots, u_n$ zu einer Orthonormalbasis des $\mathbb{R}^n$ und erhalten wieder ganz analog wie oben, daß[47]

$$U^T A U = \begin{bmatrix} R & * \\ 0 & B \end{bmatrix}$$

und die Induktion läuft so weiter wie bei der (komplexen) Schur–Zerlegung.

Bleibt also nur noch der Fall, daß $y = \gamma x$, $\gamma \in \mathbb{R}$. Dann wäre aber $A(1+i\gamma)x = \lambda(1+i\gamma)x$, also, da $0 \neq 1 + i\gamma$, ergäbe sich $Ax = \lambda x$ und die linke Seite davon ist reell, die rechte aber nicht[48]. □

11.3 Vektoriteration

Die *Vektoriteration* (oder auch *Potenzmethode*, englisch *Power method*) ist die einfachste Methode, unter bestimmten Voraussetzungen einen[49] Eigenwert und sogar den zugehörigen Eigenvektor zu bestimmen. Dabei bildet man, ausgehend von einem Vektor $y^{(0)} \in \mathbb{C}^n$ die Folge $y^{(k)}$, $k \in \mathbb{N}$, mittels der Iteration[50]

$$z^{(k)} \;=\; A y^{(k-1)},$$

[46] Ein Konzept, mit dem wir uns später noch eingehender befassen werden.

[47] Die Matrix U ist jetzt reell und damit "nur" orthogonal und nicht unitär.

[48] Das führt zu dem netten Korollar, daß bei jedem komplexen Eigenvektor einer reellen Matrix Real- und Imaginärteil linear unabhängig sein müssen, was einfach zu zeigen aber nur selten zu finden ist.

[49] ... ganz bestimmten.

[50] **Achtung:** Auch wenn die zweite Zeile der Vorschrift schrecklich kompliziert aussieht, so ist es doch lediglich eine *Normierung*, bei der die euklidische Norm des Vektor auf 1 gebracht und das Vorzeichen der ersten *signifikanten* Komponente des Vektors auf "+" fxiert wird.

$$y^{(k)} \;=\; \frac{z^{(k)}}{z^{(k)}_{j^*}}\,\frac{\left|z^{(k)}_{j^*}\right|}{\left\|z^{(k)}\right\|_\infty}, \qquad j^* = \min\left\{1 \le j \le n \;:\; \left|z^{(k)}_j\right| \ge \left(1-\frac{1}{k}\right)\left\|z^{(k)}\right\|_\infty\right\}, \tag{11.7}$$

und stellt fest, daß diese Folge unter bestimmten Umständen gegen den *dominanten Eigenvektor* konvergiert.

Proposition 11.15 *Es sei $A \in \mathbb{C}^{n\times n}$ eine diagonalisierbare Matrix, deren Eigenwerte $\lambda_1, \ldots, \lambda_n$ die Bedingung*

$$|\lambda_1| > |\lambda_2| \ge \cdots \ge |\lambda_n|$$

erfüllen[51]*. Dann konvergiert die Folge $y^{(k)}$, $k \in \mathbb{N}$, für fast alle Startwerte $y^{(0)}$ gegen ein Vielfaches des normierten Eigenvektors x_1 zum Eigenwert λ_1.*

Beweis: Es seien $x_1, \ldots, x_n$ die zu 1 normierten[52], linear unabhängigen Eigenvektoren von A zu den Eigenwerten $\lambda_1, \ldots, \lambda_n$ – ihre Existenz folgt aus der Diagonalisierbarkeit von A. Wir schreiben

$$y^{(0)} = \sum_{j=1}^n \alpha_j x_j, \qquad \alpha_j \in \mathbb{C}, \quad j = 1, \ldots, n,$$

und stellen fest, daß

$$A^k y^{(0)} = \sum_{j=1}^n \alpha_j\, A^k x_j = \sum_{j=1}^n \alpha_j\, \lambda_j^k\, x_j = \lambda_1^k \sum_{j=1}^n \alpha_j\, \left(\frac{\lambda_j}{\lambda_1}\right)^k\, x_j.$$

Daraus folgt wegen $|\lambda_1| > |\lambda_j|$, $j = 2, \ldots, n$, daß

$$\lim_{k\to\infty} \lambda_1^{-k} A^k y^{(0)} = \alpha_1 x_1 + \lim_{k\to\infty} \sum_{j=2}^n \alpha_j\, \underbrace{\left(\frac{\lambda_j}{\lambda_1}\right)^k}_{\to 0}\, x_j = \alpha_1 x_1,$$

sowie

$$\lim_{k\to\infty} |\lambda_1|^{-k} \left\|A^k y^{(0)}\right\|_\infty = \lim_{k\to\infty} \left\|\sum_{j=1}^n \alpha_j \left(\frac{\lambda_j}{\lambda_1}\right)^k\, \alpha_j\, x_j\right\|_\infty = |\alpha_1|\,\|x_1\|_\infty$$

Ist $\alpha_1 = 0$, gehört also $y^{(0)}$ zur Hyperebene

$$x_1^\perp = \left\{x \in \mathbb{C}^n \;:\; x^H x_1 = 0\right\},$$

dann haben beide Grenzwerte den Wert Null und man kann über die Konvergenz der Folge $y^{(k)}$, $k \in \mathbb{N}$, nicht viel sagen[53]; diese Hyperebene ist aber nur eine Menge vom Maß 0, so daß wir im weiteren annehmen, daß $\alpha_1 \neq 0$.

[51] Die Aussage gilt auch noch, wenn der erste Eigenwert ein mehrfacher Eigenwert ist, siehe [46, S. 44]. Problematisch wird es aber, wenn es mehrere *verschiedene* Eigenwerte gibt, die alle maximalen Betrag haben.

[52] Das macht sie zugegebenermaßen noch **nicht** eindeutig, jeder von den Eigenvektoren kann noch mit einem Vorzeichen versehen werden, bei mehrfachen Eigenwerten sind sogar Linearkombinationen möglich. Aber wollen wir uns wirklich wegen solcher Kleinigkeiten Gedanken machen?

[53] Außer daß es mit sehr hoher Wahrscheinlichkeit numerischen Müll geben wird.

Wegen (11.7) ist $y^{(k)} = \gamma_k A^k y^{(0)}$, $\gamma_k \in \mathbb{C}$, und da außerdem $\left\|y^{(k)}\right\|_\infty = 1$, ist also

$$\lim_{k\to\infty} |\lambda_1|^k |\gamma_k| = \lim_{k\to\infty} \frac{1}{|\lambda_1|^{-k} \|A^k y^{(0)}\|_\infty} = \frac{1}{|\alpha_1| \|x_1\|_\infty}.$$

Damit ist

$$y^{(k)} = \gamma_k A^k y^{(0)} = \underbrace{\frac{\gamma_k \lambda_1^k}{|\gamma_k \lambda_1^k|}}_{=:e^{-2\pi i\theta_k}} \underbrace{\frac{\alpha_1 x_1}{|\alpha_1| \ \|x_1\|_\infty}}_{=:\alpha x_1} + \underbrace{O\left(\frac{|\lambda_2|^k}{|\lambda_1|^k}\right)}_{\to 0}, \qquad k \in \mathbb{N}, \tag{11.8}$$

wobei $\theta_k \in [0, 1)$. Hier kommt nun die etwas seltsame Normierung aus (11.7) ins Spiel: Sei j der *minimale* Index, so daß $\left|(\alpha x_1)_j\right| = \|\alpha x_1\|_\infty$, dann ist nach (11.8) für hinreichend großes k auch $j^* = j$ in (11.7). Also ist

$$\lim_{k\to\infty} y_j^{(k)} = 1 \qquad \Longrightarrow \qquad \lim_{k\to\infty} e^{2\pi i\theta_k} = \lim_{k\to\infty} \frac{y_j^{(k)}}{(\alpha\ x_1)_j} = \frac{1}{(\alpha\ x_1)_j}.$$

Eingesetzt in (11.8) liefert das die Konvergenz der Folge $y^{(k)}$, $k \in \mathbb{N}$. □

Übung 11.3 Programmieren Sie die Vektoriteration in `Matlab` oder `Octave`.

Man könnte die Vektoriteration auch iterieren, um *alle* Eigenwerte und Eigenvektoren zu finden, sofern alle Eigenwerte von A betragsmäßig verschieden sind. Dazu bestimmt man den betragsgrößten Eigenwert λ_1 von A und den zugehörigen Eigenvektor x_1 und fährt mit

$$A^{(1)} = A - \lambda_1\ x_1 x_1^T$$

fort. Die diagonalisierbare Matrix $A^{(1)}$ hat dieselben orhtonormalen Eigenvektoren wie A, nur ist jetzt x_1 ein Eigenvektor zum Eigenwert 0 und spielt daher bei der Vektoriteration keine Rolle mehr, sofern man nicht gerade mit einem Vielfachen von x_1 startet. Eine Anwendung der Vektoriteration auf $A^{(1)}$ liefert dann den betragszweitgrößten[54] Eigenwert λ_2 und den zugehörigen Eigenvektor und die Iteration

$$A^{(j)} = A^{(j-1)} - \lambda_j\ x_j x_j^T, \qquad j = 1, \ldots, n, \qquad A^{(0)} = A,$$

berechnet sukzessive alle Eigenwerte und Eigenvektoren von A, immer vorausgesetzt, die Eigenwerte sind betragsmäßig unterschiedlich.

Bemerkung 11.16 *(Nachteile der Vektoriteration)*

1. *Die Methode funktioniert im allgemeinen nur, wenn es einen* "dominanten" *Eigenvektor gibt, das heißt, wenn es zum betragsmäßig größten Eigenwert genau einen Eigenvektor gibt. Betrachtet man zum Beispiel*

$$A = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix},$$

dann bildet diese einen Vektor $[x_1\ x_2]^T$ auf den Vektor $[x_2\ x_1]^T$ ab und das konvergiert genau dann, wenn man die Iteration mit einem der Eigenvektoren startet.

[54] Ein wunderschönes Wort.

2. *Die Methode funktioniert nur mit "geeigneten" Startvektoren. Es klingt natürlich toll, daß alle Startvektoren bis auf eine Hyperebene geeignet sind, aber so einfach ist das nicht. Ist nämlich der betragsgrößte Eigenwert einer reellen Matrix komplex, dann kann man mit reellen Startwerten iterieren, bis man schwarz wird – den (ebenfalls komplexen) Eigenvektor wird man nie finden.*

3. *Man müßte also immer komplex rechnen, was den Rechenaufwand ganz ordentlich erhöht: Bei der Addition braucht man doppelt so viele Operationen wie im Reellen, bei der Multiplikation sogar sechsmal so viele.*

4. *Die Konvergenzgeschwindigkeit hängt vom Verhältnis*

$$\frac{|\lambda_2|}{|\lambda_1|} < 1$$

ab, was aber beliebig nahe bei 1 *liegen kann. Ist die Dominanz des dominanten Eigenvektors also nicht allzu ausgeprägt, dann kann die Konvergenz extrem langsam werden.*

Alles in allem ist die Vektoriteration also leider keine allzu gute Methode, um Eigenwertprobleme zu lösen.

11.4 Das QR–Verfahren I – Theorie

Das praktische Verfahren zur Behandlung von Eigenwertproblemen ist heutzutage wohl das QR–Verfahren, das auf Francis [7, 8] zurückgeht und eine (unitäre) Erweiterung des LR–Verfahrens[55] von Rutishauser [35] darstellt. Wir wollen uns zuerst einmal auf den *komplexen* Fall beschränken, denn dann haben wir die "schöne" Schur–Zerlegung zur Verfügung.

Das Verfahren als solches ist eine erstaunlich einfache Iteration: Man startet mit $A^{(0)} = A$ und berechnet iterativ, mit Hilfe der QR–Zerlegungen,

$$A^{(k)} = Q_k\, R_k, \qquad A^{(k+1)} = R_k\, Q_k, \qquad k \in \mathbb{N}_0. \tag{11.9}$$

Mit ein bißchen Glück[56] konvergiert diese Folge gegen eine obere Dreiecksmatrix, deren Diagonalelemente die Eigenwerte von A sind. Daß diese Diagonalelemente tatsächlich Eigenwerte sein müssen, ist noch sehr einfach zu sehen.

Lemma 11.17 *Die in (11.9) konstruierten Matrizen* $A^{(k)}$*,* $k \in \mathbb{N}$*, sind* orthogonal ähnlich *zu* A *und haben daher dieselben Eigenwerte wie* A*.*

Beweis: Es ist

$$A^{(k+1)} = Q_k^T Q_k R_k Q_k = Q_k^T A^{(k)} Q_k = \cdots = \underbrace{Q_k^T \cdots Q_0^T}_{=:U_k^T} A \underbrace{Q_0 \cdots Q_k}_{=:U_k}.$$

[55] Im Moment nur ein Name, wir kommen aber später nochmal drauf zurück.

[56] Oder, wie Mathematiker sagen, "unter gewissen Voraussetzungen".

□

Um die Konvergenz (unter bestimmten Voraussetzungen an A) zeigen zu können, interpretieren wir die QR–Iteration als eine Verallgemeinerung der Vektoriteration (11.7) (ohne die seltsame Normierung) auf Vektorräumen. Dazu schreiben wir die Orthonormalbasis[57] $u_1, \ldots, u_m \in \mathbb{C}^n$ eines m–dimensionalen Vektorraums $U \subset \mathbb{C}^n$, $m \leq n$, als Spaltenvektoren einer unitären Matrix[58] $U \in \mathbb{R}^{n\times m}$ und iterieren die Vektorräume (bzw. Matrizen) über die QR–Zerlegung

$$U_{k+1}R_k = A\,U_k, \qquad k \in \mathbb{N}_0, \qquad U_0 \subseteq \mathbb{C}^n, \tag{11.10}$$

Daraus folgt sofort, daß

$$U_{k+1}\,(R_k \cdots R_0) = A\,U_k\,(R_{k-1}\cdots R_0) = A\,A\,U_{k-1}\,(R_{k-2}\cdots R_0) = \cdots = A^{k+1}U_0. \tag{11.11}$$

Definieren wir, für $m = n$ nun $A^{(k)} = U_k^H A U_k$, dann ist nach (11.10)

$$A^{(k)} = U_k^H A U_k = U_k^H U_{k+1} R_k$$

und

$$A^{(k+1)} = U_{k+1}^H A U_{k+1} = U_{k+1}^H A U_k U_k^H U_{k+1} = R_k U_k^H U_{k+1},$$

und mit $Q_k := U_k^H U_{k+1}$ erhalten wir die Iterationsvorschrift aus (11.9). Als "Startmatrix" können wir hierbei $U_0 = I$ wählen.

Um eine Konvergenzaussage für das QR–Verfahren zu erhalten, brauchen wir erst noch einen weiteren Typ von Matrizen.

Definition 11.18 *Eine* Phasenmatrix $\Theta \in \mathbb{C}^{n\times n}$ *ist eine unitäre Diagonalmatrix, das heißt,* Θ *hat die Form*

$$\Theta = \begin{bmatrix} e^{-i\theta_1} & & \\ & \ddots & \\ & & e^{-i\theta_n} \end{bmatrix}, \qquad \theta_j \in [0, 2\pi), \quad j = 1, \ldots, n.$$

Proposition 11.19 *Die Matrix $A \in \mathbb{C}^{n\times n}$ habe betragsmäßig getrennte Eigenwerte $|\lambda_1| > |\lambda_2| > \cdots > |\lambda_n| > 0$. Besitzt die Matrix X^{-1} in der Jordan–Normalform $A = X\Lambda X^{-1}$ von A eine LU–Zerlegung*

$$X^{-1} = ST, \qquad S = \begin{bmatrix} 1 & & & \\ * & 1 & & \\ \vdots & \ddots & \ddots & \\ * & \ldots & * & 1 \end{bmatrix}, \quad T = \begin{bmatrix} * & \ldots & * \\ & \ddots & \vdots \\ & & * \end{bmatrix},$$

dann gibt es Phasenmatrizen Θ_k, $k \in \mathbb{N}_0$, so daß die Matrizenfolge $\Theta_k U_k$, $k \in \mathbb{N}$, konvergiert.

[57] Das ist **keine** Einschränkung, schließlich kann man ja mit Gram–Schmidt jede Basis eines Vektorraums in eine Orthogonalbasis transformieren, siehe Abschnitt A.1.

[58] Die Identifikation von Unterräumen und Matrizen ist durchaus nicht ungebräuchlich und die "doppeldeutige" Notation durchaus berechtigt.

Bemerkung 11.20 *(Zu Proposition 11.19)*

1. *Die Konvergenz der Folge $\Theta_k U_k$, $k \in \mathbb{N}$, bedeutet insbesondere, daß die zugehörigen Orthonormalbasen gegen eine Orthonormalbasis des $\mathbb{C}^n$ konvergieren, wir haben also "Konvergenz der Vektorräume".*

2. *Die Existenz der LU–Zerlegung von X^{-1} ist keine echte Einschränkung: Da X^{-1} trivialerweise invertierbar ist, gibt es immer eine Permutation P, so daß $X^{-1}P^T = (PX)^{-1} = LU$ ist und PX ist ja auch eine invertierbare Matrix. Das heißt aber auch, daß die Matrix $\widehat{A} = P^T AP$, die aus A durch Vertauschung von Zeilen und Spalten hervorgeht und dieselben Eigenwerte wie A hat, die Voraussetzung von Proposition 11.19 erfüllt.*

3. *Der Beweis von Proposition 11.19 ist eine[59] Modifikation des Beweises aus [46, S. 54–56] für die Konvergenz des LR–Verfahrens, der ursprünglich von Wilkinson [49] stammt. Was ist nun das LR–Verfahren? Nun, genau wie beim QR–Verfahren zerlegt man die Matrix $A^{(k)}$, diesmal wie bei der Gauß–Elimination, in $A^{(k)} = L_k R_k$ und "baut" sie umgekehrt wieder zusammen, also $A^{(k+1)} = R_k L_k$. Auch dieses Verfahren konvergiert unter bestimmten Voraussetzungen gegen eine Rechtsdreiecksmatrix.*

Bevor wir Proposition 11.19 beweisen, schauen wir uns zuerst schnell an, warum die Konvergenz der Folge U_k die Konvergenz des QR–Verfahrens zur Folge hat. Ist nämlich[60] $\|U_{k+1} - U_k\|_2 \leq \varepsilon$ oder eben

$$U_{k+1} = U_k + E, \qquad \|E\|_2 \leq \varepsilon,$$

dann ist

$$Q_k = U_{k+1}^H U_k = (U_k + E)^H U_k = I + E^H U_k = I + F, \qquad \|F\|_2 \leq \|E\|_2 \underbrace{\|U_k\|_2}_{=1} = \varepsilon,$$

und damit ist

$$A^{(k+1)} = R_k Q_k = R_k (I + F) = R_k + G, \qquad \|G\|_2 \leq \varepsilon \|R_k\|_2 ,$$

also konvergiert die Folge $A^{(k)}$, $k \in \mathbb{N}$, gegen eine Rechtsdreiecksmatrix solange nur die Normen der R_k, $k \in \mathbb{N}$, gleichmäßig beschränkt sind. Das ist aber der Fall, da ja

$$\|R_k\|_2 = \left\|Q_k^H A^{(k)}\right\|_2 = \left\|A^{(k)}\right\|_2 = \left\|Q_{k-1}^H \cdots Q_0^H \; A \; Q_0 \cdots Q_{k-1}\right\|_2 = \|A\|_2 .$$

Übung 11.4 Wie sehen die obigen Überlegungen beim Auftreten von Phasenmatrizen aus?

Zuerst brauchen wir noch eine Hilfsaussage über die "Eindeutigkeit" der QR–Zerlegung.

[59] ... mehr oder weniger einfache ...

[60] Die Wahl der 2–Norm an dieser Stelle ist eher willkürlich nichtsdestotrotz aber natürlich, und vor allem aber hilfreich, wie wir gleich sehen werden.

Lemma 11.21 *Seien $U, V \in \mathbb{C}^{n\times n}$ unitäre Matrizen und seien $R, S \in \mathbb{C}^{n\times n}$ invertierbare Rechtsdreiecksmatrizen. Dann gilt $UR = VS$ genau dann, wenn es eine Phasenmatrix*

$$\Theta = \begin{bmatrix} e^{-i\theta_1} & & \\ & \ddots & \\ & & e^{-i\theta_n} \end{bmatrix}, \qquad \theta_j \in [0, 2\pi),\ j = 1, \ldots, n,$$

gibt, so daß $U = V\Theta^H$ und $R = \Theta S$.

Beweis: Da $UR = V\Theta^H\Theta S = VS$, ist die Richtung "$\Longleftarrow$" trivial. Für die Umkehrung folgern wir aus $UR = VS$, daß $V^HU = SR^{-1}$ eine Rechtsdreiecksmatrix sein muß, ebenso wie $\left(V^HU\right)^H = U^HV = RS^{-1}$. Also ist $\Theta = V^HU$ eine unitäre Diagonalmatrix und es ist $U = VV^HU = V\Theta$. □

Übung 11.5 Zeigen Sie: Ist $A \in \mathbb{R}^{n\times n}$, invertierbar, dann ist die (reelle) QR–Zerlegung von A eindeutig, wenn man zusätzlich $r_{jj} \geq 0$ fordert.

Beweis von Proposition 11.19: Es sei $A = X\Lambda X^{-1}$ die Jordan–Normalform von A, wobei $\Lambda = \text{diag}\ (\lambda_1, \ldots, \lambda_n)$. Mit $U_0 = I$ ist also für $k \in \mathbb{N}_0$

$$U_k \left(\prod_{j=k-1}^{0} R_j \right) = \left(X\Lambda X^{-1}\right)^k = X\Lambda^k X^{-1} = X\Lambda^k ST = X \underbrace{\left(\Lambda^k S \Lambda^{-k}\right)}_{=:L_k} \Lambda^k T,$$

wobei L_k eine untere Dreiecksmatrix mit Koeffizienten

$$(L_k)_{jm} = \left(\frac{\lambda_j}{\lambda_m}\right)^k s_{jm}, \qquad 1 \leq m \leq j \leq n, \tag{11.12}$$

ist, so daß für $k \in \mathbb{N}$

$$|L_k - I| \leq \left(\max_{1\leq m<j\leq n} |s_{jm}| \right) \underbrace{\left(\max_{1\leq m<j\leq n} \frac{|\lambda_j|}{|\lambda_m|} \right)^k}_{=:\rho^k,\ \rho<1} \begin{bmatrix} 0 & & & \\ 1 & \ddots & & \\ \vdots & \ddots & \ddots & \\ 1 & \ldots & 1 & 0 \end{bmatrix}, \qquad k \in \mathbb{N}. \tag{11.13}$$

Sei $\widehat{Q}_k\widehat{R}_k = XL_k$ die QR–Zerlegung von XL_k, die wegen (11.13) und Lemma 11.21 bis auf Phasenmatrizen gegen die QR–Zerlegung $X = QR$ von X konvergiert. Wenden wir nun Lemma 11.21 auf die Identität

$$U_k \left(\prod_{j=k-1}^{0} R_j \right) = \widehat{Q}_k\widehat{R}_k\Lambda^k T$$

an, dann gibt es Phasenmatrizen Θ_k, so daß

$$U_k = \widehat{Q}_k\, \Theta_k^H \qquad \text{und} \qquad \left(\prod_{j=k-1}^{0} R_j \right) = \Theta_k\, \widehat{R}_k\Lambda^k T$$

```
%#
%# RQTrans.m
%# RQ-Transformation einer Matrix A
%#

function B = RQTrans( A )
  [Q,R] = HousehQR( A );
  B = R*Q;
%endfunction
```

Programm 11.1 `RQTrans.m`: Ein Schritt für die QR–Iteration

und damit gibt es Phasenmatrizen $\widehat{\Theta}_k$, so daß $U_k\widehat{\Theta}_k \to U$ für $k \to \infty$. □

Es lohnt sich, noch kurz einen Blick auf den "Fehlerterm" aus (11.12) zu werfen, dessen Subdiagonalelemente die ja Bedingung

$$|L_k|_{jm} \leq \left(\frac{|\lambda_j|}{|\lambda_m|}\right)^k |s_{jm}|\,, \qquad 1 \leq m < j \leq n.$$

erfüllen. Also gilt:

Die Konvergenz eines Subdiagonalelements gegen Null ist umso schneller, je weiter das Element von der Diagonalen entfernt ist.

11.5 Das QR–Verfahren II – Praxis

Im letzten Abschnitt haben wir also gesehen, daß das QR–Verfahren unter bestimmten Bedingungen eine Folge von Matrizen $A^{(k)}$ generiert, die gegen eine Rechtsdreiecksmatrix konvergieren sollte, auf deren Diagonale dann die Eigenwerte stehen. Dieses Verfahren können wir natürlich auch "blind" auf reelle Matrizen anwenden. Die Grundlage dafür, die "RQ–Transformation" ist ganz einfach in `Matlab` implementiert, siehe `RQTrans.m`.

Beispiel 11.22 *Sei*

$$A = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 3 \\ 1 & 2 & 1 \end{bmatrix}.$$

Diese Matrix hat die Eigenwerte

$$\lambda_1 \sim 4.56155, \qquad \lambda_2 = -1, \qquad \lambda_3 \sim 0.43845.$$

Iterieren wir nun mit dem QR–Verfahren, dann erhalten wir für die Nebendiagonalelemente die folgenden Ergebnisse:

# Iterationen	a_{21}	a_{31}	a_{32}
10	6.6425×10^{-7}	-2.2601×10^{-9}	3.3995×10^{-3}
20	1.7034×10^{-13}	-1.5221×10^{-19}	8.9354×10^{-7}
30	4.3671×10^{-20}	-1.0244×10^{-29}	2.3458×10^{-10}
40	1.1196×10^{-26}	-6.8949×10^{-40}	6.1583×10^{-14}

Was man schön sieht, ist daß nach k Iterationen das Matrixelement $a_{m\ell}^{(k)}$, $\ell < m$, wie $|\lambda_\ell/\lambda_m|^k$ gegen 0 geht.

Beispiel 11.23 *Die Matrix*

$$A = \begin{bmatrix} 1 & 5 & 7 \\ 3 & 0 & 6 \\ 4 & 3 & 1 \end{bmatrix}$$

hat die Eigenwerte

$$\lambda_1 \sim 9.7407, \qquad \lambda_2 \sim -3.8703 + i\ 0.6480, \qquad \lambda_3 \sim -3.8703 - i\ 0.6480.$$

Damit können wir nicht erwarten, daß das QR–Verfahren gegen eine obere Dreiecksmatrix konvergiert, denn dann wären ja alle Eigenwerte von A reell. Und in der Tat erhalten wir nach 100 *Iterationen die Matrix*

$$A^{(100)} \sim \begin{bmatrix} 9.7407 & -4.3355 & 0.94726 \\ 8.5520 \times 10^{-39} & -4.2645 & 0.72360 \\ 3.3746 \times 10^{-39} & -0.79491 & -3.4762 \end{bmatrix},$$

die uns immerhin den reellen Eigenwert korrekt liefert. Außerdem hat die 2×2–Matrix "rechts unten" die komplexen *Eigenwerte $-3.8703 \pm i\ 0.6480$.*

Das zweite Beispiel legt schon eine Strategie nahe: Wenn Elemente direkt unterhalb der Diagonalen einfach nicht verschwinden wollen, wäre es vielleicht sinnvoll, sich die entsprechende 2×2–Matrix mal genauer anzusehen.

Definition 11.24 *Hat $A \in \mathbb{R}^{n\times n}$ die*[61] *QR–Zerlegung $A = QR$, dann ist die RQ–*Transformierte *A_* zu A definiert als $A_* = RQ$.*

Welche Probleme tauchen nun bei der praktischen Durchführung des QR–Verfahrens auf? Erinnern wir uns zuerst einmal an die Berechnung der QR–Zerlegung, dann fällt folgendes auf:

> *Die Berechnung der QR–Zerlegung einer vollbesetzten $n\times n$–Matrix hat eine Komplexität von $O\left(n^3\right)$.*

[61] Nur zur Erinnerung: Wir sind hier etwas schlampig, denn es gibt mindestens *zwei* QR–Zerlegungen von A. Bei der QR–Zerlegung mit Householder–Matrizen in Lemma 10.22 haben wir diese Mehrdeutigkeit sogar ausgenutzt, um die *numerisch stabilere* QR–Zerlegung zu bestimmen.

Es ist natürlich nicht übermäßig geschickt, ein Iterationsverfahren zu verwenden, das auf so einem aufwendigen Iterationsschritt basiert. Um dieses Problem zu vermeiden, verwenden wir Matrizen, deren QR–Zerlegung schneller berechnet werden kann. Und das sind *obere Hessenbergmatrizen*, denn deren QR–Zerlegung können wir mit n Givens–Iterationen, also mit $O\left(n^2\right)$ Operationen berechnen: Nachdem nur die Elemente $h_{j-1,j}$, $j = 2, \ldots, n$ zu eliminieren sind, bestimmen wir Drehwinkel $\phi_2, \ldots, \phi_n$, so daß

$$J\left(n-1, n\ ;\ \phi_n\right) \cdots J\left(1, 2\ ;\ \phi_2\right) H = R$$

und es ist

$$H_* = R\ J^T\left(1, 2\ ;\ \phi_2\right) \cdots J^T\left(n-1, n\ ;\ \phi_n\right). \tag{11.14}$$

Dieses Verfahren ist in `HessenRQ.m` implementiert, allerdings nicht in einer sehr effektiven Form. Das wird im Rahmen der Vorlesung allerdings nicht so schlimm werden, denn wir werden uns mit Methoden beschäftigen, die im wesentlichen die *Anzahl* der nötigen QR–Schritte reduzieren und nicht mit der[62] Frage, wie man die einzelnen Iterationen tunen kann. Aber es wird noch besser: Nicht nur, daß dieses Verfahren natürlich schneller ist, es gehorcht auch noch dem Motto "Einmal Hessenberg, immer Hessenberg"[63]

Lemma 11.25 *Ist $H \in \mathbb{R}^{n\times n}$ eine obere Hessenbergmatrix, dann ist H_* auch eine obere Hessenbergmatrix.*

Beweis: Folgt direkt aus der Darstellung (11.14). Die Multiplikation einer Jacobi–Matrix $J^T\left(j, j+1\ ;\ \phi_{j+1}\right)$, $j = 1, \ldots, n-1$, von rechts bedeutet eine Kombination der *Spalten* j und $j+1$ und erzeugt höchstens in der ersten Subdiagonale von Null verschiedene Einträge – schließlich war R ja eine Rechtsdreiecksmatrix. □

Bleibt also nur noch das "Problem", die Ausgangsmatrix $A \in \mathbb{R}^{n\times n}$ auf Hessenberg–Gestalt zu bringen. Dazu verwenden wir zur Abwechslung mal wieder unsere Householder–Transformationen. Nehmen wir an, wir hätten[64] bereits eine orthogonale Matrix Q_k gefunden, so daß die ersten k Spalten der Transformierten Hessenberg–Gestalt haben, das heißt, daß

$$Q_k A Q_k^T = \left[\begin{array}{ccc|c|ccc} * & \ldots & * & * & * & \ldots & * \\ * & \ldots & * & * & * & \ldots & * \\ & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ & & * & * & * & \ldots & * \\ \hline & & & a_1^{(k)} & * & \ldots & * \\ & & & \vdots & \vdots & \ddots & \vdots \\ & & & a_{n-k-1}^{(k)} & * & \ldots & * \end{array}\right]$$

[62] Durchaus nicht unwichtigen!

[63] Was auch der Grund ist, warum man Hessenbergmatrizen verwendet.

[64] Nach einigen Schritten ...

```
%#
%# HessenRQ.m
%# Berechne RQ-Transformierte fuer Hessenberg-Matrix mit
%# Givens-Rotationen
%# Daten:
%#   H  Hessenbergmatrix

function HH = HessenRQ( H )
  n = length( H );
  Q = eye( n );

  for k = 2:n
    a = H( k-1:k,k-1 );                    % relevanter Vektor
    an = sqrt( a'*a );                     % 2-Norm
    c = sign( a(2) ) * abs( a(1) ) / an;   % cosinus
    s = sign( a(1) ) * abs( a(2) ) / an;   % sinus

    Jm = eye( n );                         %
    Jm( k-1,k-1 ) = Jm( k,k ) = c;         % Jacobi-Matrix
    Jm( k-1,k ) = s; Jm( k,k-1 ) = -s;     %

    H = Jm * H;
    Q = Q * Jm';
  end

  HH = H*Q;
%endfunction
```

Programm 11.2 `HessenRQ.m`: Berechnung der RQ–Transformierten einer oberen Hessenberg–Matrix nach (11.14). Durch die Matrixmultiplikationen mit `Jm` ist das Verfahren aber nicht besonders effektiv.

Dann bestimmen wir wieder $\hat{y} \in \mathbb{R}^{n-k-1}$ und $\alpha \in \mathbb{R}$ (was sich ja automatisch ergibt), so daß

$$H\left(\hat{y}\right)\begin{bmatrix} a_1^{(k)} \\ \vdots \\ a_{n-k-1}^{(k)} \end{bmatrix} = \begin{bmatrix} \alpha \\ 0 \\ \vdots \\ 0 \end{bmatrix} \qquad \Longrightarrow \qquad U_{k+1} := \begin{bmatrix} I_{k+1} & \\ & H\left(\hat{y}\right) \end{bmatrix}$$

und erhalten, daß[65]

$$\underbrace{U_{k+1}Q_k}_{=:Q_{k+1}} \; A \; \underbrace{Q_k^T U_{k+1}^T}_{=Q_{k+1}^T} = \left[\begin{array}{ccc|c|ccc} * & \dots & * & * & * & \dots & * \\ * & \dots & * & * & * & \dots & * \\ & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ & & * & * & * & \dots & * \\ \hline & & & \alpha & * & \dots & * \\ & & & 0 & * & \dots & * \\ & & & \vdots & \vdots & \ddots & \vdots \\ & & & 0 & * & \dots & * \end{array}\right] \underbrace{U_{k+1}^T}_{=U_{k+1}}$$

und die Einheitsmatrix I_{k+1} "oben links" in der $n \times n$–Householder–Matrix U_{k+1} sorgt dafür, daß wir nun in den ersten $k+1$ Spalten die Hessenberg–Struktur haben. Die `Octave`–Version dieses Verfahrens ist in `HessenForm.m` implementiert. Fassen wir zusammen: Unser "praktisches" QR–Verfahren verwendet jetzt also die "Zwei–Phasen–Methode"

1. Transformiere A orthogonal auf Hessenberg–Gestalt:

$$H^{(0)} = QAQ^T, \qquad Q^TQ = QQ^T = I.$$

2. Führe die QR–Iterationen

$$H^{(k+1)} = H_*^{(k)}, \qquad k \in \mathbb{N}_0,$$

durch, in der Hoffnung, daß die Nebendiagonalelemente alle gegen Null konvergieren.

Da die Elemente der Subdiagonalen am langsamsten konvergieren, können wir als Abbruchkriterium deren betragsmäßiges Maximum verwenden. Das führt zum ersten, einfachen Eigenwertverfahren, `QRMethode1.m`. Allerdings – beim Auftreten komplexer Eigenwerte iteriert sich dieses Verfahren ins Nirwana ...

Beispiel 11.26 *Betrachten wir nochmals die Matrix aus Beispiel 11.22 und wenden unser schönes neues Verfahren auf diese an. Für vorgegebene Toleranzen ε erhalten wir dann*[66] *die folgenden Ergebnisse*

[65] Nicht vergessen: Householder–Matrizen sind symmetrisch

[66] PII-300 unter Linux, SuSE 6.0, `Octave 2.0.13`

```
%#
%# HessenForm.m
%# Berechne Hessenberg-Form einer Matrix mit
%# Householder-Transformationen
%# Ueberschreibt A, gibt orthogonale Transformationsmatrix mit zurueck
%# Daten:
%#   A  Matrix

function [A,Q] = HessenForm( A )
  n = length( A );
  Q = II = eye( n );

  for j = 1:n-2
    x = A( j+1:n,j );
    y = [ zeros(j,1); HousehStep( x ) ];
    H = II - 2*y*y';

    Q = Q * H;
    A = H * A * H;
  end
%endfunction
```

Programm 11.3 `HessenForm.m`: Transformation einer Matrix auf obere Hessenberg–Gestalt.

```
%#
%# QRMethode1.m
%# Berechne Eigenwerte einer reellen Matrix (mit hoffentlich reellen
%# Eigenwerten) mit naivem QR-Verfahren.
%# Gibt Diagonalvektor von R zurueck und zeigt #Iterationen an.
%# Daten:
%#   A  Matrix
%#   t  Toleranz

function lambda = QRMethode1( A,t )
  H = HessenForm( A );
  Iterat = 0;

  while norm( auxSubdiag( H ) ) > t
    H = HessenRQ( H );
    Iterat = Iterat + 1;
  end

  disp( Iterat );

  lambda = diag( H );
%endfunction
```

Programm 11.4 `QRMethode1.m`: Simple QR–Iteration zur Bestimmung reeller Eigenwerte.

```
%#
%# auxSubdiag.m
%# Bestimme Subdiagonale einer quadratischen Matrix und gib sie als
%# Vektor zurueck
%# Daten:
%#   A  Matrix

function x = auxSubdiag( A )
  n = length( A );
  x = zeros( n-1,1 );

  for j = 1:n-1
    x( j ) = A( j+1,j );
  end
%endfunction
```

Programm 11.5 `auxSubdiag.m`: Einfache Routine für Subdiagonalelemente.

ε	# Iterationen	λ_1	λ_2	λ_3
10^{-3}	11	4.56155	-0.99983	0.43828
10^{-4}	14	4.56155	-1.00001	0.43846
10^{-5}	17	4.56155	-1.00000	0.43845
10^{-10}	31	4.56155	-1.00000	0.43845

Man sieht also recht schön, daß man alle drei Iterationsschritte eine Dezimalstelle bei den Nebendiagonalelementen gewinnt.

Die nächste Beschleunigungsmethode ist schnell erklärt:

Kann man das Eigenwertproblem in Teilprobleme aufspalten, so sollte man das tun.

Hat nämlich eine Hessenbergmatrix die Gestalt

$$H = \left[\begin{array}{cccc|ccc} * & \dots & \dots & * & & & \\ * & \ddots & & \vdots & & & \\ & \ddots & \ddots & \vdots & & * & \\ & & * & * & & & \\ \hline & & & & * & \dots & * \\ & & & & * & \ddots & \vdots \\ & & & & & \ddots & \vdots \\ & & & & & * & * \end{array}\right] = \left[\begin{array}{cc} H_1 & * \\ & H_2 \end{array}\right],$$

```
%#
%# QRSplit1.m
%# Berechne Eigenwerte einer reellen Matrix (mit hoffentlich reellen
%# Eigenwerten) mit QR-Verfahren und Split
%# Gibt Diagonalvektor von R zurueck
%# Daten:
%#   A  Matrix
%#   t  Toleranz

function [lambda,It] = QRSplit1( A,t )
  n = length( A );

  if n > 1
    H = HessenForm( A );          % Hessenberg-Matrix

    [H1,H2,It] = QRIter( H,t );   % Zerlege
    [l1,It1] = QRSplit1( H1,t );
    [l2,It2] = QRSplit1( H2,t );
    It = It + It1 + It2;
    lambda = [ l1;l2 ];
  else
    It = 0;
    lambda = A( 1,1 );
  end
%endfunction
```

Programm 11.6 `QRSplit1.m`: QR–Zerlegung mit Aufspaltung

dann kann man das Eigenwertproblem in ein Eigenwertproblem bezüglich H_1 und eines bezüglich H_2 aufspalten[67].
Laut [16] erklärt man ein Subdiagonalelement $h_{j+1,j}$ für "klein genug", wenn

$$|h_{j+1,j}| \leq \hat{u}\left(|h_{jj}| + |h_{j+1,j+1}|\right). \tag{11.15}$$

Hier machen wir es uns leichter und zerlegen eine Matrix, wenn ihr (betragsmäßig) kleinstes Nebendiagonalelement unter eine vorgegebene Toleranz fällt. Das Vorgehen ist nun wie folgt: Die Funktion zur Eigenwertbestimmung bestimmt mit Hilfe einer QR–Iteration die Zerlegungsmatrizen H_1 und H_2 und ruft sich dann rekursiv mit diesen Matrizen auf. Ist eine "Eingabe" eine 1×1–Matrix, dann ist deren Eigenwert ziemlich leicht festzustellen, ist die Eingabe eine

[67] Wenn $a + b = c$ ist, dann ist $a^3 + b^3 \leq c^3$, wobei bis auf Ausnahmefälle "$<$" steht!

```
%#
%# QRIter.m
%# Fuehre QR-Iterationen auf Householder-Matrix durch, bis kleinstes
%# Subdiagonalement < Toleranz ist. Liefere Zerlegung zurueck
%# Daten:
%#   H  Hessenberg-Matrix
%#   t  Toleranz

function [H1,H2,It] = QRIter( H,t )
  It = 0;
  n = length( H );
  [ j,m ] = auxSubdMin( H );

  while m > t
    It = It + 1;
    H = HessenRQ( H );
    [ j,m ] = auxSubdMin( H );
  end

  H1 = H( 1:j, 1:j );
  H2 = H( j+1:n, j+1:n );
%endfunction
```

Programm 11.7 `QRIter.m`: Iterationsschritt zu `QRSplit1`.

```
%#
%# auxSubdMin.m
%# Bestimme betragsmaessiges Minimalelement auf der Subdiagonalen einer
%# quadratischen Matrix und dessen Index
%# Daten:
%#   A  Matrix

function [k,t] = auxSubdMin( A )
  n = length( A );
  k = 1;
  t = abs( A( 2,1 ) );

  for j = 2:n-1
    if abs( A( j+1,j ) ) < t
      t = abs( A( j+1,j ) );
      k = j;
    end
  end
%endfunction
```

Programm 11.8 `auxSubdMin.m`: Bestimmung des minimalen Subdiagonalwerts.

2×2–Matrix A, dann ist deren charakteristisches Polynom

$$\begin{aligned} p_A(x) &= \det(A - xI) = x^2 - \text{trace}\,(A)x + \det(A) \\ &= x^2 - \underbrace{(a_{11} + a_{22})}_{=:b} x + \underbrace{(a_{11}\, a_{22} - a_{12}\, a_{21})}_{=:c}. \end{aligned}$$

Ist nun die *Diskriminante* $b^2 - 4c$ positiv, dann hat A zwei reelle Eigenwerte, nämlich[68]

$$x_1 = \frac{1}{2}\left(b + \operatorname{sgn} b\, \sqrt{b^2 - 4c}\right) \qquad \text{und} \qquad x_2 = \frac{c}{x_1},$$

andernfalls sind die Eigenwerte komplex und zwar

$$\frac{1}{2}\left(-b \pm i\sqrt{4c - b^2}\right),$$

wir haben jetzt also auch den Fall komplexer Eigenwerte im Griff.

Beispiel 11.27 *Sehen wir uns jetzt die Matrix aus Beispiel 11.23 an und wenden die Routine* `QRSplit1a` *darauf an. Und siehe da – es funktioniert:*

ε	# Iterationen	λ_1	λ_2	λ_3
10^{-3}	12	9.7406	$-3.8703 + 0.6479\, i$	$-3.8703 - 0.6479\, i$
10^{-4}	14	9.7407	$-3.8703 + 0.6479\, i$	$-3.8703 - 0.6479\, i$
10^{-5}	17	9.7407	$-3.8703 + 0.6480\, i$	$-3.8703 - 0.6480\, i$
10^{-6}	19	9.7407	$-3.8703 + 0.6480\, i$	$-3.8703 - 0.6480\, i$
10^{-7}	21	9.7407	$-3.8703 + 0.6480\, i$	$-3.8703 - 0.6480\, i$

Die Verwendung von Hessenbergmatrizen erlaubt es uns ja, jeden einzelnen Iterationsschritt in wesentlich kürzerer Zeit auszuführen. Nun müssen wir also nur noch versuchen, die *Anzahl* der Iterationen zu verkleinern, also die *Konvergenzgeschwindigkeit* zu erhöhen, denn

Die Konvergenzrate der Nebendiagonalelemente $h_{j+1,j}$ ist von der Größenordnung

$$\left(\frac{|\lambda_{j+1}|}{|\lambda_j|}\right)^k, \qquad j = 1, \ldots, n-1.$$

Das Stichwort hier heißt *Spektralverschiebung*. Dazu bemerkt man, daß für $\mu \in \mathbb{R}$ die Matrix $A - \mu I$ die Eigenwerte $\lambda_1 - \mu, \ldots, \lambda_n - \mu$ hat. Für eine beliebige invertierbare Matrix B hat dann $B\,(A - \mu I)\,B^{-1} + \mu I$ wieder die Eigenwerte $\lambda_1, \ldots, \lambda_n$ – man kann also vor und nach einer Ähnlichkeitstransformation das Spektrum der Matrix verschieben. Ordnet man die Eigenwerte um, z.B. als $\mu_1, \ldots, \mu_n$, so daß

$$|\mu_1 - \mu| > |\mu_2 - \mu| > \cdots > |\mu_n - \mu|, \qquad \{\mu_1, \ldots, \mu_n\} = \{\lambda_1, \ldots, \lambda_n\},$$

[68] Das ist der "stabile Weg", die Nullstellen einer Parabel zu bestimmen.

```
%#
%# QRSplit1a.m
%# Berechne Eigenwerte einer reellen Matrix (mit hoffentlich reellen
%# Eigenwerten) mit QR-Verfahren, Split und 2x2-Sonderbehandlung
%# Gibt Diagonalvektor von R zurueck
%# Daten:
%#   A  Matrix
%#   t  Toleranz

function [lambda,It] = QRSplit1a( A,t )
  n = length( A );

  if n == 1
    It = 0;
    lambda = A( 1,1 );
    return;
  end

  if n == 2
    It = 0;
    lambda = Eigen2x2( A );
    return;
  end

  H = HessenForm( A );          % Hessenberg-Matrix

  [H1,H2,It] = QRIter( H,t );   % Zerlege
  [l1,It1] = QRSplit1a( H1,t );
  [l2,It2] = QRSplit1a( H2,t );
  It = It + It1 + It2;
  lambda = [ l1;l2 ];

%endfunction
```

Programm 11.9 `QRSplit1a.m`: QR–Zerlegung mit Split und "Spezialbehandlung" für 2×2–Matrizen.

```
%#
%# Eigen2x2.m
%# Berechne Eigenwerte einer 2x2-Matrix
%# Daten:
%#   A  Matrix

function lambda = Eigen2x2( A )
  b = A( 1,1 ) + A( 2,2 );
  c = A( 1,1 ) * A( 2,2) - A( 1,2 ) * A( 2,1 );
  d = b^2/4 - c;

  if d > 0                  % rell !!!
    if b == 0
      lambda = [ sqrt( c ); -sqrt( c ) ];
    else
      x = ( b/2 + sign(b)*sqrt(d) );
      lambda = [ x; c/x ];
    end
  else
    lambda = [ b/2 + i * sqrt(-d); b/2 - i * sqrt(-d) ];
  end
%endfunction
```

Programm 11.10 `Eigen2x2.m`: Bestimme Eigenwerte einer 2×2–Matrix mit elementaren Methoden.

```
%#
%# QRMethode2.m
%# Berechne Eigenwerte einer reellen Matrix (mit hoffentlich reellen
%# Eigenwerten) mit QR-Verfahren und Spektralverschiebung
%# Gibt Diagonalvektor von R zurueck
%# Daten:
%#   A  Matrix
%#   t  Toleranz

function lambda = QRMethode2( A,t )
  n = length( A ); II = eye( n );
  H = HessenForm( A );
  Iterat = 0;

  while norm( auxSubdiag( H ) ) > t
    m = H( n,n );
    H = HessenRQ( H - m * II ) + m * II;
    Iterat = Iterat + 1;
  end

  disp( Iterat );

  lambda = diag( H );
%endfunction
```

Programm 11.11 `QRMethode2.m`: Das QR–Verfahren mit expliziter Spektralverschiebung.

und ist μ eine gute Näherung von μ_n, dann konvergiert natürlich das Subdiagonalelement $h_{n-1,n}^{(k)}$ sehr schnell gegen 0, wenn man das QR–Verfahren mit $H^{(0)} = A - \mu I$ startet. Noch besser ist es aber, *in jedem einzelnen Schritt* die Spektralverschiebung durchzuführen. Dabei nehmen wir als (heuristische) Näherung für μ den Wert $h_{nn}^{(k)}$. Das führt zu folgender Iterationsvorschrift

$$H^{(k+1)} = \left(H^{(k)} - \mu_k I\right)_* + \mu_k I, \qquad \mu_k := h_{nn}^{(k)}, \qquad k \in \mathbb{N}_0, \tag{11.16}$$

mit der Startmatrix $H^{(0)} = QAQ^T$. Dieses Verfahren ist in `QRMethode2.m` realisiert.

Bemerkung 11.28 *Ist der Verschiebungswert μ hinreichend nahe bei einem Eigenwert λ, dann zerfällt die Matrix mit nur einem Iterationsschritt.*

Beispiel 11.29 *Und nochmals die Matrix aus Beispiel 11.22. Wir geben wieder Toleranzen ε vor und vergleichen nun die Anzahl der nötigen Iterationen.*

```
%#
%# QRSplit2.m
%# Berechne Eigenwerte einer reellen Matrix (mit hoffentlich reellen
%# Eigenwerten) mit QR-Verfahren, Split, Spektralverschiebung und 2x2-Fall
%# Gibt Diagonalvektor von R zurueck
%# Daten:
%#   A  Matrix
%#   t  Toleranz

function [lambda,It] = QRSplit2( A,t )
  n = length( A );

  if n == 1
    It = 0;
    lambda = A( 1,1 );
    return;
  end

  if n == 2
    It = 0;
    lambda = Eigen2x2( A );
    return;
  end

  H = HessenForm( A );          % Hessenberg-Matrix

  [H1,H2,It] = QRIter2( H,t );  % Zerlege
  [l1,It1] = QRSplit2( H1,t );
  [l2,It2] = QRSplit2( H2,t );
  It = It + It1 + It2;
  lambda = [ l1;l2 ];

%endfunction
```

Programm 11.12 `QRSplit2.m`: Das QR–Verfahren mit expliziter Spektralverschiebung und Zerlegung.

```
%#
%# QRIter2.m
%# Fuehre QR-Iterationen mit Spektralverschiebung auf Hessenberg-Matrix
%# durch, bis kleinstes
%# Subdiagonalement < Toleranz ist. Liefere Zerlegung zurueck
%# Daten:
%#   H  Hessenberg-Matrix
%#   t  Toleranz

function [H1,H2,It] = QRIter2( H,t )
  It = 0;
  n = length( H );
  II = eye( n );
  [ j,m ] = auxSubdMin( H );

  while m > t
    It = It + 1;
    H = HessenRQ( H - H( n,n )*II ) + H( n,n )*II;
    [ j,m ] = auxSubdMin( H );
  end

  H1 = H( 1:j, 1:j );
  H2 = H( j+1:n, j+1:n );
%endfunction
```

Programm 11.13 `QRIter2.m`: Iterationsschritt zu `QRSplit2`.

	# Iterationen			
ε	`QRMethode1`	`QRMethode2`	`QRSplit1`	`QRSplit2`
10^{-5}	17	11	17	5
10^{-6}	19	12	19	5
10^{-7}	22	14	22	5
10^{-8}	25	16	25	5
10^{-9}	28	18	28	5
10^{-10}	31	19	31	5
10^{-20}	58	36	58	10

Die Spektralverschiebung beschleunigt das Verfahren also deutlich, insbesondere in Verbindung mit Zerlegungen der Matrix.

Es sieht nicht nur so aus – man kann zeigen [44], daß das QR–Verfahren mit expliziter Spektralverschiebung *quadratisch* konvergiert, der Fehler ist also nach k Interationsschritten für ein $\rho < 1$ nur[69]

$$O\left(\rho^{2k}\right) \qquad \text{anstelle von} \qquad O\left(\rho^{k}\right).$$

So schön diese Idee ist – sie funktioniert nur richtig gut mit *reellen* Eigenwerten, im Falle eines komplexen Eigenwerts wird es problematischer. Trotzdem können wir uns behelfen, den schließlich treten komplexe Eigenwerte ja paarweise auf. Das führt zur Idee des "Doppelschritt–Verfahrens":

> *Anstatt das Spektrum um einen Eigenwert zu verschieben, den man heuristisch mit $h_{n,n}^{(k)}$ annähert, führt man zwei Verschiebungen, und zwar um die Eigenwerte von*
>
> $$B = \begin{bmatrix} h_{n-1,n-1}^{(k)} & h_{n-1,n}^{(k)} \\ h_{n,n-1}^{(k)} & h_{n,n}^{(k)} \end{bmatrix},$$
>
> *in "einem Schritt" durch.*

Hier gibt es zwei Möglichkeiten: Entweder sind beide Eigenwerte, μ, μ' von B reell, dann läuft das Ganze wie oben, oder sie sind komplex, dann aber konjugiert komplex und wir haben als Eigenwerte $\mu, \overline{\mu} \in \mathbb{C}$. Wie wir sehen werden, kann man auch den zweiten Fall *reell durchführen*. Jetzt aber erst einmal die Theorie: Es seien $Q_k, Q_k' \in \mathbb{C}^{n\times n}$ und $R_k, R_k' \in \mathbb{C}^{n\times n}$ die Matrizen der *komplexen* QR–Zerlegungen

$$\begin{aligned} Q_k R_k &= H^{(k)} - \mu I \\ Q_k' R_k' &= R_k Q_k + (\mu - \overline{\mu})\, I \end{aligned}$$

Dann ist

$$\begin{aligned} H^{(k+1)} &:= R_k' Q_k' + \overline{\mu} I = {Q_k'}^H \left(R_k Q_k + (\mu - \overline{\mu})\, I\right) Q_k' + \overline{\mu} I \\ &= {Q_k'}^H R_k Q_k Q_k' + \mu I = {Q_k'}^H Q_k^H \left(H^{(k)} - \mu I\right) Q_k Q_k' + \mu I = \underbrace{\left(Q_k Q_k'\right)^H}_{=U^H} H^{(k)} \underbrace{Q_k Q_k'}_{=U}. \end{aligned}$$

[69] Möglicherweise aber mit einer signifikant schlechteren Konstante!

Unter Verwendung der Rechtsdreiecksmatrix $R = R'_k R_k$ ist

$$\begin{aligned} UR &= Q_k Q'_k R'_k R_k = Q_k \left(R_k Q_k + (\mu - \overline{\mu})\, I\right) R_k \\ &= Q_k R_k Q_k R_k + (\mu - \overline{\mu})\, Q_k R_k = \left(H^{(k)} - \mu I\right)^2 + (\mu - \overline{\mu}) \left(H^{(k)} - \mu I\right) \\ &= \left(H^{(k)}\right)^2 - 2\mu H^{(k)} + \mu^2 I + (\mu - \overline{\mu})\, H^{(k)} - \left(\mu^2 - \mu\overline{\mu}\right) I \\ &= \left(H^{(k)}\right)^2 - (\mu + \overline{\mu})\, H^{(k)} + \mu\overline{\mu} I =: X \end{aligned} \tag{11.17}$$

Ist $\mu = \alpha + i\beta$, dann ist $\mu + \overline{\mu} = 2\alpha$ und $\mu\overline{\mu} = |\mu|^2 = \alpha^2 + \beta^2$, also ist die Matrix X auf der rechten Seite von (11.17) reell. Als reelle Matrix hat X aber auch eine reelle QR–Zerlegung, also $X = QR$ und nach Lemma 11.21 ist $U = \Theta Q$ für eine Phasenmatrix $\Theta \in \mathbb{C}^{n\times n}$. Nachdem wir gerne reell weiteriterieren würden, legt dies das *QR–Doppelschrittverfahren*

$$\begin{aligned} Q_k R_k &= \left(H^{(k)}\right)^2 - \left(h^{(k)}_{n-1,n-1} + h^{(k)}_{n,n}\right) H^{(k)} \\ &\qquad + \left(h^{(k)}_{n-1,n-1}\, h^{(k)}_{n,n} - h^{(k)}_{n-1,n}\, h^{(k)}_{n,n-1}\right) I, \\ H^{(k+1)} &= Q_k^T H^{(k)} Q_k, \end{aligned} \tag{11.18}$$

nahe.

Bemerkung 11.30 *(Doppelschrittverfahren)*

1. *Die Matrix X aus (11.17) ist keine Hessenberg–Matrix mehr, sie hat eine weitere Subdiagonale. Trotzdem kann die QR-Zerlegung von X noch sehr einfach berechnet werden, nur eben mit $2n-3$ Jacobi–Rotationen anstelle von $n-1$ bei einer Hessenberg–Matrix.*
2. *Die Multiplikation $Q_k^T H^{(k)} Q_k$ ist wegen ihrer hohen Komplexität natürlich keine effektive Methode zur Iteration; das kann man aber auch beheben, siehe z.B. [16] oder [41, S. 272–278].*
3. *Abschließend sollte man natürlich $H^{(k+1)}$ wieder auf Hessenberg–Gestalt bringen.*
4. *Das Doppelschrittverfahren bringt natürlich nur dann was, wenn die Matrix A komplexe Eigenwerte hat; bei symmetrischen Matrizen hingegen ist es kein Vorteil.*

Beispiel 11.31 *Betrachten wir die beiden Verfahren am Beispiel der Matrizen aus den Beispielen 11.22 und 11.23*

	# Iterationen rell		# Iterationen komplex	
ε	`QRSplit2`	`QRSplit3`	`QRSplit2`	`QRSplit3`
10^{-10}	1	1	9	4
10^{-20}	8	2	17	5
10^{-30}	25	2	43	5

Daß das Doppelschrittverfahren hier so gut abschneidet, hat den Grund, daß es immer "zwei Eigenwerte auf einmal" erledigt.

```
%#
%# QRDoppel.m
%# Fuehre QR-Doppelschritt und Ruecktransformation auf Hessenberg-Matrix
%# durch, bis kleinstes
%# Subdiagonalement < Toleranz ist. Liefere Zerlegung zurueck
%# Daten:
%#   H  Hessenberg-Matrix
%#   t  Toleranz

function [H1,H2,It] = QRDoppel( H,t )
  It = 0;
  n = length( H );
  II = eye( n );
  [ j,m ] = auxSubdMin( H );

  while m > t
    It = It + 1;
    X = H * H \                  % Matrix X
        - ( H( n-1,n-1 ) + H( n,n ) ) * H \
        + ( H( n-1,n-1 )*H( n,n ) - H( n,n-1 )*H( n-1,n ) ) * II;
    [Q,R] = HousehQR( X );     % QR-Zerlegung
    H = HessenForm( Q'*H*Q );
    [ j,m ] = auxSubdMin( H );
  end

  H1 = H( 1:j, 1:j );
  H2 = H( j+1:n, j+1:n );
%endfunction
```

Programm 11.14 `QRDoppel.m`: Zerlegungsschritt Doppelschritt

```
%#
%# QRSplit3.m
%# Berechne Eigenwerte einer reellen Matrix (mit hoffentlich reellen
%# Eigenwerten) mit QR-Verfahren, Split, Doppelschritt und 2x2-Fall
%# Gibt Diagonalvektor von R zurueck
%# Daten:
%#   A  Matrix
%#   t  Toleranz

function [lambda,It] = QRSplit3( A,t )
  n = length( A );

  if n == 1
    It = 0;
    lambda = A( 1,1 );
    return;
  end

  if n == 2
    It = 0;
    lambda = Eigen2x2( A );
    return;
  end

  H = HessenForm( A );          % Hessenberg-Matrix

  [H1,H2,It] = QRDoppel( H,t );  % Zerlege
  [l1,It1] = QRSplit3( H1,t );
  [l2,It2] = QRSplit3( H2,t );
  It = It + It1 + It2;
  lambda = [ l1;l2 ];

%endfunction
```

Programm 11.15 `QRSplit3.m`: Doppelschrittverfahren.

11.6 Das Jacobi–Verfahren für symmetrische Matrizen

Zum Abschluß dieses Kapitels wollen wir uns noch den "Klassiker" [26], nämlich das Jacobi–Verfahren ansehen. Hierbei starten wir mit einer *symmetrischen* Matrix $A \in \mathbb{R}^{n\times n}$, die man nach dem Hauptachsentheorem ja orthogonal diagonalisieren kann, das heißt, es gibt eine orthogonale Matrix $Q \in \mathbb{R}^{n\times n}$, so daß

$$A = Q\Lambda Q^T, \qquad \Lambda = \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{bmatrix}.$$

Auch das Jacobi–Verfahren ist wieder ein Iterationsverfahren. Ausgehend von $A^{(0)} = A$ bestimmt man, für $k \in \mathbb{N}$, Indizes[70] $1 \le j < m \le n$, so daß

$$\left|a_{jm}^{(k)}\right| = \max_{r<s} \left|a_{rs}^{(k)}\right|,$$

und einen Drehwinkel ϕ_k, so daß

$$0 = a_{jm}^{(k+1)} := \left(J^T\left(j,m\,;\,\phi_k\right)\; A^{(k)}\; J\left(j,m\,;\,\phi_k\right)\right)_{jm}.$$

Dieses Verfahren hat einige offensichtliche Eigenschaften:

1. Ist A symmetrisch, so sind alle Matrizen $A^{(k)}$, $k \in \mathbb{N}$, symmetrisch und orthogonal ähnlich zu A.

2. Ist $a_{jm}^{(k)} = 0$, dann ist die Matrix eine Diagonalmatrix und es ist nichts mehr zu tun.

3. Der Winkel ϕ_k kann aus der Bestimmungsgleichung

$$0 = \left(a_{jj}^{(k)} - a_{mm}^{(k)}\right)\cos\phi_k\;\sin\phi_k + a_{jm}^{(k)}\left(\cos^2\phi_k - \sin^2\phi_k\right) \tag{11.19}$$

ermittelt werden, die nach ϕ_k auflösbar ist, wenn $a_{jm}^{(k)} \neq 0$ ist. In der Tat, mit $c = \cos\phi_k$ und $s = \sin\phi_k$ dividieren wir (11.19) durch $-c^2\, a_{jm}^{(k)}$ und erhalten, daß

$$0 = -\frac{a_{jj}^{(k)} - a_{mm}^{(k)}}{a_{jm}^{(k)}}\,\frac{s}{c} - 1 + \left(\frac{s}{c}\right)^2 =: t^2 - b\,t - 1, \qquad b := \frac{a_{jj}^{(k)} - a_{mm}^{(k)}}{a_{jm}^{(k)}},$$

sein muss, was eine quadratische Gleichung in $t = \tan\phi_k$, aus deren auslöschungsfreier[71] Lösung

$$t = \frac{1}{2}\left(b + \operatorname{sgn} b\,\sqrt{b^2+4}\right)$$

sich der Drehwinkel und damit auch die Drehmatrix bestimmen lassen.

[70] Diese Indizes bestimmen also immer ein Element in der "rechten oberen" Hälfte der Matrix – da die Matrizen alle symmetrisch sein werden, ist das dasselbe, als wenn man unter allen Nichtdiagonalelementen suchen würde.

[71] Und damit auch numerisch stabilen.

4. Die Iteration beeinflußt "nur" die Zeilen und Spalten j und m und kann daher sehr effektiv durchgeführt werden.

5. Allerdings können Elemente, die im k–ten Iterationsschritt zum Verschwinden gebracht wurden, durchaus in einem späteren Iterationsschritt wieder $\neq 0$ werden – das Jacobi–Verfahren ist also ein echtes Iterationsverfahren und terminiert normalerweise nicht nach einer endlichen Anzahl von Schritten.

6. Ein großes Problem des Jacobiverfahrens besteht in dem Mißverhältnis zwischen der Anzahl der Rechenoperationen, nämlich $O(n)$, und der Anzahl der Vergleichsoperationen, nämlich $O\left(n^2\right)$, in jedem Iterationsschritt. Deswegen verwendet man zumeist das *zyklische* Jacobi–Verfahren, bei dem in *jedem* Iterationsschritt *alle* Subdiagonalelemente einmal zum Verschwinden gebracht werden. So erwischt man auf jeden Fall auch das betragsgrößte und spart sich die Vergleichsoperationen, die es ja eben auch nicht umsonst gibt[72]

Übung 11.6 Verifizieren Sie (11.19), bestimmen Sie daraus den Winkel ϕ_k und implementieren Sie das Jacobi–Verfahren in `Matlab` oder `Octave`.

Was man nun zeigen kann, ist, daß die Folge $A^{(k)}$ der im Jacobi–Verfahren erzeugten Matrizen gegen eine Diagonalmatrix konvergiert.

Satz 11.32 *Die Folge $A^{(k)}$ der vom Jacobi–Verfahren erzeugten Matrizen konvergiert gegen eine Diagonalmatrix Λ.*

Beweis: Wir werden zeigen, daß die Folge

$$\sigma_k := 2 \sum_{1 \le r < s \le n} \left(a_{rs}^{(k)}\right)^2 = \left\|A^{(k)}\right\|_F^2 - \sum_{r=1}^{n} \left(a_{rr}^{(k)}\right)^2, \qquad k \in \mathbb{N}_0,$$

der Quadratsummen der Nichtdiagonalelemente für $k \to \infty$ gegen Null konvergiert. Da[73]

$$\underbrace{\begin{bmatrix} a_{jj}^{(k+1)} & a_{jm}^{(k+1)} \\ a_{jm}^{(k+1)} & a_{mm}^{(k+1)} \end{bmatrix}}_{=:B^{(k+1)}} = \begin{bmatrix} \cos\phi_k & -\sin\phi_k \\ \sin\phi_k & \cos\phi_k \end{bmatrix} \underbrace{\begin{bmatrix} a_{jj}^{(k)} & a_{jm}^{(k)} \\ a_{jm}^{(k)} & a_{mm}^{(k)} \end{bmatrix}}_{=:B^{(k)}} \begin{bmatrix} \cos\phi_k & \sin\phi_k \\ -\sin\phi_k & \cos\phi_k \end{bmatrix},$$

ist

$$\left(a_{jj}^{(k+1)}\right)^2 + 2\left(a_{jm}^{(k+1)}\right)^2 + \left(a_{mm}^{(k+1)}\right)^2 = \left\|B^{(k+1)}\right\|_F^2 = \left\|B^{(k)}\right\|_F^2 = \left(a_{jj}^{(k)}\right)^2 + 2\left(a_{jm}^{(k)}\right)^2 + \left(a_{mm}^{(k)}\right)^2.$$

Weil außerdem

$$a_{rr}^{(k+1)} = a_{rr}^{(k)}, \qquad r \in \{1, \ldots, n\} \setminus \{j, m\},$$

[72] Im Zeitalter der Fließkommakoprozessoren ist der Zeitaufwand für den Speicherzugriff nicht mehr unbedingt vernachlässigbar gegenüber dem Zeitaufwand für Rechenoperationen.

[73] Das a_{jm} in der zweiten Zeile der Matrizen ist *kein* Tipfehler, sondern Konsequenz der Symmetrie.

ist

$$\begin{aligned}\sum_{r=1}^{n}\left(a_{rr}^{(k+1)}\right)^2 &= \sum_{r\neq j,m}\left(a_{rr}^{(k+1)}\right)^2+\left(a_{jj}^{(k+1)}\right)^2+\left(a_{mm}^{(k+1)}\right)^2\\ &= \sum_{r\neq j,m}\left(a_{rr}^{(k)}\right)^2+\left(a_{jj}^{(k)}\right)^2+\left(a_{mm}^{(k)}\right)^2-2\left(a_{jm}^{(k+1)}\right)^2+2\left(a_{jm}^{(k)}\right)^2\\ &= \sum_{r=1}^{n}\left(a_{rr}^{(k)}\right)^2-2\left(a_{jm}^{(k+1)}\right)^2+2\left(a_{jm}^{(k)}\right)^2\end{aligned}$$

Nun ist der Drehwinkel ϕ_k aber gerade so bestimmt, daß $a_{jm}^{(k+1)}=0$ ist und somit ist wegen der orthogonalen Invarianz der Frobenius–Norm

$$\begin{aligned}\sigma_{k+1} &= \left\|A^{(k+1)}\right\|_F^2-\sum_{r=1}^{n}\left(a_{rr}^{(k+1)}\right)^2=\left\|A^{(k)}\right\|_F^2-\sum_{r=1}^{n}\left(a_{rr}^{(k)}\right)^2-2\left(a_{jm}^{(k)}\right)^2\\ &= \sigma_k-2\left(a_{jm}^{(k)}\right)^2.\end{aligned}$$

Weil $a_{jm}^{(k)}$ das *betragsgrößte* Nichtdiagonalelement war, gilt

$$\left(a_{jm}^{(k)}\right)^2\geq\frac{\sigma_k}{n(n-1)},$$

und damit ist

$$\sigma_{k+1}\leq\left(1-\frac{2}{n(n-1)}\right)\sigma_k\leq\cdots\leq\left(1-\frac{2}{n(n-1)}\right)^{k+1}\sigma_0,\qquad k\in\mathbb{N}_0,$$

was offensichtlich gegen 0 konvergiert. □

Science is built up with facts, as a house is with stones. But a collection of facts is no more a science than a heap of stones is a house.

H. Poincaré

12 Nochmals Nullstellen von Polynomen

Wir haben ja schon am Anfang des letzten Kapitels gesehen, daß und wie man mit Eigenwertmethoden die Nullstellen von Polynomen berechnen kann, nämlich durch Eigenwertbestimmung bei der Frobeniusmatrix aus (11.3), die erfreulicherweise sogar bereits eine Hessenbergmatrix ist[74]. Andererseits sollte man natürlich so ein Verfahren dann auch mit dem "Standardverfahren" für nichtlineare Gleichungen, also dem *Newton–Verfahren* vergleichen.

12.1 Abdivieren, jetzt aber richtig

Daß das Newton–Verfahren immer konvergiert, wenn es nur richtig gestartet wird, das wissen wir noch aus Teil I, [36]. Zur Erinnerung:

Proposition 12.1 *Sei $p \in \Pi_n$ ein Polynom mit n reellen Nullstellen $\xi_1 \leq \cdots \leq \xi_n$. Dann konvergiert für alle Startwerte $x^{(0)} \geq \xi_n$ bzw. $x^{(0)} \leq \xi_1$ das Newton–Verfahren gegen ξ_n bzw. ξ_1.*

Damit haben wir eine "Strategie", *alle* Nullstellen eines Polynoms zu finden: Man startet das Newton–Verfahren nach Möglichkeit "weit genug" rechts oder links, findet eine Nullstelle[75], dividiert diese ab und sucht weiter, bis man schließlich alle Nullstellen gefunden hat. Das Abdividieren von Nullstellen ist nicht weiter schwer: Für $\xi \in \mathbb{R}$ ist

$$\sum_{j=0}^{n} a_j\, x^j = a_n\, x^{n-1}\,(x-\xi) + \underbrace{(a_{n-1} + a_n\xi)}_{=:b_{n-1}}\, x^{n-1} + \sum_{j=0}^{n-2} \underbrace{a_j}_{=:b_j}\, x^j = a_n\, x^{n-1}\,(x-\xi) + \sum_{j=0}^{n-1} b_j\, x^j;$$

die zugehörige Funktion ist in `DivideZero.m` implementiert. Um nun *alle* Nullstellen zu finden, dividiert man eine gefundene Nullstelle ab und verwendet sie gleichzeitig[76] als Startwert für das nächste Newton–Verfahren.

[74] Was will man denn mehr?

[75] Am besten natürlich die betragsgrößte oder betragskleinste.

[76] In der Hoffnung, daß sie eine extremale Nullstelle war.

```
%#
%# DivideZero.m
%# Abdividieren einer Nullstelle, gibt gemachten Fehler als zweiten Wert
%# zurueck
%# Daten:
%#   p  Koeffizientenvektor
%#   x  Nullstelle


function [q,err] = DivideZero( p,x )
  n = length( p );
  q = zeros( 1,n-1 );

  for j = n:-1:2
    t = q( j-1 ) = p( j );
    p( j-1 ) = p( j-1 ) + t*x;
  end
  err = p( 1 );
%endfunction
```

Programm 12.1 `DivideZero.m`: Abdividieren einer Nullstelle.

```
%#
%# PolyZeros.m
%# Finde Nullstellen eines Polynoms mit vorgegebener Toleranz und durch
%# Abdividieren
%# Daten:
%#   p  Koeffizientenvektor
%#   x0 Startwert
%#   t  Toleranz

function z = PolyZeros( p,x0,t )
  n = length( p );
  z = zeros( 1,n-1 );
  x = x0;

  for j = 1:n-1
    z( j ) = x = PolyNewton( p,x,t );
    disp(x);
    p = DivideZero( p,x );
  end
%endfunction
```

Programm 12.2 `PolyZeros.m`: Newton–Verfahren mit Adividieren.

Es bleibt aber noch die Frage: *"Wie groß muss man den Startwert wählen"*. Dazu brauchen wir eine Abschätzung für die Eigenwerte einer Matrix und ein Hilfsmittel sind die sogenannten *Gerschgorin–Kreise*.

Definition 12.2 *Für $A \in \mathbb{R}^{n\times n}$ definiert man die* Gerschgorin–Kreise *als*

$$\mathscr{D}_j := \mathscr{D}_j(A) := \left\{ z \in \mathbb{C} \,:\, |z - a_{jj}| \le \sum_{k\neq j} |a_{jk}| \right\}, \qquad j = 1, \ldots, n. \tag{12.1}$$

Satz 12.3 *Das Spektrum einer Matrix ist in den Gerschgorin–Kreisen enthalten:*

$$\lambda(A) \subseteq \bigcup_{j=1}^{n} \mathscr{D}_j(A). \tag{12.2}$$

Korollar 12.4 *Jeder Eigenwert einer Matrix A erfüllt $|\lambda| \le \|A\|_\infty$.*

Beweis: Folgt sofort aus Satz 12.3, wenn man sich daran erinnert, daß

$$\|A\|_\infty = \max_{j=1,\ldots,n} \sum_{k=1}^{n} |a_{jk}|$$

die *Zeilensummennorm* ist. □

Eigentlich kennen wir Corollar 12.4 schon aus [36], nämlich aus der Beobachtung, daß der *Spektralradius* $\rho(A)$ der Matrix A die Ungleichung $\rho(A) \le \|A\|$ für jede submultiplikative Matrixnorm[77] und damit eben auch für $\|\cdot\|_\infty$ erfüllt. Die Gerschgorin–Kreise sollte man aber trotzdem kennen, denn sie sind "subtiler" und ermöglichen möglicherweise eine "Partitionierung" des Spektrums.

Beispiel 12.5 *Die Matrix*

$$A = \begin{bmatrix} 10 & 2 & 3 \\ -1 & 0 & 2 \\ 1 & -2 & 1 \end{bmatrix}$$

hat die Gerschgorin–Kreise

$$\{|z - 10| \le 5\}, \qquad \{|z| \le 3\}, \qquad \{|z - 1| \le 3\}$$

und die Eigenwerte

```
octave> eig( [ 10 2 3 ; -1 0 2 ; 1 -2 1 ] )
ans =

  10.22600 +  0.00000i
   0.38700 +  2.22156i
   0.38700 -  2.22156i
```

[77] Also insbesondere für Operatornormen!

Schließlich brauchen wir noch ein kleines Hilfsmittel, einen echten "Klassiker" der numerischen linearen Algebra.

Lemma 12.6 *Ist $\|A\|_\infty < 1$, dann ist $I - A$ invertierbar und es ist*

$$(I - A)^{-1} = \sum_{j=0}^{\infty} A^j$$

wobei die Summe absolut konvergiert.

Beweis: Die Reihe konvergiert absolut, da

$$\sum_{j=0}^{\infty} \left\|A^j\right\|_\infty \leq \sum_{j=0}^{\infty} \|A\|_\infty^j = \frac{1}{1 - \|A\|_\infty}$$

und damit existiert die Matrix $B = \sum A^j$. Nun ist aber

$$(I - A)\, B = \sum_{j=0}^{\infty} A^j - \sum_{j=1}^{\infty} A^j = I,$$

und genau das haben wir ja behauptet. □

Dann bringen wir noch schnell den Beweis der Gerschgorin–Kreise hinter uns.
Beweis von Satz 12.3: Sei $\lambda \in \lambda(A)$ ein Eigenwert. Ist λ auch ein Diagonalelement von A, gibt es also ein j, so daß $\lambda = a_{jj}$, dann gehört λ trivialerweise zu $\mathscr{D}_j$ und wir brauchen weiter nichts mehr zu beweisen. Andernfalls schreiben wir A als

$$A = D - F = \begin{bmatrix} * & & \\ & \ddots & \\ & & * \end{bmatrix} - \begin{bmatrix} 0 & * & \dots & * \\ * & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & * \\ * & \dots & * & 0 \end{bmatrix}$$

und bemerken, daß $D - \lambda I$ invertierbar ist[78], die Matrix $A - \lambda I = D - \lambda I - F$ hingegen nicht. Damit ist aber auch

$$(D - \lambda I)^{-1} (D - \lambda I + F) = I - (D - \lambda I)^{-1} F$$

nicht invertierbar und nach Lemma 12.6 ist

$$1 \leq \left\|(D - \lambda I)^{-1} F\right\|_\infty = \max_{j=1,\dots,n} \sum_{k \neq j} \frac{|a_{jk}|}{|d_{jj} - \lambda|}.$$

Das heißt, es gibt mindestens einen Index j, so daß

$$|\lambda - d_{jj}| \leq \sum_{k \neq j} |a_{jk}|$$

[78] Weil ja λ eben **kein** Diagonaleintrag von A war.

was aber gerade $\lambda \in \mathscr{D}_j(A)$ bedeutet. □

Nach so vielen Vor-, Nach- und Zwischenbemerkungen jetzt wieder zurück zum Newton–Verfahren.

Beispiel 12.7 *Leider ist die Verwendung des Newton–Verfahrens nicht übermäßig stabil, insbesondere das Abdividieren führt zu Problemen.*

1. *Betrachten wir das Polynom*

$$p(x) = \prod_{j=0}^{13} \left(x - 2^{-j}\right), \qquad x \in \mathbb{R},$$

siehe [45, S. 259–260]. Startet man das Newton–Verfahren mit der größten Nullstelle $x^{(0)} = 1$*, dann erhält man die folgenden Nullstellen*[79]

	$\varepsilon = 10^{-6}$		$\varepsilon = \hat{u}$	
Exakt	*Berechnet*	*rel. Fehler*	*Berechnet*	*rel. Fehler*
1	1	0	1	0
0.5	0.50580	0.011600	0.5000	2.6645×10^{-15}
0.25	-0.13001	1.5201	0.25000	2.1575×10^{-11}
0.125	-0.24151	2.9321	0.12500	1.1627×10^{-6}
0.0625	$\rightarrow$		0.062352	0.0023729
0.03125			0.039506	0.26421
0.015625			$\rightarrow$	

der Startwert $x^{(0)} = 2$ *liefert sogar das noch weniger begeisternde Ergebnis*

	$\varepsilon = 10^{-6}$		$\varepsilon = \hat{u}$	
Exakt	*Berechnet*	*rel. Fehler*	*Berechnet*	*rel. Fehler*
1	1.0000	8.5183×10^{-10}	1.0000	2.2204×10^{-16}
0.5	0.50580	0.011592	0.50000	2.4125×10^{-12}
0.25	-0.12996	1.5199	0.25000	1.3175×10^{-8}
0.125	-0.24163	2.9330	0.12500	3.1917×10^{-5}
0.0625	$\rightarrow$		0.058106	0.070300
0.03125			0.054892	0.75654
0.015625			$\rightarrow$	

2. *Ein anderes "Monster" ist das schon klassische* Wilkinson–Polynom

$$p(x) = (x-1)\cdots(x-20), \qquad x \in \mathbb{R},$$

mit den ganz sauber getrennten Nullstellen $1, \ldots, 20$*. Dieses Polynom hat die Rundungsfehleranalyse sehr stark motiviert und beeinflußt, siehe [50]. Die Details sind in [47, 48]*

[79] "$\rightarrow$" bedeutet, daß die Rechnung nicht mehr terminiert, also das Verfahren sich "aufhängt".

ausgearbeitet. Die Koeffizienten dieses Polynoms sind sehr groß (bis zur Größenordnung 10^{18}*), was relative Fehler unvermeidlich macht, und das Problem ist* extrem *schlecht konditioniert bezüglich dieser Nullstellen. Deswegen ist eine numerische Berechnung praktisch unmöglich.*

Zur Stabilisierung der Berechnung *aller* Nullstellen ersetzt man das *explizite* Abdividieren durch *implizites* Abdividieren, was auf eine Idee von Maehly[80] zurückgeht, die eigentlich denkbar einfach ist.

Lemma 12.8 *Für* $p(x) = p_n\,(x-\xi_1)\cdots(x-\xi_n)$ *ist*

$$\frac{d}{dx}\frac{p(x)}{(x-\xi_1)\cdots(x-\xi_k)} = \frac{p'(x)}{(x-\xi_1)\cdots(x-\xi_k)} - \frac{p(x)}{(x-\xi_1)\cdots(x-\xi_k)}\sum_{j=1}^{k}\frac{1}{x-\xi_j}. \quad (12.3)$$

Beweis: Wir setzen $p_k = p\,(\cdot-\xi_1)^{-1}\cdots(\cdot-\xi_k)^{-1}$ und verwenden Induktion über k. Der Fall $k=1$ ergibt sich aus

$$p_1'(x) = \frac{d}{dx}\frac{p(x)}{(x-\xi_1)} = \frac{(x-\xi_1)\,p'(x)-p(x)}{(x-\xi_1)^2} = \frac{p'(x)}{(x-\xi_1)} - \frac{p(x)}{(x-\xi_1)^2}.$$

Außerdem ist für $k \geq 1$

$$\begin{aligned}
p_{k+1}'(x) &= \frac{d}{dx}\frac{p_k(x)}{(x-\xi_k)} = \frac{p_k'(x)}{(x-\xi_{k+1})} - \frac{p_k(x)}{(x-\xi_{k+1})^2} \\
&= \frac{1}{x-\xi_{k+1}}\left(\frac{p'(x)}{(x-\xi_1)\cdots(x-\xi_k)} - \frac{p(x)}{(x-\xi_1)\cdots(x-\xi_k)}\sum_{j=1}^{k}\frac{1}{x-\xi_j}\right) \\
&\qquad -\frac{1}{x-\xi_{k+1}}\,\frac{p(x)}{(x-\xi_1)\cdots(x-\xi_{k+1})} \\
&= \frac{p'(x)}{(x-\xi_1)\cdots(x-\xi_{k+1})} - \frac{p(x)}{(x-\xi_1)\cdots(x-\xi_{k+1})}\sum_{j=1}^{k+1}\frac{1}{x-\xi_j}.
\end{aligned}$$

□

Was bedeutet das nun für unser Newton–Verfahren für das Polynom

$$p_m(x) = \frac{p(x)}{(x-\xi_1)\cdots(x-\xi_m)}, \qquad x \in \mathbb{R},$$

wenn wir einmal die Nullstellen $\xi_1, \ldots, \xi_k$ bestimmt haben. Ganz einfach, aus (12.3) erhalten wir, daß

$$\frac{p_m(x)}{p_m'(x)} = \frac{p(x)}{p'(x) - p(x)\sum_{j=1}^{m}(x-\xi_j)^{-1}}, \qquad x \in \mathbb{R}, \quad (12.4)$$

[80] Leider konnte ich kein Literaturzitat finden

```
%#
%# MaehlyNewton.m
%# Nullstelle eines Polynoms mit dem Newton-Verfahren
%# Daten:
%#   p  Koeffizientenvektor
%#   z  Nullstellenvektor
%#   x0 Startwert
%#   t  Toleranz

function x = MaehlyNewton( p,z,x0,t )
  m = length( z );
  x = x0; x0 = x0+2*t;
  [f,ff] = Horner2Z( p,x );

  while abs(x-x0) > t
    x0 = x;
    y = 0;
    for j = 1:m
      y = y + 1 / ( x-z(j) );
    end
    x = x - f / ( ff - f*y );
    [f,ff] = Horner2Z( p,x );
  end
%endfunction
```

Programm 12.3 `MaehlyNewton.m`: Auffinden einer Nullstelle mit der Maehly–Newton–Iteration.

und damit, unter Verwendung der bereits gefundenen Nullstellen $\xi_1, \ldots, \xi_m$ die *Maehly–Newton–Iteration*

$$x^{(k+1)} = x^{(k)} - \frac{p\left(x^{(k)}\right)}{p'\left(x^{(k)}\right) - p\left(x^{(k)}\right) \sum_{j=1}^{m} \frac{1}{x^{(k)} - \xi_j}}, \qquad k \in \mathbb{N}_0, \quad x^{(0)} \in \mathbb{R}. \tag{12.5}$$

Beispiel 12.9 *Alle Nullstellen des Polynoms*

$$p(x) = \prod_{j=0}^{13} \left(x - 2^{-j}\right), \qquad x \in \mathbb{R},$$

aus Beispiel 12.7 können mit der "Maehly–Methode" nun hinreichend genau bestimmt werden: Bei einer Toleranz von 10^{-6} *liegen die relativen Fehler im Bereich zwischen* 10^{-7} *und* 10^{-16}*, bei Rechnung in Maschinengenauigkeit sogar bei höchstens* 10^{-15}.

```
%#
%# PolyZeros2.m
%# Finde Nullstellen eines Polynoms mit vorgegebener Toleranz und unter
%# Verwendung des Maehly-Verfahrens
%# Daten:
%#   p  Koeffizientenvektor
%#   x0 Startwert
%#   t  Toleranz

function z = PolyZeros2( p,x0,t )
  n = length( p );
  z = zeros( 1,n-1 );
  x = x0;

  for j = 1:n-1
    z( j ) = x = MaehlyNewton( p,z( 1:j-1 ),x,t );
    disp(x);
    x = .9 * x;
  end
%endfunction
```

Programm 12.4 `PolyZeros2.m`: Nullstellensuche unter Verwendung von Maehly.

Übung 12.1 Stellen Sie zu einem Polynom die Frobenius–Begleitmatrix (11.3) auf und bestimmen Sie die Nullstellen mit dem QR–Verfahren. Vergleichen Sie die numerischen Ergebnisse und experimentieren Sie mit zufälligen Nullstellen.

Bemerkung 12.10 *Um dem Ergebnis der Übung vorzugreifen: Das QR–Verfahren schafft es, die Nullstellen des Wilkinson–Polynoms ("Monster") sehr schnell und mit recht ordentlicher Genauigkeit zu ermitteln, "erfindet" aber komplexe Nullstellen für das Polynom mit den Nullstellen 2^{-j}, $j = 0, \ldots, 13$. Warum das so ist? Man muß ja wohl nicht alles verraten ...*

Bemerkung 12.11 *Ein Problem bleibt natürlich ganz genauso wie bei den Eigenwerten: Was tun mit komplexen Nullstellen, die bei reellen Polynomen leider auch auftreten können. Entweder man verwendet das QR–Verfahren (da ist das schon implementiert), oder man erweitert das Newton–Verfahren um eine komplexe Komponente. Das führt zum* Bairstow–Verfahren[81]*, das die Koeffizienten p, q eines* quadratischen *Teilers*[82] *$x^2 - px + q$ über ein Newton–Verfahren mit $n = 2$ bestimmt; siehe [41, S. 227] oder [45, S. 264–266].*

12.2 Sturmsche Ketten

Sturmsche Ketten ermöglichen es, die *Anzahl* der Nullstellen eines Polynoms vorherzusagen, ohne sie bestimmen zu müssen. Das Problem ist aber dann nicht diese Vorhersage, sondern *Existenz* und *Konstruktion* der Sturmschen Kette. Wir erinnern uns an die Definition von Π_n als Vektorraum aller Polynome von Grad $\leq n$, $n \in \mathbb{N}_0$.

Definition 12.12 *Sei $p \in \Pi_n$ ein reelles Polynom mit einfachen Nullstellen. Eine Folge $p_0, \ldots, p_m \in \Pi$ von reellen Polynomen heißt* Sturmsche Kette *(für p), falls:*

1. *$p_0 = p$.*
2. *Für jede reelle Nullstellen ξ von p ist*
$$\operatorname{sgn} p_1(\xi) = -\operatorname{sgn} p_0'(\xi). \tag{12.6}$$
3. *Für jede reelle Nullstelle ξ von p_j ist*
$$p_{j+1}(\xi)\, p_{j-1}(\xi) < 0, \qquad j = 1, \ldots, m-1. \tag{12.7}$$
4. *Das Polynom p_m ändert sein Vorzeichen nicht.*

Die "entscheidende" Eigenschaft (12.7) kann man auch anders interpretieren: Jede Nullstelle von p_j ist von unterschiedlichen Vorzeichen von p_{j-1} und p_{j+1} "eingeschlossen".
Bevor wir mit Hilfe einer Sturmschen Kette Nullstellen zählen, sehen wir uns zuerst einmal an, wie man so etwas bekommt.

[81] Schon wieder keine Literaturstelle
[82] Also ein bißchen wie die "Spezialbehandlung" für 2×2–Matrizen im QR–Verfahren.

Proposition 12.13 *Hat das Polynom $p \in \Pi_n$ nur einfache Nullstellen, dann ist die Folge $p_0 = p$, $p_1 = -p'$, sowie*[83]

$$p_{j-2}(x) = \omega(x)\, p_{j-1}(x) - p_j(x), \qquad j = 2, \ldots, m, \qquad \deg p_j < \deg p_{j-1}, \tag{12.8}$$

für ein passendes $m \leq n$ eine Sturmsche Kette.

Beweis: Die Definition von p_j, $j \geq 2$, in (12.8) ist nichts anderes als der *Euklidische Algorithmus* zur Bestimmung des größten gemeinsamen Teilers von $p_0 = p$ und $p_1 = p'$. Da der Grad in jedem Schritt reduziert wird, bricht das Verfahren nach $m \leq n$ Schritten ab, das heißt, es ist $p_{m+1} = 0$ und $p_m \neq 0$. Damit ist aber p_m der größte gemeinsame Teiler von p und p'. Hätte nun p_m eine reelle Nullstelle, dann wäre diese eine *gemeinsame* Nullstelle von p und p', also (mindestens) eine *doppelte* Nullstelle, im Gegensatz zur Annahme. Also ändert p_m auf $\mathbb{R}$ sein Vorzeichen nicht.
Ist schließlich ξ eine Nullstelle von p_j, also $p_j(\xi) = 0$, so ist nach (12.8)

$$p_{j-1}(\xi) = -p_{j+1}(\xi), \qquad j = 1, \ldots, m-1.$$

Wäre nun $p_{j-1}(\xi) = 0$, so wäre auch[84]

$$0 = p_{j-2}(\xi) = \cdots = \underbrace{p_1(\xi)}_{=-p'(\xi)} = \underbrace{p_0(\xi)}_{=p(\xi)},$$

was schon wieder eine doppelte Nullstelle liefern würde. □

Satz 12.14 *Es sei $p_0, \ldots, p_m$ eine Sturmsche Kette*[85] *zu $p \in \Pi$. Bezeichnet man mit $\sigma(x)$ die Anzahl der echten Vorzeichenwechsel in der Folge $p_0(x), \ldots, p_m(x)$, dann ist*

$$\#\{\xi \in [a,b) \ : \ p(\xi) = 0\} = \sigma(b) - \sigma(a). \tag{12.9}$$

Beweis: Wir untersuchen das Verhalten von $\sigma : [a,b) \to \mathbb{N}_0$. Solange $p_j(x) \neq 0$, $j = 0, \ldots, m$, für ein $x \in [a,b)$, ist $\sigma(y) = \sigma(x)$ für alle y aus einer Umgebung von x – die Vorzeichen von p_j, $j = 0, \ldots, m$, ändern sich ja auf dieser Umgebung nicht. Interessant wird es also nur an den Nullstellen.
Betrachten wir also zuerst den Fall, daß $\xi \in [a,b)$ eine Nullstelle von p_0 ist. Da ξ eine einfache Nullstelle ist, ist, für hinreichend kleines $h > 0$ entweder

$$p(\xi - h) < 0 < p(\xi + h) \qquad \Longleftrightarrow \qquad p'(\xi) > 0,$$

oder

$$p(\xi - h) > 0 > p(\xi + h) \qquad \Longleftrightarrow \qquad p'(\xi) < 0.$$

[83] Das ist die Definitionsgleichung für p_j.
[84] Wir ersetzen j durch $j - 1$.
[85] Dies heißt automatisch, daß p nur einfache Nullstellen besitzt!

Wegen (12.6) ist also entweder

$$\begin{array}{c|ccc} & (\xi-h) & (\xi) & (\xi+h) \\ \hline p_0 & <0 & =0 & >0 \\ p_1 & <0 & <0 & <0 \end{array} \quad \text{oder} \quad \begin{array}{c|ccc} & (\xi-h) & (\xi) & (\xi+h) \\ \hline p_0 & >0 & =0 & <0 \\ p_1 & >0 & >0 & >0 \end{array} \tag{12.10}$$

Ist ξ auch eine Nullstelle von p_j, $j>1$, dann haben nach (12.7) p_{j-1} und p_{j+1} dort echt unterschiedliche Vorzeichen und es ist für hinreichend kleines $h>0$

$$\operatorname{sgn} p_j(\xi-h) = \left\{ \begin{array}{c} \operatorname{sgn} p_{j-1}(\xi-h) \\ \operatorname{sgn} p_{j+1}(\xi-h) \end{array} \right\} \qquad \Longleftrightarrow \qquad \operatorname{sgn} p_j(\xi+h) = \left\{ \begin{array}{c} \operatorname{sgn} p_{j+1}(\xi+h) \\ \operatorname{sgn} p_{j-1}(\xi+h) \end{array} \right\},$$

was bedeutet, daß beim Übergang von $x-h$ zu $x+h$ keine neuen Vorzeichenwechsel eingeführt werden. Zusammen mit (12.10) heißt dies also, daß

$$\sigma(\xi+h) = \left\{ \begin{array}{ll} \sigma(\xi-h) & Z(p)\cap(\xi-h,\xi+h)=\emptyset, \\ \sigma(\xi-h)+1 & \#\left(Z(p)\cap(\xi-h,\xi+h)\right)=1, \end{array} \right. \tag{12.11}$$

falls h wieder klein genug ist, wobei

$$Z(p) = \{\xi\in\mathbb{R} \,:\, p(\xi)=0\}.$$

Berücksichtigt man noch, daß eine Nullstelle ξ bei $\sigma(\xi)$ noch *nicht* "mitgezählt" wird[86], so erhält man, daß

$$\sigma(b) = \sigma(a) + \#\left(Z(p)\cap[a,n)\right),$$

also (12.9). □

Übung 12.2 Implementieren Sie die Bestimmung einer Sturmschen Kette und setzen Sie diese ein, um "gute" Startwerte für ein Sekantenverfahren zu finden.

Beispiel 12.15 *Sehen wir uns die ganze Sache mal anhand eines einfachen Beispiels an, nämlich des Polynoms $p(x)=x(x+1)(x-1)=x^3-x$, das hat brav einfache und gut getrennte Nullstellen.*
Als erstes bestimmen wir die zu p gehörige Sturmsche Kette gemäß Proposition 12.13 als[87]

$$\begin{aligned} p_0(x) &= x^3-x \\ p_1(x) &= -p_0'(x) = 1-3x^2 \\ p_2(x) &= -\left(p_0\right)_{p_1} = -\left(-\frac{x}{3}p_1-\frac{2}{3}x\right)_{p_1} = \frac{2}{3}x \\ p_3(x) &= -\left(p_1\right)_{p_2} = -\left(\frac{9}{2}p_2+1\right)_{p_2} = -1 \end{aligned}$$

[86] Es werden nur strikte, also "richtige" Vorzeichenwechsel berücksichtigt.
[87] Dazu führen wir die Notataion $(p)_q$ für p *modulo* q, also den Divisionsrest r bei $p=\omega\, q+r$, ein.

und da das konstante Polynom sein Vorzeichen nicht wechselt, bilden $p_0, \ldots, p_3$ eine Sturmsche Kette.
Interessiert uns jetzt beispielsweise die Anzahl der Nullstellen im Intervall $[-2, 2)$, so bilden wir

$$\boldsymbol{p}(-2) := \begin{bmatrix} p_0(-2) \\ p_1(-2) \\ p_2(-2) \\ p_3(-2) \end{bmatrix} = \begin{bmatrix} -6 \\ -11 \\ -\frac{4}{3} \\ -1 \end{bmatrix} \qquad \textit{und} \qquad \boldsymbol{p}(2) = \begin{bmatrix} 6 \\ -11 \\ \frac{4}{3} \\ -1 \end{bmatrix},$$

also $\sigma(-2) = 0$ und $\sigma(2) = 3$, weswegen in diesem Intervall auch alle drei Nullstellen liegen müssen.
Anders wird die Sache schon auf $\left[-\frac{1}{2}, \frac{1}{2}\right]$, wo wir

$$\boldsymbol{p}\left(-\frac{1}{2}\right) = \begin{bmatrix} \frac{3}{8} \\ \frac{1}{4} \\ -\frac{1}{3} \\ -1 \end{bmatrix}, \boldsymbol{p}\left(\frac{1}{2}\right) = \begin{bmatrix} -\frac{3}{8} \\ \frac{1}{4} \\ \frac{1}{3} \\ -1 \end{bmatrix}, \qquad \textit{also} \qquad \sigma\left(-\frac{1}{2}\right) = 1,\ \sigma\left(\frac{1}{2}\right) = 2$$

erhalten – richtig, in diesem Intervall liegt genau eine Nullstelle.
Bleibt noch der "interessante" Fall $[-1, 1]$ mit

$$\boldsymbol{p}(-1) = \begin{bmatrix} 0 \\ -2 \\ -\frac{2}{3} \\ -1 \end{bmatrix}, \boldsymbol{p}(1) = \begin{bmatrix} 0 \\ -2 \\ \frac{2}{3} \\ -1 \end{bmatrix}, \qquad \sigma(-1) = 0,\ \sigma(1) = 2,$$

*und da ja nur die **echten** Vorzeichenwechsel gezählt werden, finden wir eben auch nur zwei Nullstellen, die dritte ist je gerade der rechte Rand des Intervalls – in der Tat ist für $\varepsilon > 0$*

$$\boldsymbol{p}(1+\varepsilon) = \begin{bmatrix} 2\varepsilon + 3\varepsilon^2 + \varepsilon^3 \\ -2 - 2\varepsilon - \varepsilon^2 \\ \frac{2}{3} + \frac{2}{3}\varepsilon \\ -1 \end{bmatrix} = \begin{bmatrix} + \\ - \\ + \\ - \end{bmatrix},$$

was dann auch immer die geforderten drei echten Vorzeichenwechsel hat.

Es bleibt noch ein Problem, zumindest auf den ersten Blick: Die Sturmschen Ketten funktionieren ja nur für einfache Nullstellen, was aber tun bei mehrfachen Nullstellen. Natürlich könnte man diese durch zufällige Störungen der Keoffizienten auflösen, aber das ist nicht wirklich das, was man tun sollte, denn natürlich ist dieser Ansatz weder übermäßig stabil noch befriedigt er konzeptionell. Außerdem braucht man ihn auch nicht. Ist nämlich

$$p(x) = (x - \xi_1)^{\mu_1} \cdots (x - \xi_m)^{\mu_m}\ q(x), \qquad q(\xi_j) \neq 0, \quad j = 1, \ldots, m,$$

ein Polynom, das die reellen Nullstellen $\xi_1, \ldots, \xi_m$ mit den Vielfachheiten $\mu_1, \ldots, \mu_m$ besitzt[88], dann ist

$$p'(x) = (x - \xi_1)^{\mu_1} \cdots (x - \xi_m)^{\mu_m}\ q'(x) + \sum_{j=1}^{m} \mu_j \frac{p(x)}{x - \xi_j} = (x - \xi_1)^{\mu_1 - 1} \cdots (x - \xi_m)^{\mu_m - 1}\ \widetilde{q}(x)$$

[88] Die konjugiert komplexen Nullstellen schieben wir in q.

und somit ist

$$\text{ggT}\ (p, p') = (x - \xi_1)^{\mu_1 - 1} \cdots (x - \xi_m)^{\mu_m - 1}\ r(x)$$

für ein passendes Polynom r. Das interessiert uns gar nicht so sehr, viel interessanter ist die Beobachtung, daß nun

$$\frac{p(x)}{\text{ggT}\ (p, p')\,(x)} = (x - \xi_1) \cdots (x - \xi_m)\,\frac{q(x)}{r(x)}$$

ein Polynom ist, das nur einfach Nullstellen an $\xi_1, \ldots, \xi_m$ hat – schließlich ist ja immer noch $q\,(\xi_j) \neq 0$ für $j = 1, \ldots, m$. Und wie man den größten gemeinsamen Teiler bestimmt, das wissen wir ja inzwischen.
Das ist aber noch nicht alles. Mit Hilfe der Sturmschen Ketten kann man zeigen, daß alle Polynomfolgen, die durch eine betimmte einfache rekursive Vorschrift definiert sind, ein sehr interessantes Nullstellenverhalten aufweist.

Satz 12.16 *Sei die Polynomfolge f_n, $n \in \mathbb{N}$, definiert durch*

$$f_0(x) = 1, \qquad f_{n+1}(x) = (x + \beta_n)\, f_n(x) + \gamma_n\, f_{n-1}(x), \quad n \in \mathbb{N}_0, \tag{12.12}$$

mit beliebig vorgegebener Folge $\beta_n \in \mathbb{R}$ und $\gamma_n < 0$, $n \in \mathbb{N}$.

1. *Jedes der Polynome f_n hat genau n einfache, relle Nullstellen.*

2. *Die Nullstellen sind* geschachtelt: *Zwischen je zwei benachbarten Nullstellen von f_n liegt eine Nullstelle von f_{n-1}.*

Wir werden den Satz beweisen, indem wir ihn in ein paar einfachere Teilresultate zerlegen, die alle auch für sich von gewissem Interesse sind. Doch zuerst bemerken wir, daß die in (12.12) definierte Polynomfolge aus *monischen* Polynomen vom Grad n besteht: $f_n(x) = x^n + \cdots$, $n \in \mathbb{N}$. Was das Ganze mit Sturmschen Ketten zu tun hat, sehen wir sofort.

Proposition 12.17 *Für $m \in \mathbb{N}_0$ ist die Folge*

$$p_j = (-1)^j f_{m-j}, \qquad j = 0, \ldots, m, \tag{12.13}$$

eine Sturmsche Kette für f_m.

Zum Beweis dieser Proposition müssen wir natürlich "nur" (12.6) und (12.7) verifizieren, die beiden anderen Eigenschaften sind offensichtlich, da $p_0 = f_m$ ist und $p_m = f_0 = 1$ natürlich keine Nullstellen hat.

Lemma 12.18 *Die Folge $p_0, \ldots, p_m$ aus (12.13) erfüllt (12.7).*

Beweis: Wir erhalten für eine beliebige Nullstelle ξ von $p_j = (-1)^j f_{m-j}$, daß

$$\begin{aligned} p_{j+1}(\xi)\, p_{j-1}(\xi) &= (-1)^{2j} f_{m-j-1}(\xi)\, f_{m-j+1}(\xi) \\ &= f_{m-j+1}(\xi) \left((\xi + \beta_{m-j+1}) \underbrace{f_{m-j}(\xi)}_{=0} + \gamma_{m-j} f_{m-j-1}(\xi) \right) \\ &= \gamma_{m-j} f^2_{m-j-1}(\xi) \leq 0 \end{aligned}$$

mit Gleichheit genau dann, wenn $f_{m-j+1}(\xi) = p_{j-1}(\xi) = 0$ ist. Hätten aber f_{m-j} und f_{m-j+1} eine gemeinsame Nullstelle ξ, dann sagt uns (12.12), daß

$$\underbrace{\gamma_{m-j}}_{<0} f_{m-j-1}(\xi) = \underbrace{f_{m-j+1}(\xi)}_{=0} - (\xi + \beta_{m-j}) \underbrace{f_{m-j}(\xi)}_{=0} = 0$$

sein muss, also ξ auch eine Nullstelle von f_{m-j-1} und damit letztendlich von f_0 sein müsste. Das ist aber ein offensichtlicher Widerspruch zu $f_0 \equiv 1$ und somit ist (12.7) erfüllt. □

Wenn wir uns den Beweis eben nochmal genau ansehen, dann haben wir, wie bei den Sturmschen Ketten, bei der durch (12.12) definierten Rekursionsfolge eine interessante Eigenschaft gefunden.

Korollar 12.19 *Zwei aufeinanderfolgende Polynome f_n und f_{n+1} aus (12.12) haben keine gemeinsame Nullstelle.*

Die zweite Eigenschaft, (12.6), geht nicht so direkt, weswegen wir uns zuerst einmal etwas anderes ansehen wollen, nämlich die Größen $\sigma_+(f, I)$ und $\sigma_-(f, I)$, die messen, wieviele **echte** Vorzeichenwechsel von $+$ nach $-$ bzw. von $-$ nach $+$ die Funktion f im Intervall I die Funktion f hat. Formal:

$$\sigma_\pm = \#\left\{x \in I \,:\, \pm f(x-) > 0 > \pm f(x+)\right\}.$$

Achtung! Bei $\sigma_\pm$ gibt es für *rationale Funktionen*[89] $f = p/q$ *zwei* Möglichkeiten, daß diese Funktionen wachsen können, nämlich entweder einen Vorzeichenwechsel von f, also einen Vorzeichenwechsel von p, oder aber einen *Pol* von f, also einen Vorzeichenwechsel des Nenners q.

Lemma 12.20 *Für $I = [a, b]$ und festes m ist*

$$\sigma_-\left(\frac{f_m}{f_{m-1}}, I\right) - \sigma_+\left(\frac{f_m}{f_{m-1}}, I\right) = \sigma(b) - \sigma(a), \tag{12.14}$$

wobei σ wieder die echten Vorzeichenwechsel im Vektor $p_0, \ldots, p_m$ zählt.

Beweis: Erinnern wir uns zuerst einmal an den Beweis von Satz 12.14, wo die Eigenschaft (12.7) dafür sorgte, daß sich σ wenn überhaupt nur an den Vorzeichenwechseln von $p_0 = f_m$

[89] Und solche werden wir gleich betrachten

verändern kann. Da p_1 dort zumindest keine Nullstelle haben kann, gibt es insgesamt vier Möglichkeiten für solch einen Vorzeichenwechsel, an dem sich σ oder $\sigma_\pm$ verändern[90] kann:

f_m	+	0	-	-	0	+	+	0	-	-	0	+
f_{m-1}	+	+	+	+	+	+	-	-	-	-	-	-
$\sigma_\pm$	$\sigma_+ + 1$			$\sigma_- + 1$			$\sigma_- + 1$			$\sigma_+ + 1$		
p_0	+	0	-	-	0	+	+	0	-	-	0	+
p_1	-	-	-	-	-	-	+	+	+	+	+	+
σ	$\sigma - 1$			$\sigma + 1$			$\sigma + 1$			$\sigma - 1$		

(12.15)

Wenn wir jetzt noch bemerken, daß (12.14) für $a = b$ offensichtlich korrekt ist, dann können wir uns für allgemeines b einfach wieder von a nach b bewegen und stellen fest, daß, wann immer σ_+ wächst, auch σ wächst und daß sich σ um eins verringert, wenn σ_- wächst. □

Lemma 12.21 *Für a klein genug und b groß genug*[91] *ist*

$$\sigma(a) = 0, \qquad \sigma(b) = m.$$

Beweis: Für festes m und $j = 0, \ldots, m$ ist ja

$$p_j(x) = (-1)^j f_{m-j}(x) = (-1)^m\,(-1)^{m-j}x^{m-j} + \cdots = (-1)^m(-x)^{m-j} + \cdots,$$

weswegen

$$\lim_{x\to-\infty} \operatorname{sgn} p_j(x) = (-1)^m, \qquad \lim_{x\to-\infty} \operatorname{sgn} p_j(x) = (-1)^j, \qquad j = 0, \ldots, m$$

sein muss, so daß für hinreichend kleines a der Vektor $\sigma(a)$ keinen, für hinreichend großes b hingegen der Vektor $\sigma(b)$ die maximalen m Vorzeichenwechsel hat. □

Lemma 12.22 *Ist $I = [a, b]$ ein Intervall, das alle Nullstellen von f_m enthält, dann ist*

$$\sigma_-\left(\frac{f_m}{f_{m-1}}, I\right) = m \qquad \textit{und} \qquad \sigma_+\left(\frac{f_m}{f_{m-1}}, I\right) = 0.$$

Beweis: Wegen der Lemmata 12.20 und 12.21 ist

$$\sigma_+\left(\frac{f_m}{f_{m-1}}, I\right) - \sigma_-\left(\frac{f_m}{f_{m-1}}, I\right) = \sigma(b) - \sigma(a) = m - 0 = m$$

und da außerdem und offensichtlicherweise $0 \le \sigma_\pm\left(f_m/f_{m-1}, I\right) \le m$ gelten muss, geht das eben nur für $\sigma_+ = m$ und $\sigma_- = 0$. □

[90] Aber immer nur **entweder** σ_+ **oder** σ_-!

[91] Stichwort: Gerschgorinsche Kreise, das sind also berechenbare Größen!

Beweis von Proposition 12.17: Jetzt ist (12.7) keine Hexerei mehr, denn nach Lemma 12.22 bleiben in der Tabelle aus (12.15) nur zwei Möglichkeiten übrig, nämlich die, bei denen σ_+ wächst und aus diesen lässt sich sofort ablesen, daß an jeder Nullstelle ξ von f_m

$$\operatorname{sgn} f'_m(\xi) = \operatorname{sgn} f_m(\xi), \qquad \text{d.h.,} \qquad \operatorname{sgn} p'_0(\xi) = \operatorname{sgn} p_1(\xi)$$

sein muss. □

Bemerkung 12.23 ***Achtung:*** *Die Eigenschaft* $\operatorname{sgn} f'_m(\xi) = \operatorname{sgn} f_m(\xi)$ *ist* ***nicht*** *das Gegenteil der Eigenschaft (12.6), sondern eine ganz analoge Eigenschaft! Wichtig bei der ganzen Geschichte ist nämlich, daß an* ***allen*** *Nullstellen von* f_m *die Funktion* f'_m/f_{m-1} *dasselbe Vorzeichen hat, ob dies nun* $+$ *oder* $-$ *ist, ist hingegen eher irrelevant.*

Beweis von Satz 12.16: Wie haben bereits gesehen, daß für $m \in \mathbb{N}_0$ das Polynom f_m die vollen m einfachen reellen Nullstellen haben muss, die dann auch abwechselnd Vorzeichenwechsel umgekehrter Richtung sein müssen. Nach unserer Tabelle (12.15) muss aber an zwei aufeinanderfolgenden Vorzeichenwechsel unterschiedlicher "Richtung" auch f_{m-1} sein Vorzeichen gewechselt haben - es muss also zwischen jeder der m Nullstellen mindestens eine Nullstelle von f_{m-1} liegen. Da das aber bereits $m-1$ Nullstellen vom f_{m-1} festlegt und f_{m-1} gar nicht mehr Nullstellen haben kann, folgt die Behauptung des Satzes. □

Wer eine Wissenschaft noch nicht so innehat, daß er jeden Verstoß dagegen fühlt wie einen grammatikalischen Fehler in seiner Muttersprache, der hat noch viel zu lernen.

G. Ch. Lichtenberg

13 Numerische Integration und orthogonale Polynome

In vielen praktischen Anwendungen, beispielsweise bei der Diskretisierung von Differentialgleichungen oder auch nur bei der Bestimmung von Waveletkoeffizienten[92] müssen *Integrale* berechnet werden. Leider lassen sich nur die wenigsten Funktionen, beispielsweise Polynome, wirklich *explizit* integrieren, für allgemeinere Funktionen muß man *numerische* Methoden finden. Oftmals verwenden numerische Integrationsmethoden sogar die Struktur der zugrundeliegenden Funktionen – so gibt es beispielsweise spezielle Methoden zur Integration von verfeinerbaren Funktionen.

Das Ziel in diesem Abschnitt soll es sein, (eigentliche) Integrale der Form

$$I(f) = \int_a^b f(x)\, w(x)\, dx, \qquad w(x) \geq 0,\, x \in [a,b], \tag{13.1}$$

bezüglich einer stetigen[93] *nichtnegativen* Gewichtsfunktion[94] w näherungsweise zu berechnen. Das "richtige" Kriterium ist hier nicht die Positivität oder Nichtnegativität der Gewichtsfunktion sondern die *strikte Quadratpositivität* des *Funktionals*[95] I, die fordert, daß

$$I\left(f^2\right) > 0, \qquad f \neq 0. \tag{13.2}$$

Das und nichts anderes ist die Forderung, die man wirklich an die Gewichtsfunktion w stellt. Ist w stetig, so muss natürlich $w \geq 0$ sein und w darf nicht "zu oft" den Wert Null annehmen, denn dann könnte es ja $f \neq 0$ mit $I\left(f^2\right) = 0$ geben.

[92] Siehe z.B. [36, S. 172]
[93] Der Einfachheit halber!
[94] Der bei weitem wichtigste Fall ist dabei natürlich $w \equiv 1$.
[95] Ein Funktional ist eine (lineare) Abbildung von einem Funktionenraum in einen Körper.

13.1 Quadraturformeln

Definition 13.1 *Eine* Quadraturformel Q_n *mit den* Knoten $x_0, \ldots, x_n$ *und den* Gewichten $w_0, \ldots, w_n$ *ist ein lineares Funktional der Form*

$$Q_n(f) = \sum_{j=0}^{n} w_j \; f\left(x_j\right), \qquad f \in C[a,b]. \tag{13.3}$$

Eine Quadraturformel Q_n *hat* Exaktheitsgrad m *für ein Integral* I*, falls*

$$Q_n(p) = I(p), \qquad p \in \Pi_m. \tag{13.4}$$

Die Forderung $f \in C[a,b]$ in (13.3) wird deswegen gemacht, daß die Punktauswertung überhaupt eine sinnvolle Aktion auf f ist (eine lediglich integrierbare Funktion kann man ja auf einer Nullmenge beliebig abändern).

Bemerkung 13.2 *Oftmals fordert man, daß die Knoten der Quadraturformel alle im Intervall* $[a,b]$ *liegen und daß die Gewichte alle nichtnegativ sind. Im allgemeinen ist es aber nicht so einfach, diese beiden Forderungen zu erfüllen, ironischerweise passiert dies aber automatisch bei den "besten" Quadraturformeln, der sogenannten* Gauß–Quadratur[96]

Definition 13.3 *Die* Momente μ_j*,* $j \in \mathbb{N}_0$*, des Integrals* I *sind definiert als*

$$\mu_j = I\left((\cdot)^j\right) = \int_a^b x^j \; w(x) \; dx.$$

Analog definiert man die Momente der Quadraturformel als $\gamma_{n,j} = Q_n\left((\cdot)^j\right)$.

Was bringt uns nun dieser Exaktheitsgrad? Auf alle Fälle einmal Abschätzungen der Approximationsgüte der Quadraturformel. Zu diesem Zweck definieren wir

$$E_n(f) = I(f) - Q_n(f), \qquad f \in C\left[a,b\right],$$

und erhalten die folgende (einfache) Aussage.

Proposition 13.4 *Ist* Q_n *eine Quadraturformel für* I *vom Exaktheitsgrad* $m \geq 0$ *mit nichtnegativen Gewichten und ist* $f \in C^{m+1}[a,b]$*, dann ist*

$$|E_n(f)| \leq 2\mu_0 \; \frac{(b-a)^{m+1}}{(m+1)!} \left\| f^{(m+1)} \right\|_\infty \tag{13.5}$$

[96] Dazu aber später noch mehr.

Beweis: Ist $p \in \Pi_m$ ein *beliebiges* Polynom, dann ist

$$\begin{aligned}
|E_n(f)| &= |I(f) - Q_n(f)| = |I(f) \underbrace{-I(p) + Q_n(p)}_{=0} - Q_n(f)| = |I(f-p) - Q_n(f-p)| \\
&\leq |I(f-p)| + |Q_n(f-p)| \\
&\leq \int_a^b |f(x) - p(x)| \; w(x)\, dx + \sum_{j=0}^{n} w_j \; |f(x_j) - p(x_j)| \\
&\leq \underbrace{\left(\int_a^b w(x)\, dx + \sum_{j=0}^{n} w_j \right)}_{=2\mu_0} \|f-p\|_\infty ,
\end{aligned}$$

da

$$\mu_0 = I(1) = Q_n(1) = \sum_{j=0}^{n} w_j .$$

Wählt man p als das Interpolationspolynom an *irgendwelchen* Punkten[97] $\xi_0, \ldots, \xi_m \in [a,b]$, siehe [36, S. 128], und verwendet man die Fehlerformel

$$f(x) - L_m f(x) = \frac{f^{(m+1)}(\xi)}{(m+1)!} (x-\xi_0) \cdots (x-\xi_m) , \qquad x, \xi \in [a,b],$$

aus [36, (7.40)], so ergibt sich (13.5). □

Bemerkung 13.5 *(Exaktheit)*

1. *Ist $[a,b] = [-1,1]$, so kann man sogar die* Tschebyscheffknoten *zur Interpolation verwenden und erhält die wesentlich bessere Abschätzung*

$$|E_n(f)| \leq 2\mu_0 \, \frac{1}{2^m (m+1)!} \left\| f^{(m+1)} \right\|_\infty .$$

   Außerdem kann man ja immer das Intervall $[a,b]$ auf $[-1,1]$ transformieren, indem man die Variable $t = 2\frac{x-a}{b-a} - 1$ einführt.

2. *Der Beweis zeigt auch, daß*

$$|E_n(f)| \leq 2\mu_0 \min_{p \in \Pi_m} \|f-p\|_\infty \to 0$$

   für wachsendes m, nach dem Satz von Weierstraß[98].

[97] Natürlich gilt: Je besser die Punkte, desto besser die Abschätzung.
[98] Z.B. aus [36, Satz 7.3]

3. *Die einzige Konstante, die in (13.5) eingeht, ist μ_0, ein Wert, der nur vom* Integral *selbst abhängt. Findet man also für ein vorgegebenes Integral I Quadraturformeln von* hohem *Exaktheitsgrad, dann kann man für "gute" Funktionen auch auf eine gute näherungsweise Berechnung des Integrals hoffen, vorausgesetzt, man erfüllt die Bedingung an die Nichtnegativität der Gewichte. Anderfalls erhielte man in (13.5) einen Term der Form*

$$\sum_{j=0}^{n} |w_j|$$

und der hängt nun leider sehr wohl von der Quadraturformel ab und kann noch dazu sehr stark wachsen.

"Gute" Quadraturformeln wären nach Proposition 13.4 also solche, die nichtnegative Gewichte *und* hohen Exaktheitsgrad haben. Dabei gibt es natürlich Einschränkungen bezüglich der Anzahl der Knoten.

Proposition 13.6 *(Exaktheitsgrad von Quadraturformeln)*
Für ein Integral I gilt:

1. *Jede Quadraturformel Q_n für I mit $n+1$ Knoten hat höchstens Exaktheitsgrad $2n+1$.*
2. *Jede Quadraturformel Q_n für I mit $n+1$ Knoten und Exaktheitsgrad $2n$ hat positive Gewichte.*

Beweis: Sei $\omega(x) = (x - x_0) \cdots (x - x_n)$. Dann ist $\omega^2 \in \Pi_{2n+2}$ und es gilt[99]

$$I\left(\omega^2\right) = \int_a^b \omega^2(t)\ w(t)\ dt > 0 = \sum_{j=0}^{n} w_j\ \omega^2(x_j) = Q_n\left(\omega^2\right),$$

was 1. beweist. Für 2. verwenden wir eine Idee von Stieltjes[100] und setzen

$$\ell_j(x) = \prod_{k \neq j} \frac{x - x_k}{x_j - x_k}, \qquad j = 0, \ldots, n, \qquad x \in \mathbb{R},$$

das heißt, es ist $\ell_j \in \Pi_n$ und $\ell_j(x_k) = \delta_{jk}$, $j, k = 0, \ldots, n$. Hat nun die Quadraturformel (mindestens) Exaktheitsgrad $2n$, dann ist

$$0 < \int_a^b \ell_j^2(t)\ w(t)\ dt = I\left(\ell_j^2\right) = Q_n\left(\ell_j^2\right) = \sum_{k=0}^{n} w_k \underbrace{\ell_j^2(x_k)}_{=\delta_{jk}} = w_j, \qquad j = 0, \ldots, n.$$

□

[99] Wäre $I\left(\omega^2\right) = 0$, dann wäre die "Gewichtsfunktion" w die Nullfunktion – diesen Trivialfall wollen wir ausschließen.

[100] Siehe [13, S. 163]

Es gibt nun einen "einfachen" Weg, Quadraturformeln vom Exaktheitsgrad n zu bekommen, den wir auch im Beweis der vorhergehenden Proposition schon verwendet haben: Man interpoliert f an den Punkten $x_0, \ldots, x_n$ durch ein Polynom vom Grad n, also

$$L_n f(x) = \sum_{j=0}^{n} f(x_j)\ \ell_j(x), \qquad x \in \mathbb{R},$$

und setzt

$$\begin{aligned} Q_n(f) &= I(L_n f) = \int_a^b \left(\sum_{j=0}^{n} f(x_j)\ \ell_j(t) \right) w(t)\ dt = \sum_{j=0}^{n} \left(\int_a^b \ell_j(t) w(t)\ dt \right) f(x_j) \\ &= \sum_{j=0}^{n} \underbrace{I(\ell_j)}_{=:w_j}\ f(x_j)\,. \end{aligned}$$

Die so erhaltenen Quadraturformeln bezeichnet man dann als *interpolatorisch*. Da $L_n : C[a,b] \to \Pi_n$ eine *Projektion* ist, ist $L_n p = p$ für alle $p \in \Pi_n$ und damit

$$Q_n(p) = I(L_n p) = I(p),$$

also hat Q_n (mindestens) Exaktheitsgrad n. Nun haben diese Quadraturformeln eben im allgemeinen keine nichtnegativen Gewichte mehr.

Definition 13.7 *Sei I ein Integral und Q_n eine Quadraturformel dazu. Dann nennt man die Quadraturformel Q_n*

1. interpolatorisch*, wenn*
$$w_j = I(\ell_j), \qquad j = 0, \ldots, n.$$
2. Gauß–Formel*, wenn Q_n den* maximalen *Exaktheitsgrad $2n+1$ hat.*

Mit dieser Terminologie können wir wie in alle Quadraturformeln Q_n vom Exaktheitsgrad $\geq n$ charakterisieren, siehe [13, Theorem 3.2.1].

Satz 13.8 *Seien I ein Integral und Q_n eine Quadraturformel dazu. Dann hat Q_n genau dann den Exaktheitsgrad $n+k$, $k = 0, \ldots, n+1$, wenn Q_n interpolatorisch ist und*

$$0 = I(\omega \cdot p) = \int_a^b \omega(t)\ p(t)\ w(t)\ dt, \qquad p \in \Pi_{k-1}, \qquad \omega(x) = \prod_{j=0}^{n} (x - x_j)\,. \tag{13.6}$$

Bemerkung 13.9 (Quadratur, Knoten und Exaktheit)

1. *Gleichung (13.6) ist eine Bedingung an die* Knoten x_j*, $j = 0, \ldots, n$, die passend gewählt werden müssen, um Exaktheit größer als n zu liefern – Exaktheitsgrad n ist ja bei allen interpolatorischen Formeln "eingebaut". Mehr zu diesem Thema in Kapitel 13.4.*

2. *Wir können Satz 13.8 auch noch anders interpretieren: Es ist die Bestimmung der Knoten, was zählt, die Gewichte ergeben sich für jede "halbwegs gute" Quadraturformel dann einfach aus der Tatsache, daß diese interpolatorisch sein* muss. *Und genau das ist ja auch das Thema bei der Gauß–Quadratur: Wie bestimmt man die optimalen Knoten?*

Beweis von Satz 13.8: Sei Q_n eine Quadraturformel vom Exaktheitsgrad $n+k \geq n$. Insbesondere ist also für $j = 0, \ldots, n$,

$$I(\ell_j) = Q_n(\ell_j) = \sum_{k=0}^{n} w_k \underbrace{\ell_j(x_k)}_{=\delta_{jk}} = w_j,$$

also ist Q_n interpolatorisch. Außerdem ist[101] für jedes $p \in \Pi_{k-1}$ das Produkt[102] $p \cdot \omega$ in Π_{n+k} und verschwindet[103] an x_j, $j = 0, \ldots, n$. Wegen des Exaktheitsgrads $k+n$ ist damit

$$I(\omega \cdot p) = Q_n(\omega \cdot p) = \sum_{j=0}^{n} w_j \underbrace{(\omega \cdot p)(x_j)}_{=0} = 0,$$

woraus auch (13.6) folgt.
Sei nun Q_n eine interpolatorische Quadraturformel für I, die (13.6) erfüllt und sei $p \in \Pi_{n+k}$. Wir schreiben p als[104] $p = q\omega + L_n p$, und erhalten, daß

$$I(p) = I(q \cdot \omega + L_n p) = \underbrace{I(q \cdot \omega)}_{=0} + \underbrace{I(L_n p)}_{=Q_n(L_n p)} = Q_n(p).$$

□

13.2 Klassische Formeln

Die "klassischen" Quadraturformeln für $w \equiv 1$ sind die sogenannten *Newton–Cotes*–Formeln, die man durch Polynominterpolation an *gleichverteilten*[105] Interpolationsknoten im Intervall $[a, b]$ erhält – diese Konstruktionen gehen tatsächlich für den Fall $[a, b] = [-1, 1]$ und $w \equiv 1$ auf Newton (1687) bzw. Cotes[106] (1712) zurück. Setzt man

$$h = \frac{b-a}{n} \qquad \text{und} \qquad t = \frac{x-a}{h},$$

[101] Mit der Konvention, daß $\Pi_{-1} = \{0\}$.
[102] Zur Erinnerung: $\omega \in \Pi_{n+1}$
[103] Ist $k = 0$, dann ist $p = p \cdot \omega = 0$.
[104] Diese kleine Formel ist so einfach wie genial: Algebraisch gesehen ist Polynominterpolation also nichts anderes als der Rest bei Division durch das *Knotenpolynom* ω beziehungsweise, noch hochgestochener formuliert, die Normalform modulo des (Haupt-) Ideals $\langle \omega \rangle$.
[105] Und diese Knotenwahl ist nicht gerade optimal!
[106] Roger Cotes (1682–1716) bearbeitete die zweite Auflage von I. Newtons *Principia*.

dann wird $x \in [a,b]$ in $t \in [0,n]$ transformiert und es ist

$$\ell_j(x) = \prod_{k\neq j} \frac{x - \overbrace{(a+kh)}^{=x_k}}{a+jh-a-kh} = \prod_{k\neq j} \frac{a+th-a-kh}{h\,(j-k)} = \prod_{k\neq j} \frac{t-k}{j-k},$$

also, da $w \equiv 1$,

$$w_j = \int_a^b \ell_j(x)\,dx = h\int_0^n \prod_{k\neq j} \frac{t-k}{j-k}\,dt.$$

Man sieht sofort, daß diese Quadraturformeln *rationale* Gewichte haben also z.B. mit `Maple` bestimmt und verwendet werden können. Die Newton–Cotes–Formeln kann man nun für kleine Werte von n explizit angeben – einige von ihnen haben sogar einen eigenen Namen; die folgende Tabelle stammt aus [45, S. 108], alle Gewichte sind immer noch mit der Intervalllänge $b-a$ zu multiplizieren.

n	Gewichte							Name
1	$\frac{1}{2}$	$\frac{1}{2}$						Trapezregel
2	$\frac{1}{6}$	$\frac{2}{3}$	$\frac{1}{6}$					Simpson–Regel
3	$\frac{1}{8}$	$\frac{3}{8}$	$\frac{3}{8}$	$\frac{1}{8}$				"Pulcherrima", $3/8$–Regel
4	$\frac{7}{90}$	$\frac{32}{90}$	$\frac{12}{90}$	$\frac{32}{90}$	$\frac{7}{90}$			Milne–Regel
5	$\frac{19}{288}$	$\frac{75}{288}$	$\frac{50}{288}$	$\frac{50}{288}$	$\frac{75}{288}$	$\frac{19}{288}$		
6	$\frac{41}{840}$	$\frac{216}{840}$	$\frac{27}{840}$	$\frac{272}{840}$	$\frac{27}{840}$	$\frac{216}{840}$	$\frac{41}{840}$	Weddle–Regel

Die Liste bricht nicht ganz ohne tieferen Grund ab: Für $n > 6$ werden einige der Gewichte *negativ*.

Übung 13.1 Bestimmen Sie die Gewichte der Newton–Cotes–Formel für $n = 7$.

Die Simpson–Regel[107] ist auch als "Keplersche Faßregel" bekannt und besitzt, wenn man sich den Grad des zugehörigen Interpolationspolynome ansieht, zwar nur Exaktheitsgrad 2, aber eine Fehlerabschätzung der Form

$$|E_2 f| \leq \frac{h^5}{90} \left\| f^{(4)} \right\|_\infty, \qquad f \in C^4\,[a,b]\,, \tag{13.7}$$

die um eine Größenordnung besser ist als das, was man aus (13.5) erwarten könnte. Der Name "pulcherrima" bedeutet übrigens "die allerschönste". Generell ist die Fehlerordnung der meisten Newton–Cotes–Formeln um einige Größenordnungen besser als man erwarten würde. Das hat einen relativ einfachen Grund ...

[107] Nicht Bart oder Homer, sondern Thomas Simpson (1710–1761), Selfmademathematiker. Die Formel selbst war aber schon Cavalieri (1639) bekannt.

Beispiel 13.10 *Die verbesserte Abschätzung (13.7) für die Simpson–Regel kann man beispielsweise durch* Hermite–Interpolation *beweisen. Dazu verwendet man für $x_0 = a$, $x_1 = a+h$ und $x_2 = b$ das Interpolationspolynom*[108]

$$\begin{aligned} H_3 f(x) &= \underbrace{f(x_0) + (x-x_0)[x_0,x_2]f + (x-x_0)(x-x_2)[x_0,x_1,x_2]f}_{=L_2 f} \\ &\quad + (x-x_0)(x-x_1)(x-x_2)[x_0,x_1,x_1,x_2]f, \end{aligned}$$

das die Bedingungen

$$H_3 f(x_j) = f(x_j), \quad j = 0,1,2, \qquad (H_3 f)'(x_1) = f'(x_1)$$

erfüllt. Da

$$\int_a^b (x-x_0)(x-x_1)(x-x_2)\ dx = 0,$$

ist

$$\int_a^b H_3 f(t)\ dt = \int_a^b L_2 f(t)\ dt = Q_2(f),$$

aber für $f \in C^4[a,b]$ liefert $H_3 f$ den Fehlerterm

$$\begin{aligned} I - Q_2(f) &= \int_a^b (f - H_3(f))(t)\ dt = \int_a^b (t-x_0)(t-x_1)^2(t-x_2)\frac{f^{(4)}}{4!}(\xi(t))\ dt \\ &= \frac{f^{(4)}(\xi)}{4!}\int_0^{2h} t(t-h)^2(t-2h)\ dt = -\frac{h^5}{90}f^{(4)}(\xi), \end{aligned}$$

also die Größenordnung h^5 statt h^3.

13.3 Zusammengesetzte Quadraturformeln

Bereits in der (sehr groben) Fehlerabschätzung (13.5) zeigt sich, daß eine Quadraturformel auf einem *kleinen* Intervall normalerweise genauer sein wird, als auf einem großen Intervall. Sei also Q_n eine Quadraturformel, dann definieren wir zu $N \in \mathbb{N}$ die aufsteigende Punktfolge $x_{j,k}$, $j = 1,\ldots,N$, $k = 0,\ldots,n$, wobei $x_{j,k} < x_{j',k'}$ falls $j < j'$ oder $j = j'$ und $k < k'$, sowie

$$x_{1,0} = a, \qquad x_{j,n} = x_{j+1,0}, \quad j = 1,\ldots,N-1, \qquad x_{N,n} = b,$$

so daß wir eine *überlappende* Zerlegung von $[a,b]$ erhalten. Das Teilintegral auf $[x_{j,0}, x_{j,n}]$ bestimmt man nun jeweils mit der transformierten Quadraturformel Q_n – das setzt natürlich voraus, daß die Knoten $x_{j,k}$ ebenfalls *Transformationen* der Knoten der Quadraturformel Q_n sind. Genauer: Bezeichnet man mit

$$0 = x_0 < x_1 < \cdots < x_{n-1} < x_n = 1$$

[108] Die etwas seltsame Reihenfolge x_0, x_2, x_1, x_1, in der die Interpolationspunkte in die Newton–Darstellung eingegeben werden, ist kein Tipfehler, sondern durchaus Absicht.

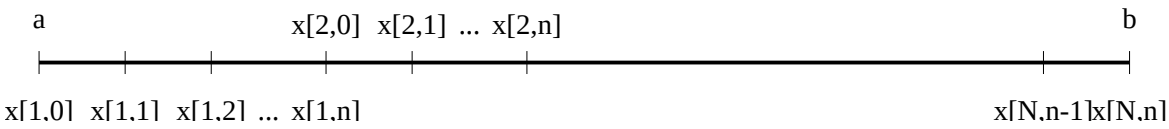


Abbildung 13.1: Zerlegung des Intervalls $[a, b]$ für zusammengesetzte Quadraturformel.

die Knoten von Q_n, dann ist

$$x_k = \frac{x_{j,k} - x_{j,0}}{x_{j,n} - x_{j,0}}, \qquad j = 1, \ldots, N. \tag{13.8}$$

Ausgehend von einer Quadraturformel Q_n für $I = \int_0^1 dt$ ergibt sich dann die zusammengesetzte Formel als

$$Q_{n,N}(f) = \sum_{j=1}^{N} \underbrace{(x_{j,n} - x_{j,0})^{-1}}_{=:h_j^{-1}} Q_n \left(f \left(\frac{\cdot - x_{j,0}}{x_{j,n} - x_{j,0}} \right) \right) = \sum_{j=1}^{N} h_j^{-1} Q_n \left(f \left(h_j^{-1}(\cdot - x_{j,0}) \right) \right). \tag{13.9}$$

Beispiel 13.11 *Im Newton–Cotes–Fall wäre*

$$x_{j,k} = a + \frac{(j-1)(n+1)+k}{Nn}(b-a) = a + h\left((j-1)(n+1)+k\right), \qquad h = \frac{b-a}{Nn}.$$

Damit entspricht die zusammengesetzte Trapezregel gerade der Integration des stückweise linearen Interpolanten.

Das Ziel ist es also, das Integral

$$I(f) = \int_a^b f(t)\, dt = \sum_{j=1}^{N} \int_{x_{j,0}}^{x_{j,n}} f(t)\, dt =: \sum_{j=1}^{N} I_j(f), \qquad f \in C[a, b],$$

anzunähern und eine Abschätzung für den Fehler anzugeben.

Proposition 13.12 *Sei Q_n die Newton–Cotes–Formel für das Integral $\int_0^1 dt$ und $Q_{n,N}$ die daraus gebildete zusammengesetzte Quadraturformel. Dann ist, für $f \in C^{n+1}[a, b]$,*

$$|I(f) - Q_{n,N}(f)| \leq (b-a) \max_{j=1,\ldots,n} \left(h_j^n \frac{\left\| f^{(n+1)} \right\|_{[x_{j,0}, x_{j,n}]}}{n!} \right). \tag{13.10}$$

Bemerkung 13.13 *Aus (13.10) folgt sofort die* globale *Fehlerabschätzung*

$$|I(f) - Q_{n,N}(f)| \leq (b-a) \left(\max_{j=1,\ldots,n} h_j \right)^n \frac{\left\| f^{(n+1)} \right\|_{[a,b]}}{n!}, \tag{13.11}$$

aber dennoch sagt uns (13.10) einiges mehr: Man könnte versuchen, den Gesamtfehler dadurch zu drücken, daß man h_j klein macht, wo der Betrag von $f^{(n+1)}$ sehr groß wird und dafür die Schrittweite in Bereichen groß macht, wo $f^{(n+1)}$ betragsmäßig sehr klein wird. Auf diese Art und Weise erhält man, unter Ausnutzung von zusätzlicher[109] *Information über f möglicherweise bessere Quadraturformeln, ohne die Zahl der Knoten und damit den Rechenaufwand zu vergrößern.*

Derartige Verfahren, die versuchen, ihre Parameter (in diesem Fall die Knoten) aus Informationen über die zu verarbeitenden Daten (in diesem Fall die Funktion, bzw. deren Werte an bestimmten Stellen) zu bestimmen, bezeichnet man als adaptiv . *Dies steht im Gegensatz zu (nichtadaptiven) Verfahren, bei denen die Parameter von Anfang an feststehen – solche Verfahren sind zwar meist etwas schneller, dafür aber weniger flexibel.*

Beweis von Proposition 13.12: Mit der Bezeichnung

$$t_{j,k} := \frac{x_{j,k} - x_{j,0}}{x_{j,n} - x_{j,0}}, \qquad j = 1, \ldots, N,$$

ist, für $j = 1, \ldots, N$,

$$\begin{aligned}
I_j(f) &= \int_{x_{j,0}}^{x_{j,n}} f(t)\, dt = \int_{x_{j,0}}^{x_{j,n}} \left(L_n f(t) + [t, x_{j,0}, \ldots, x_{j,n}]\, f \right)\, dt \\
&= \sum_{k=0}^{n} h_j^{-1} \underbrace{\int_0^1 \ell_k \left(h^{-1}(t + x_{j,k}) \right)\, dt}_{=w_k}\, f\left(x_{j,k}\right) + \int_{x_{j,0}}^{x_{j,n}} [t, x_{j,0}, \ldots, x_{j,n}]\, f\, dt \\
&= h_j^{-1} Q_{n,N} \left(f\left(h_j^{-1}(\cdot - x_{j,0}) \right) \right) + \underbrace{\int_{x_{j,0}}^{x_{j,n}} (t - x_{j,0}) \cdots (t - x_{j,n}) \frac{f^{(n+1)}}{(n+1)!} \left(\xi(t)\right)\, dt}_{=:R_j(f)}
\end{aligned}$$

Damit ist

$$I(f) = \sum_{j=1}^{N} I_j(f) = \sum_{j=1}^{N} h_j^{-1} Q_n \left(f\left(h_j^{-1}(\cdot - x_{j,0}) \right) \right) + \sum_{j=1}^{N} R_j(f) = Q_{n,N}(f) + \sum_{j=1}^{N} R_j(f).$$

[109] Stellt sich natürlich die Frage, wo man diese herbekommt.

Den Fehlerterm $R_j(f)$ spalten wir wie folgt auf und verwenden den Mittelwertsatz der Integration[110]

$$\begin{aligned}
|R_j(f)| &= \left| \sum_{k=0}^{n-1} \int_{x_{j,k}}^{x_{j,k+1}} (x-x_{j,0}) \cdots (x-x_{j,n}) \, \frac{f^{(n+1)}}{(n+1)!} \left(\xi(x)\right) \, dx \right| \\
&= \left| \sum_{k=0}^{n-1} \frac{f^{(n+1)}\left(\xi_k\right)}{(n+1)!} \int_{x_{j,k}}^{x_{j,k+1}} (x-x_{j,0}) \cdots (x-x_{j,n}) \, dx \right| \leq n \, h_j^{n+1} \frac{\left\| f^{(n+1)} \right\|_{[x_{j,0},x_{j,n}]}}{(n+1)!} \\
&\leq h_j^{n+1} \frac{\left\| f^{(n+1)} \right\|_{[x_{j,0},x_{j,n}]}}{n!}.
\end{aligned}$$

Damit ist

$$|I - Q_{n,N}(f)| \leq \sum_{j=1}^{n} |R_j(f)| \leq \left(\max_{j=1,\ldots,n} h_j^n \frac{\left\| f^{(n+1)} \right\|_{[x_{j,0},x_{j,n}]}}{n!} \right) \underbrace{\sum_{j=1}^{n} h_j}_{=b-a}.$$

□

Beispiel 13.14 *Mit der verbesserten Fehlerformel aus Beispiel 13.10 und den Punkten $x_j = a + jh$, $j = 0, \ldots, 2N$, können wir die den Fehler bei der* zusammengesetzten *Simpson–Regel bestimmen:*

$$\begin{aligned}
E_2(f) &= -\sum_{j=0}^{2N-2} \frac{1}{90} h^5 f^{(4)}\left(\xi_j\right) = -\frac{1}{90} h^4 \sum_{j=0}^{2N-2} \underbrace{\frac{b-a}{2N-1}}_{=h} f^{(4)}\left(\xi_j\right) = -\frac{b-a}{90} h^4 \sum_{j=0}^{2N-2} \frac{f^{(4)}\left(\xi_j\right)}{2N-1} \\
&= -\frac{b-a}{90} h^4 f^{(4)}(\xi), \qquad \xi \in \left[\min_{j=0,\ldots,2N-2} \xi_j, \max_{j=0,\ldots,2N-2} \xi_j \right]. \qquad (13.12)
\end{aligned}$$

Die Existenz der "magischen" Stelle ξ ergibt sich dabei aus der Ungleichung

$$\min_{j=0,\ldots,2N-2} f^{(4)}\left(\xi_j\right) \leq \sum_{j=0}^{2N-2} \frac{f^{(4)}\left(\xi_j\right)}{2N-1} \leq \min_{j=0,\ldots,2N-2} f^{(4)}\left(\xi_j\right) \leq$$

für die Konvexkombination *und nach dem Zwischenwertsatz wird deren Wert eben an einer Stelle ξ angenommen, die im kleinsten Intervall liegt, das alle ξ_j, $j = 0, \ldots, 2N-2$, enthält.*

Diese Fehlerabschätzung (13.12) ist natürlich wesentlich besser als die aus Proposition 13.12 – vorausgesetzt natürlich, daß f hinreichend oft differenzierbar ist und daß sich die vierte Ableitung von f im Vergleich zur zweiten Ableitung "brav verhält"[111], sonst muß man h schon sehr klein wählen.

[110] Dafür darf das Polynom keinen Vorzeichenwechsel haben – deswegen die Aufspaltung.

[111] Daß das nicht der Fall sein muß zeigen schon die Monome x^m, m groß, bei denen die vierte Ableitung um einen Faktor $(m-2)(m-3) = O\left(m^2\right)$ größer ist – soviel zum Wert "asymptotischer" Blankoaussagen.

Geometrisch ist die zusammengesetzte Simpsonformel

$$Q_{2,N}(f) = \frac{b-a}{6N}\left(f_0 + 4f_1 + 2f_2 + 4f_3 + 2f_4 + \cdots + 4f_{2N-1} + f_{2N}\right),$$
$$f_j = f\left(a + jh\right),\ j = 0, \ldots, 2N, \qquad h = \frac{b-a}{2N},$$

die Integration der Interpolationsparabeln an den Punkten

$$x_{2j}, x_{2j+1}, x_{2j+2}, \qquad j = 0, \ldots, 2N-2,$$

siehe Abb. 13.2.

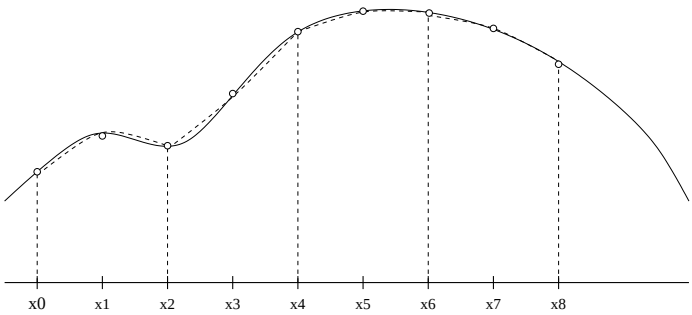


Abbildung 13.2: Geometrische Interpretation der zusammengesetzten Simpson–Formel.

13.4 Gauß–Quadratur

Gaußsche Quadraturformeln sind Quadraturformeln Q_n, die den *maximalen* Exaktheitsgrad $2n+1$ haben. Die Existenz solcher Formeln und ihre Konstruktion wurden bereits 1816 von Gauß [11] angegeben. Das Originalargument von Gauß kann man über die Aufgabe 13.6 am Ende dieses Kapitels selbst nachvollziehen[112], im "Bonuskapitel" 13.7 findet sich auch die Lösung dazu im "Gaußschen Stil".

Wir können uns aber, dank Satz 13.8, das Leben wesentlich leichter machen, denn wir wissen, daß eine Quadraturformel mit Knoten x_j, $j = 0, \ldots, n$, genau dann eine Gaußsche Quadraturformel ist, wenn

1. sie interpolatorisch ist.

2. das Polynom $\omega = (\cdot - x_0) \cdots (\cdot - x_n)$ zu allen Polynomen vom Grad $\leq n$ *orthogonal* ist, das heißt, wenn
$$\int_a^b p(x)\,\omega(x)\,dx = 0, \qquad p \in \Pi_n.$$

[112] Oder es zumindest versuchen.

Damit gehört ω aber zu einer ganz bestimmten und besonders interessanten Klasse von Polynomen.

Definition 13.15 *Ein Polynom $p \in \Pi_n$ heißt* orthogonales Polynom *der Ordnung n bezüglich der Gewichtsfunktion w, wenn $p \not\equiv 0$ und*

$$\int_a^b p(x)\ q(x)\ w(x)\ dx = 0, \qquad q \in \Pi_{n-1}.$$

Das heißt, daß p orthogonal zu Π_{n-1} bezüglich des Skalarprodukts

$$\langle f, g\rangle := \langle f, g\rangle_w := \int_a^b f(x)\ g(x)\ w(x)\ dx \tag{13.13}$$

ist.

Übung 13.2 Zeigen Sie: Ist w eine stetige, nichtnegative Funktion und ist $w(x) > 0$ für ein $x \in [a, b]$, dann ist $\langle f, g\rangle_w$ ein Skalarprodukt auf $\Pi \times \Pi$. Kann man eine der obigen Bedingungen an w weglassen?

Nun kann man leicht sehen, daß es für "vernünftige" Gewichtsfunktionen immer orthogonale Polynome *aller* Ordnungen $n \in \mathbb{N}_0$ gibt. Man erhält diese Polynome zum Beispiel durch das *Gram–Schmidtsche* Orthogonalisierungsverfahren: Sind $p_0, \ldots, p_n$ orthogonale Polynome der Ordnung $0, \ldots, n$ und ist $w \geq 0$ und *stetig*[113], dann ist

$$p_{n+1}(x) = x^{n+1} - \sum_{j=0}^{n} \frac{\langle (\cdot)^{n+1}, p_j\rangle}{\langle p_j, p_j\rangle}\ p_j(x), \qquad x \in \mathbb{R}, \tag{13.14}$$

ein orthogonales Polynom der Ordnung $n+1$, denn es ist, für $k = 0, \ldots, n$,

$$\langle p_{n+1}, p_k\rangle = \left\langle (\cdot)^{n+1}, p_k\right\rangle - \sum_{j=0}^{n} \frac{\langle (\cdot)^{n+1}, p_j\rangle}{\langle p_j, p_j\rangle}\ \underbrace{\langle p_j, p_k\rangle}_{=\delta_{jk}} = \left\langle (\cdot)^{n+1}, p_k\right\rangle - \left\langle (\cdot)^{n+1}, p_k\right\rangle = 0.$$

Zusammen mit Übung 13.3 zeigt dies, daß p_{n+1} ein orthogonales Polynom der Ordnung $n+1$ ist[114]. Da der *Leitterm* von p_{n+1} bei dieser Konstruktion das *Monom* x^{n+1} ist, bezeichnet man die durch (13.14) induktiv konstruierten orthogonalen Polynome als *monische* orthogonale Polynome.

Übung 13.3 Zeigen Sie, daß die orthogonalen Polynome $p_0, \ldots, p_n$ eine Basis von Π_n bilden.

Damit könnte man also Gaußsche Quadraturformeln dadurch konstruieren, daß man mittels (13.14) eine Folge $p_0, \ldots, p_n$ von monischen orthogonalen Polynomen konstruiert[115] und dann

[113] Das ist sogar mehr als wirklich nötig wäre, es genügt, wenn das Integral $I = \int_a^b w(x)\ dx$ *strikt quadratpositiv* i st, d.h. $I\left(f^2\right) > 0$ für alle $f \in \Pi$.

[114] ... das bis auf Normierung eindeutig bestimmt ist.

[115] Nur zur Erinnerung: für die Berechnung von p_k braucht man $p_0, \ldots, p_{k-1}$.

mit einem geeigneten Verfahren die Nullstellen von p_n bestimmt – daß die alle einfach und rell sind, werden wir noch sehen, und zwar in Proposition 13.17.

Ein etwas netterer Ansatz verwendet lineare Algebra und die *Momentenmatrix*[116]

$$M = \left[\left\langle (\cdot)^j, (\cdot)^k \right\rangle \ : \ j,k = 0,\ldots,n\right] = \begin{bmatrix} \mu_0 & \mu_1 & \ldots & \mu_n \\ \mu_1 & \mu_2 & \ldots & \mu_{n+1} \\ \vdots & \vdots & \ddots & \vdots \\ \mu_n & \mu_{n+1} & \ldots & \mu_{2n} \end{bmatrix}.$$

Diese $n+1 \times n+1$–Matrix ist positiv definit,

$$x^T M x = \left\langle \sum_{j=0}^{n} x_j (\cdot)^j, \sum_{k=0}^{n} x_k (\cdot)^k \right\rangle =: \langle p, p\rangle = I\left(p^2\right) > 0, \qquad x \in \mathbb{R}^{n+1} \setminus \{0\},$$

und besitzt eine QR–Zerlegung $M = QR$, bei der alle Diagonalelemente von R ungleich Null sind. Daher ist, für $j = 0,\ldots,n$,

$$\begin{aligned} r_{nn}\delta_{nj} &= r_{nj} = \left(Q^T M\right)_{nj} = \sum_{k=0}^{n} \underbrace{q_{kn}}_{=\left(Q^T\right)_{nk}} m_{kj} = \sum_{k=0}^{n} q_{kn} \left\langle (\cdot)^k, (\cdot)^j \right\rangle \\ &= \left\langle \sum_{k=0}^{n} q_{kn} \ (\cdot)^k, (\cdot)^j \right\rangle, \end{aligned}$$

das heißt, das Polynom

$$q_n(x) = \sum_{k=0}^{n} q_{kn} x^k, \qquad x \in \mathbb{R}, \qquad q_n \in \Pi_n,$$

ist orthogonal zu Π_{n-1}.

Übung 13.4 Untersuchen Sie die Beziehung zwischen der klassischen Gram–Schmidt Orthogonalisierung und der QR–Zerlegung.

Wenn wir jetzt mal für einen Moment[117] annehmen, daß Gaußsche Quadraturformeln für alle "vernünftigen" Gewichtsfunktionen existieren, dann können wir mit den inzwischen vertrauten Methoden auch den Fehler angeben.

Satz 13.16 *Sei $w \in C[a,b]$ eine stetige, nichtnegative Gewichtsfunktion und Q_n^* die Gauß–Quadraturformel für das Integral $I = \int_a^b w(x)\ dx$. Dann gibt es für jedes $f \in C^{2n+2}[a,b]$ ein $\xi \in [a,b]$, so daß*

$$I(f) - Q_n^*(f) = \frac{f^{(2n+2)}(\xi)}{(2n+2)!} \ \left\langle \omega_{n+1}, \omega_{n+1} \right\rangle, \tag{13.15}$$

wobei $\omega_{n+1} = (\cdot - x_0)\cdots(\cdot - x_n)$ das monische *orthogonale Polynom der Ordnung $n+1$ ist.*

[116] Derartige Matrizen, deren Einträge auf den "Antidiagonalen" konstant sind, bezeichnet man als *Hankelmatrizen* zu dem Vektor $(\mu_0,\ldots,\mu_{2n})$.

[117] Genauer: bis zum Beweis von Proposition 13.17.

Beweis: Aus den vorherigen Bemerkungen wissen wir, daß die Knoten der Gaußschen Quadraturformel[118] gerade die Nullstellen von ω_{n+1} sind. Nun sei $H_n \in \Pi_{2n+1}$ wieder der *Hermite–*Interpolant, der die Bedingungen

$$H_n f(x_j) = f(x_j), \qquad (H_n f)'(x_j) = f'(x_j), \qquad j = 0, \ldots, n,$$

erfüllt. Wegen der Exaktheit der Gauß–Formel ist dann

$$I(H_n f) = Q_n^*(H_n f) = \sum_{j=0}^{n} w_j \underbrace{H_n f(x_j)}_{=f(x_j)} = \sum_{j=0}^{n} w_j f(x_j) = Q_n^*(f),$$

also

$$I(f) - Q_n^*(f) = I(f) - I(H_n f) = I(f - H_n f).$$

Für $f - H_n f$ verwenden wir wieder die Fehlerformel[119]

$$(f - H_n f)(x) = \frac{f^{(2n+2)}(\xi(x))}{(2n+2)!} (x - x_0)^2 \cdots (x - x_n)^2 = \frac{f^{(2n+2)}(\xi(x))}{(2n+2)!} \omega_{n+1}^2(x),$$

und mit dem Mittelwertsatz der Integralrechnung erhalten wir, daß

$$\begin{aligned} I(f - H_n f) &= \int_a^b \frac{f^{(2n+2)}(\xi(x))}{(2n+2)!} \omega_{n+1}^2(x)\, w(x)\, dx = \frac{f^{(2n+2)}(\xi)}{(2n+2)!} \int_a^b \omega_{n+1}^2(x)\, w(x)\, dx \\ &= \frac{f^{(2n+2)}(\xi)}{(2n+2)!} \langle \omega_{n+1}, \omega_{n+1} \rangle . \end{aligned}$$

□

Zum Abschluß noch die versprochene Übungsaufgabe zur Gauß–Quadratur mit einer Vorbereitung.

Übung 13.5 (*Kettenbrüche.*)
Seien Zahlen v_k, w_k, $k \in \mathbb{N}_0$, gegeben. Dann heißt, für $n \in \mathbb{N}$,

$$C_n := \frac{v_1|}{|w_1} + \frac{v_2|}{|w_2} + \cdots \frac{v_n|}{|w_n} = \cfrac{v_1}{w_1 + \cfrac{v_2}{w_2 + \ddots + \cfrac{v_n}{w_n}}}, \qquad C_0 := 0,$$

n–te Konvergente des *Kettenbruchs*

$$\frac{v_1|}{|w_1} + \frac{v_2|}{|w_2} + \cdots = \cfrac{v_1}{w_1 + \cfrac{v_2}{w_2 + \ddots}}.$$

Zeigen Sie:

[118] Immer vorausgesetzt, diese existiert, aber ...
[119] Siehe auch Beispiel 13.10.

1. Sei $C_n = V_n/W_n$. Dann erfüllen V_n und W_n die Rekursionsformeln

$$\begin{array}{ll} V_0 = 0,\ V_1 = v_1, & V_{n+1} = w_{n+1}V_n + v_{n+1}V_{n-1} \\ W_0 = 1,\ W_1 = w_1, & W_{n+1} = w_{n+1}W_n + v_{n+1}W_{n-1} \end{array}, \quad n \in \mathbb{N}.$$

2. Für $n \in \mathbb{N}_0$ gilt

$$\frac{V_n}{W_n} - \frac{V_{n+1}}{W_{n+1}} = (-1)^{n+1}\,\frac{v_1 \cdots v_{n+1}}{W_n W_{n+1}},$$

und damit die (formale) Identität

$$\frac{v_1|}{|w_1} + \frac{v_2|}{|w_2} + \cdots = \sum_{k=0}^{\infty} (-1)^k\,\frac{v_1 \cdots v_{k+1}}{W_k W_{k+1}}.$$

3. (*D. Bernoulli*) Für jede Folge von reellen Zahlen c_n, $n \in \mathbb{N}_0$, mit $c_{n+1} \neq c_n$ gibt es einen Kettenbruch

$$\frac{v_1|}{|w_1} + \frac{v_2|}{|w_2} + \frac{v_3|}{|w_3} + \cdots,$$

so daß

$$c_n = \frac{v_1|}{|w_1} + \cdots + \frac{v_n|}{|w_n}$$

ist, und zwar

$$v_1 = c_1, \qquad w_1 = 1,$$

$$v_n = \frac{c_{n-1} - c_n}{c_{n-1} - c_{n-2}}, \qquad w_n = \frac{c_n - c_{n-2}}{c_{n-1} - c_{n-2}}, \qquad n \geq 2.$$

Übung 13.6 (*Quadratur á la Gauß*)
Seien paarweise verschiedene Knoten $x_0, x_1, \ldots x_n \in [0,1]$ gegeben und sei

$$\omega(x) := (x - x_0) \cdots (x - x_n) \in \Pi_{n+1}.$$

Außerdem seien $\omega_1(x)$ und $\omega_2(x)$ gegeben durch

$$\omega(x) \sum_{j=0}^{\infty} \mu_j\, x^{-j} =: \omega_1(x) + \omega_2(x), \qquad \omega_1 \in \Pi_n, \quad \omega_2(x) = \sum_{j=1}^{\infty} c_j x^{-j}.$$

1. Zeigen Sie:

$$\int_0^1 \frac{\omega(t)}{t - x_k} dt = \omega_1(x_k), \qquad k = 0, \ldots, n,$$

und bestimmen Sie daraus die Gewichte w_j^*, $j = 0, \ldots, n$, der zugehörigen interpolatorischen Quadraturformel

$$Q_n^*(f) = \sum_{j=0}^{n} w_j^*\, f(x_j).$$

2. Seien θ_j, $j \in \mathbb{N}_0$, die *Fehlermomente* der Quadraturformel Q_n^*, d.h.,

$$\theta_j := \int_0^1 t^j dt - Q_n\left((\cdot)^j\right),$$

und sei

$$\Theta(x) := \sum_{j=1}^{\infty} \theta_{j-1} x^{-j}.$$

Untersuchen Sie den Term

$$\omega(x) \sum_{j=0}^{n} \frac{w_j^*}{x - x_j}$$

und zeigen Sie, daß

$$\Theta(x) = \frac{\omega_2(x)}{\omega(x)}, \qquad x \in \mathbb{R} \setminus \{0\}.$$

Hinweis: Verwenden Sie für $0 \neq t \in \mathbb{R}$ eine Taylorentwicklung der Funktion $(t - x)^{-1}$ nach x um den Punkt $x = 0$.

3. Betrachten Sie die Kettenbruchentwicklung

$$\sum_{j=1}^{\infty} \mu_{j-1}\, x^{-j} = \frac{1}{|x} - \frac{\alpha_1|}{|x} - \frac{\alpha_2|}{|x} - \cdots, \qquad a_i > 0,$$

und zeigen Sie (Notation wie in der vorhergehenden Aufgabe): W_n ist ein Polynom vom Grad n in x und setzt man setzt man $\omega(x) = W_{n+1}(x)$, so gilt

$$\omega_2(x) = \sum_{i=2n+2}^{\infty} c_j x^{-j}.$$

4. Folgern Sie daraus, daß die Quadraturformel bezüglich der Nullstellen von $W_{n+1}(x)$ (warum hat dieses Polynom $n + 1$ Nullstellen?) Exaktheitsgrad $2n + 1$ hat.

Die Lösung findet sich in Kapitel 13.7.

13.5 Orthogonale Polynome

Wir haben gesehen, daß die Existenz und Bestimmung Gaußscher Quadraturformeln wesentlich mit orthogonalen Polynomen verknüpft ist. Das ist sicherlich eine, aber nicht die einzige, Begründung, sich orthogonale Polynome ein bißchen näher anzusehen. Zuerst rechtfertigen wir einmal die Existenz (und Eindeutigkeit) Gaußscher Quadraturformeln, indem wir zeigen, daß ein orthogonales Polynom immer einfache, reelle Nullstellen haben muß.

Proposition 13.17 *Ein orthogonales Polynom $p_n \in \Pi_n$ der Ordnung n zum Integral $\int_a^b w(x)\, dx$ bezüglich einer nichtnegativen, stetigen Gewichtsfunktion $w \in C[a, b]$ hat n einfache, reelle Nullstellen in $[a, b]$.*

Beweis: Nehmen wir zuerst an, irgendein p_n, $n \geq 1$, hätte *keine* Nullstelle in $[a, b]$, sei also beispielsweise $p_n > 0$. Dann ist

$$0 < \int_a^b p_n(x)\, w(x)\, dx = \langle 1, p_n \rangle\,,$$

im Widerspruch zur Orthogonalität. Hat p_n eine Nullstelle der Ordnung ≥ 2 in $[a, b]$, sagen wir ξ, dann ist $p_n/(x-\xi)^2 \in \Pi_{n-2}$ ein Polynom, das überall genau dasselbe Vorzeichen hat wie p_n. Wegen der Orthogonalität ist dann

$$0 = \left\langle p_n, \frac{p_n}{(\cdot - \xi)^2} \right\rangle = \left\langle 1, \frac{p_n^2}{(\cdot - \xi)^2} \right\rangle,$$

woraus $p_n = 0$ folgen würde – auch wieder nicht das, was man sich unter einem "anständigen" orthogonalen Polynom vorstellt.

Also hat jedes p_n in $[a, b]$ nur *einfache* Nullstellen, sagen wir $\xi_1, \ldots, \xi_m$, $m \leq n$. Nun ist aber das Polynom

$$p_n(x)\ \underbrace{(x-\xi_1)\cdots(x-\xi_m)}_{=:q \in \Pi_m}$$

auf $[a, b]$ entweder ≥ 0 oder ≤ 0 und damit

$$0 \neq \int_a^b p_n(x)\, q(x)\, w(x)\, dx = \langle p_n, q \rangle\,,$$

weswegen $m = n$ sein muß, denn ansonsten wäre das Integral wegen der Orthogonalität von p_n ja $= 0$. □

Aus Satz 13.8 und den Betrachtungen über die Gaußschen Quadraturformeln erhalten wir nun die Existenz und Eindeutigkeit dieser speziellen Formeln.

Korollar 13.18 *Sei $w \in C[a, b]$ eine positive[120] Gewichtsfunktion. Dann gibt es genau eine Quadraturformel Q_n vom Exaktheitsgrad $2n + 1$, deren Knoten die Nullstellen des[121] orthogonalen Polynoms sind.*

Die nächste Eigenschaft ist eine *Rekursionsformel* für *orthonormale* Polynome, der Beweis ist aus [5, S. 234].

Satz 13.19 *Die orthonormalen[122] Polynome p_n bezüglich einer positiven Gewichtsfunktion $w \in C[a, b]$ erfüllen eine* Drei–Term–Rekursionsformel

$$p_n(x) = (\alpha_n x + \beta_n)\, p_{n-1}(x) - \gamma_n p_{n-2}(x), \qquad x \in \mathbb{R}, \quad n \in \mathbb{N}, \tag{13.16}$$

mit Startwerten

$$p_0 \equiv \mu_0^{-1/2}, \qquad p_{-1} \equiv 0. \tag{13.17}$$

[120] Das heißt: $w \geq 0$ und es gibt mindestens ein $x \in [a, b]$, so daß $w(x) > 0$.

[121] Normierung spielt ja für die *Lage* der Nullstellen keine Rolle.

[122] Dadurch sind die Polynome bis auf Vorzeichen *eindeutig* definiert, in mehreren Variablen stimmt das übrigens nicht mehr – nur ein Beispiel, warum es da noch einiges zu tun gibt, siehe [51].

Beweis: Wir zeigen, daß die Polynome q_n, definiert durch

$$q_{n+1}(x) = x\ p_n(x) - \underbrace{\langle (\cdot) p_n, p_n\rangle}_{=:a_n} p_n(x) - \underbrace{\sqrt{\langle q_n, q_n\rangle}}_{=:b_n>0} p_{n-1}(x), \qquad x \in \mathbb{R}, \quad n \in \mathbb{N}_0,$$

mit den Startwerten wir in (13.17) orthogonale Polynome sind. Die orthonormalen Polynome sind dann

$$\begin{aligned} p_{n+1}(x) &= \frac{q_{n+1}(x)}{\sqrt{\langle q_{n+1}, q_{n+1}\rangle}} \\ &= \left(\frac{1}{\sqrt{\langle q_{n+1}, q_{n+1}\rangle}} x - \frac{\langle (\cdot) p_n, p_n\rangle}{\sqrt{\langle q_{n+1}, q_{n+1}\rangle}} \right) p_n(x) - \sqrt{\frac{\langle q_n, q_n\rangle}{\langle q_{n+1}, q_{n+1}\rangle}} p_{n-1}(x), \end{aligned}$$

woraus (13.16) sofort folgt.

Die Orthogonalität selbst beweisen wir durch Induktion über n; für $n = 0$, das heißt $q_0 \equiv 1$, ist nichts zu zeigen und $n = 1$ ist auch offensichtlich. Sei also die Orthogonalität für ein $n \geq 1$ bewiesen, dann betrachten wir, für $j = 0, \ldots, n$, die Werte

$$\langle q_{n+1}, p_j\rangle = \langle (\cdot)\ p_n, p_j\rangle - a_n\ \langle p_n, p_j\rangle - b_n\ \langle p_{n-1}, p_j\rangle\,;$$

für $j < n-1$ sind diese Werte natürlich Null. Unter Verwendung der Induktionshypothese ist außerdem

$$\begin{aligned} \langle q_{n+1}, p_{n-1}\rangle &= \langle (\cdot)\ p_n, p_{n-1}\rangle - b_n\ \langle p_{n-1}, p_{n-1}\rangle = \langle p_n, (\cdot)\ p_{n-1}\rangle - b_n\ \langle p_{n-1}, p_{n-1}\rangle \\ &= \langle p_n, q_n\rangle + a_{n-1} \underbrace{\langle p_n, p_{n-1}\rangle}_{=0} + b_{n-1} \underbrace{\langle p_n, p_{n-2}\rangle}_{=0} - b_n\ \underbrace{\langle p_{n-1}, p_{n-1}\rangle}_{=1} \\ &= \left\langle p_n, \sqrt{\langle q_n, q_n\rangle} p_n \right\rangle - b_n = \underbrace{\sqrt{\langle q_n, q_n\rangle}}_{=b_n} \underbrace{\langle p_n, p_n\rangle}_{=1} - b_n = 0, \end{aligned}$$

sowie

$$\langle q_{n+1}, p_n\rangle = \underbrace{\langle x\ p_n, p_n\rangle}_{=a_n} - a_n \underbrace{\langle p_n, p_n\rangle}_{=1} = 0,$$

womit also $q_{n+1} \perp \Pi_n$. Daß $q_{n+1} \not\equiv 0$ ist, folgt induktiv aus der Tatsache, daß $p_n \not\equiv 0$. □

Die Rekursionsformel (13.16) gibt uns nun eine *stabile* Methode, um die Nullstellen eines orthogonalen Polynoms zu bestimmen. Dazu erinnern wir uns an Kapitel 11.1, wo wir festgestell haben, daß man die Nullstellen eines Polynoms dadurch bestimmen kann, daß man alle Eigenwerte der Multiplikation "modulo" dieses Polynoms berechnet. Dabei gibt es einen "Freiheitsgrad", der in Kapitel 11.1 zwar erwähnt, aber nicht genutzt wurde: die zugrundeliegende Basis! Nun, eine Basis von Π sind die Polynome p_j, $j = 0, \ldots, n$ und nach (13.16) hat die Multiplikation mit x unter Vernachlässigung[123] von p_{n+1} die Form

$$x\ p_j(x) = \begin{cases} \alpha_{j+1}^{-1} \left(p_{j+1}(x) - \beta_{j+1} p_j(x) + \gamma_{j+1} p_{j-1}(x) \right), & j < n, \\ \alpha_{n+1}^{-1} \left(-\beta_{n+1} p_n(x) + \gamma_{n+1} p_{n-1}(x) \right), & j = n, \end{cases} \qquad j = 0, \ldots, n, \quad x \in \mathbb{R}, \tag{13.18}$$

[123] Das arme, vernachlässigte Polynom, da braucht man sich dann nicht wundern, wenn es verhaltensauffällig wird ...

also in Matrixform:

$$x \begin{bmatrix} p_0(x) \\ \vdots \\ p_n(x) \end{bmatrix} = \underbrace{\begin{bmatrix} -\frac{\beta_1}{\alpha_1} & \frac{1}{\alpha_1} & & & \\ \frac{\gamma_2}{\alpha_2} & -\frac{\beta_2}{\alpha_2} & \frac{1}{\alpha_2} & & \\ & \ddots & \ddots & \ddots & \\ & & \frac{\gamma_n}{\alpha_n} & -\frac{\beta_n}{\alpha_n} & \frac{1}{\alpha_n} \\ & & & \frac{\gamma_{n+1}}{\alpha_{n+1}} & -\frac{\beta_{n+1}}{\alpha_{n+1}} \end{bmatrix}}_{=M_n} \begin{bmatrix} p_0(x) \\ \vdots \\ p_n(x) \end{bmatrix}, \qquad x \in \mathbb{R}. \tag{13.19}$$

Diese *Multiplikationsmatrix* ist die Verallgemeinerung der Frobenius–Matrix aus Kapitel 11.1.

Mit anderen Worten – wir müssen "nur noch" die Koeffizienten der Rekursionsformel ermitteln, die Nullstellen des orthogonalen Polynoms der Ordnung $n+1$ sind dann die Eigenwerte der Matrix M_n.

Als nächstes werden wir sehen, daß man die Koeffizienten der Rekursionsformel aus den Leittermen der Monomdarstellung von orthogonalen Polynomen ablesen kann. Das Resultat und sein Beweis stammen aus [5].

Satz 13.20 *Sei p_n, $n \in \mathbb{N}_0$, eine Folge von* orthonormalen *Polynomen und sei*

$$p_n(x) = \lambda_n x^n + \kappa_n x^{n-1} + \cdots, \qquad x \in \mathbb{R}.$$

Dann ergeben sich die Koeffizienten in der Drei–Term–Rekursionsformel

$$p_n(x) = (\alpha_n x + \beta_n)\, p_{n-1}(x) - \gamma_n p_{n-2}(x), \qquad x \in \mathbb{R},$$

für die orthonormalen Polynome als

$$\alpha_n = \frac{\lambda_n}{\lambda_{n-1}}, \qquad \beta_n = \alpha_n \left(\frac{\kappa_n}{\lambda_n} - \frac{\kappa_{n-1}}{\lambda_{n-1}} \right), \qquad \gamma_n = \frac{\lambda_n \lambda_{n-2}}{\lambda_{n-1}^2}. \tag{13.20}$$

Wir können nun diesen Satz nutzen, um uns die Multiplikationsmatrix aus (13.19) noch einmal anzusehen. Mit (13.20) erhalten wir nämlich, daß

$$\begin{aligned} \frac{1}{\alpha_j} &= \frac{\lambda_{j-1}}{\lambda_j} \\ -\frac{\beta_j}{\alpha_j} &= \frac{\kappa_{j-1}}{\lambda_{j-1}} - \frac{\kappa_j}{\lambda_j} \\ \frac{\gamma_j}{\alpha_j} &= \frac{\lambda_{j-2}}{\lambda_{j-1}} = \frac{1}{\alpha_{j-1}}, \end{aligned}$$

und somit hat M_n die Gestalt

$$M_n = \begin{bmatrix} -\frac{\beta_1}{\alpha_1} & \frac{1}{\alpha_1} & & & \\ \frac{1}{\alpha_1} & -\frac{\beta_2}{\alpha_2} & \frac{1}{\alpha_2} & & \\ & \ddots & \ddots & \ddots & \\ & & \frac{1}{\alpha_{n-1}} & -\frac{\beta_n}{\alpha_n} & \frac{1}{\alpha_n} \\ & & & \frac{1}{\alpha_n} & -\frac{\beta_n}{\alpha_{n+1}} \end{bmatrix}$$

und ist damit eine *symmetrische Tridiagonalmatrix*. Insbesondere hat sie Hessenbergform und da orthogonale Transformationen der Form $A \to Q^T AQ$ die Symmetrie von A erhalten, liefert also das QR–Verfahren eine Folge symmetrischer Tridiagonalmatrizen, die natürlich besonders effizient zu speichern und zu behandeln sind. Und die Eigenwerte dieser Matrix sind ja gerade die gesuchten Knoten unsrer Gauß–Formel.

Beweis von Satz 13.20: Koeffizientenvergleich bei den Monomen der Ordnung n und $n-1$ in der Drei–Term–Rekursionsformel liefert sofort, daß

$$\lambda_n = \alpha_n \lambda_{n-1} \qquad \Longrightarrow \qquad \alpha_n = \frac{\lambda_n}{\lambda_{n-1}}$$

und

$$\kappa_n = \alpha_n \, \kappa_{n-1} + \beta_n \, \lambda_{n-1},$$

also

$$\beta_n = \alpha_n \left(\frac{\kappa_n}{\lambda_{n-1}} \frac{1}{\alpha_n} - \frac{\kappa_{n-1}}{\lambda_{n-1}} \right) = \alpha_n \left(\frac{\kappa_n}{\lambda_{n-1}} \frac{\lambda_{n-1}}{\lambda_n} - \frac{\kappa_{n-1}}{\lambda_{n-1}} \right) = \alpha_n \left(\frac{\kappa_n}{\lambda_n} - \frac{\kappa_{n-1}}{\lambda_{n-1}} \right).$$

Zum Beweis der Formel für γ_n wenden wir die Rekursionsformel wie folgt an,

$$\begin{aligned} 0 &= \langle p_n, p_{n-2} \rangle = \alpha_n \, \langle x p_{n-1}, p_{n-2} \rangle + \beta_n \underbrace{\langle p_{n-1}, p_{n-2} \rangle}_{=0} - \gamma_n \underbrace{\langle p_{n-2}, p_{n-2} \rangle}_{=1} \\ &= \alpha_n \, \langle p_{n-1}, x p_{n-2} \rangle - \gamma_n, \end{aligned}$$

und erhalten, daß

$$\begin{aligned} \gamma_n &= \alpha_n \langle p_{n-1}, x p_{n-2} \rangle \\ &= \alpha_n \left\langle p_{n-1}, \lambda_{n-2} \, x^{n-1} + \cdots \right\rangle = \alpha_n \left\langle p_{n-1}, \lambda_{n-2} \, x^{n-1} \right\rangle \\ &= \alpha_n \frac{\lambda_{n-2}}{\lambda_{n-1}} \left\langle p_{n-1}, \lambda_{n-1} \, x^{n-1} \right\rangle = \alpha_n \frac{\lambda_{n-2}}{\lambda_{n-1}} \left\langle p_{n-1}, \lambda_{n-1} \, x^{n-1} + \cdots \right\rangle \\ &= \alpha_n \frac{\lambda_{n-2}}{\lambda_{n-1}} \langle p_{n-1}, p_{n-1} \rangle = \frac{\lambda_n}{\lambda_{n-1}} \, \frac{\lambda_{n-2}}{\lambda_{n-1}}. \end{aligned}$$

□

Erfreulicherweise können wir aber nicht nur die *Knoten* der Quadraturformel aus den Koeffizienten der Rekursionsformel bestimmen, sondern auch die *Gewichte*. Dazu bedienen wir uns einer Idee von Golub und Welsch [18], siehe auch [45, S. 133].

Satz 13.21 *Sei $w \in C[a,b]$ eine positive Gewichtsfunktion und seien $v_0, \ldots, v_n$ die Eigenvektoren von M_n, dann ergeben sich die Gewichte der Gaußschen Quadraturformel als*

$$w_j = \mu_0 \, \frac{(v_j)_1^2}{v_j^T v_j}, \qquad j = 0, \ldots, n. \tag{13.21}$$

Beweis: Wir erinnern uns, daß nach (13.19) die Nullstellen $x_0, \ldots, x_n$ von p_{n+1} die Eigenwerte der Matrix M_n sind – und die dazugehörigen Eigenvektoren sind "natürlich"

$$v_j = \rho_j \begin{bmatrix} p_0(x_j) \\ \vdots \\ p_n(x_j) \end{bmatrix} \quad \in \mathbb{R}^{n+1}, \quad \rho_j \neq 0, \qquad j = 0, \ldots, n,$$

was man ganz einfach dadurch bekommt, daß man x_j in (13.19) einsetzt. Wegen der Exaktheit der Quadraturformel ist außerdem für $k = 0, \ldots, n$

$$\sum_{j=0}^{n} w_j \, p_k(x_j) = Q_n(p_k) = I(p_k) = \langle 1, p_k \rangle = \delta_{0k} \langle 1, p_0 \rangle = \delta_{0k} \mu_0^{-1/2} \underbrace{\langle 1, 1 \rangle}_{=\mu_0} = \mu_0^{1/2} \delta_{0k},$$

also bestimmt sich der Vektor $w = (w_0, \ldots, w_n)^T$ der Gewichte als Lösung des linearen Gleichungssystems

$$\underbrace{\begin{bmatrix} p_0(x_0) & \ldots & p_0(x_n) \\ \vdots & \ddots & \vdots \\ p_n(x_0) & \ldots & p_n(x_n) \end{bmatrix}}_{=:P_n \in \mathbb{R}^{n+1 \times n+1}} w = \mu_0^{1/2} e_1; \tag{13.22}$$

da die Knoten $x_0, \ldots, x_n$ alle verschieden sind und die Polynome $p_0, \ldots, p_n$ eine Basis von Π_n bilden ist dieses *Interpolationsproblem* eindeutig lösbar und daher die Matrix P_n invertierbar. Andererseits sind aber die Vektoren v_j/ρ_j die *Spalten* von P_n, das heißt,

$$P_n = \begin{bmatrix} \frac{v_0}{\rho_0} & \cdots & \frac{v_n}{\rho_n} \end{bmatrix}.$$

Als Eigenvektoren einer symmetrischen Matrix zu verschiedenen Eigenwerten sind die Vektoren v_j, $j = 0, \ldots, n$ zudem orthogonal, was wir ausnutzen, um beide Seiten von (13.22) mit v_j^T zu multiplizieren, was uns

$$\mu_0^{1/2} (v_j)_1 = v_j^T \, \mu_0^{1/2} e_1 = v_j^T P_n w = v_j^T \begin{bmatrix} \frac{v_0}{\rho_0} & \cdots & \frac{v_n}{\rho_n} \end{bmatrix} w = \frac{v_j^T v_j}{\rho_j} \, e_j^T w = \frac{v_j^T v_j}{\rho_j} \, w_j,$$

liefert, also

$$w_j = \mu_0^{1/2} \frac{\rho_j \, (v_j)_1}{v_j^T v_j}, \qquad j = 0, \ldots, n. \tag{13.23}$$

Wegen $p_0 \equiv \mu_0^{-1/2}$ ist aber

$$\rho_j = \frac{(v_j)_1}{p_0(x_j)} = \mu_0^{1/2} (v_j)_1, \qquad j = 0, \ldots, n,$$

und eingesetzt in (13.23) liefert dies (13.21). □

"Prinzipiell" können wir also, sobald wir die Koeffizienten der Rekursionsformel, oder, äquivalent die beiden führenden Koeffizienten der *orthonormalen* Polynome kennen, mit Hilfe linearer Algebra[124] "sofort" die Knoten und die Gewichte der zugehörigen Gaußschen Quadraturformel berechnen. Allerdings gibt es auch hier den berühmten Pferdefuß: Die Bestimmung dieser Werte ist nicht übermäßig stabil, siehe [18] oder [12]. Deswegen ist es vorteilhaft, für "wichtige"[125] orthogonale Polynome *explizite* und hoffentlich einfache Ausdrücke für die Koeffizienten in den Rekursionsformeln zu finden.

13.6 Jacobipolynome

Zum Abschluß sehen wir uns eine Klasse von Gewichtsfunktionen auf $[-1,1]$ an, die eine ganz besonders wesentlich Rolle spielen, nämlich die sogenannten *Jacobi–Gewichte*.

Definition 13.22 *Für Parameter $\alpha, \beta > -1$ ist das* Jacobi–Gewicht $w^{(\alpha,\beta)}$ *definiert als*

$$w^{(\alpha,\beta)}(x) = (1+x)^\alpha (1-x)^\beta.$$

Man beachte, daß für $\alpha < 0$ oder $\beta < 0$ das Jacobigewicht eine Singularität am entsprechenden Endpunkt von $[-1,1]$ hat, die aber für $\alpha, \beta > -1$ immer noch *integrierbar* bleibt. Die zugehörigen orthogonalen Polynome, die[126] als *Jacobipolynome* bezeichnet werden, kann man nun recht einfach in geschlossener Form angeben. Eine Warnung: Die Normierung der Jacobipolynome ist *nicht* einheitlich "geregelt", manche Leute verwenden den Namen für das *orthonormale* Polynom, manche Leute normieren den Wert des Polynoms an einem Endpunkt des Intervalls[127].

Proposition 13.23 *Das n–te Jacobipolynom $P_n^{(\alpha,\beta)}$ bestimmt sich bis auf Normierung über die* Rodriguez–Formel

$$P_n^{(\alpha,\beta)}(x) = \frac{1}{w^{(\alpha,\beta)}(x)} \frac{d^n}{dx^n} \left(w^{(\alpha,\beta)}(x)(1+x)^n(1-x)^n \right), \qquad x \in \mathbb{R}. \tag{13.24}$$

Beweis: Zuerst müssen wir einmal zeigen, daß (13.24) tatsächlich ein *Polynom* vom Grad n definiert. Dazu schreiben wir (13.24) als[128]

$$P_n^{(\alpha,\beta)}(x) = (1+x)^{-\alpha}(1-x)^{-\beta} \frac{d^n}{dx^n} \left((1+x)^{n+\alpha}(1-x)^{n+\beta} \right)$$

[124] Beispielsweise liefert die `Matlab`–Funktion `eig` die Eigenwerte einer Matrix *und*, ebenfalls als Matrix organisiert, auch die zugehörigen normierten Eigenvektoren.

[125] Mit Sicherheit gehört hier der Fall $w \equiv 1$ dazu.

[126] Wen wundert's?

[127] Das gibt dann auf jeden Fall *rationale* Koeffizienten und definiert das orthogonale Polynom eindeutig.

[128] Für $n \in \mathbb{N}$ und $\alpha \in [0,1)$ soll die Fakultät zu

$$(n+\alpha)! = \prod_{j=1}^{n} (j+\alpha)$$

erweitert werden – das reicht hier. Eine "saubere" Notation würde die Γ–Funktion verwenden.

$$
\begin{aligned}
&= (1+x)^{-\alpha}(1-x)^{-\beta}\sum_{k=0}^{n}\binom{n}{k}\frac{d^k}{dx^k}(1+x)^{n+\alpha}\ \frac{d^{n-k}}{dx^{n-k}}(1-x)^{n+\beta}\\
&= (1+x)^{-\alpha}(1-x)^{-\beta}\sum_{k=0}^{n}\binom{n}{k}\frac{(n+\alpha)!}{(n-k+\alpha)!}(1+x)^{(n-k+\alpha)}\ (-1)^{n-k}\frac{(n+\beta)!}{(k+\beta)!}(1-x)^{k+\beta}\\
&= (-1)^n\sum_{k=0}^{n}(-1)^k\binom{n}{k}\frac{(n+\alpha)!}{(n-k+\alpha)!}\frac{(n+\beta)!}{(k+\beta)!}\ \underbrace{(1+x)^{n-k}(1-x)^k}_{\in\Pi_n}\\
&= (-1)^n\sum_{k=0}^{n}(-1)^k\frac{(n+\alpha)!}{(n-k+\alpha)!}\frac{(n+\beta)!}{(k+\beta)!}\ B_k^n(x),
\end{aligned}
$$

unter Verwendung der baryzentrischen Koordinaten und Bernstein–Bézier Basispolynome aus [36, S. 111–116].

Für die Orthogonalität bemerken wir zuerst, daß die Funktionen

$$
\begin{aligned}
f_j(x) &= \frac{d^j}{dx^j}\left(w^{(\alpha,\beta)}(x)(1+x)^n(1-x)^n\right) = \frac{d^j}{dx^j}\left((1+x)^{n+\alpha}(1-x)^{n+\beta}\right)\\
&= \sum_{k=0}^{j}(-1)^k\binom{j}{k}\frac{(n+\alpha)!}{(n-j+k+\alpha)!}\frac{(n+\beta)!}{(n-j+\beta)!}(1+x)^{n-j+k+\alpha}(1-x)^{n-k+\beta}\\
&= (1+x)^{n-j+\alpha}(1-x)^{n-j+\beta}\sum_{k=0}^{j}(-1)^k\frac{(n+\alpha)!}{(n-j+k+\alpha)!}\frac{(n+\beta)!}{(n-j+\beta)!}B_k^j(x)
\end{aligned}
$$

für $j = 0, \ldots, n-1$ eine Nullstelle an ± 1 besitzen. Damit ist, mit partieller Integration[129], für $p \in \Pi_{n-1}$,

$$
\begin{aligned}
\int_{-1}^{1} p(x)\ P_n^{(\alpha,\beta)}(x)\ w^{(\alpha,\beta)}(x)\ dx &= \int_{-1}^{1} p(x)\ \frac{d^n}{dx^n}\left(w^{(\alpha,\beta)}(x)(1+x)^n(1-x)^n\right)\ dx\\
&= p(x)\ \frac{d^{n-1}}{dx^{n-1}}\left(w(x)(1+x)^n(1-x)^n\right)\Big|_{x=-1}^{1}\\
&\qquad - \int_{-1}^{1} p(x)'\ \frac{d^{n-1}}{dx^{n-1}}\left(w(x)(1+x)^n(1-x)^n\right)\ dx\\
&= \sum_{j=0}^{n-1}\underbrace{p^{(j)}(x)\ \frac{d^{n-1-j}}{dx^{n-1-j}}\left(w(x)(1+x)^n(1-x)^n\right)\Big|_{x=-1}^{1}}_{=0}\\
&\qquad +(-1)^n\int_{-1}^{1}\underbrace{p^{(n)}(x)}_{=0}\ \left(w(x)(1+x)^n(1-x)^n\right)\ dx\\
&= 0.
\end{aligned}
$$

□

[129] Und der offensichtlichen Abkürzung $w = w^{(\alpha,\beta)}$

Korollar 13.24 *Die* orthonormalen Legendrepolynome*, die orthonormalen Jacobipolynome zu* $\alpha = \beta = 0$*, haben die Form*

$$L_n(x) = \frac{P_n^{(0,0)}(x)}{\left\langle P_n^{(0,0)}, P_n^{(0,0)} \right\rangle^{1/2}} = \frac{(-1)^n}{2^n\ n!}\ \sqrt{n+\frac{1}{2}}\ \frac{d^n}{dx^n}\left(1-x^2\right)^n, \qquad x \in \mathbb{R}. \tag{13.25}$$

Beweis: Es ist nur noch die Normierungskonstante zu bestimmen[130]. Dazu bemerken wir, daß

$$(1+x)^n(1-x)^n = \left(1-x^2\right)^n = (-1)^n x^{2n} + \cdots,$$

also

$$\frac{d^n}{dx^n}\left(1-x^2\right)^n = (-1)^n \frac{(2n)!}{n!} x^n + \cdots,$$

und somit (mit partieller Integration)

$$\begin{aligned}
&\left\langle \frac{d^n}{dx^n}\left(1-(\cdot)^2\right)^n, \frac{d^n}{dx^n}\left(1-(\cdot)^2\right)^n \right\rangle \\
&= (-1)^n \int_{-1}^{1} \underbrace{\frac{d^n}{dx^n}\left((-1)^n \frac{(2n)!}{n!} x^n + \cdots\right)}_{=\frac{(2n)!}{n!}\ n!=(2n)!} \left(1-x^2\right)^n\ dx \\
&= (2n)! \int_{-1}^{1} \left(1-x^2\right)^n\ dx = \frac{n!^2\ 2^{2n+1}}{2n+1}.
\end{aligned}$$

□

Übung 13.7 Berechnen Sie das Integral $\int_{-1}^{1} \left(1-x^2\right)^n\ dx$.

Mit diesen Informationen und Satz 13.20 können wir nun für die Legendre–Polynome die Koeffizienten der Rekursionsformel und die "Multiplikationsmatrix" aus (13.19) *explizit* angeben: Nachdem

$$\left(1-x^2\right)^n = (-1)^n \left(x^{2n} - n x^{2n-2} + \cdots\right),$$

ist

$$\frac{d^n}{dx^n}\left(1-x^2\right)^n = (-1)^n \left(\frac{(2n)!}{n!}\ x^n - n\frac{(2n-2)!}{(n-2)!}\ x^{n-2} + \cdots\right),$$

und damit ist nach (13.25), für $n \in \mathbb{N}$,

$$\begin{aligned}
\lambda_n &= \frac{(2n)!}{2^n\ n!^2}\sqrt{n+\frac{1}{2}}, \qquad n \in \mathbb{N}. \\
\kappa_n &= 0,
\end{aligned}$$

[130] Das Vorzeichen kann man schließlich wählen, wie man will – es beeinflußt die Orthonormalität ja nicht.

Also ist $\beta_n = 0$, $n \in \mathbb{N}$, sowie

$$\begin{aligned}\alpha_n &= \frac{\lambda_n}{\lambda_{n-1}} = \frac{(2n)!}{2^n\, n!^2}\, \frac{2^{n-1}\,(n-1)!^2}{(2n-2)!}\, \sqrt{\frac{n+\frac{1}{2}}{n-\frac{1}{2}}} = \frac{2n(2n-1)}{2n^2}\, \sqrt{\frac{2n+1}{2n-1}} \\ &= \frac{\sqrt{4n^2-1}}{n} = \sqrt{4-\frac{1}{n^2}}\end{aligned}$$

Bleibt zum krönenden Abschluß noch der Quadraturfehler. Nach (13.15) ist für $f \in C^{2n+2}[a,b]$

$$\begin{aligned}E_n(f) &= \frac{f^{(2n+2)}(\xi)}{(2n+2)!}\, \langle \omega_{n+1}, \omega_{n+1} \rangle = \frac{f^{(2n+2)}(\xi)}{(2n+2)!}\, \left\langle \frac{L_{n+1}}{\lambda_{n+1}}, \frac{L_{n+1}}{\lambda_{n+1}} \right\rangle \\ &= \frac{f^{(2n+2)}(\xi)}{(2n+2)!} \frac{1}{\lambda_{n+1}^2} = \frac{2^{2n+3}\,(n+1)!^4}{(2n+3)!(2n+2)!^2}\, f^{(2n+2)}(\xi) \\ &= \frac{1}{2^{2n+1}\,(2n+3)!} \underbrace{\left(\frac{2^{2n+2}(n+1)!^2}{(2n+2)!^2} \right)^2}_{\sim \left(\sqrt{(n+1)\pi}\right)^2} f^{(2n+2)}(\xi) \sim \frac{(n+1)\pi}{2^{2n+1}\,(2n+3)!}\, f^{(2n+2)}(\xi)\end{aligned}$$

Hierbei wurde das *Wallis-Produkt*, siehe z.B. [22, S. 504–505], verwendet.

Übung 13.8 Bestimmen Sie numerisch die Knoten und Gewichte der Quadraturformeln Q_n für das Integral $I = \int_{-1}^{1} dx$, $n = 1, \ldots, 20$.

Übung 13.9 Zeigen Sie, daß die Tschebyscheffpolynome Jacobipolynome für $\alpha = \beta = -\frac{1}{2}$ sind.

13.7 Quadratur nach Art des Gaußes

In diesem "Bonus–Abschnitt" sehen wir uns noch an, wie Gauß die Knoten und Gewichte der Gauß–Quadratur für das Intervall $[0,1]$ hergeleitet hat, also die Lösung zu Aufgabe 13.6. Gauß selbst hat mit *formalen* Potenzreihen gerechnet[131] und sich um Fragen der Konvergenz, die man sich zu dieser Zeit auch nicht unbedingt gestellt hat, gekümmert. Man kann aber zeigen, daß alle betrachteten Reihen tatsächlich auf einem passenden Bereich konvergieren – und nur da sieht man sie sich dann auch an. Auch bei der Darstellung der meisten Rechenschritte versuche ich hier, den "Gaußschen Geist" beizubehalten, also eine (freie) Übertragung der Originalarbeit [11].
Die Herleitung der Gauß–Quadratur erfolgt in mehreren Schritten.

Bestimmung der Gewichte. Da wir eine *interpolatorische* Quadraturformel im Auge haben, ergeben sich die Gewichte als $w_j = I(\ell_j)$, $j = 0, \ldots, n$. Bedenkt man, daß

$$\ell_j = \frac{\omega}{\omega'(x_j)\ (\cdot - x_j)}, \qquad j = 0, \ldots, n, \tag{13.26}$$

[131] Ja, wirklich ge*rechnet*!

dann besteht der "interessante" Teil der Bestimmung von w_j^* in der Berechnung des Wertes $I\left(\omega/(\cdot - x_j)\right)$. Sei nun $\omega = x^{n+1} + a_n x^n + \cdots + a_0$, dann ist, da $\omega\left(x_j\right) = 0$,

$$\omega(x) = \omega(x) - \omega\left(x_j\right) = \left(x^{n+1} - x_j^{n+1}\right) + a_n \ \left(x^n - x_j^n\right) + \cdots + a_1 \ \left(x - x_j\right).$$

Da für $k = 0, \ldots, n$

$$\frac{x^{k+1} - x_j^{k+1}}{x - x_j} = x^k + x^{k-1}\, x_j + \cdots + x\, x_j^{k-1} + x_j^k,$$

erhalten wir, daß für $x \in \mathbb{R} \setminus \{x_j\}$

$$\begin{array}{rcccccccccc}
\dfrac{\omega(x)}{x - x_j} = & x^n & + & x^{n-1}\, x_j & + & x^{n-2}\, x_j^2 & + & \cdots & + & x_j^n \\
& & + & a_n x^{n-1} & + & a_n x^{n-2}\, x_j & + & \cdots & + & a_n x_j^{n-1} \\
& & & & + & a_{n-1} x^{n-2} & + & \cdots & + & a_{n-1} x_j^{n-2} \\
& & & & & & & \ddots & & \vdots \\
& & & & & & & & + & a_1
\end{array}$$

und damit

$$\begin{array}{rcccccccccc}
I\left(\dfrac{\omega}{\cdot - x_j}\right) = & & & & & & & & & & \\
= & I\left(x^n\right) & + & x_j\, I\left(x^{n-1}\right) & + & x_j^2\, I\left(x^{n-2}\right) & + & \cdots & + & x_j^n I(1) \\
& & + & a_n I\left(x^{n-1}\right) & + & a_n x_j\, I\left(x^{n-2}\right) & + & \cdots & + & a_n x_j^{n-1} I(1) \\
& & & & + & a_{n-1} I\left(x^{n-2}\right) & + & \cdots & + & a_{n-1} x_j^{n-2} I(1) \\
& & & & & & & \ddots & & \vdots \\
& & & & & & & & + & a_1 I(1) \\
= & \mu_n & + & x_j \mu_{n-1} & + & x_j^2 \mu_{n-2} & + & \cdots & + & x_j^n \mu_0 \\
& & + & a_n \mu_{n-1} & + & a_n x_j \mu_{n-2} & + & \cdots & + & a_n x_j^{n-1} \mu_0 \\
& & & & + & a_{n-1} \mu_{n-2} & + & \cdots & + & a_{n-1} x_j^{n-2} \mu_0 \\
& & & & & & & \ddots & & \vdots \\
& & & & & & & & + & a_1 \mu_0 \\
= & \mu_n + \mu_{n-1}\left(x_j + a_n\right) + \ldots + \mu_0\left(x_j^n + a_n x_j^{n-1} + \cdots + a_1\right) & & & & & & & & & \\
= & \displaystyle\sum_{k=0}^{n} \mu_k \underbrace{\left(x_j^{n-k} + \sum_{s=k+1}^{n} a_s x_j^{n-s}\right)}_{= \left(\omega\,(\cdot)^{-k}\right)\big|_{\Pi}(x_j)} & & & & & & & & & \\
= & \omega_1\left(x_j\right). & & & & & & & & &
\end{array}$$

Zusammen mit (13.26) liefert dies, daß

$$w_j^* = I\left(\ell_j\right) = \frac{\omega_1\left(x_j\right)}{\omega'\left(x_j\right)}, \qquad j = 0, \ldots, n. \tag{13.27}$$

Eine kleine Bemerkung am Rande: (13.27) ist noch eine Formel, um nur aus der Kenntnis der Knoten und der Momente die Gewichte der interpolatorischen Quadraturformel zu bestimmen, und das gilt für *alle* Integrale I.

Bestimmung der Fehlermomente. Da[132] für $a, x \in \mathbb{R}$, $a \neq x$,

$$\frac{d^j}{dx^j} (a-x)^{-1} = j! \, (a-x)^{-j-1},$$

können wir für $j = 0, \ldots, n$ die Taylorentwicklung von $\frac{w_j^*}{x-x_j}$ nach x_j um $x_j = 0$ als

$$\frac{w_j^*}{x-x_j} = w_j^* \sum_{k=0}^{\infty} \frac{x_j^k}{k!} \left(\frac{d^k}{dx_j^k} \frac{1}{x-x_j} \right) \Bigg|_{x_j=0} = w_j^* \sum_{k=0}^{\infty} \frac{x_j^k}{k!} k! \, \frac{1}{(x-x_j)^{k+1}} \Bigg|_{x_j=0} = w_j^* \sum_{k=0}^{\infty} \frac{x_j^k}{x^{k+1}}$$

bestimmen und erhalten somit, daß

$$\begin{aligned} \sum_{j=0}^{n} \frac{w_j^*}{x-x_j} &= x^{-1} \underbrace{\sum_{j=0}^{n} w_j^*}_{=Q_n^*(1)} + x^{-2} \underbrace{\sum_{j=0}^{n} w_j^* \, x_j}_{=Q_n^*((\cdot))} + x^{-3} \underbrace{\sum_{j=0}^{n} w_j^* \, x_j^2}_{=Q_n^*((\cdot)^2)} + \cdots \\ &= \sum_{k=0}^{\infty} \frac{\mu_k - \left(\mu_k - Q_n^* \left((\cdot)^k \right) \right)}{x^{k+1}} = \sum_{k=1}^{\infty} \left(\mu_{k-1} - \theta_{k-1} \right) \, x^{-k} = \sum_{k=1}^{\infty} \frac{\mu_{k-1}}{x^k} - \Theta(x), \end{aligned}$$

also

$$\Theta(x) = \underbrace{\sum_{k=1}^{\infty} \frac{\mu_{k-1}}{x^k}}_{=:M(x)} - \sum_{j=0}^{n} \frac{w_j^*}{x-x_j}. \tag{13.28}$$

Da die Quadraturformel interpolatorisch ist, ist auf alle Fälle $\theta_j = 0$, $j = 0, \ldots, n$, das heißt,

$$\Theta(x) = \sum_{j=n+1}^{\infty} \frac{\theta_j}{x^{j+1}} = x^{-(n+2)} \left(\theta_{n+1} + \theta_{n+2} x^{-1} + \cdots \right)$$

und damit besitzt die Funktion $x \mapsto \omega(x) \, \Theta(x)$ nur *negative* Exponenten. Nun ist, mit (13.28),

$$\begin{aligned} \omega(x) \, \Theta(x) &= \omega(x) \left(M(x) - \sum_{j=0}^{n} \frac{w_j^*}{x-x_j} \right) = \omega(x) \, M(x) - \omega(x) \, \sum_{j=0}^{n} \frac{w_j^*}{x-x_j} \\ &= \underbrace{\omega_1(x)}_{\in \Pi_n} + \omega_2(x) - \underbrace{\sum_{j=0}^{n} w_j^* \frac{\omega(x)}{x-x_j}}_{\in \Pi_n}. \end{aligned}$$

[132] Induktion!

Also ergibt der polynomiale Teil die Identität

$$\omega_1(x) = \omega(x) \sum_{j=0}^{n} \frac{w_j^*}{x - x_j} \tag{13.29}$$

und, weil $\omega(x)\ \Theta(x)$ nur negative Potenzen von x enthält, erhalten wir

$$\omega_2(x) = \omega(x)\ \Theta(x). \tag{13.30}$$

Kettenbrüche. Jetzt verwenden wir die Bernoullische Formel aus Aufgabe 13.5, um die Koeffizienten v_j und w_j der Kettenbruchentwicklung

$$M(x) = \sum_{j=1}^{\infty} \mu_{j-1}\ x^{-j} = \frac{v_1}{|\ w_1} + \frac{v_2}{|\ w_2} + \cdots$$

zu bestimmen, und zwar ist, mit

$$c_j = \sum_{k=1}^{j} \mu_{k-1}\ x^{-k}, j \in \mathbb{N},$$

zuerst einmal $v_1 = \mu_0$ und $w_1 = x$ sowie

$$v_j = \frac{c_{j-1} - c_j}{c_{j-1} - c_{j-2}} = \frac{-\mu_{j-1}\ x^{-j}}{\mu_{j-2}\ x^{-j+1}} = -\frac{\mu_{j-1}}{\mu_{j-2}}\ x^{-1}, \qquad j > 1,$$

sowie

$$w_j = \frac{c_j - c_{j-2}}{c_{j-1} - c_{j-2}} = \frac{\mu_{j-1}x^{-j} + \mu_{j-2}x^{-j+1}}{\mu_{j-2}x^{-j+1}} = \frac{\mu_{j-1}}{\mu_{j-2}}x^{-1} + 1, \qquad j > 1.$$

Multiplizieren wir nun[133] jeweils v_j und w_j mit x, dann erhalten wir daß $v_1 = 1$, $w_1 = x$ und

$$v_j = -\frac{\mu_{j-1}}{\mu_{j-2}} \qquad \text{und} \qquad w_j = x + \frac{\mu_{j-1}}{\mu_{j-2}} \qquad j \geq 2. \tag{13.31}$$

Insbesondere ist $V_j \in \Pi_{j-1}$ und $W_j \in \Pi_j$, $j \in \mathbb{N}$, und für alle $m \in \mathbb{N}$ ist nach Teil 2 von Aufgabe 13.5

$$M(x) - \frac{V_m(x)}{W_m(x)} = \sum_{k=m}^{\infty} \left(\frac{V_{k+1}(x)}{W_{k+1}(x)} - \frac{V_k(x)}{W_k(x)} \right) = \sum_{k=m}^{\infty} (-1)^k \frac{v_1 \cdots v_{k+1}}{W_k(x) W_{k+1}(x)}.$$

Da man für $p \in \Pi_j$ die *Laurententwicklung*

$$\frac{1}{p} = \sum_{k=j}^{\infty} \alpha_k\ x^{-k}$$

[133] Das ändert den Wert des Kettebruchs nicht.

erhält[134], ist also

$$M(x) - \frac{V_{n+1}(x)}{W_{n+1}(x)} = \sum_{k=n+1}^{\infty} (-1)^k \underbrace{\frac{v_1 \cdots v_{k+1}}{W_k(x) W_{k+1}(x)}}_{\in \Pi_{2k+1}} = \sum_{k=2n+2}^{\infty} \frac{\alpha_k}{x^{k+1}},$$

oder

$$M(x)\, W_{n+1}(x) = V_{n+1}(x) + W_{n+1}(x) \sum_{k=2n+2}^{\infty} \frac{\alpha_k}{x^{k+1}}. \tag{13.32}$$

Exaktheit der Quadraturformel. Da W_{n+1} durch eine Drei–Term–Rekursionsformel definiert ist, hat es $n+1$ verschiedene reelle Nullstellen, x_j^*, $j = 0, \ldots, n$. Dann ist

$$\omega^*(x) := (x - x_0^*) \cdots (x - x_n^*)$$

ein Vielfaches von W_{n+1} und es ist

$$\omega_1^*(x) + \omega_2^*(x) = M(x)\, \omega^*(x) = V_{n+1}(x) + \omega^*(x) \sum_{k=2n+2}^{\infty} \frac{\alpha_k}{x^{k+1}},$$

das heißt, wenn wir nur die Terme mit negativem Exponenten betrachten,

$$\omega_2^*(x) = \omega^*(x) \sum_{k=2n+2}^{\infty} \frac{\alpha_k}{x^{k+1}}$$

und damit liefert (13.30), daß für diese Knotenwahl

$$\sum_{k=1}^{\infty} \frac{\theta_{k-1}}{x^k} = \Theta(x) = \frac{\omega_2^*(x)}{\omega^*(x)} = \sum_{k=2n+1}^{\infty} \frac{\alpha_{k-1}}{x^k},$$

also, nach Koeffizientenvergleich, daß $\theta_0 = \cdots = \theta_{2n+1} = 0$. Das war's!

[134] Wie man das beweist? Einfach mit p durchmultiplizieren, das liefert dann die Bedingungsgleichungen für die α_j, $j \geq n$.

Nichts setzt dem Fortgang von Wissenschaft mehr Hindernis entgegen, als wenn man zu wissen glaubt, was man noch nicht weiß. In diesen Fehler fallen gewöhnlich die schwärmerischen Erfinder von Hypothesen.

G. Ch. Lichtenberg

14 Numerik gewöhlicher Differentialgleichungen

In diesem Kapitel wollen wir einen Schnelldurchlauf durch die numerischen Verfahren für Systeme von gewöhnlichen Differentialgleichungen *erster Ordnung* der Form

$$u'(t) = f\left(t, u(t)\right), \qquad u(0) = u_0, \qquad u : \mathbb{R} \to \mathbb{R}^n, \tag{14.1}$$

durchführen[135]. Daß wir uns auf erste Ordnung beschränken ist keine echte Einschränkung, da sich ja jede Differentialgleichung höherer Ordnung der Form

$$v^{(k)}(t) = f\left(t, v(t), \cdots, v^{(k-1)}(t)\right)$$

auf das System

$$u_{j+1} = u_j', \qquad j = 1, \ldots, k-1, \qquad u_k' = f\left(t, u\right)$$

zurückführen läßt. Für die Existenz und Eindeutigkeit der Lösungen einer gewöhlichen Differentialgleichung hat man den Satz von *Picard–Lindelöf*, den man auch in den meisten Standardbüchern über Analysis finden kann.

Wir wollen uns hier aber nicht mit der Frage nach der Existenz einer Lösung beschäftigen, sondern mit den numerischen Methoden zur *Bestimmung* dieser Lösung.

14.1 Woher kommen die eigentlich?

Der klassische Ausgangspunkt für gewöhnliche Differentialgleichungen[136] sind sicherlich Anwendungen der Naturwissenschaft, in der das Verhalten einer Funktion oder eines "Systems" nur durch sein Änderungsverhalten gegeben ist, beispielsweise ein Körper, der sich unter Krafteinfluss bewegt. Die Kraft führt zu einer Beschleunigung $\boldsymbol{a}(t) = \ddot{\boldsymbol{s}}(t)$, wobei die Bahn $\boldsymbol{s}(t) \in \mathbb{R}^3$ gesucht ist. Andere Beispiele sind Räuber–Beute–Modelle, wo die Fortpflanzungsraten vom

[135] Genaugenommen ist (14.1) ja sogar ein *Anfangswertproblem*.

[136] Oder *ODEs*, **O**rdinary **D**ifferential **E**quations, wie der Fachmann in Unterscheidung zu den *PDEs*, **P**artial **D**ifferential **E**quations sagt.

Nahrungsangebot und damit vom jeweiligen Verhältnis zwischen der Anzahl der Räuber und der Beutetiere abhängen, oder das Wetter, bei dem der Austausch der Luftmassen vom jeweiligen lokalen Luftdruck, der Luftdruck aber wieder vom Austausch der Luftmassen abhängt. Manchmal kann man Differentialgleichungen auch verwenden, um Probleme zu lösen, die sonst ziemlich eklig wären.

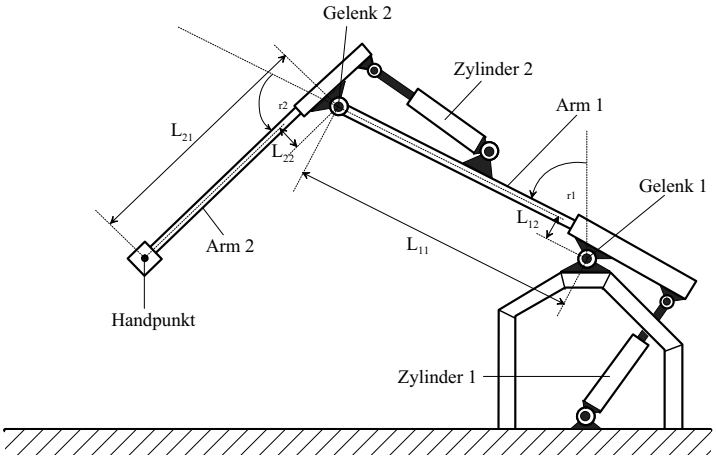


Abbildung 14.1: Schematisches Bild eines (einfachen, eigentlich sehr einfachen) hydraulischen Zweigelenkroboters.

Beispiel 14.1 (Kinematik von Robotern) *In der Robotik besteht ein Hauptproblem darin, einen Roboter so zu steuern, daß er eine Bahn mit einer bestimmten Dynamik abfährt, daß also die* Bahnkurve $\boldsymbol{s}(t)$ *für den Arbeitspunkt*[137] *vorgegeben ist – die Bestimmung dieser Kurven ist nun wieder eine Kunst für sich. Die Regelgrößen unseres Roboters sind aber in Abb. 14.1 gerade die Gelenkwinkel $\varphi_{1/2}$ und diese müssen bestimmt werden. Das ginge in diesem Beispiel zwar noch, und zwar sogar explizit und bijektiv zwischen $\boldsymbol{s}(t)$ und $\varphi_{1/2}$, aber bei etwas realistischeren Mehrachsmaschinen, siehe Abb. 14.2, ist das hoffnungslos. Was wir wissen ist lediglich, daß es normalerweise*[138] *eine relativ leicht zu berechnende Funktion $F(\Phi)$ gibt, die den* Maschinenkonfigurationen $\Phi = [\varphi_1, \ldots, \varphi_n]$ *die Manipulatorenposition $F(\Phi)$ zuordnet, die sogenannte* kinematische Vorwärtstransformation*. Dabei muss F nicht injektiv sein und ist es normalerweise auch nicht*[139]*, womit F auch nicht invertierbar ist. Nun muss man aber $\Phi(t)$ bestimmen, so daß*

$$\boldsymbol{s}(t) = F\left(\Phi(t)\right) \tag{14.2}$$

ist. Was also tun?

[137] Meistens als *Manipulator* bezeichnet.
[138] Es gibt auch Roboter, nämlich die vom "Plattformtyp", da ist es genau andersrum.
[139] Viele Kinematiken sind redundant.

Ganz einfach: Wir differenzieren (14.2) nach t und erhalten unter Verwendung der Kettenregel, daß

$$\dot{\boldsymbol{s}}(t) = \frac{d}{dt} F\left(\Phi(t)\right) = \nabla_{\Phi} F\left(\Phi(t)\right) \frac{d}{dt}\Phi(t) = \begin{bmatrix} \frac{\partial F_1}{\partial \varphi_1}\left(\Phi(t)\right) & \dots & \frac{\partial F_1}{\partial \varphi_n}\left(\Phi(t)\right) \\ \frac{\partial F_2}{\partial \varphi_1}\left(\Phi(t)\right) & \dots & \frac{\partial F_2}{\partial \varphi_n}\left(\Phi(t)\right) \\ \frac{\partial F_3}{\partial \varphi_1}\left(\Phi(t)\right) & \dots & \frac{\partial F_3}{\partial \varphi_n}\left(\Phi(t)\right) \end{bmatrix} \begin{bmatrix} \dot{\varphi}_1(t) \\ \vdots \\ \dot{\varphi}_n(t) \end{bmatrix},$$

was ein unterbestimmtes *lineares Gleichungssystem in $\dot{\Phi}$ ist und aus $\dot{\Phi}$ und den Anfangsbedingungen lässt sich durch Integration (= Lösen einer Differentialgleichung) die Steuerungsfunktion $\Phi(t)$ bestimmen.*

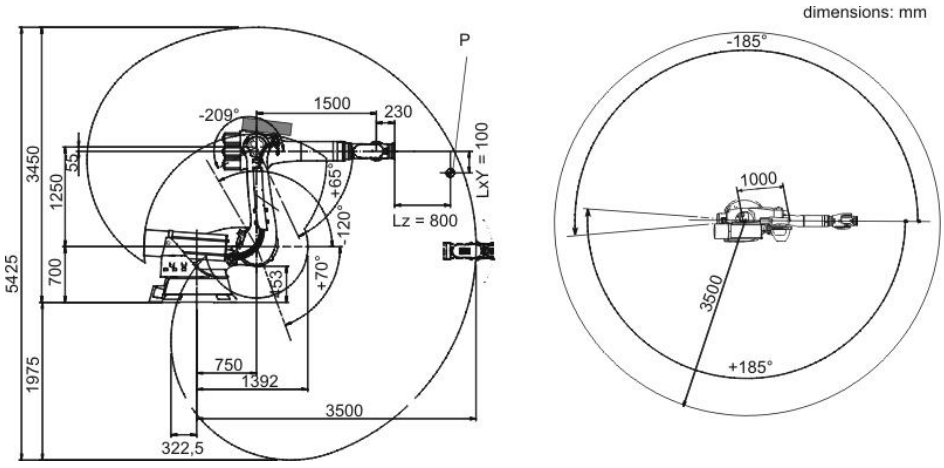


Abbildung 14.2: Etwas realistischere Werte von Verfahrbereich und kinematischen Transformationen..

Das bringt uns auch gleich zu einer anderen Anwendung, nämlich zur *Kontrolltheorie*: Hier hat man es wieder mit einer Systembeschreibung der Form

$$u'(t) = f\left(t, u(t)\right)$$

zu tun, und muss eine "Gleichung" $u(t) = g(t)$ lösen, oftmals unter weiteren Bedingungen an höhere Ableitungen von u.

Und schließlich gibt es noch die *Variationsrechnung*, bei der man beispielsweise, siehe [14], ein Funktional der Form

$$J[f] = \int_a^b F\left(x, f, f'\right) dx \tag{14.3}$$

bezüglich der Funktion f zu minimieren versucht, wobei $F\left(x, y, y'\right) : \mathbb{R}^3 \to \mathbb{R}$ eine Funktion in den formalen Variablen x, y, y' ist.

Beispiel 14.2 (Kürzester Weg) *Gesucht ist eine Funktion f mit $f(a) = f_a$, $f(b) = f_b$ und kürzester Bogenlänge*

$$\int_a^b \sqrt{1 + f'(x)}\, dx, \qquad \Rightarrow \qquad F(x, y, y') = \sqrt{1 + y'}.$$

Der Trick bei solchen, sehr klassischen, Problemen besteht darin, den Ausdruck in (14.3) "nach f abzuleiten" und diese "Ableitung" Null zu setzen. Macht man das richtig, dann erhält man wieder ein System von Differentialgleichungen, und zwar die sogenannten *Euler–Lagrange–Gleichungen*.

Satz 14.3 *Ist eine Funktion f Minimallösung des Variationsproblems (14.3), dann ist*

$$\frac{\partial F}{\partial y}(x, f, f') - \frac{\partial}{\partial x}\frac{\partial F}{\partial y'}(x, f, f') = 0. \tag{14.4}$$

Nun aber auch ein bisschen was über Lösungsmethoden ...

14.2 Einschrittverfahren I – Euler und Taylor

Der einfachste Fall ist mit Sicherheit eine Differentialgleichung erster Ordnung (also kein System). Bei der einfachsten Methode hierfür, dem *Eulerschen Polygonzugverfahren* beobachtet man ganz einfach, daß man die Eigenschaft $u'(t) = f(t, u(t))$ auch folgendermaßen interpretieren kann:

> *Aus der Kenntnis von t und $u(t)$ können wir die Tangente von u an der Stelle t bestimmen.*

Tatsächlich ist das **die** fundamentale Idee, auf der die meisten "klassischen" Lösungsverfahren[140] für gewöhnliche Differentialgleichungen beruhen. Aber zurück zum Lösungsverfahren: Nun kann man ja der Tangente folgen und erhält somit

$$u(t + h) \sim u(t) + h\, u'(t) = u(t) + h\, f(t, u(t)),$$

oder, für eine Schrittweite $h \neq 0$, die Werte $u(t_j)$, $t_j = hj$, $j \in \mathbb{N}_0$, durch

$$u(t_{j+1}) \sim u_{j+1} = u_j + h\, f(t_j, u_j), \qquad j \in \mathbb{N}_0, \tag{14.5}$$

bestimmen. Die Vorgehensweise von (14.5) bezeichnet man als Eulersches Polygonzugverfahren – die geometrische Interpretation in Bild 14.3 ist dabei recht naheliegend.

Man sieht sofort, daß diese Methode immer unterhalb der echten Lösung der Differentialgleichung bleibt, wenn beispielsweise u eine *konvexe* Funktion ist – in diesem Fall werden sich also alle *lokalen Diskretisierungsfehler* $u(t_j) - u_j$ fleißig und stetig aufsummieren, siehe nochmals

[140] In Gegensatz zu den Galerkin–Verfahren, bei denen eine Operatorgleichung ("normalerweise" Integral- oder Differentialgleichung) durch Einsetzen von "Ansatzfunktionen" diskretisiert wird.

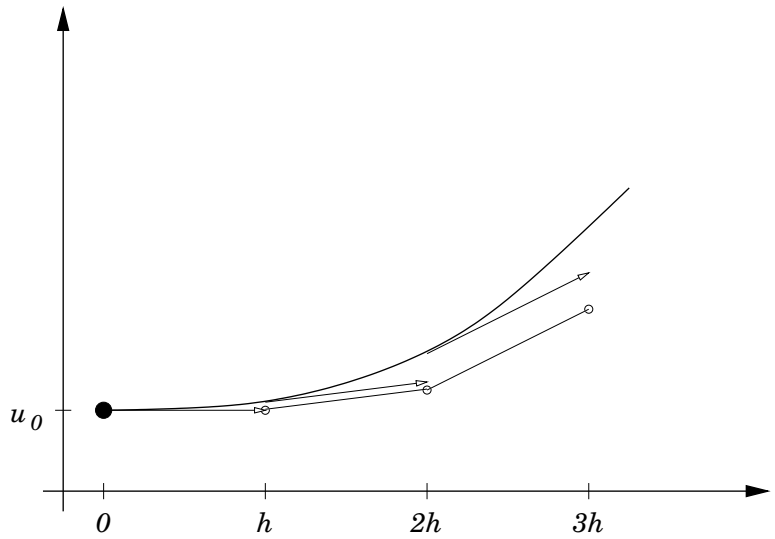


Abbildung 14.3: Das Eulersche Polygonzugverfahren – bei jedem Schritt wird die Tangente an das "echte" u genutzt, weswegen sich Fehler gerne aufsummieren..

Abb. 14.3. Das ist natürlich nicht sonderlich erstrebenswert und wird wohl auch nur funktionieren, wenn die Schrittweite h im Absolutbetrag sehr klein gewählt wird.

Etwas besser wird das Ganze schon, wenn man anstelle der "ersten" Näherung eine lokale Taylorentwicklung von u (falls sowas existier) verwendet, also

$$u\left(t+h\right)=u(t)+\frac{h}{1!}u'(t)+\frac{h^2}{2!}u''(t)+\cdots+\frac{h^k}{k!}u^{(k)}(t)+\frac{h^{k+1}}{k!}u^{(k+1)}(\xi), \tag{14.6}$$

für ein $\xi\in[t,t+h]$, denn nun ist der Fehler nur noch $O\left(h^{k+1}\right)$. Nun kann man sich natürlich fragen, wo der Gewinn bei der Sache ist, wenn man anstelle der gesuchten Funktion auch noch deren Ableitungen kennen muß! Aber in Wirklichkeit ist das gar nicht so schlimm, wenn man berücksichtigt, daß

$$\begin{aligned}
u'(t) &= f\left(t,u(t)\right),\\
u''(t) &= \frac{d}{dt}u'(t)=\frac{d}{dt}f\left(t,u(t)\right)\\
u'''(t) &= \frac{d}{dt}u''(t)=\frac{d^2}{dt^2}f\left(t,u(t)\right)\\
&\vdots\\
u^{(k)}(t) &= \frac{d^{k-1}}{dt^{k-1}}f\left(t,u(t)\right).
\end{aligned}$$

Allerdings ist diese Sache mit jeder Menge Nachdifferenzierei verbunden und deswegen auch nicht so begeisternd.

Eine andere, taylorbasierte Methode, die laut [41, S. 361] auch in manchen Anwendungen durchaus gute Erfolge erzielt, ist besonders gut geeignet für einfache, insbesondere *polynomiale* Funktionen f. Hierbei setzt man für "festes" $t \in \mathbb{R}$ die Taylor–Entwicklung

$$u(t+h) = \sum_{j=0}^{k} \underbrace{\frac{u^{(j)}(t)}{j!}}_{=:c_j} h^j + \underbrace{R_{k+1}(t,h)\, h^{k+1}}_{=:R(h)} = \sum_{j=0}^{k} c_j h^j + R(h),$$

aufgefasst als Funktion von h, in die Identität

$$u'(t+h) = f(t+h, u(t+h))$$

ein und erhält

$$\sum_{j=0}^{k} j\, c_j h^{j-1} + R'(h) = f\left(t+h, \sum_{j=0}^{k} c_j h^j + R(h)\right),$$

woraus man durch Koeffzientenvergleich[141] der Terme der Ordnung $\leq k$ in h die Koeffizienten

$$c_j = \frac{u^{(j)}(t)}{j!}, \qquad j = 1, \ldots, k,$$

bestimmen kann. Und die kann man dann in (14.6) einsetzen und erhält somit $u(t+h)$ mit einem Fehler von $O\left(h^{k+1}\right)$.

Beispiel 14.4 *Betrachten wir die Differentialgleichung*

$$u'(t) = t\, u(t), \qquad \textit{das heißt} \qquad f(x,y) = xy,$$

mit Anfangswert $u(0) = u_0 = 1$, die die exakte Lösung $u(t) = e^{t^2/2}$ hat. Hier ergibt sich also

$$\sum_{j=0}^{k-1} (j+1)\, c_{j+1} h^j + R'(h) = (t+h)\left(\sum_{j=0}^{k} c_j h^j + R(h)\right) = t c_0 + \sum_{j=1}^{k} (t c_j + c_{j-1})\, h^j + \widetilde{R}(h),$$

wobei in $\widetilde{R}(h)$ nur Potenzen h^ℓ mit $\ell \geq k+1$ vorkommen. Somit könne wir also rekursiv die Werte

$$\begin{aligned} c_1 &= t\, c_0 = t\, u(t) \\ c_2 &= \frac{1}{2}(t\, c_1 + c_0) \\ &\vdots \\ c_k &= \frac{1}{k!}(t\, c_{k-1} + c_{k-2}) \end{aligned}$$

[141] Das wird am einfachsten, wenn die Funktion f ein Polynom ist, denn dann hat man auf beiden Seiten Polynome, deren Terme sich schön vergleichen lassen.

liefert. Diese Methode ist in Programm 14.1 realisiert. Berechnet man die Lösungen, so stellt man fest, daß natürlich das lineare Verfahren immer unterhalb *der Lösung verläuft, die Verfahren höherer Ordnung hingegen oberhalb und daß erstaunlicherweise die Verfahren höherer Ordnung* schlechtere *Ergebnisse bringen. Das ist so überraschend nun auch wieder nicht – die höheren Ableitungen von e^{t^2} wachsen ziemlich schnell. Siehe Abb. 14.4. Mit Schrittweite $h = 0.01$ sehen übrigens alle drei Verfahren sehr gut aus.*

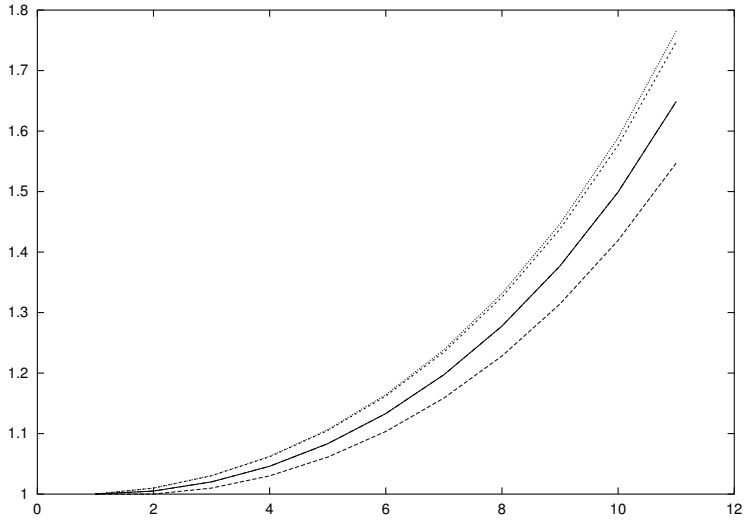


Abbildung 14.4: Die "richtige" Lösung für 10 Schritte mit $h = 0.1$ und die Näherungen erster, zweiter, und dritter Ordnung. Von unten nach oben: erste Ordnung, Original, zweite Ordnung, dritte Ordnung. Bei so einer geringen Schrittweite bestimmt das, was im ersten Schritt passiert ganz entscheidend das Ergebnis.

14.3 Einschrittverfahren II – Extrapolation

Die nächste Gruppe von Verfahren, die wir uns nun kurz ansehen wollen, lebt von der Kombination *unterschiedlicher* Methoden. Nehmen wir mal an, daß wir ausgehend von $u_j \sim u(t_j)$ einmal einen Schritt des Polygonzugverfahrens mit Schrittweite h und einmal *zwei* Schritte mit Schrittweite $\frac{h}{2}$ – in beiden Fällen landen wir beim Punkt $t_{j+1} = t^*_{j+2} = t_j + h$. Im ersten Fall erhalten wir

$$u_{j+1} = u_j + h\, f(t_j, u_j), \tag{14.7}$$

und im zweiten Fall

$$u^*_{j+2} = u^*_{j+1} + \frac{h}{2} f\left(t^*_{j+1}, u^*_{j+1}\right) = u_j + \frac{h}{2} f(t_j, u_j) + \frac{h}{2} f\left(t^*_{j+1}, u_j + \frac{h}{2} f(t_j, u_j)\right). \tag{14.8}$$

```
%#
%# TaylorDGL.m
%# Loese u' = t u ueber Taylor-Methoden
%# Daten:
%#   n  Anzahl Schritte
%#   h  Schrittweite
%#   k  Ordnung

function y = TaylorDGL( n,h,k )
  c = zeros( 1,k+1 );
  y = zeros( 2,n+1 );
  u = 1;
  t = 0;
  y( :,1 ) = [ u;t ];
  for j = 1:n

    %# Taylor-Verfahren

    c(1) = u; c(2) = t * u;
    for l = 3:k+1
      c(l) = t * c(l-1) + c(l-2);
    end
    u = Horner( c,h );
    t = t+h;

    y( :,j+1 ) = [ u;t ];
end
```

Programm 14.1 `TaylorDGL.m`: Lösung der Differentialgleichung aus Beispiel 14.4.

Mittels (14.8) und (14.7) definieren wir dann den "neuen" Wert als gewichtetes Mittel

$$\begin{aligned}\widetilde{u}_{j+1} &= 2u^*_{j+2} - u_{j+1} = u_j + h\, f\left(t^*_{j+1}, u_j + \frac{h}{2} f(t_j, u_j)\right) \\ &= u_j + h\, f\left(t_j + \frac{h}{2}, u_j + \frac{h}{2} f(t_j, u_j)\right).\end{aligned}$$

Das "kleine Wunder" dieses Ansatzes besteht darin, daß diese Methode zur Berechnung eines "neuen" Werts, im Gegensatz zum Polygonzugverfahren[142], eine Fehler der Ordnung $O\left(h^3\right)$. So ergibt eine Taylor–Entwicklung erster Ordnung der zweiten Terms, daß, unter der Annahme $u_j = u\left(t_j\right)$

$$f\left(t_j + \frac{h}{2}, u_j + \frac{h}{2} f(t_j, u_j)\right) = \underbrace{f\left(t_j, u_j\right)}_{=u'(t_j)} + \frac{h}{2}\frac{\partial}{\partial t} f\left(t_j, u_j\right) + \frac{h}{2}\underbrace{f\left(t_j, u_j\right)}_{=u'(t_j)}\frac{\partial}{\partial u} f\left(t_j, u_j\right),$$

was sich mittels

$$u''(t) = \frac{d}{dt}\, u'(t) = \frac{d}{dt} f\left(t, u(t)\right) = \frac{\partial}{\partial t} f\left(t, u(t)\right) + \frac{\partial}{\partial u} f\left(t, u(t)\right)\, u'(t)$$

in

$$f\left(t_j + \frac{h}{2}, u_j + \frac{h}{2} f(t_j, u_j)\right) = u'\left(t_j\right) + \frac{h}{2} u''\left(t_j\right)$$

umformen läßt, weswegen $\widetilde{u}_{j+1} - u\left(t_{j+1}\right) = O\left(h^3\right)$ ist.

14.4 Quadraturformeln

Die meisten numerischen Verfahren zur Lösung von gewöhlichen Differentialgleichungen basieren aber auf Quadraturformeln. Integriert man nämlich (14.1) bezüglich t von t_j bis t_{j+1}, so erhält man

$$u\left(t_{j+1}\right) - u\left(t_j\right) = \int_{t_j}^{t_{j+1}} f\left(t, u(t)\right)\, dt \qquad \Longrightarrow \qquad u_{j+1} = u_j + \int_{t_j}^{t_j + h_j} f\left(t, u(t)\right)\, dt. \tag{14.9}$$

Mit anderen Worten: Um u_{j+1} aus u_j zu bestimmen, braucht man eine gute Näherung für das Integral. Und die kennen wir ja: unsere Quadraturformeln. Eine "einfache" Methode, nämlich die Trapezmethode, liefert beispielsweise

$$u_{j+1} - u_j = \frac{h}{2}\left(f\left(t_j, u_j\right) + f\left(t_{j+1}, u_{j+1}\right)\right),$$

was u_{j+1} *implizit* in Abhängigkeit von u_j festlegt; aber: Zur Berechnung von u_{j+1} muß jedesmal ein nichtlineares Gleichungssystem gelöst werden.

[142] Das, wie man sich leicht überlegt, nur lineare Funktionen korrekt vorhersagt

Also verwendet man für (14.9) “am besten” gleich eine Quadraturformel

$$u_{j+1} = u_j + \sum_{k=0}^{m} w_k \, f\left(\xi_k, u\left(\xi_k\right)\right), \qquad \xi_k \in [t_j, t_{j+1}], \quad k = 0, \ldots, m, \tag{14.10}$$

aber jetzt stehen wir natürlich vor dem Problem, daß wir diese Zwischenwerte $u\left(\xi_k\right)$ kennen müssen.
Fangen wir wieder mal einfach an, also mit der Trapezregel, da sind die beiden Knoten $\xi_0 = t_j$ und $\xi_1 = t_{j+1}$ und $w_0 = w_1 = \frac{h}{2}$. Wir haben also die Regel

$$u_{j+1} = u_j + \frac{h}{2} \underbrace{f\left(\xi_0, u\left(\xi_0\right)\right)}_{=f(t_j, u_j)} + \underbrace{f\left(\xi_1, u\left(\xi_0\right)\right)}_{=f\left(t_{j+1}, u^*_{j+1}\right)},$$

wobei wir eigentlich $u^*_{j+1} = u_{j+1}$ haben müßten – nur leider bringt das nicht viel, es sei denn, wir betrachten

$$u_{j+1} = u_j + \frac{h}{2}\left(f\left(t_j, u_j\right) + f\left(t_{j+1}, u_{j+1}\right)\right) \tag{14.11}$$

als *nichtlineare(s) Gleichung(ssystem)* für u_{j+1} und löst es mit einem geeigneten numerischen Verfahren, z.B., dem Newton–Verfahren.
Ansonsten verwendet man einen *Prädiktor* u^*_{j+1}, den man mittels einer *einfachen* Regel bestimmen können sollte, beispielsweise wieder mal mit unserem guten alten Eulerschen Polygonzugverfahren, also

$$u^*_{j+1} = u_j + h\, f\left(t_j, u_j\right), \qquad u_{j+1} = u_j + \frac{h}{2}\left(f\left(t_j, u_j\right) + f\left(t_j + h, u^*_{j+1}\right)\right).$$

Das ist das Verfahren von *Heun*[143] zur numerischen Integration von gewöhlichen Differentialgleichungen und auch schon ein erstes Beispiel für ein Runge–Kutta–Verfahren. Ein *Runge–Kutta–Verfahren* besteht aus der Bestimmung von

- Knoten ξ_k für die Quadraturformel,
- Prädiktoren für die Werte von $u\left(\xi_k\right)$.

Allgemein setzt man dann also für $k = 0, \ldots, m$

$$\xi_k = t_j + a_k\, h, \qquad 0 \le a_0 < \cdots < a_m \le 1,$$

sowie, mit Werten $v^*_j := f\left(\xi_j, u^*_j\right)$,

$$u^*_k = u_j + \sum_{\ell=0}^{k-1} b_{k\ell}\, h\, v^*_\ell + b_{kk}\, h\, f\left(t_j, u_j\right), \qquad b_{k\ell} \in \mathbb{R}, \quad \ell = 0, \ldots, k,\ k = 0, \ldots, m,$$

und bestimmt die Parameter a_k und $b_{k\ell}$, $\ell = 0, \ldots, k$, $k = 0, \ldots, m$, so, daß das Verfahren möglichst hohe Genauigkeit hat.

[143] Um 1900!

Beispiel 14.5 *Spezialisiert man dieses Verfahren zu* $a_0 = 0$, $a_1 = \frac{1}{2}$, $a_2 = 1$ *und verwendet man die Simpsonregel zur Durchführung der numerischen Integration, dann erhält man die* Methode von Kutta dritter Ordnung *[28], die sich explizit über die Rechenregel*

$$\begin{aligned} v_0^* &= f(t_j, u_j), \\ v_1^* &= f\left(t_j + \frac{h}{2}, u_j + \frac{h}{2} v_0^*\right), \\ v_2^* &= f(t_j + h, u_j - h v_0^* + 2h v_1^*) \\ u_{j+1} &= u_j + \frac{h}{6}(v_0^* + 4v_1^* + v_2^*) \end{aligned}$$

bestimmen läßt. Die Prädiktoren dabei sind

$$\begin{aligned} u_0^* &= u_j \\ u_1^* &= u_j + \frac{h}{2} f(t_j, u_0^*) \\ u_2^* &= u_j - h f(t_j, u_0^*) + 2h f\left(t_j + \frac{h}{2}, u_1^*\right) \end{aligned}$$

Beispiel 14.6 *Das klassischste Verfahren ist die sogenannte* Runge–Kutta–Methode *vierter Ordnung, die man durch "Optimierung" der Parameter erhält und die sich als*

$$\begin{aligned} v_0^* &= f(t_j, u_j), \\ v_1^* &= f\left(t_j + \frac{h}{2}, u_j + \frac{h}{2} v_0^*\right), \\ v_2^* &= f\left(t_j + \frac{h}{2}, u_j + \frac{h}{2} v_1^*\right), \\ v_3^* &= f(t_j + h, u_j + h\, v_2^*), \\ u_{j+1} &= u_j + \frac{h}{6}(v_0^* + 2v_1^* + 2v_2^* + v_3^*), \end{aligned}$$

beschreiben läßt. Ein anderes Verfahren vierter Ordnung hat die Struktur

$$\begin{aligned} v_0^* &= f(t_j, u_j), \\ v_1^* &= f\left(t_j + \frac{h}{3}, u_j + \frac{h}{3} v_0^*\right), \\ v_2^* &= f\left(t_j + \frac{2}{3} h, u_j - \frac{h}{3} v_0^* + h\, v_1^*\right), \\ v_3^* &= f(t_j + h, u_j + h\, u_0^* - h\, v_1^* + h v_2^*), \\ u_{j+1} &= u_j + \frac{h}{8}(v_0^* + 3\, v_1^* + 3\, v_2^* + v_3^*), \end{aligned}$$

und folgt aus der $\frac{3}{8}$*–Regel.*

Bisher haben wir die Schrittweite h immer als eine Art globalen Parameter des Lösungsverfahrens betrachtet und sind einfach "stur" in jedem Schritt um h weitergegangen, das heißt, wir haben $t_{j+1} = t_j + h$ oder $t_j = t_0 + j\,h$ gewählt. Das ist natürlich nicht unbedingt sinnvoll, denn in der Praxis wird es sicherlich sinnvoll sein, die Schrittweite *in Abhängigkeit vom Index* j zu wählen, um so bessere Kontrolle über das Verhalten der berechneten Lösung zu erhalten. Diese Vorgehensweise wird ja auch von unseren bisher hergeleiteten Verfahren unterstützt, die sehr wohl Flexibilität bei der Berechnung von u_{j+1} aus u_j zulassen. Es soll allerdings nicht unterschlagen werden, daß die Wahl der "richtigen" Schrittweite absolut nichttrivial ist.

14.5 Mehrschrittverfahren

Die Idee des Mehrschrittverfahrens besteht darin, die bereits berechneten Werte u_j zu verwenden und lieber die Quadraturformel geeignet anzupassen. Haben wir nämlich schon Näherungswerte $u_1, \ldots, u_j$ und zugehörige Zeitpunkte $t_0, \ldots, t_j$ berechnet[144], dann können wir für ein $m \leq j$ die letzten $m+1$ Werte $u_{j-m}, \ldots, u_j$ verwenden, um das Integral

$$\int_{t_j}^{t_{j+1}} f\left(t, u(t)\right)\, dt$$

anzunähern: Wir bestimmen ein Polynom vom Grad m, das an den Stellen t_{j-k} die Werte $f\left(t_{j-k}, u_{j-k}\right)$ interpoliert, $k = 0, \ldots, m$, und integrieren dieses. Das liefert die Regel

$$u_{j+1} = u_j + \sum_{k=0}^{m} f\left(t_{j-k}, u_{j-k}\right) \underbrace{\int_{t_j}^{t_{j+1}} \ell_k(t)\, dt}_{=:w_k}, \qquad \ell_k\left(t_{j-r}\right) = \delta_{kr}, \quad k, r = 0, \ldots, m. \tag{14.12}$$

Das ist nun eine der "gruseligen" Quadraturformeln, die wir in Bemerkung 13.2 noch ausdrücklich ausschließen wollten: Die meisten der Knoten, nämlich $t_{j-m}, \ldots, t_{j-1}$ liegen nun *nicht* mehr im Integrationsintervall $[t_j, t_{j+1}]$. Die Integrale in (14.12) lassen sich nun verhältnismäßig elementar auswerten und abgesehen davon ist die Programmierung des bestimmten Integrals eines Polynoms eine eher leichte Aufgabe, siehe Programm 14.2. Die einfachsten derartigen Methoden erhält man natürlich, wenn man die Schrittweite h fest wählt. Für $m = 3$ kann man so ganz elementar die Methode von *Adams–Bashfort*

$$u_{j+1} = u_j + \frac{h}{24}\left(55\, f_j - 59\, f_{j-1} + 37\, f_{j-2} - 9\, f_{j-3}\right), \qquad f_j = f\left(t_j, u_j\right)$$

ausrechnen. Das gute an diesen Verfahren ist, daß sie recht geringe Diskretisierungsfehler, nämlich $O\left(h^{m+1}\right)$ haben (es werden ja Polynome der Ordnung m exakt integriert), aber wehe, wenn die Lösung der Differentialgleichung nicht hinreichend oft differenzierbar ist! In diesem Falle wird ja bereits der Fehler durch das Interpolationspolynoms recht groß, und dieses benimmt sich außerhalb der konvexen Hülle seiner Interpolationspunkte (und dort liegt ja der Integrationsbereich!) noch wesentlich unflätiger als innerhalb.

[144] Und hier kommt es nicht darauf an, on wir mit flexibler oder fester Schrittweite gearbeitet haben.

```
%#
%# PolyZeros.m
%# Berechne bestimmtes Integral eines Polynoms
%# Daten:
%#   p  Koeffizientenvektor
%#   a  Integrationsgrenzen
%#   b

function f = PolyInt( p,a,b )
  n = length( p );
  pp = zeros( 1,n+1 );

  for j = 1:n
    pp( j+1 ) = p( j ) / n;
  end

  f = Horner( pp,b ) - Horner( pp,a );
%endfunction
```

Programm 14.2 `PolyInt.m`: Bestimmtes Integral eines Polynoms – eine einfache Sache.

Eine Erweiterung dieser Idee ist die Methode von *Adams–Moulton*[145], bei der an $m+2$ Punkten interpoliert wird und zwar zusätzlich zu t_{j-k}, $k=0,\ldots,m$, noch der "gesuchte" Wert $f\left(t_{j+1},u_{j+1}\right)$ an der Stelle t_{j+1}. Das liefert also die Formel

$$u_{j+1}=w_{-1}\,f\left(t_{j+1},u_{j+1}\right)+\sum_{k=0}^{m}w_k\,f\left(t_{j-k},u_{j-k}\right), \tag{14.13}$$

bei der u_{j+1} nur *implizit* gegeben ist und beispielsweise durch ein Newton–Verfahren ermittelt werden muß. Dafür liegt jetzt aber der Integrationsbereich in der konvexen Hülle der Interpolationspunkte, was das Interpolationspolynom und damit auch die Quadraturformel nicht unbeträchtlich stabilisiert. Die klassische Adams–Moulton–Methode verwende $m=3$ und liefert die Regel

$$u_{j+1}=u_j+\frac{h}{720}\left(251\,f\left(t_{j+1},u_{j+1}\right)+664\,f_j-264\,f_{j-1}+106\,f_{j-2}-19\,f_{j-3}\right).$$

Man sieht übrigens bei beiden expliziten Formeln ein "typsiches" Verhalten dieser Quadraturformeln, deren Knoten nicht mehr gänzlich im Integrationsbereich liegen: Die Gewichte haben oszillierende Vorzeichen und sind deswegen auch numerisch nicht besonders erbaulich.

14.6 Instabilität und Stabilität

Viele Differentialgleichungen haben die unangenehme Eigenschaft, daß sich die Lösungen der Anfangswertaufgaben sehr stark verändern können, wenn man den Anfangswert leicht verändert. Das hat natürlich unmittelbare Auswirkungen auf unsere numerischen Verfahren, die ja ausgehend vom Anfangswert die Näherung der Lösung berechnen: Wenn nun das Verhalten der Lösung auf sehr empfindliche und sensible Art von dem Ausgangswert abhängt, dann werden wir eine extrem hohe Diskretisierungsgenauigkeit, also im wesentlichen eine sehr kleine Schrittweite benötigen. Wir sehen uns dies mal anhand eines Beispiels aus [41, S. 402] an, nämlich der Anfangswertaufgabe

$$u'(t)=\lambda\left(u(t)-g(t)\right)+g'(t),\qquad u\left(t_0\right)=u_0,\qquad g\in C^1(\mathbb{R}). \tag{14.14}$$

Die Lösung der *homogenen* ($g\equiv 0$) Differentialgleichung ist natürlich $u(t)=e^{\lambda t}$, und für $\lambda=0$ ist offenbar $u=g$ eine Lösung. Damit hat die Lösung von (14.14) die Form

$$u(t)=\left(u_0-g\left(t_0\right)\right)\,e^{\lambda(t-t_0)}+g(t), \tag{14.15}$$

wie man auch durch Nachrechnen verifizieren kann. Insbesondere erhält man für $u_0=g\left(t_0\right)$ die Lösung $u_*(t)=g(t)$, modifiziert man hingegen den Anfangswert zu $u_0=\left(t_0\right)+\varepsilon$ für ein (betragsmäßig kleines) $\epsilon\in\mathbb{R}$, dann bekommt man die Lösung

$$u_\varepsilon(t)=\varepsilon\,e^{\lambda(t-t_0)}+g(t),\qquad\Longrightarrow\qquad u_\varepsilon(t)-u_*(t)=\varepsilon\,e^{\lambda(t-t_0)}.$$

[145] Es sieht so aus, als ob der gute Mr. Adams einer der Väter der Mehrschrittverfahren gewesen wäre.

Und je nachdem, ob nun der Realteil von λ größer oder kleiner als 0 ist[146], klingt dieser "Fehler" mit zunehmendem $|t - t_0|$ ab oder aber nimmt zu – in letzterem Fall bringt uns das natürlich in Schwierigkeiten, denn das Problem der Lösung der Differentialgleichung ist dann natürlich äußerst schlecht konditioniert. In diesem Fall muß man also mit Methoden extrem hohere Genauigkeit und noch dazu mit sehr kleinen Schrittweiten arbeiten, was natürlich den Rechenaufwand und auch die Anforderungen an die Rechengenauigkeit hochtreiben wird.

Deswegen führt man den Begriff des *Stabilitätsbereichs* einer Methode zur Lösung einer gewöhnlichen Differentialgleichung ein. Hierzu bemerken wir erst mal, daß die Methode ja von der Differentialgleichung unabhängig sein sollte, weswegen man Stabilität über das einfache "Modellproblem" $u'(t) = \lambda u(t)$, also $f(x, y) = \lambda\, y$, mit der bekannten Lösung $u(t) = e^{\lambda t}$ definiert.

Definition 14.7 *Für ein Einschrittverfahren, das für das "Testproblem" $u'(t) = \lambda u(t)$ eine Regel der Form*

$$u_{j+1} = F\left(\lambda h\right)\, u_j, \qquad j \in \mathbb{N}_0, \tag{14.16}$$

liefert, heißt die Menge

$$B := \{\mu \in \mathbb{C} \,:\, |F(\mu)| < 1\}$$

Gebiet der absouluten Stabilität.

Die Eigenschaft (14.16) ist eine Forderung, die nicht allzu dramatisch ist – für das Eulersche Polygonzugverfahren ist sie evident, denn die Regel ist

$$u_{j+1} = u_j + h\lambda u_j = \underbrace{(1 + \lambda h)}_{=:F(\lambda h)}\, u_j$$

und das zugehörige Gebiet der absoluten Stabilität ist der Einheitskreis mit Mittelpunkt $(-1, 0)$ in $\mathbb{C}$. Für das klassische Runge–Kutta–Verfahren aus Beispiel 14.6 ergibt sich beispielsweise

$$\begin{aligned}
u_0^* &= f\left(t_j, u_j\right) = \lambda\, u_j \\
u_1^* &= f\left(\cdot, u_j + \frac{h}{2} u_0^*\right) = \lambda\left(u_j + \frac{h}{2}\lambda u_j\right) = u_j\left(\lambda + \frac{h}{2}\lambda^2\right) \\
u_2^* &= f\left(\cdot, u_j + \frac{h}{2} u_1^*\right) = \lambda\left(u_j + \frac{h}{2} u_1^*\right) = u_j\left(\lambda + \frac{h}{2}\lambda^2 + \frac{h^2}{4}\lambda^3\right) \\
u_3^* &= f\left(\cdot, u_j + h\, u_2^*\right) = u_j\left(\lambda + h\,\lambda^2 + \frac{h^2}{2}\lambda^3 + \frac{h^3}{4}\lambda^4\right) \\
u_{j+1} &= u_j + \frac{h}{6}\left(u_0^* + 2u_1^* + 2u_2^* + u_3^*\right) = \left(1 + h\,\lambda + \frac{1}{2}(h\lambda)^2 + \frac{1}{6}(h\lambda)^3 + \frac{1}{24}(h\lambda)^4\right)\, u_j,
\end{aligned}$$

aber das Stabilitätsgebiet wird jetzt natürlich um einiges komplizierter.

Ist nun $\lambda \in \mathbb{R}_+$, so ist mit der Berechnungsformel (14.16) eigentlich "alles im grünen Bereich", denn die Lösungsberechnung erfolgt genau entsprechend der Bildungsregel der Exponentialfunktion. Für $\lambda < 0$ wird es ein bißchen ekliger, denn in diesem Fall hat man die

[146] In $\mathbb{C}$ ist das die Frage, ob λ in der linken oder rechten Halbebene liegt ...

permanenten Vorzeichenwechsel und es ist gar nicht so klar, was man jetzt machen muß. Leider ist aber $\Re\lambda < 0$ angeblich gerade der Fall, der häufig in naturwissenschaftlichen Anwendungen auftritt, also die Situation "abklingender" Lösungen der Differentialgleichung. Und hier spielen nun Stabilität und Schrittweite zusammen:

> *Die Schrittweite h ist so zu wählen, daß für alle λ mit* negativem *Realteil stets $h\lambda \in B$ liegt.*

14.7 Systeme und Steifigkeit

Die Beziehung zwischen Schrittweite und Stabilität kann nun bei *Systemen* von gewöhnlichen Differentialgleichungen zu einem Dilemma führen.

Beispiel 14.8 *Wir betrachten [41, S. 410] das lineare System*

$$u' = \begin{bmatrix} u_1' \\ u_2' \\ u_3' \end{bmatrix} = \begin{bmatrix} -0.5 & 32.6 & 35.7 \\ & -48 & 9 \\ & 9 & -72 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix}, \qquad u(0) = \begin{bmatrix} 4 \\ 13 \\ 1 \end{bmatrix}$$

von Differentialgleichungen erster Ordnung. Der Lösungsansatz $u_j(t) = a_j\, e^{\lambda t}$ liefert, eingesetzt, ein Eigenwertproblem aus dem sich unter Berücksichtigung der Anfangswerte die Lösung

$$u(t) = \begin{bmatrix} 15\, e^{-t/2} - & 12\, e^{-45\, t} + & e^{-75\, t} \\ & 12\, e^{-45\, t} + & e^{-75\, t} \\ & 4\, e^{-45\, t} - & 3\, e^{-75\, t} \end{bmatrix}$$

ergibt. Nun wird zwar das "Langzeitverhalten" im wesentlichen durch die Komponente $e^{-t/2}$ bestimmt, doch um eine hinreichende numerische *Genauigkeit zu gewinnen, muß sich die Schrittweite an dem betragsmäßig größten Wert, also an $e^{-75\, t}$ orientieren.*

Definition 14.9 *Ein System von Differentialgleichungen heißt* steif*, falls die* Steifigkeit *oder* Steifheit

$$S := \frac{\max\limits_j |\Re\lambda_j|}{\min\limits_j |\Re\lambda_j|}$$

groß ist.

Die Steifigkeit des Systems aus Beispiel 14.8 ist also 150, was fast noch schlimmer klingt, als es ist: In realistischen Situation kann die Steifigkeit ohne weiteres in Größenordnungen von 10^3 oder 10^6 vorstoßen.

Eine Strategie, steife Systeme zu lösen, besteht darin, daß man Verfahren wählt, deren Stabilitätsbereich die ganze Halbebene $\{z \in \mathbb{C} \,:\, \Re z < 0\}$ ist. Und solche gibt es tatsächlich, nämlich die Trapezmethode (14.11) und die impliziten Runge–Kutta–Verfahren[147]; aber in beiden Fällen müssen dann nichtlineare Gleichungen gelöst werden. Es gibt eben nichts umsonst[148]. Und das ist ein schönes Schlusswort!

[147] Hier werden t_{j+1} und u_j in die Quadraturformel miteingebaut.

[148] "There is no free lunch", David Lodge, *Nice work*

14.8 Galerkin–Verfahren

Es hätte ein schönes Schlusswort sein können, aber nachdem noch Zeit ist, können wir uns noch ein bisschen mehr zum Thema "gewöhliche Differentialgleichungen" ansehen, wobei wir uns weniger auf die Theorie als vielmehr auf Lösungsverfahren konzentrieren werden. Wir haben es noch nicht explizit gesagt, aber alle numerischen Lösungsverfahren *diskretisieren* auf irgendeine Weise die Lösung - das ist ja auch naheliegend, denn wir können halt am Computer nur endlich viel Information speichern.

Unsere bisherigen Strategien beruhten im wesentlichen darauf, die Lösungsfunktion *zeitlich* zu diskretisieren, das heißt, an den (Zeit-) Punkten $t_0 + j\,h$, $j = 0, 1, 2, \ldots$, zu bestimmen. Der Ansatz bei den Galerkin-Verfahren ist ein anderer: Hier bestimmen wir eine Funktion, die aber dafür nur durch endlich viel Information beschrieben ist. Fangen wir einfach an.

Definition 14.10 *Ein linearer Differentialoperator D der Ordnung n mit konstanten Koeffizienten hat die Form*

$$D = a\left(\frac{d}{dt}\right) = \sum_{j=0}^{n} a_j \frac{d^j}{dt^j}, \qquad a_j \in \mathbb{R}, \quad a \in \Pi_n,$$

also

$$Df = \sum_{j=0}^{n} a_j\, f^{(j)}.$$

Wir möchten nun das Problem $Du = g$ lösen, wahlweise als *Anfangswertproblem*

$$Du = g, \qquad u\,(t_0) = u_0, \tag{14.17}$$

oder als *Randwertproblem*

$$Du = g, \qquad u\,(t_0) = u_0, \quad u\,(t_1) = u_1 \tag{14.18}$$

mit vorgegebenen Randwerten u_0, u_1. Jetzt diskretisieren wir nicht mehr die Zeit sondern die Funktion! Dazu verwenden wir einen *endlichdimensionalen Funktionenraum* $\mathscr{F}$ mit $D\mathscr{F} \subseteq \mathscr{F}$, auf dem D definiert ist. Sei außerdem $\Phi \subset F$ eine Basis von F, dann hat jede Funktion $f \in \mathscr{F}$ eine Darstellung der Form

$$f = \sum_{\varphi\in\Phi} f_\varphi\, \varphi$$

und unsere Differentialgleichung nimmt für $u, g \in \mathscr{F}$ die Form

$$\sum_{\varphi\in\Phi} g_\varphi\, \varphi = g = Du = D\left(\sum_{\varphi\in\Phi} u_\varphi\, \varphi\right) = \sum_{\varphi\in\Phi} u_\varphi\, D\varphi \tag{14.19}$$

an. Weil D ein *linearer* Operator ist, der $\mathscr{F}$ in sich selbst abbildet, kann aber $D\phi$ wieder als Linearkombination von Φ dargestellt werden, das heißt, es gibt Koeffizienten $d_{\varphi,\varphi'} \in \mathbb{R}$, so daß

$$D\varphi = \sum_{\varphi'\in\Phi} d_{\varphi,\varphi'}\, \varphi'. \tag{14.20}$$

Setzen wir jetzt noch (14.20) in (14.19) ein, dann erhalten wir, daß

$$\sum_{\varphi\in\Phi} g_\varphi\,\varphi = \sum_{\varphi\in\Phi} u_\varphi \sum_{\varphi'\in\Phi} d_{\varphi,\varphi'}\,\varphi' = \sum_{\varphi'\in\Phi}\left(\sum_{\varphi\in\Phi} d_{\varphi,\varphi'}u_\varphi\right)\,\varphi'$$

und ein Koeffizientenvergleich führt sofort zu dem linearen Gleichungssystem

$$g_\varphi = \sum_{\varphi'\in\Phi} d_{\varphi',\varphi}u_{\varphi'},\quad \varphi\in\Phi, \qquad \text{bzw.} \qquad \boldsymbol{g} = [d_{\varphi',\varphi}\ :\ \varphi,\varphi'\in\Phi]\,\boldsymbol{f} =: D_\Phi \boldsymbol{f}, \tag{14.21}$$

wobei $\boldsymbol{g} = [g_\varphi\ :\ \varphi\in\Phi]$ und $\boldsymbol{u} = [u_\varphi\ :\ \varphi\in\Phi]$ die Koeffizientenvektoren zu g und u sind. Ach ja - und die Randbedingungen tun uns in diesem Kontext natürlich auch nicht weh, die führen lediglich zu einer oder zwei weiteren Nebenbedingung(en).

Beispiel 14.11 *Lösen wir doch mal die Differentialgleichung $u'' = -u$, also $u''+u=0$ mit Hilfe des Galerkin-Verfahrens und unter Verwendung der Polynomräume*[149] *$\mathscr{F} = \Pi_n$, $n\in\mathbb{N}$. Für die Monombasis gilt natürlich*

$$D(\cdot)^k = \frac{d^2}{dx^2}x^k + (\cdot)^k = k(k-1)(\cdot)^{k-2} + (\cdot)^k,$$

was zur Diskretisierungsmatrix

$$D_\Phi = \begin{bmatrix} 1 & 0 & 2 & & & & \\ & 1 & 0 & 6 & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & 1 & 0 & n(n-1) & \\ & & & & 1 & 0 & \\ & & & & & 1 & \end{bmatrix}$$

führt. Nehmen wir nun beispielsweise als Anfangswert $u(0)=1$ und $u'(0)=0$, dann müssen wir gerade mal eine Zeile hinzufügen und landen bei den überbestimmten *Gleichungssystemen*

$$\begin{bmatrix} 1\\0\\0\\0\\\vdots\\0\\0\\0 \end{bmatrix} = \widehat{D}_\Phi\,\boldsymbol{u} = \begin{bmatrix} 1 & & & & & & \\ & 1 & & & & & \\ 1 & 0 & 2 & & & & \\ & 1 & 0 & 6 & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & 1 & 0 & n(n-1) & \\ & & & & 1 & 0 & \\ & & & & & 1 & \end{bmatrix}\boldsymbol{u}.$$

[149] Wir wählen die Polynome hier **nicht**, weil sie besonders geeignet wären, sondern lediglich, weil sie einen endlichdimensionalen Raum von Funktionen bilden, die sich besonders einfach ableiten lassen!

Eine exakte Lösung ist natürlich unmöglich! Die Matrix D_Φ ist offensichtlich invertierbar und deswegen müsste wegen der letzten $n+1$ Gleichungen $\boldsymbol{u} = 0$ sein, was nun wieder der ersten Gleichung entspricht. Also bleibt nur eine Least-Squares-Lösung, und die wählt `Octave` *ja sowieso automatisch.*

Das Ergebnis hängt natürlich von der Dimension des Polynomraus und der daraus resultierenden "Flexibilität" ab: Während die Näherung für kleine Grade wie in Abb. 14.5 nicht viel mit der zu erwartenden Lösung zu tun hat, erhält man für hohe Grade durchaus ansprechende und sinusähnlich Ergebnisse, siehe Abb. 14.6.

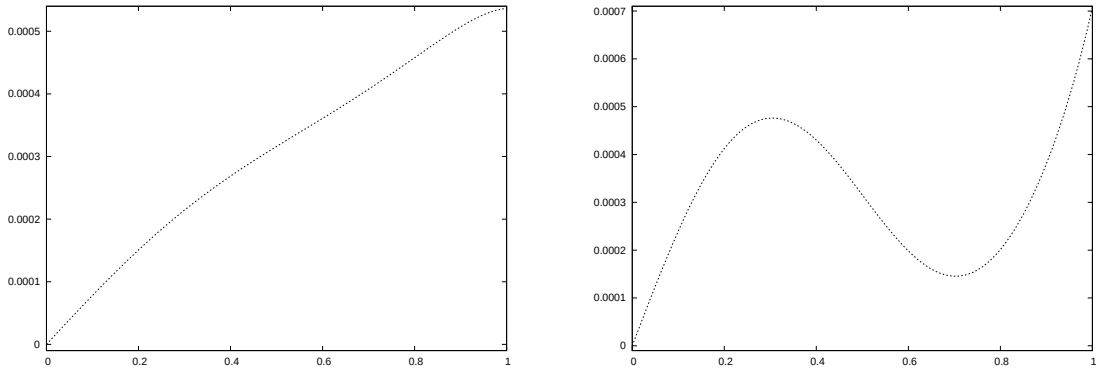


Abbildung 14.5: Lösungen der Differentialgleichung aus Beispiel 14.11 mit Polynomen der Grade 12 *(links)* und 14 *(rechts)*. Die so erhaltenen Funktionen haben mit dem Sinus, den man bekommen sollte, noch nicht so richtig viel zu tun.

Das Beispiel zeigt eines ganz deutlich: Ohne Informationen, wie weit der Ansatzraum $\mathscr{F}$ von den potentiellen Lösungen "entfernt" ist, also wie gut sich Lösungen der Differentialgleichung durch Funktionen aus $\mathscr{F}$ approximieren lassen, geht nicht unbedingt viel. Und die Herleitung solcher Aussagen ist auch ein wesentlicher Aspekt der Numerik gewöhnlicher und vor allem *partieller* Differentialgleichungen.

Und weil wir gerade bei den allgemeinen Ideen sind: Beispiel 14.11 hat uns ja ziemlich deutlich vor Augen geführt, daß die Gleichunggsysteme, mit denen wir es bei der naiven Diskretisierung von Differentialoperatoren herumzuschlagen haben, oftmals überbestimmt und ohnehin nicht exakt lösbar sein können. Dann können wir aber auch gleich bei die Diskretisierung die Forderung, daß $Du = g$ sein soll, durch die Orthogonalitäts- bzw. Projektionsbedingung

$$\langle Du - g, \mathscr{F} \rangle\,, = 0 \qquad \text{d.h.} \qquad \langle Du - g, f \rangle = 0, \quad f \in \mathscr{F},$$

ersetzen, das heißt, daß

$$0 = \left\langle \sum_{\varphi\in\Phi} u_\varphi\, D\varphi - \sum_{\varphi\in\Phi} g_\varphi\, \varphi, \varphi' \right\rangle = \sum_{\varphi\in\Phi} \langle D\varphi, \varphi' \rangle\ u_\varphi - \sum_{\varphi\in\Phi} \langle \varphi, \varphi' \rangle\ g_\varphi, \qquad \varphi' \in \Phi$$

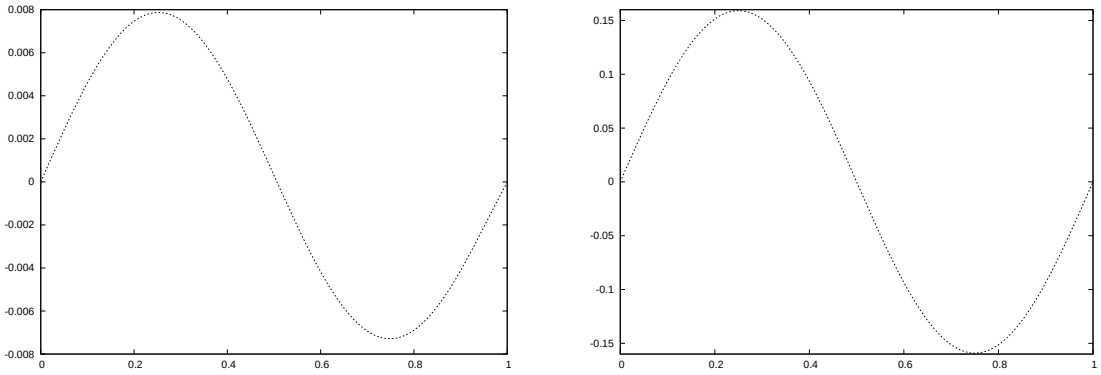


Abbildung 14.6: Die Situation ändert sich im Vergleich zu Abb. 14.5 ziemlich drastisch, wenn man den Polynomgrad noch ein klein wenig erhöht - bei Grad 15 *(links)* sieht das Ganze schon ziemlich so aus, wie es soll, beim Übergang zu noch höheren Graden wie 25 *(links)* ändert sich dann nur noch die Normierung der Funktion, doch auch das wird dann ziemlich schnell stationär.

und somit

$$\left[\langle D\varphi, \varphi'\rangle \ : \ \begin{array}{c}\varphi' \in \Phi \\ \varphi \in \Phi\end{array}\right] \boldsymbol{u} = \left[\langle \varphi, \varphi'\rangle \ : \ \begin{array}{c}\varphi' \in \Phi \\ \varphi \in \Phi\end{array}\right] \boldsymbol{g} \tag{14.22}$$

sein muss. Dabei hat die Darstellung aus (14.22) noch einen weiteren Vorteil.

Beispiel 14.12 *Kehren wir zurück zu unserer Differentialgleichung $u''(t) = -u(t)$, und nehmen wir an, daß wir beim Skalarprodukt partiell integrieren dürfen*[150]*, dann ist für $\varphi, \psi \in \Phi$*

$$\langle D\varphi, \psi\rangle = \langle \varphi'' + \varphi, \psi\rangle = -\langle \varphi', \psi'\rangle + \langle \varphi, \psi\rangle \tag{14.23}$$

*und obwohl wir eine Differentialgleichung **zweiter** Ordnung zu lösen haben, brauchen die Ansatzfunktionen aus Φ in der* schwachen Form *(14.23) der Differentialgleichung nur einmal stetig differenzierbar zu sein.*

Die schwache Form (14.23) der Differentialgleichung hat noch einen weiteren Vorteil: Die *Diskretisierungsmatrix*

$$D_\Phi := [-\langle \varphi', \psi'\rangle + \langle \varphi, \psi\rangle \ : \ \phi, \psi \in \Phi]$$

ist jetzt sogar symmetrisch! Daß sowas nur mit Differentialoperatoren *gerader* Ordnung funktionieren kann, ist natürlich klar. Wählt man außerdem die Basisfunktionen in Φ so, daß sie möglichst kleinen und disjunkten Träger haben, dann ist D_Φ auch noch eine *bandierte* Matrix und damit numerisch effizient handhabbar, beispielsweise mit dem *Gauß-Seidel-Verfahren*,

[150] Daß die Randwerte also auf Null gesetzt sind, gerade bei *partiellen* Differentialgleichungen, bei denen Galerkin-Verfahren ohnehin viel öfter zum Einsatz kommen, ist das eine relativ "normale" Situation.

siehe z.B. [36]. Und Funktionen mit kompaktem Träger und regelbarer Glattheit erinnern uns doch an etwas - richtig, die guten alten *B-Splines* aus Teil I sind sicherlich gute Kandidaten für Ansatzfunktionen, auch wenn es natürlich ein paar Details gibt, die man noch berücksichtigen müsst.

Man könnte[151] noch eine ganze Menge über Galerkin-Verfahren sagen[152], aber wir wollen uns lieber noch einen Diskretisierungsansatz ansehen, der bei Ingenieuren sehr beliebt ist und uns einen kurzen Ausflug ins Reich der Systemtheorie bescheren wird.

14.9 Differential- und Differenzengleichungen

Eigentlich kennen wir Diskretisierungen von Differentialoperatoren sogar ein klein wenig länger als die Differentialoperatoren selbst, wird doch die Differentiation in der Analysis meist über die *Differentialquotienten*

$$\frac{f(x+h)-f(x)}{h}$$

eingeführt. Vergessen wir mal kurz das h im Nenner, dann steht da nichts weiter als der Differenzenoperator Δ_h, definiert durch

$$\Delta_h f = f\left(\cdot + h\right) - f,$$

und ein[153] Differentialoperator[154] $D = a\left(d/dt\right)$ kann dann durch den *Differenzenoperator*

$$a\left(\Delta_h\right) = \sum_{j=0}^{n} a_j \, \Delta_h^j$$

"ersetzt" werden.

Übung 14.1 Zeigen Sie, daß für $n \in \mathbb{N}$

$$\Delta_h^n f = (-1)^n \sum_{k=0}^{n} \binom{n}{k} (-1)^k \, f\left(\cdot + kh\right). \tag{14.24}$$

gilt. ♢

Gleichung (14.24) zeigt, daß Δ_h^n ein *Vorwärtsoperator* ist, der nur Werte von f rechts vom Auswertungspunkt der Differenz verwendet. Man kann natürlich auch den *Rückwärtsoperator* $-\Delta_{-h}^n$ oder den *zentrierten* Differenzenoperator $\Delta_h^n f\left(\cdot - nh/2\right)$ verwenden, das ist reine Geschmackssache[155]. Machen wir das alles mal ein wenig formaler.

Definition 14.13 (Differenzenoperatoren)

[151] Das ist offensichtlich das Kapitel der Konjunktive.

[152] Was wieder einmal ein Thema für eine "eigene" Spezialvorlesung wäre, die wohl nie gehalten werden wird.

[153] Der Einfachheit halber linearer ...

[154] Mit der Einfachheit halber konstanten Koeffizienten ...

[155] Nicht ganz: Beispielsweise in der Bildverarbeitung ist es ziemlich wichtig, *symmetrische*, also zentrierte Operatoren zu verwenden, denn sonst läuft die Lokalisierung von Ecken und Kanten relativ leicht aus dem Ruder.

1. *Der* Translationsoperator τ_h *ist definiert durch* $\tau_h f = f(\cdot + h)$.

2. *Die* Diskretisierung *des Differentialoperators* $D = a(d/dt)$, *mit* Schrittweite $h > 0$ *ist der Differenzenoperator*

$$h^{\deg a} a\left(h^{-1}\Delta_h\right) = \sum_{j=0}^{n} h^{n-j}\, a_j\, \Delta_h^j = \sum_{j=0}^{n} b_j(h)\, \tau_h^j.$$

3. *Die* Abtastung *einer Funktion* f *mit* Gitterweite $h > 0$ *auf dem Intervall* $[a, b]$ *ist der Vektor*

$$\sigma_h f := \left[f(a + jh) \;:\; 0 \le j \le \frac{b-a}{h} \right].$$

4. *Die Diskretisierung der Differentialgleichung* $Du = g$ *ist dann die Differenzengleichung*

$$h^{\deg a} a\left(h^{-1}\Delta_h\right)\, \boldsymbol{u} = h^{\deg a} \sigma_h g \tag{14.25}$$

mit dem Lösungsvektor oder der Lösungsfolge $\boldsymbol{u} = \sigma_h u$.

Für jedes feste $h > 0$ erhalten wir somit also als Diskretisierung eines Anfangswertproblems ein Differenzengleichungssystem der Form

$$\sum_{k=0}^{n} a_k u_{j+k} = g_j, \quad j = 0, \ldots, m, \qquad u_j = v_j, \quad j = 0, \ldots, n-1, \tag{14.26}$$

mit *Anfangswerten* $v_0, \ldots, v_{n-1}$. Dabei machen wir die Annahme, daß

$$a_0\, a_n \neq 0 \tag{14.27}$$

sein soll, daß wir also im Vektor a in (14.26) keine "überflüssigen" Koeffizienten mitschleppen müssen.

Lemma 14.14 (Lösungen der Differenzengleichung)

1. *Unter der Voraussetzung (14.27) hat das Differenzengleichung (14.26) genau eine Lösung.*

2. *Der Lösungsraum des Differenzengleichungssystems*

$$\sum_{k=0}^{n} a_k u_{j+k} = g_j, \quad j = 0, \ldots, m,$$

***ohne** Anfangswerte hat Dimension* n.

Beweis: Wir formen den ersten Teil von (14.26) in

$$u_{n+j} = \frac{1}{a_n}\left(g_j - \sum_{j=0}^{n-1} a_k\, u_{j+k}\right), \qquad j = 0, \ldots, m,$$

um, was nach (14.27) erlaubt ist und womit u_{n+j} *eindeutig* durch $u_j, \ldots, u_{j+n-1}$ festgelegt wird. Das beweist dann auch schon die erste Behauptung und die zweite folgt, indem wir für $j = 0, \ldots, n-1$ einfach die offensichtlich linear unabhängigen Lösungen zu den Anfangswerten

$$u_k = \delta_{jk}, \qquad k = 0, \ldots, n-1,$$

betrachten. □

Es ist recht illustrativ, sich (14.26) einmal als ein lineares Gleichungssystem hinzuschreiben, nämlich – zuerst einmal ohne Nebenbedingungen – als

$$\begin{bmatrix} a_0 & \ldots & a_n & & & \\ & a_0 & \ldots & a_n & & \\ & & \ddots & & \ddots & \\ & & & a_0 & \ldots & a_n \end{bmatrix} \begin{bmatrix} u_0 \\ \vdots \\ u_{n+m} \end{bmatrix} = \begin{bmatrix} g_0 \\ \vdots \\ g_m \end{bmatrix} \tag{14.28}$$

und mit mit Nebenbedingungen als

$$\underbrace{\begin{bmatrix} 1 & & & & & \\ & \ddots & & & & \\ & & 1 & & & \\ a_0 & \ldots & a_{n-1} & a_n & & \\ & \ddots & & \ddots & \ddots & \\ & & a_0 & \ldots & a_{n-1} & a_n \end{bmatrix}}_{=:A} \underbrace{\begin{bmatrix} u_0 \\ \vdots \\ u_{n+m} \end{bmatrix}}_{=:u} = \underbrace{\begin{bmatrix} v_0 \\ \vdots \\ v_{n-1} \\ g_0 \\ \vdots \\ g_m \end{bmatrix}}_{=:g}.$$

Matrizen wie in (14.28), die man dadurch erhält, daß man einen Vektor entlang der Hauptdiagonalen verschiebt, bezeichnet man als *Toeplitz–Matrizen*, und sie spielen eine zentrale Rolle (nicht nur) in der digitalen Signalverarbeitung [20, 29, 39].

Beispiel 14.15 *Betrachten wir wieder einmal unsere "Mustergleichung"*

$$u'' + u = 0, \qquad u(0) = 1, \quad u'(0) = 0,$$

deren h–Diskretisierung sich[156] *als*

$$0 = h^2\left(\frac{u_{j+2} - 2u_{j+1} + u_j}{h^2} + u_j\right) = u_{j+1} - 2u_{j+1} + \left(1 + h^2\right) u_j,$$

[156] In nichtzentrierter Form.

mit den Anfangswerten

$$u_0 = 1, \quad \Delta u_0 = 0, \qquad \Leftrightarrow \qquad u_0 = u_1 = 1,$$

also

$$a = \begin{bmatrix} 1 \\ -2 \\ 1+h^2 \end{bmatrix}, \qquad v = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

sowie die untere Dreiecksmatrix

$$A = \begin{bmatrix} 1 & & & & \\ & 1 & & & \\ 1 & -2 & 1+h^2 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1+h^2 \end{bmatrix}$$

ergibt. In `octave` *geht das dann alles sehr einfach:*

```
octave> h = .1; m = floor( 2*pi/h ) + 1; a = [ 1 -2 1+h^2 ];
octave> A = toeplitz ( [ 1,zeros( 1,m-3) ], \
> [ a,zeros( 1,m-length(a) ) ] );
octave> B = [ eye(2), zeros( 2,m-2) ; A ];
octave> b = [ 1,1,zeros( 1,m-2) ]';
octave> u = B\b;
```

Und daß sich die Ergebnisse dieser Methode durchaus sehen lassen kann, das zeigt ja bereits Abb. 14.7.

Differenzengleichungen sind durchaus gute Methoden, um Differentialgleichungen anzupacken, allerdings muss natürlich die Differenzengleichung zur Differentialgleichung "passen". Wir haben hier den Ansatz verfolgt, die Ableitungsoperatoren durch Differenzenquotienten zu ersetzen, aber das ist natürlich nicht der einzig mögliche Weg. Gemäß [25] heisst eine Differenzengleichung der Form (14.26) *konsistent* mit der Differentialgleichung $Du = g$, wenn für die beiden Lösungen die Beziehung

$$\lim_{h \to 0} \sup_{j=0,\ldots,m} |u_j(h) - u\,(a + hj)| = 0$$

erfüllt ist. Eine Differentialgleichung kann also immer durch eine entsprechende konsistente Differenzengleichung gelöst werden. Allerdings sollte man ein paar Punkte beachten:

1. Die Diskretisierung hier ist *globaler* Natur, also wieder nicht adaptiv! Der Aufwand steigt also linear mit h^{-1}.

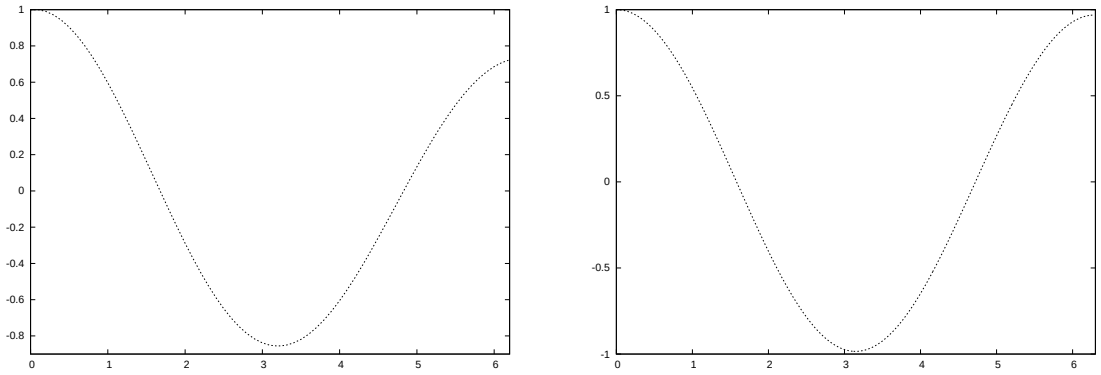


Abbildung 14.7: Lösung der Differenzengleichung für $h = 0.1$ *(links)* und $h = 0.01$ *(rechts)*. Daß die höhere Diskretisierungsgenauigkeit auch Vorteile beim Ergebnis bringt, das sollte uns nicht überraschen.

2. Irgendwann ist das auch numerisch nicht unproblematisch! Betrachten wir die Differenzengleichung aus Beispiel 14.15, dann haben wird dort den Koeffizienten $1 + h^2$, und ist h nun sehr klein, dann ist[157] h^2 noch viel kleiner und wird irgendwann von der 1 bei der Addition unterdrückt, man erinnere sich nur an die Rundungsfehleranalysen aus [36]. Diskretisiert man also zu fein, dann wird die Diskretiserung möglicherweise ziemlich ungenau.

Alles in allem gibt uns das aber trotzdem eine ausreichende Rechtfertigung, Differenzengleichungen auch per se zu untersuchen. Immerhin wird es uns zum numerischen Verfahren Nummer 1, zur FFT, führen.

14.10 Lineare Systeme und die FFT

Als erstes einmal betten wir unsere Differenzengleichungen in die Welt der Folgen ein, betrachten jetzt also

$$g_j = \sum_{k=0}^{n} a_k \, u_{j+k}, \quad j \in \mathbb{Z}, \qquad (u_0, \ldots, u_{n-1}) = \boldsymbol{v} \in \mathbb{R}^n. \tag{14.29}$$

Die Lösungen aus dem vorangegangenen Kapitel sind also lediglich Komponenten $0, \ldots, n$ des *diskreten Signals* u.

Definition 14.16 *Der (doppeltunendlich) Folgenraum $\ell(\mathbb{Z})$ wird als Menge der* diskreten Signale *bezeichnet.*

[157] Welche Binsenweisheit!

Wenn wir schon dabei sind, dann können wir eigentlich auch unsere Koeffizienten a der Differenzengleichung aus $\ell(\mathbb{Z})$ wählen, allerdings eben nur als eine *endliche* Folge, bzw. als ein *endliches Signal*. Um zu sehen, warum man bei Differenzengleichungen ausgerechnet von *linearen Systemen* spricht, brauchen wir noch ein wenig Terminologie.

Definition 14.17 (Korrelation & Symbole)

1. *Die* Korrelation[158] *zweier Signale* $a, b \in \ell(\mathbb{Z})$ *ist definiert als*

$$a \star b := \left(\sum_{k\in\mathbb{Z}} a_k b_{j+k} \ : \ j \in \mathbb{Z} \right) = \left(\sum_{k\in\mathbb{Z}} a_{k-j} b_k \ : \ j \in \mathbb{Z} \right) = (b \star a)(-\cdot),$$

   wobei wir a *und* b *einfach mit Nullen auf ganz* $\mathbb{Z}$ *fortsetzen.*

2. *Das* Symbol *oder die* erzeugende Funktion *eines Signals* $a \in \ell(\mathbb{Z})$ *ist die formale*[159] *Laurentreihe* [160]

$$a(z) = \sum_{k\in\mathbb{Z}} a_k \, z^k.$$

Bemerkung 14.18 *Die Einbettung in* $\ell(\mathbb{Z})$ *verschenkt nichts! Sind* a, b *beide nur für nichtnegative Indizes von Null verschieden, also sogenannte* kausale Signale*, dann wird diese Eigenschaft auch von der Korrelation geteilt. Und es ist niemand gezwungen, Signale mit* $a_{-k} \neq 0$ *für das eine oder andere* $k \in \mathbb{N}$ *zu betrachten - mit einer kleinen Verschiebung bekommt man sie immer in den "positiven Bereich".*

Die Korrelation ist eine *faltungsähnliche* Operation. Zur Erinnerung: Die *Faltung* zweier Folgen ist definiert als

$$(a * b)_j := \sum_{k\in\mathbb{Z}} a_{j-k} \, b_k = \sum_{k\in\mathbb{Z}} a_k \, b_{j-k},$$

der kleine aber feine Unterschied zwischen Faltung und Korrelation besteht also nur in der "Richtung", in die b summiert wird. Nach dieser kurzen Begriffsklärung können wir auch schon erkennen, warum man bei Differenzengleichungen auch von *linearen Systemen* spricht - im Kontext der Symbole wird nämlich aus der Korrelation ein einfaches Produkt.

Lemma 14.19 *Für* $a, b \in \ell(\mathbb{Z})$ *ist*

$$(a \star b)(z) = a\left(z^{-1}\right) \, b(z). \tag{14.30}$$

[158] Hinter der Korrelation im Sinne der Signalverarbeitung verbirgt sich schon im wesentlichen dieselbe Idee wir hinter der Korrelation im Sinne der Stochastik; Details finden sich beispielsweise in [1, 27, 32].

[159] Also eine Potenzreihe, sogar mit negativen Exponenten, bei der wir uns dafür aber **nicht** für Konvergenz interessieren! Diese formalen Potenzreihen kann man (komponentenweise) addieren und (mit dem Cauchyprodukt) multiplizieren.

[160] Laurentreihe!formale

Beweis: Eigentlich ganz einfach, denn die Definition ergibt unmittelbar[161]

$$(a \star b)(z) \;=\; \sum_{j\in\mathbb{Z}} (a\star b)_j \; z^j = \sum_{j\in\mathbb{Z}}\sum_{k\in\mathbb{Z}} a_k\, b_{j+k}\, z^{j+k} z^{-k} = \left(\sum_{k\in\mathbb{Z}} a_k z^{-k}\right)\left(\sum_{j\in\mathbb{Z}} b_j\, z^j\right),$$

was gerade (14.30) ist. □

Aus (14.30) können wir eine beinahe schon banale Metode herleiten, um die Differenzengleichung $a \star u = g$ zu lösen, nämlich indem wir

$$g(z) = a\left(z^{-1}\right)\, u(z) \qquad \Leftrightarrow \qquad u(z) = \frac{g(z)}{a\left(z^{-1}\right)}$$

verwenden - alles, was wir brauchen ist eine Entwicklung

$$\frac{1}{a\left(z^{-1}\right)} = \sum_{j=0}^{\infty} b_j\, z^{-j} =: b\left(z^{-1}\right),$$

denn dann könnten wir - vorausgestzt, das würde eine hinreichend gute Näherung ergeben - den Vektor b abschneiden und erhielten eine Methode, die Differenzengleichung durch einfache Korrelation zu lösen.

Definition 14.20 *Die Differenzengleichung mit Koeffizienten a heißt* stabil, *wenn alle Nullstellen von $a(z)$* innerhalb *des Einheitskreises liegen.*

Satz 14.21 *Ist a stabil, dann gibt es eine Konstante $C > 0$ und $\rho \in (0,1)$, so daß $|b_j| \le C\,\rho^j$, $j \in \mathbb{N}$, das heißt, so daß die Koeffizienten von b exponentiell abklingen*[162].

Korollar 14.22 *Ist a stabil, so gilt für die Lösung der Differenzengleichung, daß*

$$\lim_{j\to\infty} u_j = 0$$

ist.

Lemma 14.23 *Für $\zeta \in \mathbb{C}$ ist*

$$\frac{1}{z^{-1}-\zeta} = \frac{z}{1-\zeta\, z} = \sum_{j=0}^{\infty} \zeta^j\, z^{j+1}.$$

[161] Hier erweist sich die Einbettung der endlichen Folgen in $\ell(\mathbb{Z})$ als sehr hilfreich, man kann in den Summen ganz einfach die Indizes verschieben.

[162] Und man damit abschneiden kann!

Beweis: Wieder einmal die harmonische Reihe:

$$(1-\zeta\,z)\sum_{j=0}^{\infty}\zeta^j\,z^j=\sum_{j=0}^{\infty}\zeta^j\,z^j-\sum_{j=1}^{\infty}\zeta^j\,z^j=1,$$

also

$$\frac{z}{1-\zeta\,z}=z\sum_{j=0}^{\infty}\zeta^j\,z^j$$

wie behauptet. □

Für ein lineares Symbol zeigt uns Lemma 14.23 als, daß die Koeffizienten der Inversen genau dann exponentiell afallen, wenn die Nullstelle des Symbols *im* Einheitskreis liegt und exponentiell ansteigen, wenn die Nullstelle außerhalb des Einheitskreises liegt. Für Potenzen eines linearen Polynoms ist das auch nicht schlimmer.

Lemma 14.24 *Für $n\in\mathbb{N}$ gibt es ein Polynom $p_n\in\Pi_{n-1}$, so daß für jedes $\zeta\in\mathbb{C}$*

$$\frac{1}{(z^{-1}-\zeta)^n}=\sum_{j=0}^{\infty}p_n(j)\,\zeta^j\,z^{j+n}$$

Beweis: Induktion über n, der Fall $n=1$ ist gerade Lemma 14.23. Ansonsten ist

$$\begin{aligned}\frac{1}{(z^{-1}-\zeta)^{n+1}} &= \frac{1}{z^{-1}-\zeta}\frac{1}{(z^{-1}-\zeta)^n}=\left(\sum_{j=0}^{\infty}\zeta^j\,z^{j+1}\right)\left(\sum_{j=0}^{\infty}p_n(\zeta)\,\zeta^j\,z^{j+n}\right)\\ &= \sum_{j=0}^{\infty}\sum_{k=0}^{\infty}p_n(k)\zeta^{j+k}\,z^{j+k+n+1}=\sum_{j=0}^{\infty}\left(\sum_{k=0}^{j}p_n(k)\right)\,\zeta^j\,z^{j+n+1}\\ &=: \sum_{j=0}^{\infty}p_{n+1}(j)\,\zeta^j\,z^{j+n+1},\end{aligned}$$

womit die Induktion auch schon erledigt ist. Mehr über die Polynome p_n findet sich übrigens in [19], siehe auch [39]. □

Übung 14.2 Zeigen Sie, daß die "diskrete Integration"

$$p_{n+1}(j)=\sum_{k=0}^{j}p_n(k),\qquad p_1\equiv 1,$$

tatsächlich *Polynome* $p_n\in\Pi_{n-1}$ definiert. ♢

Und wenn wir schon beim Beweisen sind, dann kümmern wir uns gleich noch um zwei vergleichsweise einfach zu beweisende Resultate, die aber trotzdem zur mathematischen Allgemeinbildung zählen.

Satz 14.25 (Bézout-Identität & Partialbruchzerlegung)

1. *Sind f, g Polynome ohne gemeinsame Nullstelle, dann gibt es Polynome p, q mit* $\deg p = \deg g - 1$ *und* $\deg q = \deg f - 1$*, die die* Bézout-Identität

$$fp + gq = 1 \tag{14.31}$$

erfüllen.

2. *Hat das Polynom q die Nullstellen ζ_j mit Vielfachheit μ_j, $j = 1, \ldots, n$, dann gibt es für jedes Polynom p Polynome p_j, $j = 1, \ldots, n$, so daß*

$$\frac{p}{q} = \sum_{j=1}^{n} \frac{p_j}{(\cdot - \zeta_j)^{\mu_j}}. \tag{14.32}$$

Beweis: Seien also f, g komprime Polynome[163] und seien $\zeta_1, \ldots, \zeta_n$, $n = \deg f$ die Nullstellen von f. Dann definieren wir das Polynom q vom Grad $\deg f - 1$ als Lösung des Interpolationsproblems

$$q(\zeta_j) = \frac{1}{g(\zeta_j)}, \qquad j = 1, \ldots, n,$$

was wohldefiniert, da ja $g(\zeta_j) \neq 0$ ist und was uns

$$\underbrace{f(\zeta_j)\, p(\zeta_j)}_{=0} + g(\zeta_j)\, q(\zeta_j) = g(\zeta_j)\, \frac{1}{g(\zeta_j)} = 1$$

liefert. Dasselbe Spiel mit den Nullstellen $\zeta'_1, \ldots, \zeta'_m$ von g definiert dann auch p mit analogen Eigenschaften. Nun ist aber $fp + gq - 1$ ein Polynom vom Grad $n + m - 1$, das an den $n + m$ Punkten $\zeta_1, \ldots, \zeta_n, \zeta'_1, \ldots, \zeta'_m$ verschwindet und muss deswegen das Nullpolynom sein - nichts anderes behauptet (14.31).

Die *Partialbruchzerlegung* (14.32) ist nun wieder eine Konsequenz aus (14.31): Kann man den Nenner q der rationalen Funktion p/q in komprime Faktoren $q = q_1\, q_2$ zerlegen, dann gibt es f_1, f_2, so daß $q_1\, f_1 + q_2\, f_2 = 1$ ist, und wir erhalten, daß

$$\frac{p\, f_2}{q_1} + \frac{p\, f_1}{q_2} = \frac{p\,(f_2\, q_2 + f_1\, q_1)}{q_1\, q_2} = \frac{p}{q}.$$

Die Zerlegung (14.32) folgt nun durch Iteration dieser Idee. □

Bemerkung 14.26 *Ganz unbemerkt darf Satz 14.25 nicht bleiben, denn diese Aussagen gelten nicht nur für Polynome, sondern für beliebige* euklidische Ringe*, siehe z.B. [10, 37]. Daher ist es vielleicht auch nicht so überraschend, daß die Aussagen generell auf dem euklidischen*

[163] Also Polynome ohne gemeinsame Nullstelle! Da an jeder gemeinsamen Nullstelle ζ von f und g ja $f(\zeta)p(\zeta) + g(\zeta)q(\zeta) = 0$ für alle p, q gilt, kann (14.31) ohnehin nur für teilerfremde bzw. koprime Polynome erreichbar sein. Für die klappt es aber auch immer.

Algorithmus beruhen. Die Partialbruchzerlegung ist auch für "normale" Brüche (also über $\mathbb{Z}$) ein Klassiker, da werden die Nullstellen zur Primfaktorenzerlegung und liefern Resultate der Form

$$\frac{1}{6} = \frac{1}{2} - \frac{1}{3}.$$

Beweis von Satz 14.21: Sei also[164] $a(z) = (z-\zeta_1)^{\mu_1} \cdots (z-\zeta_n)^{\mu_n}$. Mit einer Partialbruchzerlegung und mit Lemma 14.24[165] erhalten wir dann, daß

$$\begin{aligned} \frac{1}{a\left(z^{-1}\right)} &= \frac{1}{(z^{-1}-\zeta_1)^{\mu_1} \cdots (z^{-1}-\zeta_n)^{\mu_n}} = \frac{z^{\mu_1+\cdots+\mu_n}}{(1-\zeta_1 z)^{\mu_1} \cdots (1-\zeta_n z)^{\mu_n}} \\ &= \sum_{j=1}^{n} q_j(z)\, \frac{z^{\mu_j}}{(1-\zeta_j z)^{\mu_j}} = \sum_{j=1}^{n} q_j(z) \sum_{k=0}^{\infty} p_{\mu_j}(k)\, \zeta_j^k z^{k+\mu_j}, \end{aligned}$$

und das Verhalten der Koeffizienten für diese Reihe wird durch die betragsgrößte Nullstelle, sagen wir ζ_n, bestimmt: Ist $|\zeta_n| < 1$, dann konvergieren die Koeffizienten exponentiell gegen Null und die Differenzengleichung ist stabil, ist hingegen $|\zeta_n| > 1$, dann ist es nichts mit Stabilität. Im Fall $|\zeta_n| = 1$ kann man keine so ganz einfachen Aussagen treffen. □

Um es nochmals zusammenzufassen: Im Prinzip können wir unsere Differenzengleichung dadurch lösen, daß wir $g(z)/a\left(z^{-1}\right)$ bestimmen, beziehungsweise $b\left(z^{-1}\right)\ g(z)$ berechnen. Eine generelle Frage - die auch und vor allem in der digitalen Signalverarbeitung auftritt - ist demzufolge:

> Wie kann man Faltungen bzw. Korrelationen *effizient* berechnen?

Gut, was bedeutet hier effizient? Naja, rechnen wir doch einmal die Faltung

$$a * b = \sum_{k\in\mathbb{Z}} a_{j-k}\, b_k$$

zweier Signale der Länge N, also mit N von Null verschiedenen Einträgen, aus, dann haben wir im Ergebnis $2N$ von Null verschiedene Einträge und bei einer "typischen" Summation müssen wir normalerweise über N Einträge summieren, also wird uns eine Komplexität von $O\left(N^2\right)$ nicht erspart bleiben. Das klingt so überzeugend, daß man es fast für einen Beweis halten könnte, daß es gar nicht besser gehen kann.

Um uns vom Gegenteil zu überzeugen, erst einmal noch ein wenig Notation. Schränken wir das z in $a(z)$ auf den Einheitkreis oder *Torus*

$$\mathbb{T} = \{z \in \mathbb{C} \,:\, |z| = 1\} = \left\{e^{i\xi} \,:\, \xi \in [-\pi, \pi]\right\}$$

ein, dann ist

$$\widehat{a}(\xi) = a\left(e^{i\xi}\right) = \sum_{j\in\mathbb{Z}} a_j\, e^{ij\xi} = \sum_{j\in\mathbb{Z}} a_j \left(\cos j\xi + i \sin j\xi\right)$$

[164] Wir normieren hier der Einfachheit halber den Höchstkoeffizienten zu 1.

[165] Für irgendwas muss es ja gut sein.

ein trigonometrisches Polynom - zumindest solange a endlich ist, andernfalls[166] hätten wir es mit einer trigonometrischen Reihe zu tun. Nehmen wir jetzt an, daß $a = [a_0, \ldots, a_{N-1}]^T \in \mathbb{R}^N$ wäre, dann stehen wir vor der seltsamen Situation, daß wir einen Vektor in ein trigonometrisches Polynom transformieren und damit also die Struktur verändern[167]. Möchten wie wieder einen Vektor, dann könnten wir das trigonometrische Polynom ja einfach an N gleichverteilten Punkten im zugehörigen Intervall[168] abtasten und erhalten so den Vektor

$$\begin{aligned}\widehat{a} &= \left[\widehat{a}\left(e^{2i\pi k/N}\right) \;:\; k = 0, \ldots, N\right] \\ &= \begin{bmatrix} 1 & 1 & \ldots & 1 & 1 \\ 1 & e^{2i\pi/N} & \ldots & e^{2i\pi(N-2)/N} & e^{2i\pi(N-1)/N} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & e^{2i\pi(N-1)/N} & \ldots & e^{2i\pi(N-2)(N-1)/N} & e^{2i\pi(N-1)(N-1)/N} \end{bmatrix} a =: V_N a,\end{aligned}$$

den man als *diskrete Fouriertransformation* oder *DFT* von a bezeichnet. Die Matrix V_N hat übrigens deutlich mehr Struktur! Betrachten wir zum Beispiel nur das Element "unten rechts", dann ist

$$e^{2i\pi(N-1)(N-1)/N} = \underbrace{e^{2i\pi N}}_{=1}\,\underbrace{e^{-4i\pi}}_{=1}\,e^{2i\pi/N} = e^{2i\pi/N}$$

und die letzte Spalte entspricht einem "Countdown" von $N/N, (N-1)/N, \ldots, 1/N$ im Exponenten. Bezeichnen wir mit $\omega := \omega_N := e^{2\pi i/N}$ die *N-te komplexe Einheitswurzel* , dann hat V_N die Form

$$V_N = \left[\omega^{jk} \;:\; j,k = 0, \ldots, N-1\right], \qquad \omega^N = 1.$$

Übung 14.3 Zeigen Sie, daß V_N invertierbar ist, genauer, daß $V_N^H V_N = N\,I$ ist. Mit anderen Worten: $V_N/\sqrt{N}$ ist sogar unitär. ♢

Aber zurück zu unseren Differenzengleichungen bzw. zu den Faltungen von Vektoren, also zu $a * b$. Da aus der Faltungsstruktur für die Symbole sofort

$$(a * b)^\wedge(\xi) = (a * b)\left(e^{i\xi}\right) = \widehat{a}(\xi)\,\widehat{b}(\xi)$$

folgt, ist insbesondere

$$(a * b)^\wedge\left(\frac{2k\pi}{N}\right) = \widehat{a}\left(\frac{2k\pi}{N}\right)\widehat{b}\left(\frac{2k\pi}{N}\right), \qquad k = 0, \ldots, N-1$$

und somit gilt für die DFT einer Faltung

$$(a * b)^\wedge = \widehat{a} \odot \widehat{b}, \tag{14.33}$$

[166] Aber mit diesem Fall wollen wir uns nicht beschäftigen.

[167] Es gibt in der abstrakten harmonischen Analysis aber Konzepte wie das der dualen Gruppe, die dieses Verhalten durchaus erklären.

[168] Wegen der 2π-Periodizität trigonometrischer Polynome "verpasst" man auf kleineren Intervallen etwas, während auf größeren Intervallen die Periodizität zu Artefakten führen würde.

wobei $\odot$ die *komponentenweise Multiplikation*, also einen Spezialfall des *Hadamard-Produkts* zweier Matrizen, bezeichnen soll.

Was bringt uns das nun? Nun, ganz einfach: Die Faltung ist nun so schnell oder langsam wie die Berechnung der Transformation, denn um die Faltung $a*b$ zu berechnen, können wir zuerst a und b transformieren, dann komponentenweise multiplizieren[169] und das Ergebnis zurücktransformieren. Das macht dann drei Transformationen und eine komponentenweise Multiplikation, und wenn die Transformation schneller ist als $O\left(N^2\right)$, dann haben wir gewonnen. Aber Moment: Das Ergebnis

$$c = \left(\widehat{a} \odot \widehat{b}\right)^{\vee}$$

liegt doch wieder in $\mathbb{R}^N$, so daß es nicht die "richtige" Faltung sein kann, denn die hätte ja mehr von Null verschiedene Einträge. Ganz richtig, das ist der Preis, wir berechnen "nur" die zyklische Faltung

$$a *_N b = \left[\sum_{k=0}^{j} a_{j-k} b_k + \sum_{k=j+1}^{N-1} a_{N+j-k} b_k \ : \ j = 0, \ldots, N\right]$$

bei der negative Indizes an a modulo N "aufgelöst" werden. Also gut - dann nehmen wir doch einmal an, daß $a *_N b$ das ist, was wir berechnen wollen[170] und versuchen nun, die Berechnung ein klein wenig zu beschleunigen. Dazu sehen wir uns einmal einen Eintrag der DFT für *gerades* $N = 2M$ an:

$$\begin{aligned}
\widehat{a}_k &= \sum_{j=0}^{N-1} a_j \, \omega^{jk} = \sum_{j=0}^{M-1} a_{2j} \, \omega^{2jk} + \sum_{j=0}^{M-1} a_{2j+1} \, \omega^{(2j+1)k} \\
&= \sum_{j=0}^{M-1} a_{2j} \, \left(\omega^2\right)^{jk} + \omega^k \sum_{j=0}^{M-1} a_{2j+1} \, \left(\omega^2\right)^{jk}.
\end{aligned}$$

Und das war's eigentlich auch schon! Da ω eine $N = 2M$-te Einheitswurzel ist, ist ω^2 eine M-te Einheitswurzel und wir können die Berechnung der DFT der Ordnung N darauf zurückführen, daß wir

1. eine DFT der Ordnung M mit den geradzahlig indizierten Koeffizienten a_{2j} berechnen,
2. eine DFT der Ordnung M mit den ungeradzahlig indizierten Koeffizienten a_{2j+1} berechnen,
3. die Einträge der letzteren mit den Potenzen von ω "skalieren",
4. das Ergebnis addieren.

[169] Und das sind N Operationen, damit ist es billig!

[170] Wir würden hierbei nach einer periodischen Lösung der Differenzengleichung suchen - eine Annahme, die man immer machen kann aber nicht immer machen sollte.

Eines klingt noch ein wenig seltsam: Wir werden DFTs der Ordnung M a Indizes $0, \ldots, N-1$ aus, was natürlich größer wird als M. Aber da wir es mit Diskretisierungen periodischer FUnktionen zu tun haben, müssen wir diese Indizes lediglich modulo M betrachten und erhalten so unsere DFT der Länge N. Wenn a^0 und a^1 die beiden Teilvektoren $a_{2j+\epsilon}$ bezeichnen, dann ist das Berechnungsschema

$$
\begin{array}{|cccc|cccc|}
\hline
\widehat{a}^0_0 & \ldots & \widehat{a}^0_{M-1} & & \widehat{a}^0_0 & \ldots & \widehat{a}^0_{M-1} & \\
\hline
\multicolumn{8}{c}{+} \\
\hline
\widehat{a}^1_0 & \ldots & \widehat{a}^1_{M-1} & & \widehat{a}^1_0 & \ldots & \widehat{a}^1_{M-1} & \\
\hline
\multicolumn{8}{c}{\odot} \\
\hline
1 & \ldots & \omega^{M-1} & & \omega^M & \ldots & \omega^{N-1} & \\
\hline
\multicolumn{8}{c}{\downarrow} \\
\hline
\widehat{a}_0 & \ldots & \widehat{a}_{M-1} & & \widehat{a}_M & \ldots & \widehat{a}_{N-1} & \\
\hline
\end{array}
$$

Worin liegt aber nun der Gewinn dieser Vorgehensweise? Wenn wir mit $A(N)$ den Aufwand[171] dieser Berechnung bezeichnen, dann müssen wir zweimal transformieren, das sind also $2A(N/2)$ Operationen und zusätzlich noch N Mutiplikationen mit den Potenzen von ω zuzüglich $N-1$ Operationen zur Berechnung dieser Potenzen und schließlich noch N Additionen durchführen. Damit ergibt sich die rekursive Aufwandsgleichung

$$A(N) = 2A(N/2) + 2N - 1, \qquad A(1) = 0, \tag{14.34}$$

und das ist viel besser als $O\left(N^2\right)$, wie uns die folgende vereinfachte Version des *"Master Theorem"* der Komplexitätstheorie, siehe z.B. [43], zeigt. Dieses Resultat findet sich als Lemma A.1 im Anhang und sagt uns, daß sich der Aufwand für unser Berechnungsverfahren als

$$A(N) = O\left(N \log_2 N\right)$$

angeben lässt. Und das ist – nicht nur im Vergleich mit den vorher erwähnten $O\left(N^2\right)$ – schnell, so schnell, daß man bei dem Verfahren von der *schnellen Fouriertransformation* bzw. *Fast Fourier Transform* oder kurz *FFT* spricht.

Was aber hat das nun mit unserem Problem der Differenzengleichungen zu tun? Ganz einfach, wenn wir die Differenzengleichung $a \star u = g$ betrachten, dann ergibt sich ja

$$g(z) = (a \star u) = a\left(z^{-1}\right)\, u(z) \quad \Rightarrow \quad g\left(e^{i\xi}\right) = a\left(e^{-i\xi}\right)\, u\left(e^{i\xi}\right) \quad \Rightarrow \quad \widehat{g} = \widehat{\sigma_{-1}a} \odot \widehat{u}$$

und die *periodische* Lösung der Differenzengleichung ist die *inverse DFT*

$$u = \left(\widehat{g} \oslash \widehat{\sigma_{-1}a}\right)^{\vee}.$$

Beispiel 14.27 *Wir möchten die Lösung von $u''(t) = \sin t$ bestimmen, und zwar vermittels der Differenzengleichung*

$$u_{j+2} - 2u_{j+1} + u_j = \sin hj, \qquad j = 0, \ldots, \left\lfloor \frac{2\pi}{h} \right\rfloor.$$

[171] Bezüglich Rechenoperationen.

Damit ist nach Auffüllen auf Länge N

$$a = (1, 2, 1, 0, \ldots, 0) \qquad \textit{und} \qquad \sigma_{-1} a = (0, \ldots, 0, 1, 2, 1)\,,$$

wobei letzteres "natürlich" modulo N gerechnet ist. Bilden wir nun, mittels `fft(a)` *die DFT von a, dann sind die ersten beiden Einträge des Vektors Null - das entspricht genau den Anfangswerten, die wir hier in weiser Voraussicht auf Null setzen werden. Die eigentliche Berechnung des Restes ist dann der Einzeiler*

```
octave> u = ifft( fft( g(3:N) ) ./ fft( a(3:N) ) );
```

und ein Plot des Realteils von u, `plot( real(u) )`*, liefert dann auch eine richtig schöne Sinuskurve, der Imaginärteil hingegen enthält eine kleine Portion "numerischen Schmutz", siehe Abb. 14.8.*

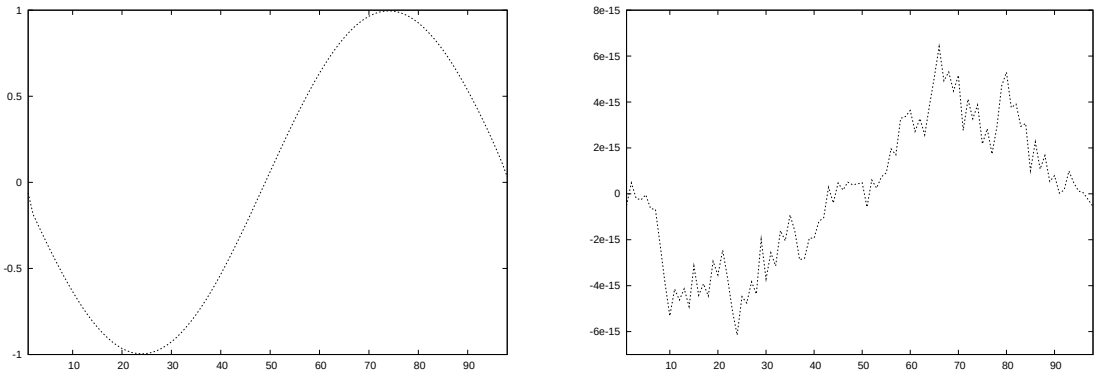


Abbildung 14.8: Das Ergebnis von Beispiel 14.27 für $N = 100$, wobei Realteil *(links)* und Imaginärteil *(rechts)* separat geplottet sind. Immerhin sieht der Realtein ziemlich nach $-\sin$ aus, während sich der Imaginärteil sich im Bereich normaler Rundungsfehler bewegt.

Bemerkung 14.28 *Will man bei komponentenweisen Divisionen durch die DFT auf Nummer sicher gehen, dann kann man Aufrufe der Form*

```
u = ifft( fft( g ) ./ ( fft( a ) + ( fft(a) == 0 ) ) );
```

verwenden, die an solchen Stellen einfach durch Eins dividiert. Und so häufig sind Stellen, an denen die DFT den Wert Null hat, ja ohnehin nicht.

> *Darum [. . .] bringt uns Euere Beweise oder des Aristoteles Gründe und Beweise, nicht aber Zitate und bloße Autoritäten . . .*
>
> G. Galilei, *Dialog über die hauptsächlichsten Weltsysteme, das ptolemäische und das kopernikanische*

A Anhang: Gut zu wissen

In diesem Anhang finden sich ein paar Dinge, die eigentlich aus anderen Vorlesungen bekannt sein sollten[172] und somit nicht direkt zum Stoff gehören. Es sind, so gesehen, Antworten auf Zwischenfragen während der Vorlesung.

A.1 Orthogonalisierung und Gram–Schmidt

Das *Gram–Schmidtsche Orthogonalisierungsverfahren* ist aus der linearen Algebra bekannt und verwandelt eine Basis eines Vektorraums in eine Orthonormalbasis. Seien also $a_1, \ldots, a_m \in \mathbb{R}^n$ eine *linear unabhängige* Familie von Vektoren[173], die einen m–dimensionalen Unterraum A des $\mathbb{R}^n$ aufspannen. Die Orthonormalisierung ist nun eine denkbar einfache Geschichte: Betrachten wir die Vektoren

$$\begin{aligned} x_j &= a_j - \frac{a_j^T a_m}{a_m^T a_m}\, a_m, \qquad j = 1, \ldots, m-1, \\ x_m &= \frac{1}{\sqrt{a_m^T a_m}} a_m, \end{aligned}$$

dann ist offensichtlich $x_m^T x_m = 1$ und für $j = 1, \ldots, m-1$ gilt

$$x_j^T x_m = \frac{1}{\sqrt{a_m^T a_m}} a_m \left(a_j^T a_m - \frac{a_j^T a_m}{a_m^T a_m} a_m^T a_m \right) = 0,$$

also

$$x_j^T x_m = \delta_{jm}, \qquad j = 1, \ldots, m. \tag{A.1}$$

[172] Konjunktiv! Garantie gibt's da ja leider keine.

[173] Daß wir uns im Reellen befinden ist keine wirklich Einschränkung, im $\mathbb{C}^n$ geht's genauso, im $\mathbb{Q}^n$ gibt es allerdings Schwierigkeiten mit der Ortho**normal**ität.

Anders gesagt: $x_1, \ldots, x_{m-1}$ bilden eine Basis des orthogonalen Komplements zu x_m und müssen eigentlich nur noch rekursiv in denselben Prozess gesteckt werden. Das kann man in folgendem Verfahren formalisieren[174]:

Für $k = m, m-1, \ldots, 1$ ersetze

$$\begin{aligned} a_j &\leftarrow a_j - \frac{a_j^T a_k}{a_k^T a_k}\, a_k, \qquad j = 1, \ldots, k-1, \\ a_k &\leftarrow \frac{1}{\sqrt{a_k^T a_k}}\, a_k, \end{aligned}$$

und was dabei übrigbleibt ist eine Orthonormalbasis $a_1, \ldots, a_m$ desselben Vektorraums A wie vorher.

Übung A.1 Implementieren Sie das Gram–Schmidt–Verfahren und testen Sie es an selbstgewählten Beispielen. ♢

Man kann Gram–Schmidt allerdings auch im Kontext der QR–Zerlegung betrachten, sozusagen als "QR rückwärts". Dazu identifizieren wir wieder den Vektorraum mit der Matrix $A = [a_1, \ldots, a_m] \in \mathbb{R}^{n\times m}$, was zu der faszinierenden "Identität" $A = A\mathbb{R}^m$ führt, und betrachten die QR–Zerlegung dieser Matrix

$$A = QR, \qquad Q = \left[\widetilde{Q} \,|\, Q'\right] \in \mathbb{R}^{n\times n}, \qquad R = \begin{bmatrix} \widetilde{R} \\ 0 \end{bmatrix} \in \mathbb{R}^{n\times m},$$

wobei $\widetilde{R}$ eine invertierbare Matrix sein muß, da $a_1, \ldots, a_m$ als *linear unabhängig* vorausgesetzt wurden. Mittels dieser Partitionierung ist dann

$$A = \left[\widetilde{Q} \,|\, Q'\right] \begin{bmatrix} \widetilde{R} \\ 0 \end{bmatrix} = \widetilde{Q}\,\widetilde{R} \qquad \text{bzw.} \qquad A\widetilde{R}^{-1} = \widetilde{Q}. \tag{A.2}$$

Damit sind die Spalten von $\widetilde{Q}$ aber eine Orthonormalbasis des Vektorraums $A = A\mathbb{R}^m = A\widetilde{R}^{-1}\mathbb{R}^m$ und da $\widetilde{R}^{-1}$ ja wieder eine obere Dreiecksmatrix ist, ist die k–te Spalte q_k von $\widetilde{Q}$ gerade eine Linearkombination von $a_1, \ldots, a_k$ – ganz genau wie im Gram–Schmidt–Verfahren. Und in der Tat ist das sogar genau dasselbe!

Übung A.2 Zeigen Sie, daß die Spalten von $\widetilde{Q}$ aus (A.2) genau die Ergebnisse des Gram–Schmidt–Verfahrens sind. ♢

A.2 Wie schnell kann man ...

Die Komplexitätstheorie beschäftigt sich mit der Frage, wie schnell man bestimmte Operationen ausführen kann. Oftmals wird dabei der Aufwand, wie beispielsweise bei der FFT in (14.34) rekursiv bestimmt. Solche Verfahren, bei denen **ein- und dieselbe** Funktion rekursiv für immer kleinere Datenhäppchen aufgerufen wird, bezeichnet man als *"Divide & Conquer"*–Verfahren. Die Aufwandsbestimmung erfolgt im folgenden Lemma.

[174] Um Indizes zu sparen verwenden wir Zuweisungen der Form $\leftarrow$, in `Octave` wären das ganz normale "=".

Lemma A.1 *Sei $N = b^n$ und F die Lösung der Rekursionsgleichung*

$$F(N) = aF(N/b) + cN + d, \qquad F(1) = 0,$$

dann ist

$$F(N) = \begin{cases} O(N), & a < b, \\ O\left(N \log_b N\right), & a = b, \\ O\left(N^{\log_b a}\right), & a > b. \end{cases}$$

Beweis: Wir rechnen einfach mal ein paar Schritte:

$$\begin{aligned} F(N) &= aF(N/b) + cN + d = a\left(aF\left(N/b^2\right) + cN/b + d\right) + cN + d \\ &= a^2 F\left(N/b^2\right) + \left(1 + \frac{a}{b}\right) cN + (1+a)\, d = a^n F(1) + cN \sum_{j=0}^{n-1} (a/b)^j + d \sum_{j=0}^{n-1} a^j. \end{aligned}$$

Fangen wir mit der ersten Summe an: Ist $a < b$ dann ist diese nach oben durch die von N unabhängige Konstante

$$\sum_{j=0}^{\infty} (a/b)^j = \frac{1}{1 - a/b} = \frac{b}{b-a}$$

beschränkt, ist $a = b$ so hat sie den Wert

$$\sum_{j=0}^{n-1} (a/b)^j = \sum_{j=0}^{n-1} 1^j = n = \log_b N$$

und im Falle $a > b$ ergibt sich

$$\frac{(a/b)^n}{a/b - 1} = \frac{b}{a-b} \frac{a^{\log_b N}}{b^{\log_b N}} = \frac{b}{N(a-b)}\, b^{\log_b a \, \log_b N} = \frac{b}{N(a-b)}\, N^{\log_b a}.$$

Mit demselben Trick bstimmt sich die zweite Summe noch als $n = \log_b N$ falls $a = 1$ ist und als

$$\frac{a^n}{a-1} = \frac{1}{a-1}\, b^{\log_b a \log_b N} = \frac{1}{a-1} N^{\log_b a}$$

andernfalls.

Im ersten Fall, $a < b$ ergibt sich daher eine Gesamtkomplexität von höchstens

$$\frac{cb}{b-a} N + \frac{d}{a-1} N^{\log_b a} = O(N), \qquad \log_b a < 0,$$

für $a = b$

$$cN \log_b N + d \left\{ \begin{array}{rl} \log_b N, & a = 1, \\ N, & a > 1, \end{array} \right\} = O\left(N \log_b N\right)$$

und schließlich

$$\left(\frac{b}{a-b} + \frac{1}{a-1}\right) N^{\log_b a} = O\left(N^{\log_b a}\right)$$

falls $b > a$. $\square$

Man könnte nun sagen, daß Lemma A.1 den Nachteil hat, nur für Werte von N zu gelten, die Potenzen von b sind. Das stimmt glücklicherweise nicht, denn ansonsten ist ja immer noch

$$b^{\lfloor \log_b N \rfloor} \leq N \leq b^{\lceil \log_b N \lceil}$$

und damit ist N höchstens einen Faktor b von der nächsten Potenz von b entfernt, was auch in den Abschtzungen höchstens einen Faktor der Form b, $b \log_b b = b$ oder $b^{\log_b a}$ ausmacht, was in allen drei Fällen eine moderate, von N unabhngige Konstante ist.

Es gibt noch eine schöne Anwendung von Lemma A.1, nämlich das schnelle Multiplizieren von Zahlen mit N Ziffern. Nach der Schulmethode braucht das N^2 Ziffernmultiplikation[175] und dann maximal $2N$ Addition und man ist versucht, das auch für die beste Komplexitt zu halten. Weit gefehlt, denn schreiben wir für $N = 2^n$

$$a = a_1 \times B^{N/2} + a_0, \qquad b = b_1 \times B^{N/2} + b_0,$$

dann ist

$$a\,b = a_1\,b_1 \times B^N + (a_1\,b_0 + a_0\,b_1) \times B^{N/2} + a_0\,b_0,$$

dann können wir den "mittleren" Term auch als

$$a_1\,b_0 + a_0\,b_1 = (a_1 + a_0)\,(b_1 + b_0) - a_1\,b_1 - a_0\,b_0$$

berechnen, wofür wir nur die **drei** Multiplikationen halber Länge sowie zwei (lineare) Additionen halber Länge und zwei Subtraktionen ganzer Länge brauchen. Das "Zusammensetzen" von ab kostet uns auch nicht mehr als $3N$ Operationen und wir haben die Aufwandsrekursion

$$F(N) = 3F(N/2) + 6N, \qquad F(1) = 1,$$

also, nach Lemma A.1, insgesamt $O\left(N^{\log_2 3}\right)$. Wer sich für dieses Verfahren, das *Karatsuba–Verfahren*, interessiert, der sei auf [10, 37] verwiesen. Es geht übrigens noch besser, nmlich $O(N \log N)$ mit der FFT[176] oder mit wesentlich mehr Aufwand in $O\left(N\ \log N\ \log\log N\right)$, was dann das Verfahren von Schönhage und Strassen ist, das ebenfalls auf der FFT basiert, aber sich passende Einheitswurzeln adjungiert.

[175] Das "kleine Einmaleins".

[176] Aber das ist erst einmal gemogelt, weil wir da komplexe Zahlen bräuchten!

Uns ist in alten mæren
wunders viel geseit
von Helden lobebæren
von grôzer arebeit

Das Nibelungenlied

B Literatur

[1] J. S. Bendat and A. G. Piersol. *Engineering Applications of Correlation and Spectral Analysis*. John Wiley & Sons, 1980.

[2] C. de Boor. *A practical guide to splines*. Springer–Verlag, New York, 1978.

[3] C. de Boor. *Splinefunktionen*. Lectures in Mathematics, ETH Zürich. Birkhäuser, 1990.

[4] R. Christensen. *Advanced Linear Modeling*. Springer Texts in Statistics. Springer, 2. edition, 2001.

[5] P. J. Davis. *Interpolation and Approximation*. Dover Books on Advanced Mathematics. Dover Publications, 1975.

[6] P. J. Davis and P. Rabinowitz. *Methods of numerical integration*. Academic Press, 1975.

[7] J. G. F. Francis. The QR transformation: A unitary analogue to the LR transformation. *Comp. J.*, **4** (1961), 265–271.

[8] J. G. F. Francis. The QR transformation: A unitary analogue to the LR transformation II. *Comp. J.*, **4** (1961), 332–334.

[9] F. R. Gantmacher. *Matrix Theory. Vol. I.* Chelsea Publishing Company, 1959. Reprinted by AMS, 2000.

[10] J. von zur Gathen and J. Gerhard. *Modern Computer Algebra*. Cambridge University Press, 1999.

[11] C. F. Gauss. Methodus nova integralium valores per approximationem inveniendi. *Commentationes societate regiae scientiarum Gottingensis recentiores*, **III** (1816).

[12] W. Gautschi. On the construction of Gaussia quadrature rules from modified moments. *Math. Comp.*, **24** (1970), 245–260.

[13] W. Gautschi. *Numerical Analysis. An Introduction*. Birkhäuser, 1997.

[14] I. M. Gelfand and S. V. Fomin. *Calculus of Variations*. Prentice–Hall, 1963. Dover reprint, 2000.

[15] W. Givens. Computation of plane unitary rotations transforming a matrix to triangular form. *SIAM J. Appl. Math.*, **6** (1958), 26–50.

[16] G. Golub and C. F. van Loan. *Matrix Computations*. The Johns Hopkins University Press, 1983.

[17] G. H. Golub. Numerical methods for solving linear least squares problems. *Numer. Math.*, **7** (1965), 206–216.

[18] G. H. Golub and J. A. Welsch. Calculation of Gauss quadrature rules. *Math. Comp.*, **23** (1969), 221–230.

[19] R. L. Graham, D. E. Knuth, and O. Patashnik. *Concrete Mathematics*. Addison–Wesley, 2nd edition, 1998.

[20] R. W. Hamming. *Digital Filters*. Prentice–Hall, 1989. Republished by Dover Publications, 1998.

[21] H. Heuser. *Lehrbuch der Analysis. Teil 2*. B. G. Teubner, 2. edition, 1983.

[22] H. Heuser. *Lehrbuch der Analysis, Teil I*. B. G. Teubner, 3. edition, 1984.

[23] N. J. Higham. *Accuracy and stability of numerical algorithms*. SIAM, 1996.

[24] A. S. Householder. Unitary triangularization of a nonsymmetric matrix. *J. Assoc. Comp. Mach.*, **5** (1958), 339–342.

[25] E. Isaacson and H. B. Keller. *Analysis of Numerical Methods*. John Wiley & Sons, 1966.

[26] C. G. J. Jacobi. Über ein leichtes Verfahren, die in der Theorie der Säkularstörungen vorkommenden Gleichungen numerisch aufzulösen. *Crelle's Journal*, **30** (1846), 51–94.

[27] K. D. Kammeyer and K. Kroschel. *Digitale Signalverarbeitung*. Teubner Studienbücher Elektrotechnik. B. G. Teubner, Stuttgart, 1998.

[28] W. Kutta. Beitrag zur näherungsweisen Integration totaler Differentialgleichungen. *Z. Math. Phys.*, **46** (1901), 435–453.

[29] S. Mallat. *A Wavelet Tour of Signal Processing*. Academic Press, 2. edition, 1999.

[30] M. Marcus and H. Minc. *A Survey of Matrix Theory and Matrix Inequalities*. Prindle, Weber & Schmidt, 1969. Paperback reprint, Dover Publications, 1992.

[31] H. M. Möller and H. J. Stetter. Multivariate polynomial equations with multiple zeros solved by matrix eigenproblems. *Numer. Math.*, **70** (1995), 311–329.

[32] P. Z. Peebles. *Probability, Random Variables and Random Signal Principles*. McGraw–Hill, 1980.

[33] A. Riese. *Rechenbuch / auff Linien und Ziphren / in allerley Handhierung / Geschäfften unnd Kauffmannschafft*. Franck. Bey. Chr. Egen. Erben, 1574. Facsimile: Verlag Th. Schäfer, Hannover, 1987.

[34] S. Rushdie. *Shame*. Jonathan Cape Ltd., 1983.

[35] H. Rutishauser. Solution of the eigenvalue problems with the LR transformation. *Nat. Bur. Stand. App. Math. Ser.*, **49** (1958), 47–81.

[36] T. Sauer. Numerische Mathematik I. Vorlesungsskript, Friedrich–Alexander–Universität Erlangen–Nürnberg, Justus–Liebig–Universität Gießen, 2000. `http://www.math.uni-giessen.de/tomas.sauer`.

[37] T. Sauer. Computeralgebra. Vorlesungsskript, Justus–Liebig–Universität Gießen, 2001. `http://www.math.uni-giessen.de/tomas.sauer`.

[38] T. Sauer. Optimierung. Vorlesungsskript, Justus–Liebig–Universität Gießen, 2002. `http://www.math.uni-giessen.de/tomas.sauer`.

[39] T. Sauer. Digitale Signalverarbeitung. Vorlesungsskript, Justus–Liebig–Universität Gießen, 2003. `http://www.math.uni-giessen.de/tomas.sauer`.

[40] T. Sauer. Splines in industrial applications. Vorlesungsskript, Zaragoza, Februar 2007, Justus–Liebig–Universität Gießen, 2007. `http://www.math.uni-giessen.de/tomas.sauer`.

[41] H. R. Schwarz. *Numerische Mathematik*. B. G. Teubner, Stuttgart, 1988.

[42] P. Spellucci. *Numerische Verfahren der nichtlinearen Optimierung*. Internationale Schriftenreihe zu Numerischen Mathematik. Birkhäuser, 1993.

[43] A. Steger. *Diskrete Strukturen 1. Kombinatorik – Graphentheorie – Algebra*. Springer, 2001.

[44] G. W. Stewart. *Introduction to Matrix Computations*. Academic Press, 1973.

[45] J. Stoer. *Einführung in die Numerische Mathematik I*. Heidelberger Taschenbücher. Springer Verlag, 4 edition, 1983.

[46] J. Stoer and R. Bulirsch. *Einführung in die Numerische Mathematik II*. Heidelberger Taschenbücher. Springer Verlag, 2 edition, 1978.

[47] J. H. Wilkinson. The evaluation of zeros of ill–conditioned polynomials I. *Numer. Math.*, **1** (1959), 150–166.

[48] J. H. Wilkinson. The evaluation of zeros of ill–conditioned polynomials II. *Numer. Math.*, **1** (1959), 167–180.

[49] J. H. Wilkinson. *The algebraic eigenvalue problem*. Clarendon Press, Oxford, 1962.

[50] J. H. Wilkinson. The perfidious polynomial. In G. H. Golub, editor, *Studies in Numerical Analysis*, volume 24 of *MAA Studies in Mathematics*, pages 1–28. The Mathematical Association of America, 1984.

[51] Yuan Xu. *Common zeros of polynomials in several variables and higher dimensional quadrature*. Pittman Research Monographs. Longman Scientific and Technical, 1994.

Index C

Ähnlichkeit, 36
  orthogonale, 43
  unitäre, 37

Algorithmus
  euklidischer, 147
Anfangswert, 139
Anfangswertproblem
  Diskretisierung, 139
Auslöschung, 21

B–Spline, 138
Basis
  Orthonormal-, 44

CAVALIERI, 94
COTES, R., 93

DFT, 148
  inverse, 150
Differentialoperator
  Diskretisierung, 139
Differentialquotient, 138
Differenzengleichung
  stabile, 144
Diskretisierung
  eines Differentialoperators, 139
  konsistente, 141
  Schrittweite, 139
Diskriminante, 58
Drei–Term–Rekursionsformel, 105

Eigenvektor, *34*
  dominanter, 40
Eigenwert, *34*
Einheitskreis, 147
Euklidischer Algorithmus, 81

Faltung, 143
FFT, 150, 153
Folge
  doppeltunendliche, 142
Form
  schwache, 137
Fouriertransformation
  diskrete, 148
Funktion
  erzeugende, 143
  rationale, 85

GAUSS, C. F., 98
Gitterweite, 139
Glattheit, 31
Gleichung
  Differenzen-, 139
    konsistente, 141
Gleichungssystem
  überbestimmtes, 6
  lineares, 3
  unterbestimmtes, 3, *3*

Identität
  Bézout, 146
Interpolation, 31
  Hermite-, 94
  polynomiale, 7, 91
Invarianz
  orthogonale, 15

Jacobi–Gewicht, 110

Keplersche Faßregel, 94
Knotenpolynom, 93
Konditionszahl, 11, 14
Korrelation, 143
Kreise
  Gerschgorin-, 74

Lösung
Least squares-, 7, 14, 26

Matrix
ähnliche, 36
bandierte, 137
diagonalisierbare, 37, 41
Diskretisierungs-, 137
Drehungs-, *siehe* Matrix, Jacobi- 28
Frobenius, 80
Frobenius-, 35, 106
Hessenberg-, 40, 49
Householder-, 21, 24
Jacobi-, 28, 68
kommutierende, 38
Momenten-, 100
Multiplikations-, 106
nichtderogatorische, 37
normale, 38
orthogonale, *11*, 16, 28
Phasen-, 44
Rechtsdreiecks-, 23, 37, 45
symmetrische, 16, 68
Toeplitz-, 140
unitäre, *11*
Milne–Regel, 94
Momente, 89
Moore–Penrose–Inverse, 13

NEWTON, I., 93
Norm
∞, 74
Energie-, 31
Zeilensummen-, 74
Normalengleichung, *9*, 25
Normalform
Jordansche, 36, 44
Nullstellen
geschachtelte, 84

Operator
Differenzen-, 138
Optimierung, 5
orthogonale Matrix, *siehe* Matrix, orthogonale 11
Orthogonalisierung, *152*

Pol, 85
Polynom
charakteristisches, 34
Jacobi
Rodriguez–Formel, 110
Jacobi-, 110
Knoten-, 93
Legendre-, 111
Leitterm, 100
monisches, 84
orthogonales, *99*
aus Momentenmatrix, 101
Bestimmung, 99
Nullstellen, 104
Rekursionsformel, 105
Rekursionskoeffizienten, 107
Wilkinson-, 76
Potenzmethode, *siehe* Vektoriteration 40
Problem
Anfangswert-, 134
Randwert-, 134
Produkt
Hadamard-, 149
Pseudoinverse, *13*, 14
Pulcherrima, 94

Quadratpositivität, 99
Quadraturformel, *88*
Approximationsgüte, 89
Exaktheitsgrad, 88–90, 92
Gauß-, 92, *98*
Eindeutigkeit, 105
Existenz, 105
Fehler, 101
Gewichte, 108
Knoten, 107
Gewichte, 88
interpolatorische, 91, 92, *92*
Knoten, 88
Momente, 89
Newton–Cotes-, 93
zusammengesetzte, 96

zusammengesetzte, 95
Fehler, 96
Fehlerabschätzung, 96
Newton–Cotes-, 96
Simpson–Regel, 98

Rang
maximaler, 3
Raum
Ansatz-, 136
Regularisierung, 32
Residuenvektor, 10
Ring
euklidischer, 146

Signal
diskretes, 142
endliches, 143
kausales, 143
SIMPSON, TH., 94
Simpson–Regel, 94
Fehlerabschätzung, 94
zusammengesetzte, 98
Singuläre–Werte–Zerlegung, *siehe* SVD 11
Smoothing Spline, 31
Spektralradius, 11, 74
Spektralverschiebung, 58
komplexe, 64
Konvergenzordnung, 64
Spektrum, *34*, 74
Spiegelung, 16
Spline, 31
Stabilität, 144
Sturmsche Kette, *80*
Bestimmung, 81
Vorzeichenwechsel, 81
SVD, *11*, 13
Symbol, 143
System
inverses, 3
lineares, 143

Teilraum
invarianter, 36
Teufelswerk, 15
Theorem
Master, 150
Tomographie
optische, 4
Torus, 147
Transformation
Fourier
schnelle, 150
Givens-, *28*, 49
Householder-, 16, 22, 49
orthogonale, *15*
RQ-, 47, *48*
Trapezregel, 94
Tschebyscheffknoten, 90

unitäre Matrix, *siehe* Matrix, unitäre 11

Vektor
Householder-, 16, 21
Vektoriteration, 40
für Vektorräume, 44
Nachteile, 42
Verfahren
adaptives, 96
Bairstow-, 80
Cholesky-, 10
Gauß–Seidel, 138
Gram–Schmidt, 152
Gram–Schmidt-, 99
Jacobi-, 68
Konvergenz, 69
Karatsuba, 155
LR-, 45
Maehly-, 78
Orthogonalisierungs-, 99
Potenzmethode, *siehe* Vektoriteration 40
Power method, *siehe* Vektoriteration 40
QR-, *43*
Doppelschritt, 65
Konvergenz, 44
Konvergenzgeschwindigkeit, 47, 58
Spektralverschiebung, *siehe* Spektralverschiebung 58

Split, 54
Vektoriteration, *siehe* Vektoriteration 40

Weddle–Regel, 94
Werte
singuläre, 12
Wurzel
Einheits-, 148

Zerlegung
Partialbruch-, 146
QR-, *23*, 43, 44, 100
Aufwand, 25
Eindeutigkeit, 46
Komplexität, 49
mit Pivotsuche, 27
Stabilität, 27
Schur-, 37
relle, 39
Singuläre–Werte-, *siehe* SVD 11