Numerik in der Schule

Vorlesung, zuerst gehalten im Sommersemester 08

Tomas Sauer & Rudolf Sträßer

Version 1.0
Letzte Änderung: 13.7.2011

Statt einer Leerseite . . . | 0

Welcher aber . . . durch die Geometria sein Ding beweist und die gründliche Wahrheit anzeigt, dem soll alle Welt glauben. Denn da ist man gefangen.

Albrecht Dürer

A peculiarity of the higher arithmetic is the great difficulty which has often been experienced in proving simple general theorems which had been suggested quite naturally by numerical evidence.

H. Davenport, *The Higher Arithmetic*, 1952

I always found it shameful that mere technologists should have arrogated to themselves the right to be called that, scientists, men of knowledge.

S. Rushdie, *Grimus*

You may call it 'nonsense' if you like, [. . .] but I've heard nonsense, compared with which that would be as sensible as a dictionary.

L. Carroll, *Through the looking glass*

Tomas Sauer
Lehrstuhl für Numerische Mathematik
Justus-Liebig-Universität Gießen
Heinrich-Buff-Ring 44
D-35392 Gießen

0 Inhaltsverzeichnis

Wenn nicht mehr Zahlen und Figuren
Sind Schlüssel aller Kreaturen ...

F. von Hardenberg – Novalis

1 Zahlen und Zahlendarstellungen

In der Mathematik machen wir es uns zumeist sehr einfach und bezeichnen Zahlenmengen einfach mit abstracten Zeichen wie $\mathbb{Q}$, $\mathbb{R}$ oder $\mathbb{C}$. Und am besten definieren wir diese Zahlenbereiche dann auch noch axiomatisch, was für mathematische Zwecke zwar der beste Weg sein mag, aber "normalen Menschen" nur sehr schwer zu vermitteln ist.

1.1 Rationale und reelle Zahlen

Gehen wir einmal davon aus, daß die "Urintention" des Zählens zusammen mit der kulturellen Errungenschaft der Null[1] uns die natürlichen Zahlen $\mathbb{N} = \{0, 1, 2, \ldots\}$ liefert und daß weiterhin das "Kreditwesen" ganz natürlich auch negative Zahlen benötigt, dann fängt die Mathmatik da an, wo man **Zahlenkörper** verwendet, also Mengen, die unter den vier Grundrechenarten Addition, Subtraktion, Multiplikation und Division abgeschlossen sind - insbesondere hat jede von Null verschiedene Zahl x einen **Reziprokwert** $x^{-1} = 1/x$.

Der einfachste Körper sind sicherlich die rationalen Zahlen $\mathbb{Q}$, also die Menge aller Brüche:

$$\mathbb{Q} = \left\{ \frac{p}{q} \ : \ p, q \in \mathbb{Z},\, q \neq 0 \right\}.$$

Brüche sind von Natur aus mehrdeutig, es gibt viele Möglichkeiten ein- und dieselbe Zahl darzustellen, beispielsweise

$$\frac{1}{2} = \frac{-1}{-2} = \frac{4}{8} = \frac{-999}{-1998} = \cdots$$

Die Gleichheit zweier Brüche kann man recht einfach nachprüfen:

$$\frac{p}{q} = \frac{r}{s} \qquad \Leftrightarrow \qquad ps = rq.$$

[1] Die Babylonier und Inder kannten Sie, die Römer nicht, dem Mittelalter war sie suspekt und sie konnte erst langsam über Spanien aus dem Arabischen nach Europa einwandern.

Das ist eine recht banal erscheinende algebraische Umformung, die aber einen interessanten "Nebeneffekt" hat: Die Gleichheit auf der rechten Seite ist eine Gleichheit von *ganzen Zahlen*, die erst einmal nichts mehr mit $\mathbb{Q}$ zu tun hat. Und das ist praktisch relevant: Können wir erst mal in $\mathbb{Z}$ rechnen, dann können wir in $\mathbb{Q}$ rechnen.

Um die Mehrdeutigkeiten der **Darstellung**[2] von Brüchen umgehen zu können, verwendet man eine **Normalform**, bei der der Bruch *gekürzt* dargestellt wird - das heißt, es gibt keine Zahl, die Zähler und Nenner teilt - und bei der man das Vorzeichen in den Nenner schreibt. Also:

$$\mathbb{Q} \ni x = \frac{p}{q}, \qquad p \in \mathbb{Z},\ q \in \mathbb{N},\ \mathrm{ggT}\,(p,q) = 1. \tag{1.1}$$

Die Schwäche der Brüche, von Pythagoras als "Harmonien" bezeichnet[3], ist, daß nicht alle geometrischen Größen als Brüche dargestellt werden können, beispielsweise $\sqrt{2}$, die Länge der Diagonale im Einheitsquadrat oder π, das Verhältnis zwischen Kreisumfang und -durchmesser. Und weil's so schön ist, ...

Satz 1.1 *Die Zahl $\sqrt{2}$ ist kein Bruch.*

Beweis: Nehmen wir an, es gäbe "normalformige" p, q gemäß (1.1), so daß $\frac{p}{q} = \sqrt{2}$, also $p^2 = 2q^2$. Da $2q^2$ eine gerade Zahl ist, muß p^2 und damit auch p eine gerade Zahl sein, also $p = 2\widetilde{p}$ für ein $\widetilde{p} \in \mathbb{Z}$. Setzen wir das für p ein, so ist

$$2q^2 = p^2 = \left(2\widetilde{p}\right)^2 = 4\widetilde{p}^2 \qquad \Leftrightarrow \qquad q^2 = 2\widetilde{p}^2,$$

und damit ist auch q gerade und wir hätten bei der Darstellung unseres Bruches vergessen mit 2 zu kürzen - ein Widerspruch zur Annahme, daß wir eine Normalform gemäß (1.1) gewählt haben. □

Die reellen Zahlen einfach und verständlich zu beschreiben ist schon nicht mehr so einfach, immerhin braucht man da druchaus axiomatische Schwergewicht wie die *Dedekindschen Schnitte*, um sie wirklich sauber zu definieren, siehe [6]. Der mathematisch einfachste Weg ist vielleicht der als **Vervollständigung** von $\mathbb{Q}$, aber selbst dazu braucht man so Begriffe wie Cauchyfolgen und Äquivalenzklassen bezüglich Konvergenz. Alternativ verwendet man im Schulbetrieb ja gerne unendliche Dezimalbrüche, aber die sind auch nicht besser und vor allem - sie kleben an einer speziellen Darstellung. Also sehen wir uns diese besser an.

[2] Um das noch einmal ganz klar zu sagen: Der Bruch als solcher ist eindeutig, es gibt nur verschiedene Möglichkeiten, ihn zu schreiben.

[3] Das weltanschauliche Credo der Pythagoräer war, daß Harmonie ein Zahlenverhältnis mit **kleinem** Zähler und Nenner ist, und daß alles Harmonie ist, zumindest mehr oder weniger.

1.2 Stellenwertsysteme

In der "Mengenlehre" der Grunschulausbildung[4] um 1970 wurden Zahlen als "Häuschen" interpretiert: 123 stand für "ein *Hunderter-Häuschen*, zwei *Zehner-Häuschen* und drei Einzelne". Und das ist ja auch das Wesen unseres Stellenwertsystems, nämlich die Darstellung[5]

$$z_n \dots z_0 = z_0 + z_1 \times 10 + z_2 \times 10^2 + \dots + z_n \times 10^n = \sum_{j=0}^{n} z_j \, 10^j.$$

Fügen wir noch ein Vorzeichen hinzu, dann können wir also jede ganze Zahl durch dieses Vorzeichen und die **Ziffern** $z_0, \dots, z_n$ **darstellen**, wobei jede der Ziffern zwischen 0 und 9 liegt. Warum wir die Basis 10 verwenden, das lässt sich sehr einfach an den Fingern beider Hände abzählen, ansonsten ist diese Wahl aber absolut willkürlich: Wir könnten ohne weiteres die 10 durch das Symbol B wie "Basis" ersetzen, unsere Darstellungsformel als

$$(\pm z_n \dots z_0)_B := \pm \sum_{j=0}^{n} z_j \, B^j, \qquad z_j \in \{0, \dots, B-1\}, \tag{1.2}$$

schreiben und dann B durch eine beliebige andere Zahl ersetzen - das führt dann zur B**-adischen Zahlendarstellung**. Und so eine Zahlendarstellung bekommt man sogar ziemlich einfach! Wir müssen lediglich[6] (1.2) ein ganz klein wenig in

$$(z_n \dots z_0)_B = z_0 + B \sum_{j=1}^{n} z_j \, B^{j-1} = z_0 + B \, (z_n \dots z_1)_B$$

umschreiben, um zu sehen, was hier passiert, nämlich **Division mit Rest**. Wir teilen also einfach unsere Zahl durch B, der **Divisionsrest** wird unsere letzte Ziffer z_0, der Faktor an B wird weiter zerlegt.

Ganz besonders einfach wird dies für die Basis $B = 2$, also das **Binärsystem**, denn da müssen wir in jedem Schritt nur prüfen, ob wir es mit einer geraden oder einer ungeraden Zahl zu tun haben.

---

[4] Um genau zu sein: der Grundschulausbildung in Bayern!

[5] Daß wir die Zahlen eigentlich von rechts nach links schreiben, liegt daran, daß unsere Altvorderen das Zahlensystem von den Arabern übernommen haben, auch wenn die Ziffern**symbole** an sich indischen Ursprungs sind - ziemlich globalisiert also. Wer sich für sowas interessiert, dem sei ein Blick in [8] empfohlen.

[6] Das Vorzeichen können wir uns schenken - das ist ja von der Basis völlig unabhängig.

Beispiel 1.2 *Bestimmen wir doch einmal die Binärdastellung von* 29*:*

$$\begin{array}{ccccccccccc} 29 & \to & 14 & \to & 7 & \to & 3 & \to & 1 & \to & 0 \\ & \searrow & & \searrow & & \searrow & & \searrow & & \searrow & \\ & & 1 & & 0 & & 1 & & 1 & & 1 \\ \hline & & z_0 & & z_2 & & z_3 & & z_4 & & z_5 \end{array}$$

also $(29)_{10} = (11101)_2$*. Zu Basis* $B = 5$ *ist das auch nicht anders, wie wir uns an den Fingern einer Hand abzählen:*

$$\begin{array}{ccccccc} 29 & \to & 5 & \to & 1 & \to & 0 \\ & \searrow & & \searrow & & \searrow & \\ & & 4 & & 0 & & 1 \end{array}$$

also $(29)_{10} = (104)_5$.

Auch wenn die Binärdarstellung bereits von Leibniz entwickelt und propagiert wurde und bereits die "alten Ägypter" binär multiplizierten, siehe [4], hat sich das Binärsystem doch erst mit dem Entstehen der Digitalcomputer durchsetzen können, und zwar aus dem einfachen Grund, daß die Ziffern 0 und 1 technisch durch *Strom aus* und *Strom an* realisiert werden können. Daher werden auf Digitalrechnern Zahlen normalerweise im Binärformat, also zur Basis $B = 2$ dargestellt, auch wenn der dafür zuständige Standard **IEEE 754** explizit die Verwendung der Basis $B = 10$ einschließt!

1.3 Dezimalbrüche und B–zimalbrüche

So, und nun kommen wir auch schon zu den Dezimalbrüchen und deren B-adischen Brüdern, indem wir in (1.2) auch negative Potenzen von B zulassen:

$$(z_n \cdots z_0.z_{-1} \ldots z_{-m})_B := \sum_{j=-m}^{n} z_j \, B^j, \tag{1.3}$$

die **Nachkommastellen** gehören hier eben zu negativen Potenzen von B. Da

$$(z_n \cdots z_0.z_{-1} \ldots z_{-m})_B = B^{-m} \sum_{j=-m}^{n} z_j \, B^{j+m} = B^{-m} \, (z_n \cdots z_0 z_{-1} \ldots z_{-m})_B \,,$$

entspricht diese Zahl einem Bruch:

$$(z_n \cdots z_0.z_{-1} \ldots z_{-m})_B = \frac{(z_n \cdots z_0 z_{-1} \ldots z_{-m})_B}{B^m}, \tag{1.4}$$

ergo

Jeder **endliche** B-adische Bruch ist eine rationale Zahl.

Aber wie sieht es mit der Umkehrung aus, ist denn auch jede rationale Zahl ein endlicher Bruch? Die Antwort ist natürlich "nein", wie uns schon das einfache Beispiel der Dezimalbruchentwicklung

$$\frac{1}{3} = 0.3333\ldots = 0.\overline{3}$$

zeigt. Und schon befinden wir uns in der wunderbaren Welt der überstrichenen Nachkommastellen, der unendlichen periodischen Dezimalbrüche. Aber immer der Reihe nach: Ein **unendlicher Dezimalbruch** ist ein Ausdruck der Form

$$(z_n \ldots z_0.z_{-1}z_{-2}\ldots)_B = \sum_{j=-\infty}^{n} z_j B^j = (z_n \ldots z_0)_B + \sum_{j=1}^{\infty} \frac{z_{-j}}{B^j}.$$

Die Reihe der Nachkommastellen konvergiert[7], das ist nicht das Problem, aber wie das immer so bei Reihen ist, ist der **Wert** dieser Reihe ihr **Grenzwert** und Reihen mit demselben Grenzwert stellen dieselbe Zahl dar, das berühmte $0.\overline{9} = 1.0$.

Definieren wir reelle Zahlen als unendliche Dezimalbrüche, so müssen wir Zahlen, deren Reihenentwicklung denselben Grenzwert haben, auch miteinander identifizieren.

Aber wie sieht es nun mit der Endlichkeit von rationalen Zahlen aus? Klar, Endlichkeit können wir auch mit Reihen und Grenzwerten nicht erreichen, aber es gibt doch sowas ähliches, nämlich Periodizität! In der Tat gilt die folgende Aussage.

Satz 1.3 *Die B-adische Entwicklung eines Bruchs p/q ist entweder endlich oder periodisch.*

Beweis: Der Beweis ist gar nicht so schwer: Nach Abzug der "Vorkommastellen" können wir annehmen, daß $p < q$ ist und für die erste Nachkommastelle teilen wir nun $p \times B$ durch q und erhalten wieder einen Divisor und einen Divisiosrest. Der Divisor ist unsere erste Nachkommastelle, z_{-1}, der Rest eine Zahl p_1 mit $0 \leq p_1 < q$. Ist $p_1 = 0$, dann ist die B-adische Entwicklung endlich, andernfalls machen wir mit p_1 dasselbe Spiel wie mit p, erhalten also ein Schema der Form

$$\begin{array}{ccccccccc} p & \to & p_1 & \to & p_2 & \to & \ldots & \to & p_q \\ & \searrow & & \searrow & & \searrow & & \searrow & \\ & & z_{-1} & & z_{-2} & & \ldots & & z_{-q} \end{array}$$

[7] Sie ist monoton steigend und durch 1 beschränkt!

Nun haben wir q Zahlen $p_1, \ldots, p_q$, die alle zwischen 1 und $q-1$ liegen, weswegen[8] mindesten zwei von den beiden Zahlen gleich sein müssen - und genau ab da fängt die Geschichte an, sich zu wiederholen. □

Aber jetzt zum wirklich interessanten Phänomen:

> Ob eine rationale Zahl eine endliche oder periodische Darstellung hat, das hängt vom Verhältnis dieses Bruchs zur Basis B ab!

Das einfachste Beispiel ist $\frac{1}{3} = (0.1)_3 = \left(0.\overline{3}\right)_{10}$, aber es gibt noch ein numerisch wesentlich relevanteres, das wir uns auch als Illustration des obigen Beweises genauer ansehen wollen.

Beispiel 1.4 *Wir berechnen einmal die Binärdarstellung von $\frac{1}{5}$, also $p = 1$ und $q = 5$:*

p		p_1						
1	$\to$	2	$\to$	4	$\to$	3	$\to$	1
	$\searrow$		$\searrow$		$\searrow$		$\searrow$	
		0		0		1		1
		z_{-1}		z_{-2}		z_{-3}		z_{-4}

also

$$\frac{1}{5} = \left(0.\overline{0011}\right)_2 \qquad \textit{und damit} \qquad \frac{1}{10} = \left(0.0\overline{0011}\right)_2 . \tag{1.5}$$

Aber eine andere Darstellung der Brüche und auch der rellen Zahlen haben wir noch, und zwar eine, die schon von den Griechen verwendet wurde. Dazu sehen wir uns eine reelle Zahl[9] an. Diese Zahl, nennen wir sie x, sei der Einfachheit halber positiv und liegt also irgendwo zwischen 0 und ∞. Nun holen wir uns ihren **Ganzzahlanteil** $a_0 = \lfloor x \rfloor$, so daß wir die Beziehung

$$x = a_0 + x_1, \qquad 0 \le x_1 < 1$$

haben. Ist $x_1 = 0$, dann ist $x \in \mathbb{N}$, was ja nun wirklich nichts schlimmes ist. Andernfalls liegt die Zahl $1/x_1$ irgendwo zwischen 1 und ∞ und kann als

$$\frac{1}{x_1} = a_1 + x_2, \qquad a_1 = \left\lfloor \frac{1}{x_1} \right\rfloor , \quad 0 \le x_2 < 1$$

geschrieben werden. Und das Spiel treiben wir vermittels der Iteration

$$a_k = \left\lfloor \frac{1}{x_k} \right\rfloor , \qquad x_{k+1} = \frac{1}{x_k} - a_k, \qquad k \in \mathbb{N}, \tag{1.6}$$

[8] Ein Kombinatoriker oder einer, der ein solcher werden will, würde nun vom "Schubfachprinzip" sprechen und daraus einen mindestens zweisemestrigen Zauber machen.

[9] Immer noch ohne zu wissen, was das nun wirklich ist ...

immer weiter, brechen höchstens ab, wenn irgendwann $x_k = 0$ werden sollte. Aus den Zahlen $a_0, \ldots, a_k$ können wir den zugehörigen **Kettenbruch**

$$[a_0; a_1, \ldots, a_k] = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{\ddots + \cfrac{1}{a_{k-1} + \cfrac{1}{a_k}}}}}$$

bilden und der ist genau diejenige Zahl x für die unser Algorithmus $a_0, \ldots, a_k$ bestimmt und dann mit $x_{k+1} = 0$ abbricht.

Kettenbruchentwicklungen waren ein mathematischer Klassiker, der unter anderem Bernoulli und Gauß beschäftigte, höchstwahrscheinlich den alten Griechen bekannt war und deren Vorstellung von reellen Zahlen geprägt hat. Kettenbrüche sind dafür verantwortlich, daß unsere Oktave in der Musik gerade 12 Halbtöne umfasst, Huygens verwendete sie für die Berechnung der Zahnräder seines Planetenmodells und sie zeigen, daß die als **Goldener Schnitt** bekannte Zahl $\psi = \frac{1+\sqrt{5}}{2}$ die irrationalste[10] Zahl mit der denkbar einfachsten Kettenbruchentwicklung $\psi = [1; 1, 1, \ldots]$ ist. Allerdings: Auch wenn die **Darstellung** extrem effizient, ja sogar optimal ist, wenn man mit Kettenbrüchen *rechnen* muß, dann sind sie ein Alptraum. Was ist beispielsweise $[1; 1, 1] + [1; 2, 3]$?

1.4 Ein Exkurs: Zeno und der Unterschied zwischen Zahl und Darstellung

Ein klassisches Paradoxon, das uns sehr schön den Unterschied zwischen Zahl und Darstellung und die Konsequenzen des fahrlässigen Umgangs mit der Unendlichkeit in Kovergenzform zeigt, ist das berühmte Paradoxon von Zeno[11] mit Achilles und der Schildkröte.

Beispiel 1.5 *Achilles, der schnellste Mensch seiner Zeit (siehe z.B. [16, 23. Gesang, Vers 555]) wird von einer Schildkröte zum Wettlauf über 100m herausgefordert. Achill erreicht 10m/s*[12]*, die Schildkröte 1m/s. Sie erhält einen Vorsprung von 10m. Zeno behauptet nun, daß Achilles die Schildkröte nie einholt.*

[10] Im Sinne von am langsamsten durch rationale Zahlen approximierbar.

[11] Zeno von Elea, 5. Jhdt. vor Christus, Verfasser von Paradoxien, die nachzuweisen suchten, daß es Bewegung eigentlich nicht gibt.

[12] Er läuft also die 100m, ohne Spikes, Steroide und Nandrolon in beachtlichen 10 Sekunden. Für's Doping waren seinerzeit die Götter zuständig, siehe [16, G. 23, V. 768–776].

Erinnern wir uns erst einmal an Zenos Argument: Achilles erreicht den Punkt, an dem die Schildkröte gestartet ist, nach einer Sekunde, nur ist diese inzwischen ein Stückchen weiter, nämlich 0.1m. Für diese Strecke braucht Achilles nur noch eine Zehntelsekunde, aber da ist die Schildkröte eben auch schon wieder ein Stückchen weiter und somit erreicht Achilles die Schildkröte nie.

Natürlich ist da was faul, denn wie man schon in der Mittelstufe lernt, ist der Zeitpunkt t (in Sekunden) zu dem Achilles die Schildkröte einholt die Lösung der Gleichung[13]

$$10\,t = 10 + 1\,t \qquad \Longrightarrow \qquad t = \frac{10}{9}.$$

Und alles, was Zeno zeigt ist, daß er mit seiner Methode diese Zahl beliebig gut annähern kann, aber eben nie erreicht.
Mathematisch gesprochen: Wenn man eine Folge von Zahlen, in unserem Fall von Zeitintervallen, hat, dann kann deren Summe einen Grenzwert haben oder auch nicht. Hat sie einen Grenzwert, dann ist das genau der Wert, also Zeitpunkt, an dem wir interessiert sind. Im Falle von Beispiel 1.5 sind das die Zeitintervalle

$$1\,s, \quad \frac{1}{10}\,s, \quad \frac{1}{100}\,s, \ldots$$

und die Summe ist (in Sekunden)

$$1 + \frac{1}{10} + \frac{1}{100} + \cdots = \sum_{j=0}^{\infty} 10^{-j} = 1.\overline{1} = 1\frac{1}{9}.$$

Das zeigt uns auch schon, wo es bei Zeno klemmt und wie banal die Lösung des Paradoxons eigentlich ist: Der Punkt, an dem Achilles die Schildkröte einholt, ist der **Grenzwert** der Reihe und eben keine **Partialsumme** derselben, ein kleiner, feiner aber eben furchtbar relevanter Unterschied.

1.5 Zahlendarstellung am Computer

Das war ja bisher alles ganz nett und gut, nur eben für die Realität und die Praxis nicht wirklich brauchbar. Schließlich verfügt ein Computer und/oder Taschenrechner nur über endlich viele Bits und Bytes und kann daher auch nur *endlich viele* Zahlen *endlicher Länge* speichern. Der erste Ansatz wären daher sicherlich die Verwendung von Brüchen in einer sogenannten **Langzahldarstellung**: Von der Zahl werden alle Ziffern in einer geeigneten Darstellung gespeichert, so daß eine kleine Zahl eben wenig Speicher

[13] Wir Mathematiker dürfen die Einheiten weglassen und außerdem sind wir ja nicht mehr und noch nicht wieder in der Schule.

benötigt, eine große Zahl hingegen viel. Die Grundrechenarten lassen sich dann auch recht einfach realisieren[14], es gibt aber eine ganze Menge von Einwänden:

- Je größer die an einer Rechenoperationen beteiligten Zahlen sind, desto länger dauert diese.
- Das Ergebnis von Rechenoperationen mit zwei großen Zahlen liefert normalerweise eine noch größere Zahl, man denke nur an die Multiplikation zweier Zahlen gleicher Länge, das Ergebnis ist dann schon doppelt so lang.
- Irgendwann gibt es entweder Probleme, das Ergebnis zu speichern oder ewig lange Rechenoperationen oder am besten gleich beides.

Brüche sind nun konzeptionell wieder sehr einfach, wir schreiben sie einfach als Paare von ganzen Zahlen, Zähler und Nenner eben. Und die abers gehen weiter:

- Die Nenner bei der Addition und Multiplikation von Brüchen wachsen ziemlich schnell ziemlich massiv, man hat es also sehr bald mit großen ganzen Zahlen zu tun.
- Das kann man manchmal vermeiden oder zumindest die Auswirkungen minimieren, indem man die Brüche nach jeder Rechenoperation durchkürzt. Das tut allerdings in dem Sinne weh, daß der Rechenaufwand ziemlich drastisch steigt.

Das symbolische Rechnen mit Brüchen ist in Computeralgebrasystemen wie `Maple`, `MuPAD`, `CoCoA` oder `Mathematica` integriert, wo man auch sehr schnell herausfinden kann, warum es nicht funktioniert.

Eine andere, numerische Philosophie besteht darin, jeder Zahl von vornherein nur eine fest Größe zuzuweisen, was zum Konzept der Gleitkommazahl[15]führt, bei der man eine ganzzahlige **Mantisse** M mit einer festen Stellenzahl, sagen wir m durch eine Multiplikation mit einer geeigneten **Basispotenz** einer Kommaverschiebung unterziehen.

Zur Erinnerung: Die Multiplikation einer B-adischen Zahl mit der Basispotenz B^k entspricht einer Verschiebung des Bezimalpunkts/-kommas um k Stellen nach rechts, und entsprechend um $-k$ Stellen nach links, falls $k < 0$ ist.

[14]Einfach zumindest, wenn man das auf naive, dann aber auch sehr langsame Art und Weise macht - zeitkritisch sind hierbei die "Punktoperationen" Multiplikation und Division, da muss man einiges an mathematischem Aufwand betreiben, damit das wirklich flott wird, siehe [2, 15].

[15]Hier kommt die Standardfußnote zur Terminologie: Im Deutschen gleitet das Dezimalkomma zwischen den Ziffern hin und her während im Englischen ein Dezimalpunkt fröhlich vor sich hinfloatet. Die beiden korrekten Termini wären also der deutsche Begriff "Gleitkomma" und die wörtliche Übersetzung "Fließpunkt" des englischen Terminus Technicus. Nichtsdestotrotz findet man im Sprachgebrauch eigentlich jede beliebige Kombination von Gleit-/Fließ- und -komma/-punkt.

Das führt dann zur Darstellung

$$\pm.\, d_1 \cdots d_m \;\times\; B^e := \pm \left(\sum_{j=1}^{m} d_j B^{-j} \right) B^e, \qquad d_j \in \{0, \ldots, B-1\},\ e \in E, \quad (1.7)$$

wobei der **Exponentenbereich** E eine normalerweise um die Null symmetrische *endliche* Teilmenge von $\mathbb{Z}$ ist. Schematisch kann man so eine Zahl auch als

$\pm$	$d_1 \cdots d_m$	$e_1 \cdots e_k$
↑	↑	↑
Vz	Mantisse	Exponent

schreiben, wobei es irrelevant ist, ob man die Mantisse nun als ganze Zahl oder als einen "reinen Bezimalbruch" schreibt. Ganz Aufmerksame haben vielleicht schon gemerkt, daß es zwei Möglichkeiten gibt, die Null zu schreiben, nämlich als $+0$ und als -0 und mit ein bisschen Pech werden diese beiden Zahlen in einem Programm auch schon mal als *unterschiedliche* Zahlen angesehen!
Zwei wichtige Punkte sind bei der Gleitkommadarstellung zu beachten:

1. Die Gleitkommadarstellung ist **der** Standard für technisch-wissenschaftliche bzw. numerische Rechnungen und in allen Taschenrechnern sowie in allen aktuellen Computerprozessoren integriert. Numerische Rechnung in diesen Zahlenformaten wird schnell[16] und auf definierte Art und Weise ausgeführt.

2. Die Konsequenzen aus der Verwendung dieser Arithmetik, die "normalerweise" bei allen computergestützten[17] Berechnungen unvermeidbar sind, beeinflussen **alle** Rechnungen mit elektronischer Unterstützung - und das hat, wie wir bald sehen werden, sehr weitreichende Auswirkungen.

Es ist für das Verständnis sehr wichtig, numerische Rechnungen am PC, am besten mit einem Programm wie `Octave` oder `Matlab`, durch- und vorzuführen, da Taschenrechner bei der Ergebnisausgabe gerne mogeln und beispielsweise Werte, die knapp neben einer Ganzzahl liegen, auf diese Ganzzahl "runden".

Bemerkung 1.6 *(IEEE 754) Die in heutigen PC–Prozessoren verwendete Fließpunktarithmetik entspricht dem* IEEE[18] 754*–Standard. Die dabei verwendete Basis ist* 2 *und die arithmetischen Datentypen sind als*

Typ	*Mantisse*	*Exponent*	*Roundoff*	*Größenordnung*
`float`	*23+1*	*8*	$2^{-24} = 5.96 \times 10^{-8}$	$10^{\pm 38}$
`double`	*52+1*	*11*	$2^{-53} = 1.11 \times 10^{-16}$	$10^{\pm 308}$

[16] Und zwar in vielen Fällen sogar schneller als "einfache" Ganzzahlrechnung!
[17] Und das schließt den Taschenrechner mit ein!
[18] IEEE = Institute of Electrical and Electronical Engineers

festgelegt. Das eigentlich interessante am Standard ist aber die Festlegung des Rundungsverhaltens und die Existenz und Weiterverarbeitung von `NaNs`[19] *und* `Infs`*.*
Der Standard IEEE 854 läßt übrigens als Basis die Werte 2 *und* 10 *zu! Letzteres heisst aber nur, daß Prozessoren die Basis* 10 *verwenden* dürfen*, nicht aber, daß sie es können müssten.*

[19] `Not a Number`, Ergebnisse von unzulässigen Operationen wie Wurzeln aus negativen Zahlen. Die Existenz dieser "Zahlen" ermöglicht es einem Computerprogramm, auch dann weiterzuarbeiten, wenn eine unzulässige Operation aufgetreten ist – es wäre ja niemandem damit gedient, wenn der Prozessor plötzlich die Arbeit einstellen würde. Allerdings sollte man dann auch diese Werte abprüfen und geeignet reagieren.

Was nicht passt wird passend gemacht

Filmtitel (1997)

2 Rundung und Fehler

Der wesentliche Vorteil der Gleitkommadarstellung besteht in ihrem ‘Festkostenprinzip”:

> Alle Gleitkommazahlen haben, unabhängig von ihrer Größe, denselben Speicherbedarf und alle Rechnungen mit Gleitkommazahlen dauern gleich lang.

Wir werden aber gleich sehen, daß wir für diesen Vorteil auch bezahlen müssen, nämlich durch **Rundung** von Zwischen- und Endergebnissen.

2.1 Rundung als Näherungsprinzip

Hat man die **Mantissenlänge** m erst einmal festgelegt, dann gibt es offensichtlich nur endlich viele Gleitkommazahlen; bedenkt man, daß die Menge $\mathbb{R}$ der reellen Zahlen ziemlich unendlich ist, dann ist es eigentlich klar, daß man sich mit Näherungslösungen behelfen muss.

Sei also $x \in \mathbb{R}$ eine Zahl, die wir als Gleitkommazahl darstellen, genauer gesagt, durch eine Gleikommazahl annähern können. Dieses x liegt irgendwo zwischen zwei Basispotenzen[20], $B^{k-1} \leq x < B^k$ und dieses k gibt uns auch schon unseren Exponenten e, denn nun ist[21]

$$x = .x_1x_2 \ldots \times B^k,$$

mit dem potentiellen “kleinen” Nachteil, daß wir es hier erst einmal mit unendlich vielen Stellen zu tun haben könnten, also mehr als wir in die m Stellen $d_1, \ldots, d_m$ unserer Mantisse unterbringen knnten. Aber der Trick ist sehr einfach, wir runden nämlich so, wie man es eigentlich schon in der Schule gelernt hat:

1. Wir setzen zuerst einmal $d_j = x_j$, $j = 1, \ldots, m$.

[20] Achtung: Das strikte “$<$” der rechten Ungleichung sorgt dafür, daß k *eindeutig* bestimmt ist.

[21] Wir lassen die Basis B jetzt einfach fest und erachten diese für gottgegeben und festgelegt.

2. Wir überprüfen die Stelle x_{m+1} und "runden ab", wenn $x_{k+1} < B/2$ ist, und runden auf, wenn $x_{k+1} \geq B/2$ ist. Aufrunden bedeutet hierbei, daß wir $d_1, \ldots, d_m$ so wählen, daß
$$.d_1 \ldots d_m = .x_1 \ldots x_m + B^{-m}$$
ist.

So ganz banal ist die Methode aber nicht, ein paar Kleinigkeiten müssen wir schon beachten, für Details siehe [13]:

- Das Verfahren funktioniert nur[22] für *geradzahlige* Basen B, denn da wird eben gerade die Hälfte auf- und die andere Hälfte abgerundet. Das ist natürlich nicht so wild, da die gebräuchlichen Basen ohnehin 2 und 10 sind.

  Bei ungeradzahligen Basen hingegen gibt es immer eine "mittlere" Ziffer, die man weder auf- noch abrunden kann, weil sonst die Rundung asymmetrisch wäre, so einen systematsischen Fehler einführen würde und daher im Prinzip nicht besser als reines Abschneiden wäre. Deswegen verschiebt man dann die Entscheidung auf die nächste Nachkommastelle, wenn diese wieder die "Mittelziffer" ist, dann wieder auf die nächste, und so weiter. Das sorgt aber nun wieder dafür, daß man nicht vorhersagen kann, wieviele nachfolgende Stellen der Zahl man für das Runden berücksichtigen muss – ein unmöglicher Zustand.

- Es gibt einen Spezialfall, nämlich (in Dezimaldarstellung)
$$0.\underbrace{9\ldots9}_{m}5,$$
  wo auf $1.0\ldots0$ aufgerundet wird, was dann wieder in $.10\ldots0$ umzuwandeln ist - wobei man den Exponenten e natürlich anpassen muss.

Der große Vorteil dieser Methode ist, daß wir sagen können, wie genau eine beliebige relle Zahl x durch eine Gleitkommazahl $\widehat{x}$ angenähert werden kann, und zwar mit einer *relativen* Genauigkeit von
$$\frac{|x - \widehat{x}|}{|x|} \leq \frac{1}{2} B^{1-m} =: u. \tag{2.1}$$
Die Zahl u bezeichnet man als **Rundungsfehlereinheit**. Sie ist **der** Maßstab für die Genauigkeit der verwendeten Artihmetik.

Definition 2.1 *Bei der Untersuchung von Fehlern zwischen einem **Sollwert** x und einem **Istwert** $\widehat{x}$ unterscheidet man zwischen zwei Konzepten:*

[22] Und das sieht man im Beweis, wo man die Annahme "B gerade" an einer Stelle wirklich braucht!

- *Beim **absoluten Fehler** betrachtet man einfach die Abweichung*

$$e_a := |x - \widehat{x}| \,.$$

Sind x und $\widehat{x}$ einheitenbehaftete *Größen, also beispielweise in Metern gegeben, dann ist auch der absolute Fehler eine Größe in derselben Einheit.*

- *Beim **relativen Fehler***

$$e_r := \frac{|x - \widehat{x}|}{|x|}$$

hingegen betrachtet man den Fehler im Verähltnis zum Sollwert[23]*, was eine* einheitenlose *Größe liefert. Fehlerabschätzungen für relative Fehler kann man auf zwei äquivalente Weisen beschreiben:*

$$\frac{|x - \widehat{x}|}{|x|} \leq e \qquad \Leftrightarrow \qquad \widehat{x} = (1 + \delta)x, \quad |\delta| \leq e.$$

Welches der beiden Fehlerkonzepte sinnvoller ist, ist normalerweise eine Frage der Anwendung: Beim CNC–Fräsen sind beispielsweise absolute Fertigungstoleranzen wichtig[24]*, wenn es uns, wie hier, hingegen um die Rechen**genauigkeit** geht, dann ist der relative Fehler angemessener. Deswegen werden wir in dieser Vorlesung eigentlich nur relative Fehler betrachten.*

Die Formel (2.1) für die Rundungsfehlereinheit hat eine unmittelbare Konsequenz:

> Die *Genauigkeit* eine Gleitkommaarithmetik hängt ausschließlich von der Mantissenlänge ab, der "Exponentenbereich" sagt uns lediglich, wie groß und klein darstellbare Zahlen werden dürfen.

Was wir aufgeben müssen, ist die Vorstellung von *absoluter* Genauigkeit, und das selbst für Zahlen, die wir ganz exakt eingeben können. Das klassische Beispiel ist `0.1` in einer Arithmetik mit $B = 2$: Wie wir in (1.5) hat $\frac{1}{10}$ eine *unendliche* Dualbruchentwicklung und kann daher nicht exakt dargestellt werden - hier haben wir *immer* einen Fehler! Und es ist entscheidend, hier immer den relativen Fehler zu betrachten, nicht aber den absoluten Fehler:

> Der **absolute Fehler** hängt ja immer auch vom Exponenten ab und kann daher bei der Rundung nahezu beliebig groß werden, während der **relative Fehler** unabhängig von der Größe der Zahl selbst ist.

[23] Beziehungsweise dessen Absolutbetrag.

[24] Es ist in Wirklichkeit sogar komplizierter: Oftmals gibt es sogar unterschiedliche Schranken für positive und negative Abweichungen, einen *inneren* und einen *äußeren* Fehler.

2.2 Rundungsfehler bei Rechnung

Soweit haben wir uns nur mit der *Darstellung* von Zahlen beschäftigt, aber natürlich wollen wir auch mit diesen Zahlen rechnen, die müssen zumindest einmal addiert, subtrahiert, miteinander multipliziert und durcheinander dividiert werden.

Fangen wir[25] also mit Multiplikation bzw. Division an, dann sehen wir, daß

$$(.x_1 \dots x_m \times B^e) \times \left(.y_1 \dots y_m \times B^f\right) = (.x_1 \dots x_m \times .y_1 \dots y_m) \times B^{e+f}$$

bzw.

$$(.x_1 \dots x_m \times B^e) \div \left(.y_1 \dots y_m \times B^f\right) = (.x_1 \dots x_m \div .y_1 \dots y_m) \times B^{e-f}$$

gelten muss – die Exponenten bereiten uns keine Schwierigkeiten, wohingegen Produkt bzw. Quotient der Mantissen normalerweise deutlich mehr als m Stellen haben werden. Genauer ist beim Produkt von $2m$ Stellen auszugehen, die Division gönnt sich auch gerne einmal unendlich viele[26]. Das ist aber nicht so schlimm, denn wir müssen das Ergebnis ja ohnehin wieder auf m Stellen runden, wozu wir nur die $m+1$–te Stelle benötigen und daher relativ bald mit dem Rechnen aufhören können.

Etwas interessanter sind Addition und Subtraktion! Wie wir die bei *gleichem Exponenten* durchführen, das ist klar, denn dann brauchen wir ja nur noch die Mantissen zu addieren bzw. zu subtrahieren. Was aber tun bei unterschiedlichen Exponenten? Naja, wir erinnern uns einfach daran, daß

$$.x_1 \dots x_m \times B^e = .0x_1 \dots x_m \times B^{e+1} = .\underbrace{0 \dots 0}_{k} x_1 \dots x_m \times B^{e+k}$$

ist, so daß wir also die Zahl mit dem kleineren Exponenten solange nach rechts "schieben" müssen, bis die Exponenten "passen", dann werden die Mantissen addiert bzw. subtrahiert und schließlich das Ergebnis geeignet normalisiert.

Diese Form des Rechnens ist an sich gar nicht einmal so schlecht! Hat man nämlich intern im Rechenwerk des Computers, dem sogenannten *Akkumulator* **Akkumulator**, $m+1$–stellige Mantissen zur Verfügung, dann kann man in der Tat so genau rechnen, wie man runden kann, erhält also Näherungen, die wieder (2.1) erfüllen. Das ist praktisch sehr nützlich und theoretisch auch beweisbar, siehe [7, 13]; die "Extrastelle" in der Mantisse bezeichnet man übrigens treffenderweise als **Guard Digit**, also als "Wächterziffer".

[25] Getreu dem guten alten Schulmotto *Punkt vor Strich.*

[26] Zum Beispiel bei $0.1 \times 2^1 \div .101 \times 2^4$ in binärer Rechnung – wem das nicht sofort klar, für den ist es eine nette Übung, sich zu überlegen, warum dem so ist.

Unter Verwendung einer Guard Digit erhält man für jede der vier Grundrechenarten $\cdot = +, -, \times, \div$ ein numerisches Ergebnis $\widehat{x} = a \odot b$, das

$$\widehat{x} = (1 + \delta)\, x, \qquad |\delta| \leq u, \tag{2.3}$$

mit der exakten Lösung $x = a \cdot b$ erfüllt.

Prima: (2.3) sagt uns, daß der *relative Fehler* jeder Grundrechenart durch die **Rundungsfehlereinheit** u beschränkt ist, wir können also genauso gut rechnen wie runden. Und mehr Rechengenauigkeit würde uns ja auch gar nichts helfen, da wir ja keine höhere Genauigkeit der Operanden garantieren können. Wie können das auch ein wenig anders formulieren:

Standardmodell der Fließpunktarithmetik: Bei jeder Rechenoperation wird zuerst **exakt** gerechnet, also auf so viele Stellen genau wie nötig, und anschließend das Ergebnis auf Mantissenlänge gerundet.

Trotzdem reicht auch (2.3) nicht aus, um reichlich desaströse Ergebnisse zu vermeiden.

Beispiel 2.2 (Der Fluch der Subtraktion) *Sehen wir uns mal ein ganz unschuldiges Beispiel zur Subtraktion mit der gewohnten Basis $B = 10$ an, und zwar*

$$0.10\ldots0 \times 10^1 - 0.9\ldots9 \times 10^0.$$

Gemäß unserer Regel müssen wir also die zweite Zahl um eine Stelle nach rechts schieben und erhalten so

$$0.10\ldots0 \times 10^1 - 0.09\ldots9 \times 10^1 = 0.\underbrace{0\ldots0}_{m}1 \times 10^1 = 0.10\ldots0 \times 10^{-m},$$

ganz genau so, wie man es erwartet!

Zu diesem Beispiel gibt es allerdings einiges zu sagen:

- Die **Schiebeoperation**, die wir im letzten Schritt durchgeführt haben, dienen dazu, daß die erste Stelle in der Mantisse, also die erste Nachkommastelle, von Null verschieden ist! Dies bezeichnet man als **Normalisierung** der Zahl. So heißt denn eine Gleitkommazahl nach Art von (1.7) **normalisiert**, wenn $d_1 \neq 0$ und **denormalisiert**, wenn $d_1 = 0$. Denormalisierte Zahlen sind normalerweise unerwünscht, weil sie Rechengenauigkeit "herschenken": Anstatt vorne eine total überflüssige Null aufzustellen, könnte man die Stelle sehr viel besser am Ende nutzen, um die Genauigkeit der dargestellten Zahl zu erhöhen. Deswegen:

So lange er es kann, normalisiert ein Computer/Taschenrechner alle Ergebnisse von Rechnungen.

- Das Ergebnis von Beispiel 2.2 ist absolut in Ordnung, solange die beiden Zahlen als **exakt** angenommen werden. Das ändert sich allerdings dramatisch, wenn man annimmt, daß sie Ergebnis eines Rundungsprozesses sind, denn dann steht $0.10\ldots0\times 10^1$ für eine Zahl aus dem Intervall[27]

$$[0.\underbrace{9\ldots9}_{m}5, 1.\underbrace{0\ldots0}_{m}5)$$

und entsprechend $0.9\ldots9$ für eine Zahl aus dem Intervall

$$[0.\underbrace{9\ldots9}_{m-1}85, 0.\underbrace{9\ldots9}_{m-1}95),$$

das wirkliche Endergebnis liegt also in dem Intervall

$$(0, 0.2)\times 10^{-m},$$

dessen Mittelpunkt uns die Berechnung geliefert hat.

- Der relative Fehler bei dieser Subtraktion von **gerundeten** Daten kann also ganz schnell 100% betragen! Man kann es auch anderes sehen: All die wunderschönen Nullen, mit denen wir das Ergebnis bei der Normalisierung von rechts aufgefüllt haben, sind bei ungenauen Daten durch nichts gerechtfertigt und könnten ganz genauso gut jede andere Form haben. Dieses Phänomen ist so böse und wohlbekannt, daß es sogar einen eigenen Namen hat: Es ist als **Auslöschung** bekannt.

Es hat einen Grund, warum wir in Beispiel 2.2 gerade diese Form der Auslöschung betrachtet haben, denn schlimmer geht's nicht. Das numerisch schlimmste, was man anstellen kann, ist tatsächlich die Subtraktionen zweier näherungsweise gleicher Zahlen, eine Operation, die man nach Kräften immer vermeiden sollte!

Bemerkung 2.3 *Um das nochmals klarzustellen: Die Rechenoperation in Beispiel 2.2 also solche war sogar **exakt**, das Problem mit der Ungenauigkeit rührte daher, daß man bei zwei Operanden einer Rechenoperation nie sagen kann, ob diese genau so gewollt und eingegeben wurden, oder ob sie durch Rundung entstanden sind.*

Es gibt aber noch einen zweiten Auslöschungseffekt bei Addition, nämlich den, daß Operanden einfach "unter den Tisch fallen".

[27] Hier sind wir sogar großzügig, das wirkliche Intervall ist noch größer.

Beispiel 2.4 (Zu viel summiert . . .) *Wir wollen das* arithmetische Mittel *von N Zahlen $x_1, \ldots, x_N$ berechnen, also*

$$\frac{1}{N} \sum_{j=1}^{N} x_j.$$

Das einfachste Verfahren hierfür ist "aufsummieren und hinterher durch N dividieren". Nun nehmen wir an, wir hätten für $B = 10$ eine Mantissenlänge von $m = 1$ zur Verfügung und möchten den Mittelwert einer Folge berechnen, die aus 20 Variablen mit dem Wert 0.1 besteht. Und die ersten 9 Schritte unserer Summation laufen auch wie erwartet ab, die Zwischensummen haben die Werte

$$.1 \times 10^0 \quad \to \quad .2 \times 10^0 \quad \to \quad \cdots \quad .9 \times 10^0.$$

Im nächsten Schritt passiert eine Umnormalisierung[28]*:*

$$.9 \times 10^0 + .1 \times 10^0 = 1.0 \times 10^0 \quad \to \quad 0.1 \times 10^1$$

und ab dem elften Schritt wird gerundet:

$$.1 \times 10^1 + .1 \times 10^0 = .1 \times 10^1 + .01 \times 10^1 = .11 \times 10^1 \quad \to \quad .1 \times 10^1,$$

so daß sich die Summe nicht mehr ändert. Unser berechneter Mittelwert $\overline{x}$ erfüllt also

$$\overline{x} = \begin{cases} 0.1, & N \leq 10, \\ \frac{1}{N}, & N > 10. \end{cases}$$

Das obige Beispiel zeigt eine weitere "unfreundliche" Eigenschaft der Rundungsfehler: Sie müssen nicht langsam und kontinuierlich auftreten, sondern können sprunghaft erscheinen.

Übung 2.1 Erklären Sie, warum eine Reihe $\sum a_j$ mit positiven a_j immer **numerisch** konvergiert, wenn a_j, $j \to \infty$, eine Nullfolge ist. ♢

Beispiel 2.5 *Rundungsfehler durch Aufsummieren und daraus resultierenden "Stellenverlust" können fatale Folgen haben, wie in [7] beschrieben: Im ersten Golfkrieg (1990) sicherten die US–Streitkräfte ihre Militärbasen durch* Patriot*–Raketenabwehrsysteme gegen irakische Angriffe mit* Skud*–Raketen, was auch fast immer erfolgreich war, mit einer Ausnahme, als eine US–Basis in Bahrein von einer Rakete getroffen wurde. Der Grund für das Versagen des Patriot–Systems war ein Zähler, eine Fließpunktzahl, die jede Sekunde hochgezählt wurde. Wie in Beispiel 2.4 gingen dabei im Laufe der Zeit mehr und mehr Stellen verloren und die relative Genauigkeit der Zeitmessung wurde immer schlechter, was sich natürlich auch in der Genauigkeit der Geschwindigkeitsmessung ankommender Flugkörper und bei der Steuerung ausgehender Raketen niederschlägt. Und das System in Bahrein war schlichtweg zu lange gelaufen. . .*

[28] Zur Erinnerung: Wir haben alle Stellen der Mantisse als **Nachkommastellen** festgelegt!

2.3 Fehlerfortpflanzung

Weil man es gar nicht oft genug sagen kann: Jede einzelne individuelle Rechenoperation ist nach (2.3) sehr genau und damit wäre auch das Ergebnis in Ordnung, wenn nur **die Eingangsdaten exakt** wären. Was sie aber halt nun mal nicht sind.

Ein sehr einfacher Weg zu nicht exakten Operanden in einer der Grundrechenarten besteht darin, Ergebnisse einer früheren Rechnung zu verwenden, die ja normalerweise bereits mit mehr oder weniger Rundung belastet sind. Und das ist ein typisches Szenario bei computergestützter Rechnung: Wegen einmaliger Durchführung einer Grundrechenart wird man das Gerät ja wohl kaum anschalten[29]. Und plötzlich wird es wichtig, darauf zu achten, **wie** man etwas macht.

Beispiel 2.6 *Wir wollen für $m = 3$ und $B = 10$, den Wert $x^2 - y^2$ berechnen, wobei $x = .334$ und $y = .333$ sein sollen*[30] *– "irrelevante" Exponenten der Form B^0 lassen wir hier einfach weg. Diese Berechnung können wir auf zwei Arten durchführen.*

$(x \times x) - (y \times y)$: *Hier ist*[31] *$x \times x = .112$ und $y \times y = .111$ und damit bekommen wir $.100 \times 10^{-2}$ als Ergebnis.*

$(x - y) \times (x - y)$: *Es ist $x + y = .667$ und $x - y = .100 \times 10^{-2}$, also ergibt sich jetzt $.667 \times 10^{-3}$.*

Welches der beiden Ergebnisse ist aber nun richtig oder zumindest richtiger? Wenn man genau hinsieht, zeigt sich, daß wir im zweiten Fall immer exakt *gerechnet haben, also muß das auch das richtige Ergebnis geliefert haben. Das heißt aber, wir machen im ersten Fall einen relativen Fehler von*[32]

$$\frac{.100 \times 10^{-2} - .667 \times 10^{-3}}{.667 \times 10^{-3}} = \frac{.333 \times 10^{-3}}{.667 \times 10^{-3}} \sim .5,$$

also von der Größenordnung $50\hat{u}$ (zur Erinnerung: $\hat{u} = 2\frac{1}{2}B^{-M} = B^{1-M} = 10^{-2}$)!

Was uns dieses Beispiel zeigt, ist, daß es für das Problem "Berechne $x^2 - y^2$" zwei *mathematisch* äquivalente numerische Verfahren gibt, nämlich

[29] Es sei denn, um sich per Internet mit der Grundrechenarten–Homepage bei Wikipedia zu verbinden, von wo aus ein Link zu einer Seite führt, bei der man sich das Ergebnis gebührenpflichtig zusammen mit einem Klingelton auf das Mobiltelefon senden lassen kann, von wo aus man dieses in ein Video konvertiert, per Bluetooth auf den Rechner berträgt, um dann dort festzustellen, daß der Media–Player den Codec noch nicht unterstützt, weswegen dringend ein Betriebssyste–Upgrade nötig ist. Wie konnte die Menschheit nur je ohne Computer auskommen? Und vor allem ohne Multimedia und die zugehörige Kompetenz, ein wesentlicher Soft-Skill jedes kontemporären Hochschulstudiums.

[30] Dies verwundert nun nicht mehr – schließlich wissen wir ja schon, daß die "interessanten" Effekte gerne im Zusammenspiel mit Auslöschung auftreten.

[31] Exakt gerechnet und dann gerundet, also genau wie in (2.3).

[32] Exakt gerechnet ...

1. Berechne $a = x \times x$ sowie $b = y \times y$ und dann $a - b$

2. Berechne $a = x + y$ sowie $b = x - y$ und dann $a \times b$

die sich in ihrer *numerischen* Qualität ziemlich dramatisch unterscheiden können. Schlechte Rechenergebnisse müssen also nicht notwendigerweise aus schlechten Problemen resultieren, sondern können auch durch ungeschickt gewählte numerische Verfahren verursacht sein. Dazu gibt es eine Faustregel:

> Die meisten numerischen Katastrophen sind Folge von Stellenauslöschungen. Daher sollte man bei allen numerischen Verfahren unbedingt darauf achten, die Subtraktion von Zahlen gleichen Vorzeichens und ungefähr gleichen Betrags zu vermeiden!

2.4 Fehler vor und zurück

Zum Abschluss des Fehlerkapitels noch ein etwas systematischer Zugang zur Natur der Fehler. Wenn wie eine **Berechnung** $f : D \to \mathbb{R}$, $D \subseteq \mathbb{R}^N$, am Computer als $\widehat{f} : D \to \mathbb{F}$ realisieren, wobei $\mathbb{F}$ die **Fließpunktzahlen** sind, dann haben wir mit allen möglichen Fehlerquellen zu kämpfen:

Rundungsfehler , die in jeder einzelnen Rechenoperation auftreten können und dies auch werden.

Datenrundung kann dafür sorgen, daß schon die ersten Operanden nicht mehr exakt sind, sondern bereits fehlerbehaftet.

Meßfehler müssen in der realen Welt immer einkalkuliert werden – die Eingangsdaten gibt es praktisch nie exakt, sondern immer mit irgendwelchen Ungenauigkeiten.

Das alles zusammen summiert sich dann zum **Verfahrensfehler**, den man gerne irgendwie beschreiben und wenn möglich auch in den Griff bekommen möchte. Das Konzept, das alle drei Fehlerquellen unter einen Hut bekommt, nennt sich **Rückwärtsfehler** und geht auf Wilkinson[33]:

[33] Sozusagen der Vater der Numerik.

Zu Eingabedaten $x \in D$ bestimme *gestörte Eingabedaten* $\widehat{x}$, so daß exakte Rechnung mit den gestörten Eingabedaten das selbe Ergebnis liefert wie numerische Rechnung mit den exakten Eingabedaten:

$$\widehat{f}(x) = f\left(\widehat{x}\right).$$

Der **Rückwärtsfehler** δ ist dann die (kleinste) Abweichung

$$\delta = \frac{|x - \widehat{x}|}{|x|} \qquad \text{bzw.} \qquad \widehat{x} = (1+\varepsilon)\, x, \quad |\epsilon| \leq \delta.$$

Dadurch, daß der Rückwärtsfehler alles auf die Eingabedaten schiebt, lassen sich die oben genannten Fehlerquellen so einheitlich und konsistent behandeln, und man erhält einen gemeinsamen **Verfahrensfehler**.

Beispiel 2.7 *Um zu sehen, daß der Rückwärtsfehler auch eine theoretische Rechfertigung für unsere Erfahrung bei der Berechnung von $x^2 - y^2$ liefert, sehen wir ihn uns eben einmal an. Dabei gilt wieder die Spielregel für die numerische Durchführung der Grundrechenarten, jetzt mit einem Kringel markiert:*

$$a \odot b = (1+\epsilon)\,(a \cdot b)\,, \qquad |\epsilon| \leq u.$$

Beginnen wir also mit dem "schlechten" Verfahren, das zuerst x^2 und y^2 berechnet, und zwar als

$$x \otimes x = (1+\epsilon)\, x^2 \qquad \textit{und} \qquad y \otimes y = (1+\eta)\, y^2.$$

Durch Subtraktion dieser Werte erhalten wir dann

$$\begin{aligned}(x \otimes x) \ominus (y \otimes y) &= (1+\delta)\left((1+\epsilon)\, x^2 - (1+\eta)\, y^2\right) \\ &= x^2 - y^2 + (\delta + \epsilon + \delta\epsilon)\, x^2 - (\delta + \eta + \delta\eta)\, y^2\end{aligned}$$

und somit[34]

$$\frac{|(x \otimes x) \ominus (y \otimes y)|}{|x^2 - y^2|} \leq 1 + \left(2u + u^2\right) \frac{x^2 + y^2}{|x^2 - y^2|},$$

der Rückwärtsfehler kann[35] *also sogar beliebig groß werden, wenn nur x und y nahe genug beisammenliegen ohne selbst Null zu sein.*

[34] Die genaue Abschätzung kann man sich als Übung leicht überlegen, das ist wirklich nicht wild, nur eben technisch.

[35] Die Betonung liegt allerdings nach wie vor auf „kann"! Hier wird fleissig die *Dreiecksungleichung* $|a \pm b| \leq |a| + |b|$ verwendet und die kann, je nach Vorzeichenverteilung von a und b exakt sein oder aber den Fehler recht großzügig überschätzen. So gesehen geht's, zumindest ohne weitere Annahmen, nicht wirklich besser, was nicht bedeutet, daß die Abschätzung gut sein muss.

Beim anderen Verfahren hingegen ist

$$(x \oplus y) \otimes (x \ominus y) = (1+\delta)\,(x \oplus y)\,(x \ominus y) = (1+\delta)\,(1+\epsilon)\,(1+\eta)\left(x^2 - y^2\right),$$

was zu der wesentlich besseren Abschätzung

$$\frac{|(x \oplus y) \otimes (x \ominus y)|}{|x^2 - y^2|} \leq 1 + 3u + 3u^2 + u^3$$

*führt – nicht vergessen: u ist eine sehr **kleine** Zahl! Insbesondere ist der Rückwärtsfehler des zweiten Verfahrens sogar von den EIngangsdaten unabhängig.*

Man sieht: Rückwärtsfehleranalyse gibt durchaus Informationen, und zwar beweisbare Informationen, darüber, was gute und was schlechte numerische Verfahren zur Durchführung einer Berechnung sind. Und auch wenn die Rückwärtsfehleranalyse selten mathematisch aufregend ist, sondern vielmehr eine akribische und aufwendige Fleissarbeit darstellt, und obendrein die hergeleiteten Fehlerabschätzungen den Fehler gerne um Größenordnungen überschätzen, so ist sie dennoch wichtig und hilfreich, denn meistens erkennt man bei so einer Rückwärtsfehleranalyse, was die kritischen Parameterkonfigurationen sind, die für Ärger sorgen.

Gut, nun haben wir also alle Verfahrensfehler auf die Störung der Eingabedaten geschoben, nun müssen wir eigentlich nur noch betrachten, wie störungsempfindlich die Berechnungsfunktion f ist, also wie stark sie unter kleinen Veränderungen des Eingansparameters variiert. Damit wir den Aufwand nicht übertreiben, **linearisieren** wir diese Größe:

Die **Konditionszahl** κ ist die (kleinste) Zahl, so daß

$$|f(y) - f(x)| \leq \kappa\,|x - y|, \qquad x \sim y. \tag{2.5}$$

Vielleicht kommt einem das κ ja bekannter vor, wenn wir die Gleichung in

$$\frac{|f(y) - f(x)|}{|x - y|} \leq \kappa, \qquad |x - y| \to 0$$

umschreiben, wo die linke Seite verdächtig an einen Differentialquotienten[36] erinnert.

Die Konditionszahl ist eigentlich "nur" eine obere Schranke für die Ableitung von f.

[36] Eigentlich ist die Ableitung ja auch nichts anderes als eine *lokale Linearisierung* der Funktion, also genau das, was wir hier auch machen.

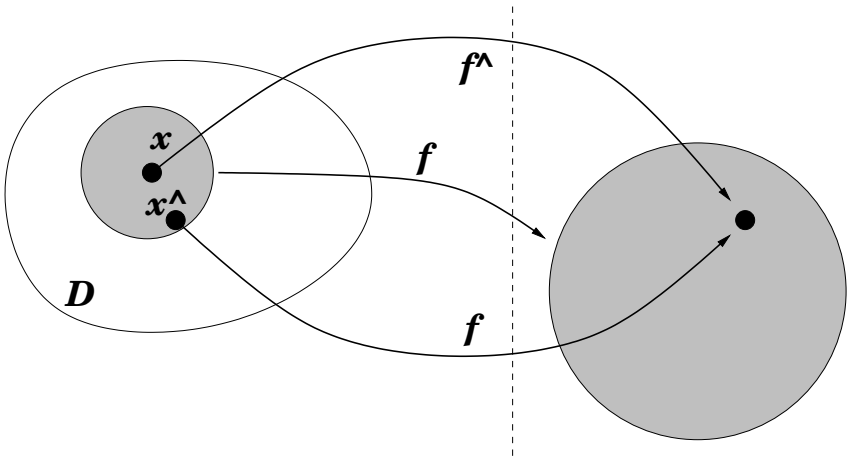

Abbildung 2.1: Schematische Darstellung von Rückwärts- und Vorwärtsfehler. Zu jedem $x \in D$ suchen wir einen Punkt $\widehat{x} \in D$, der mit genauer Rechnung dasselbe Ergebnis liefert wie die fehlerbehaftetet Rechnung mit x. Der Rückwärtsfehler ist nun ein Radius, so daß ein Kreis um x mit diesem Radius für **jedes** $x \in D$ auch das "magische" $\widehat{x}$ enthält. Die Konditionszahl sagt uns nun, wie stark solch ein Kreis von der Funktion f "aufgeblasen" wird. Und in solch einem "aufgeblasenen" Kreis liegen dann letztlich der "Sollwert" $f(x)$, den wir gerne berechnet hätten und der "Istwert" $\widehat{f}(x)$, den wir in Wirklichkeit berechnet haben.

Und damit haben wir eigentlich schon den "Hauptsatz der Fehleranalyse", nämlich die wundervolle Faustformel

$$\text{Fehler} \leq \text{Rückwärtsfehler} \times \text{Konditionszahl}$$

in der Verfahren und Problem entkoppelt sind:

- Der Rückwärtsfehler misst die Qualität des numerischen Verfahrens, also **wie gut** wir unseren Job machen.
- Die Konditionszahl hingegen beschreibt, wie sensibel das Problem gegen Störungen ist und wie genau wir demzufolge rechnen müssen. Ein gut konditioniertes Problem ist somit ein einfaches Problem, bei einem schlecht konditionierten Problem hingegen müssen wir uns beim Berechnungsverfahren deutlich mehr anstrengen.

> *And it didn't stop being magic just because you found out how it was done.*
>
> T. Pratchett, *Wee Free Men*

3 Lineare Gleichungssysteme

Linare Gleichungssysteme spielen auch heute noch eine zentrale Rolle in vielen Naturwissenschaftlichen Anwendungen, auch wenn sie bereits aus den ältesten überhaupt erhaltenen mathematischen Überlieferungen bekannt sind, wie diesem in [11] genannten:

> $\frac{1}{4}$ Breite und Länge zusammen sind 7 Handbreiten, Länge und Breite zusammen sind 10 Handbreiten. (Susa, 2. Jahrtausend v. Chr.)

Etwas neuer, aber nicht weniger gut das folgende Beispiel aus dem Rechenbuch [12] des Adam Ries, besser als *Adam Riese* bekannt:

> Item / zween wöllen ein Pferdt kaufen / Als A. vnd B. für 15. fl. Spricht A. zum B. gib mir deines gelts ein drittheil / so will ich meins darzu thun / vnd das Pferdt bezahlen. Spricht B. zum A. gib mir von deinem gelt ein viertheil / so wil ich mit meinem gelt hinzu das pferdt bezahlen. Nun frage ich / wie viel jeglicher in sonderheit gelts hab?

Übung 3.1 Lösen Sie diese beiden linearen Gleichungssysteme. ♢

Ein **lineares Gleichungssystem** besteht aus einer **endlichen** Anzahl von linearen Gleichungen – **linear** ist eine Gleichung in den Variablen $x_1, \ldots, x_n$, wenn sie von der Form

$$y = a_1\, x_1 + \cdots + a_n\, x_n \qquad \text{also} \qquad y = \sum_{k=1}^{n} a_k x_k$$

ist, was wir unter Verwendung des **Skalarprodukts** auch kurz und knapp als[37] $y = \boldsymbol{a}^T \boldsymbol{x}$ schreiben können. Haben wir es nun mit m derartigen linearen Gleichungen zu tun, also

[37] Wir werden die Konvention nutzen. Vektoren durch **Fettschrift** zu kennzeichnen, was zwar nicht unbedingt nötig, aber deutlicher und nicht so aufdrinlicher wie die lächerlichen Pfeilchen ist.

mit

$$y_j = \boldsymbol{a}_j^T \boldsymbol{x}, \qquad j = 1, \ldots, m.$$

Diese Gleichungen stapeln wir nun übereinander[38] und erhalten so das **Gleichungssystem**

$$\boldsymbol{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix} = \begin{bmatrix} \boldsymbol{a}_1^T \boldsymbol{x} \\ \vdots \\ \boldsymbol{a}_m^T \boldsymbol{x} \end{bmatrix} = \begin{bmatrix} \boldsymbol{a}_1^T \\ \vdots \\ \boldsymbol{a}_m^T \end{bmatrix} \boldsymbol{x} = \begin{bmatrix} a_{11} & \ldots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \ldots & a_{mn} \end{bmatrix} \boldsymbol{x} = \boldsymbol{A}\,\boldsymbol{x},$$

wobei sich in dieser einen Zeile bereits eine Vielzahl von möglichen, verschiedenen aber trotzdem äquivalenten, Schreibweisen finden lässt. Eigentlich sind's alle, die wir brauchen und benutzen werden.

Bemerkung 3.1 *Eine **Matrix***

$$\boldsymbol{A} = \begin{bmatrix} a_{11} & \ldots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \ldots & a_{mn} \end{bmatrix} \in \mathbb{R}^{m \times n}$$

kann in der Mathematik viele Bedeutungen haben! Sie kann eine lineare Abbildung bezüglich einer Basis darstellen[39], sie kann eine quadratische Form[40] darstellen oder aber einfach, wie hier, ein rechteckiges Zahlenschema darstellen, in dem ein lineares Gleichungssystem codiert ist.

Dankbarerweise hat die Menge der Matrizen auch eine Struktur die es uns erlaubt, Matrizen miteinander und mit Vektoren zu multiplizieren, und die wir auch fleissig ausnutzen werden, sei es auch nur, um Dinger kompakter und eleganter zu schreiben.

Auf alle Fälle aber liefern uns die Matrizen nun eine schöne und kompakte Methode, um lineare Gleichungssysteme hinzuschreiben:

Ein **lineares Gleichungssystem** von m Gleichungen in n Unbekannten kann immer in der Form

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \qquad \boldsymbol{A} \in \mathbb{R}^{m \times n}, \quad \boldsymbol{b} \in \mathbb{R}^m,$$

geschrieben werden, wobei die Lösung $\boldsymbol{x} \in \mathbb{R}^n$ gesucht ist.

Bei linearen Gleichungssystemen unterscheiden wir drei Situationen:

[38] So eine **Vektorisierung** ist sehr praktisch und nutzt auch nur die recht banale Erkenntnis aus, daß $\boldsymbol{x} = \boldsymbol{y}$ zu $x_j = y_j$, $j = 1, \ldots, n$, ist.

[39] So sieht man sie in der Linearen **Algebra**.

[40] Das ist die Abbildung $x \mapsto x^T Ax$, die man nun wieder in der analytischen **Geometrie** untersucht.

$m < n$**:** Das Gleichungssystem hat weniger Bestimmungsgleichungen als Variablen[41] und wird daher als **unterbestimmt** bezeichnet. Derartige Gleichungssysteme haben "im Normalfall" unendlich viele Lösungen und die Schwierigkeit[42] besteht eher darin, aus dieser Vielzahl von Lösungen eine gute auszuwählen.

$m = n$**:** Die Anzahl der Gleichungen und der Variablen stimmt überein, das System ist **quadratisch** und hat "im Normalfall" genau eine Lösung. Das ist dann die "schöne" Situation.

$m > n$**:** Hat man mehr Gleichungen als Variablen, so ist das System **überbestimmt** und man hat ein Problem. Das kann man sich leicht klarmachen, denn wählt man n beliebige Gleichungen aus, so bestimmen diese normalerweise eine eindeutige Lösung, die nun aber auch noch die verbliebenen $m - n$ Gleichungen erfüllen muss und wenn diese nicht zufällig *redundant* sind, dann wird es zu einem Widerspruch zwischen den Gleichungen kommen, so daß das Gleichungssystem **keine** Lösung haben wird.

Wir haben jetzt sehr oft auf einen "Normalfall" bzw. eine "normale Situation" verwiesen, etwas, das wir uns genauer ansehen sollten, weil es eigentlich zentral für das Verständnis der numerischen Probleme beim Lösen von linearen Gleichungssystemen ist.

3.1 Lösungsgeometrie

Um zu verstehen, was bei über- und unterbestimmten Gleichungssytemen passiert, betrachten wir Matrizen einmal etwas anders, nämlich als "Codierung" eines Vektorraums. Dazu schreiben wir

$$\mathbb{R}^{m\times n} \ni \boldsymbol{A} = [\boldsymbol{a}_1 \cdots \boldsymbol{a}_n], \qquad \boldsymbol{a}_j \in \mathbb{R}^n, \quad j = 1, \ldots, m,$$

bezüglich seiner Spaltenvektoren[43] und interpretieren das Matrix–Vektor–Produkt

$$\boldsymbol{A}\boldsymbol{x} = \sum_{j=1}^{n} \boldsymbol{a}_j \, x_j, \qquad \boldsymbol{x} \in \mathbb{R}^n,$$

[41] Oft auch als **Freiheitsgrade** bezeichnet.

[42] Man spricht dann oft von *Inversen Problemen*, die in vielen Anwendungen von der Bildverarbeitung bis zur Computertomographie auftauchen, siehe [14].

[43] Ist eine Matrix nun in Wirklichkeit eine Spalte von übereinandergestapelten Zeilenvektoren wie bei der Definition des linearen Gleichungssystems oder eine Zeile von nebeneinandergeschriebenen Zeilen? Sie ist schlichtweg beides (und noch viel mehr) und man kann, darf und soll in der Mathematik immer das Konzept verwenden, das hilfreicher und nützlicher ist.

als **Linearkombination** von $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n$, was uns einen Vektor, also ein Element des Unterraums

$$\mathscr{A} = \boldsymbol{A}\,\mathbb{R}^n = \{\boldsymbol{A}\boldsymbol{x} \;:\; \boldsymbol{x} \in \mathbb{R}^n\} = \text{span}\,\{\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n\} \subset \mathbb{R}^m$$

liefert, und dieser Unterraum $\mathscr{A}$ ist somit in der Matrix $\boldsymbol{A}$ "codiert". Es gibt zwar zu jedem Unterraum viele Matrizen, die ihn darstellen[44], aber zu jeder Basis gehört ein eindeutiger linearer Unterraum. Die theoretische Aussage zum "Normalfall" liest sich nun folgendermaßen.

Lemma 3.2 *Ist $m \geq n$, dann ist praktisch jede Wahl $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n \in \mathbb{R}^m \setminus \{0\}$ von Vektoren linear unabhängig.*

Der Beweis wird auch zeigen, was unter "praktisch jede" zu verstehen ist, nämlich daß sich lineare Unabhängigkeit durch beliebig kleine Störungen der Vektoren erreichen lässt.

Beweis: Induktion über n. Im Fall $n = 1$ haben wir einen Vektor $\boldsymbol{v}_1 \neq 0$, der nach Definition **trivialerweise**[45] linear unabhängig ist.

Für den Induktionsschritt $n-1 \to n$ brauchen wir nur den Fall zu untersuchen, daß $v_1, \ldots, v_n$ linear abhängig wären, daß es also ein $\boldsymbol{x} \neq 0$ gibt, so daß

$$0 = [\boldsymbol{v}_1 \;\ldots\; \boldsymbol{v}_n]\,\boldsymbol{x} = \sum_{j=1}^{n} \boldsymbol{v}_j\, x_j$$

ist. Weil $\boldsymbol{x} \neq 0$ ist, ist mindestens eine Komponente von $\boldsymbol{x}$ von Null verschieden, sagen wir, der Einfachheit halber[46], $x_n \neq 0$, und wir erhalten, daß

$$\boldsymbol{v}_n = -\sum_{j=1}^{n-1} \boldsymbol{v}_j\, \frac{x_j}{x_n}. \tag{3.1}$$

Nach Induktionsannahme können wir $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{n-1}$ so stören, daß die resultierenden Vektoren linear unabhängig sind. Sollte aufgrund dieser Störung auch $\boldsymbol{v}_n$ vom Rest linear unabhängig sein – prima, dann sind wir fertig. Ansonsten gilt (3.1). Da $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{n-1}$ einen Vektorraum der Dimension $n-1 < m$ aufspannen, gibt es mindestens einen Vektor $\boldsymbol{y} \in \mathbb{R}^m$, der sich **nicht** als Linearkombination von $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{n-1}$ schreiben lässt, also linear unabhängig ist. Wegen (3.1) ist dann aber auch $\widetilde{\boldsymbol{v}}_n := \boldsymbol{v}_n + \varepsilon\,\boldsymbol{y}$ linear unabängig, und zwar für jedes $\varepsilon \neq 0$ und sei dieses noch so klein. □

Betrachten wir die Vektoren aus Lemma 3.2 als Spalten einer Matrix und berücksichtigen wir noch, daß es im $\mathbb{R}^m$ maximal m voneinander linear unabhängige Vektoren geben kann, so erhalten wir das folgende Resultat.

[44] So wie ein Vektorraum eben auch viele Basen haben kann...

[45] Versuchen Sie mal, einen von Null verschiedenen Vektor mit einer von Null verschiedenen Zahl so zu multiplizieren, daß das Ergebnis der Nullvektor ist!

[46] Wer will, kann hier gerne das ebenso unsägliche wie nervtötende **o.B.d.A** einsetzen. QED!

Korollar 3.3 *Praktisch jede Matrix $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ hat Rang $\min(m,n)$ und somit hat praktisch jedes lineare Gleichungssystem von m Gleichungen in n Unbekannten einen $n-m$–dimensionalen Lösungsraum, wobei Vektorräume negativer Dimension die leere Menge sind.*

Übung 3.2 Beweisen Sie Korollar 3.3 im Detail. ♢

3.2 Gauß–Elimination, die Schulmethode

Die Idee hinter der naiven Gauß–Elimination ist recht einfach formuliert: Man formt das Gleichungssystem **äquivalent** um. Das kann man tun, indem man zu einer beliebigen Gleichung beliebige, besser aber passende, Vielfache anderer Gleichungen addiert.

Beispiel 3.4 *Die linearen Gleichnungssysteme zu den Matrizen*[47]

$$\boldsymbol{A} = \begin{bmatrix} 1 & 2 & 3 \\ 1 & 0 & 1 \\ 0 & 1 & 2 \end{bmatrix}, \quad \boldsymbol{b} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \qquad \textit{und} \qquad \boldsymbol{A}' = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 2 & 4 \\ 0 & 2 & 3 \end{bmatrix}, \quad \boldsymbol{b}' = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}$$

sind äquivalent, da man $\boldsymbol{A}'$ erhalten kann, indem man zuerst die halbe Differenz der ersten und zweiten Gleichung zur dritten addiert und dann noch die erste Gleichung zur zweiten addiert, und das natürlich ohne die rechten Seiten zu vergessen.

Man sieht schon: Jede derartige **Zeilenumformung** ist ein rein mechanischer Vorgang und wie es sich für mechanische Vorgänge gehört sollten wir sowas vom Computer machen lassen. Der verrechnet sich zwar, wie wir wissen, immer ein bisschen, aber was soll's?
Sehen wir uns also mal an, wie das Spiel in `Octave` für Beispiel 3.4 dann so aussehen würde. Zuerst geben wir die Matrix ein und beachten dabei, daß das Semikolon "`;`" als Trenner zwischen den Zeilen der Matrix fungiert:

```
> A = [ 1 2 3 ; 1 0 1 ; 0 1 2 ]
A =

   1   2   3
   1   0   1
   0   1   2
```

Den Vektor $\boldsymbol{b}$ geben wir in transponierter Form ein[48]:

[47] Von hier an verwenden wir Matrizen, um lineare Gleichungssysteme hinzuschreiben, zuerst einmal nur als Notation, als bequemere Schreibweise, schließlich sparen wir uns so die ohnehin irrelevanten "Namen" der Variablen.

[48] Der Apostroph bezeichnet in `Matlab`/`Octave` die Transposition!

```
> b = [ 1 1 1 ]'
b =

   1
   1
   1
```

Nun könnten wir die Additionen von Zeilen natürlich immer für die Matrix $\boldsymbol{A}$ und die rechte Seite $\boldsymbol{b}$ separat durchführen, aber man kann das auch eleganter machen, indem man den Vektor $\boldsymbol{b}$ rechts an die Matrix $\boldsymbol{A}$ anhängt, was sich in `Octave` durch den einfachen Befehl

```
> Ab = [ A b ]
Ab =

   1   2   3   1
   1   0   1   1
   0   1   2   1
```

erledigen lässt. Und nun machen wir die im Beispiel beschriebenen Transformationen, was mit `Octave` sehr einfach geht[49]. Beginnen wir mit der ersten Operation

```
> Ab( 3,: ) = Ab( 3,: ) + .5*( Ab( 1,: ) - Ab( 2,: ))
Ab =

   1   2   3   1
   1   0   1   1
   0   2   3   1
```

und addieren dann noch die erste Zeile zur zweiten

```
> Ab( 2,: ) = Ab( 2,: ) + Ab( 1,: )
Ab =

   1   2   3   1
   2   2   4   2
   0   2   3   1
```

[49] Der Name `Matlab` bedeutet eigenlich *"Matrix Laboratory"*, was einen ersten Hinweis auf die ursprüngliche Ausrichtung der Software gibt, und `Octave` ist ja auch nichts anderes als ein Open–Source–Klon. Insbesondere die Notation `A( j,: )` bzw. `A( :, k)` für die `j`te Zeile bzw. die `k`te Spalte ist sehr intuitiv und mehr als hilfreich.

und schon haben wir das in Beispiel 3.4 angegebene Ergebnis erhalten. Doch bei aller Freude über die Einfachheit von `Octave` – gebracht hat uns das bisher nichts, das Gleichungssystem am Ende sieht nicht einfacher aus als das am Anfang, nicht verwunderlich, wenn man bedenkt, daß wir ja einfach nur ins Blaue hinein äquivalent umgeformt haben. Ein bisschen zielgerichteter sollte man da schon vorgehen, aber zu welchem Ziel?

Beispiel 3.5 *Die einfachsten linearen Gleichungssysteme sind sicherlich die, bei denen* ***A*** *eine* ***Diagonalmatrix*** *ist, denn diese laufen ja lediglich auf* ***entkoppelte Gleichungen*** *der Form*

$$\begin{array}{rcl} a_{11}\,x_1 & = & b_1 \\ a_{22}\,x_2 & = & b_2 \\ \ddots & \vdots & \vdots \\ a_{nn}\,x_n & = & b_n \end{array}$$

und damit auf

$$x_j = \frac{b_j}{a_{jj}}, \qquad j = 1, \ldots, n,$$

hinaus.

Und genau das ist die Idee – wir bringen die Matrix durch äquivalente Zeilenumformungen Schritt für Schritt auf Diagonalgestalt, und dies in zwei Phasen: Zuerst ziehen wir Zeilen von ihren Nachfolgern ab, um die **Rechtsdreiecksgestalt**

$$\begin{bmatrix} * & * & \ldots & * \\ 0 & * & \ldots & * \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \ldots & 0 & * \end{bmatrix}$$

zu erhalten, wobei $*$ für einen beliebigen Matrixeintrag steht. Ein Gleichungssystem dieser Form kann man bereits mit der sogenannten **Rücksubstitution** direkt lösen, siehe [13], aber – wenn schon naiv, dann richtig naiv – wir bestehen auf Diagonalgestalt, die wir auch erhalten, indem wir nun von unten nach oben Zeilen zu ihren Vorgängern addieren. Sehen wir uns das für unsere einfache 3×3–Matrix mal an, die wir sicherheitshalber nochmals neu eingeben wollen:

```
> Ab = [ 1 2 3 1 ; 1 0 1 1 ; 0 1 2 1 ]
Ab =
```

```
   1    2    3    1
   1    0    1    1
   0    1    2    1
```

Zuerst wollen wir Nullen unterhalb der Diagonalen in der *ersten Spalte* erhalten, und fangen mit der zweiten Zeile an. Um da eine Null zu bekommen, müssen wir offensichlich das 1–fache der ersten Zeile von der zweiten Zeile abziehen:

```
> Ab( 2,: ) = Ab( 2,: ) - Ab( 1,: )
Ab =

   1    2    3    1
   0   -2   -2    0
   0    1    2    1
```

Und siehe da – die erste Spalte passt bereits und es ist bereits Zeit, sich der zweiten Spalte zu widmen. Um da ganz unten eine Null zu bekommen, müssen wir die Hälfte der zweiten Zeile addieren:

```
> Ab( 3,: ) = Ab( 3,: ) + 1/2*Ab( 2,: )
Ab =

   1    2    3    1
   0   -2   -2    0
   0    0    1    1
```

Das war dann auch schon die halbe Miete! Nun wieder von unten nach oben, um die Diagonalgestalt zu bekommen. Dazu addieren wir das Doppelte der dritten Zeile zur zweiten

```
> Ab( 2,: ) = Ab( 2,: ) + 2*Ab( 3,: )
Ab =

   1    2    3    1
   0   -2    0    2
   0    0    1    1
```

subtrahieren das Dreifache der dritten von der ersten,

```
> Ab( 1,: ) = Ab( 1,: ) - 3*Ab( 3,: )
Ab =

   1    2    0   -2
   0   -2    0    2
   0    0    1    1
```

und addieren schließlich noch die zweite Zeile zur ersten, was uns dann

```
> Ab( 1,: ) = Ab( 1,: ) + Ab( 2,: )
Ab =

   1   0   0   0
   0  -2   0   2
   0   0   1   1
```

liefert, von wo wir unseren Lösungsvektor $\boldsymbol{x} = [0, -1, 1]^T$ ohne Probleme ablesen können.
Fassen wir doch noch einmal zusammen, was wir hier wirklich gemacht haben:

Algorithmus 3.6 (Gauß–Elimination, die Schulmethode)

- *Wir haben Vielfache der ersten Zeile zu allen darunterliegenden Zeilen addiert und zwar so, daß in all diesen Zeilen in der ersten Spalte Nullen generiert wurden.*
- *Wir haben passende Vielfache der zweiten Zeile zu den darunterliegenden Zeilen addiert und so in der zweiten Spalte Nullen generiert.*
- *Nach einiger Zeit ist so $\boldsymbol{A}$ in eine **obere Dreiecksmatrix** bzw. eine **Rechtsdreiecksmatrix**[50] transformiert worden, die halbe Miete sozusagen.*
- *Nun führen wir den ganze Prozess nochmals von unten nach oben durch, um zuerst in der letzten Spalte, dann in der vorletzten und so weiter Nullen oberhalb der Diagonalen zu erzeugen.*
- *Von der resultierenden Diagonalmatrix können wir dann sehr einfach die Lösung des Gleichungssystems ablesen.*

Übung 3.3 Führen Sie diesen Prozess für

$$\boldsymbol{A} = \begin{bmatrix} 5 & 0 & 3 & 1 & 5 & 7 \\ 6 & 9 & 9 & 2 & 6 & 2 \\ 2 & 4 & 0 & 3 & 1 & 7 \\ 7 & 7 & 5 & 8 & 7 & 8 \\ 1 & 0 & 3 & 9 & 1 & 2 \\ 1 & 6 & 1 & 2 & 5 & 6 \end{bmatrix}, \qquad \boldsymbol{b} = \begin{bmatrix} 4 \\ 3 \\ 2 \\ 2 \\ 9 \\ 1 \end{bmatrix}$$

durch, wahlweise von Hand, mit dem Taschenrechner, oder mit `Octave`. ♢

[50] Die Begriffe “obere Dreiecksmatrix” und “Rechtsdreiecksmatrix” sind uneingeschränkt und vollkommen synonym.

3.3 Die Freuden des Matrizenkalküls

Im vorhergehenden Kapitel haben wir Matrizen einzig und allein als eine Art Tabellen angesehen, in denen die Koeffizienten des Gleichungssystems gespeichert sind. Nun können Matrizen aber viel mehr, schließlich bilden sie einen **Ring**, in dem **Addition** und **Multiplikation** wohldefiniert sind, nämlich für $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{R}^{n\times n}$

$$(\boldsymbol{A}+\boldsymbol{B})_{jk} = a_{jk} + b_{jk}, \qquad (\boldsymbol{A}\,\boldsymbol{B})_{jk} = \sum_{\ell=1}^{n} a_{j\ell}\, b_{\ell k}, \qquad j,k = 1,\ldots,n. \quad (3.3)$$

Die Formel (3.3) gilt auch für nichtquadratische Matrizen solange im Fall der Addition $\boldsymbol{A}$ und $\boldsymbol{B}$ die gleichen Dimensionen haben, bzw. wenn bei der Multiplikation $\boldsymbol{A} \in \mathbb{R}^{m\times k}$ und $\boldsymbol{B} \in \mathbb{R}^{k\times n}$ ist, was übrigens den Fall des Skalarprodukts mit einschliesst, in dem $m = n = 1$ ist.

Das haben wir, um ganz genau zu sein, schon genutzt, als wir das Gleichungssystem kompakt als $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ zu schreiben, aber das war bisher nur formaler Kleinkram. Was wir noch brauchen, sind die **Einheitsvektoren** $\boldsymbol{e}_j$, $j = 1,\ldots,n$, die in der $j$ten Komponente den Wert 1 haben und sonst nur Nullen enthalten. Ein paar ganz banale Beobachtungen für diesen Kalkül:

- Jeder Vektor $\boldsymbol{y} \in \mathbb{R}^n$ lässt sich trivialerweise als $\boldsymbol{y} = y_1\,\boldsymbol{e}_1 + \cdots + y_n\,\boldsymbol{e}_n$ schreiben.
- $\boldsymbol{e}_j^T\,\boldsymbol{A}$ ergibt einen Zeilenvektor[51], und zwar die $j$te *Zeile* der Matrix $\boldsymbol{A}$.
- Analog liefert $\boldsymbol{A}\boldsymbol{e}_j$ die $j$te *Spalte* von $\boldsymbol{A}$.
- Die Matrix[52] $\boldsymbol{e}_j\boldsymbol{y}^T$ hat in der $j$ten Zeile den Zeilenvektor $\boldsymbol{y}^T$ und sonst lauter Nullen.

Wirklich alles ganz einfach und netter *formaler* Kalkül, aber es ist immer gut, wenn man in der Mathematik Dinge formalisieren kann, denn nur dann lassen sie sich auch wirklich theoretisch untersuchen. Erinneren wir uns nun an unsere Umformungen aus dem letzten Abschnitt, dann haben wir da im ersten Teil immer Zeilen zu weiter unten liegenden Zeilen addiert. Und das geht nun ganz einfach:

[51] Eigentlich eine $1 \times n$–Matrix, denn wir sollten uns schon darüber im Klaren sein, daß wir, um (3.3) verwenden zu können, ja eigentlich die Vektoren in die Matrizen eingebettet haben und einen **Zeilenvektor** mit einer einer $1 \times n$–Matrix identifizieren, während ein **Spaltenvektor** eine $n \times 1$–Matrix ist. Für das Konzept des Vektors als solcher sind Zeile oder Spalte vollkommen irrelevant! Trotzdem ist es weitgehend Standard, Vektoren als Zeilenvektoren aufzufassen, das heißt, die Identifikation $\mathbb{R}^n \simeq \mathbb{R}^{n\times 1}$ vorzunehmen. Daran werden wir uns auch halten.

[52] Als Produkt von $\boldsymbol{e}_j \in \mathbb{R}^n \simeq \mathbb{R}^{n\times 1}$ und $\boldsymbol{y}^T \in \mathbb{R}^{1\times n}$ ist das Ergebnis eine $n \times n$–Matrix.

- $j$te Zeile extrahieren: $\boldsymbol{y}^T = \boldsymbol{e}_j^T \boldsymbol{A}$,
- Matrix bilden, die diese in der k–ten Zeile enthält: $\boldsymbol{Y} := \boldsymbol{e}_k \boldsymbol{y}^T = \boldsymbol{e}_k \boldsymbol{e}_j^T \boldsymbol{A}$,
- diese Matrix mit einem Faktor multiplizieren: $c\boldsymbol{Y} = c\,\boldsymbol{e}_k \boldsymbol{e}_j^T \boldsymbol{A}$,
- und zu $\boldsymbol{A}$ addieren[53]

$$\boldsymbol{A}' = \boldsymbol{A} + c\boldsymbol{Y} = \boldsymbol{A} + c\,\boldsymbol{e}_k \boldsymbol{e}_j^T \boldsymbol{A} = \left(\boldsymbol{I} + c\,\boldsymbol{e}_k \boldsymbol{e}_j^T\right) \boldsymbol{A}.$$

Bleibt noch die Bestimmung von c, was aber auch ganz einfach ist, schließlich war unser Ziel ja, daß

$$0 = \left(\boldsymbol{A}'\right)_{kj} = a_{kj} + c\,a_{jj} \qquad \Longleftrightarrow \qquad c = -\frac{a_{kj}}{a_{jj}}.$$

erfüllt sein sollte. Jetzt können wir aber die ganze $j$te Spalte unterhalb der Diagonalen auf einen Schlag erledigen, indem wir von links mit der Matrix

$$\boldsymbol{L}_j = \boldsymbol{I} - \sum_{k=j+1}^{n} \frac{a_{kj}}{a_{jj}}\, \boldsymbol{e}_k \boldsymbol{e}_j^T, \qquad j = 1, \ldots, n-1, \tag{3.4}$$

multiplizieren. Solch eine Matrix bezeichnet man als **Gauß–Matrix** , sie ist eine **untere Dreiecksmatrix** bzw. **Linksdreiecksmatrix** und vor allem ganz einfach zu invertieren:

$$\left(\boldsymbol{I} - \sum_{k=j+1}^{n} y_k\, \boldsymbol{e}_k \boldsymbol{e}_j^T\right)\left(\boldsymbol{I} + \sum_{k=j+1}^{n} y_k\, \boldsymbol{e}_k \boldsymbol{e}_j^T\right) = \boldsymbol{I} - \sum_{k,\ell=j+1}^{n} y_j y_k \boldsymbol{e}_k \underbrace{\boldsymbol{e}_j^T \boldsymbol{e}_\ell}_{=0} \boldsymbol{e}_j^T = \boldsymbol{I},$$

also

> Die Inverse einer Gauß–Matrix ist wieder eine Gauß–Matrix, bei der lediglich die Vorzeichen der Subdiagonalelemente umgedreht werden.

Damit können wir die erste Hälfte unseres Gauß–Eliminationsverfahrens auch mathematischer auffassen, nämlich als Bestimmung von unteren Dreiecksmatrizen $\boldsymbol{L}_1, \ldots, \boldsymbol{L}_{n-1}$, so daß

$$\boldsymbol{L}_{n-1} \cdots \boldsymbol{L}_1 \boldsymbol{A} = \begin{bmatrix} * & \ldots & * \\ & \ddots & \vdots \\ & & * \end{bmatrix} =: \boldsymbol{R}$$

[53] Oder subtrahieren, aber dafür sorgt ja das Vorzeichen von c.

eine **obere Dreiecksmatrix** bzw. **Rechtsdreiecksmatrix** wird. Die andere Hälfte des Verfahrens funktioniert analog, nur addieren wir jetzt Zeilen zu Ihren *Vorgängern*, was der Multiplikation mit

$$\boldsymbol{R}_j = \boldsymbol{I} - \sum_{k=1}^{j-1} \frac{a_{kj}}{a_{jj}}\, \boldsymbol{e}_k \boldsymbol{e}_j^T, \qquad j = n, n-1, \ldots, 2, \tag{3.5}$$

entspricht. Die Inversen der $\boldsymbol{R}_j$ bildet man dann genau wie oben, indem man wieder "einfach" das "$-$" in (3.5) durch ein "$+$" ersetzt. Da am Schluss ja eine **Diagonalmatrix** herauskommt, ergibt sich somit

$$\boldsymbol{R}_2 \cdots \boldsymbol{R}_n \boldsymbol{L}_{n-1} \cdots \boldsymbol{L}_1 \boldsymbol{A} = \boldsymbol{D} = \begin{bmatrix} * & & \\ & \ddots & \\ & & * \end{bmatrix}$$

oder eben[54]

$$\boldsymbol{A} \;=\; \underbrace{\left(\boldsymbol{L}_1^{-1} \cdots \boldsymbol{L}_{n-1}^{-1}\right)}_{=:\boldsymbol{L}} \underbrace{\left(\boldsymbol{R}_n^{-1} \cdots \boldsymbol{R}_2^{-1}\right)}_{=:\boldsymbol{R}'} \boldsymbol{D} = \boldsymbol{L}\boldsymbol{D}\, \underbrace{\boldsymbol{D}^{-1}\boldsymbol{R}'\boldsymbol{D}}_{=:\boldsymbol{R}} = \boldsymbol{L}\boldsymbol{D}\boldsymbol{R},$$

wobei $\boldsymbol{L}$ und $\boldsymbol{R}$ untere bzw. obere Dreiecksmatrizen mit Einsen auf der Diagonalen sind.

Übung 3.4 Zeigen Sie:

1. Sind $\boldsymbol{L}$ und $\boldsymbol{L}$ untere Dreiecksmatrizen, so ist auch $\boldsymbol{L}\boldsymbol{L}'$ eine untere Dreiecksmatrix.

2. Ist $\boldsymbol{D}$ eine invertierbare Diagonalmatrix, dann ist

$$\left(\boldsymbol{D}^{-1}\boldsymbol{A}\boldsymbol{D}\right)_{jk} = \frac{d_{kk}}{d_{jj}}\, a_{jk}, \qquad j, k = 1, \ldots, n.$$

   Folgern Sie daraus, daß $\boldsymbol{D}^{-1}\boldsymbol{R}\boldsymbol{D}$ eine obere Dreiecksmatrix mit Einsen auf der Diagonale ist, wenn $\boldsymbol{R}$ diese Eigenschaft hat.

♢

Aber sehen wir uns zuerst an, wie wir so ein lineares Gleichungssystem ganz einfach lösen können, wenn wir mal die Matrizen $\boldsymbol{L}_1, \ldots, \boldsymbol{L}_{n-1}$ und $\boldsymbol{R}_n, \ldots, \boldsymbol{R}_2$ bestimmt

[54] Mit ein bisschen elementarster linearer Algebra.

haben. Dazu müssen wir nur die Äquivalenz

$$\begin{aligned} \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} &\iff \underbrace{\boldsymbol{R}_2 \cdots \boldsymbol{R}_n \boldsymbol{L}_{n-1} \cdots \boldsymbol{L}_1 \boldsymbol{A}}_{=\boldsymbol{D}} \boldsymbol{x} = \boldsymbol{R}_2 \cdots \boldsymbol{R}_n \boldsymbol{L}_{n-1} \cdots \boldsymbol{L}_1 \boldsymbol{b} \\ &\iff \boldsymbol{D}\boldsymbol{x} = \boldsymbol{R}_2 \cdots \boldsymbol{R}_n \boldsymbol{L}_{n-1} \cdots \boldsymbol{L}_1 \boldsymbol{b} \\ &\iff \boldsymbol{x} = \underbrace{\boldsymbol{D}^{-1} \boldsymbol{R}_2 \cdots \boldsymbol{R}_n \boldsymbol{L}_{n-1} \cdots \boldsymbol{L}_1}_{=\boldsymbol{D}^{-1}\boldsymbol{L}^{-1}\boldsymbol{R}'^{-1}=\boldsymbol{A}^{-1}} \boldsymbol{b}, \end{aligned}$$

betrachten; die einzige Matrix, die hier zu invertieren ist, ist die *Diagonalmatrix* $\boldsymbol{D}$, und das sollte ja nun wirklich zu schaffen sein! So ganz nebenbei ist das dann auch ein einfacher und effizienter Weg, die Matrix $\boldsymbol{A}^{-1}$ zu berechnen.
Die Formel

$$\boldsymbol{A} = \boldsymbol{L}\boldsymbol{D}\boldsymbol{R} \tag{3.6}$$

ist dann auch die Art, in der ein Mathematiker[55] die Gauß–Elimination behandelt, nämlich als eine **Matrixzerlegung**, die Gauß–Elimination ist nur der Weg, mittels dessen man sie erhält.

3.4 Pivotsuche

Einen Aspekt haben wir aber bei unserer äußerst naiven Schulmethode zur Gauß–Elimination aber bisher total vergessen, nämlich den, daß sie schlicht und einfach so nicht unbeding funktioniert! Anstatt aber lange herumzutheoretisieren, verdeutlichen wir das einfach anhand eines einfachen, kleinen Beispiels.

Beispiel 3.7 *Das lineare Gleichungssytem*

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \boldsymbol{x} = \boldsymbol{b}, \qquad \boldsymbol{b} \in \mathbb{R}^2,$$

hat die offensichtliche Lösung $x_1 = b_2$, $x_2 = b_1$, trotzdem können wir aber die erste Zeile so oft von der zweiten abziehen, wie wir wollen, wir werden es nicht schaffen, eine Null oberhalb der Hauptdiagonalen zu bekommen. Alternativ müssten wir in unserer Formel (3.4) bei der Berechnung von $\boldsymbol{L}_1$ durch $a_{11} = 0$ dividieren, was normalerweise auch nicht besonders gern gesehen wird.

Dieses Problem ist auch recht einfach zu lösen, wenn wir uns an ein weiteres wichtiges Konzept bei linearen Gleichungssystemen erinnern:

> Die Reihenfolge der Gleichungen bei einem Gleichungssystem ist irrelevant – wir können also in der Matrix beliebig Zeilen vertauschen.

[55] Und entgegen aller anderslautenden Propaganda sind Numeriker Mathematiker!

Solange wir die rechten Seiten mitvertauschen, also eben wirklich die Gleichungen austauschen, müssen wir also im Schritt Nummer j nicht unbedingt Zeile Nummer j von allen ihren Nachfolgern abziehen, sondern können die Zeile erst einmal mit einer ihrer Nachfolgezeilen vertauschen, und so dafür sorgen, daß das Diagonalement oder **Pivotelement** von Null verschieden ist. Dabei sind natürlich noch ein paar Fragen offen:

1. *"Was, wenn das nicht geht"?* Wenn an irgendeiner Stelle des Verfahrens die $j$te Spalte der Matrix ab der $j$ten Zeile nur noch Nullen enthält, wenn die Matrix also die Gestalt
$$\boldsymbol{A} = \begin{bmatrix} * & \dots & * & * & * & \dots & * \\ & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ & & * & * & * & \dots & * \\ & & & 0 & * & \dots & * \\ & & & \vdots & \vdots & \ddots & \vdots \\ & & & 0 & * & \dots & * \end{bmatrix},$$
dann liegen die ersten j Spaltenvektoren der Matrix allesamt im $\mathbb{R}^{j-1}$ und müssen daher *linear abhängig* sein – das Gleichungssystem ist nicht lösbar!

2. *"Welche Zeile soll ich austauschen"?* Gute Frage! Das hängt von der Anwendung ab, bei der normalen Programmierung am Computer sucht man sich Zeile Nummer k zum Austauschen so aus, daß $|a_{kj}|$ möglichst groß wird, denn schließlich teilt man ja durch diesen Wert und Division durch kleine Zahlen gilt als nicht so schön[56]. Bei manueller oder symbolischer Rechnung kann es hingegen nützlich sein, die Zeile so zu wählen, daß a_{kj} möglichst einfach ist, am besten den Wert ± 1 hat, denn dann muss man gar nicht dividieren und spart sich eventuelle Brüche.

3. *"Muss ich Zeilen vertauschen"?* Normalerweise nicht, ein "muss" ist es nur, wenn das Pivotelement den Wert Null hat, ansonsten ist es freiwillig und **kann** aber genutzt werden, um die Berechnungen zu vereinfachen oder zu stabilisieren.

Zuerst also unser Fazit:

> Gauß–Elimination mit Pivotsuche, also eventueller Zeilenvertauschung, funktioniert für jede invertierbare Matrix und für andere ist das mit dem Gleichungssystem sowieso nicht so wirklich gut.

Es gibt noch einen witzigen theoretischen Aspekt der Pivotsuche: Auch wenn wir möglicherweise erst *während des Verfahrens* herausfinden, ob und wann wir Zeilen

[56] Das ist nicht der wahre Grund, aber ein durchaus nettes Argument, denn der Exponent einer Fließpunktzahl hat nicht wirklich etwas mit ihrer Genauigkeit zu tun.

vertauschen müssen oder wollen, können wir rein theoretisch auch annehmen, die Gleichungen wären passend sortiert und wir hätten die Vertauschungen in weiser Voraussicht schon *vor der Rechnung* passend durchgeführt. Das wollen wir jetzt in der Folge einfach mal annehmen. Details dieser nicht vollständig trivialen Beobachtung finden sich wieder mal in [13] oder noch detaillierter in [5, 7].

3.5 Gauß und wie man es "richtig" macht

So schön die $\boldsymbol{LDR}$–Zerlegung aus (3.6) auch ist, wir haben uns da viel zu viel Arbeit gemacht! Man kann nämlich auch Dreieckssysteme direkt lösen. Schreiben wir nämlich $\boldsymbol{Lx} = \boldsymbol{b}$ explizit als

$$\begin{array}{llcll} \ell_{11}x_1 & & & & = b_1 \\ \ell_{21}x_1 & + \ell_{22}x_2 & & & = b_2 \\ \vdots & \vdots & \ddots & & \vdots \quad \vdots \\ \ell_{n1}x_1 & + \ell_{n2}x_2 & \dots & + \ell_{nn}x_n & = b_n \end{array}$$

dann sehen wir, daß uns die erste Gleichung sofort x_1 als $x_1 = b_1/\ell_{11}$ liefert, daß wir das in die zweite Gleichung einsetzen und nach $x_2 = (b_2 - \ell_{21}x_1)/\ell_{22}$ auflösen können, was uns letztendlich die Formel

$$x_j = \frac{1}{\ell_{jj}} \left(b_j - \sum_{k=1}^{j-1} \ell_{jk}\, x_k \right), \qquad j = 1, \dots, n, \tag{3.7}$$

liefert, die man als **Vorwärtselimination** bezeichnet, ihr Gegenstück

$$x_j = \frac{1}{r_{jj}} \left(b_j - \sum_{k=j+1}^{n} r_{jk}\, x_k \right), \qquad j = n, \dots, 1, \tag{3.8}$$

das man durch Auflösen des Rechtsdreieckssystems $\boldsymbol{Rx} = \boldsymbol{b}$ "von unten nach oben" erhält, heißt hingegen **Rücksubstitution**.

Nachdem wir nun Dreiecksgestalt als ausreichend erkannt haben, können wir also bereits nach der ersten Hälfte unserer Elimination aufhören, das heißt, wir bestimmen lediglich $\boldsymbol{L}_1, \dots, \boldsymbol{L}_{n-1}$, so daß

$$\boldsymbol{L}_{n-1} \cdots \boldsymbol{L}_1 \boldsymbol{A} = \boldsymbol{R}$$

und erhalten so die Zerlegung[57]

$$\boldsymbol{A} = \boldsymbol{L}_1^{-1} \cdots \boldsymbol{L}_{n-1}^{-1} \boldsymbol{R} =: \boldsymbol{LR}, \tag{3.9}$$

[57] Nicht vergessen: Wie wir die $\boldsymbol{L}_j$ invertieren, das wissen wir und übrigens ist auch das Produkt dieser Matrizen einfach berechnet, man schreibt nur ihre Einträge unterhalb der Diagonalen in die resultierende Matrix, braucht also bei der numerischen Rechnung noch nicht einmal Matrixmultiplikationen!

die man als LR**–Zerlegung** bezeichnet. Mit deren Hilfe ist nun die Lösung des Gleichunssystems ein Kinderspiel:

$$\boldsymbol{b} = \boldsymbol{Ax} = \boldsymbol{LRx} =: \boldsymbol{Ly}, \qquad \boldsymbol{y} := \boldsymbol{Rx},$$

so daß wir zuerst das Gleichungssystem $\boldsymbol{Ly} = \boldsymbol{b}$ lösen und damit durch Lösen von $\boldsymbol{Rx} = \boldsymbol{y}$ unsere eigentlich Lösung des Systems. Das sind, wenn wir die Zerlegung einmal haben, zwei Dreieckssysteme, die sich mit (3.7) und (3.8) leicht lösen lassen, sofern wir nur einmal die LR–Zerlegung bestimmt haben[58]. Fassen wir zusammen:

> Im Gegensatz zur "Schulmethode" besteht die "richtige" Gauß–Elimination in der Bestimmung einer LR–Zerlegung und nachfolgendem Lösen von Dreieeckssystemen. Die rechte Seite kommt erst im zweiten Schritt, nicht aber bei der Zerlegung ins Spiel.

3.6 Normen, Fehler und Numerik

Zum Abschluss des Kapitels über lineare Gleichungssysteme noch ein bisschen was zum Thema Fehler. Nachdem nun die berechneten Werte Vektoren sind, brauchen wir natürlich sowas wie ein Maß zur Bestimmung der **Größe** oder **Länge eines Vektors**. Die axiomatische Konkretisierung dieses Konzepts ist der Begriff der Norm.

Definition 3.8 *Eine Abbildung* $\|\cdot\| : \mathbb{R}^n \to \mathbb{R}$ *heißt* ***Norm****, wenn sie den Bedingungen*

1. ***(Positivität)*** $\|\boldsymbol{x}\| \geq 0$ *und*[59] $\|\boldsymbol{x}\| = 0$ *genau dann wenn* $\boldsymbol{x} = 0$*,*
2. ***(Skalierbarkeit/positive Homogenität)*** *für* $c \in \mathbb{R}$ *ist* $\|c\boldsymbol{x}\| = |c|\,\|\boldsymbol{x}\|$*,*
3. ***(Dreiecksungleichung)*** $\|\boldsymbol{x} + \boldsymbol{y}\| \leq \|\boldsymbol{x}\| + \|\boldsymbol{y}\|$*,* $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$*,*

genügt.

Übung 3.5 Wie viele und welche Normen gibt es auf dem $\mathbb{R}^1$? ♢

Die drei populärsten Vektornormen auf dem $\mathbb{R}^n$ sind

[58] Tatsächlich ist die Zerlegung der Matrix viel aufwendiger als das Lösen der Dreieckssysteme, so daß diese Strategie große Vorteile hat, wenn dasselbe lineare Gleichungssystem für mehrere verschiedene rechte Seiten gelöst werden muss, was in der Praxis durchaus vorkommt.

[59] Fehlt diese Eigenschaft, daß nur der Nullvektor Norm Null hat,so spricht man von einer **Halbnorm**.

Euklidische Norm: Die "pythagorasbedingte" Länge

$$\|\boldsymbol{x}\|_2 = \sqrt{\boldsymbol{x}^T\boldsymbol{x}} = \sqrt{\sum_{j=1}^{n} x_j^2},$$

die die Länge des "Pfeils" vom Koordinatenursprung zum Punkt $\boldsymbol{x}$ angibt. Dieser Längenbegriff ist deswegen so bedeutsam, weil er **affin invariant** ist, sich also unter Rotation und Verschiebung des Koordinatensystems nicht verändert.

"Manhattan–Norm": Markiert man auf einem karierten Blatt Papier einen Nullpunkt, so ist die minimale Anzahl von Schritten entlang des Rasters, die man benötigt, um nach $\boldsymbol{x}$ zu kommen gerade

$$\|\boldsymbol{x}\|_1 := \sum_{j=1}^{n} |x_j| \,.$$

Der etwas prosaische "Populärname" dieser Norm stammt von der verkehrstechnischen Struktur der Halbinsel Manhattan mit ihren rechtwinklig verlaufenden Straßen, deren Einbahnstruktur allerdings bei der Norm nicht berücksichtigt wird.

Supremumsnorm: Auch die Betrachtung des größten Eintrags führt zu einer wichtigen Norm:

$$\|\boldsymbol{x}\|_\infty = \max_{j=1,\ldots,n} |x_j| = \sqrt[\infty]{\sum_{j=1}^{n} |x_j|^\infty}.$$

Diese drei Normen sind Spezialfälle der sogenannten p–Normen, die für $1 \leq p < \infty$ als

$$\|\boldsymbol{x}\|_p = \left(\sum_{j=1}^{n} |x_j|^p\right)^{1/p}, \qquad \boldsymbol{x} \in \mathbb{R}^n,$$

definiert sind.

Übung 3.6 Zeichnen Sie für diese drei Normen die **Einheitssphäre** $\{\boldsymbol{x} \,:\, \|\boldsymbol{x}\| = 1\}$ und beweisen Sie die Inklusionen, die von der Zeichnung suggeriert werden. ♢

Was den Normen recht ist, das kann den Matrizen nur billig sein, denn schließlich bilden auch die Matrizen einen Vektorraum[60]! Und tatsächlich besteht die Übertragung der Norm auf Matrizen nur aus wortwörtlichem Abschreiben der Definition.

Definition 3.9 *Eine Abbildung* $\|\cdot\| \,:\, \mathbb{R}^{m\times n} \to \mathbb{R}$ *heißt* ***Matrixnorm****, wenn sie den Bedingungen*

[60] Was hoffentlich endlich die schlichtweg unrichtige Vorstellung *Vektoren = Pfeile* korrigiert.

1. ***(Positivität)*** $\|\boldsymbol{A}\| \geq 0$ *und* $\|\boldsymbol{A}\| = 0$ *genau dann wenn* $\boldsymbol{A} = 0$*,*
2. ***(Skalierbarkeit/positive Homogenität)*** *für* $c \in \mathbb{R}$ *ist* $\|c\boldsymbol{A}\| = |c| \, \|\boldsymbol{A}\|$*,*
3. ***(Dreiecksungleichung)*** $\|\boldsymbol{A} + \boldsymbol{B}\| \leq \|\boldsymbol{A}\| + \|\boldsymbol{B}\|$*,* $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{R}^{m \times n}$*,*

genügt.

Etwas interessanter ist dann schon die Frage, wie man konkret Matrixnormen definieren kann. Dazu gibt es zwei einfache Prozeduren, sich aus einer Vektornorm Matrixnormen zu "basteln":

Vektorisierung der Matrix: Durch "Stapeln" der Spaltenvektoren kann eine Matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ immer auf kanonische Weise in einen Vektor der $\mathbb{R}^{mn}$ umgewandelt werden, auf den man dann die Vektornorm anwenden kann. Ein typisches Beispiel hierfür ist die **Frobenius–Norm** , bei der die euklidische Norm verwendet wird, was zu

$$\|\boldsymbol{A}\|_F = \sqrt{\sum_{j=1}^{m} \sum_{k=1}^{n} |a_{jk}|^2}$$

führt.

Operatornormen hingegen messen, wie stark ein Matrix die Länge von Vektoren vergrößert oder verkleinert. Die zur Vektornorm $\|\cdot\|_v$ gehörige[61] **Operatornorm** $\|\cdot\|$ ist dann als

$$\|\boldsymbol{A}\| := \max_{\boldsymbol{x} \neq 0} \frac{\|\boldsymbol{A}\boldsymbol{x}\|_v}{\|\boldsymbol{x}\|_v} = \max_{\|\boldsymbol{x}\|_v = 1} \|\boldsymbol{A}\boldsymbol{x}\|_v \tag{3.10}$$

definiert.

Die erste Anwendung dieser Nomenklatur besteht darin, daß wir unser Korollar 3.3 jetzt endlich mathematisch korrekt hinschreiben können, und damit ein "belastbare" Aussage haben.

Korollar 3.10 *Zu jeder Matrix* $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ *und zu jedem* $\varepsilon > 0$ *gibt es eine* ***invertierbare*** *Matrix* $\boldsymbol{A}' \in \mathbb{R}^{n \times n}$*, so daß* $\|\boldsymbol{A} - \boldsymbol{A}'\| < \epsilon$ *ist.*

Matrizen nach "invertierbar" und "nicht invertierbar" zu klassifizieren ist also aus der Sicht der Numerik, in der es aufgrund der ubiquitären Rundung keine absolute Genauigkeit geben kann, also nicht sinnvoll. Was wir brauchen ist eine Größe, die uns sagt, ob eine Matrix **gut invertierbar** oder **schlecht invertierbar** ist. Und solch eine Größe, die nach

[61] Hier steckt die impizite Annahme, daß $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ eine *quadratische* Matrix ist, so daß wir auf $\boldsymbol{x}$ und $\boldsymbol{A}\boldsymbol{x}$ dieselbe Vektornorm anwenden können – ansonsten bräuchten wir halt zwei Vektornormen, um die Operatornorm zu basteln, aber das sind an dieser Stelle unwichtige Details.

Korollar 3.10 auch für praktisch alle Matrizen definiert ist, gibt es in der Tat, nämlich die **Konditionszahl**

$$\kappa(\boldsymbol{A}) := \|\boldsymbol{A}\| \left\|\boldsymbol{A}^{-1}\right\|. \tag{3.11}$$

Bemerkung 3.11 (Konditionszahl)

1. *Natürlich hängt die Konditionszahl von der verwendeten Norm ab. Da aber im Endlichdimensionalen ohnehin alle Normen äquivalent sind, ist das ohnehin nicht so wichtig.*

2. *Je größer die Konditionszahl, desto weniger oder schlechter invertierbar ist die Matrix. Man kann sogar zeigen, daß für jede Folge $\boldsymbol{A}_n$ von Matrizen, die gegen eine nicht invertierbare Matrix $\boldsymbol{A}$ konvergiert, die Folge $\kappa(\boldsymbol{A}_n)$ der Konditionszahlen gegen ∞ divergieren muss.*

3. *Systeme wie `Matlab` oder `Octave` überprüfen das sogar und warnen gegebenenfalls den Benutzer. Das kann man sich an einem einfachen Beispiel verdeutlichen*[62]*:*

```
> A = [ 1+eps 1 ; 1 1 ]; b = [ 1 2 ]';
> A\b
warning: matrix singular to machine precision,
         rcond = 5.55112e-17
warning: attempting to find minimum norm solution
ans =

   0.75000
   0.75000
```

Hierbei steht `rcond` für reverse condition number*, also den Reziprokwert $\kappa^{-1}(\boldsymbol{A})$ der Konditionszahl*[63] *von $\boldsymbol{A}$. Die ausgegebene Lösung ist übrigens falsch:*

```
> A*ans
ans =

   1.5000
   1.5000
```

[62] Mit `eps` erhält man in `Matlab`/`Octave` den Rundungseinheitsfehler.
[63] Die im übrigen nur abgeschätzt, nicht berechnet wird, letzteres wäre zu aufwendig.

4. *Konditionszahl? Den Begriff hatten wir doch schon mal, wenn auch in einem anderen Kontext, nämlich in (3.11) als Linearisierung der Fehlerempfindlichkeit einer Funktion f. Und tatsächlich ist $\kappa(\boldsymbol{A})$ auch wirklich nichts anderes als eine solche Linearisierung der Funktion $f(\boldsymbol{A}) = \boldsymbol{A}^{-1}$, ein Beweis findet sich beispielsweise in [13]. Und das passt auch, denn schließlich ist ganz mathematisch–allgemein und theoretisch–oberflächlich das Lösen von $\boldsymbol{Ax} = \boldsymbol{b}$ nichts anderes als das Berechnen von $\boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{b}$. Die Konditionszahl ist also eine Konditionszahl im Sinne der Konditionszahl.*

Fassen wir zusammen:

> Die Konditionszahl $\kappa(\boldsymbol{A})$ ist eine Größe, die uns eine **quantitative** Information über die Invertierbarkeit der Matrix $\boldsymbol{A}$ gibt.

Übung 3.7 Zeigen Sie: Ist $\|\cdot\|$ eine **konsistente Matrixnorm**, , d.h, gilt $\|\boldsymbol{AB}\| \leq \|\boldsymbol{A}\|\,\|\boldsymbol{B}\|$, dann ist $\kappa(\boldsymbol{A}) \geq \|\boldsymbol{I}\|$. ♢

Man kann nun auch Fehlerabschätzungen für das Lösen von linearen Gleichungssystemen angeben, sie finden sich beispielsweise in [13], wer sich mehr dafür interessiert sollte einen Blick in [7] werfen. Was aber interessant ist, ist die Beziehung

$$\kappa(\boldsymbol{A}) < u^{-1} \qquad \text{bzw.} \qquad \kappa(\boldsymbol{A})^{-1} > u$$

zwischen der Konditionszahl der Matrix und der Genauigkeit der verwendeten Arithmetik. Alle Matrizen, deren Konditionszahl diese Bedingung nicht erfüllt, sind für diese Arithmetik einfach zu schlecht und man sollte es besser bleiben lassen – das ist auch die Bedeutung der Warnung, die `Matlab`/`Octave` liefern.

Hier noch ein kurzes Beispiel mit `Octave`, da zeigt, wie unterschiedlich die Ergebnisse eines "naiven" und eines "professionellen" Lösungsverfahrens sein können. Dazu betrachten wir zuerst einmal

```
> A = [ 1+eps 1 ; 1 1 ]; b = [ 1; 2 ];
```

und lösen mit einem selbstgebauten, naiven Gauss[64]:

```
> x = solveLS( A,b )
x =

  -4.5036e+15
   4.5036e+15
```

[64] Berechnung der LR–Zerlegung wie oben und dann Vorwärtselimination und Rücksubstitution, also **kein** Schulgauss, aber auch keine Pivotsuche.

Tolle Lösung – die liegt in einer Größenordnung, die eigentlich unbrauchbar ist, ist aber erstaunlicherweise korrekt:

```
> A*x - b
ans =

   0
   0
```

Das alles funktioniert aber nur noch im Bereich des numerischen Rauschens, denn eigentlich ist ja

$$\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} -x \\ x \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

und somit ist das Ergebnis oben schon ziemlicher Mist[65], denn die beiden Einträge der "Lösung" unterscheiden sich praktisch nicht:

```
> format long
> x = solveLS( A,b )
x =

  -4.50359962737050e+15
   4.50359962737050e+15
```

Ganz anders sieht es aus, wenn wir die eingebaute Lösungsroutine[66] von `Octave` verwenden, denn da erhalten wir

```
> x = A\b
warning: matrix singular to machine precision,
         rcond = 5.55112e-17
warning: attempting to find minimum norm solution
x =

   0.750000000000000
   0.750000000000000

> A*x - b
```

[65] Und die Resultate können durchaus von Rechner zu Rechner variieren.

[66] Eine kleine Warnung an dieser Stelle: Wie genau sich `Octave` in diesen numerischen Grenzsituationen verhält, das hängt stark von dem verwendeten Rechner (Prozessor/Betriebssystem), der `Octave`–Version ab und eventuell „darunter liegenden" Bibliotheken ab. Das mag etwas widersinnig und fehlerhaft klingen, ist es aber nicht: Reproduzierbarkeit geht bei den numerischen Gradwanderungen normalerweise als erstes verloren, denn nun spielen all die kleinen aber feinen Details der Arithmetik, also exakte Mantissenlänge, Behandlung von Rundung etc., sogar die **dominierende** Rolle.

```
ans =

   0.500000000000000
  -0.500000000000000
```

Wieder taucht die Meldung auf, daß das Problem schlecht konditioniert ist, die Lösung ist absolut gesehen schlechter, dafür aber liegt sie in vernünftigen Größenordnungen und kann vernünftig weiterverarbeitet werden.

> Das Schlimme an der "Lösung" unseres naiven Verfahrens ist die Tatsache, da diese riesigen Zahlen alle **folgenden** Berechnungen unrbauchbar machen werden – die Dimensionen dieser Zahlen stehen in keinem Verhältnis zu den Dimensionen der Parameter des linearen Gleichungssystems!

Lustig wird es auch, wenn wir unser Gleichungssystem etwas ändern und die Gleichungen unterschiedlich skalieren:

```
> x = solveLS( A*diag([100,1]), b )
x =

  -3.00239975158033e+13
   3.00239975158033e+15
```

Und das ist mysteriös! Denn eigentlich sollte, wenn $[x, y]^T$ eine Lösung von $\boldsymbol{Ax} = \boldsymbol{b}$ ist, ja $[x/100, y]$ eine Lösung des umskalierten Problems sein, was wieder einmal zeigt, daß diese "Lösungen" reine Phantasiegebilde sind. Und nochmals:

> Während der Berechnung dieser Lösungen tritt kein einziger Fehler auf, den der Computer bemerkt, für ihn sind alle Rechnungen sauber und korrekt! Das Ergebnis ist trotzdem Mist!

Und schließlich noch ein ganz besonderes Schmankerl:

```
> A = [ 100+eps 100 ; 1 1 ]; b = [ 1; 2 ];
octave> x = solveLS( A,b )
warning: division by zero
x =

  -Inf
   Inf
```

Hier wird bei der Division durch 100 die Störung `eps` so klein, daß bei der Rechnung komplett ignoriert wird und es wird "exakt" durch Null dividiert. Naja, immerhin sowas wie eine Fehlermeldung. `Octave` selbst liefert in diesem Fall

```
> x = A\b
warning: matrix singular to machine precision,
         rcond = 1.03053e-19
warning: attempting to find minimum norm solution
x =

   0.00509949005099490
   0.00509949005099490

octave> A*x-b
ans =

   0.0198980101989803
  -1.9898010198980103
```

der seltsame Fehler erklärt sich dadurch, daß die Lösungsroutine von Octave die Zeilen immer automatisch skaliert und dasselbe auch mit der rechten Seite macht.

Sine victoriae spe nemo volens in aciem descendit.
Ohne Hoffnung auf Sieg zieht niemand freiwillig in den Kampf.

Petrarca, *De spe vincendi – Von der Hoffnung auf Sieg*, 1366

4 Nichtlineare Gleichungen und Nullstellen

Nachdem wir uns bisher mit *vielen einfachen*, nämlich linearen, Gleichungen in *vielen* Variablen zugewandt haben, betrachten wir jetzt die umgekehrte Problemstellung, nämlich eine Gleichung in nur einer Variablen, die dafür aber auch beliebig kompliziert sein darf. Eine allgemeine Gleichung ist von der Form

$$f(x) = g(x), \qquad f, g \in C(\mathbb{R}), \tag{4.1}$$

wobei Stetigkeit von rechter und linker Seite eine vernünftige Minimalannahme ist. Durch den Taschenspielertrick, die rechte Seite von der linken zu subtrahieren, wird unser Problem aber sofort zu

$$0 = f(x) - g(x) =: \widetilde{f}(x).$$

Das Lösen beliebiger nichtlinearer Gleichungen kann immer auf das Finden einer **Nullstelle** einer Funktion zurückgeführt werden.

Und genau damit werden wir uns nun befassen, mit der Frage, wie man eine Nullstelle einer Funktion finden kann.

4.1 Bisektion und Regula Falsi

Das **Bisektionsverfahren**, auch als **Intervallschachtelung** bezeichnet, ist die einfachste Methode zum Auffinden einer Nullstelle und basiert lediglich auf dem Zwischenwertsatz[67] der Analysis:

[67] Der ist so einleuchtend, daß er intuitiv fast keines Beweises bedarf. Umso wichtiger ist in solchen Fällen aber ein Beweis!

Zwischen zwei Punkten x und y nimmt eine *stetige* Funktion jeden Wert zwischen $f(x)$ und $f(y)$ mindestens einmal an.

Die Idee, um eine Nullstelle von f zu finden, ist nun sehr einfach. Wir beginnen mit zwei Punkten x_+ und x_-, die so zu wählen sind, daß

$$f(x_-) < 0 < f(x_+)$$

ist. Wie man solche Punkte findet? Im Allgemeinen mit viel Glück! Aber im Ernst: Die Existenz dieser Punkte geht schlicht und ergreifend als Voraussetzung ein, sie zu finden kann in der Realität durchaus mit Aufwand verbunden sein – im Zweifelsfalle muss man einfach f sehr oft auswerten bzw. fein abtasten, bis man zwei derartige Punkte findet.

Aber wenn man sie einmal hat, dann muss sich nach dem Zwischenwertsatz irgendwo zwischen x^- und x^+ *mindestens* eine Nullstelle befinden, der wir nun auf den Leib rücken wollen. Dazu bilden wir $x = \frac{1}{2}(x^+ + x^-)$ und sehen uns den Wert $f(x)$ an:

$f(x) = 0$**:** Wir haben die Nullstelle gefunden und sind fertig. Besser kann es nicht kommen.

$f(x) > 0$**:** Wir ersetzen x^+ durch x.

$f(x) < 0$**:** Wir ersetzen x^- durch x.

Ist die Nullstelle nicht gefunden, dann wiederholen wir diese Operation. Da hierbei in jedem Schritt der Abstand zwischen x^+ und x^- halbiert wird und da die Nullstelle x^* zwischen den beiden Punkten liegt, also

$$\left| x^* - \frac{1}{2}\left(x^+ + x^-\right) \right| \leq \frac{1}{2}\left|x^+ - x^-\right|$$

erfüllt, nähern wir uns unser Nullstelle sehr schnell an. Genauer: Wenn wir mit $x_k^\pm$ die Grenzen im k–ten Schritt bezeichnen und mit $x_k = \frac{1}{2}\left(x_k^+ + x_k^-\right)$ den Mittelwert, dann gilt

$$|x^* - x_k| \leq \frac{1}{2}\left|x_k^+ - x_k^-\right| = \frac{1}{2^{k+1}}\left|x_0^+ - x_0^-\right|, \tag{4.2}$$

wir gewinnen also in jedem Iterationsschritt eine Binärstelle an Genauigkeit! Ganz klar – das Bisektionsverfahren ist nun wirklich die einfachste Methode, um Nullstellen zu bestimmen. Allerdings funktioniert es nur, wenn man auch wirklich zwei Startpunkte kennt, an denen f unterschiedliches Vorzeichen hat und es ist ausserdem nicht klar, *welche* Nullstelle gefunden wird, wenn es mehrere Nullstellen zwischen x^+ und x^- gibt, siehe Abb. 4.2. Das Bisektionsverfahren versucht sein Glück einfach in der Mitte zwischen den beiden Punkten $x^\pm$ und interessiert sich dabei lediglich für das *Vorzeichen* von

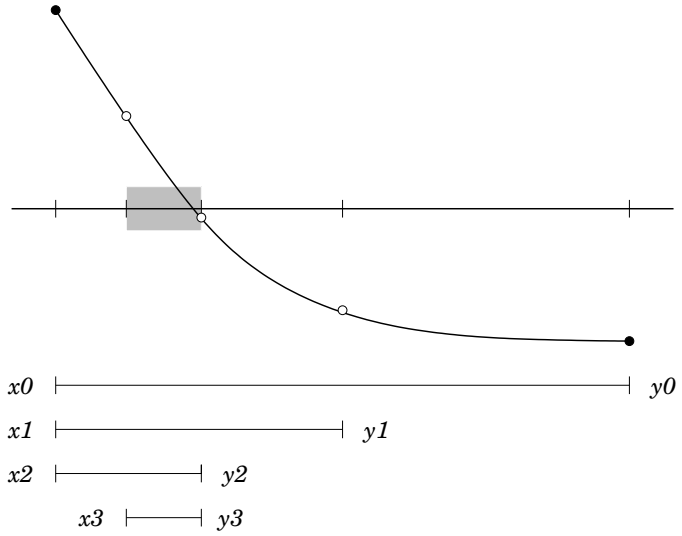

Abbildung 4.1: Die Vorgegehensweise beim Bisektionsverfahren. Ersetzt wird immer der Endpunkt, dessen Vorzeichen mit dem Vorzeichen am Mittelpunkt übereinstimmt. Nach jedem Iterationsschritt dieses Verfahrens besteht – wegen der Stetigkeit der Funktion – nun immer die Gewißheit, daß sich zwischen den Endpunkten eine Nullstelle befindet, nach unseren drei Schritten hier also irgendwo im schraffierten Intervall.

f an diesen Stellen. Man könnte heuristisch aber annehmen, daß die Nullstelle vielleicht näher bei der Stelle liegen könnte, an der der Absolutbetrag von f kleiner ist. Und genau das ist die Idee bei der **Regula Falsi**, die nicht einfach den Mittelpunkt verwendet, sondern die Werte $f\left(x^{\pm}\right)$ verbindet und den Nulldurchgang dieses Streckenzugs als "Mittelwert" x verwendet, siehe Abb. 4.3. Berechnen wir doch mal als Fingerübung, wo dieser Nulldurchgang liegen müsste. Die Gerade, die an den Stellen $x^{\pm}$ die Werte $f\left(x^{\pm}\right)$ annimmt, hat die Gleichung[68]

$$\begin{aligned} \ell(x) &= \frac{x-x^-}{x^+-x^-} f\left(x^+\right) + \frac{x-x^+}{x^--x^+} f\left(x^-\right) \\ &= x\frac{f\left(x^+\right)-f\left(x^-\right)}{x^+-x^-} - \frac{x^- f\left(x^+\right) - x^+ f\left(x^-\right)}{x^+-x^-}, \end{aligned}$$

und wir können $\ell(x) = 0$ nach x auflösen, was uns

$$x = \frac{x^- f\left(x^+\right) - x^+ f\left(x^-\right)}{f\left(x^+\right) - f\left(x^-\right)} \tag{4.3}$$

[68] Wie man das herausfindet? Man bemerkt, daß ℓ eine lineare/affine Funktion ist und setzt dann einfach $x^{\pm}$ ein. Und durch zwei Punkte geht halt nun mal genau eine Gerade.

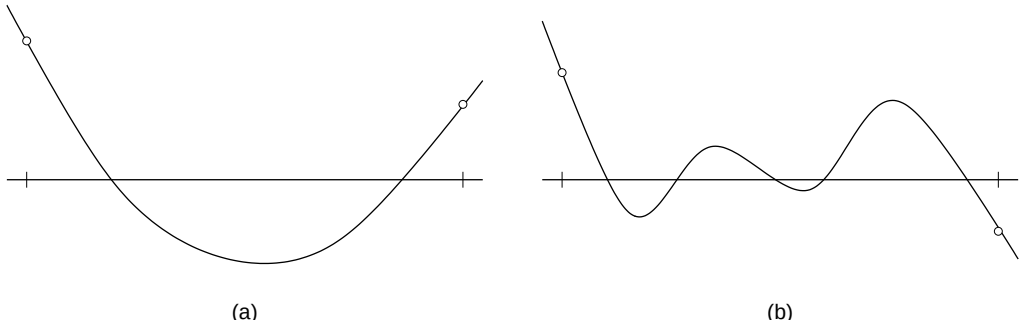

Abbildung 4.2: Im Beispiel (a) kann das Bisektionsverfahren nicht angewandt werden, da die Anfangsbedingung nicht erfüllt ist, obwohl man nach einem Iterationsschritt und Austausch eines beliebigen Endpunkts "in business" wäre, im Beispiel (b) ist die Anfangsbedingung zwar erfüllt, welche Nullstelle aber gefunden wird, ist mehr oder weniger eine Glückssache.

liefert. Der Rest ist dann wieder wie vorher, eine Iteration auf der Basis der Fallunterscheidung

$f(x) = 0$**:** Wir haben die Nullstelle gefunden und sind fertig. Besser kann es nicht kommen.

$f(x) > 0$**:** Wir ersetzen x^+ durch x.

$f(x) < 0$**:** Wir ersetzen x^- durch x.

Die Regula Falsi findet sich in Europa[69] erstmals bei Leonardo da Pisa, besser als Fibonacci bekannt, hat aber nichts mit Kaninchen zu tun. Bei Adam Ries [12] findet sich dann allerdings schon die folgende sehr einleuchtende Beschreibung von (4.3):

Regula Falsi oder Position.

Wirdt gefaßt von zweyen falschen zahlen / welche der auffgab nach / mit fleiß examinirt sollen werden / in massen das fragstück begeren ist / sagen sie der warheit zu viel / so bezeichne sie mit dem zeichen + plus / wo aber zu wenig / so beschreib sie mit dem zeichen − minus genannt. Als dann nimb ein lügen von der andern / was da bleibt / behalt für den theiler / multiplicir darnach im Creuz ein falsche zahl mit der andern lügen / nimb eins vom andern / vnd das da bleibt theil ab mit fürgemachtem theiler / so kompt berichtung der frag.

[69] Oder, wem das lieber ist, im "Abendland".

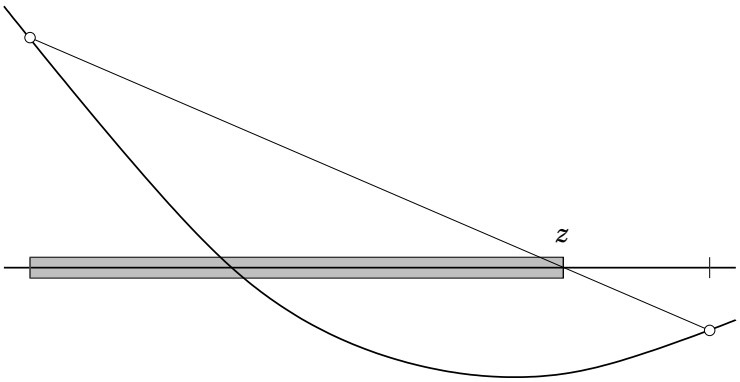

Abbildung 4.3: Ein Iterationsschritt der Regula Falsi. Der "Testpunkt" wird nicht einfach als Intervallmitte gewählt, sondern als (eindeutige!) Nullstelle der Verbindungsstrecke zwischen $f(x^+)$ und $f(x^-)$. Weitergemacht wird also mit dem schraffierten Bereich.

4.2 Rein iterative Verfahren – Sekanten und Newton

Bisektion und Regula Falsi gehören zur "Familie" der **Einschlussverfahren**, die als Ergebnis ein Folge von Intervallen liefern, zwischen denen eine Nullstelle liegen **muss**. Das gibt einem natürlich jede Menge Sicherheit über Existenz der Nullstelle und Konvergenz des Verfahrens[70], stellt aber eben auch Anforderungen an die Startwerte. Hier setzt nun das **Sekantenverfahren** an, das man als einfache Weiterentwicklung der Regula Falsi ansehen kann. Wir beginnen nun mit zwei *beliebigen* Startwerten x_0 und x_1, interessieren uns überhaupt nicht mehr für irgendwelche Vorzeichen oder systematische Ersetzungen und bestimmen aus diesen beiden Werten einen neuen Wert, ganz genau nach der Methode der Regula Falsi,

$$\begin{aligned} x_2 &= \frac{x_0 f(x_1) - x_1 f(x_0)}{f(x_1) - f(x_0)} = \frac{x_0 f(x_1) - x_0 f(x_0) + x_0 f(x_0) - x_1 f(x_0)}{f(x_1) - f(x_0)} \\ &= x_0 \frac{f(x_1) - f(x_0)}{f(x_1) - f(x_0)} - \frac{x_1 - x_0}{f(x_1) - f(x_0)} f(x_0) = x_0 - \frac{x_1 - x_0}{f(x_1) - f(x_0)} f(x_0)\,. \end{aligned}$$

Geometrisch haben wir hier einfach die Sekante durch die Paare $(x_0, f(x_0))$ und $(x_1, f(x_1))$ mit der x–Achse geschnitten und das Ergebnis als neuen Punkt x_2 verwendet. Das Spiel können wir jetzt natürlich mit x_1 und x_2 wiederholen und erhalten so eine Interations-

[70] Zumindest beim Bisektionsverfahren, bei dem sich die Intervalllänge ja garantiert in jedem Iterationsschritt halbiert.

folge

$$x_{j+1} = x_{j-1} - \frac{x_j - x_{j-1}}{f(x_j) - f(x_j)} f(x_{j-1}), \qquad j \in \mathbb{N}, \tag{4.4}$$

von Nullstellen von Sekanten, siehe 4.4, was auch den Namen *Sekantenverfahren* begründet und rechtfertigt. Das Sekantenverfahren ist nun ein *rein iteratives* Verfahren,

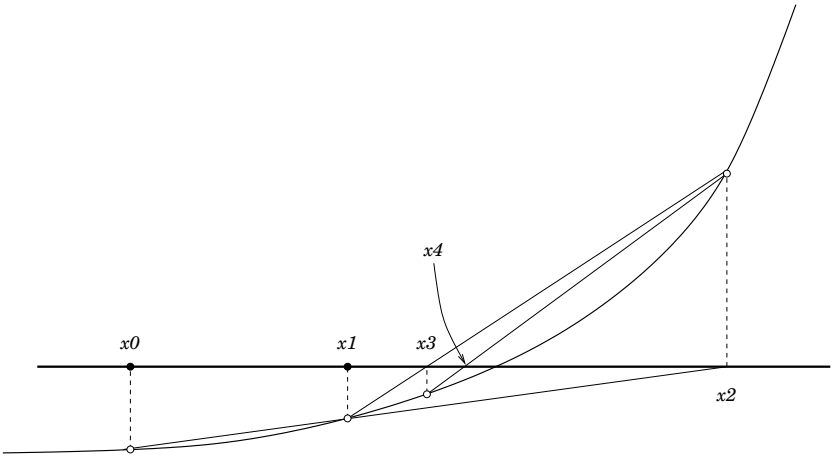

Abbildung 4.4: Einige Iterationsschritte des Sekantenverfahrens.

das ausgehend von den Startwerten x_0, x_1 eine Folge von Werten x_j, $j \in \mathbb{N}$, erzeugt, die hoffentlich gegen die Nullstelle von f konvergieren. Und in der Tat gibt es auch beweisbare Aussagen über die Kovergenz des Sekantenverfahrens, beispielsweise die folgende aus [13].

Satz 4.1 *Es sei $x \in \mathbb{R}$ eine* einfache *Nullstelle von $f \in C^2(a,b)$ mit $x \in (a,b)$. Dann gibt es eine Umgebung*[71] *U_x von x, so daß für alle Startwerte $x_0, x_1 \in U_x$ die Folge x_j, $j \in \mathbb{N}$, gegen die Nullstelle x konvergiert.*

Wie man sieht ist da einiges an Voraussetzungen zu erfüllen, die Funktion muss deutlich glatter[72] sein als nötig und das Verfahren funktioniert nur dann, wenn die Startwerte bereits hinreichend nahe an der Lösung liegen.

Definition 4.2 *Ein numerisches Verfahren, das ausgehend von Startwerten x_0, x_1 eine Iterationsfolge x_j, $j \in \mathbb{N}$, für einen gesuchten Wert x erzeugt, heißt*

[71] Zur Erinnerung: Eine **Umgebung** U_x eines Punktes x ist eine *offene* Menge mit $x \in U_x$. Die typische Vorstellung von Umgebungen in diesem Kontext hier ist, daß es sich um **kleine** Umgebungen handelt

[72] Im Sinne von Differenzierbarkeit.

1. ***global konvergent**, wenn $x_j \to x$ für **alle** Startwerte x_0, x_1.*
2. ***lokal konvergent**, wenn es eine Umgebung U_x von x gibt, so daß $x_j \to x$ für alle Startwerte $x_0, x_1 \in U_x$.*

1. Das Sekantenverfahren ist ein lokal konvergentes Verfahren.
2. Lokal konvergente Verfahren stehen und fallen mit der Wahl des Startwerts bzw. der Startwerte. Nur wenn diese gut gewählt sind, funktioniert das Verfahren auch, für schlechte Startwerte kann es divergieren, zwischen irgendwelchen Werten pendeln oder aber auch gegen eine andere Nullstelle kovergieren.

Bringt uns dann das Sekantenverfahren dann auch etwas außer der Ungewissheit über die Konvergenz des Verfahrens? Aber klar! Zuerst einmal können wir die Startwerte jetzt beliebig wählen, es gibt keine Forderungen mehr an die Vorzeichen von $f\left(x_{0/1}\right)$, und zweitens konvergiert das Sekantenverfahren schneller, es hat nämlich Konvergenzordnung[73] $\psi = \frac{1+\sqrt{5}}{2} \approx 1.6180$. Die **Konvergenzordnung** ρ eines Verfahrens ist eine Zahl, für die

$$\sup_{j\in\mathbb{N}} \frac{|x_{j+1} - x|}{|x_j - x|^\rho} < \infty, \qquad \text{also} \qquad |x_{j+1} - x| \approx |x_j - x|^\rho$$

gilt. Details sollen uns hier nicht interessieren[74], aber da $|x_j - x|$ eine Nullfolge ist, gilt für die Konvergenzordnung natürlich: “Je größer, desto besser”. Was quadratische Konvergenz wirklich bedeutet, werden wir uns später aber anhand eines Beispiels klarmachen.

Was am Sekantenverfahren etwas seltsam ist, ist die Tatsache, daß die Rollen der beiden Startwerte x_0 und x_1 nicht symmetrisch sind, denn x_0 wird nach dem ersten Iterationsschritt verworfen, während x_1 noch bei der Bestimmung von x_2 mitspielen darf. Darüberhinaus hat man im Kovergenzfall sowieso, daß irgendwann $x_j \approx x_{j-1}$ sein muss. Was es aber in (4.4) bedeutet, wenn $x_j = x_{j-1} + h$ mit einem sehr kleinen h ist, das können wir uns ja leicht überlegen:

$$\begin{aligned} x_{j+1} &= x_{j-1} - \frac{x_{j-1} + h - x_{j-1}}{f\left(x_{j-1} + h\right) - f\left(x_{j-1}\right)} f\left(x_{j-1}\right) \\ &= x_{j-1} - \left(\frac{f\left(x_{j-1} + h\right) - f\left(x_{j-1}\right)}{h}\right)^{-1} f\left(x_{j-1}\right). \end{aligned}$$

[73] Diese Zahl ist als **goldener Schnitt** berühmt.
[74] Die finden sich wieder einmal in [13].

Das ist eine Beziehung, in der x_j gar nicht mehr auftaucht und in der für $h \to 0$ der **Differenzenquotient** $\frac{f(x_{j-1}+h)-f(x_{j-1})}{h}$ gegen die **Ableitung** $f'(x_{j-1})$ konvergiert. Wir können es auch anders sehen: Lassen wir die beiden Punkte, die die Sekante bestimmen, zusammenfallen, so erhalten wir die **Tangente** und aus dem Sekantenverfahren wird das "Tangentenverfahren"

$$x_{j+1} = x_j - \frac{f(x_j)}{f'(x_j)}, \qquad j \in \mathbb{N}, \tag{4.5}$$

das unter dem Namen **Newton–Verfahren** bekannt ist. Beim Newton–Verfahren brauchen

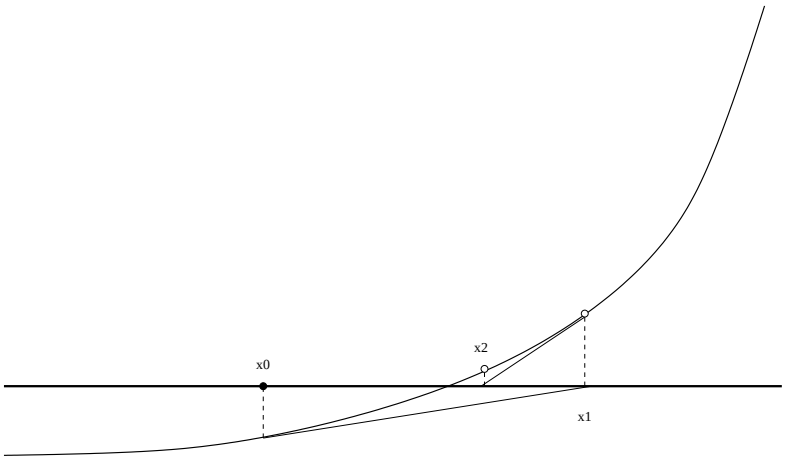

Abbildung 4.5: Und hier ist auch schon das Newton–Verfahren als "Tangentenverfahren", also als Grenzfall des Sekantenverfahrens.

wir nur noch einen Startwert, sagen wir x_1, der "Extra–Startwert" x_0 ist nicht mehr nötig; alternativ kann man natürlich, wie in Abb 4.5, auch mit x_0 anfangen, das ist reine Geschmackssache. Die Konvergenz des Newton–Verfahrens verhält sich im wesentlichen genauso wie die des Sekantenverfahrens.

Satz 4.3 *Es sei $x \in \mathbb{R}$ eine* einfache *Nullstelle von $f \in C^2(a,b)$ mit $x \in (a,b)$. Dann gibt es eine Umgebung U_x von x, so daß für alle Startwerte $x_1 \in U_x$ die Folge x_j, $j \in \mathbb{N}$, gegen die Nullstelle x konvergiert.*

Wir sollten nicht vergessen, daß wir beim Newton–Verfahren mehr brauchen als beim Sekantenverfahren, nämlich die Fähigkeit, die Ableitung von f auszuwerten. Das ist bei einfachen Funktionen wie Polynomen natürlich kein Problem, kann bei komplexeren Funktionen aber durchaus Arbeit oder sogar Schwierigkeiten machen. Dafür gewinnen

wir aber auch etwas, nämlich **quadratische Konvergenz**, die Konvergenzordnung des Newton–Verfahrens[75] ist 2.

Man könnte hier eine Menge Theorie machen, sich mit Fixpunktverfahren und Kontraktionen beschäftigen[76], aber da sich das im Schulunterricht nicht so einfach vermitteln lässt, sehen wir uns lieber zwei illustrative Beispiele an.

4.3 Heron und Division

Das **Heron–Verfahren**[77] dient zur näherungsweisen Bestimmung der Quadratwurzel einer Zahl $y \in \mathbb{R}_+$. Formal ist das nun einfach! Da $\sqrt{y}$ eine Nullstelle des quadratischen Polynoms $f(x) = x^2 - y$ ist, brauchen wir nur dieses f in (4.5) einzusetzen,

$$x_{j+1} = x_j - \frac{x_j^2 - y}{2x_j} = x_j - \frac{1}{2}x_j + \frac{1}{2}\,\frac{y}{x_j} = \frac{1}{2}\left(x_j + \frac{y}{x_j}\right)$$

um die Heron–Iteration

$$x_{j+1} = \frac{1}{2}\left(x_j + \frac{y}{x_j}\right), \qquad x_1 = y, \tag{4.6}$$

zu erhalten, die man übrigens komplett mit Zirkel und Lineal durchführen kann, siehe Abb. 4.6. Nun kannten die Babylonier aber keine Ableitungen, keine Polynome und es existieren auch keine mystischen Keilschriftprophezeihungen, die auf Newton hinweisen, sie haben das Heron–Verfahren wohl über eine einfache geometrische Heuristik gefunden: Die Zahl $\sqrt{y}$ ist nichts anderes als die Seitenlänge eines Quadrats mit Fläche y, so daß der Job darin besteht, ein Rechteck mit den Seitenlängen a und b in ein flächengleiches Quadrat zu verwandeln. Exakt geht das nicht, aber näherungsweise ist es eine gute Idee, den Mittelwert der beiden Seiten, $\frac{a+b}{2}$ als eine Seite des neuen Rechtecks zu nehmen, die andere Seite wäre dann Fläche durch diese Seitenlänge, also $\frac{2ab}{a+b}$. Und dann mittelt man einfach weiter.

Etwas eleganter geht das, wenn wir unser Rechteck anders beschreiben, nämlich über seine Fläche, y und eine Seitenlänge, sagen wir x. Die beiden Seiten des Rechtecks sind dann x und $\frac{y}{x}$ und die Mittelung ergibt die neue Seitenlänge

$$x' = \frac{1}{2}\left(x + \frac{y}{x}\right),$$

[75] Zumindest bei einfachen Nullstellen, ansonsten siehe z.B. [9]

[76] Was schöne und dabei noch recht elementare Mathematik ist, also durchaus ein lohnender Zeitvertreib.

[77] Nach *Heron von Alexandria*, lebte etwa von 10 n. Chr. bis 75 n. Chr., Mathematiker, Physiker und Erfinder. Neben einer Formel für die Berechnung der Dreiecksfläche bringt man seinen Namen auch mit diesem numerischen Verfahren zur Berechnung von Quadratwurzeln in Verbindung, das aber bereits in babylonischen Keilschrifttexten nachgewiesen ist. Außerdem konstruierte er Automaten, darunter selbsttätig zwitschernde Vögel und eine *Dampfmaschine*.

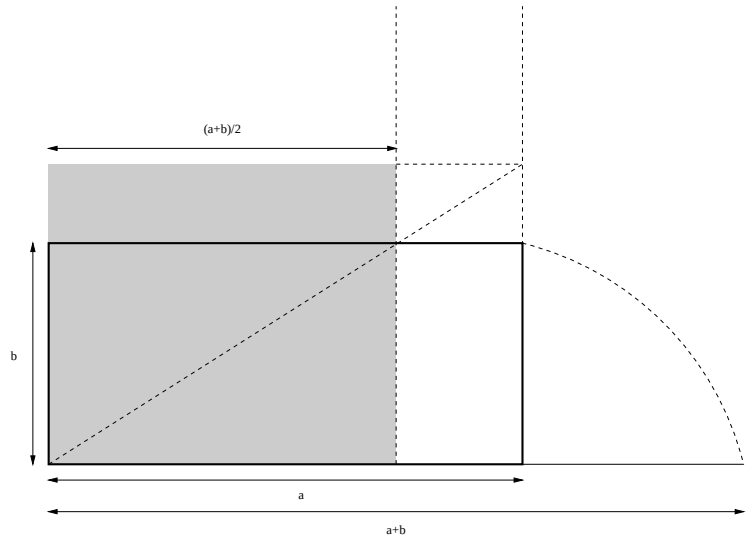

Abbildung 4.6: Ein Iterationsschritt des Heron–Verfahrens in Zirkel- und Linealgeometrie: Man konstruiert zu vorgegebenen Seitenlängen a und b das flächengleiche Rechteck mit den Seitenlängen $\frac{a+b}{2}$ und $\frac{2ab}{a+b}$. Alles was man hier braucht ist ein kleines Bisschen Strahlensatz.

also genau die Heron–Iteration (4.6). So einfach kann das Leben sein.

Das zweite Beispiel stammt aus der Frühzeit der Computer und betrifft die Division zweier Zahlen. Daß und wie Computer addieren und multiplizieren können, damit haben wir uns ja in früheren Kapiteln ziemlich lang und breit beschäftigt. Das Dividieren hingegen haben wir aus guten Gründen zuerst mal weggelassen, denn die "Schulmethode" ist nicht wirklich effizient oder auch nur gut durchführbar. Da wir ja multiplizieren können, brauchen wir, um $x \div y$ zu bestimmen, nur eine Methode, um den **Reizprokwert** y^{-1} von y in gewünschter Genauigkeit zu bestimmen. $1/y$ ist eine Nullstelle der Funktion $f(x) = 1/x - y$ und die Newtoniteration hierzu ist

$$x_{j+1} = x_j - \frac{x_j^{-1} - y}{-x_j^{-2}} = x_j + x_j - y\,x_j^2 = 2x_j - y\,x_j^2$$

und das kann man exklusiv mit Addition und Multiplikation schaffen! Sehen wir uns ein Beispiel an: Wir möchten $\frac{1}{3}$ berechnen, also den Reziprokwert zu $y = 3$ und verwenden als Startwert[78] den Wert $x_1 = 0.3$. Dann ist

$$x_2 \;=\; 2 \times 0.3 - 3 \times (0.3)^2 = 0.6 - 0.27 = 0.33$$

[78] Ein guter Startwert kann nie schaden!

$$\begin{aligned} x_3 &= 2 \times 0.33 - 3 \times (0.33)^2 = 0.66 - 3 \times 0.1089 = 0.66 - 0.3267 = 0.3333 \\ x_4 &= \ldots = 0.33333333 \\ x_n &= 0.\underbrace{3 \ldots 3}_{2^{n-1}} \end{aligned}$$

Können wir das vielleicht sogar beweisen? Können wir! Wir wollen ja zeigen, daß

$$x_n = \sum_{j=1}^{2^{n-1}} 3 \times 10^{-j} = \frac{1}{3} \sum_{j=1}^{2^{n-1}} 9 \times 10^{-j} = \frac{1}{3} \left(1 - 10^{-2^{n-1}}\right)$$

ist. Dann ist

$$\begin{aligned} x_{n+1} &= 2x_n - 3x_n^2 = 2\frac{1}{3} \left(1 - 10^{-2^{n-1}}\right) + 3\frac{1}{9} \left(1 - 10^{-2^{n-1}}\right)^2 \\ &= \frac{2}{3} \left(1 - 10^{-2^{n-1}}\right) - \frac{1}{3} \left(1 - 2 \times 10^{-2^{n-1}} + 10^{-2^n}\right) \\ &= \frac{2}{3} - \frac{1}{3} + \left(-\frac{2}{3} + \frac{2}{3}\right) \times 10^{-2^{n-1}} - \frac{1}{3} \times 10^{-2^n} \\ &= \frac{1}{3} \left(1 - 10^{-2^n}\right), \end{aligned}$$

so daß wir nur noch “QED” sagen können.

Nature is not embarrassed by difficulties of analysis.

A. Fresnel

5 Integration auf die numerische Art

In der Schule (und auch in der Analysis) führt man das **Riemann–Integral**[79] normalerweise über **Treppenfunktionen** ein, also stückweise konstante Funktionen, deren Integral sich dann ganz einfach berechnen lässt. Was man dabei aber eigentlich macht, ist eine *numerische* Integrationsmethode, die als **Quadratur** bezeichnet wird und die viel älter ist als der Integralbegriff selbst.

5.1 Treppenfunktionen

Für ein gegebenes Intervall $[a, b]$, das endlich und nichtentartet sein soll, ist das Integral $\int_a^b f(x)\,dx$ intuitiv die *vorzeichenbehaftete*[80] Fläche, die vom Graphen der Funktion und der x–Achse eingeschlossen wird. Die traditionell pisanische[81] Erklärung der Integration involviert normalerweise einen idealisierten Fliesenleger, der eine Terasse mit einem gekrümmten Rand pflastern soll und der wissen will, wie viele Fliesen er benötigt. Das Integral ist hier unbrauchbar, die Aufgabe selbst dank der Verschnittoptimierung ein interessantes und gar nicht mal triviales Optimierungsproblem.

Dabei ist auch die theoretische Idee hinter dem Integral sehr pragmatisch: Wir führen die Integration einer beliebigen, normalerweise stetigen[82], Funktion auf die Integration *einfacher* Funktionen zurück, deren Integral einfach berechnet werden kann und mit deren Hilfe man die Funktion hinreichend genau annähern kann. Der allereinfachste Fall wäre natürlich eine Funktion, die auf $[a, b]$ konstant ist und dort den Wert y hat, denn dann ist die Fläche ja nichts anderes als die Rechtecksfläche $y(b-a)$. Nun sind konstante Funktionen ja schon etwas langweilig, aber wir müssen nur zu stückweise konstanten Funktionen übergehen, also vielen kleinen Rechtecken, um etwas sehr leistungsfähiges zu bekommen.

[79] Im schulischen Kontext einfach **Integral** genannt.

[80] Was oberhalb der x–Achse liegt wird positiv gezählt, was darunter liegt hingegen negativ.

[81] Eine Pisa–Aufgabe ist per definitionem eine Textaufgabe, in der vermittels eines nicht existenten Realitätsbezugs auswendig gelernte Rechentechniken abgefragt werden.

[82] Wobei endlich viele Sprungstellen bekanntlich nicht wehtun.

Dazu suchen wir uns $n+2$ Punkte

$$a = x_0 < x_1 < \cdots < x_n < x_{n+1} = b$$

und betrachen die **stückweise konstante Funktion** , die auf dem Intervall $[x_j, x_{j+1}]$ den Wert y_j hat, $j = 0, \ldots, n$. Da wir die Integration in

$$\int_a^b f(x)\,dx = \sum_{j=0}^{n} \int_{x_j}^{x_{j+1}} f(x)\,dx, \tag{5.1}$$

also in Integration über Teilstücke, zerlegen können[83], ist das Integral unserer stückweise konstanten Funktion nichts anderes als

$$\int_a^b f(x)\,dx = \sum_{j=0}^{n} \int_{x_j}^{x_{j+1}} y_j\,dx = \sum_{j=0}^{n} y_j\,(x_{j+1} - x_j)\,.$$

Wie es weitergeht wissen natürlich alle: Für ein beliebiges stetiges f setzen wir nun beispielsweise $y_j = f\,(x_j)$ und erhalten so die Näherung

$$\int_a^b f(x)\,dx \approx \sum_{j=0}^{n} (x_{j+1} - x_j)\,f\,(x_j)\,,$$

die noch einfacher wird, wenn wir unsere x_j gleichmäßig im Intervall $[a, b]$ verteilen, also als

$$x_j = a + jh, \qquad j = 0, \ldots, n+1, \qquad h = \frac{b-a}{n+1}, \tag{5.2}$$

mit der **Schrittweite** h wählen. Mit ein bisschen Formalismus haben wir damit also eine Näherung der Form

$$I(f) := \int_a^b f(x)\,dx \approx \sum_{j=0}^{n} w_j\,f\,(\xi_j) =: Q_n(f) \tag{5.3}$$

bestimmt. Dieses Q_n heißt **Quadraturformel** der Ordnung n und jede solche Quadraturformel besteht aus zwei Bestandteilen:

- **Knoten** ξ_j, $j = 0, \ldots, n$, also Stellen, an denen man die Funktion auswerten muss,
- **Gewichte** w_j, $j = 0, \ldots, n$, die die Funktionswerte geeignet bewerten.

[83] Was eigentlich ein Axiom jedes vernünftigen Integralbegriffs ist. Und intuitiv klar oder zumindest sinnvoll zu sein scheint.

Eine anschauliche Zusatzforderung wäre es, $\xi_j \in [a,b]$ zu wählen, denn sonst bestimmt man ja *"... die Masse eines Ziegelsteins aus der Dichte der ihn umgebenden Luft"*, aber solange f außerhalb $[a,b]$ definiert ist, ist die Forderung nicht wirklich nötig. Numeriker haben es außerdem gern, wenn die Gewichte positiv sind, da sich dann das Integral positiver Funktionen besser und genauer berechnen lässt[84]. Aber auch das muss nicht sein!

Das Argument für den Übergang zum Integral ist nun recht einfach[85]: Man wählt die Punkte nach (5.2), setzt $w_j = h$ und macht einen formalen Grenzübergang $h \to 0$, was zumindest für Polynome gut klappt – das, was dann rauskommt, ist das Integral, und los geht's. Zum Berechnen des Integrals einer beliebigen Funktion aber ungeeignet – unser armer Fliesenleger kann also nur polynomiale oder sinusförmige Terassen legen. Aus rein praktischen Überlegungen ist die Quadraturformel doch viel besser!

Fragen wir uns lieber mal, wie gut die Quadraturformel ist, was uns allerdings in eine leichte didaktische Zwickmühle bringt: Um zu *verstehen*, welche Quadraturformel gut ist, brauchen wir ein klein wenig Theorie, die Integrale involviert, aber wir können es ja demonstrieren, indem wir den Computer nutzen. Aber zuerst die Theorie! Wie gut ist so eine Quadraturformel nun? Gehen wir es doch einmal numerisch an und integrieren $f(x) = x^2$ auf dem Intervall $[0,1]$, aber numerisch, indem wir die Punkte $x_j = j/(n+1)$, $j = 0, \ldots, n+1$, benutzen. Das Setup ist also

```
octave> n=10; h=1/(n+1); x = (0:n+1)/(n+1);
```

Zuerst eine "linksseitige" Quadratur, die $\xi_j = x_j$, $j = 0, \ldots, n$, verwendet, was uns numerisch den Wert

```
octave> h*sum( x(1:n+1).^2 )
ans =  0.28926
```

liefert, was den realen Wert, $\frac{1}{3}$, natürlich deutlich unterschätzt. Macht nichts, denn dafür liefert uns die "rechtsseitige"Quadratur $\xi_j = x_{j+1}$

```
octave> h*sum( x(2:n+2).^2 )
ans =  0.38017
```

einen viel zu großen Wert. Am besten ist es noch, wenn wir die Mittelpunkte nehmen, also $\xi_j = \frac{1}{2}(x_j + x_{j+1})$:

```
octave> h*sum( ( ( x(1:n+1) + x(2:n+2) ) /2 ).^2 )
ans =  0.33264
```

Das sieht dann schon eher nach $\frac{1}{3}$ aus. Für $n = 100$ sind die ensprechenden Werte dann schon

[84] Zur Erinnerung: Da war mal was mit Auslöschung bei der Subtraktion ...

[85] So kenne ich es zumindest aus der Schule, aber das ist natürlich auch schon ein paar Jahre her.

```
octave> n=100;h=1/(n+1); x = (0:n+1)/(n+1);
octave> [ h*sum( x(1:n+1).^2 ),
          h*sum( ( ( x(1:n+1) + x(2:n+2) ) /2 ).^2 ),
          h*sum( x(2:n+2).^2 ), ]
ans =

   0.32840   0.33333   0.33830
```

was in allen drei Fällen schon recht nahe am korrekten Wert liegt, tatsächlich sind die Fehler

```
octave> ans.-1/3
ans =

  -4.9342e-03  -8.1691e-06   4.9668e-03
```

Prima! Aber knnen wir das auch begründen? Können wir! Nach dem Hauptsatz der Differential- und Integralrechnung ist für $x \in [x_j, x_{j+1}]$

$$f(x) - f(\xi_j) = \int_{\xi_j}^{x} f'(t)\,dt, \qquad \text{also} \qquad |f(x) - f(\xi_j)| \le |x - \xi_j| \max_{t\in[x_j,x_{j+1}]} |f'(t)|$$

und somit[86]

$$\begin{aligned}
&\left| \int_{x_j}^{x_{j+1}} f(t)dt - \underbrace{h}_{=x_{j+1}-x_j} f(\xi_j) \right| \\
&= \left| \int_{x_j}^{x_{j+1}} f(t) - f(\xi_j)\; dt \right| \le \int_{x_j}^{x_{j+1}} |f(t) - f(\xi_j)|\; dt \\
&\le \max_{t\in[x_j,x_{j+1}]} |f'(t)| \int_{x_j}^{x_{j+1}} |t - \xi_j|\; dt \\
&= \max_{t\in[x_j,x_{j+1}]} |f'(t)| \left(\int_{x_j}^{\xi_j} (\xi_j - t)\; dt + \int_{\xi_j}^{x_{j+1}} (t - \xi_j)\; dt \right) \\
&= \max_{t\in[x_j,x_{j+1}]} |f'(t)| \; \frac{1}{2} \left((x_{j+1} - \xi_j)^2 + (x_j - \xi_j)^2 \right)
\end{aligned}$$

Wie bekommen wir den Fehler nun am kleinsten, wenn wir nur an ξ drehen dürfen und f nicht kennen? Natürlich indem wir ξ "in die Mitte" legen, denn dann wird er

[86] Nichts anderes als Dreiecksungleichungen ...

$\frac{1}{2}(x_{j+1}-x_j)^2$, was immerhin um einen **garantierten**[87] Faktor 2 besser ist. Das erklärt aber noch nicht die so deutlich kleineren Wert für die Regel mit dem Mittelwert, denn wenn wir die nur um 4% verändern, also den Punkt ein wenig nach links legen, dann erhalten wir bereits

```
octave:3> h*sum( ( (.52* x(1:n+1) + .48*x(2:n+2) ) ).^2 )
ans =  0.33313
octave:4> 1/3-ans
ans =  2.0615e-04
octave:5> 1/3 - h*sum( ( (.51* x(1:n+1) + .49*x(2:n+2) ) ).^2 )
ans =  1.0717e-04
```

was ebenfalls schon wieder um zwei Dezimalstellen schlechter ist, was sich durch unsere eben erhaltene einfache Abschätzung nicht erklären lässt.

5.2 Exaktheitsgrad

Es muss also einen signifikanten Unterschied zwischen dem Mittelpunkt und allen Konfigurationen geben. Und der hat was mit Polynomen zu tun! Ist nämlich f eine *lineare Funktion*, dann ist das Integral von f die Fläche des Trapezes unter x_j und x_{j+1} und die ist bekanntlich

$$\frac{1}{2}\left(f\left(x_j\right)+f\left(x_{j+1}\right)\right),$$

weswegen man in diesem Fall auch von der **Trapezregel** spricht. Diese Trapezregel ist **nicht** die Regel aus dem letzten Abschnitt, wo wir die Fläche des Rechtecks benutzt, aber die **Höhe des Rechtecks** durch den Funktionswert in der Mitte dieses Rechtecks festgelegt haben. Unsere eben gemachte Beobachtung besagt, daß diese Trapezregel alle linearen[88] Polynome **exakt** integriert. Na gut, so ist sie eigentlich gebastelt, da sie genau das schafft.

Definition 5.1 *Eine Quadraturformel Q_n hat **Exaktheitsgrad** m, wenn*

$$Q_n(p) = I(p)$$

für alle Polynome p vom Grad $\leq m$ gilt.

Beispiel 5.2 *Die einfache Treppenfunktionsformel hat Exaktheitsgrad* 0*, die Trapezregel Exaktheitsgrad* 1.

[87] Das sind alles nur obere Abschätzung, die den schlimmsten aller möglichen Fälle betrachten, der Unterschied kann und wird normalerweise deutlich weniger signifikant sein. Trotzdem – man bekommt es ja eigentlich umsonst.

[88] Oder affinen, je nach Sprechweise.

Was bringt uns diese Exaktheit nun? Ganz einfach: Signifikant kleineren Fehler! Dazu betrachten wir mal nur ein Integrationshäppchen, also $a = x_1$, $b = x_2$, und nehmen an, daß $f \in C^{m+1}$, also $m+1$–mal stetig differenzierbar ist. Dann gibt es die gute alte Taylerformel, die uns sagt, daß für beliebiges $\xi \in [a,b]$

$$f(x) = \underbrace{\sum_{j=0}^{m} \frac{(x-\xi)^j}{j!} f^{(j)}(\xi)}_{=:T_m f(x)} + \underbrace{\frac{(x-\xi)^{m+1}}{(m+1)!} f^{(m+1)}(\theta_x)}_{=:R_m f(x)}, \qquad \theta \in [a,b],$$

gilt. Das Polynom T_m heißt **Taylorpolynom** der Ordnung m zu f und R_M bezeichnet man als **Restglied** der **Taylorformel**. Nun ist für ein *beliebiges* Polynom $p \in \Pi_m$

$$\begin{aligned}
|I(f) - Q_n(f)| &= |I(f) \underbrace{-I(p) + Q_n(p)}_{=0} - Q_n(f)| = |I(f-p) - Q_n(f-p)| \\
&\leq |I(f-p)| + |Q_n(f-p)| \\
&\leq \int_a^b |f(x) - p(x)| \; dx + \sum_{j=0}^{n} |w_j| \; |f(x_j) - p(x_j)| \\
&\leq \underbrace{\left(\int_a^b dx + \sum_{j=0}^{n} |w_j| \right)}_{=:C} \max_{x \in [a,b]} |f(x) - p(x)| ,
\end{aligned}$$

wobei die Konstante C ein Parameter der Quadraturformel ist. Sind die Gewichte alle positiv[89], dann ist $C = I(1) + Q_n(1)$, also $C = 2I(1)$, solange die Quadraturformel wenigstens Exaktheitsgrad 0 hat.

Wählen wir nun ganz speziell $p = T_m f$, dann ist für $x \in [a,b]$

$$\begin{aligned}
|f(x) - p(x)| &= |R_m f(x)| = \left| \frac{(x-\xi)^{m+1}}{(m+1)!} f^{(m+1)}(\theta_x) \right| \\
&\leq \max_{t \in [a,b]} \left| f^{(m+1)}(t) \right| \frac{(b-a)^{m+1}}{(m+1)!}
\end{aligned}$$

und das ist **unabhängig** von x und ξ. Also:

$$|I(f) - Q_n(f)| \leq 2I(1) \max_{t \in [a,b]} \left| f^{(m+1)}(t) \right| \frac{(b-a)^{m+1}}{(m+1)!}. \tag{5.4}$$

Die Konsequenz: Integriert man mit der Rechtecksregel numerisch über Intervalle der Länge h, dann liefert diese einen Fehler der Größenordnung h, die Trapezregel hingegen die viel bessere Größenordnung h^2 – die Idee ist ja immer, daß h klein wird!

[89] Und das ist ein, wenn nicht der Grund, warum Positivität der Gewichte ganz gern gefordert wird.

5.3 Quadratur und zusammengesetzte Quadratur

Die Fehlerabschätzung (5.4) gibt uns die zwei wesentlichen Ideen für die numerische Integration vor, nämlich

- hoher Exaktheitsgrad,
- kleine Integrationsintervalle.

Wie wir das mit dem Exaktheitsgrad machen, das müssen wir noch sehen, aber die kleinen Intervalle sind eigentlich kein Problem. Wenn wir nämlich wieder die Zerlegung von $[a, b]$ in N Teile

$$[a,b] = \bigcup_{j=0}^{N} [x_j, x_{j+1}] =: \bigcup_{j=0}^{N} I_j, \qquad a = x_0 < x_1 < \cdots < x_N < x_{N+1} = b,$$

also eine Zerlegung des Integrals in

$$I(f) = \sum_{j=0}^{N} \int_{x_j}^{x_{j+1}} f(x)\,dx$$

bilden, dann können wir für jedes der Intervalle eine separate Quadraturformel verwenden und das so zu einer großen Quadratur zusammensetzen:

$$Q_n(f) = \sum_{j=0}^{N} Q_n^j(f), \qquad Q_n^j(f) = \sum_{k=0}^{n} w_k^j\, f\left(\xi_k^j\right).$$

Warum man sowas wohl als **zusammengesetzte Qadraturformel** bezeichnet? Die einzelnen Formel Q_n^j können identisch sein[90], im Prinzip könnte sogar jede dieser Quadraturformeln eine eigene Ordnung n_j und einen eigenen Exaktheitsgrad m_j besitzen, aber das würde das Ganze nur noch unübersichtlicher und indexlastiger machen, ohne dabei wirklich zur Einsicht beizutragen.

Sehen wir uns lieber das einfachste Beispiel an: Die Rechtecksregel, die wir am Anfang kennengelernt haben, wird plötzlich zu einer zusammengesetzten Quadraturformel mit sehr einfachen Bestandteilen, nämlich Formeln Q_0^j der Ordnung n und den "lokalen" Parametern

$$w_0^j = x_{j+1} - x_j, \qquad \text{sowie} \qquad \xi_0^j = \begin{cases} x_j, \\ x_{j+1}, \\ \frac{1}{2}\left(x_j + x_{j+1}\right). \end{cases}$$

[90] Zumindest mehr oder weniger.

Die Trapezregel ist da schon einfacher und interessanter:

$$w_0^j = w_1^j = \frac{x_{j+1} - x_j}{2} \qquad \text{und} \qquad \xi_0^j = x_j, \quad \xi_1^j = x_{j+1},$$

also

$$\begin{aligned} Q_n(f) &= \sum_{j=0}^{N} Q_n^j(f) = \sum_{j=0}^{N} \frac{x_{j+1} - x_j}{2} \left(f\left(x_j\right) + f\left(x_{j+1}\right) \right) \\ &= \frac{x_1 - x_0}{2} f\left(x_0\right) + \sum_{j=1}^{N} \frac{x_{j+1} - x_{j-1}}{2} f\left(x_j\right) + \frac{x_{N+1} - x_N}{2} f\left(x_{N+1}\right). \end{aligned}$$

Haben die Punkte alle gleichen Abstand, sagen wir h, voneinander, dann ist natürlich auch $x_{j+1} - x_{j-1} = 2h$ und die zusammengesetzte Trapezregel[91] hat die besonders einfache Form

$$Q_n(f) = h \left(\frac{1}{2} f\left(a\right) + \sum_{j=1}^{N} f\left(x_j\right) + \frac{1}{2} f(b) \right). \tag{5.5}$$

Das sieht fast aus wie die zusammengesetzte Rechtecksregel[92] und ist eigentlich auch nichts anderes als der Mittelwert der Regeln für den rechten und den linken Randpunkt. Wir erinnern uns: Die eine war bei der Parabel immer zu niedrig, die andere immer zu groß und dieser Mittelungsprozess hat Exaktheitsgrad 1, gibt also eine um eine Größendordnung bessere Konvergenzordnung. Vergleichen wird das nochmals für unsere Parabel und zwar mit Hilfe einer Funktion `vierRegeln`, die die Quadratur mit Hilfe der vier uns bisher bekannten Regeln bestimmt, also:

1. Rechtecksregel mit linkem Randpunkt
2. Rechtecksregel mit Mittelpunkt
3. Trapezregel
4. Rechtecksregel mit rechtem Randpunkt

Bei unserem "klassischen" Beispiel erhalten wir dann

```
octave> function y = sqr(x) y = x.^2; endfunction
octave> vierRegeln( "sqr",(0:.01:1) ) .- 1/3
ans =

  -4.9833e-03  -8.3333e-06   1.6667e-05   5.0167e-03
```

[91] Das ist eigentlich **die** Trapezregel.
[92] Auch wieder **die** Rechtecksregel.

Wir sehen, daß die Trapezregel wie erwartet deutlich besser ist als die einfachen Rechtecksregeln. Aber warum bitte schneidet die Rechtecksregel mit Mittelpunkt so gut ab? Eigentlich ganz einfach:

> Auch die Rechtecksregel mit Auswertung am Mittelpunkt ist exakt für *alle linearen Funktionen* und daher funktionert der "Trick" mit dem wir (5.4) hergeleitet haben. Daher muss sie ebenfalls einen Fehler von der Ordnung h^2 haben.

Die lineare Exaktheit der Rechtecksregel mit Mittelpunkt kann man leicht nachrechnen oder geometrisch begründen, aber man kann sie auch in einem größeren Kontext sehen, wie sich ganz am Ende des Kapitels zwigen wird.
Man muss fast schon suchen, um mal einen Fall zu finden, wo die Trapezregel ein klein wenig besser ist:

```
octave> vierRegeln( "cos",(0:.01:pi) )
ans =

   0.0115926   0.0015927   0.0015926  -0.0084074
```

Es gibt aber auch Fälle, da ist die Trapezregel nicht besser als die einfachen Rechtecksregeln. Das heisst, um es wahrheitsgemäßer zu formulieren, in diesen Fällen[93] sind die Rechtecksregeln besser als erwartet[94]!

```
octave> function y = parab(x) y=x.*(1.-x); endfunction
octave> vierRegeln( "parab",(0:.01:1) ) .- 1/6
ans =

  -1.6667e-05   8.3333e-06  -1.6667e-05  -1.6667e-05
```

5.4 Interpolatorische Quadratur und Interpolation

Kehren wir nun zu unserem Ziel einer möglichst großen Exaktheit zurück, also zu Quadraturformeln, die möglichst viele Polynome exakt integrieren. Das ist bei zusammengesetzten Quadraturformeln natürlich eine Eigenschaft der Teilformeln Q_n^j, also letztendlich nichts anderes als eine Eigenschaft "einfacher" Quadraturfomeln für $[a, b]$. Hätten wir nämlich eine derartige Formel mit Knoten $\xi_0, \ldots, \xi_n$ gefunden, dann brauchen

[93] Beispielsweise wenn die zu integrierende Funktion symmetrisch um die Intervallmitte ist – warum ist das so?

[94] Bitte nicht vergessen: (5.4) ist immer nur eine **obere** Abschätzung und der Fehler kann in Einzelfällen deutlich kleiner sein.

wir sie nur noch auf jedes der Intervalle $[x_j, x_{j+1}]$ abzubilden, indem wir die **affine Transformation**

$$\xi_k^j = x_j + \frac{x_{j+1} - x_j}{b-a}(\xi_k - a), \qquad k = 0, \ldots, n, \qquad j = 0, \ldots, N,$$

verwenden, schließlich sind letzten Endes alle Intervalle dasselbe, zumindest bis auf affine Transformationen[95].

Erinnern wir uns nochmals an unsere Trapezregel. Was man hierbei gemacht hat, war, die beiden Endpunkte a und b des Intervalls[96] zu nehmen und diejenige Gerade zu integrieren, die an diesen Stellen die Werte $f(a)$ und $f(b)$ annimmt. Da es durch zwei Punkte eben genau eine Gerade gibt, erhält man das exakte Integral von f, solange $f(x) = ax + b$ eine affine Funktion ist. Wir hätten eigentlich gar nicht die Punkte a, b gebraucht, zwei *beliebige* Knoten $\xi_0, \xi_1 \in [a, b]$ hätten ausgereicht, um aus $f(\xi_0)$ und $f(\xi_1)$ die affine Funktion zu rekonstruieren, und zwar als

$$\ell(x) = \frac{x-\xi_0}{\xi_1-\xi_0} f(\xi_1) + \frac{\xi_1 - x}{\xi_1 - \xi_0} f(\xi_0) = \underbrace{\frac{f(\xi_1) - f(\xi_0)}{\xi_1 - \xi_0}}_{=:a} x + \underbrace{\frac{\xi_1 f(\xi_0) - \xi_0 f(\xi_1)}{\xi_1 - \xi_0}}_{=:b}. \tag{5.6}$$

Die erste der beiden Formeln in (5.6) ist hilfreicher, um einzusehen, daß $\ell(\xi_j) = f(\xi_j)$ ist, und vor allem um eine Idee für den allgemeinen Fall zu bekommen. Die beiden Funktionen,

$$\frac{x-\xi_0}{\xi_1-\xi_0} \qquad \text{und} \qquad \frac{\xi_1 - x}{\xi_1 - \xi_0}$$

haben nämlich die interessante Eigenschaft, daß sie an einer der Stellen ξ_0, ξ_1 verschwinden und an der anderen den Wert 1 haben, so daß die Interpolationseigenschaft von ℓ unmittelbar ersichtlich ist. Interpolationswasbitte?

Definition 5.3 (Interpolation) *Das **Interpolationsproblem** besteht darin, zu vorgegebenen Punkten $\xi_0, \ldots, \xi_n$ und Werten $y_0, \ldots, y_n$ bzw. $f(\xi_0), \ldots, f(\xi_n)$ eine[97] Funktion g zu finden, so daß*

$$g(\xi_j) = y_j \qquad \textit{bzw.} \qquad g(\xi_j) = f(\xi_j), \qquad j = 0, \ldots, n,$$

ist.

[95] Es gibt eine sehr schöne geometrische Theorie dazu, die unter dem Stichwort **baryzentrische Koordinaten** zu finden ist, siehe[10]. Und weil man Intervalle so schön einfach transformieren kann, finden sich die Parameter für Quadraturformeln in Formelsammlungen wie [1] auch nur für ein Standardintervall wie $[0, 1]$ oder $[-1, 1]$, die Variablentransformation im Integral wird dem Anwender da durchaus zugetraut.

[96] Nicht vergessen: Auch wenn wir am Ende auf eine zusammengesetzte Quadraturformel hinauswollen, so haben wir uns trotzdem gerade darauf geeinigt, uns die Sache intervallweise anzusehen.

[97] normalerweise zumindest stetige

Historisch wurde Interpolation lange Zeit dazu verwendet, tabellierte Funktionen, beispielsweise trigonometrische Funktionen oder Logarithmen, an Stellen auszuwerten, die in der Tabelle nicht aufgeführt waren. Heute besteht die Aufgabe zumeist darin, ein Objekt aus digitalen Messungen zu rekonstruieren, wobei man es gerne auch mal mit ein paar Millionen Punkten zu tun haben kann.

Die Trapezregel basiert auf der Tatsache, daß man durch zwei Punkte immer eine Gerade legen kann, was wir in unserer schönen neuen Interpolationsterminologie auch folgendermaßen formulieren können:

> Zu zwei Punkten ξ_0, ξ_1 hat das Interpolationsproblem genau eine Lösung, die ein affines Polynom ist.

Dieses Spiel geht aber weiter: So wie sich durch zwei Punkte eine Gerade legen lässt, bestimmen drei Punkte eine Parabel, vier Punkte eine kubische Parabel und so weiter. Was in allgmeiner Form wie folgt formuliert werden kann.

Satz 5.4 *Zu $n+1$ Punkten $\xi_0, \ldots, \xi_n \in \mathbb{R}$ und gegebenem f gibt es* genau ein *Polynom p vom Grad $\leq n$, so daß $p(\xi_j) = f(\xi_j)$, $j = 0, \ldots, n$.*

Beweis: Das ist so einfach, das bekommen wir hin! Für die Existenz verwenden wir wieder Polynome, die an einem ξ_j den Wert 1 haben und an allen anderen ξ_k, $k \neq j$, den Wert 0. Diese Polynome bekommen wir ganz einfach als

$$\ell_j(x) = \prod_{k \neq j} \frac{x - \xi_k}{\xi_j - \xi_k} = \frac{(x-\xi_0)\cdots(x-\xi_{j-1})\,(x-\xi_{j+1})\cdots(x-\xi_n)}{(\xi_j-\xi_0)\cdots(x_j-\xi_{j-1})\,(x_j-\xi_{j+1})\cdots(x_j-\xi_n)}$$

vom Grad n und kombinieren Sie zum Interpolationspolynom

$$p(x) = \sum_{j=0}^{n} f(\xi_j)\ \ell_j(x) = \sum_{j=0}^{n} f(\xi_j) \prod_{k \neq j} \frac{x - \xi_k}{\xi_j - \xi_k}, \tag{5.7}$$

was dann auch schon die Frage nach der Existenz beantwortet. Für die Eindeutigkeit nehmen wir an, p und q wären zwei Polynome vom Grad n, die das Interpolationsproblem lösen, aber dann ist $p - q$ ebenfalls ein Polynom vom Grad n, das an $\xi_0, \ldots, \xi_n$ verschwindet und sich daher als

$$(p-q)(x) = (x-\xi_0)\cdots(x-\xi_n)\ r(x)$$

faktorisieren lässt. Aber das Polynom auf der rechten Seite hat mindestens Grad $n+1$, was einen Widerspruch darstellt sobald $r \neq 0$, also $p \neq q$ ist. □

Die Eindeutigkeit ist besonders hilfreich, sagt sie uns doch, daß Interpolation ein Polynom **reproduziert**: Ist $f \in \Pi_n$ nämlich ein Polynom vom Grad $\leq n$ und p der Interpolant aus (5.7) dazu, dann sind f und p beides Polynome vom Grad $\leq n$, die dasselbe

Interpolationsproblem lösen[98] und müssen wegen eben dieser Eindeutigkeit identisch sein. Das zeigt uns aber schon, wie wir eine Quadraturformel bauen können, nämlich, indem wir ein Interpolationspolynom berechnen und integrieren und dann das Integral ein wenig aufräumen:

$$\begin{aligned} Q_n(f) &= \int_a^b p(x)\,dx = \int_a^b \sum_{j=0}^{n} f(\xi_j) \prod_{k\neq j} \frac{x-\xi_k}{\xi_j-\xi_k}\,dx \\ &= \sum_{j=0}^{n} f(\xi_j) \underbrace{\int_a^b \prod_{k\neq j} \frac{x-\xi_k}{\xi_j-\xi_k}\,dx}_{=w_j}, \end{aligned}$$

die Berechnung der Gewichte ist also lediglich eine Fleissaufgabe[99], aber nicht in irgendeiner Form mit konzeptionellen Schwierigkeiten verbunden.

Ein einfaches Beispiel können wir uns ansehen, nämlich die Quadraturformel zum *quadratischen* Interpolanten auf $[0,1]$ mit den Knoten $\xi_0 = 0$, $\xi_1 = \frac{1}{2}$ und $\xi_2 = 1$. Die drei Fundamentalpolynome sind dann

$$\begin{aligned} \ell_0(x) &= \frac{\left(x-\frac{1}{2}\right)(x-1)}{\left(0-\frac{1}{2}\right)(0-1)} = 2x^2-3x+1, \\ \ell_1(x) &= \frac{(x-0)(x-1)}{\left(\frac{1}{2}-0\right)\left(\frac{1}{2}-1\right)} = -4x^2+4x, \\ \ell_2(x) &= \frac{(x-0)\left(x-\frac{1}{2}\right)}{(1-0)\left(1-\frac{1}{2}\right)} = 2x^2-x, \end{aligned}$$

Die Gewichte sind also

$$\begin{aligned} w_0 &= \left[\frac{2}{3}x^3-\frac{3}{2}x^2+x\right]_0^1 = \frac{1}{6}, \\ w_1 &= \left[-\frac{4}{3}x^3+2x^2\right]_0^1 = \frac{2}{3}, \\ w_2 &= \left[\frac{2}{3}x^3-\frac{1}{2}x^2\right]_0^1 = \frac{1}{6}, \end{aligned}$$

was uns die **zusammegesetzte Simpsonformel** im einfachen Fall $x_{j+1} = x_j + h$ als

$$Q_2(f) = \sum_{j=0}^{N} Q_2^j(f) = h \sum_{j=0}^{N} w_0\, f(x_j) + w_1\, f\left(\frac{x_j+x_{j+1}}{2}\right) + w_2\, f(x_{j+1})$$

[98] f hat trivialerweise an jeder Stelle denselben Wert wie f.

[99] Die man Schülern übrigens als nette Übung für den Umgang mit einem Computeralgebrasystem stellen könnte.

$$= \frac{h}{6} f(a) + \frac{h}{3} \sum_{j=1}^{N} f(x_j) + \frac{h}{6} f(b) + \frac{2}{3} \sum_{j=0}^{N} f\left(\frac{x_j + x_{j+1}}{2}\right)$$

liefert. Und siehe da, die handgemachte Simpsonregel liefert dann auch

```
octave> Simpson( "sqr", (0:.01:1) ) .- 1/3
ans = 0
```

denn schliesslich ist diese Formel ja **exakt** für quadratische Polynome. Für den Cosinus sind wir allerdings nicht besser,

```
octave> Simpson( "cos", (0:.01:pi) )
ans =  0.0015927
```

aber das ist eigenlich auch keine Überraschung, denn eigentlich ist die Simpsonregel auch wieder nur eine Kombination aus Trapezregel und Rechtecksregel mit Mittelpunkt mit Gewichten $\frac{1}{3}$ und $\frac{2}{3}$, und kann daher nicht besser sein als die bessere von den beiden Formeln.

Quadraturformeln, die dadurch erhalten werden, daß das Intervall unter Einbeziehung der Randpunkte[100] gleichmäßig unterteil und dann mit Polynomen interpoliert wird, bezeichnet man als **Newton–Cotes–Formeln**. Für $n = 1, \ldots, 6$ sind diese in der folgenden Tabelle aufgelistet. Daß die Liste bei 6 aufhört ist im übrigen kein Zufall: Ab $n = 7$ werden einige der Gewichte negativ und die Formel ist dann nicht mehr so gut.

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

Übung 5.1 Programmieren Sie eine zusammengesetzte Pulcherrima in `Matlab/Octave` und testen Sie sie an den bisherigen Beispielen. ♢

[100] Es gibt einen einfachen Grund für die Verwendung der Randpunkte: Bei der zusammengesetzten Quadraturformel spielt jeder Randpunkt bei zwei Stücken mit und so kann man den Wert $f(x_j)$ an so einem Punkt zweimal verwenden, und das ist einfach ökonomischer – man konnt halt mit weniger Auswertungen der Funktion aus.

5.5 Wie weit können wir gehen?

Bleibt nur noch eine Frage: Wenn wir nun all die tollen Quadraturformeln haben und einen Quadraturbaukasten via Interpolation, wie viel Exaktheit kann eigentlich so eine Quadraturformel Q_n mit $n+1$ Knoten liefern? Naja, zuerst stellen wir mal fest, daß mehr als $2n+1$ auf keinen Fall gehen kann, denn für das Polynom

$$p(x) = (x-\xi_0)^2 \cdots (x-\xi_0)^2 = \prod_{j=0}^{n} (x-\xi_j)^2$$

vom Grad $2n+2$ gilt

$$\int_a^b p(x)\,dx > 0 \qquad \text{aber} \qquad Q_n f(p) = \sum_{j=0}^{n} w_j \underbrace{p(\xi_j)}_{=0} = 0.$$

Wir können also auf einen Exaktheitsgrad $\geq n$ hoffen, wenn wir die Gewichte **interpolatorisch** als

$$w_j = I(\ell_j) = \int_a^n \prod_{k\neq j} \frac{x-\xi_k}{\xi_j-\xi_k}\,dx \tag{5.8}$$

wählen, kommen aber keinesfalls über einen Exaktheitsgrad von $2n+1$ hinaus. Andererseits gilt aber für jede Quadraturformel Q_n vom Exaktheitsgrad $\geq n$, daß

$$I(\ell_j) = Q_n(\ell_j) = \sum_{k=0}^{n} w_k \underbrace{\ell_j(\xi_k)}_{=\delta_{jk}} = w_j$$

sein muss, jede Quadraturformel Q_n mit Exaktheitsgrad $\geq n$ muss also interpolatorisch sein, was die Gewichte festlegt. Fazit:

> Die einzige Möglichkeit, einen Exaktheitsgrad $\geq n$ zu bekommen, ist die geeignete Wahl der Knoten $\xi_0, \ldots, \xi_n$.

Und tatsächlich kann man die Knoten so wählen, daß man einen Exaktheitsgrad von $2n+1$ erhält. Wie das geht, wurde 1816 von Gauß gezeigt [3]. Tatsächlich müssen die Knoten Nullstellen eines orthogonalen Polynoms vom Grad $n+1$ sein, also eines Polynoms $p \neq 0$ so daß

$$\int_a^b p(x)\,q(x)\,dx = 0, \qquad q \in \Pi_n$$

erfüllt ist. Sowas gibt es immer, sowas hat immer $n+1$ einfache reelle Nullstellen im Intervall, und liefert immer positive Gewichte – die **Gauß–Quadratur** lässt tatsächlich keine Wünsche offen.

Und das erklärt zum Abschluss auch, warum die Rechtecksregel mit Mittelpunktswahl so erfolgreich war: Die Intervallmitte ist die Nullstelle der affinen Funktion ℓ mit Integral 0, für die also

$$0 = \int_a^b \ell(x)\,dx = \int_a^b \ell(x)\,q(x)\,dx, \qquad q \in \Pi_0 = \mathbb{R},$$

erfüllt ist. Richtig: ℓ ist das orthogonale Polynom vom Grad 1 und die Mittelpunktsregel ist eben die Gauß-Formel vom Grad 0, die damit nicht Exaktheitsgrad 0 sondern das maximale $2 \cdot 0 + 1 = 1$ hat.

5 Literatur

[1] M. Abramowitz and I. A. Stegun (eds.), *Handbook of mathematical functions*, Dover, 1972, 10th printing.

[2] J. von zur Gathen and J. Gerhard, *Modern computer algebra*, Cambridge University Press, 1999.

[3] C. F. Gauss, *Methodus nova integralium valores per approximationem inveniendi*, Commentationes societate regiae scientiarum Gottingensis recentiores **III** (1816).

[4] R. Gillings, *Mathematics in the time of the Pharaos*, Messachusetts Institute of Technology, 1972, Dover reprint, 1982.

[5] G. Golub and C. F. van Loan, *Matrix computations*, 3rd ed., The Johns Hopkins University Press, 1996.

[6] H. Heuser, *Lehrbuch der Analysis. Teil 1*, 3. ed., B. G. Teubner, 1984.

[7] N. J. Higham, *Accuracy and stability of numerical algorithms*, 2nd ed., SIAM, 2002.

[8] G. Ifrah, *Universalgeschichte der zahlen*, Campus Verlag, Frankfurt a. Main, New York, 1987.

[9] E. Isaacson and H. B. Keller, *Analysis of Numerical Methods*, John Wiley & Sons, 1966.

[10] A. F. Möbius, *Der barycentrische Calcul*, Johann Ambrosius Barth, 1827.

[11] W. Popp, *Wege des exakten denkens. vier jahrtausende mathematik*, Weltbild Verlag, 1987, Originalausgabe Franz Ehrenwirth Verlag, 1981.

[12] 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.

[13] T. Sauer, *Numerische Mathematik I*, Vorlesungsskript, Friedrich–Alexander–Universität Erlangen–Nürnberg, Justus–Liebig–Universität Gießen, 2000, `http://www.math.uni-giessen.de/tomas.sauer`.

[14] _____, *Numerische Mathematik II*, Vorlesungsskript, Friedrich–Alexander–Universität Erlangen–Nürnberg, Justus–Liebig–Universität Gießen, 2000, `http://www.math.uni-giessen.de/tomas.sauer`.

[15] _____, *Computeralgebra*, Vorlesungsskript, Justus–Liebig–Universität Gießen, 2001, `http://www.math.uni-giessen.de/tomas.sauer`.

[16] E. Schwartz (ed.), *Homer, ilias*, Weltbild Verlag, 1994, Zweisprachige Ausgabe, Altgriechisch–Deutsch.

5 Index