

Vorlesung 1

Einleitung

1.1 Grundlegende Aufgaben der Numerik

Die numerische Mathematik beschäftigt sich mit der Berechnung mathematischer Ausdrücke, wie z. B. $x = \sqrt{2}$ oder $I = \int_0^1 e^{z^2} dz$, dem Auflösen von Gleichungen, z.B. $x^2 e^x = 3$ oder dem Lösen von Differentialgleichungen, etwa z. B. $\frac{dy(t)}{dt} = ty(t)$, mit einem Startwert $y(1) = 1$. Charakteristisch ist dabei die Lösung dieser Aufgaben für *für konkrete Zahlenwerte*. Dies ist immer dann sinnvoll, wenn die Lösung einer Gleichung nicht in geschlossener Form darstellbar ist.

Selbst wenn so eine Darstellung möglich ist, enthält sie oft nichtlineare Funktionen, deren Werte nur mit einer Rechenmaschine (näherungsweise) berechenbar sind (oder früher in Tabellenwerken nachgeschlagen werden mussten). Generell sind numerische Berechnungen mit einer Reihe von *Fehlern* behaftet deren Abschätzung und Kontrolle einen zentralen Aspekt der *numerischen Analysis* darstellt.

Als erstes konkretes Beispiel betrachte den Ausdruck $x = \sqrt{2}$. Die Quadratwurzel $\sqrt{2}$ ist ein Symbol. Es definiert x als positive Lösung der quadratischen Gleichung $x^2 = 2$. Dieser Wert ist mit endlich vielen Grundoperationen $+, -, \cdot, /$ sowie Zahlen mit endlich vielen Ziffern nicht berechenbar. Stattdessen kann man nur eine Folge angeben, deren Glieder elementar berechenbar sind und die gegen $\sqrt{2}$ konvergiert. Ein Beispiel ist

$$x_{n+1} = \frac{1}{2} \left(x_n + \frac{2}{x_n} \right), \quad x_0 = 1.$$

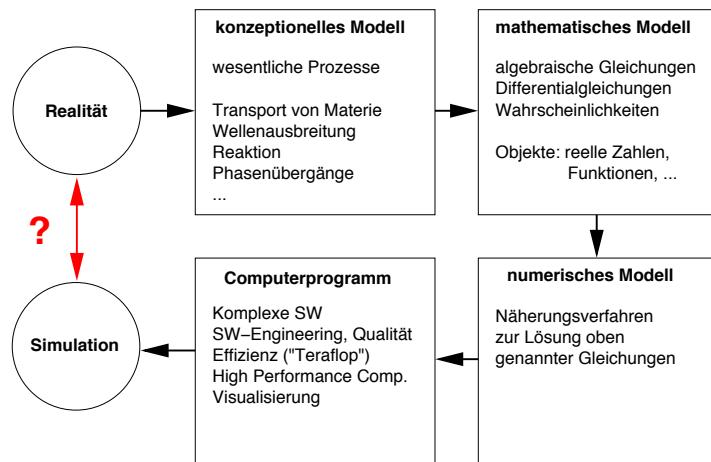
Diese Iteration erhält man durch Anwendung des Newton-Verfahrens auf die Gleichung $f(x) = x^2 - 2 = 0$. Die Folgenglieder x_n sind mit elementaren Operationen berechenbar. Die numerische Mathematik untersucht dann unter anderem folgende Fragestellungen:

- 1) Wie leitet man systematisch Iterationsverfahren zur Lösung von Gleichungen der Form $f(x) = 0$ her?
- 2) Unter welchen Bedingungen konvergiert so ein Iterationsverfahren?
- 3) Wie schnell konvergiert so ein Iterationsverfahren?
- 4) Wann kann man die Iteration abbrechen?
- 5) Welchen Einfluss hat die endliche Zahlendarstellung im Computer?

1.2 Wissenschaftliches Rechnen

Die numerische Mathematik ist in ein größeres Fachgebiet, dem *Wissenschaftlichen Rechnen*, eingebettet. Das Wissenschaftliche Rechnen verbindet Theorie

und Experiment, die klassischen Säulen der wissenschaftlichen Methode. Mit Hilfe einer Theorie bzw. eines Modells, in Form mathematischer Gleichungen, will man dabei eine Beobachtung, gegeben durch Messungen, erklären. Oft sind Modelle z.B. in der Form von Differentialgleichungen gegeben, welche nur sehr eingeschränkt in geschlossener Form lösbar sind. Als Beispiel dient das Phänomen des Klimawandels. Hier möchte man den Einfluss der Konzentration von Treibhausgasen auf die Temperatur der Atmosphäre beurteilen. In den vergangenen Jahrzehnten hat das Wissenschaftliche Rechnen immer mehr an Einfluss gewonnen, da es undurchführbare Experimente ermöglicht (z.B. Galaxiensimulation, Klimawandel), teure Experimente einspart (z.B. Windkanalexperimente) oder die Abschätzung von Unsicherheiten ermöglicht (z.B. atomares Endlager). Generell geht man im Wissenschaftlichen Rechnen wie folgt vor:



Im *konzeptionellen Modell* klärt man welche Prozesse zur Modellierung der Beobachtung relevant sind und welche Annahmen man trifft. Diese Entscheidung ist in der Regel nicht eindeutig und in Folge ergeben sich Modelle unterschiedlicher Aussagekraft und Genauigkeit. Ausgehend vom konzeptionellen Modell erstellt man ein *mathematisches Modell* in der Form von Gleichungen. Sehr oft spielen auch Unsicherheiten eine Rolle (wie etwa beim Klimawandel) und man zieht Methoden der Wahrscheinlichkeitsrechnung heran (dies werden wir aber in dieser Vorlesung nicht behandeln). Liegt das Modell fest tritt die eigentliche Numerik auf den Plan um mit Hilfe von Näherungsverfahren die mathematischen Gleichungen zu Lösen, man spricht vom *numerischen Modell*. Schließlich ist noch das Fachgebiet *Informatik* wesentlich beteiligt, da die Berechnungen teils einen sehr hohen Aufwand erfordern, der nur von parallelen Supercomputern bereit gestellt wird. Viele Probleme und Verfahren erfordern auch einen hohen Softwareentwicklungsaufwand. Schließlich liefert die Modellbildung und Simulation Zahlenwerte welche mit den Messungen verglichen werden können. In der Regel stimmen die berechneten Werte nicht exakt mit den Messungen überein und dann stellen sich mindestens folgende Fragen:

- 1) Welche Unterschiede zwischen Messung und Simulation sind akzeptabel?
- 2) Was sind die Ursachen für diese Unterschiede?
- 3) Wie können diese Unterschiede reduziert werden?

Zum zweiten Punkt sollen folgende Fehlerquellen genannt werden:

- a) *Modellfehler*: Ein relevanter Prozess wurde nicht oder ungenau modelliert (z.B. Temperatur konstant, Luftwiderstand vernachlässigt, ...).
- b) *Datenfehler*: Messungen von Anfangsbedingungen, Randbedingungen, Werte für Parameter sind fehlerbehaftet.
- c) *Abschneidefehler*: Abbruch von Reihen oder Iterationsverfahren, Approximation von Funktionen (z.B. stückweise Polynome), diskreter statt kontinuierlicher Zeitverlauf.
- d) *Rundungsfehler*: Reelle Zahlen werden im Rechner genähert dargestellt.

Mit den beiden letztgenannten Fehlerquellen werden wir uns in der Numerik ausführlich beschäftigen.

1.3 Modellierung der Ausbreitung des Coronavirus

Als Beispiel für die Untersuchung eines Prozesses mit numerischen Methoden betrachten wir die Ausbreitung des Coronavirus.

Ein Wachstumsmodell

Die Funktion $N : [t_0, \infty) \rightarrow \mathbb{N} \cup \{0\}$ beschreibe die Anzahl der infizierten Personen zur Zeit $t \geq t_0$. Wie könnte so eine Funktion aussehen? Als ersten Abstraktionsschritt beschreiben wir $N(t)$ durch eine Funktion $u(t)$ welche Werte aus den reellen Zahlen $u(t) \in \mathbb{R}$ annehmen darf. Dies erscheint zunächst ungewöhnlich, da ja eine Person entweder infiziert ist oder nicht. Der Vorteil dieser Verallgemeinerung wird sein, dass wir damit eine bestimmte Klasse von mächtigen mathematischen Modellen nutzen können, sogenannte *Differentialgleichungen*. Für hinreichend große Zahlen $N(t)$ wird die Näherung mit nichtganzen Zahlen nur einen kleinen (relativen) Fehler mit sich bringen.

Zur Modellierung der zeitlichen Entwicklung der Zahl der infizierten Personen machen wir die folgende Annahme: *Die Änderung der Zahl der infizierten Personen im Zeitintervall $[t, t + \Delta t]$ sei proportional zur Zahl der infizierten Personen zur Zeit t und zur Länge des Zeitintervalls Δt .* In Formeln:

$$u(t + \Delta t) = u(t) + \lambda \Delta t u(t). \quad (1.1)$$

Diese Annahme ist sinnvoll solange unbegrenzte Ressourcen für das Wachstum zur Verfügung stehen, z.B. kann jede infizierte Person stets neue, nicht infizierte Personen treffen. Die Konstante $\lambda \in \mathbb{R}$ beschreibt die Wachstumsrate. Umstellen von Gleichung (1.1) liefert

$$\frac{u(t + \Delta t) - u(t)}{\Delta t} = \lambda u(t)$$

und schließlich liefert der Grenzübergang $\Delta t \rightarrow 0$ die lineare, gewöhnliche Differentialgleichung erster Ordnung

$$\frac{du}{dt}(t) = \lambda u(t). \quad (1.2)$$

Man kann zeigen, dass alle Lösungen dieser Gleichung die Form

$$u(t) = ce^{\lambda t}$$

mit einer Konstanten $c \in \mathbb{R}$ besitzen. Die Konstante c wird in der Regel durch einen Anfangswert $u(t_0) = u_0$ festgelegt. Dann spricht man von einer *Anfangswertaufgabe* deren Eigenschaften der folgende Satz zusammenfasst.

Satz 1.1. Die Anfangswertaufgabe

$$\frac{du}{dt}(t) = \lambda u(t), \quad u(t_0) = u_0, \quad (1.3)$$

besitzt die Lösung

$$u(t) = u_0 e^{\lambda(t-t_0)}. \quad (1.4)$$

Beweis. Durch einsetzen und $u(t_0) = u_0 e^{\lambda t_0} = u_0$. \square

Das Modell (1.3) ist ein sehr einfaches Modell zur Beschreibung von Wachstumsprozessen welches viele Effekte, z.B. räumliche Abhängigkeiten, vernachlässigt. Trotzdem beschreibt es z.B. erstaunlich gut das Wachstum der Weltbevölkerung im Zeitraum 1700-1961, siehe Braun [1994]. Wir wollen nun dieses Modell zur Beschreibung der Ausbreitung des Coronavirus einsetzen und z.B. der Frage nachgehen wann und ob die zur Eindämmung der Ausbreitung getroffenen Maßnahmen greifen. Auf dem Weg werden wir einige grundlegende Aufgaben und Methoden der Numerik kennenlernen, welche im Rest der Vorlesung ausführlich behandelt werden.

Die Datenlage

Abbildung 1.1 zeigt die Daten der täglichen Entwicklung der mit dem Coronavirus infizierten Personen in Deutschland ab dem 24. Februar 2020, also

$$N_i = N(t_i), \quad i = 0, \dots, m-1,$$

wobei die Zeiteinheit als ‘‘Tag’’ gewählt wurde, d.h. $t_i = i$. Die obere Kurve ‘‘sieht nach einem exponentiellen Verlauf’’ aus, aber ist das tatsächlich der Fall?

Würde der Verlauf dem Gesetz (1.4) folgen, so erhält man für die Logarithmen bei $t_i = i$:

$$\ln u_i = \ln u(t_i) = \ln u_0 + \lambda i,$$

d.h. der (natürliche) Logarithmus der Fallzahlen sollte sich wie eine Gerade verhalten. In der unteren Grafik in Abbildung 1.1 sieht man dass dies nur eingeschränkt der Fall ist.

Um die Daten mit der Bibliothek HDNum zu analysieren müssen wir diese zunächst eingeben. Das gelingt mit den folgenden Anweisungen:

```
using Number = double;
using Vec = Vector<Number>

// Corona Fälle Deutschland ab dem 24.2.2020
// https://de.wikipedia.org/wiki/COVID-19-Pandemie_in_Deutschland
Vec N = {16, 18, 21, 26, 53, 66, 117, 150, 188, 240,
400, 639, 795, 902, 1139, 1296, 1567, 2369,
3062, 3795, 4838, 6012, 7156, 8198, 10999, 13957,
16662, 18610, 22672, 27436, 31554, 36508, 42288,
48582, 52547, 57298, 61913, 67366, 73522, 79696, 85778, 91714, 95391,
99225, 103228};
Vec t(N.size());
fill(t, 0.0, 1.0); // Zeitpunkt in Tagen
vector<string> tstring = {"24/02/2020", "25/02/2020", "26/02/2020",
```

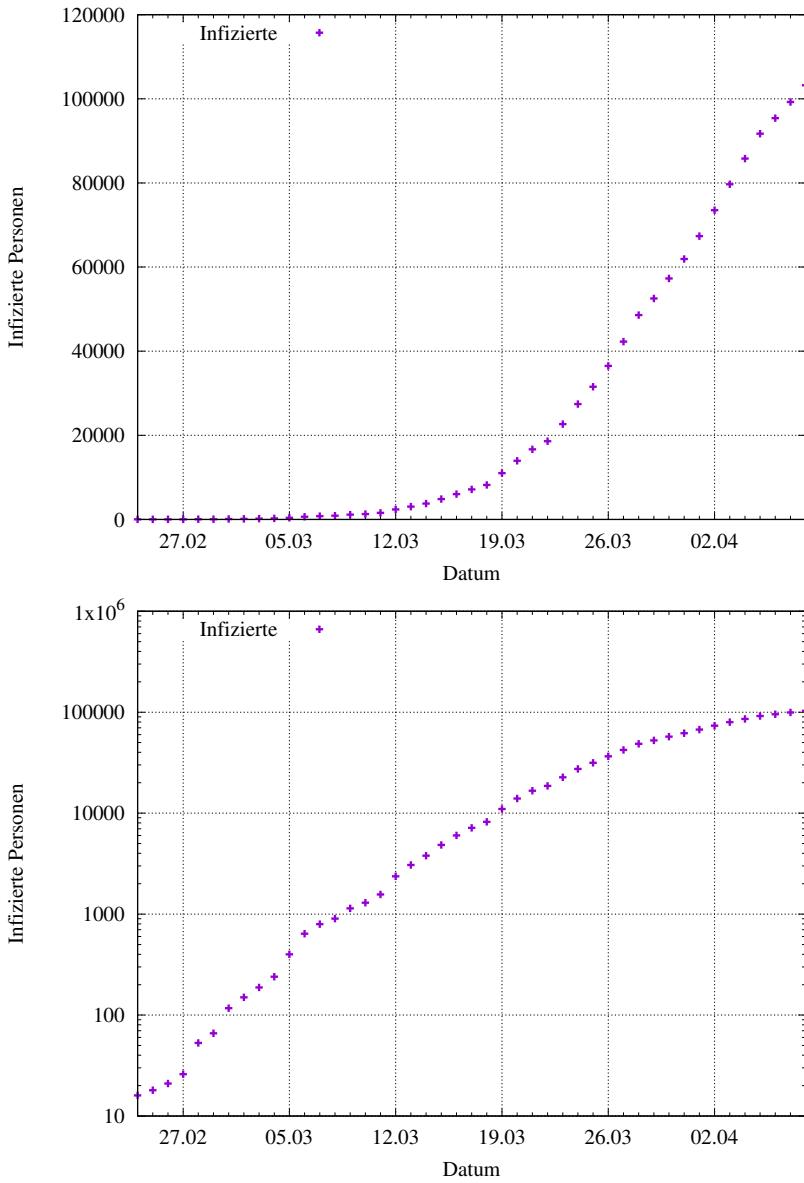


Abbildung 1.1: Tägliche Zahl der mit dem Coronavirus infizierten Personen ab dem 24. Februar 2020. Quelle: Wikipedia https://de.wikipedia.org/wiki/COVID-19-Pandemie_in_Deutschland. Oben in absoluter Darstellung, unten in logarithmischer Darstellung.

```

    "27/02/2020", "28/02/2020", "29/02/2020", "01/03/2020", "02/03/2020",
    "03/03/2020", "04/03/2020", "05/03/2020", "06/03/2020", "07/03/2020",
    "08/03/2020", "09/03/2020", "10/03/2020", "11/03/2020", "12/03/2020",
    "13/03/2020", "14/03/2020", "15/03/2020", "16/03/2020", "17/03/2020",
    "18/03/2020", "19/03/2020", "20/03/2020", "21/03/2020", "22/03/2020",
    "23/03/2020", "24/03/2020", "25/03/2020", "26/03/2020", "27/03/2020",
    "28/03/2020", "29/03/2020", "30/03/2020", "31/03/2020", "01/04/2020",
    "02/04/2020", "03/04/2020", "04/04/2020", "05/04/2020", "06/04/2020",
    "07/04/2020", "08/04/2020"
}; // Datum zur Ausgabe
gnuplot("N.dat", tstring, N); // Ausgabe der Daten

```

Vorhersage und Parameterschätzung

Ein Anwendungsszenario von Modellbildung und Simulation ist die Vorhersage. So möchte man etwa die Zahl der mit dem Coronavirus infizierten Personen vorhersagen um sich auf Maßnahmen vorbereiten zu können. Auch beim Klimawandel spielt die Vorhersage verschiedener Szenarien eine große Rolle.

Voraussetzung für eine Vorhersage ist die Kenntnis von *Parametern* des Modells. In unserem Modell (1.4) sind die Parameter der Anfangswert u_0 und die Wachstumsrate λ . Hingegen ist die Startzeit t_0 in der Regel bekannt. Die übliche Vorgehensweise bei der Vorhersage besteht daher aus zwei Schritten:

- 1) *Parameterschätzung*: Bestimmung der Parameter des Modells aus der Literatur oder gemessenen Daten.
- 2) *Auswertung*: Berechnung des Modells mit den bestimmten Parametern.

In unserem Fall liegt das Modell in geschlossener (analytischer) Form vor. Die Auswertung erfordert daher nur die Berechnung der Formel (1.4). Hier werden wir später die Auswirkung von *Rundungsfehlern* näher betrachten. Wenden wir uns daher vor allem dem Parameterschätzproblem zu.

Interpolation Ein einfacher Fall liegt vor wenn genau zwei Datenpunkte (t_0, N_0) , (t_i, N_i) zur Verfügung stehen. Dann können die Parameter u_0 und λ einfach berechnet werden:

Erweitern Sie die Herleitung auf den Fall $u(t_i) = N_i$,

$u(t_j) = N_j$, $j \neq i$.

$$\begin{aligned} \frac{u_0 e^{\lambda t_0}}{u_0 e^{\lambda(t_i-t_0)}} &= \frac{N_0}{N_i} \end{aligned} \Rightarrow u_0 = N_0, \lambda = \frac{\ln N_i - \ln N_0}{t_i - t_0}. \quad (1.5)$$

Offensichtlich muss $N_0 \neq 0$, $N_i \neq 0$ und $t_i \neq t_0$ gelten. Da die so bestimmte Funktion $u(t)$ die Datenpunkte (t_0, N_0) , (t_i, N_i) exakt reproduziert spricht man von *Interpolation*. Folgende Funktion realisiert die Interpolation in HDNum:

```

Vec interpolation (const Vec& N, const Vec& t, int i)
{
    Vec p(2); // Ergebnis hat zwei Komponenten
    p[0] = N[0]; // u_0
    p[1] = (log(N[i])-log(N[0]))/(t[i]-t[0]); // λ
    return p;
}

```

Ausgleichsgerade Ein anderer einfacher Fall liegt vor, wenn wir versuchen die Logarithmen zu interpolieren. Dies führt auf das Gleichungssystem

$$\ln u_0 + \lambda(t_i - t_0) = \ln N_i \quad i = 0, \dots, m-1.$$

Betrachten wir statt u_0 den Logarithmus $\ln u_0$ als Unbekannte, stellt dies ein lineares Gleichungssystem dar:

$$\begin{bmatrix} 1 & 0 \\ 1 & t_1 - t_0 \\ \vdots & \vdots \\ 1 & t_{m-1} - t_0 \end{bmatrix} \begin{bmatrix} \ln u_0 \\ \lambda \end{bmatrix} = \begin{bmatrix} \ln N_0 \\ \ln N_1 \\ \vdots \\ \ln N_{m-1} \end{bmatrix} \Leftrightarrow Ax = b. \quad (1.6)$$

Für $m > 2$ ist dieses Gleichungssystem allerdings überbestimmt und wir können in der Regel nicht erwarten, dass es eine Lösung besitzt. Statt dessen kann man versuchen die Lösung "so gut wie möglich" zu machen, indem man die Größe

$$J_{\text{lin}}(x) = \|Ax - b\|^2 = \sum_{i=0}^{m-1} (\ln u_0 + (t_i - t_0)\lambda - \ln N_i)^2 \quad (1.7)$$

zu minimieren. Dieses Verfahren wird auch *Gauß'sche Ausgleichsrechnung* genannt. Die resultierende Gerade $\ln u_0 + t\lambda$ bezeichnet man als *Ausgleichsgerade*. Sie approximiert die Logarithmen der Fallzahlen durch eine Gerade. Wir werden später zeigen, dass man die Lösung des Minimierungsproblems durch *Lösung* des folgenden Gleichungssystems erhält:

$$A^T Ax = A^T b. \quad (1.8)$$

In HDNum lässt sich das wie folgt realisieren:

```
Vec linear_fit (const Vec& N, const Vec& t)
{
    using Mat = DenseMatrix<Number>; // Matrixklasse
    Mat A(N.size(), 2); // Erstelle die Matrix A
    for (int i=0; i<N.size(); ++i) {
        A[i][0] = 1.0; A[i][1] = t[i] - t[0];
    }

    // Erstelle die rechte Seite
    Vec lnN(N.size());
    for (int i=0; i<N.size(); ++i) logN[i] = ln(N[i]);

    // berechne Ausgleichsgerade
    Mat AT(A.transpose()); // transponieren
    Mat ATA(2, 2); ATA.mm(AT, A); // A^T A
    Vec b(2);
    AT.mv(b, lnN); // b = A^T ln N
    Vec p(2);
    linsolve(ATA, p, b);
    return p;
}
```

Wiederholen Sie die Lösbarkeit linearer Gleichungssystem aus der Vorlesung Lineare Algebra.

Nichtlineares Parameterschätzproblem Statt den Logarithmen kann man auch versuchen die Datenpunkte direkt zu interpolieren. D.h. wir setzen an:

$$u(t_i) = u_0 e^{\lambda(t_i - t_0)} = N_i \quad i = 0, \dots, m-1.$$

Für $m > 2$ führt dies auf ein überbestimmtes *nichtlineares* Gleichungssystem. Analog zum linearen Fall oben, kann man hier die Größe

$$J_{\text{nl}}(u_0, \lambda) = \sum_{i=0}^{m-1} (u_0 e^{\lambda(t_i - t_0)} - N_i)^2 \quad (1.9)$$

minimieren und so die Parameter u_0 und λ aus m Datenpunkten bestimmen. Eine notwendige Bedingung für ein Minimum ist das Verschwinden der ersten partiellen Ableitungen von J_{nl} nach den Parametern:

$$F(u_0, \lambda) = \begin{bmatrix} \frac{\partial J_{\text{nl}}}{\partial u_0}(u_0, \lambda) \\ \frac{\partial J_{\text{nl}}}{\partial \lambda}(u_0, \lambda) \end{bmatrix} = \begin{bmatrix} \sum_{i=0}^{m-1} [(u_0 e^{\lambda(t_i - t_0)} - N_i) e^{\lambda(t_i - t_0)}] \\ \sum_{i=0}^{m-1} [(u_0 e^{\lambda(t_i - t_0)} - N_i)(t_i - t_0) e^{\lambda(t_i - t_0)}] \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

Das Gleichungssystem

$$F(x) = 0, \quad x = (u_0, \lambda)^T,$$

kann mit dem Newton-Verfahren iterativ gelöst werden. Dieses lautet

$$x^{k+1} = x^k - \eta_k H(x^k) F(x^k) \quad (1.10)$$

mit der Matrix

$$H(u_0, \lambda) = \begin{bmatrix} \frac{\partial F_0}{\partial u_0}(u_0, \lambda) & \frac{\partial F_0}{\partial \lambda}(u_0, \lambda) \\ \frac{\partial F_1}{\partial u_0}(u_0, \lambda) & \frac{\partial F_1}{\partial \lambda}(u_0, \lambda) \end{bmatrix} = \begin{bmatrix} \frac{\partial^2 J_{\text{nl}}}{\partial u_0^2}(u_0, \lambda) & \frac{\partial^2 J_{\text{nl}}}{\partial u_0 \partial \lambda}(u_0, \lambda) \\ \frac{\partial^2 J_{\text{nl}}}{\partial \lambda \partial u_0}(u_0, \lambda) & \frac{\partial^2 J_{\text{nl}}}{\partial \lambda^2}(u_0, \lambda) \end{bmatrix}.$$

H heisst *Jacobimatrix von F* bzw. *Hessematrix von J_{nl}* . Man benötigt also im Newtonverfahren insgesamt zweite Ableitungen von J_{nl} . Die Bereitstellung von Ableitungen kann bei komplexeren Modellen aufwändig werden. Daher werden in der Praxis auch *numerisches und automatisches Differenzieren* eingesetzt. Die Größe η_k in der Iteration (1.10) ist ein geeignet gewählter *Dämpfungsfaktor*, der für die Konvergenz des Verfahrens wichtig sein kann. Ebenfalls wichtig für die Konvergenz des Verfahrens ist ein geeignet gewählter *Startwert x^0* .

In HDNum lässt sich das Newtonverfahren wie folgt realisieren:

```
Vec nonlinear_fit (const Vec& N, const Vec& t, const Vec& p0)
{
    auto F = [&] (const Vec& p) // eine Lambdafunktion
    {
        Vec r(2, 0.0);
        for (int i=0; i<N.size(); ++i) {
            r[0] += (p[0]*exp(p[1]*(t[i]-t[0]))-N[i])
                     *exp(p[1]*(t[i]-t[0]));
            r[1] += (p[0]*exp(p[1]*(t[i]-t[0]))-N[i])*(t[i]-t[0])
                     *exp(p[1]*(t[i]-t[0]));
        }
        return r;
    };
    auto nlp = getNonlinearProblem(F, p0);
```

```

Newton newton;           // Ein Newtonobjekt
newton.set_maxit(50);    // Setze diverse Parameter
newton.set_verbosity(2);
newton.set_reduction(1e-10);
newton.set_abslimit(1e-100);
newton.set_linesearchsteps(8);

Vec p(p0);              // die Loesung
newton.solve(nlp,p);    // Berechne Loesung
return p;
}
  
```

Die Jacobimatrix von F wird in dieser Implementierung numerisch berechnet.

Abbildung 1.2 zeigt das Ergebnis der Vorhersage für die drei verschiedenen Parameterschätzmethoden. Dabei wurden die Fallzahlen für die letzten fünf Tage aus allen vorherigen Daten mit den drei verschiedenen Methoden vorher gesagt. Die Ausgleichsgerade überschätzt die Fallzahlen deutlich. Die Interpolation der Daten vom 24. Februar und sechstletzten Tag liegt schon besser und am besten liegt man mit der Lösung des nichtlinearen Parameterschätzproblems. Deutlich kann man auch die gute Annäherung der Datenpunkte vor dem Vorhersagezeitraum mit dieser Methode erkennen. Allerdings überschätzt auch die beste Methode die realen Daten deutlich. Dies liegt daran, dass die am 22. März getroffenen Eindämmungsmaßnahmen Wirkung zeigen. Das Modell mit einer konstanten Wachstumsrate λ ist kein gutes Modell für diesen Fall. Eine Verbesserung stellt eine zeitlich variable Wachstumsrate $\lambda(t)$ dar.

Können Sie dieses Ergebnis erklären?

Analyse der Verdopplungszeiten

Ziel der am 22. März veranlassten Maßnahmen war es die Ausbreitungs geschwindigkeit des Virus zu verlangsamen. Als Maß für die Ausbreitungs geschwindigkeit ziehen wir die *Verdopplungszeit*, d.h. die Zeitspanne innerhalb der sich die Anzahl der Infizierten Personen verdoppelt, heran. Aus unserem Modell erhalten wir die Verdopplungszeit mittels

$$u_0 e^{\lambda T_2} = 2u_0 \Rightarrow T_2 = \frac{\ln 2}{\lambda}. \quad (1.11)$$

Zur Analyse der Wirkung der Eindämmungsmaßnahmen postulieren wir dass λ (und damit die Verdopplungszeit) mit der Zeit variiert. Über einen Zeitraum weniger Tage können wir jedoch annehmen, dass λ annähernd konstant ist. Die Idee ist nun, zum Zeitpunkt t_k die Wachstumsrate $\lambda(t_k)$ aus den w Datenpunkten $(t_{k-w-1}, N_{k-w-1}), \dots, (t_k, N_k)$ mit Hilfe der oben beschriebenen nichtlinearen Parameterschätzung zu bestimmen. w bezeichnet man dabei als *Fenstergröße*. Abbildung 1.3 zeigt die mit dieser Analyse berechneten Verdopplungszeiten für zwei Fenstergrößen. Innerhalb einer Woche nach Einleitung der Maßnahmen ist die Verdopplungszeit von etwa 4 auf etwa 7-8 angestiegen. Dabei beobachten wir: größere Fenstergröße führt zu einem glatteren Verlauf, allerdings auch zu absolut niedrigeren Zahlen.

1.4 Zusammenfassung

Numerik ist ein wesentliches Teilgebiet des Wissenschaftlichen Rechnens. Mittels mathematischer Modelle und deren numerischer Simulation lassen sich las-

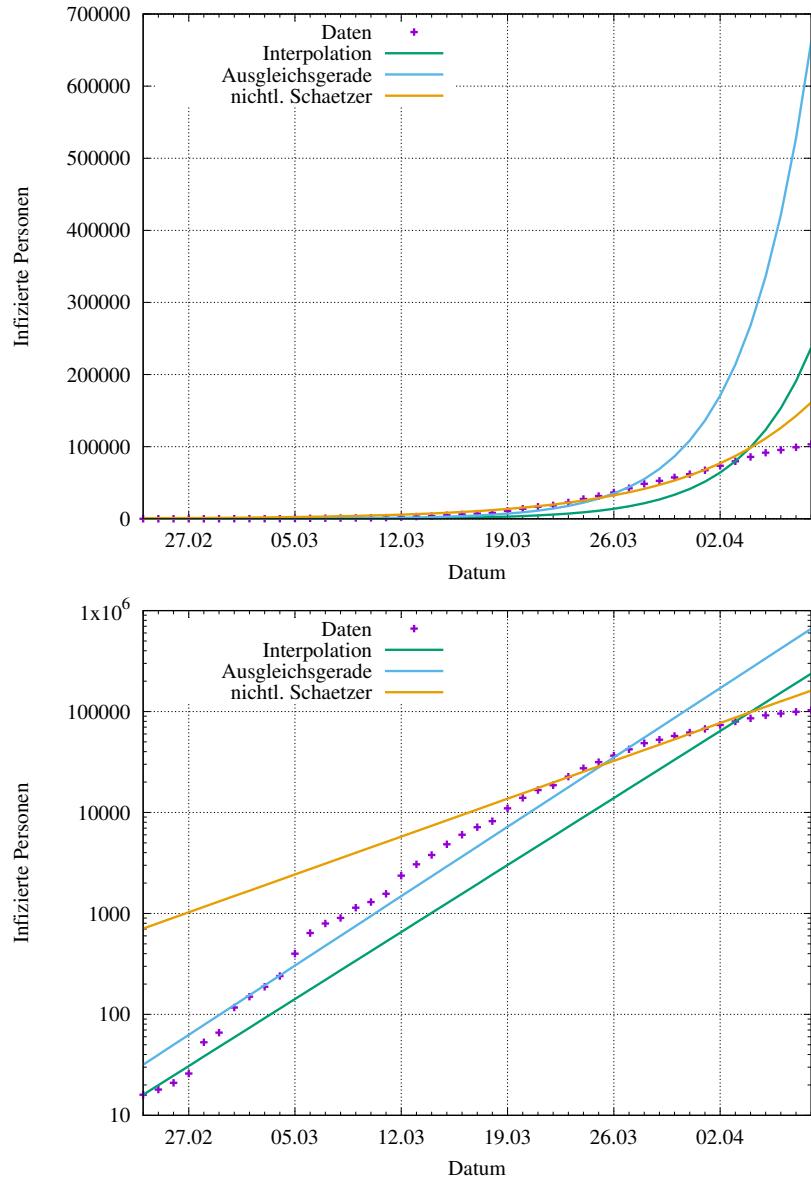


Abbildung 1.2: Vorhersage durch die drei verschiedene Parameterschätzmethoden in normaler und logarithmischer Darstellung. Die Daten bis zum sechstletzten Tag werden benutzt um die Fallzahlen für die letzten fünf Tage vorherzusagen.

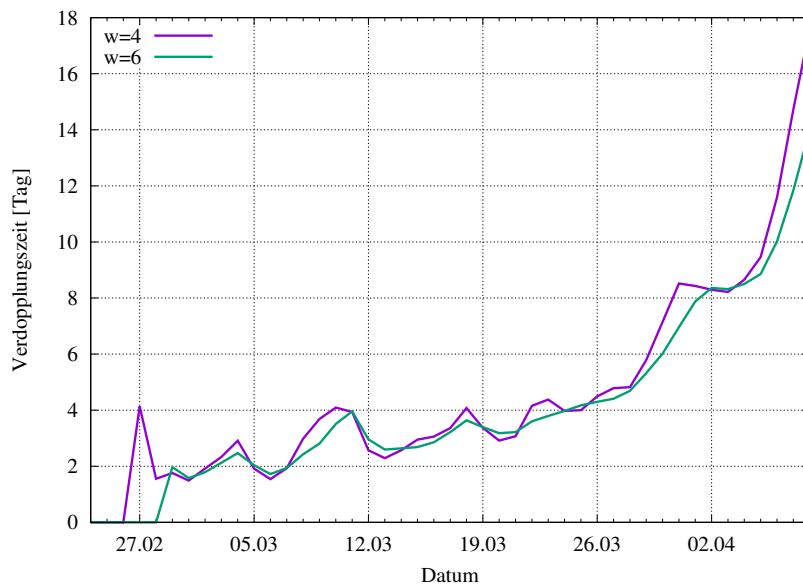


Abbildung 1.3: Verdopplungszeiten für verschiedene Fenstergrößen w .

sen sich Hypothesen testen und Beobachtungen erklären. Am Beispiel der Ausbreitung des Coronavirus wurden Parameterschätzprobleme eingeführt, welche auf lineare bzw. nichtlineare Gleichungssysteme führen. Gauß'sche Ausgleichsrechnung und Newtonverfahren wurden als Lösungsverfahren kurz vorgestellt.

Vorlesung 2

Fließkommadarstellung

2.1 Einführendes Beispiel

Alle Programmiersprachen stellen elementare Datentypen zur Repräsentation von Zahlen zur Verfügung. Diese lauten mit ihrer mathematischen Entsprechung wie folgt:

Datentyp	Mathematik
<code>unsigned int, unsigned short, unsigned long</code>	\mathbb{N}_0
<code>int, short, long</code>	\mathbb{Z}
<code>float, double</code>	\mathbb{R}
<code>complex<float>, complex<double></code>	\mathbb{C}

Natürlich sind diese Datentypen Idealisierungen der Zahlenmengen $\mathbb{N}_0, \mathbb{Z}, \mathbb{R}, \mathbb{C}$, da diese jeweils unendliche viele Elemente enthalten.

Bei `unsigned int, int ...` besteht die Idealisierung darin, dass es eine größte (bzw. kleinste) darstellbare Zahl gibt. Solange man diesen Bereich nicht überschreitet sind die Ergebnisse exakt.

Bei `float` und `double` kommt hinzu, dass die meisten innerhalb des darstellbaren Bereichs liegenden Zahlen nur *näherungsweise* dargestellt werden können. Wir betrachten in einem Beispiel welche Auswirkungen diese Näherung auf das Ergebnis einer Berechnung haben kann.

Beispiel 2.1. Die Funktion e^x lässt sich mit einer Potenzreihe berechnen

$$e^x = 1 + \sum_{i=1}^{\infty} \frac{x^i}{i!}.$$

Damit ist die Partialsumme $S_n = 1 + \sum_{i=1}^n \frac{x^i}{i!}$ eine Näherung. Praktisch erfolgt die Berechnung mit der Rekursionformel

$$y_n = \frac{x}{n} y_{n-1}, \quad S_n = S_{n-1} + y_n$$

und dem Rekursionsanfang

$$y_1 = x, \quad S_1 = 1 + y_1.$$

Wir untersuchen nun diese Rekursion für verschiedene Genauigkeiten und Werte von x .

Für den Wert $x = 1$ und `float`-Genauigkeit erhalten wir:

<code>1.000000000000000e+00</code>	<code>1</code>	<code>2.000000000000000e+00</code>
<code>5.000000000000000e-01</code>	<code>2</code>	<code>2.500000000000000e+00</code>
<code>1.666666716337204e-01</code>	<code>3</code>	<code>2.666666746139526e+00</code>
<code>4.166666790843010e-02</code>	<code>4</code>	<code>2.708333492279053e+00</code>
<code>8.33333767950535e-03</code>	<code>5</code>	<code>2.716666936874390e+00</code>
<code>1.388888922519982e-03</code>	<code>6</code>	<code>2.718055725097656e+00</code>

1.984127011382952e-04	7	2.718254089355469e+00
2.480158764228690e-05	8	2.718278884887695e+00
2.755731884462875e-06	9	2.718281745910645e+00
2.755731998149713e-07	10	2.718281984329224e+00
0.000000000000000e+00	100	2.718281984329224e+00
	ex	2.718281828459045e+00

Dabei stehen in der ersten Spalte die Summanden $y_n = x^n/n!$, in der zweiten Spalte der Index n und in der dritten Spalte die Größe S_n als Näherung von e^x . Zwischen $n = 10$ und $n = 100$ wurden die Werte der Übersichtlichkeit wegen weggelassen. Der Vergleich mit dem exakten Wert ergibt eine Genauigkeit von 7 Ziffern.

Für $x = 5$ erhält man entsprechend

9.333108209830243e-06	21	1.484131774902344e+02
	ex	1.484131591025766e+02

also keine Änderung in der Genauigkeit.

Gehen wir nun zu negativen Argumenten über. Für $x = -1$ und float-Genauigkeit erhält man das folgende Ergebnis

2.755731998149713e-07	10	3.678794205188751e-01
-2.505210972003624e-08	11	3.678793907165527e-01
2.087675810003020e-09	12	3.678793907165527e-01
	ex	3.678794411714423e-01

also nur noch 6 gültige Ziffern.

Für $x = -5$ ergibt sich

-5.000000000000000e+00	1	-4.000000000000000e+00
1.250000000000000e+01	2	8.500000000000000e+00
-2.083333396911621e+01	3	-1.233333396911621e+01
2.604166793823242e+01	4	1.370833396911621e+01
-2.333729527890682e-02	15	1.118892803788185e-03
7.292904891073704e-03	16	8.411797694861889e-03
1.221854423194557e-10	28	6.737461313605309e-03
0.000000000000000e+00	100	6.737461313605309e-03
	ex	6.737946999085467e-03

also nur noch 4 gültige Ziffern!

Schließlich ergibt sich für $x = -20$ und float-Genauigkeit

-2.000000000000000e+01	1	-1.900000000000000e+01
2.000000000000000e+02	2	1.810000000000000e+02
-1.33333374023438e+03	3	-1.152333374023438e+03
6.666666992187500e+03	4	5.514333496093750e+03
-2.666666796875000e+04	5	-2.115233398437500e+04
-2.611609750000000e+06	31	-1.011914250000000e+06
1.632256125000000e+06	32	6.203418750000000e+05
-9.89246125000000e+05	33	-3.689042500000000e+05
5.81909500000000e+05	34	2.130052500000000e+05
-3.32519718750000e+05	35	-1.195144687500000e+05
1.847331718750000e+05	36	6.521870312500000e+04
-4.473213550681976e-07	65	7.566840052604675e-01
1.355519287926654e-07	66	7.566841244697571e-01
-4.046326296247571e-08	67	7.566840648651123e-01
1.190095932912527e-08	68	7.566840648651123e-01
	ex	2.061153622438557e-09

In diesem Fall ist *keine* Stelle im Ergebnis korrekt! Das Ergebnis ist absolut um 8 Größenordnungen falsch!

Nun liegt es nahe die Berechnungen mit einer genaueren Zahlendarstellung zu wiederholen. Für $x = -20$ und `double`-Genauigkeit erhält man die folgenden Werte

-1.232613988175268e+07	27	-5.180694836889297e+06
8.804385629823344e+06	28	3.623690792934047e+06
1.821561256740375e-24	94	6.147561828914626e-09
-3.834865803663947e-25	95	6.147561828914626e-09
	ex	2.061153622438557e-09

Das Ergebnis hat immer noch keine korrekte Stelle, aber immerhin liegt es in der gleichen Größenordnung.

Erst mit „vierfacher Genauigkeit“ gegenüber dem Datentyp `float` erhält man

118 2.0611536224385583392700458752947e-09
ex 2.0611536224385578279659403801558e-09

also 15 gültige Ziffern im Ergebnis, bei ca 30 Ziffern „Rechengenauigkeit“.

Damit sind wir für die folgenden Fragen motiviert:

- Was bedeutet „Rechengenauigkeit“?
- Wann und wie entstehen Fehler in einer Berechnung?
- Können diese Fehler vermieden oder minimiert werden?

Im übrigen können die Fehler in diesem Beispiel einfach vermieden werden: Man nutzt einfach die Beziehung $e^x = 1/e^{-x}$ und führt damit die Berechnung der Exponentialfunktion für negative Argumente $x < 0$ auf die Berechnung mit einem positiven Argument zurück. \square

2.2 Fließkommazahlen

Zur Zahlendarstellung hat sich schon vor Jahrtausenden das *Stellenwertsystem* herausgebildet. Eine natürliche Zahl $x \in \mathbb{N}$ wird dabei dargestellt als

$$x = m_n \beta^n + \dots + m_1 \beta + m_0 = \sum_{i=0}^n m_i \beta^i$$

mit den *Ziffern* $m_0, \dots, m_n \in \{0, 1, \dots, \beta - 1\}$ und der *Basis* $\mathbb{N} \ni \beta \geq 2$. Schon im alten Babylon hat man dieses System mit $\beta = 60$ verwendet. Die Basis 10 hat sich in Europa ab dem 16. Jahrhundert durchgesetzt und Blaise Pascal konnte zeigen, dass jede natürliche Zahl $\beta \geq 2$ als Basis geeignet ist, siehe [Knuth, 1998, p. 194]. Als ein Beispiel für die Zahlendarstellung ohne Stellenwertsystem seien die römischen Zahlen erwähnt. Diese Darstellung ist für arithmetische Operationen wenig geeignet.

Für negative und nichtganze Zahlen erweitert man das Stellenwertsystem einfach um ein Vorzeichen und negative Potenzen und erhält die *Festkomma-darstellung*:

$$x = \pm m_n \beta^n + \dots + m_1 \beta + m_0 + m_{-1} \beta^{-1} + \dots + m_{-k} \beta^{-k} = \pm \sum_{i=-k}^n m_i \beta^i.$$

Allerdings kommen in naturwissenschaftlichen Berechnungen sehr verschiedene große Zahlen vor. So hat das Planck'sche Wirkungsquantum den Wert $6.6260693 \cdot 10^{-34} \text{ Js}$ und die Avogadrokonstante aus der Chemie hat den Wert $6.021415 \cdot 10^{23} \text{ mol}^{-1}$. Die Darstellung dieser Zahlen in einem Festkommasytem würde eine sehr große Stellenzahl erfordern, z.B. mit der im Computer üblichen Basis $\beta = 2$ wären ungefähr 190 Binärstellen erforderlich. Da Zahlen mit solchen Größenunterschieden oft *nicht gleichzeitig* verarbeitet werden müssen führt man die effizientere Fließkommadarstellung ein.

Definition 2.2 (Normierte Fließkommadarstellung). Sei $\beta, r, s \in \mathbb{N}$ und $\beta \geq 2$. $\mathbb{F}(\beta, r, s) \subset \mathbb{R}$ heißt *Menge der normierten Fließkommazahlen*. Ihre Elemente erfüllen die folgenden Eigenschaften

a) Für alle $x \in \mathbb{F}(\beta, r, s)$ gilt

$$x = m(x)\beta^{e(x)}, \quad m(x) = \pm \sum_{i=1}^r m_i \beta^{-i}, \quad e(x) = \pm \sum_{j=0}^{s-1} e_j \beta^j.$$

Dabei heißt $m(x)$ Mantisse von x und $e(x)$ Exponent von x .

b) Für alle $x \in \mathbb{F}(\beta, r, s)$ gilt: $x = 0 \vee m_1 \neq 0$. Dies *Normierungsbedingung* führt zu einer eindeutigen Fließkommadarstellung von $x \neq 0$ und erlaubt Abschätzungen über die Größe der Mantisse. \square

Leiten Sie diese Abschätzungen her.

Aus dieser Definition ergeben sich folgende Abschätzungen für $x \neq 0$:

$$\frac{1}{\beta} \leq |m(x)| < 1, \quad \beta^{e(x)-1} \leq |x| \leq \beta^{e(x)}. \quad (2.1)$$

Beispiel 2.3. a) $\mathbb{F}(10, 3, 1)$ besteht aus Zahlen der Form

$$x = \pm(m_1 \cdot 0.1 + m_2 \cdot 0.01 + m_3 \cdot 0.001) \cdot 10^{\pm e_0}$$

mit $0 \leq m_i, e_0 < 10$, $m_1 \neq 0$ sowie der Zahl 0, also insgesamt $9 \cdot 10 \cdot 10 \cdot 19 \cdot 2 + 1 = 34201$ Zahlen.

b) $\mathbb{F}(2, 2, 1)$ besteht aus Zahlen der Form

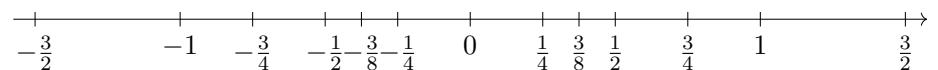
$$x = \pm(m_1 \cdot \frac{1}{2} + m_2 \cdot \frac{1}{4}) \cdot 2^{\pm e_0}$$

mit $m_1 = 1$, $0 \leq m_2, e_0 \leq 1$ sowie der Null. Beachte, dass hier die Darstellung zur Basis 10 erfolgt! Damit ergibt sich

$$\mathbb{F}(2, 2, 1) = \left\{ -\frac{3}{2}, -1, -\frac{3}{4}, -\frac{1}{2}, -\frac{3}{8}, -\frac{1}{4}, 0, \frac{1}{4}, \frac{3}{8}, \frac{1}{2}, \frac{3}{4}, 1, \frac{3}{2} \right\}.$$

Im Fall $\beta = 2$ bedeutet die Normierung, dass die Stelle $m_1 = 1$ festgelegt ist. Damit hat man eigentlich $r + 1$ Stellen in der Mantisse zur Verfügung ohne den Speicheraufwand zu erhöhen. Dies nennt man *hidden bit*.

Stellt man $\mathbb{F}(2, 2, 1)$ auf dem Zahlenstrahl dar, ergibt sich



Der minimale Abstand zweier Zahlen in $\mathbb{F}(2, 2, 1)$ ist $\frac{1}{8}$ allerdings wird dieser nicht bei der Null erreicht – ein Nachteil der Normierung. In praktischen Implementierungen normierter Fließkommazahlen gibt man daher die Normierung nahe der Null auf. \square

Zahlenbereich In einem gegebenen, normierten Fließkommazystem $\mathbb{F}(\beta, r, s)$ ist die größte bzw. kleinste darstellbare Zahl gegeben durch

$$X_{\pm} = \pm(\beta - 1) \left(\sum_{i=1}^r \beta^{-i} \right) \cdot \beta^{(\beta-1) \sum_{j=0}^{s-1} \beta^j}.$$

Die kleinste positive bzw. größte negative Zahl lautet

$$x_{\pm} = \pm \beta^{-1 - (\beta-1) \sum_{j=0}^{s-1} \beta^j}$$

und damit gilt

$$\mathbb{F}(\beta, r, s) \subset D(\beta, r, s) = [X_{-}, x_{-}] \cup \{0\} \cup [x_{+}, X_{+}].$$

Abstände zwischen Fließkommazahlen Wie in obigem Beispiel 2.3 schon ersichtlich sind die Abstände zwischen Fließkommazahlen nicht gleich. Aufschluss gibt folgende Beobachtung.

Beobachtung 2.4. Es werde das Fließkommazystem $\mathbb{F}(\beta, r, s)$ betrachtet.

a) Für zwei Fließkommazahlen $x \neq x'$ im Bereich $\beta^{e-1} \leq x, x' \leq \beta^e$ gilt

$$|x - x'| \geq \beta^{e-r}$$

wobei $e = e(x) = e(x')$. Gilt $|x - x'| = \beta^{e-r}$ so bezeichnen wir die beiden Zahlen als *benachbart*.

b) Für zwei benachbarte Fließkommazahlen $x \neq x'$ im Bereich $\beta^{e-1} \leq x, x' \leq \beta^e$ gilt

$$\beta^{-r} < \frac{|x - x'|}{|x|} \leq \beta^{1-r}.$$

Dieser sogenannte *relative* Abstand zweier benachbarter Zahlen schwankt also maximal um einen Faktor β . Daher sind kleine Werte für die Basis β in dieser Hinsicht von Vorteil.

Beweis. a) Wegen $\beta^{e-1} \leq x < x' \leq \beta^e$ gilt $x = m(x)\beta^e$ und $x' = m(x')\beta^e$ mit gleichem Exponenten e , dabei erlauben wir ausnahmsweise auch die Mantisse $m(x') = 1$ im Fall $x' = \beta^e$ (eigentlich wäre dann $m(x') = \beta^{-1}$ und $e(x) = e+1$ aber dies ändert nichts an der Zahl x' , analog dürfte auch $x = \beta^e$ sein). Außerdem ist $m(x), m(x') > 0$. In diesem Bereich gilt $|m(x) - m(x')| \geq \beta^{-r}$ dabei ist β^{-r} die Wertigkeit der kleinsten Mantissenstelle. Somit erhalten wir

$$|x - x'| = |m(x)\beta^e - m(x')\beta^e| = |m(x) - m(x')|\beta^e \geq \beta^{e-r}.$$

b) Es sind $x < x'$ benachbarte Fließkommazahlen im Bereich $\beta^{e-1} \leq x, x' \leq \beta^e$. Mit dem Wissen von a) gilt

$$\frac{|x - x'|}{|x|} = \frac{\beta^{e-r}}{m(x)\beta^e} = \frac{\beta^{-r}}{m(x)}.$$

Aufgrund der Normierung gilt $\beta^{-1} \leq m(x) \leq 1 \Leftrightarrow 1 \geq m(x)^{-1} \leq \beta$ und damit gilt insgesamt

$$\beta^{-r} \geq \frac{|x - x'|}{|x|} \leq \beta^{1-r}.$$

□

2.3 IEEE 754 Standard

Dieser wurde 1985 verabschiedet um die Fließkommadarstellung und auch Arithmetik in Rechnern zu standardisieren. Mit dem Ziel, dass auch Programme die Fließkommazahlen verarbeiten auf verschiedenen Rechnern das gleiche Ergebnis liefern. Der Standard fordert $\beta = 2$ mit vier Genauigkeitsstufen:

Größe	single	single-ext	double	double-ext
e_{max}	127	≥ 1024	1023	≥ 16384
e_{min}	-126	≤ -1021	-1022	≤ -16381
Bits Exponent	8	≤ 11	11	15
Bits insgesamt	32	≥ 43	64	≥ 79

Betrachten wir die `double`-Genauigkeit genauer. Insgesamt stehen 64 Bit zur Verfügung, davon 11 Bit für den Exponenten. Der Exponent wird als vorzeichenlose Zahl $c \in [1, 2046]$ gespeichert. Es gilt dann $-1022 \leq e = c - 1023 \leq 1023$. Die Werte $c \in \{0, 2047\}$ werden zusammen mit der Mantisse anderweitig genutzt:

c	m	Situation
0	0	kodiert die Null
0	$\neq 0$	denormalisierte Darstellung
2047	$\neq 0$	kodiert not a number
2047	0	kodiert ∞ (Überlauf)

Für die Mantisse stehen $64 - 11 = 53$ Bit zur Verfügung. Ein Bit wird für das Vorzeichen benötigt, es bleiben 52 Bit. Wegen des hidden bit ist die Wertigkeit der kleinsten Mantissenstelle allerdings wieder 2^{-53} .

Außerdem stellt der Standard vier Rundungsarten zur Verfügung: Aufrunden, Abrunden, zur Null hin runden und die natürliche Rundung (siehe unten). Schließlich sind die Fließkommoperationen exakt gerundet (wird ebenfalls unten erläutert).

2.4 Runden und Rundungsfehler

Um $x \in \mathbb{R}$ zu approximieren benötigen wir eine Abbildung

$$\text{rd} : D(\beta, r, s) \rightarrow \mathbb{F}(\beta, r, s).$$

Die Abbildung setzt also voraus, dass x im prinzipiell darstellbaren Bereich liegt. Ansonsten sind ist der Parameter s entsprechend zu vergrößern. Eine sinnvolle Forderung für rd ist in jedem Fall

$$|x - \text{rd}(x)| = \min_{y \in \mathbb{F}(\beta, r, s)} |x - y| \quad (\text{Bestapproximation}).$$

Wir setzen

$$l(x) = \max\{y \in \mathbb{F}(\beta, r, s) : y \leq x\}, \quad r(x) = \min\{y \in \mathbb{F}(\beta, r, s) : y \geq x\}$$

und

$$\text{rd}(x) = \begin{cases} x & l(x) = r(x), \text{ i.e. } x \in \mathbb{F} \\ l(x) & |x - l(x)| < |x - r(x)| \\ r(x) & |x - r(x)| < |x - l(x)| \\ ? & |x - r(x)| = |x - l(x)| \end{cases} \quad (2.2)$$

Im letzten Fall, wenn x genau zwischen $l(x)$ und $r(x)$ ist, gibt es verschiedene Möglichkeiten, von denen wir zwei weiter unten betrachten wollen.

Die Abbildung rd soll nun konkret im Computer realisiert werden. Da entsteht zunächst die Frage, wie die Zahl $x \in \mathbb{R}$, die gerundet werden soll, überhaupt in den Computer gelangt. Dort soll sie ja gerade in einem Fließkomma-System dargestellt werden. Daher wollen wir uns hier auf den Fall

$$rd : \mathbb{F}(\beta, r', s) \rightarrow \mathbb{F}(\beta, r, s), \quad r' > r,$$

beschränken, d.h. es wird von einem Fließkommasystem mit mehr Mantissenstellen in einem mit weniger Mantissenstellen gerundet. Bei der Rundung transzenter Zahlen wie e oder π kann das *Tabellenmacherdilemma* entstehen: Zunächst kann jede reelle Zahl als Fließkommazahl mit unendlicher Zahl von Mantissenstellen dargestellt werden, $x = \text{sign}(x) (\sum_{i=1}^{\infty} m_i \beta^{-i}) \beta^e$, und es entsteht die Frage ob die Ziffern m_i der Reihe nach berechnet werden können. Nun werden transzente Zahlen als Reihen entwickelt (siehe etwa Beispiel 2.1) und da könnte folgende Situation entstehen:

$$\begin{aligned} x_n &= 5.0835 \\ x_{n+1} &= 5.0835000 \\ x_{n+2} &= 5.083500000 \\ &\vdots \end{aligned}$$

und man kann a-priori nicht angeben nach wievielen Schritten $x < 5.0835$ oder $x > 5.0835$ entschieden werden kann. Wir nehmen also nun $x \in \mathbb{F}(\beta, r', s)$, $r' > r$ an und definieren die beiden Rundungsarten:

1) *Natürliche (bzw. kaufmännische) Rundung*: Hier setzt man

$$rd(x) = \begin{cases} \text{sign}(x) (\sum_{i=1}^r m_i \beta^{-i}) \beta^e & \text{falls } m_{r+1} < \beta/2, \\ \text{sign}(x) [(\sum_{i=1}^r m_i \beta^{-i}) + \beta^{-r}] \beta^e & \text{falls } m_{r+1} \geq \beta/2. \end{cases}$$

Zur Entscheidung wird die Stelle m_{r+1} herangezogen, es genügt also im Prinzip $r' = r + 1$. Im Sinne von Gleichung (2.2) entscheidet sich die natürliche Rundung im Fall $|x - r(x)| = |x - l(x)|$ immer für den betragsmäßig größeren Wert, d.h. positive Zahlen werden aufgerundet, negative Zahlen werden abgerundet.

2) *Gerade Rundung*: Hier setzt man β als gerade Zahl voraus und definiert

$$rd(x) = \begin{cases} x & l(x) = r(x), \text{ i.e. } x \in \mathbb{F} \\ l(x) & |x - l(x)| < |x - r(x)| \\ r(x) & |x - r(x)| < |x - l(x)| \\ l(x) & |x - r(x)| = |x - l(x)| \wedge m_r \text{ gerade} \\ r(x) & |x - r(x)| = |x - l(x)| \wedge m_r \text{ ungerade} \end{cases}$$

Hier rundet man im Fall $|x - r(x)| = |x - l(x)|$ sowohl auf als auch ab, in Abhängigkeit der Ziffer m_r . Sind diese statistisch gleichmäßig verteilt so rundet man genauso oft ab wie auf und vermeidet dadurch mögliche Verzerrungen von Statistiken die bei der kaufmännischen Rundung entstehen können. Die Entscheidung $|x - r(x)| = |x - l(x)|$ erfordert Kenntnis aller r' Mantissenstellen.

Beispiel 2.5. Gegeben sei $x_1 = 0.32642876$. Sowohl kaufmännisches als auch gerades Runden ergeben bei Rundung auf zwei Mantissenstellen 0.33 und bei Rundung auf drei Mantissenstellen 0.326.

Für $x_2 = 0.74500$ und Rundung auf 2 Mantissenstellen ergibt kaufmännisches Runden 0.75 und gerades Runden 0.74.

Für $x_3 = 0.75500$ und Rundung auf 2 Mantissenstellen ergibt kaufmännisches Runden 0.76 und gerades Runden ebenfalls 0.76. \square

Über den maximal möglichen Fehler der bei *einer* Rundungsoperation entsteht gibt das folgende Lemma Auskunft. Dabei unterscheiden wir zwischen absolutem und relativem Fehler die wir zunächst einführen.

Definition 2.6 (Relativer und absoluter Fehler). Wird eine Größe x durch eine Größe x' angenähert, dann bezeichnet

$$e_{\text{abs}} = x - x' \quad (2.3)$$

den absoluten Fehler und

$$e_{\text{rel}} = \frac{x - x'}{x} \quad (x \neq 0) \quad (2.4)$$

den relativen Fehler in der Näherung. \square

Relative Fehler sind in der Praxis oft relevanter als absolute Fehler. So ist bei einer Aussage "Der Unterschied beider Werte beträgt ein Meter" nicht klar ob dies ein Großer oder kleiner Unterschied ist. Beim Hochsprung wäre ein Unterschied von einem Meter relativ viel, bei der Entfernung Heidelberg-Hamburg relativ wenig.

Lemma 2.7 (Rundungsfehler). Es sei $x \in D(\beta, r, s)$ und $\text{rd} : D(\beta, r, s) \rightarrow \mathbb{F}(\beta, r, s)$ eine der oben beschriebenen Rundungsarten. Dann gilt für den absoluten Rundungsfehler

$$|x - \text{rd}(x)| \leq \frac{1}{2} \beta^{e(x)-r} \quad (2.5)$$

und für den relativen Rundungsfehler

$$\frac{|x - \text{rd}(x)|}{|x|} \leq \frac{1}{2} \beta^{1-r} \quad x \neq 0. \quad (2.6)$$

Beweis. Es sei $x = m(x)\beta^{e(x)}$ die normierte Fließkommadarstellung von x mit möglicherweise unendlich großer Stellenzahl (wir werden die genaue Mantisse $m(x)$ nicht benötigen).

$l(x)$ und $r(x)$ sind benachbarte Fließkommazahlen. Somit gilt wegen Beobachtung 2.4

$$|r(x) - l(x)| = \beta^{e(x)-r}.$$

Der maximale Rundungsfehler ergibt sich für $x = (l(x) + r(x))/2$, also

$$|x - \text{rd}(x)| \leq \left| \frac{l(x) + r(x)}{2} - l(x) \right| = \frac{1}{2} |r(x) - l(x)| = \frac{1}{2} \beta^{e(x)-r}.$$

Dies beweist die erste Aussage. Für den relativen Fehler bei $x \neq 0$ gilt

$$\frac{|x - \text{rd}(x)|}{|x|} \leq \frac{\frac{1}{2} \beta^{e(x)-r}}{|m(x)| \beta^{e(x)}} = \frac{1}{2} \frac{1}{|m(x)|} \beta^{-r} \leq \frac{1}{2} \beta^{1-r}.$$

Dabei haben wir die Normierung $|m(x)| \geq \beta^{-1}$ ausgenutzt. \square

Die Zahl $\text{eps} = \frac{1}{2} \beta^{1-r}$ heißt *Maschinengenauigkeit*, *machine precision* oder *machine epsilon*.

Vorlesung 3

Rechnen mit Fließkommazahlen

3.1 Fließkommaarithmetik

Nun möchte man mit Fließkommazahlen natürlich auch rechnen, d.h. wir benötigen zweistellige Operationen

$$\circledast : \mathbb{F} \times \mathbb{F} \rightarrow \mathbb{F} \quad \circledast \in \{\oplus, \ominus, \odot, \oslash\}, \quad (3.1)$$

welche den Grundrechenarten $+, -, \cdot, /$ auf den reellen Zahlen entsprechen. In diesem Kapitel steht \mathbb{F} für ein beliebiges Fließkommensystem $\mathbb{F}(\beta, r, s)$. Die genauen Parameter werden nur dann spezifiziert wenn dies wichtig ist.

Die Problematik der Fließkommaoperationen ist, dass in der Regel selbst für zwei Fließkommazahlen $x, y \in \mathbb{F}$ das Ergebnis der exakten Verknüpfung $x * y$ nicht in notwendigerweise in \mathbb{F} liegt. So wird sich in der Regel bei der Multiplikation die Zahl der Mantissenstellen verdoppeln. Auch das Ergebnis einer Addition zweier Zahlen mit verschiedenen Exponenten erfordert in der Regel mehr Mantissenstellen.

Ein naheliegende Definition für die Fließkommaarithmetik ist daher

$$x \circledast y = \text{rd}(x * y) \quad * \in \{+, -, \cdot, /\}. \quad (3.2)$$

Ein Fließkommaarithmetik mit dieser Eigenschaft nennt man *exakt gerundet*.

Eine triviale Realisierung exakter Rundung ist jedoch nicht effizient. Betrachten wir die Addition von $x = 10^{10}$ und $y = 10^{-10}$ in $\mathbb{F}(10, 4, 2)$, Bevor die Mantissen addiert werden können sind beide Summanden auf den gleichen Exponenten zu bringen, also $x = 0.1 \cdot 10^{11}$ und $y = 10^{-10} = 10^{-10-11} \cdot 10^{11} = 10^{-21} \cdot 10^{11}$. Dementsprechend bräuchte man 21 Nachkommastellen um in einem Zwischenschritt $x * y$ exakt zu berechnen. Die anschließende Rundung schneidet viele der zusätzlichen Stellen wieder ab, das ist sehr ineffizient. Die Rundung gleich nach Egalisierung der Exponenten durch zuführen ist allerdings gefährlich und kann zu großen relativen Fehlern führen. Es zeigt sich, dass folgende Vorgehensweise zum Ziel führt:

- 1) Bringe beide Argumente auf den gleichen Exponenten, also die betragsmäßig kleinere Zahl auf den Exponenten der betragsmäßig Größeren. Diese ist dann nicht mehr normiert.
- 2) Runde die geschobene Zahl auf $r + 2$ Mantissenstellen.
- 3) Addiere beide Zahlen mit $r + 2$ Mantissenstellen. Diese zusätzlichen Stellen nennt man *guard digits*.
- 4) Runde das Ergebnis auf r Stellen.

Mit zwei zusätzlichen Stellen erreicht man so ein exakte Rundung. Im IEEE 754 Standard sind sowohl die Grundoperationen als auch die Berechnung der Quadratwurzel exakt gerundet.

Die Fließkommaarithmetik unterscheidet sich von der Arithmetik in den reellen Zahlen, hier eine Liste relevanter Eigenschaften:

- a) Es gibt Zahlen $y \in \mathbb{F}$ so dass $x \oplus y = x$ gilt.
- b) Assoziativ und Distributivgesetz gelten nicht (unbedingt). Es kommt also auf die Reihenfolge der Operationen an. Z.B. gilt mit dem y aus a): $(x \oplus y) \ominus x = 0$ und $y \oplus (x \ominus x) = y$. Hier haben wir ausgenutzt, dass $x \ominus x = 0$.
- c) Es gilt jedoch das Kommutativgesetz!
- d) Auch folgende einfache Regeln gelten

$$(-x) \odot y = -(x \odot y)$$

$$1 \odot x = x$$

$$x \odot y = 0 \quad \text{nur dann wenn } x = 0 \text{ oder } y = 0$$

$$x \odot z \leq y \odot z \quad \text{wenn } x \leq y \text{ und } z > 0$$

3.2 Fehleranalyse

Wir kommen nun zu den Auswirkungen von Rundungsfehlern in Rechnungen die aus einer Folge von Grundoperation zusammengesetzt sind. Abstrakt betrachten wir die Berechnung als Auswertung einer vektorwertigen Funktion

$$F : \mathbb{R}^m \rightarrow \mathbb{R}^n.$$

Die Berechnung von F im Computer wird durch eine entsprechende Funktion

$$F' : \mathbb{F}^m \rightarrow \mathbb{F}^n$$

realisiert. F' wird durch endlich viele Elementaroperationen

$$F'(x') = \varphi_l(\dots \varphi_2(\varphi_1(x')) \dots)$$

realisiert. Jedes φ_i führt eine Grundoperation aus $\{\oplus, \ominus, \odot, \oslash\}$ durch und steuert einen unbekannten Teilfehler bei. Die numerische Realisierung F' von F ist in der Regel nicht eindeutig. Zum einen gibt es mathematisch äquivalente Möglichkeiten einen Ausdruck zu berechnen, z.B $x(y+z) = xy + xz$ und selbst unterschiedlich Reihenfolgen der Zwischenschritte können zu unterschiedlichen Ergebnissen in \mathbb{F} führen.

Im größeren praktischen Kontext sind auch die Eingaben der exakten Funktion F mit Unsicherheiten behaftet. Dann ist es interessant zu untersuchen wie sich Variationen in der Eingabe x gegenüber den akkumulierten Rundungsfehlern verhalten. Dies erreicht man mit der nun folgenden Analyse. Dazu betrachten wir die Aufspaltung

$$F(x) - F'(\text{rd}(x)) = \underbrace{F(x) - F(\text{rd}(x))}_{\text{Konditionsanalyse}} + \underbrace{F(\text{rd}(x)) - F'(\text{rd}(x))}_{\text{Rundungsfehleranalyse}}. \quad (3.3)$$

Die erste Differenz analysiert wie die exakte Abbildung F auf Änderungen in der Eingabe (hier Rundungsfehler) reagiert. Diese Analyse ist unabhängig von der Problematik der Fließkommaarithmetik. Die zweite Differenz analysiert wie sich die beiden Abbildungen F, F' bei der selben Eingabe unterscheiden, also die eigentliche Rundungsfehleranalyse.

Schlussendlich wird man Fehlernormen analysieren. Dann folgt mittels Dreiecksungleichung

$$\|F(x) - F'(\text{rd}(x))\| \leq \|F(x) - F(\text{rd}(x))\| + \|F(\text{rd}(x)) - F'(\text{rd}(x))\|.$$

3.3 Differentielle Konditionsanalyse

Die folgende Analyse braucht den Satz von Tayler aus der Analysis. Da dieser Satz in der Vorlesung sehr häufig zitiert wird wollen wir ihn in zwei Varianten anführen.

Satz 3.1 (Satz von Taylor für Funktionen in einer Variable). Sei $f : (a, b) \rightarrow \mathbb{R}$ $(r+1)$ mal stetig differenzierbar und seien $x, z \in (a, b)$ gegeben. Dann gibt es ein $\xi \in (a, b)$ zwischen x und z (und abhängig von z), so dass gilt:

$$f(z) = \sum_{k=0}^r \frac{1}{k!} \frac{d^k f(x)}{dx^k} (z-x)^k + \frac{1}{(r+1)!} \frac{d^{r+1} f(\xi)}{dx^{r+1}} (z-x)^{r+1}. \quad (3.4)$$

Dabei heisst $t_n(x, z) = \sum_{k=0}^r \frac{1}{k!} \frac{d^k f(x)}{dx^k} (z-x)^k$ *Taylorpolynom* in z um den *Entwickelpunkt* x und $R_r(x, z) = \frac{1}{(r+1)!} \frac{d^{r+1} f(\xi)}{dx^{r+1}} (z-x)^{r+1}$ *Lagranges Restglied*.

Beweis. Siehe [Rannacher, 2018a, Satz 5.8] \square

Bemerkung 3.2. Oft setzt man $z = x + h$ und somit $z - x = h$ mit der Idee, dass $|h|$ “klein” ist. Damit lauten die Taylorformel und das Restglied alternativ

$$f(x+h) = \sum_{k=0}^r \frac{1}{k!} \frac{d^k f(x)}{dx^k} h^k + \frac{1}{(r+1)!} \frac{d^{r+1} f(\xi)}{dx^{r+1}} h^{r+1}. \quad (3.5)$$

Der Satz von Taylor kann auf $r+1$ mal stetig partiell differenzierbare Funktionen $f : D \rightarrow \mathbb{R}$, mit D einer offenen Teilmenge des \mathbb{R}^m , erweitert werden. Um komplexe Notation zu vermeiden zitieren wir hier nur eine spezialisierte Variante des Satzes für $r=1$. Mehr werden wir im folgenden nicht benötigen.

Satz 3.3 (Spezialfall des Satz von Taylor für Funktionen in mehreren Variablen). Sei $D \subset \mathbb{R}^m$ eine offene Menge und $f : D \rightarrow \mathbb{R}$ zweimal stetig differenzierbar. Weiter seien $x, \eta \in \mathbb{R}^m$ derart, dass $x + s\eta \in D$ für alle $s \in [0, 1]$. Dann gibt es zu so einem η ein $\theta \in (0, 1)$ so dass

$$\begin{aligned} f(x + \eta) &= f(x) + \sum_{i=1}^m \frac{\partial f(x)}{\partial x_i} \eta_i \\ &+ \frac{1}{2} \sum_{i=1}^m \frac{\partial^2 f(x + \theta\eta)}{\partial x_i^2} \eta_i^2 + \sum_{i=1}^m \sum_{i < j \leq m} \frac{\partial^2 f(x + \theta\eta)}{\partial x_i \partial x_j} \eta_i \eta_j. \end{aligned} \quad (3.6)$$

Dabei ist in der zweiten Zeile das Restglied $R_1(x, \eta) = \frac{1}{2} \sum_{i=1}^m \frac{\partial^2 f(x + \theta\eta)}{\partial x_i^2} \eta_i^2 + \sum_{i=1}^m \sum_{i < j \leq m} \frac{\partial^2 f(x + \theta\eta)}{\partial x_i \partial x_j} \eta_i \eta_j$ in Lagranger Form angegeben.

Beweis. Folgerung aus [Rannacher, 2018b, Satz 3.6] \square

Bemerkung 3.4. Oft benötigen wir Abschätzungen des Restglieds. Mit $h := \max_{1 \leq i \leq n} |\eta_i|$ gilt dann

$$|R_1(x, \eta)| \leq \left(\frac{1}{2} \sum_{i=1}^m \left| \frac{\partial^2 f(x + \theta\eta)}{\partial x_i^2} \right| + \sum_{i=1}^m \sum_{i < j \leq m} \left| \frac{\partial^2 f(x + \theta\eta)}{\partial x_i \partial x_j} \right| \right) h^2.$$

Für $h \rightarrow 0$ konvergiert der Wert in der Klammer und der Betrag des Restgliedes “geht wie h^2 ” gegen Null. Man schreibt hierfür auch “ $|R_1(x, \eta)| = O(h^2)$ ”.

Zur allgemeinen Quantifizierung der Konvergenzgeschwindigkeit dienen die

Definition 3.5 (Landau Symbole). a) Man schreibt

$$g(t) = O(h(t)) \quad (t \rightarrow 0)$$

falls es ein $t_0 > 0$ und $c_0 \geq 0$ gibt so dass für alle $0 < t \leq t_0$ die Abschätzung

$$|g(t)| \leq c_0|h(t)|$$

gilt. Man sagt “ $g(t)$ geht mindestens wie $h(t)$ gegen Null”.

b) Weiter schreibt man

$$g(t) = o(h(t)) \quad (t \rightarrow 0)$$

wenn es ein $t_0 > 0$ und eine Funktion $c(t)$ mit $\lim_{t \rightarrow 0} c(t) = 0$ gibt, so dass für alle $0 < t \leq t_0$ die Abschätzung

$$|g(t)| \leq c(t)|h(t)|$$

gilt. Geht $h(t)$ ebenfalls gegen Null dann drückt dies aus: “ $g(t)$ geht schneller als $h(t)$ gegen Null”.

c) Schließlich schreiben wir

$$g(t) \doteq h(t)$$

und sagen „ $g(t)$ ist in erster Näherung gleich $h(t)$ “, wenn gilt

$$g(t) - h(t) = o(t).$$

□

Aus dem Satz von Taylor folgt mittels der Beobachtung $g(t) = O(t^2) \Rightarrow g(t) = o(t)$

$$f(x + \eta) \doteq f(x) + \sum_{i=1}^m \frac{\partial f(x)}{\partial x_i} \eta_i$$

Nach diesem Ausflug in die Analysis wenden wir uns nun wieder der Konditionsanalyse zu. Zur Analyse der Auswirkung von Änderungen in der Eingabe der Funktion F betrachten wir diese als zweimal stetig differenzierbare Abbildung. Nach dem Satz von Taylor 3.3 gilt damit für jedem Komponente F_i :

$$F_i(x + \Delta x) = F_i(x) + \sum_{j=1}^m \frac{\partial F_i}{\partial x_j}(x) \Delta x_j + R_{1,i}(x, \Delta x), \quad i = 1, \dots, n, \quad (3.7)$$

mit einem Restglied $R_{1,i}(x, \Delta x) = O(h^2)$ und $h = \max_{1 \leq i \leq n} |\Delta x_i|$. Wir formen um und gehen zur \doteq -Notation über:

$$F_i(x + \Delta x) - F_i(x) \doteq \sum_{j=1}^m \frac{\partial F_i}{\partial x_j}(x) \Delta x_j, \quad i = 1, \dots, n.$$

Für die relative Änderung erhalten wir dann

$$\frac{F_i(x + \Delta x) - F_i(x)}{F_i(x)} \doteq \sum_{j=1}^m \frac{\partial F_i}{\partial x_j}(x) \frac{\Delta x_j}{F_i(x)} = \sum_{j=1}^m \underbrace{\left(\frac{\partial F_i}{\partial x_j}(x) \frac{x_j}{F_i(x)} \right)}_{\text{Verstärkungsfaktor } k_{ij}(x)} \frac{\Delta x_j}{x_j}, \quad i = 1, \dots, n.$$

Der Verstärkungsfaktor k_{ij} beschreibt wie stark sich die relative Änderung in der j -ten Komponente der Eingabe auf die relative Änderung in der i -ten Komponente des Ergebnisses auswirkt.

Definition 3.6. Wir nennen die Auswertung $y = F(x)$ “schlecht konditioniert” im Punkt x falls $|k_{ij}(x)| \gg 1$, andernfalls heißt die Auswertung “gut konditioniert”. Bei $|k_{ij}(x)| < 1$ spricht man von Fehlerdämpfung, bei $|k_{ij}(x)| > 1$ von Fehlerverstärkung. \square

In der Aufspaltung (3.3) entspricht $\Delta x = \text{rd}(x) - x$, also gerade dem absoluten Rundungsfehler in der Eingabe. Damit gilt $\frac{\Delta x_j}{x_j} = \frac{1}{2}\beta^{1-r}$.

Beispiel 3.7 (Konditionierung der Grundoperationen). a) Wir untersuchen die

Konditionierung der Addition, also $F(x_1, x_2) = x_1 + x_2$. Offensichtlich gilt $\frac{\partial F}{\partial x_1} = 1$, $\frac{\partial F}{\partial x_2} = 1$ und damit nach obiger Formel

$$\frac{F(x_1 + \Delta x_1, x_2 + \Delta x_2) - F(x_1, x_2)}{F(x_1, x_2)} = \left(1 \cdot \frac{x_1}{x_1 + x_2}\right) \frac{\Delta x_1}{x_1} + \left(1 \cdot \frac{x_2}{x_1 + x_2}\right) \frac{\Delta x_2}{x_2}.$$

Für $x_1 \rightarrow -x_2$ werden beide Verstärkungsfaktoren sehr groß. In diesem Fall ist die Addition schlecht konditioniert. Dies gilt analog für die Subtraktion falls $x_1 \rightarrow x_2$.

b) Für die Multiplikation $F(x_1, x_2) = x_1 \cdot x_2$ gilt $\frac{\partial F}{\partial x_1} = x_2$, $\frac{\partial F}{\partial x_2} = x_1$ und damit

$$\frac{F(x_1 + \Delta x_1, x_2 + \Delta x_2) - F(x_1, x_2)}{F(x_1, x_2)} = \left(x_2 \cdot \frac{x_1}{x_1 \cdot x_2}\right) \frac{\Delta x_1}{x_1} + \left(x_1 \cdot \frac{x_2}{x_1 \cdot x_2}\right) \frac{\Delta x_2}{x_2}.$$

Die Verstärkungsfaktoren sind somit beide 1 unabhängig von x_1, x_2 . Die Multiplikation ist eine gut konditionierte Operation!

c) Schließlich betrachten wir noch die Funktion $F(x_1, x_2) = x_1^2 - x_2^2$. Offensichtlich gilt $\frac{\partial F}{\partial x_1} = 2x_1$, $\frac{\partial F}{\partial x_2} = -2x_2$ und damit nach obiger Formel

$$\frac{F(x_1 + \Delta x_1, x_2 + \Delta x_2) - F(x_1, x_2)}{F(x_1, x_2)} = \left(2x_1 \cdot \frac{x_1}{x_1^2 - x_2^2}\right) \frac{\Delta x_1}{x_1} + \left(2x_2 \cdot \frac{x_2}{x_2^2 - x_1^2}\right) \frac{\Delta x_2}{x_2}.$$

3.4 Rundungsfehleranalyse

In der eigentlichen Rundungsfehleranalyse betrachten wir entsprechend der Aufspaltung (3.3) die Differenz $F(x) - F'(x)$ für Maschinenzahlen $x \in \mathbb{F}$. Entspricht F genau einer Rechenoperation \circledast so gilt wegen der exakten Rundung

$$\frac{(x * y) - (x \circledast y)}{x * y} = \epsilon \quad |\epsilon| \leq \text{eps} = \frac{1}{2}\beta^{1-r}$$

Der genaue Rundungsfehler ϵ hängt von den Argumenten x und y ab und ist damit für jede Operation verschieden. Umstellen der letzten Beziehung liefert

$$x \circledast y = (x * y)(1 + \epsilon) \quad \text{für ein } |\epsilon(x, y)| \leq \text{eps}$$

In der Rundungsfehleranalyse lassen sich ebenfalls Verstärkungsfaktoren ähnlich wie in der Konditionsanalyse herleiten. Dies illustrieren wir mit Beispielen.

Beispiel 3.8. Wir untersuchen die Abbildung $F(x_1, x_2) = x_1^2 - x_2^2$ und zwei verschiedenen Realisierungen

$$F_a(x_1, x_2) = (x_1 \odot x_1) \ominus (x_2 \odot x_2), \quad F_b(x_1, x_2) = (x_1 \ominus x_2) \odot (x_1 \oplus x_2).$$

a) In diesem Fall erhalten wir für die ersten beiden Operationen

$$\begin{aligned} u &= x_1 \odot x_1 = x_1^2(1 + \epsilon_1) & |\epsilon_1| \leq \text{eps}, \\ v &= x_2 \odot x_2 = x_2^2(1 + \epsilon_2) & |\epsilon_2| \leq \text{eps} \end{aligned}$$

und dann

$$\begin{aligned} F_a(x_1, x_2) &= u \ominus v \\ &= (x_1^2(1 + \epsilon_1) - x_2^2(1 + \epsilon_2))(1 + \epsilon_3) \\ &= (x_1^2 + \epsilon_1 x_1^2 - x_2^2 - \epsilon_2 x_2^2)(1 + \epsilon_3) \\ &= x_1^2 - x_2^2 + (\epsilon_1 + \epsilon_3)x_1^2 - (\epsilon_2 + \epsilon_3)x_2^2 + \epsilon_1 \epsilon_3 x_1^2 - \epsilon_2 \epsilon_3 x_2^2. \end{aligned}$$

Damit erhalten wir in erster Näherung für den relativen Rundungsfehler

$$\frac{F_a(x_1, x_2) - F(x_1, x_2)}{F(x_1, x_2)} \doteq \frac{x_1^2}{x_1^2 - x_2^2}(\epsilon_1 + \epsilon_3) + \frac{x_2^2}{x_2^2 - x_1^2}(\epsilon_2 + \epsilon_3).$$

Ein Vergleich mit Beispiel 3.7 ergibt bis auf Konstanten die gleichen Verstärkungsfaktoren.

b) Für die zweite Variante ergibt sich

$$\begin{aligned} u &= x_1 \ominus x_2 = (x_1 - x_2)(1 + \epsilon_1) & |\epsilon_1| \leq \text{eps}, \\ v &= x_1 \oplus x_2 = (x_1 + x_2)(1 + \epsilon_2) & |\epsilon_2| \leq \text{eps} \end{aligned}$$

und dann

$$\begin{aligned} F_b(x_1, x_2) &= u \odot v \\ &= (u \cdot v)(1 + \epsilon_3) \\ &= ((x_1 - x_2)(1 + \epsilon_1) \cdot (x_1 + x_2)(1 + \epsilon_2))(1 + \epsilon_3) \\ &= (x_1 - x_2)(x_1 + x_2)(1 + \epsilon_1)(1 + \epsilon_2)(1 + \epsilon_3) \\ &= (x_1^2 - x_2^2)(1 + \epsilon_1 + \epsilon_2 + \epsilon_3 + \dots \epsilon_1 \epsilon_2 \epsilon_3). \end{aligned}$$

Damit erhalten wir in erster Näherung für den relativen Rundungsfehler

$$\frac{F_b(x_1, x_2) - F(x_1, x_2)}{F(x_1, x_2)} \doteq \epsilon_1 + \epsilon_2 + \epsilon_3.$$

Hier ist der Verstärkungsfaktor 1!. □

In der Gesamtschau sind Konditionsanalyse und Rundungsfehleranalyse gemeinsam zu betrachten. Dazu die folgende

Definition 3.9. Wir nennen einen numerischen Algorithmus *numerisch stabil*, wenn die Verstärkungsfaktoren aus der Rundungsfehleranalyse die aus der Konditionsanalyse nicht übersteigen.

Nach dieser Definition sind beide Realisierungen aus Beispiel 3.8 numerisch stabil.

3.5 Die quadratische Gleichung

Als ein weiteres, wichtiges Beispiel betrachten wir die Lösung der quadratischen Gleichung mit der p, q -Formel. Dieses Beispiel zeigt auch, dass die Minimierung von Rundungsfehlern für jedes Problem speziell betrachtet werden muss.

Die Gleichung

$$x^2 - px + q = 0$$

hat für $p^2/4 > q \neq 0$ die beiden reellen und verschiedenen Lösungen

$$x_{1/2} = f_{\pm}(p, q) = \frac{p}{2} \pm \sqrt{\frac{p^2}{4} - q}.$$

Die *Konditionsanalyse* der (beiden!) Abbildungen $f_{\pm}(p, q)$ liefert (Berechnung der partiellen Ableitungen):

$$\begin{aligned} \frac{f_{\pm}(p + \Delta p, q + \Delta q) - f(p, q)}{f(p, q)} &= \\ \left(1 \pm \frac{p}{2\sqrt{\frac{p^2}{4} - q}}\right) \frac{p}{p \pm 2\sqrt{\frac{p^2}{4} - q}} \frac{\Delta p}{p} &\mp \frac{q}{\sqrt{\frac{p^2}{4} - q}} \left(p \pm 2\sqrt{\frac{p^2}{4} - q}\right) \frac{\Delta q}{q}. \end{aligned}$$

Daraus ersehen wir die folgenden Fälle

- 1) Für $\frac{p^2}{4} \rightarrow q$ sind beide Lösungen schlecht konditioniert, da beide Verstärkungsfaktoren unweigerlich groß werden.
- 2) Für $q \rightarrow 0$ und $|p|$ weg von der Null kommt es auf das Vorzeichen von p an. Für $p < 0$ ist die negative Lösung $f_{-}(p, q) = \frac{p}{2} - \sqrt{\frac{p^2}{4} - q}$ in jedem Fall gut konditioniert und für $p > 0$ ist die positive Lösung $f_{+}(p, q) = \frac{p}{2} + \sqrt{\frac{p^2}{4} - q}$ in jedem Fall gut konditioniert.

Für die *Rundungsfehler* ist die Situation ähnlich. Für den ersten Fall oben kann man Probleme nicht vermeiden, es tritt unweigerlich Auslösung ein. Im zweiten Fall ist es vermeidet man Auslöschung in dem man die jeweils andere Lösung über den “Satz von Vieta”

$$x_1 + x_2 = p, \quad x_1 \cdot x_2 = q,$$

ermittelt. D.h. man berechnet zunächst

$$w = \sqrt{(p \odot p) \oslash 4 \ominus q}.$$

Im Fall $p < 0$ rechnet man dann

$$x_2 = p \oslash 2 \ominus w, \quad x_1 = q \oslash x_2$$

und im Fall $p > 0$ entsprechend

$$x_1 = p \oslash 2 \oplus w, \quad x_2 = q \oslash x_1.$$

3.6 Auslöschung

Aus Beispiel 3.7 a) und Beispiel 3.8 a) entnehmen wir, dass die Addition (bzw. Subtraktion) die gefährlichen Operationen sind (da $x - y = x + (-y)$ sind Addition und Subtraktion keine verschiedenen Operationen). Das Grundübel ist die sogenannte Auslöschung, die immer dann auftritt wenn annähernd gleiche Zahlen von einander subtrahiert werden, bzw. wenn betragsmäßig annähernd gleiche Zahlen mit verschiedenen Vorzeichen addiert werden. Dabei treten Probleme erst dann auf, wenn die beiden Operanden selbst schon fehlerbehaftet sind (was aber in der Regel bei längeren Rechnungen der Fall ist). Wir illustrieren das mit einem Beispiel.

Beispiel 3.10 (Zur Auslöschung). a) Für zwei beliebige *Maschinenzahlen* $x_1, x_2 \in \mathbb{F}$ gilt (unter der Bedingung $x_1 - x_2 \neq 0$) wegen der exakten Rundung der Fließkommaarithmetik (3.2):

$$\left| \frac{(x_1 \ominus x_2) - (x_1 - x_2)}{x_1 - x_2} \right| \leq \text{eps.}$$

b) Auslöschung tritt erst auf, wenn die beiden Argumente *selbst schon mit Fehlern behaftet sind*. Woher diese Fehler kommen ist dabei unerheblich. So kann in Beispiel 3.8 a) der Verstärkungsfaktor sehr groß werden wenn $|x_1 - x_2|$ klein ist. Sind $x_1 = m_1 \beta^e, x_2 = (m_1 - \beta^{-r}) \beta^e$ Maschinenzahlen, ist ein Verstärkungsfaktor der Größenordnung

$$\frac{x_1^2}{x_1^2 - x_2^2} = \frac{x_1^2}{(x_1 - x_2)(x_1 + x_2)} \approx \frac{m_1^2 \beta^{2e}}{\beta^{-r} \beta^e 2m_1 \beta^e} \approx \beta^{r-1}$$

möglich!

c) Betrachte die Berechnung $F'(\text{rd}(x_1), \text{rd}(x_2)) = \text{rd}(x_1) \ominus \text{rd}(x_2)$ in $\mathbb{F}(10, 4, 1)$ an einem konkreten Beispiel. Als Eingabe seien $x_1 = 0.11258762 \cdot 10^2$ und $x_2 = 0.11244891 \cdot 10^2$ gegeben. Offensichtlich gilt $x_1, x_2 \neq \mathbb{F}(10, 4, 1)$ und wir betrachten den relativen Fehler *inklusive Rundung* der Eingaben. Als exaktes Ergebnis erhalten wir

$$x_1 - x_2 = 0.13871 \cdot 10^{-1}.$$

Die (kaufmännische) Rundung der Eingaben liefert

$$\text{rd}(x_1) = 0.1126 \cdot 10^2, \quad \text{rd}(x_2) = 0.1124 \cdot 10^2$$

und die anschließende Fließkommaoperation liefert

$$\text{rd}(x_1) \ominus \text{rd}(x_2) = 0.2 \cdot 10^{-1}$$

wobei im übrigen *kein* weiterer Rundungsfehler eingeführt wird! Trotzdem ist keine Stelle im Gesamtergebnis korrekt und es ergibt sich als relativer Gesamtfehler

$$\frac{0.2 \cdot 10^{-1} - 0.13871 \cdot 10^{-1}}{0.13871 \cdot 10^{-1}} \approx 0.44 \approx 883 \frac{10^{-3}}{2} = 883 \text{ eps.}$$

□

Als Regel kann man sich merken: *Setze die potentiell gefährlichen Operationen \oplus, \ominus möglichst früh ein.*

Vorlesung 4

Analyse von Netzwerken

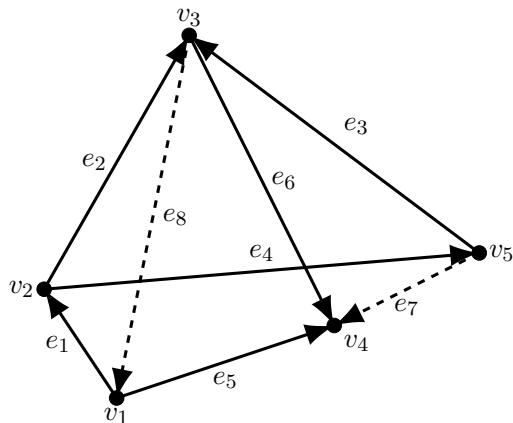
In dieser Vorlesung kommen wir noch einmal auf den Aspekt der Modellbildung zurück. Um die Wichtigkeit der Lösung (großer) linearer Gleichungssysteme zu demonstrieren stellen wir das Knotenpotentialverfahren zur Analyse der Flüssen in Netzwerken vor.

4.1 Rohrleitungsnetze

Wir betrachten ein Netzwerk wie es in Städten zur Wasser-, Gas- oder Stromversorgung vorkommt. Um ein konkretes Beispiel vor Augen zu haben beschränken wir uns auf das Wasserleitungsnetzwerk, das Verfahren kann aber auf die anderen Anwendungen übertragen werden.

Ein Wasserleitungsnetzwerk besteht aus Rohren unterschiedlichen Durchmessers, welche an Verbindungsstellen miteinander verknüpft sind. Wir nehmen an, dass die Rohre vollständig mit Wasser gefüllt sind und dass die Menge an Wasser in den Verbindungsstellen vernachlässigbar ist. Zusätzlich zu den normalen Rohrleitungen gibt es noch Pumpen die Wasser von einer Verbindungsstelle zu einer anderen Verbindungsstelle pumpen.

Abstrakt wird ein Wasserleitungsnetzwerk als ein Graph dargestellt. Folgende Abbildung gibt ein Beispiel:



Ein Graph $G = (V, E)$ besteht aus einer Menge von Knoten V und einer Menge von Kanten $E \subseteq V \times V$. Eine (gerichtete) Kante $e = (v, w)$ verbindet den Knoten v mit dem Knoten w . An einem Knoten werden mindestens zwei Rohre bzw. Pumpen miteinander verknüpft (d.h. wir erlauben keine Graphen mit Knoten an denen keine oder nur eine Kante anliegt). Wir unterscheiden außerdem zwei Arten von Kanten $E = E_R \cup E_P$, Rohre und Pumpen. Die Kanten sind gerichtet und es darf höchstens eine Kante zwischen zwei Knoten geben. Die Orientierung von Kanten ist wichtig um zu kodieren in welche Richtung das Wasser fließt (das Wasser kann aber in beide Richtungen fließen). Mit $n = \#V$ bezeichnen wir die Anzahl der Knoten in V und mit $m = \#E$

die Anzahl der Kanten, wobei $m = m_R + m_P$ die Anzahl Rohre und Anzahl Pumpen bezeichnet. In dem Beispiel oben gibt es fünf Knoten und acht Kanten. Die ersten sechs Kanten entsprechen Rohrleitungen und die Kanten e_7, e_8 entsprechen Pumpen.

Hagen-Poiseuille Gesetz Die Strömung in *einem* Rohr $e = (v, w) \in E_R$ wird in der Strömungsmechanik durch das Gesetz von Hagen-Poiseuille beschrieben:

$$q_e = \frac{\pi r_e^4}{8\eta l_e} \Delta p_e, \quad L_e = \frac{\pi r_e^4}{8\eta l_e} \text{ "Leitfähigkeit"}, \quad (4.1)$$

mit den folgenden Größen:

q_e Gerichteter Volumenstrom in m^3/s . Das Vorzeichen von q_e kodiert die Richtung in die das Wasser fließt: $q_e > 0$ bedeutet, das Wasser fließt in Richtung der Orientierung der Kante e , $q_e < 0$ bedeutet das Wasser fließt entgegen der Orientierung der Kante e .

Δp_e Gerichtete Potentialdifferenz (oder auch manchmal auch Druckdifferenz) $\Delta p_e = p_v - p_w$ mit den Knotenpotentialen (oder Drücken) p_v und p_w in Pa . Δp_e hat das selbe Vorzeichen wie q_e , d.h. ein positiver Volumenstrom entspricht einem Fluss von hohem Potential zu niedrigem Potential.

r_e Radius des Rohres in m .

l_e Länge des Rohres in m .

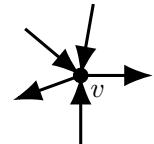
η dynamische Viskosität von Wasser in $Pa \cdot s$.

Ziel des nun folgenden Verfahrens ist die Bestimmung aller Knotendrücke p_v sowie aller Kantenflüsse q_e aus den Daten r_e, l_e der einzelnen Rohre und den Stärken der Pumpen.

4.2 Kirchhoff'sche Gesetze

Knotenregel (1. Kirchhoff'sches Gesetz) Für jeden Knoten $v \in V$ definieren wir die ausgehenden Kanten E_v^+ sowie die eingehenden Kanten E_v^- als

$$E_v^+ = \{(u, w) \in E : u = v\}, \quad E_v^- = \{(u, w) \in E : w = v\}.$$



Dann gilt die Knotenregel

$$\sum_{e \in E_v^+} q_e - \sum_{e \in E_v^-} q_e = 0. \quad (4.2)$$

Die Knotenregel ist eine Folge der Massenerhaltung, denn in einem Knoten wird kein Wasser gespeichert. Alles Wasser das über ein Rohr in den Knoten hinein fließt muss ihn sofort über einen anderes Rohr wieder verlassen.

Die Gleichung (4.2) stellt eine lineare Beziehung zwischen den Kantenflüssen dar. Allerdings sind nur $n - 1$ dieser Beziehungen linear unabhängig. Denn

angenommen es gilt die Knotenregel für alle Knoten $v \in V \setminus \{w\}$ ausser im Knoten w , dann folgt

$$\begin{aligned} 0 &= \sum_{v \in V \setminus \{w\}} \left(\sum_{e \in E_v^+} q_e - \sum_{e \in E_v^-} q_e \right) = \sum_{v \in V \setminus \{w\}} \sum_{e \in E_v^+} q_e - \sum_{v \in V \setminus \{w\}} \sum_{e \in E_v^-} q_e \\ &= \sum_{e \in E_w^-} q_e - \sum_{e \in E_w^+} q_e, \end{aligned}$$

also die Knotenregel (mal -1) im Knoten w . Denn für jede Kante $e = (u, v)$ gilt genau einer der drei folgenden Fälle:

- i) Es ist $u \neq w \wedge v \neq w$. Dann kommt q_e in jeder der beiden Summen am Ende der ersten Zeile genau einmal vor, einmal mit positivem und einmal mit negativem Vorzeichen. Somit kann man die beiden Flüsse weglassen.
- ii) Es ist $u = w \wedge v \neq w$. Damit kommt q_e in den Summen genau einmal vor und zwar in E_v^+ als eingehende Kante, somit mit negativem Vorzeichen.
- iii) Es ist $u \neq w \wedge v = w$. Damit kommt q_e in den Summen genau einmal vor und zwar in E_u^- als ausgehende Kante, somit mit positivem Vorzeichen.

Maschenregel (2. Kirchhoff'sches Gesetz) Es sei $C \subseteq E$ eine sogenannte Masche, d.h.

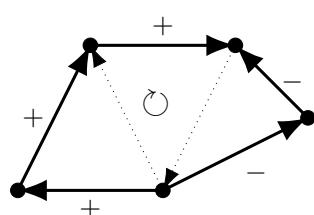
- 1) Die Kanten in C bilden einen geschlossenen Pfad.
- 2) Dem Pfad C wird ein Umlaufsinn zugeordnet und wir setzen

$$\begin{aligned} C^+ &= \{e \in C : e \text{ und } C \text{ gleich orientiert}\}, \\ C^- &= \{e \in C : e \text{ und } C \text{ verschieden orientiert}\}. \end{aligned}$$

Dann gilt längs so einen Pfad die Maschenregel für die gerichteten Potentialdifferenzen:

$$\sum_{e \in C^+} \Delta p_e - \sum_{e \in C^-} \Delta p_e = 0. \quad (4.3)$$

Die Maschenregel ist eine Folge der Energieerhaltung. Die Größe $\int_v^w \Delta p_e dA$ ist eine Energie. Würde die Maschenregel nicht gelten wäre Energiegewinnung durch "im Kreis laufen" möglich.



Die Abbildung links zeigt eine Masche bestehend aus fünf Kanten. Davon haben drei dieselbe Orientierung wie die Masche und zwei die entgegengesetzte. Jede mögliche Masche in einem Graphen liefert somit eine lineare Gleichung der Form (4.3) für die Potentialdifferenzen. Allerdings zeigt sich, dass nur maximal $n-1$ dieser Gleichungen linear unabhängig sind. Somit sind die Maschen in der Praxis geeignet auszuwählen.

4.3 Knotenpotentialverfahren

Das Knotenpotentialverfahren nutzt das Gesetz von Hagen-Poiseuille und die beiden Kirchhoff'schen Gesetze um die Druckdifferenzen und Flüsse systematisch zu bestimmen.

Zunächst setzen wir, wie bereits oben bemerkt, für jede Kante $e = (v, w) \in E$

$$\Delta p_e = p_v - p_w, \quad e \in E \quad (4.4)$$

dabei ist p_v das *Potential im Knoten* v .

Mit dieser Wahl ist die Maschenregel für jede Masche automatisch erfüllt. Denn sei $v \in V$ ein Knoten welcher in der Masche vorkommt, d.h. zwei Kanten der Masche liegen an v an. Sind beide Kanten mit der Masche orientiert so hebt sich p_v in der Gleichung auf. Analog gilt dies auch für alle anderen Fälle der Orientierung.

Da in der Maschenregel bzw. im Gesetz von Hagen-Poiseuille nur Potentialdifferenzen auftauchen, kann das Potential in einem Knoten, dem sogenannten *Referenzknoten* beliebig festgelegt werden. D.h. wir wählen o.B.d.A. $r = v_n$ und setzen $p_{v_n} = 0$. Die Potentiale in den Knoten v_1, \dots, v_{n-1} sind die zu bestimmenden unbekannten Größen welche wir in einem Vektor $p \in \mathbb{R}^{n-1}$ zusammenfassen:

$$p = (p_{v_1}, \dots, p_{v_{n-1}})^T. \quad (4.5)$$

Für alle Kanten $e = (v_i, v_j) \in E_R$, also der *Menge der Rohre* (!), können die Potentialdifferenzen ebenfalls in einem Vektor Δp zusammengefasst werden:

$$\Delta p = (\Delta p_{e_1}, \dots, \Delta p_{e_{m_R}})^T, \quad (4.6)$$

wobei wir die Rohre von 1 bis m_R nummeriert haben.

Nun können die Gleichungen (4.4) in Matrixform zusammengefasst werden:

$$\Delta p = Bp \quad (4.7)$$

mit der Matrix $B \in \mathbb{R}^{m_R \times n-1}$. Jede Kante $e \in E_R$ liefert dabei eine Zeile von B , d.h. die Einträge von B werden wie folgt bestimmt:

$$(B)_{ij} = \begin{cases} +1 & e_i = (v, w) \in E_R \wedge v = v_j, \\ -1 & e_i = (v, w) \in E_R \wedge w = v_j, \\ 0 & \text{sonst.} \end{cases} \quad (4.8)$$

Beachte, dass $1 \leq j < n$ da der Referenzknoten v_n das Potential Null hat und nicht vorkommt. Die Matrix B ist dünn besetzt, da jede Zeile höchstens zwei von Null verschiedene Elemente hat.

Das Gesetz von Hagen-Poiseuille (4.1) kann ebenfalls in Matrixform formuliert werden:

$$q = L\Delta p \quad (4.9)$$

wobei die Einträge der Diagonalmatrix $L \in \mathbb{R}^{m_R \times m_R}$ die Leitfähigkeiten $(L)_{ii} = L_{e_i}$ sind und der Vektor

$$q = (q_{e_1}, \dots, q_{e_{m_R}})^T \quad (4.10)$$

die gerichteten Kantenflüsse durch alle Rohre zusammenfasst.

Schließlich bleibt noch die Knotenregel zu berücksichtigen. Hier spielen die Pumpen nun eine zentrale Rolle. Wie oben bemerkt sind auch nur $n - 1$ Knotenregeln linear unabhängig, d.h. wir lassen den Referenzknoten v_n unberücksichtigt. Für jedes $v \in V \setminus \{v_n\}$ spalten wir die Knotenregel in Rohre in Pumpen auf und erhalten

$$\sum_{e \in E_v^+} q_e - \sum_{e \in E_v^-} q_e = \sum_{e \in E_v^+ \cap E_R} q_e - \sum_{e \in E_v^- \cap E_R} q_e + \sum_{e \in E_v^+ \cap E_P} q_e - \sum_{e \in E_v^- \cap E_P} q_e.$$

Da die (gerichteten) Pumpenströme $q_e, e \in E_P$ gegeben sind können wir diese auf die rechte Seite schaffen und erhalten für jedes $v \in V \setminus \{v_n\}$:

$$\sum_{e \in E_v^+ \cap E_R} q_e - \sum_{e \in E_v^- \cap E_R} q_e = \sum_{e \in E_v^- \cap E_P} q_e - \sum_{e \in E_v^+ \cap E_P} q_e =: b_v. \quad (4.11)$$

Auch diese Gleichungen können in Matrixform geschrieben werden:

$$\tilde{B}q = b \quad (4.12)$$

mit dem Vektor

$$b = (b_{v_1}, \dots, b_{v_{n-1}})^T \in \mathbb{R}^{n-1} \quad (4.13)$$

und der $n - 1 \times m_R$ -Matrix

$$(\tilde{B})_{ij} = \begin{cases} +1 & e_j \in E_{v_i}^+ \cap E_R, \\ -1 & e_j \in E_{v_i}^- \cap E_R, \\ 0 & \text{sonst.} \end{cases} \quad (4.14)$$

Ein Vergleich mit (4.8) zeigt, dass $\tilde{B} = B^T$!

Setzt man die Beziehungen (4.12), (4.9) und (4.7) ineinander ein, so erhält man

$$b = B^T q = B^T L \Delta p = B^T L B p,$$

also ein lineares Gleichungssystem

$$Ap = b \quad (4.15)$$

mit der $n - 1 \times n - 1$ -Matrix $A = B^T L B$. Die Matrix A hat folgende interessante Eigenschaften:

- 1) A ist symmetrisch, d.h. $(A)_{ij} = (A)_{ji}$. Dies folgt unmittelbar aus der Darstellung $A = B^T L B$ und der Tatsache dass L eine Diagonalmatrix ist.
- 2) A ist positiv definit, d.h. $x^T A x > 0$ für alle $x \neq 0$. Dies folgt aus der linearen Unabhängigkeit der Spalten von B und der Tatsache dass $L_{ii} > 0$, denn sei $x \neq 0$ so gilt $y = Bx \neq 0$ genau wenn die Spalten von B linear unabhängig sind. Damit gilt $x^T A x = y^T Ly = \sum_i L_{ii} y_i^2 > 0$.
- 3) A ist eine dünn besetzte Matrix, d.h. es sind nur sehr wenige Einträge, typischerweise $O(n)$ ungleich Null.

4.4 Zusammenfassung

In dieser Vorlesung haben wir ein Verfahren zur systematischen Analyse der Strömung in Rohrleitungsnetzwerken kennen gelernt. Die Analyse führt auf die Lösung linearer Gleichungssysteme mit möglicherweise sehr vielen Unbekannten.

Das Verfahren lässt sich übertragen auf andere Situationen, z.B. die Analyse elektrischer Netzwerke bestehend aus Widerständen und Stromquellen. Dann gelten folgende Entsprechungen:

q_e	elektrischer Strom durch eine Kante
p_v	elektrische Spannung am Knoten relativ zur Masse
Hagen-Poiseuille	Ohmsches Gesetz

Eine Erweiterung dieses Verfahrens auf Netzwerke aus Widerständen, Spulen und Kondensatoren (sogenannte RLC-Netzwerke) mit harmonischer (sinusförmiger) Anregung führt auf komplexwertige, lineare Gleichungssysteme.

Vorlesung 5

Einige Grundlagen aus der linearen Algebra

Bevor wir uns intensiv mit der numerischen Lösung linearer Gleichungssysteme beschäftigen wollen wir einige Grundlagen aus der Linearen Algebra und (weniger) der Analysis wiederholen. Dabei liegt der Fokus auf Konzepten die für die Numerik besonders wichtig sind, wie Normen, sowie die Verbindung von Konzepten aus Numerik und Analysis. Die folgenden Begriffe setzen wir als bekannt voraus. Rufen sie sich deren Bedeutung ins Gedächtnis wenn sie unsicher sind:

- Vektorraum über einem Körper \mathbb{K} , wobei hier nur $\mathbb{K} = \mathbb{R}$ oder $\mathbb{K} = \mathbb{C}$ betrachtet wird. Wir werden in dieser Vorlesung nur endlichdimensionale Vektorräume $V = \mathbb{K}^n$, $n \in \mathbb{N}$ betrachten.
- Linearkombination und lineare Unabhängigkeit.
- Basis und Dimension. Wir werden uns in dieser Vorlesung nur mit endlichdimensionalen Vektorräumen beschäftigen.
- Untervektorraum und Summe von Vektorräumen.
- Lineare Abbildung zwischen Vektorräumen (Homomorphismen).
- Bild und Kern einer linearen Abbildung.
- Die Begriffe injektiv, surjektiv und bijektiv.
- Matrizen als Darstellung linearer Abbildungen zwischen Vektorräumen.
- Rang, Determinante und Inverse einer Matrix.

Wir wollen in diesem Kapitel insbesondere Zusammenhänge zwischen linearer Algebra und Analysis herstellen und auf Aspekte eingehen die für die Numerik besonders relevant sind.

5.1 Normen für Vektoren

Ein zentrales Anliegen der numerischen Analysis ist die Herleitung von Fehlerabschätzungen. Hierfür benötigt man in der Regel Normen.

Definition 5.1. Sei V ein Vektorraum über \mathbb{K} . Eine Abbildung $\|\cdot\| : V \rightarrow \mathbb{R}_0^+ = \{x \in \mathbb{R} : x \geq 0\}$ heißt Norm auf V , falls gilt:

$$\|x\| > 0 \quad V \ni x \neq 0 \quad (\text{Definitheit}), \quad (5.1a)$$

$$\|\alpha x\| = |\alpha| \|x\| \quad x \in V, \alpha \in \mathbb{K} \quad (\text{positive Homogenität}), \quad (5.1b)$$

$$\|x + y\| \leq \|x\| + \|y\| \quad x, y \in V \quad (\text{Dreiecksungleichung}). \quad (5.1c)$$

Beispiel 5.2. Häufig verwendete Normen für $V = \mathbb{K}^n$ sind

$$\|x\|_2 = \left(\sum_{i=1}^n |x_i|^2 \right)^{\frac{1}{2}} \quad \text{Euklidsche Norm, } l_2\text{-Norm,} \quad (5.2a)$$

$$\|x\|_1 = \sum_{i=1}^n |x_i| \quad l_1\text{-Norm,} \quad (5.2b)$$

$$\|x\|_\infty = \max_{i=1,\dots,n} |x_i| \quad \text{Maximumnorm, } l_\infty\text{-Norm.} \quad (5.2c)$$

Mit Hilfe von Normen lässt sich der Begriff der Konvergenz aus der Analysis auf Folgen von Vektoren (bzw. Elementen eines Vektorraumes) übertragen.

Definition 5.3 (Konvergenz, Vollständigkeit). Eine Folge $\{x^{(k)}\}_{k=1}^\infty$ von Elementen eines Vektorraumes V konvergiert gegen $x \in V$, falls

$$\forall \epsilon > 0 \exists N \in \mathbb{N} \forall k \geq N : \|x - x^{(k)}\| < \epsilon,$$

d.h. die Folge der Zahlen $z^{(k)} = \|x - x^{(k)}\|$ konvergiert gegen Null. Diese Formulierung der Konvergenz setzt die Kenntnis des Grenzelementes x voraus.

Eine Folge $\{x^{(k)}\}_{k=1}^\infty$ von Elementen eines Vektorraumes V heisst *Cauchy-Folge* falls gilt

$$\forall \epsilon > 0 \exists N \in \mathbb{N} \forall k, l \geq N : \|x^{(k)} - x^{(l)}\| < \epsilon.$$

Ein Vektorraum V heisst *vollständig* wenn jede Cauchy-Folge gegen ein Element aus V konvergiert. \square

Die Vollständigkeit des \mathbb{K}^n ergibt sich aus der Vollständigkeit von \mathbb{K} .

Eine weitere wichtige Folgerung ist die

Lemma 5.4 (Inverse Dreiecksungleichung). Für alle $x, y \in V$ gilt

$$\|x - y\| \geq \|\|x\| - \|y\|\|.$$

Beweis. Es gilt

$$\|x\| = \|x - y + y\| \leq \|x - y\| + \|y\| \Rightarrow \|x - y\| \geq \|x\| - \|y\|.$$

Analog zeigt man durch vertauschen von x und y :

$$\|y\| = \|y - x + x\| \leq \|y - x\| + \|x\| \Rightarrow \|x - y\| \geq \|y\| - \|x\| = -(\|x\| - \|y\|).$$

Aus beiden Abschätzung zusammen folgt die Behauptung. \square

Aus der inversen Dreiecksungleichung folgt man die Stetigkeit der Norm,

$$x^{(k)} \rightarrow x \Rightarrow \|x^{(k)}\| \rightarrow \|x\|,$$

lese dazu die inverse Dreiecksungleichung von ‘rechts nach links’.

Die Definition der Konvergenz von Folgen 5.3 basiert auf dem Normbegriff. Nun gibt es ja diverse Normen und damit stellt sich die Frage ob man für unterschiedliche Normen einen unterschiedlichen Konvergenzbegriff erhält. Dass dies für den \mathbb{K}^n *nicht* der Fall ist zeigt der folgende Satz.

Satz 5.5 (Äquivalenz aller Normen). Auf \mathbb{K}^n sind alle Normen in folgendem Sinne äquivalent. Zu je zwei Normen $\|\cdot\|$ und $\|\cdot\|'$ gibt es Zahlen $m, M > 0$ aus \mathbb{R} so dass gilt

$$m\|x\|' \leq \|x\| \leq M\|x\|' \quad \forall x \in \mathbb{K}^n.$$

Beweis. Es genügt die Äquivalenz aller Normen zur Maximumnorn $\|\cdot\|_\infty$ zu zeigen. Wir definieren die Menge von Vektoren

$$S = \{x \in \mathbb{K}^n : \|x\|_\infty = 1\} \subset \mathbb{K}^n,$$

auch als ‘‘Einheitskugel bezüglich $\|\cdot\|_\infty$ ’’ bezeichnet. Die Menge S ist beschränkt (die Norm ist ja gerade 1) und abgeschlossen (Grenzwerte von Folgen werden angenommen) im \mathbb{K}^n . Daher nimmt die stetige Funktion $\|\cdot\| : S \rightarrow \mathbb{R}_0^+$ ihr Minimum und Maximum an, d.h. es existieren $x^{\min}, x^{\max} \in S$ so dass

$$0 < \|x^{\min}\| \leq \|x\| \leq \|x^{\max}\| \quad \forall x \in S.$$

Beachte dass $x \in S \Rightarrow x \neq 0$. Für ein beliebiges $x \in \mathbb{K}^n, x \neq 0$ gilt nun $\|x\|_\infty^{-1}x \in S$ und somit

$$0 < \|x^{\min}\| \leq \|\|x\|_\infty^{-1}x\| = \|x\|_\infty^{-1}\|x\| \leq \|x^{\max}\| \quad \forall x \in \mathbb{K}^n, x \neq 0.$$

Damit folgt

$$\|x^{\min}\|\|x\|_\infty \leq \|x\| \leq \|x^{\max}\|\|x\|_\infty \quad \forall x \in \mathbb{K}^n, x \neq 0.$$

□

5.2 Normen für Matrizen

Wir betrachten nun Matrizen $A \in \mathbb{K}^{m \times n}$. Diese können zum einen als Schema von mn Zahlen interpretiert werden

$$A = \begin{bmatrix} a_{1,1} & \dots & a_{1,n} \\ \vdots & & \vdots \\ a_{m,1} & \dots & a_{m,n} \end{bmatrix}, \quad a_{i,j} \in \mathbb{K}$$

und zum anderen als lineare Abbildung zwischen Vektorräumen

$$F : \mathbb{K}^n \rightarrow \mathbb{K}^m, \quad F(x) = Ax.$$

Die Menge der Matrizen mit der Addition von Matrizen und der skalaren Multiplikation stellt selbst wieder einen Vektorraum dar, der mit dem \mathbb{K}^{mn} identifiziert werden kann. Insofern kann mittels jeder Norm auf dem Vektorraum \mathbb{K}^{mn} eine Norm auf dem Vektorraum der Matrizen definiert werden.

Andererseits bringt die Sicht als lineare Abbildung den Definitionsräum \mathbb{K}^n und den Bildraum \mathbb{K}^m ins Spiel und die Hintereinanderausführung von Abbildungen kann als Multiplikation auf Vektorräumen von Matrizen interpretiert werden. Es ist daher wünschenswert, dass ausser den Normeneigenschaften zusätzliche Eigenschaften im Zusammenspiel der verschiedenen Konzepte gelten.

Definition 5.6. Für alle $n, m \in \mathbb{N}$ sei $\|\cdot\|$ eine Vektornorm auf \mathbb{K}^n und $\|\cdot\|$ eine Matrixnorm auf $\mathbb{K}^{m \times n}$. Aufgrund des Argumentes ist jeweils klar welche Norm gemeint ist.

Die Matrixnorm heißt *verträglich* zur Vektornorm falls gilt

$$\|Ax\| \leq \|A\| \|x\| \quad x \in \mathbb{K}^n, A \in \mathbb{K}^{m \times n} \quad (5.3)$$

und sie heißt *submultiplikativ* falls gilt

$$\|AB\| \leq \|A\| \|B\| \quad A \in \mathbb{K}^{m \times n}, B \in \mathbb{K}^{n \times k}. \quad (5.4)$$

Manchmal wird Submultiplikativität auch als zwingend notwendige Bedingung für eine Matrixnorm gefordert. \square

Beispiel 5.7. Betrachte die *Frobeniusnorm*

$$\|A\|_{\text{Fr}} = \left(\sum_{i=1}^m \sum_{j=1}^n |a_{ij}|^2 \right)^{\frac{1}{2}}.$$

Die Frobeniusnorm ist verträglich mit der Euklidschen Norm. \square

Nun stellt sich die Frage wie man zu einer Norm auf dem \mathbb{K}^n eine geeignete Norm auf $\mathbb{K}^{m \times n}$ findet die obige Bedingungen erfüllen. Antwort gibt

Definition 5.8 (Zugeordnete Matrixnorm). Es sei $\|\cdot\|$ eine Vektornorm auf \mathbb{K}^n für alle $n \in \mathbb{N}$. Dann heißt

$$\|A\| = \sup_{x \in \mathbb{K}^n \setminus \{0\}} \frac{\|Ax\|}{\|x\|}$$

zugeordnete oder *natürliche* Matrixnorm. \square

Die zugeordnete Matrixnorm ist stets verträglich und submultiplikativ.

Die Supremumsbildung lässt sich auch anders formulieren. wegen $\|A(cx)\| = |c| \|Ax\|$ und $\|cx\| = |c| \|x\|$ für jede Norm gilt

$$\sup_{x \in \mathbb{K}^n \setminus \{0\}} \frac{\|Ax\|}{\|x\|} = \sup_{\|x\|=1} \|Ax\|.$$

Auskunft über spezielle zugeordnete Matrixnormen gibt das Lemma

Lemma 5.9. Die zugeordneten Matrixnormen zu $\|\cdot\|_\infty$ und $\|\cdot\|_1$ sind

$$\|A\|_\infty = \max_{1 \leq i \leq m} \sum_{j=1}^n |a_{ij}| \quad (\text{Zeilensummennorm}), \quad (5.5a)$$

$$\|A\|_1 = \max_{1 \leq j \leq n} \sum_{i=1}^m |a_{ij}| \quad (\text{Spaltensummennorm}) \quad (5.5b)$$

Beweis. Es wird nur der Beweis für die (wichtigere) Zeilensummennorm gegeben.

a) Zunächst rechnen wir die Verträglichkeit nach

$$\begin{aligned}\|Ax\|_\infty &= \max_{1 \leq i \leq m} \left| \sum_{j=1}^n a_{ij} x_j \right| \leq \max_{1 \leq i \leq m} \sum_{j=1}^n |a_{ij}| |x_j| \\ &\leq \max_{1 \leq i \leq m} \sum_{j=1}^n |a_{ij}| \left(\max_{1 \leq k \leq n} |x_k| \right) = \|A\|_\infty \|x\|_\infty.\end{aligned}$$

b) Damit gilt nun zunächst mal

$$\sup_{x \in \mathbb{K}^n \setminus \{0\}} \frac{\|Ax\|_\infty}{\|x\|_\infty} \leq \sup_{x \in \mathbb{K}^n \setminus \{0\}} \frac{\|A\|_\infty \|x\|_\infty}{\|x\|_\infty} = \|A\|_\infty.$$

c) Für $A = 0$ gilt trivialerweise $\sup_{x \in \mathbb{K}^n \setminus \{0\}} \frac{\|Ax\|_\infty}{\|x\|_\infty} = \|A\|_\infty = 0$. Sei also $A \neq 0$ und mithin $\|A\|_\infty > 0$. Dann gibt es einen Zeilenindex $k \in \{1, \dots, m\}$ so dass

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

Nun definiere den Vektor z mit den Komponenten

$$z_j = \begin{cases} \frac{|a_{kj}|}{a_{kj}} & \text{falls } a_{kj} \neq 0, \\ 0 & \text{sonst.} \end{cases}$$

Da die Einträge des Vektors z nur die Werte $\{-1, 0, +1\}$ annehmen können und ein a_{kj} ungleich Null sein muss, gilt $\|z\|_\infty = 1$. Für $v = Az$ gilt dann für die k -te Komponente

$$v_k = (Az)_k = \sum_{j=1}^n a_{kj} z_j = \sum_{j=1}^n |a_{kj}| = \|A\|_\infty.$$

Damit gilt

$$\|A\|_\infty = v_k \leq \|v\|_\infty = \|Az\|_\infty \leq \sup_{x \in \mathbb{K}^n \setminus \{0\}} \frac{\|Ax\|_\infty}{\|x\|_\infty}$$

und somit zusammen mit b) die Gleichheit. □

5.3 Eigenwerte und Eigenvektoren

Definition 5.10. Sei $A \in \mathbb{K}^{n \times n}$ mit $\mathbb{K} = \mathbb{R}$ oder $\mathbb{K} = \mathbb{C}$. Die komplexe Zahl $\lambda \in \mathbb{C}$ heißt *Eigenwert* wenn es einen Vektor $e \in \mathbb{K}^n$, $e \neq 0$, gibt für den gilt

$$Ae = \lambda e.$$

e heißt *Eigenvektor* zum Eigenwert λ und (λ, e) heißt auch *Eigenpaar*. □

Zu jedem Eigenwert gibt es mindestens einen Eigenvektor. Mit jedem Eigenvektor e ist auch ce ein Eigenvektor für jedes $c \in \mathbb{K}$, die Eigenvektoren zu einem Eigenwert spannen deshalb einen Unterraum des \mathbb{K}^n auf.

Definition 5.11 (Inverse). Zu $A \in \mathbb{K}^{n \times n}$ nennt man $A^{-1} \in \mathbb{K}^{n \times n}$ *inverse Matrix* wenn gilt

$$AA^{-1} = I$$

mit I der Einheitsmatrix. Die Existenz der Inversen hängt eng mit der Lösbarkeit von linearen Gleichungssystemen zusammen, der wir uns weiter unten widmen. \square

Definition 5.12 (Diagonalisierbarkeit). Gibt es eine Basis bestehend aus Eigenvektoren nennt man A *diagonalisierbar*. Dann existiert eine Matrix $S \in \mathbb{K}^{n \times n}$ so dass

$$S^{-1}AS = \text{diag}(\lambda_1, \dots, \lambda_n) = \begin{pmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_n \end{pmatrix}.$$

Denn man setzt einfach $S = [e_1, \dots, e_n]$ (spaltenweiser Aufbau der Matrix S aus den Vektoren e_i), dann gilt $AS = [\lambda_1 e_1, \dots, \lambda_n e_n] = S \text{diag}(\lambda_1, \dots, \lambda_n)$. Das Produkt $S^{-1}AS$ heißt auch *Ähnlichkeitstransformation*. \square

Die Eigenwerte sind Nullstellen des *charakteristischen Polynoms*

$$p(\lambda) = \det(A - \lambda I) = 0.$$

Nach dem Fundamentalsatz der Algebra hat jedes Polynom vom Grad n genau n Nullstellen in \mathbb{C} , wenn man die Vielfachheiten mitzählt. Bei reellwertigen Matrizen treten komplexe Eigenwerte stets in konjugiert komplexen Paaren auf, d.h. mit $a + ib$ ist auch $a - ib$ ein Eigenwert. In der Praxis bestimmt man die Eigenwerte jedoch nicht als Nullstellen des charakteristischen Polynoms da es hierfür keine stabilen numerischen Algorithmen gibt.

Eigenwerte sind in der Regel schwierig zu berechnen. Folgendes Lemma gibt eine einfache Abschätzung.

Lemma 5.13. Für jede zu einer Vektornorm $\|\cdot\|$ verträglichen Matrixnorm gilt

$$\|A\| \geq \varrho(A) = \max_{1 \leq i \leq n} |\lambda_i|.$$

$\varrho(A)$ heißt *Spektralradius*.

Beweis. Zu $\lambda \in \mathbb{C}$ wähle einen Eigenvektor e mit $\|e\| = 1$. Dies ist immer möglich. Es gilt dann

$$|\lambda| = |\lambda| \|e\| = \|\lambda e\| = \|Ae\| \leq \|A\| \|e\| = \|A\|.$$

Somit gilt $\|A\| \geq |\lambda|$ für jeden Eigenwert und damit auch das Maximum. \square

Einige Klassen von Matrizen sind für die Praxis besonders relevant.

Definition 5.14. Zu $A \in \mathbb{K}^{m \times n}$ heißt A^T *transponierte Matrix* wenn gilt $(A^T)_{ij} = a_{ji}$. Zu $A \in \mathbb{K}^{m \times n}$ heißt \bar{A} *konjugiert komplexe Matrix* wenn gilt $(\bar{A})_{ij} = \bar{a}_{ij}$.

a) $A \in \mathbb{K}^{n \times n}$ heißt *hermitesch* falls $A = \bar{A}^T$ gilt. Manche Autoren kürzen $A^H = \bar{A}^T$ ab. Reelle hermitesche Matrizen heißen *symmetrisch*.

b) Eine komplexwertige Matrix $A \in \mathbb{C}^{n \times n}$ mit

$$A\bar{A}^T = \bar{A}^T A \quad \text{heißt } \textit{normal}, \text{ eine mit} \quad (5.6)$$

$$A^{-1} = \bar{A}^T \quad \text{heißt } \textit{unitär}. \quad (5.7)$$

c) Eine reellwertige Matrix $A \in \mathbb{R}^{n \times n}$ mit

$$A^{-1} = A^T \quad \text{heißt } \textit{orthogonal}. \quad (5.8)$$

Satz 5.15 (Diagonalisierbarkeit hermitescher Matrizen). Eine Matrix $A \in \mathbb{C}^n$ ist genau dann *reell* diagonalisierbar, wenn A hermitesch ist. Die Transformationsmatrix Q ist in diesem Fall sogar unitär, d.h. es gilt $\bar{Q}^T A Q = \text{diag}(\lambda_1, \dots, \lambda_n)$ mit $\lambda_i \in \mathbb{R}$.

Beweis. [Hackbusch, 1991, Satz 2.8.7] \square

Aus diesem Satz folgern wir, dass reelle symmetrische Matrizen stets mit einer orthogonalen Transformationsmatrix reell diagonalisierbar sind.

5.4 Skalarprodukt

Definition 5.16. Sei $\mathbb{K} = \mathbb{C}$ oder $\mathbb{K} = \mathbb{R}$. Eine Abbildung $(., .) : \mathbb{K}^n \times \mathbb{K}^n \rightarrow \mathbb{K}$ heißt *Skalarprodukt* falls sie folgende Eigenschaften besitzt:

$$(x, y) = \overline{(y, x)} \quad x, y \in \mathbb{K}^n \quad (\text{Symmetrie}), \quad (5.9a)$$

$$(\alpha x + \beta y, z) = \alpha(x, z) + \beta(y, z) \quad x, y, z \in \mathbb{K}^n, \quad (\text{Linearität}), \quad (5.9b)$$

$$\alpha, \beta \in \mathbb{K}$$

$$(x, x) > 0 \quad x \in \mathbb{K}^n \setminus \{0\} \quad (\text{Definitheit}). \quad (5.9c)$$

Bemerkung 5.17. Man beachte:

- a) Aus der Symmetrie folgt wegen $(x, x) = \overline{(x, x)}$ dass (x, x) stets reell ist. Somit macht es Sinn die Definitheit zu fordern (auf \mathbb{C} ist kein $>$ definiert).
- b) Die Linearität im zweiten Argument gilt ebenfalls aber in der folgenden Form

$$\begin{aligned} (z, \alpha x + \beta y) &= \overline{(\alpha x + \beta y, z)} = \overline{\alpha(x, z) + \beta(y, z)} = \bar{\alpha}\overline{(x, z)} + \bar{\beta}\overline{(y, z)} \\ &= \bar{\alpha}(z, x) + \bar{\beta}(z, y). \end{aligned}$$

Es werden also die skalaren Faktoren zusätzlich konjugiert. \square

Wegen der Definitheit kann man zu jedem Skalarprodukt in kanonischer Weise eine Norm konstruieren, die sogenannte *induzierte Norm*. Man rechnet nach, dass

$$\|x\| = \sqrt{(x, x)} \quad (5.10)$$

alle Eigenschaften einer Norm erfüllt. Für Skalarprodukt und induzierte Norm gilt stets die *Ungleichung von Cauchy-Schwarz*:

$$|(x, y)| \leq \|x\| \|y\|. \quad (5.11)$$

Diese Ungleichung ist neben der Dreiecksungleichung wohl die am häufigsten verwendete Ungleichung der numerischen Analysis.

Definition 5.18 (Euklidsches Skalarprodukt). Das *Euklidsche Skalarprodukt* in \mathbb{K}^n lautet

$$(x, y)_2 = \sum_{i=1}^n x_i \bar{y}_i. \quad (5.12)$$

□

Damit gilt für $A \in \mathbb{K}^n$

$$(Ax, y)_2 = (x, \bar{A}^T y)_2 \quad \forall x, y \in \mathbb{K}^n \quad (5.13)$$

denn

$$\begin{aligned} (Ax, y)_2 &= \sum_{i=1}^n \left(\sum_{j=1}^n a_{ij} x_j \right) \bar{y}_i = \sum_{i=1}^n \sum_{j=1}^n x_j \bar{a}_{ij} \bar{y}_i \\ &= \sum_{j=1}^n x_j \overline{\left(\sum_{i=1}^n \bar{a}_{ij} y_i \right)} = (x, \bar{A}^T y)_2. \end{aligned}$$

Damit kann man hermitesche Matrizen auch charakterisieren als

$$A = \bar{A}^T \Leftrightarrow (Ax, y)_2 = (x, Ay)_2 \quad \forall x, y \in \mathbb{K}^n. \quad (5.14)$$

Denn die Richtung \Rightarrow folgt aus $(Ax, y)_2 = (x, \bar{A}^T y)_2 = (x, Ay)_2$ und die Richtung \Leftarrow folgt wenn man die Koordinatenvektoren (nicht Eigenvektoren) $x = e_i$ und $y = e_j$ jeweils einsetzt.

Im allgemeinen heißt eine Matrix A^* *adjungiert* zu A bezüglich eines Skalarproduktes $(., .)$ wenn

$$(Ax, y) = (x, A^*) \quad \forall x, y \in \mathbb{K}^n.$$

Gilt $A = A^*$ heißt A *selbstdjungiert* bezüglich $(., .)$. Somit nennt man hermitesche Matrizen auch selbstdjungiert bezüglich des Euklidschen Skalarproduktes.

5.5 Spektralnorm

Oben haben wir in Lemma 5.9 zugeordnete Matrixnormen für die Maximumnorm und die 1-Norm charakterisieren können. Die der Euklischen Norm zugeordnete Matrixnorm

$$\|A\|_2 = \sup_{x \in \mathbb{K}^n \setminus \{0\}} \frac{\|Ax\|_2}{\|x\|_2},$$

die sogenannte *Spektralnorm*, wurde bisher noch nicht untersucht. Das folgende Lemma gibt hierzu Auskunft.

Lemma 5.19. Für hermitesche Matrizen $A \in \mathbb{K}^n$ gilt

$$\|A\|_2 = \varrho(A) = \max_{1 \leq i \leq n} |\lambda_i|,$$

d.h. in diesem Fall stimmt die Spektralnorm mit dem Spektralradius überein. Für jede Matrix $A \in \mathbb{K}^n$ gilt

$$\|A\|_2 = \sqrt{\varrho(\bar{A}^T A)}.$$

Definition 5.18 (Euklidsches Skalarprodukt). Das *Euklidsche Skalarprodukt* in \mathbb{K}^n lautet

$$(x, y)_2 = \sum_{i=1}^n x_i \bar{y}_i. \quad (5.12)$$

□

Damit gilt für $A \in \mathbb{K}^n$

$$(Ax, y)_2 = (x, \bar{A}^T y)_2 \quad \forall x, y \in \mathbb{K}^n \quad (5.13)$$

denn

$$\begin{aligned} (Ax, y)_2 &= \sum_{i=1}^n \left(\sum_{j=1}^n a_{ij} x_j \right) \bar{y}_i = \sum_{i=1}^n \sum_{j=1}^n x_j \bar{a}_{ij} \bar{y}_i \\ &= \sum_{j=1}^n x_j \overline{\left(\sum_{i=1}^n \bar{a}_{ij} y_i \right)} = (x, \bar{A}^T y)_2. \end{aligned}$$

Damit kann man hermitesche Matrizen auch charakterisieren als

$$A = \bar{A}^T \Leftrightarrow (Ax, y)_2 = (x, Ay)_2 \quad \forall x, y \in \mathbb{K}^n. \quad (5.14)$$

Denn die Richtung \Rightarrow folgt aus $(Ax, y)_2 = (x, \bar{A}^T y)_2 = (x, Ay)_2$ und die Richtung \Leftarrow folgt wenn man die Koordinatenvektoren (nicht Eigenvektoren) $x = e_i$ und $y = e_j$ jeweils einsetzt.

Im allgemeinen heißt eine Matrix A^* *adjungiert* zu A bezüglich eines Skalarproduktes $(., .)$ wenn

$$(Ax, y) = (x, A^*) \quad \forall x, y \in \mathbb{K}^n.$$

Gilt $A = A^*$ heißt A *selbstdjungiert* bezüglich $(., .)$. Somit nennt man hermitesche Matrizen auch selbstdjungiert bezüglich des Euklidschen Skalarproduktes.

5.5 Spektralnorm

Oben haben wir in Lemma 5.9 zugeordnete Matrixnormen für die Maximumnorm und die 1-Norm charakterisieren können. Die der Euklischen Norm zugeordnete Matrixnorm

$$\|A\|_2 = \sup_{x \in \mathbb{K}^n \setminus \{0\}} \frac{\|Ax\|_2}{\|x\|_2},$$

die sogenannte *Spektralnorm*, wurde bisher noch nicht untersucht. Das folgende Lemma gibt hierzu Auskunft.

Lemma 5.19. Für hermitesche Matrizen $A \in \mathbb{K}^n$ gilt

$$\|A\|_2 = \varrho(A) = \max_{1 \leq i \leq n} |\lambda_i|,$$

d.h. in diesem Fall stimmt die Spektralnorm mit dem Spektralradius überein. Für jede Matrix $A \in \mathbb{K}^n$ gilt

$$\|A\|_2 = \sqrt{\varrho(\bar{A}^T A)}.$$

Beweis. a) Im ersten Fall ist A hermitesch, hat also nach Satz 5.15 n reelle Eigenwerte λ_i und einen vollständigen Satz orthonormaler Eigenvektoren e_i . Damit lässt sich jedes $x \in \mathbb{K}^n$ in der Eigenvektorbasis darstellen als $x = \sum_{i=1}^n \alpha_i e_i$ und es gilt $(e_i, e_j) = \delta_{ij}$ ($\delta_{ij} = 1$ wenn $i = j$, sonst 0). Dann gilt

$$\begin{aligned}\|x\|_2^2 &= (x, x)_2 = \left(\sum_{i=1}^n \alpha_i e_i, \sum_{j=1}^n \alpha_j e_j \right)_2 = \sum_{i=1}^n \sum_{j=1}^n \alpha_i \bar{\alpha}_j (e_i, e_j)_2 \\ &= \sum_{i=1}^n |\alpha_i|^2, \\ \|Ax\|_2^2 &= (Ax, Ax)_2 = \left(\sum_{i=1}^n \alpha_i A e_i, \sum_{j=1}^n \alpha_j A e_j \right)_2 \\ &= \left(\sum_{i=1}^n \alpha_i \lambda_i e_i, \sum_{j=1}^n \alpha_j \lambda_j e_j \right)_2 = \sum_{i=1}^n \sum_{j=1}^n \alpha_i \lambda_i \bar{\alpha}_j \bar{\lambda}_j (e_i, e_j)_2 \\ &= \sum_{i=1}^n \lambda_i^2 |\alpha_i|^2.\end{aligned}$$

Somit

$$\|A\|_2^2 = \sup_{x \in \mathbb{K}^n \setminus \{0\}} \frac{\|Ax\|_2^2}{\|x\|_2^2} = \sup_{x \in \mathbb{K}^n \setminus \{0\}} \frac{\sum_{i=1}^n \lambda_i^2 |\alpha_i|^2}{\sum_{i=1}^n |\alpha_i|^2} \leq (\varrho(A))^2$$

also $\|A\|_2 \leq \varrho(A)$. Andererseits gilt wegen Lemma 5.13 für jede Matrixnorm $\|A\| \geq \varrho(A)$ und somit muss $\|A\|_2 = \varrho(A)$ gelten.

b) Zweiter Fall. Sei nun $A \in \mathbb{K}^n$ beliebig. Dann gilt unter Nutzung von Gleichung (5.13)

$$\|Ax\|_2^2 = (Ax, Ax)_2 = (\bar{A}^T Ax, x)_2.$$

Die Matrix $\bar{A}^T A$ ist hermitesch mit betragsgrößtem Eigenwert $\varrho(\bar{A}^T A)$. Also

$$\|A\|_2^2 = \sup_{x \in \mathbb{K}^n \setminus \{0\}} \frac{\|Ax\|_2^2}{\|x\|_2^2} = \sup_{x \in \mathbb{K}^n \setminus \{0\}} \frac{(\bar{A}^T Ax, x)_2}{\|x\|_2^2} = \varrho(\bar{A}^T A),$$

woraus die Behauptung folgt. □

5.6 Positiv Definite Matrizen

Eine wichtige Klasse von Matrizen mit vorteilhaften Eigenschaften sind die positiv definiten Matrizen. Diese sind bereits in Kapitel 4 vorgekommen und sind wie folgt definiert.

Definition 5.20. Eine Matrix $A \in \mathbb{K}^{n \times n}$ heißt *positiv definit* wenn die beiden folgenden Bedingungen erfüllt sind:

$$(Ax, x)_2 \in \mathbb{R} \quad \forall x \in \mathbb{K}^n, \tag{5.15a}$$

$$(Ax, x)_2 > 0 \quad \forall x \in \mathbb{K}^n \setminus \{0\}. \tag{5.15b}$$

Im Fall $\mathbb{K} = \mathbb{R}$ ist die erste Bedingung trivialerweise immer erfüllt und nur die zweite Bedingung ist relevant. Im Fall $\mathbb{K} = \mathbb{C}$ ist die erste Bedingung wichtig damit die zweite Bedingung überhaupt Sinn macht. \square

Damit stellt sich die Frage für welche Matrizen die Bedingung (5.15a) überhaupt Sinn macht. Antwort gibt

Lemma 5.21. Für $A \in \mathbb{C}^{n \times n}$ gilt: A ist hermitesch genau dann wenn $(Ax, x)_2 \in \mathbb{R} \forall x \in \mathbb{C}^n$.

Beweis. Richtung \Rightarrow . $A \in \mathbb{C}^{n \times n}$ sei hermitesch. Dann gilt wegen (5.14) $(Ax, x)_2 = \underline{(x, Ax)_2}$. Andererseits folgt aus der Symmetrie des Skalarproduktes $(Ax, x)_2 = \underline{(x, Ax)_2}$. Mithin gilt $(x, Ax)_2 = \underline{(x, Ax)_2}$, also $(Ax, x) \in \mathbb{R}$.

Für die Richtung \Leftarrow muss man etwas mehr aufwenden. Man rechnet

$$(A(x+y), x+y)_2 = (Ax, x)_2 + (Ax, y)_2 + (Ay, x)_2 + (Ay, y)_2.$$

Umstellen liefert

$$(Ax, y)_2 + (Ay, x)_2 = (A(x+y), x+y)_2 - (Ax, x)_2 - (Ay, y)_2 \in \mathbb{R}$$

und deswegen muss $\text{Im}(Ax, y)_2 = -\text{Im}(Ay, x)_2$ sein. Nun rechne

$$(A(ix+y), ix+y)_2 = (A(ix), ix)_2 + (A(ix), y)_2 + (Ay, ix)_2 + (Ay, y)_2$$

und stelle um zu

$$\begin{aligned} (A(ix), y)_2 + (Ay, ix)_2 &= i(Ax, y)_2 - i(Ay, x)_2 = i((Ax, y)_2 - (Ay, x)_2) \\ &= (A(ix+y), ix+y)_2 - (A(ix), ix)_2 - (Ay, y)_2 \in \mathbb{R}. \end{aligned}$$

Da $i((Ax, y)_2 - (Ay, x)_2) \in \mathbb{R}$ muss $\text{Re}((Ax, y)_2 - (Ay, x)_2) = 0$ sein also $\text{Re}(Ax, y)_2 = \text{Re}(Ay, x)_2$. Mit beiden Argumenten zusammen folgt

$$\begin{aligned} (Ax, y)_2 &= \text{Re}(Ax, y)_2 + i\text{Im}(Ax, y) \\ &= \text{Re}(Ay, x)_2 - i\text{Im}(Ay, x)_2 \\ &= \overline{(Ay, x)_2} \\ &= (x, Ay)_2. \end{aligned}$$

\square

Im Komplexen macht also die Bedingung $(Ax, x) > 0$ nur für hermitesch Matrizen Sinn. Im Reellen hingegen macht diese Bedingung für alle Matrizen Sinn und die Bedingung der Symmetrie stellt eine zusätzliche, unabhängige Bedingung dar.

Eine Charakterisierung der positiv definiten Matrizen über Eigenwerte liefert das folgende Lemma.

Lemma 5.22. Eine hermitesch Matrix A (im Reellen: symmetrische Matrix) ist genau dann positiv definit, wenn alle ihre (reellen) Eigenwerte positiv sind. Alle Hauptdiagonalelemente von A sind stets reell und positiv.

Beweis. a) A sei hermitesch mit nur positiven Eigenwerten $\lambda_i > 0$. Dann hat $x \in \mathbb{K}^n$ die Darstellung $x = \sum_{i=1}^n \alpha_i e_i$ mit orthonormalen Eigenvektoren e_i und es gilt

$$(Ax, x)_2 = \left(\sum_{i=1}^n \alpha_i A e_i, \sum_{i=1}^n \alpha_i e_i \right)_2 = \sum_{i=1}^n \lambda_i |\alpha_i|^2 > 0.$$

- b) A sei hermitesch und positiv definit. Dann gilt für jeden Eigenvektor e_i zum Eigenwert λ_i :

$$0 < (Ae_i, e_i)_2 = (\lambda_i e_i, e_i)_2 = \lambda_i (e_i, e_i)_2.$$

Da $(e_i, e_i)_2 > 0$ ist $\lambda_i > 0$.

- c) Sei z_i der i -te Koordinateneinheitsvektor, dann gilt

$$0 < (Az_i, z_i)_2 = a_{ii} \in \mathbb{R}.$$

□

Speziell für reellwertige Matrizen gilt

Lemma 5.23. Sei $A \in \mathbb{R}^{n \times n}$ symmetrisch und positiv definit. Dann liegt das betragsmäßig größte Element stets auf der Hauptdiagonale der Matrix.

Beweis. Sei z_k der k -te Koordinateneinheitsvektor. Angenommen a_{ij} , $i \neq j$ sei das betragsmäßig größte Element (also außerhalb der Hauptdiagonale). Wir rechnen

$$\begin{aligned} 0 &< (A(z_i - \text{sgn}(a_{ij})z_j), z_i - \text{sgn}(a_{ij})z_j)_2 \\ &= (Az_i, z_i)_2 - 2 \text{sgn}(a_{ij})(Az_i, z_j)_2 + \text{sgn}(a_{ij})^2 (Az_j, z_j)_2 \\ &= a_{ii} - 2 \text{sgn}(a_{ij})a_{ij} + \text{sgn}(a_{ij})^2 a_{jj} \\ &= a_{ii} - 2|a_{ij}| + a_{jj} \end{aligned}$$

also $|a_{ij}| < \frac{1}{2}(a_{ii} + a_{jj}) = \frac{1}{2}(|a_{ii}| + |a_{jj}|)$ da Hauptdiagonalelemente stets positiv sind. Dies ist ein Widerspruch, da a_{ij} als betragsmäßig größtes Element angenommen war. □

Vorlesung 6

Konditionsanalyse für lineare Gleichungssysteme

Wir wenden uns nun der Lösung linearer Gleichungssysteme zu. Zunächst untersuchen wir in dieser Vorlesung die Konditionierung dieser Aufgabe.

6.1 Notation und Lösbarkeit linearer Gleichungssystem

Gegeben seien eine Matrix $A \in \mathbb{K}^{m \times n}$ und ein Vektor $b \in \mathbb{K}^m$

$$A = \begin{bmatrix} a_{11} & \dots & a_{1n} \\ \vdots & & \vdots \\ a_{m1} & \dots & a_{mn} \end{bmatrix}, \quad b = \begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix}. \quad (6.1)$$

Elemente einer Matrix A werden mit a_{ij} oder $(A)_{ij}$ bezeichnet, entsprechend sind die Komponenten eines Vektors b dann b_i oder $(b)_i$. Die Schreibweise mit den runden Klammern bietet sich an wenn die Matrix oder der Vektor selbst schon einen Index tragen. Als Körper lassen wir $\mathbb{K} = \mathbb{R}$ oder $\mathbb{K} = \mathbb{C}$ zu.

In einem *linearen Gleichungssystem* ist ein Vektor $x \in \mathbb{K}^n$, $x = (x_1, \dots, x_n)^T$, gesucht, welcher

$$Ax = b \quad (6.2)$$

erfüllt. Das Gleichungssystem heißt

- i) *unterbestimmt*, falls $m < n$,
- ii) *quadratisch*, falls $m = n$ und
- iii) *überbestimmt*, falls $m > n$.

Das lineare Gleichungssystem hat mindestens eine Lösung falls

$$\text{Rang}(A) = \text{Rang}([A|b]) = \text{Rang} \left(\begin{bmatrix} a_{11} & \dots & a_{1n} & b_1 \\ \vdots & & & \vdots \\ a_{m1} & \dots & a_{mn} & b_m \end{bmatrix} \right)$$

d.h. b liegt im Bild der durch A gegebenen linearen Abbildung.

Für *quadratische* Matrizen (und mit diesen werden wir uns zunächst beschäftigen) sind folgende Aussagen äquivalent:

- i) $Ax = b$ ist für jedes b eindeutig lösbar.
- ii) $\text{Rang}(A) = n$.
- iii) $\det(A) \neq 0$.
- iv) A hat keinen Eigenwert 0.

6.2 Kondition einer Matrix

In der Konditionsanalyse untersuchen wir die Empfindlichkeit von Abbildungen F bezüglich der Eingaben. Im Fall eines linearen Gleichungssystems $Ax = b$ sind A, b die Eingaben und x ist die Ausgabe. Es gilt also

$$x = F(A, b) = A^{-1}b.$$

Betrachten wir zunächst nur b als Eingabe, d.h. A sei nun fest gewählt. Wir untersuchen die Auswirkungen einer Änderung von b zu $b + \Delta b$. Dann ändert sich entsprechend x zu $x + \Delta x$ und es gilt

$$A(x + \Delta x) = b + \Delta b \quad \Leftrightarrow \quad x + \Delta x = A^{-1}(b + \Delta b).$$

Die absolute Änderung im Ergebnis ist somit

$$x + \Delta x - x = \Delta x = A^{-1}(b + \Delta b) - A^{-1}b = A^{-1}\Delta b.$$

Zur Untersuchung der *relativen* Kondition gehen wir zu Normen über:

$$\begin{aligned} \frac{\|\Delta x\|}{\|x\|} &= \frac{\|A^{-1}\Delta b\|}{\|A^{-1}b\|} \\ &\leq \frac{\|A^{-1}\|\|\Delta b\|}{\|A^{-1}b\|} = \frac{\|A^{-1}\|\|b\|}{\|A^{-1}b\|} \frac{\|\Delta b\|}{\|b\|} = \frac{\|A^{-1}\|\|A A^{-1}b\|}{\|A^{-1}b\|} \frac{\|\Delta b\|}{\|b\|} \\ &\leq \|A\|\|A^{-1}\| \frac{\|\Delta b\|}{\|b\|}. \end{aligned}$$

Dabei haben wir zwei mal ausgenutzt, dass die Normen auf Matrizen und Vektoren verträglich sind.

Wir können die Größe $\|A\|\|A^{-1}\|$ als Verstärkungsfaktor für die relativen Änderungen in der Eingabe b interpretieren. Die relative Kondition ist bei diesem Ansatz durch die Verwendung der Norm in einer einzigen Zahl zusammengefasst. Wir können also nicht mehr unterscheiden welche Komponente der Eingabe sich wie auf welche Komponente der Ausgabe auswirkt. Die Analyse ist in diesem Sinn ungenau und kann zu pessimistischen Ergebnissen führen („worst-case Analyse“).

Definition 6.1 (Konditionszahl). Für jede invertierbare Matrix A heißt die Zahl

$$\text{cond}(A) = \|A\|\|A^{-1}\|$$

Konditionszahl von A . Dabei kann $\|\cdot\|$ irgendeine verträgliche Norm auf dem Vektorraum der Matrizen sein. \square

Bemerkung 6.2 (Spektralkondition). Für $\|\cdot\|_2$ heißt

$$\text{cond}_2(A) = \|A\|_2\|A^{-1}\|_2$$

Spektralkondition von A . Ist A symmetrisch und positiv definit dann gilt

$$\text{cond}_2(A) = \frac{\lambda_{\max}(A)}{\lambda_{\min}(A)}.$$

Denn dann gilt nach Lemma 5.22 für die Eigenwerte $0 < \lambda_{\min}(A) \leq \lambda \leq \lambda_{\max}(A)$ und damit wegen Lemma 5.19 $\|A\|_2 = \lambda_{\max}$. Ist λ Eigenwerte von A so ist λ^{-1} ein Eigenwerte der Inversen A^{-1} . Daher ist der kleinste Eigenwert der Inversen λ_{\max}^{-1} , der größte Eigenwert λ_{\min}^{-1} und $\|A^{-1}\|_2 = \lambda_{\min}^{-1}$. \square

Dass die Konditionszahl nicht unbedingt etwas mit der Determinante zu tun hat zeigt folgendes

Beispiel 6.3. a) Betrachte die 2×2 Matrix für $0 < \epsilon < 1$

$$A = \begin{bmatrix} -1 & 1 \\ 1-\epsilon & -1 \end{bmatrix}, \quad A^{-1} = \frac{1}{\epsilon} \begin{bmatrix} -1 & -1 \\ \epsilon-1 & -1 \end{bmatrix}.$$

Es gilt $\det(A) = \epsilon$ und die Matrix wird für $\epsilon \rightarrow 0$ singulär. Für die Konditionszahl mit der Zeilensummennorm gilt

$$\|A\|_\infty = \max(2, 2-\epsilon) = 2, \quad \|A^{-1}\|_\infty = \frac{2}{\epsilon}, \quad \text{cond}_\infty(A) = \frac{4}{\epsilon}.$$

Man könnte also vermuten dass eine kleine Determinante zu einer großen Kondition führt.

b) Dies ist jedoch nicht immer der Fall. Betrachte

$$B = \begin{bmatrix} 10^{-10} & 0 \\ 0 & 10^{-10} \end{bmatrix}, \quad B^{-1} = \begin{bmatrix} 10^{10} & 0 \\ 0 & 10^{10} \end{bmatrix}.$$

Es ist $\det(B) = 10^{-20}$ aber für die Konditionszahl gilt

$$\text{cond}_\infty(B) = \|B\|_\infty \|B^{-1}\|_\infty = 10^{-10} \cdot 10^{10} = 1.$$

c) Schließlich rechnet man nach:

$$\text{cond}(\epsilon A) = \text{cond}(A), \quad \text{aber} \quad \det(\epsilon A) = \epsilon^n \det(A).$$

Beide Größen skalieren also völlig unterschiedlich und haben in der Regel nichts miteinander zu tun.

6.3 Störungssatz

Wir wollen die Analyse von oben nun zusätzlich auf Änderungen in der Matrix A erweitern, d.h. es wird nun auch A zu $A + \Delta A$ geändert. Dann ändert sich entsprechend x zu $x + \Delta x$ und es gilt

$$(A + \Delta A)(x + \Delta x) = b + \Delta b \quad \Leftrightarrow \quad x + \Delta x = (A + \Delta A)^{-1}(b + \Delta b).$$

Auch wenn A als invertierbar angenommen wird, ist $A + \Delta A$ nicht notwendigerweise invertierbar. Wir untersuchen zunächst wann dies der Fall ist.

Lemma 6.4. $B \in \mathbb{K}^{n \times n}$ habe Norm $\|B\| < 1$. Dann ist $I + B$ regulär und es gilt

$$\|(I + B)^{-1}\| \leq \frac{1}{1 - \|B\|}. \quad (6.3)$$

Beweis. a) Für alle $x \in \mathbb{K}^n$ gilt $\|x\| = \|x + Bx - Bx\| \leq \|(I + B)x\| + \|Bx\|$ und somit

$$\|(I + B)x\| \geq \|x\| - \|Bx\| \geq \|x\| - \|B\|\|x\| = (1 - \|B\|)\|x\|.$$

Wegen $\|B\| < 1$ gilt $1 - \|B\| > 0$ und für jedes $x \neq 0$ muss $(I + B)x \neq 0$ gelten, somit ist $I + B$ regulär.

- b) Wie oben zeigt man für beliebige A, B , $\|A\| \leq \|A+B\| + \|B\| \Rightarrow \|A+B\| \geq \|A\| - \|B\|$ (inverse Dreiecksungleichung). Somit

$$\begin{aligned} 1 &= \|I\| = \|(I+B)(I+B)^{-1}\| = \|(I+B)^{-1} + B(I+B)^{-1}\| \\ &\geq \|(I+B)^{-1}\| - \|B(I+B)^{-1}\| \geq \|(I+B)^{-1}\| - \|B\|\|(I+B)^{-1}\| \\ &= (1 - \|B\|)\|(I+B)^{-1}\|. \end{aligned}$$

woraus die Abschätzung folgt. \square

Damit können wir den folgenden Satz formulieren

Satz 6.5 (Störungssatz). Sei $A \in \mathbb{K}^{n \times n}$ regulär und $\|\Delta A\| < 1/\|A^{-1}\|$. Dann ist $A + \Delta A$ ebenfalls regulär und es gilt für die Lösung des gestörten Systems $(A + \Delta A)(x + \Delta x) = b + \Delta b$:

$$\frac{\|\Delta x\|}{\|x\|} \leq \frac{\text{cond}(A)}{1 - \text{cond}(A) \frac{\|\Delta A\|}{\|A\|}} \left(\frac{\|\Delta b\|}{\|b\|} + \frac{\|\Delta A\|}{\|A\|} \right). \quad (6.4)$$

Beweis. a) Zunächst zeigen wir dass $A + \Delta A$ regulär ist. $A + \Delta A = A(I + A^{-1}\Delta A)$ ist regulär wenn A und $I + A^{-1}\Delta A$ regulär sind. A ist regulär nach Voraussetzung und wegen $\|A^{-1}\Delta A\| \leq \|A^{-1}\|\|\Delta A\| < 1$ ist nach Lemma 6.4 auch $I + A^{-1}\Delta A$ regulär.

- b) Wir rechnen zunächst nach

$$\begin{aligned} (A + \Delta A)(x + \Delta x) &= b + \Delta b \\ \Leftrightarrow Ax + (\Delta A)x + (A + \Delta A)\Delta x &= b + \Delta b \\ \Leftrightarrow (\Delta A)x + (A + \Delta A)\Delta x &= \Delta b \\ \Leftrightarrow \Delta x &= (A + \Delta A)^{-1}(\Delta b - (\Delta A)x). \end{aligned}$$

Nun gehen wir zu Normen über

$$\begin{aligned} \|\Delta x\| &= \|(A + \Delta A)^{-1}(\Delta b - (\Delta A)x)\| \\ &\leq \|(A + \Delta A)^{-1}\| (\|\Delta b\| + \|\Delta A\|\|x\|) && \text{Dreieck, verträglich} \\ &= \|[A(I + A^{-1}\Delta A)]^{-1}\| (\|\Delta b\| + \|\Delta A\|\|x\|) \\ &= \|(I + A^{-1}\Delta A)^{-1}A^{-1}\| (\|\Delta b\| + \|\Delta A\|\|x\|) \\ &\leq \|(I + A^{-1}\Delta A)^{-1}\| \|A^{-1}\| (\|\Delta b\| + \|\Delta A\|\|x\|) && \text{submultiplikativ} \\ &\leq \frac{\|A^{-1}\|}{1 - \|A^{-1}\|\|\Delta A\|} (\|\Delta b\| + \|\Delta A\|\|x\|) && \text{Lemma 6.4} \\ &= \frac{\|A^{-1}\| \|A\| \|x\|}{1 - \|A^{-1}\|\|\Delta A\|} \left(\frac{\|\Delta b\|}{\|x\|} + \frac{\|\Delta A\|}{\|A\|} \right) && \text{ausklammern, ergänzen} \\ &= \frac{\|A^{-1}\| \|A\| \|x\|}{1 - \|A^{-1}\|\|\Delta A\|} \left(\frac{\|\Delta b\|}{\|b\|} + \frac{\|\Delta A\|}{\|A\|} \right) \\ &\leq \frac{\|A^{-1}\| \|A\| \|x\|}{1 - \|A^{-1}\|\|\Delta A\|} \left(\frac{\|\Delta b\|}{\|b\|} + \frac{\|\Delta A\|}{\|A\|} \right) && \text{verträglich, submultiplikativ} \\ &= \|x\| \frac{\text{cond}(A)}{1 - \text{cond}(A) \frac{\|\Delta A\|}{\|A\|}} \left(\frac{\|\Delta b\|}{\|b\|} + \frac{\|\Delta A\|}{\|A\|} \right) \end{aligned}$$

was die Aussage beweist. \square

Beispiel 6.6. Es sei die relativen Änderung in den Eingaben von der Größenordnung 10^{-k} , d.h. $\frac{\|\Delta A\|}{\|A\|} = 10^{-k}$ und $\frac{\|\Delta b\|}{\|b\|} = 10^{-k}$. Es sei weiter $\text{cond}(A) = 10^s$ aber $10^s \cdot 10^{-k} \ll 1$. Dann gilt

$$\frac{\|\Delta x\|}{\|x\|} \leq \frac{10^s}{1 - 10^s 10^{-k}} \cdot 2 \cdot 10^{-k} \approx 10^{s-k}.$$

Man verliert somit s Stellen an Genauigkeit (hier wird die Basis $\beta = 10$ betrachtet). Ein Zahlenbeispiel gefällig. Man ändert Matrixeinträge und rechte Seite relativ um 10^{-10} , die Kondition der Matrix sei 10^5 , dann kann man relative Änderungen im Ergebnis von 10^{-5} erwarten. \square

Allerdings zeigt dass folgende Beispiel, dass eine große Kondition nicht zwangsläufig zu großen Fehlern führen muss.

Beispiel 6.7. Betrachte die Diagonalmatrix

$$D = \begin{bmatrix} d_1 & 0 \\ 0 & d_2 \end{bmatrix}, \quad D^{-1} = \begin{bmatrix} d_1^{-1} & 0 \\ 0 & d_2^{-1} \end{bmatrix}, \quad d_1, d_2 \neq 0.$$

Es gilt

$$\text{cond}_\infty(D) = \frac{\max(|d_1|, |d_2|)}{\min(|d_1|, |d_2|)},$$

also etwa für $d_1 = 10^{10}$ und $d_2 = 1$ gilt $\text{cond}_\infty(D) = 10^{10}$. Andererseits sind alle Gleichungen in dem System $Dx = b$ unabhängig voneinander und die Berechnung $x_i = b_i/d_i$ ist gut konditioniert (Multiplikation und Division sind gut konditioniert). Das Problem hier ist, dass die Analyse mittels der Konditionszahl annimmt, dass Abhängigkeiten zwischen allen Komponenten der Eingabe und der Ausgabe bestehen. Dies ist hier nicht der Fall. \square

Vorlesung 7

Eliminationsverfahren

Zunächst rekapitulieren wir das Gaußsche Eliminationsverfahren.

7.1 Dreieckssysteme

Die Lösung eines quadratischen linearen Gleichungssystems $Ax = b$ gelingt besonders einfach wenn dieses Dreiecksgestalt hat:

$$\begin{aligned} a_{11}x_1 + a_{22}x_2 + \dots + a_{nn}x_n &= b_1, \\ a_{22}x_2 + \dots + a_{nn}x_n &= b_2, \\ &\vdots \\ a_{nn}x_n &= b_n. \end{aligned} \tag{7.1}$$

Dieses System ist regulär genau dann wenn für alle Koeffizienten $a_{ii} \neq 0$ gilt. Die Lösung gelingt dann mittels *rückwärts einsetzen*:

$$x_n = b_n/a_{nn}, \tag{7.2}$$

$$x_i = \left(b_i - \sum_{j=i+1}^n a_{ij}x_j \right) / a_{ii} \quad i = n-1, \dots, 1. \tag{7.3}$$

Der Rechenaufwand beträgt

$$N_{\text{Dreieck}}(n) = \sum_{i=0}^{n-1} (2i+1) = n^2$$

elementare Rechenoperationen (Addition, Multiplikation und Division werden hier nicht unterschieden obwohl sie möglicherweise unterschiedliche Rechenzeit benötigen).

Analog kann man auch *untere Dreieckssysteme* betrachten welche sich durch vorwärts einsetzen einfach lösen lassen.

7.2 Transformation auf Dreiecksgestalt

Hat ein Gleichungssystem keine Dreiecksform so lässt es sich mittels *elementarer Umformungen* in ein solches *transformieren*. Unter elementaren Umformungen verstehen wir:

- i) Vertauschen zweier Gleichungen.
- ii) Addition des Vielfachen einer Gleichung zu einer anderen Gleichung.

Das Gaußsche Eliminationsverfahren nutzt eine endliche Folge solcher elementarer Umformungen um das System auf Dreiecksgestalt zu bringen.

Algorithmus 7.1 (Gaußelimantion, 1. Formulierung). Gegeben sei ein reguläres lineares Gleichungssystem $Ax = b$ mit einer $n \times n$ Matrix A . Das folgende Verfahren transformiert das System auf obere Dreiecksgestalt.

- 1) Das Verfahren konstruiert eine endliche Folge von Matrizen $A^{(k)} = \left(a_{i,j}^{(k)} \right)_{i,j=1}^n$ und Vektoren $b^{(k)} = \left(b_i^{(k)} \right)_{i=1}^n$. Zu Beginn setzen wir $A^{(0)} = A$, $b^{(0)} = b$ und $k = 1$.
- 2) Gilt $k = n$ so ist das Verfahren beendet.
- 3) Sonst ist nun $k < n$, also $n - k \geq 1$. Gehe wie folgt vor:

- a) Setze $\tilde{A}^{(k)} = A^{(k-1)}$ und $\tilde{b}^{(k)} = b^{(k-1)}$. Falls $\tilde{a}_{k,k}^{(k)} = 0$ finde einen Index $s \in \{k+1, \dots, n\}$ so dass $\tilde{a}_{s,k}^{(k)} \neq 0$ und vertausche die Zeilen k und s in $\tilde{A}^{(k)}$ und die Einträge k und s in $\tilde{b}^{(k)}$. Am Ende dieses Schrittes gilt

$$\tilde{a}_{k,k}^{(k)} \neq 0.$$

Das Element $\tilde{a}_{k,k}^{(k)}$ heißt *Pivotelement*, die k -te Zeile heißt *Pivotzeile*.

- b) Setze $A^{(k)} = \tilde{A}^{(k)}$, $b^{(k)} = \tilde{b}^{(k)}$ und definiere die Vektoren

$$l_i^{(k)} = \begin{cases} 0 & 1 \leq i \leq k \\ \tilde{a}_{i,k}^{(k)} / \tilde{a}_{k,k}^{(k)} & k+1 \leq i \leq n \end{cases}, \quad u_j^{(k)} = \begin{cases} 0 & 1 \leq j < k \\ \tilde{a}_{k,j}^{(k)} & k \leq j \leq n \end{cases} \quad (7.4)$$

Für $k+1 \leq i \leq n$ überschreibe die Elemente von $A^{(k)}$ und $b^{(k)}$ wie folgt.

$$a_{i,j}^{(k)} = \tilde{a}_{i,j}^{(k)} - l_i^{(k)} u_j^{(k)}, \quad j \in \{k, \dots, n\}, \quad (7.5)$$

$$b_i^{(k)} = \tilde{b}_i^{(k)} - l_i^{(k)} \tilde{b}_k^{(k)}, \quad (7.6)$$

d.h. man subtrahiert das $l_i^{(k)}$ -fache der Zeile k von der Zeile $i > k$. Am Ende dieses Schrittes hat die Matrix die Gestalt

$$A^{(k)} = \begin{bmatrix} a_{1,1}^{(k)} & \dots & a_{1,k}^{(k)} & a_{1,k+1}^{(k)} & \dots & a_{1,n}^{(k)} \\ 0 & \ddots & \vdots & & & \vdots \\ \vdots & & a_{k,k}^{(k)} & a_{k,k+1}^{(k)} & \dots & a_{k,n}^{(k)} \\ \vdots & & 0 & a_{k+1,k+1}^{(k)} & \dots & a_{k+1,n}^{(k)} \\ \vdots & & \vdots & \vdots & & \vdots \\ 0 & \dots & 0 & a_{n,k+1}^{(k)} & \dots & a_{n,n}^{(k)} \end{bmatrix}$$

- c) Setze $k = k + 1$ und gehe zu Schritt 2). □

Satz 7.2 (Gaußsches Eliminationsverfahren). Das oben beschriebene Verfahren transformiert jedes quadratische, reguläre lineare Gleichungssystem der Dimension n in $n - 1$ Schritten auf obere Dreiecksgestalt.

Beweis. Würde sich im Schritt 3a) kein Pivotelement finden lassen, dann wären die Zeilen k bis n linear abhängig. Dies ist ein Widerspruch zur Voraussetzung. In Gleichung (7.5) gilt $a_{i,k}^{(k)} = \tilde{a}_{i,k}^{(k)} - l_i^{(k)} u_k^{(k)} = \tilde{a}_{i,k}^{(k)} - (\tilde{a}_{i,k}^{(k)} / \tilde{a}_{k,k}^{(k)}) \tilde{a}_{k,k}^{(k)} = 0$. Somit wird die k -te Spalte für die Zeilen $i \geq k + 1$ zu Null eliminiert, was die Nullstruktur von $A^{(k)}$ erklärt. Wenn $k = n - 1$ erreicht ist, gilt offensichtlich $k + 1 = n$ und die Matrix $A^{(n-1)}$ hat obere Dreiecksgestalt. □

Bemerkung 7.3. 1) Der Schritt 3a) oben heißt *Pivotisierung*. In exakter Arithmetik ist es nur notwendig ein Nichtnullelement in der Spalte k zu finden. In Fließkommaarithmetik wird dieser Schritt eine wichtige Rolle spielen.

2) In einer praktischen Implementierung wird man für die Matrizen $A^{(k)}$ nicht jeweils extra Speicher reservieren sondern die Matrixelemente sukzessive überschreiben. Die Formulierung hier dient nur dazu präzise formulieren zu können welche Einträge die Matrix in jedem Schritt hat.

Beispiel 7.4. Folgendes Beispiel zeigt die Gauß-Elimination eines Gleichungssystems mit $n = 4$. Wir fassen Matrix und rechte Seite in Form einer $n \times n+1$ Matrix $[A^{(k)}|b^{(k)}]$ zusammen. Das Beispiel ist so gewählt, dass keine Zeilenvertauschungen nötig sind und Lösung und Matrix Einträge sind ganze Zahlen. Das Pivotelement ist jeweils durch einen Kasten gekennzeichnet.

$$[A^{(0)}|b^{(0)}] = \left[\begin{array}{ccccc} 2 & 4 & 6 & 8 & 40 \\ 16 & 33 & 50 & 67 & 330 \\ 4 & 15 & 31 & 44 & 167 \\ 10 & 29 & 63 & 97 & 350 \end{array} \right], \quad [A^{(1)}|b^{(1)}] = \left[\begin{array}{ccccc} 2 & 4 & 6 & 8 & 40 \\ 0 & 1 & 2 & 3 & 10 \\ 0 & 7 & 19 & 28 & 87 \\ 0 & 9 & 33 & 57 & 150 \end{array} \right],$$

$$[A^{(2)}|b^{(2)}] = \left[\begin{array}{ccccc} 2 & 4 & 6 & 8 & 40 \\ 0 & 1 & 2 & 3 & 10 \\ 0 & 0 & 5 & 7 & 17 \\ 0 & 0 & 15 & 30 & 60 \end{array} \right], \quad [A^{(3)}|b^{(3)}] = \left[\begin{array}{ccccc} 2 & 4 & 6 & 8 & 40 \\ 0 & 1 & 2 & 3 & 10 \\ 0 & 0 & 5 & 7 & 17 \\ 0 & 0 & 0 & 9 & 9 \end{array} \right].$$

Schließlich liefert Rückwärtseinsetzen:

$$x_4 = 9/9 = \boxed{1}, \quad x_3 = (17 - 7 \cdot 1)/5 = \boxed{2}, \\ x_2 = (10 - 2 \cdot 2 - 3 \cdot 1)/1 = \boxed{3}, \quad x_1 = (40 - 4 \cdot 3 - 6 \cdot 2 - 8 \cdot 1)/2 = \boxed{4}.$$

□

Lemma 7.5. Das Gaußsche Eliminationsverfahren benötigt

$$N_{\text{Gauß}}(n) = \frac{2}{3}n^3 + O(n^2)$$

Rechenoperationen um eine $n \times n$ Matrix auf obere Dreiecksgestalt zu transformieren.

Beweis. Wir zählen die Anzahl der Operationen im Schritt 3b.

$$N_{\text{Gauß}}(n) = \sum_{k=1}^{n-1} \left\{ \underbrace{n-k}_{l_i^{(k)}} + \underbrace{(n-k)}_{\# \text{ Zeilen}} \underbrace{[(n-k) \cdot 2 + 2]}_{(7.5),(7.6)} \right\} \\ = 2 \sum_{k=1}^{n-1} (n-k)^2 + 3 \sum_{k=1}^{n-1} (n-k) = \frac{2}{3}n^3 + O(n^2).$$

□

Somit lässt sich ein lineares Gleichungssystem mit insgesamt $\frac{2}{3}n^3 + O(n^2)$ Operationen lösen.

7.3 Matrixformulierung der Gauß-Elimination

Das Gaußsche Eliminationsverfahren kann auf verschiedene Weisen kompakt formuliert werden wenn man zu einer Matrixform übergeht.

Definition 7.6 (Permutationsmatrizen). Es seien Zahlen $r, s, n \in \mathbb{N}$ mit $1 \leq r \leq s \leq n$ gegeben. Die $n \times n$ Matrix $P_{r,s}$ heißt *Permutationsmatrix* wenn gilt

$$(P_{r,s})_{i,j} = \begin{cases} 1 & i = j \wedge i \neq r \wedge j \neq s \\ 1 & (i = r \wedge j = s) \vee (i = s \wedge j = r) \\ 0 & \text{sonst} \end{cases}$$

□

Permutationsmatrizen haben also die Form

$$P_{rs} = \begin{bmatrix} 1 & & & & & & & \\ & \ddots & & & & & & \\ & & 1 & & & & & \\ & & & 0 & & & 1 & \\ & & & & 1 & & & \\ & & & & & \ddots & & \\ & & & & & & 1 & \\ & & & & & & & 0 \\ & & & & & & & & 1 \\ & & & & & & & & & \ddots \\ & & & & & & & & & & 1 \end{bmatrix}$$

Lemma 7.7. Für Permutationsmatrizen $P_{r,s}$ gilt

- i) Für eine $n \times n$ Matrix A enthält das Produkt $P_{r,s}A$ die Matrix A mit den Zeilen r und s vertauscht.
- ii) Für eine $n \times n$ Matrix A enthält das Produkt $AP_{r,s}$ die Matrix A mit den Spalten r und s vertauscht.
- iii) Es gilt $P_{r,s} = P_{r,s}^T$.
- iv) Es gilt $P_{r,s}^{-1} = P_{r,s}$. Damit ist $P_{r,s}$ eine orthogonale Matrix.

Beweis. Alle Eigenschaften beruhen auf der speziellen Form der Matrizen $P_{r,s}$ und der Definition der Matrixmultiplikation.

- i) Für $\tilde{A} = P_{r,s}A$ gilt $\tilde{a}_{i,j} = \sum_{k=1}^n (P_{r,s})_{i,k} a_{k,j}$. Damit gilt

$$\begin{aligned} i \neq r \wedge i \neq s &\Rightarrow (P_{r,s})_{i,i} = 1 \Rightarrow \tilde{a}_{i,j} = a_{ij}, \\ i = r &\Rightarrow (P_{r,s})_{i,s} = 1 \Rightarrow \tilde{a}_{i,j} = a_{sj}, \\ i = s &\Rightarrow (P_{r,s})_{i,r} = 1 \Rightarrow \tilde{a}_{i,j} = a_{rj}. \end{aligned}$$

- ii) $\tilde{A} = AP_{r,s}$ zeigt man analog, nur gilt $\tilde{a}_{i,j} = \sum_{k=1}^n a_{i,k} (P_{r,s})_{k,j}$.

- iii) Folgt direkt aus der Definition.

- iv) Wegen i) und der Gestalt von $P_{r,s}P_{r,s} = I$ also $P_{r,s}^{-1} = P_{r,s}$. Wegen
 iii) gilt $P_{r,s}^{-1} = P_{r,s}^T$.

□

Der Vertauschungsschritt 3a in der Gaußelimination lässt sich mittels einer Permutationsmatrix formulieren. Für den Eliminationsschritt 3b betrachten wir folgende Matrizen. Seien l und u Vektoren der Länge n , dann ist $A = lu^T$ eine $n \times n$ Matrix mit den Einträgen $a_{i,j} = l_i u_j$. Da alle Zeilen Vielfache von einander sind hat A den Rang 1. Für den Eliminationsschritt 3b erhalten wir daher mit den Definitionen von dort $A^{(k)} = \tilde{A}^{(k)} - l^{(k)} (u^{(k)})^T$. Damit erhalten wir die folgende, kompaktere Formulierung des Gaußschen Eliminationsverfahrens.

Algorithmus 7.8 (Gaußelimination, Formulierung in Matrixform).

- 1) Setze $A^{(0)} = A$, $b^{(0)} = b$ und $k = 1$.
- 2) Ist $k = n$ so ist das Verfahren beendet.
- 3) Sonst ist nun $k < n$, also $n - k \geq 1$.
 - a) Finde einen Index $s \in \{k, \dots, n\}$ so dass $a_{s,k}^{(k-1)} \neq 0$ und setze $\tilde{A}^{(k)} = P_{k,s}A^{(k-1)}$, $\tilde{b}^{(k)} = P_{k,s}b^{(k-1)}$
 - b) Definiere wie oben die Vektoren der Länge n

$$l_i^{(k)} = \begin{cases} 0 & 1 \leq i \leq k \\ \tilde{a}_{i,k}^{(k)} / \tilde{a}_{k,k}^{(k)} & k+1 \leq i \leq n \end{cases}, \quad u_j^{(k)} = \begin{cases} 0 & 1 \leq j < k \\ \tilde{a}_{k,j}^{(k)} & k \leq j \leq n \end{cases}$$

und setze $A^{(k)} = \tilde{A}^{(k)} - l^{(k)} (u^{(k)})^T$, $b^{(k)} = \tilde{b}^{(k)} - l^{(k)} \tilde{b}_k^{(k)}$. Man sagt $A^{(k)}$ entsteht aus $\tilde{A}^{(k)}$ durch ein *Rang-1-Update*. Achtung: Wie beschrieben führt dieser Schritt *mehr* Rechenoperationen aus wie der ursprüngliche Algorithmus, da $l^{(k)}$, $u^{(k)}$ die Länge n haben. In einer Implementierung würde man ausnutzen, dass nur $(n - k - 1)^2$ der Updates wirklich durchzuführen sind. wirklich auszuführen

- c) Setze $k = k + 1$ und gehe zu Schritt 2).

□

Eine weitere Umformulierung gelingt mittels der

Definition 7.9 (Frobeniusmatrizen). Für $n, k \in \mathbb{N}$, $k \leq n$, sei I_n die $n \times n$ Einheitsmatrix, $e^{(k)}$ der k -te Koordinatenvektor und g ein Vektor der Länge n mit $g_i = 0$ für $i \leq k$. Dann heißt

$$G_k(g) = I_n + g \left(e^{(k)} \right)^T \quad (7.7)$$

Frobeniusmatrix.

□

Frobeniusmatrizen bestehen also aus der Identität mit einer Rang-1-Modifikation

und haben die Form

$$G_k(g) = \begin{bmatrix} 1 & & & & & \\ & 1 & & & & \\ & & \ddots & & & \\ & & & 1 & & \\ & & & & 1 & \\ & & & & & \ddots \\ & & & & & \\ g_n & & & & & 1 \end{bmatrix}.$$

Lemma 7.10. Für Frobeniusmatrizen der Form $G_k(g) = I_n + g (e^{(k)})^T$ gelten die folgenden Eigenschaften.

i) Sei \tilde{A} eine $n \times n$ Matrix. Für das Produkt $A = G_k(g)\tilde{A}$ mit einer Frobeniusmatrix von links gilt

$$a_{ij} = \begin{cases} \tilde{a}_{i,j} & i \leq k \\ \tilde{a}_{i,j} + g_i^{(k)} \tilde{a}_{k,j} & i > k \end{cases}.$$

Es wird also gerade das $g_i^{(k)}$ -fache der Zeile k auf die Zeile i addiert.

ii) Für die Inverse gilt $G_k^{-1}(g) = I_n - g (e^{(k)})^T = G_k(-g)$.

iii) Für das Produkt der Inversen von Frobeniusmatrizen gilt mit $k < n$

$$G_1^{-1}(g^{(1)}) G_2^{-1}(g^{(2)}) \dots G_k^{-1}(g^{(k)}) = I_n - \sum_{j=1}^k g^{(j)} (e^{(j)})^T.$$

Das Produkt ist demnach eine untere Dreiecksmatrix mit 1 auf der Hauptdiagonale.

iv) Für alle $k < \alpha \leq \beta \leq n$ gilt

$$P_{\alpha,\beta} G_k(g) = G_k(P_{\alpha,\beta} g) P_{\alpha,\beta}.$$

Beweis. i) Wir rechnen

$$A = G_k(g)\tilde{A} = \left(I_n + g (e^{(k)})^T \right) \tilde{A} = \tilde{A} + g (e^{(k)})^T \tilde{A}.$$

Nun extrahiert $(e^{(k)})^T \tilde{A}$ gerade die k -te Zeile von \tilde{A} und $g (e^{(k)})^T \tilde{A}$ ist somit gerade die Rang-1-Matrix welche aus $g_i^{(k)}$ -fachen der k -ten Zeile von \tilde{A} besteht, was behauptet wurde. Da $g_i^{(k)} = 0$ für $i \leq k$, nach Definition der Frobeniusmatrizen, gilt $\tilde{a}_{i,j} = a_{i,j}$ für $i \leq k$.

ii) Wir rechnen los

$$\begin{aligned} \left(I_n - g (e^{(k)})^T \right) \left(I_n + g (e^{(k)})^T \right) &= \\ I_n - g (e^{(k)})^T + g (e^{(k)})^T - g (e^{(k)})^T g (e^{(k)})^T &= 0 \end{aligned}$$

da im letzten Term $(e^{(k)})^T g = 0$ da $g_i = 0$ für $i \leq k$.

- iii) Beweis durch Induktion. Für $k = 1$ hat $G_1^{-1}(g)$ offensichtlich die angegebene Form. Sei die Behauptung bis $k - 1$ bewiesen. Dann gilt

$$\begin{aligned} G_1^{-1}(g^{(1)})G_2^{-1}(g^{(2)}) \dots G_k^{-1}(g^{(k)}) &= \\ &= \left(I_n - \sum_{j=1}^{k-1} g^{(j)} (e^{(j)})^T \right) \left(I_n - g^{(k)} (e^{(k)})^T \right) \\ &= I_n - \sum_{j=1}^{k-1} g^{(j)} (e^{(j)})^T - g^{(k)} (e^{(k)})^T + \sum_{j=1}^{k-1} \left(g^{(j)} (e^{(j)})^T g^{(k)} (e^{(k)})^T \right) \\ &= I_n - \sum_{j=1}^k g^{(j)} (e^{(j)})^T \end{aligned}$$

da $(e^{(j)})^T g^{(k)} = 0$ für alle $j < k$.

- iv) Wir rechnen

$$\begin{aligned} P_{\alpha,\beta} G_k(g) &= P_{\alpha,\beta} \left(I_n + g (e^{(k)})^T \right) \\ &= P_{\alpha,\beta} + P_{\alpha,\beta} g (e^{(k)})^T P_{\alpha,\beta} P_{\alpha,\beta} \\ &= \left(I_n + P_{\alpha,\beta} g (e^{(k)})^T P_{\alpha,\beta} \right) P_{\alpha,\beta} \\ &= \left(I_n + P_{\alpha,\beta} g (e^{(k)})^T \right) P_{\alpha,\beta} = G_k(P_{\alpha,\beta} g) P_{\alpha,\beta} \end{aligned}$$

da $(e^{(k)})^T P_{\alpha,\beta} = (e^{(k)})^T$ wegen $k < \alpha \leq \beta$ (die k -te Zeile von $P_{\alpha,\beta}$ entspricht der k -ten Zeile der Einheitsmatrix I_n).

□

Mit den Frobeniusmatrizen können wir nun zu Schritt 3b in Algorithmus 7.8 zurückkehren. Der Vektor $l^{(k)}$ erfüllt $l_i^{(k)} = 0$ für alle $i \leq k$ und kann somit als g -Vektor in einer Frobeniusmatrix eingesetzt werden. Wegen $\tilde{a}_{k,j}^{(k)} = 0$ für $j < k$ darf man $(u^{(k)})^T$ durch $(e^{(k)})^T \tilde{A}^{(k)}$ ersetzen und es gilt

$$A^{(k)} = \tilde{A}^{(k)} - l^{(k)} (u^{(k)})^T = \tilde{A}^{(k)} - l^{(k)} (e^{(k)})^T \tilde{A}^{(k)} = G_k(-l^{(k)}) \tilde{A}^{(k)}.$$

Algorithmus 7.11 (Gaußelimination, Formulierung mit Frobeniusmatrizen). 1)

$$A^{(0)} = A, b^{(0)} = b \text{ und } k = 1.$$

2) Ist $k = n$ so ist das Verfahren beendet.

3) Sonst ist nun $k < n$, also $n - k \geq 1$.

a) Finde einen Index $s \in \{k, \dots, n\}$ so dass $a_{s,k}^{(k-1)} \neq 0$ und setze $\tilde{A}^{(k)} = P_{k,s} A^{(k-1)}, \tilde{b}^{(k)} = P_{k,s} b^{(k-1)}$

b) Definiere den Vektor $l_i^{(k)} = \begin{cases} 0 & 1 \leq i \leq k \\ \tilde{a}_{i,k}^{(k)} / \tilde{a}_{k,k}^{(k)} & k+1 \leq i \leq n \end{cases}$ und setze $A^{(k)} = G_k(-l^{(k)}) \tilde{A}^{(k)}, b^{(k)} = G_k(-l^{(k)}) \tilde{b}^{(k)}$.

c) Setze $k = k + 1$ und gehe zu Schritt 2).

□

Vorlesung 8

LU-Zerlegung

Im letzten Kapitel haben wir das Gaußsche Eliminationsverfahren mittels Permutations- und Frobeniusmatrizen in Matrixform geschrieben. Dies erlaubt eine geschickte Implementierung des Verfahrens welches üblicherweise in der numerischen Lösung von Gleichungssystemen Verwendung findet.

8.1 Herleitung der Zerlegung

Unten treten häufig Produkte von Matrizen auf. Dabei verwenden wir die folgende Notation. Für $a, b \in \mathbb{N}_0$, ist

$$\prod_{i=a}^b M_i = M_b M_{b-1} \dots M_a. \quad (8.1)$$

Man beachte die Reihenfolge der Faktoren. Der Faktor mit Index a ist immer rechts und der Faktor mit Index b ist immer links unabhängig davon ob $a \leq b$ oder $a > b$. Zum Beispiel

$$\prod_{i=1}^3 M_i = M_3 M_2 M_1, \quad \text{und} \quad \prod_{i=3}^1 M_i = M_1 M_2 M_3.$$

Mit den Vorbereitungen aus dem letzten Kapitel können wir das Hauptresultat direkt angehen.

Satz 8.1. Sei $A \in \mathbb{K}^{n \times n}$ regulär. Dann gibt es eine Zerlegung

$$PA = LU \quad (8.2)$$

in ein Produkt von $n \times n$ Matrizen mit folgenden Eigenschaften

i) $P = \prod_{k=1}^{n-1} P_{k,s_k}$ ist ein Produkt von Permutationsmatrizen wobei $s_k \geq k$ die Indizes für die Zeilentausche aus Schritt 3a des Gauß-Algorithmus sind.

ii) L ist eine untere Dreiecksmatrix mit den Einträgen

$$L = \begin{bmatrix} 1 & 0 & \dots & 0 \\ l_{2,1} & 1 & & 0 \\ \vdots & & \ddots & \vdots \\ l_{n,1} & \dots & l_{n,n-1} & 1 \end{bmatrix} = I_n + \sum_{k=1}^{n-1} \left(\prod_{j=k+1}^{n-1} P_{j,s_j} l^{(k)} \left(e^{(k)} \right)^T \right)$$

mit den Vektoren $l^{(k)}$ aus Schritt 3b des Gauß-Algorithmus.

iii) U ist eine obere Dreiecksmatrix

$$U = \begin{bmatrix} u_{1,1} & u_{1,2} & \dots & u_{1,n} \\ 0 & u_{2,2} & & u_{2,n} \\ \vdots & & \ddots & \vdots \\ 0 & \dots & 0 & u_{n,n} \end{bmatrix}, \quad u_{i,j} = a_{i,j}^{(n-1)} \quad 1 \leq i \leq j \leq n,$$

d.h. die Einträge von U stimmen mit denen der Matrix $A^{(n-1)}$ am Ende des Gauß-Algorithmus überein.

Der Name der Zerlegung (8.2) ergibt sich aus den englischen Bezeichnungen für untere (lower) und obere (upper) Dreiecksmatrizen (triangular matrices).

Beweis. Aus der dritten Formulierung der Gaußelimination in Algorithmus 7.11 entnehmen wir, dass in jedem Schritt des Verfahrens das Gleichungssystem erst mit einer Permutationsmatrix und dann mit einer Frobeniusmatrix von links multipliziert wird:

$$\begin{aligned}
 & Ax = b \\
 \Leftrightarrow & G_1 \begin{pmatrix} -l^{(1)} \end{pmatrix} P_{1,s_1} Ax = G_1 \begin{pmatrix} -l^{(1)} \end{pmatrix} P_{1,s_1} b \\
 \Leftrightarrow & G_2 \begin{pmatrix} -l^{(2)} \end{pmatrix} P_{2,s_2} G_1 \begin{pmatrix} -l^{(1)} \end{pmatrix} P_{1,s_1} Ax = G_2 \begin{pmatrix} -l^{(2)} \end{pmatrix} P_{2,s_2} G_1 \begin{pmatrix} -l^{(1)} \end{pmatrix} P_{1,s_1} b \\
 & \vdots \\
 \underbrace{G_{n-1} P_{n-1,s_{n-1}} \dots G_2 P_{2,s_2} G_1 P_{1,s_1} A}_{A^{(n-1)}=U} x &= G_{n-1} P_{n-1,s_{n-1}} \dots G_2 P_{2,s_2} G_1 P_{1,s_1} b
 \end{aligned}$$

(Um Platz zu sparen wurden die Argumente der Frobeniusmatrizen in der letzten Zeile weggelassen). Nach $n-1$ Schritten hat die Matrix links obere Dreiecksgestalt und wir nennen diese Matrix U , also:

$$G_{n-1} P_{n-1,s_{n-1}} \dots G_2 P_{2,s_2} G_1 P_{1,s_1} A = U. \quad (8.3)$$

Nach Lemma 7.10 Nr. iv) kann man eine Frobeniusmatrix G_k und eine Permutationsmatrix $P_{\alpha,\beta}$ tauschen wenn $\alpha > k$, etwa

$$P_{2,s_2} G_1 \begin{pmatrix} -l^{(1)} \end{pmatrix} = G_1 \begin{pmatrix} -P_{2,s_2} l^{(1)} \end{pmatrix} P_{2,s_2}.$$

Auf diese Weise kann man alle Permutationsmatrizen sukzessive „nach rechts wandern“ lassen bis Gleichung (8.3) die folgende Form hat:

$$G_{n-1} \begin{pmatrix} -l^{(n-1)} \end{pmatrix} \dots G_1 \left(\underbrace{\left(\prod_{k=2}^{n-1} P_{k,s_k} \right) \begin{pmatrix} -l^{(1)} \end{pmatrix}}_P \right) \underbrace{\left(\prod_{k=1}^{n-1} P_{k,s_k} \right)}_P A = U. \quad (8.4)$$

Nun nutzen wir die Eigenschaft $G_k^{-1}(g) = G_k(-g)$ aus Lemma 7.10 Nr. ii) um die Frobeniusmatrizen auf die rechte Seite zu schaffen und erhalten

$$\underbrace{PA = G_1 \left(\left(\prod_{k=2}^{n-1} P_{k,s_k} \right) l^{(1)} \right) \dots G_{n-1} \begin{pmatrix} l^{(n-1)} \end{pmatrix} U}_L \quad (8.5)$$

wobei wir auf der rechten Seite den Faktor L identifizieren. Schließlich zeigt Lemma 7.10 Nr. iii), dass L die behauptete Form hat. \square

Wie kann man diese Zerlegung nun nutzen um eine lineares Gleichungssystem $Ax = b$ zu lösen. Dazu rechne

$$Ax = b \Leftrightarrow PAx = Pb \Leftrightarrow LUx = Pb \Leftrightarrow Ly = Pb \wedge Ux = y.$$

Algorithmus 8.2. Satz 8.1 kann man folgendermaßen nutzen um ein lineares Gleichungssystem $Ax = b$ zu lösen:

- 1) Berechne die LU -Zerlegung $PA = LU$, also die Matrizen L und U .
- 2) Berechne $\tilde{b} = Pb$.
- 3) Löse das Dreieckssystem $Ly = \tilde{b}$ durch vorwärts einsetzen.
- 4) Löse das Dreieckssystem $Ux = y$ durch rückwärts einsetzen.

Bemerkung 8.3. a) Die LU -Zerlegung ist eine (geschickte) Umformulierung der Gaußelimination bei der die Transformation von A auf obere Dreiecksgestalt und die Modifikation der rechten Seite b separiert werden. Löst man ein Gleichungssystem für genau eine rechte Seite so werden exakt die gleichen Operationen wie in der Gaußelimination durchgeführt, nur in anderer Reihenfolge.

- b) Der Vorteil der LU -Zerlegung kommt dann zum tragen wenn ein Gleichungssystem mit der selben Matrix A für *mehrere* rechte Seiten gelöst werden soll. Der Aufwand zur Berechnung der LU -Zerlegung ist im wesentlichen gleich der der Gaußelimination, also $\frac{2}{3}n^3 + O(n^2)$, während der Aufwand zur Lösung der beiden Dreieckssysteme nur $2n^2$ beträgt. Somit erhöht sich der Aufwand für eine weitere rechte Seite nur um $2n^2$.
- c) Will man beispielsweise die Inverse A^{-1} einer Matrix berechnen, so kann man dies als n simultane Gleichungssysteme mit den Koordinatenvektoren als rechte Seiten sehen. Mit den Überlegungen von oben kann man somit die Inverse mit dem Aufwand $\frac{8}{3}n^3 + O(n^2)$ berechnen.

8.2 Vollpivotisierung

Aufgabe des Schrittes 3a in der Gaußelimination ist es sicher zu stellen, dass $\tilde{a}_{k,k}^{(k)}$ ungleich Null ist. Bisher haben wir dazu, falls notwendig, einen Zeilentausch mit einer Zeile $s_k > k$ durchgeführt. Ebenso könnte man allerdings auch einen Spaltentausch durchführen. Angenommen es sei $\tilde{a}_{k,r_k}^{(k)} \neq 0$ mit $r_k > k$, dann tauscht $\tilde{A}^{(k)} P_{k,r_k}$ die k -te und die r_k -te Spalte in $\tilde{A}^{(k)}$. Im ersten Schritt verwendet man das so:

$$Ax = b \quad \Leftrightarrow \quad AP_{1,r_1} P_{1,r_1} x = b.$$

$P_{1,r_1} x$ entspricht einer Vertauschung der Komponenten x_k und x_{r_k} im Lösungsvektor.

Danach kann die Elimination mittels der Frobeniusmatrix erfolgen wie bisher. Schließlich kann sowohl ein Zeilentausch als auch ein Spaltentausch vorgenommen werden um eine beliebiges Element $\tilde{a}_{s_k,r_k}^{(k)} \neq 0$ zum Pivotelement zu machen. Dies führt zu der folgenden Variante der LU -Zerlegung.

Satz 8.4 (LU -Zerlegung mit Vollpivotisierung). Sei $A \in \mathbb{K}^{n \times n}$ regulär. Dann gibt es eine Zerlegung

$$PAQ = LU \tag{8.6}$$

in ein Produkt von $n \times n$ Matrizen mit folgenden Eigenschaften

- i) $P = \prod_{k=1}^{n-1} P_{k,s_k}$ und $Q = \prod_{k=n-1}^1 P_{k,r_k}$ sind Produkte von Permutationsmatrizen (man beachte die Reihenfolge) wobei in Schritt 3a des Gauß-Algorithmus das Element mit Index (s_k, r_k) zum Pivotelement erklärt wird.

- ii) L ist eine untere Dreiecksmatrix mit den Einträgen

$$L = \begin{bmatrix} 1 & 0 & \dots & 0 \\ l_{2,1} & 1 & & 0 \\ \vdots & & \ddots & \vdots \\ l_{n,1} & \dots & l_{n,n-1} & 1 \end{bmatrix} = I_n + \sum_{k=1}^{n-1} \left(\prod_{j=k+1}^{n-1} P_{j,s_j} l^{(k)} (e^{(k)})^T \right)$$

mit den Vektoren $l^{(k)}$ aus Schritt 3b des Gauß-Algorithmus der um den zusätzlichen Spaltentausch modifiziert wurde.

- iii) U ist eine obere Dreiecksmatrix

$$U = \begin{bmatrix} u_{1,1} & u_{1,2} & \dots & u_{1,n} \\ 0 & u_{2,2} & & u_{2,n} \\ \vdots & & \ddots & \vdots \\ 0 & \dots & 0 & u_{n,n} \end{bmatrix}, \quad u_{i,j} = a_{i,j}^{(n-1)} \quad 1 \leq i \leq j \leq n,$$

d.h. die Einträge von U stimmen mit denen der Matrix $A^{(n-1)}$ am Ende des Gauß-Algorithmus mit dem zusätzlichen Spaltentausch überein.

Beweis. Wir wenden Schrittweise die Zeilen- und Spaltentausche sowie die Eliminationen an und erhalten:

$$\begin{aligned} Ax &= b \\ G_1 P_{1,s_1} A P_{1,r_1} P_{1,r_1} x &= G_1 P_{1,s_1} b \\ G_2 P_{2,s_2} G_1 P_{1,s_1} A P_{1,r_1} P_{2,r_2} P_{2,r_2} P_{1,r_1} x &= G_1 P_{1,s_1} b \\ &\vdots \\ \underbrace{G_{n-1} P_{n-1,s_{n-1}} \dots G_1 P_{1,s_1} A P_{1,r_1} \dots P_{n-1,r_{n-1}}}_{A^{(n-1)}=U} P_{n-1,r_{n-1}} \dots P_{1,r_1} x &= G_{n-1} P_{n-1,s_{n-1}} \dots G_1 P_{1,s_1} b. \end{aligned}$$

Wie in der Variante mit den reinen Spaltentauschen kann man wieder die Permutationsmatrizen mit den Frobeniusmatrizen tauschen und man erhält mit der eingeführten Notation

$$\left(\prod_{k=1}^{n-1} G_k \right) \left(\prod_{k=1}^{n-1} P_{k,s_k} \right) A \left(\prod_{k=n-1}^1 P_{k,r_k} \right) \left(\prod_{k=1}^{n-1} P_{k,r_k} \right) = \left(\prod_{k=1}^{n-1} G_k \right) \left(\prod_{k=1}^{n-1} P_{k,s_k} \right) b,$$

also

$$\left(\prod_{k=1}^{n-1} G_k \right) PAQ = U \quad \Leftrightarrow \quad PAQ = LU.$$

□

Die Anwendung der Zerlegung (8.6) basiert dann auf der Äquivalenz

$$\begin{aligned} Ax = b &\Leftrightarrow PAQQ^{-1}x = Pb \Leftrightarrow LUQ^{-1}x = Pb \\ &\Leftrightarrow Lz = Pb \wedge Uy = z \wedge Q^{-1}x = y \end{aligned}$$

und führt zu folgendem Algorithmus

Algorithmus 8.5. 1) Berechne die LU -Zerlegung $PAQ = LU$, also die Matrizen L , U sowie die Permutations Matrizen P und Q .

- 2) Berechne $\tilde{b} = Pb$.
- 3) Löse das Dreieckssystem $Lz = \tilde{b}$ durch vorwärts einsetzen.
- 4) Löse das Dreieckssystem $Uy = z$ durch rückwärts einsetzen.
- 5) Berechne $x = Qy$.

Es ist somit nur ein zusätzlicher Permutationsschritt am Ende nötig. \square

8.3 Praktische Implementierung

Nun einige praktische Hinweise zur Implementierung der LU -Zerlegung.

Wenden wir uns erst mal den Permutationsmatrizen zu. Diese werden nicht wie gewöhnliche Matrizen als n^2 Zahlen gespeichert und auch die Multiplikation mit Permutationsmatrizen wird nicht als gewöhnliche Matrix-Vektor-Multiplikation bzw. Matrixmultiplikation durchgeführt da dies zuviel Speicher und Rechenzeit in Anspruch nehmen würde. Stattdessen benötigt die Speicherung einer Permutationsmatrix nur die Dimension n sowie die Indizes r, s . In der LU -Zerlegung treten die Matrizen zudem immer als Sequenz auf, mit den speziellen Indizes k, s_k . Daher muss man zur Realisierung der Permutationsmatrix $P = \prod_{k=1}^{n-1} P_{k,s_k}$ nur einen Vektor von natürlichen Zahlen $p = (s_1, s_2, \dots)^T$ speichern. Folgende Funktion zeigt die Anwendung der P -Matrix auf einen Vektor aus der HDNum-Bibliothek:

```
template<class T>
void permute_forward (const Vector<std::size_t>& p, Vector<T>& b)
{
    if (b.size() != p.size())
        HDNUM_ERROR("permutation_vector_incompatible_with_rhs");

    for (std::size_t k=0; k<b.size()-1; ++k)
        if (p[k] != k) std::swap(b[k], b[p[k]]);
}
```

Die Speicherung der Matrizen L und U erfolgt üblicherweise im Speicherbereich der zu zerlegenden Matrix A , welche dadurch überschrieben wird. Die Einträge (i, j) , $j \geq i$ enthalten gehören zu U , die Einträge (i, j) , $j < i$ gehören zu L und die Hauptdiagonalelemente von L werden nicht gespeichert da diese immer 1 sind. Folgende Funktion realisiert Algorithmus 8.5 in HDNum.

```
template<class T>
void linsolve (DenseMatrix<T>& A, Vector<T>& x, Vector<T>& b)
{
    if (A.rowsize() != A.colsize() || A.rowsize() == 0)
        HDNUM_ERROR("need_square_and_nonempty_matrix");
    if (A.rowsize() != b.size())
        HDNUM_ERROR("right_hand_side_incompatible_with_matrix");

    Vector<std::size_t> p(x.size());
    Vector<std::size_t> q(x.size());
    lr_fullpivot(A, p, q);
    permute_forward(p, b);
    solveL(A, b, b);
    solveR(A, x, b);
    permute_backward(q, x);
}
```

Die Integer-Vektoren \mathbf{p} und \mathbf{q} realisieren die Matrizen P , bzw. Q die während der LU -Zerlegung von A in `1r_fullpivot` berechnet werden (in HDNum heißt die Zerlegung nach ihrem deutschen Namen LR -Zerlegung). `permute_forward` und `permute_backward` realisieren deren Anwendung auf einen Vektor. Schließlich realisieren `solveL` und `solveR` die Lösung der Dreiecksfaktoren.

8.4 Lineare Integralgleichungen

Wir wollen nun ein größeres Beispiel behandeln um die bisher behandelten Konzepte zu illustrieren. Dies tun wir anhand der numerischen Lösung von Integralgleichungen.

Wir suchen eine unbekannte Funktion $u : (\alpha, \beta) \rightarrow \mathbb{R}$ die folgende Gleichung erfüllt

$$\lambda(x)u(x) + \int_{\alpha}^{\beta} K(x, y)u(y) dy = f(x) \quad \forall x \in (\alpha, \beta). \quad (8.7)$$

Eine solche Gleichung nennt man eine Integralgleichung, da die unbekannte Funktion u unter einem Integral vorkommt (so wie man Differentialgleichung sagt, wenn Ableitungen von u in einer Gleichung für u vorkommen). Dabei sind $\lambda(x)$, $f(x)$ und $K(x, y)$ gegebene Funktionen auf dem Intervall (α, β) . Integralgleichungen haben viele Anwendung, z.B. in der Mechanik aber auch in der Computergrafik. Dort berechnet man die Lichtintensität welche aus einer bestimmten Richtung von einer Szene beim Betrachter ankommt. Beim Ray-Tracing-Verfahren werden Lichtstrahlen auf der Oberfläche von Objekten nach der Regel „Einfallsinkel gleich Ausfallswinkel“ reflektiert. Dies führt unter anderem zu sehr harten Schatten. Bei der Radiosity-Methode wird das aus einer gegebenen Richtung auf einen Punkt einer Oberfläche treffende Licht in alle Richtungen reflektiert, allerdings unterschiedlich stark. Wieviel Licht aus einer Richtung x in eine Richtung y reflektiert wird beschreibt die oben eingeführte Funktion $K(x, y)$, die sogenannte *Kernfunktion*, in der Integralgleichung.

Integralgleichungen kommen in diversen Varianten vor, einige davon sind:

- Obige Integralgleichung nennt man linear, da der Ausdruck auf der linken Seite linear in der Funktion u ist. Eine nichtlineare Integralgleichung hat z.B. die Form $\int_{\alpha}^{\beta} K(x, y, u(y)) dy = f(x)$.
- Weiter heißt eine lineare Integralgleichung von „1. Art“ wenn $\lambda(x) = 0$.
- Schließlich heißt eine Integralgleichung „Fredholmsch“ wenn, wie im obigen Fall, die Integralgrenzen fest sind. Man spricht von einer „Volteraschen“ Integralgleichung wenn der Integrationsbereich von x abhängt.

Im folgenden wollen wir die lineare Fredholmsche Integralgleichung 1. Art

$$\int_{\alpha}^{\beta} K(x, y)u(y) dy = f(x) \quad \forall x \in (\alpha, \beta). \quad (8.8)$$

betrachten. Ein typisches Beispiel wäre die Kernfunktion

$$K(x, y) = \frac{1}{|x - y|^{\gamma}}, \quad 0 < \gamma < 1.$$

Für den angegebenen Bereich des Parameters γ ist der Kern zwar singulär (dies ist typisch für Integralgleichungen) aber integrierbar. Man spricht von einer *schwach singulären* Kernfunktion.

Die Frage nach Existenz, Eindeutigkeit sowie weiterer Eigenschaften der Lösungen von Integralgleichungen sind Gegenstand der Funktionalanalysis. *An dieser Stelle ein kleiner Ausflug für Mathematiker:* Sei u eine Funktion aus einer gegebenen Menge U (z.B. ein Banachraum) von Funktionen. Dann definiert

$$I_u(x) = \int_{\alpha}^{\beta} K(x, y)u(y) dy$$

eine Funktion aus einer Menge V (z.B. ein anderer Banachraum). Zu jedem $u \in U$ gehört also eine Funktion $I_u \in V$. Diese Zuordnung $A : U \rightarrow V$, $A(u) = I_u$ ist, wie man nachrechnet, eine lineare Abbildung zwischen den Räumen U und V . In dieser abstrakten Formulierung lautet die Integralgleichung: Finde ein $u \in U$ so dass

$$A(u) = f,$$

also eine lineare Gleichung in Funktionenräumen.

Numerische Lösung mit der Kollokationsmethode

Wir betrachten nun ein einfache numerische Methode zur Lösung von Integralgleichungen, die sogenannte *Kollokationsmethode*.

Dazu unterteilen wir das endliche Intervall $(\alpha, \beta) \subset \mathbb{R}$ in n gleichgroße Teilintervalle der Länge $h = (\beta - \alpha)/n$ durch einführen der Punkte

$$x_i = \alpha + ih, \quad 0 \leq i \leq n.$$

Zusätzlich bezeichnen wir die Mittelpunkte der Teilintervalle mit

$$c_i = \frac{x_i + x_{i+1}}{2} = \alpha + \left(i + \frac{1}{2}\right)h, \quad 0 \leq i < n.$$

Auf dieser Zerlegung des Intervalls (α, β) führen wir nun die folgenden n Funktionen ein

$$\varphi_i(x) = \begin{cases} 1 & x_i < x < x_{i+1} \\ 0 & \text{sonst} \end{cases}, \quad 0 \leq i < n,$$

und machen den Ansatz

$$u_h(x) = \sum_{j=0}^{n-1} z_j \varphi_j(x).$$

Die Funktion u_h ist ein Element der Menge der *stückweise konstante Funktion* auf der Zerlegung gegeben durch die x_i und ist festgelegt durch die n Koeffizienten z_i . Die Menge der stückweise Konstanten Funktionen ist ein n -dimensionaler Vektorraum der in eineindeutiger Weise dem \mathbb{R}^n entspricht:

$$u_h = \sum_{j=0}^{n-1} z_j \varphi_j(x) \quad \Leftrightarrow \quad z_j = u_h(c_j).$$

Die Kollokationsmethode bestimmt nun die n unbekannten Koeffizienten z_i aus den n Gleichungen

$$\int_{\alpha}^{\beta} K(c_i, y) u_h(y) dy = f(c_i) \quad 0 \leq i < n. \quad (8.9)$$

Einsetzen der Darstellung von u_h in den Kollokationsansatz liefert dann

$$\begin{aligned} \int_{\alpha}^{\beta} K(c_i, y) u_h(y) dy &= \int_{\alpha}^{\beta} K(c_i, y) \left(\sum_{j=0}^{n-1} z_j \varphi_j(y) \right) dy \\ &= \sum_{j=0}^{n-1} z_j \int_{x_j}^{x_{j+1}} K(c_i, y) dy \\ &= \sum_{j=0}^{n-1} z_j \int_{c_j - h/2}^{c_j + h/2} K(c_i, y) dy = f(c_i). \end{aligned} \quad (8.10)$$

Wir erhalten demnach ein lineares Gleichungssystem

$$Az = b$$

für die unbekannten Koeffizienten $z = (z_0, \dots, z_{n-1})^T$ (hier nummeriert ab 0!) mit den Einträgen

$$a_{i,j} = \int_{c_j - h/2}^{c_j + h/2} K(c_i, y) dy, \quad b_i = f(c_i). \quad (8.11)$$

Eine weitere Methode ist die *Galerkinmethode* bei der man fordert

$$\int_{\alpha}^{\beta} \int_{\alpha}^{\beta} K(x, y) u_h(y) dy \varphi_i(x) dx = \int_{\alpha}^{\beta} f(c_i) \varphi_i(x) dx \quad 0 \leq i < n. \quad (8.12)$$

Auch in diesem Fall wird man ein lineares Gleichungssystem erhalten. Wir wollen nun ein spezielles Beispiel betrachten.

Beispiel 8.6. Wir betrachten die oben angegebene schwach singuläre Kernfunktion

$$K(x, y) = \frac{1}{|x - y|^\gamma}, \quad 0 < \gamma < 1.$$

und erhalten damit für die Koeffizienten des linearen Gleichungssystems:

$$a_{i,j} = \int_{c_j - h/2}^{c_j + h/2} \frac{1}{|c_i - y|^\gamma} dy = \begin{cases} \frac{2}{1-\gamma} \left(\frac{h}{2} \right)^{1-\gamma} & c_i = c_j \\ \frac{1}{1-\gamma} \left[(|c_i - c_j| + \frac{h}{2})^{1-\gamma} - (|c_i - c_j| - \frac{h}{2})^{1-\gamma} \right] & \text{sonst.} \end{cases} \quad (8.13)$$

Hier kann man das Integral über die Kernfunktion geschlossen angeben. Später werden wir auch numerische Integration kennenlernen. Speziell wählen wir nun das Einheitsintervall $(\alpha, \beta) = (0, 1)$ sowie die rechte Seite

$$f(x) = \frac{1}{1.1 + \sin(7\pi x)}.$$

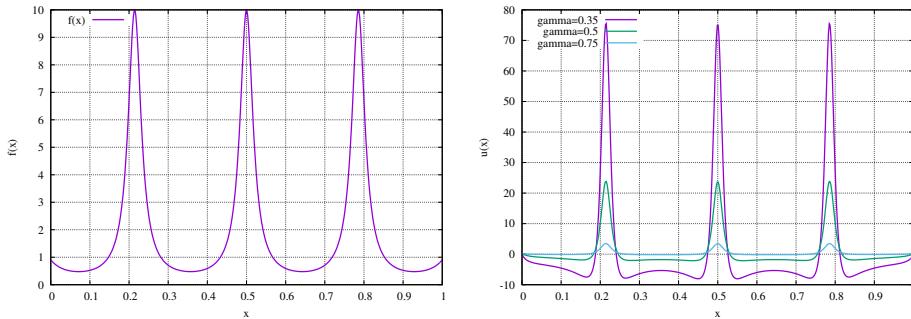


Abbildung 8.1: Rechte Seite $f(x)$ sowie numerisch berechnete Lösungen $u(x)$ zu den Parametern $\gamma \in \{0.35, 0.5, 0.75\}$.

Abbildung 8.1 zeigt diese Funktion sowie numerisch berechnete Lösungen $u(x)$ zu den Parametern $\gamma \in \{0.35, 0.5, 0.75\}$.

Die folgende Tabelle enthält Konditionszahlen $\text{cond}_\infty(A)$ für verschiedene Parameter γ und Gitterweiten $h = 1/n$. Diese Zahlen wurden mit sehr hoher Genauigkeit (mehr als hundert Nachkommstellen im Zehnersystem) berechnet.

γ/n	8	32	128	512
0.9	1.48	1.85	2.28	2.77
0.5	6.48	14.03	28.99	58.90
0.1	66.30	234.67	819.37	2855.07
1e-2	774.13	3076.99	12145.30	47915.50
1e-3	7862.13	31618.50	126349.00	504711.00
1e-4	78743.10	317047.00	1.26e+06	5.07e+06
1e-5	787553.00	3.17e+06	1.26e+07	5.07e+07

Wir beobachten, dass die Konditionszahl für $\gamma \rightarrow 1$ nur schwach mit n wächst und nahe bei 1 liegt. Für $\gamma \rightarrow 0$ werden die Konditionszahlen jedoch sehr groß und steigen linear mit n an. Schon für $n = 8$ liegt die Kondition nahe 10^6 .

Wir wollen nun die Größe der Rundungsfehler in Abhängigkeit der Konditionszahl untersuchen. Dazu bestimmen wir zu A die Inverse A^{-1} und berechnen die Fehlernorm $\|AA^{-1} - I\|_\infty$. Wegen $\|I\|_\infty = 1$ ist hier absoluter gleich relativem Fehler.

Bei Berechnung mit `double`-Genauigkeit erhalten wir die folgende Tabelle:

γ/n	8	32	128	512
0.9	4.95e-16	1.06e-15	2.58e-15	7.63e-15
0.5	6.62e-16	4.20e-15	2.61e-14	1.37e-13
0.1	8.02e-15	5.53e-14	7.69e-13	8.25e-12
1e-2	9.41e-14	1.52e-12	9.35e-12	2.00e-10
1e-3	8.81e-13	2.59e-11	1.85e-10	1.18e-09
1e-4	1.14e-11	1.68e-10	1.64e-09	3.19e-08
1e-5	8.73e-11	1.07e-09	2.28e-08	3.86e-07

Bei γ nahe 1 erhalten wir die volle Genauigkeit welche wir von `double` erwarten können. Bei $\gamma = 10^{-5}$ und $n = 512$ hingegen verlieren wir ungefähr die Hälfte der zur Verfügung stehenden Rechengenauigkeit, was ungefähr dem Ergebnis aus dem Störungssatz 6.5 und den Überlegungen aus Beispiel 6.6 entspricht.

Führen wir die Berechnungen mit `float`-Genauigkeit durch dann erhalten wir die folgende Tabelle:

γ/n	8	32	128	512
0.9	1.92e-07	5.83e-07	2.07e-06	8.53e-06
0.5	4.24e-07	2.47e-06	1.49e-05	1.73e-04
0.1	5.54e-06	5.70e-05	4.16e-04	1.78e-02
1e-2	5.72e-05	1.33e-03	8.07e-03	4.08e-01
1e-3	4.88e-04	1.10e-02	2.22e-01	2.95e+00
1e-4	3.17e-03	1.60e-01	2.09e+00	1.58e+01
1e-5	4.68e-02	2.11e+00	1.33e+01	3.24e+02

Hier sieht man ein vergleichbares Verhalten. Im gut konditionierten Fall erhält man einen Fehler entsprechend der Genauigkeit von `float`. Für die schlecht konditionierten Probleme kann mit `float` kein zufriedenstellendes Ergebnis berechnet werden. \square

Vorlesung 9

Rundungsfehleranalyse der LU-Zerlegung

Wir erinnern uns an die Analyse in Kapitel 3. Dort waren $F : \mathbb{R}^m \rightarrow \mathbb{R}^n$ und $F' : \mathbb{F}^m \rightarrow \mathbb{F}^n$ einander entsprechende Abbildungen in dem Sinne dass F' die Abbildung F durch $k \in \mathbb{N}$ arithmetische Grundoperationen in Fließkommaarithmetik realisiert. In der Rundungsfehleranalyse geht es um die Differenz $F(x) - F'(x)$ für $x \in \mathbb{F}^m$.

In der, z.B. in Beispiel 3.8 durchgeführten, sogenannten *Vorwärtsanalyse* waren wir in der Lage eine Beziehung der Form

$$F'(x) = F(x) + E(x, \epsilon_1, \dots, \epsilon_k)$$

herzuleiten mit $\epsilon_1, \dots, \epsilon_k$ den Rundungsfehlern aus den einzelnen Rechenoperationen. Für den Fehler gilt $E(x, \epsilon_1, \dots, \epsilon_k) = 0$, wenn $\epsilon_1 = \dots = \epsilon_k = 0$ weil ohne Rundungsfehler ja $F'(x) = F(x)$ gilt. In der Regel betrachten wir nur die führende Ordnung

$$E(x, \epsilon_1, \dots, \epsilon_k) \doteq A(x) \begin{bmatrix} \epsilon_1 \\ \vdots \\ \epsilon_k \end{bmatrix}$$

mit einer $n \times k$ Matrix A der Koeffizienten Ausdrücke in den Eingaben x sind.

In diesem Kapitel werden wir uns mit der sogenannten *Rückwärtsanalyse* beschäftigen. Dabei versucht man eine Beziehung der Form

$$F'(x) = F(x + H(x, \epsilon_1, \dots, \epsilon_k)), \quad H(x, \epsilon_1, \dots, \epsilon_k) \doteq B(x) \begin{bmatrix} \epsilon_1 \\ \vdots \\ \epsilon_k \end{bmatrix}$$

zu finden, d.h. die numerische Berechnung $F'(x)$ erscheint als Ergebnis der exakten Berechnung F mit modifizierten Eingaben $x + H(x, \epsilon_1, \dots, \epsilon_k)$, wobei die Modifikationen in erster Näherung linear in den gemachten Rundungsfehlern sind. Also: Wie muss man die Eingabe der exakten Rechnung modifizieren um das selbe Ergebnis wie in der Fließkommaarithmetik zu erhalten. Daher röhrt der Name „Rückwärtsanalyse“.

Dieses Kapitel folgt [Golub and Van Loan, 2013, Kapitel 3].

9.1 Notation

In diesem Kapitel betrachten wir die Rückwärtsanalyse für Probleme der linearen Algebra. Dafür ist es gut etwas Notation einzuführen um die Resultate kompakter ausdrücken zu können.

Für Vektoren $x \in \mathbb{R}^n$ und Matrizen $A \in \mathbb{R}^{m \times n}$ vereinbaren wir die Absolutwertnotation:

$$|x| = \begin{bmatrix} |x_1| \\ \vdots \\ |x_n| \end{bmatrix}, \quad |A| = \begin{bmatrix} |a_{1,1}| & \cdots & |a_{1,n}| \\ \vdots & & \vdots \\ |a_{m,1}| & \cdots & |a_{m,n}| \end{bmatrix}.$$

Beachte, dass der Absolutwert keine Norm ist sondern elementweise den Betrag nimmt!

Ebenso wird die Rundungsoperation $\text{rd} : \mathbb{R}^{m \times n} \rightarrow \mathbb{F}^{m \times n}$ komponentenweise auf Vektoren und Matrizen erweitert:

$$\text{rd}(x) = \begin{bmatrix} \text{rd}(x_1) \\ \vdots \\ \text{rd}(x_n) \end{bmatrix}, \quad \text{rd}(A) = \begin{bmatrix} \text{rd}(a_{1,1}) & \cdots & \text{rd}(a_{1,n}) \\ \vdots & & \vdots \\ \text{rd}(a_{m,1}) & \cdots & \text{rd}(a_{m,n}) \end{bmatrix}.$$

Und schließlich können wir auch Relationen wie $=$ oder \leq auf alle Komponenten erweitern:

$$A \leq B \quad \Leftrightarrow \quad a_{i,j} \leq b_{i,j}, \quad 1 \leq i \leq m, 1 \leq j \leq n.$$

Beachte dass dies ein System von nm simultanen Ungleichungen darstellt!

Damit können wir etwa kurz schreiben:

$$\text{rd}(A) = A + A' \quad \text{mit} \quad |A'| \leq |A| \text{ eps.}$$

Schließlich vereinbaren wir die fl-Notation. Diese überführt einen gegebenen Ausdruck bestehend aus arithmetischen Grundoperationen in einen entsprechenden Ausdruck mit Fließkommaoperationen. Mit den oben eingeführten Bezeichnungen gilt also

$$F'(x) = \text{fl}(F)(x).$$

Als ein Beispiel betrachte das Euklidsche Skalarprodukt zweier (reellwertiger) Vektoren $x \cdot y = x^T y = \sum_{i=1}^n x_i y_i$. Dann gilt

$$\text{fl}(x \cdot y) = x_1 \odot y_1 \oplus \dots \oplus x_n \odot y_n.$$

Im Prinzip kann diese Notation auch auf andere Funktionen erweitert werden, so wäre $\text{fl}(\sqrt{x})$ eine Berechnung von \sqrt{x} in Fließkommaarithmetik.

Kommen wir nun zurück zur Rückwärtsanalyse. Angewandt auf das Problem „Lineares Gleichungssystem Lösen“ würde die Vorwärtsanalyse eine Beziehung der Form

$$\hat{x} - x = E(A, b, \epsilon_1, \dots, \epsilon_k)$$

herleiten, wobei $\hat{x} \in \mathbb{F}^n$ die Lösung des linearen Gleichungssystems in Fließkommaarithmetik ist. Die Rückwärtsanalyse dagegen würde die Lösung darstellen als

$$(A + H(A, b, \epsilon_1, \dots, \epsilon_k))\hat{x} = b$$

Dies hat den Vorteil, dass wir nun die Auswirkung der Rundungsfehler mittels des Störungssatzes 6.5 aus der Konditionsanalyse abschätzen können:

$$\frac{\|\hat{x} - x\|}{\|x\|} \leq \frac{\text{cond}(A)}{1 - \text{cond}(A) \frac{\|H\|}{\|A\|}} \frac{\|H\|}{\|A\|}.$$

9.2 Dreieckssysteme

Wir beginnen mit einem Hilfssatz.

Lemma 9.1. Für gegebene Vektoren $x, y \in \mathbb{F}^n$ gilt für das Skalarprodukt

$$\text{fl}(x \cdot y) = (x + f) \cdot y, \quad |f| \leq n \text{ eps } |x| + O(\text{eps}^2).$$

Beweis. Wir beweisen mittels Induktion.

$n = 1$. Wegen exakt gerundeter Fließkommaarithmetik gilt $\text{fl}(x \cdot y) = \text{fl}(x_1 y_1) = x_1 y_1 (1 + \epsilon_1) = (x_1 + x_1 \epsilon_1) y_1$, also $|f_1| = |\epsilon_1| |x_1| \leq \text{eps} |x_1|$.

$n \geq 2$. Die Behauptung gelte schon für Vektoren der Länge $n - 1$. Es seien also $x, y \in \mathbb{F}^n$ und $\tilde{x}, \tilde{y} \in \mathbb{F}^{n-1}$ mit $\tilde{x}_i = x_i, \tilde{y}_i = y_i$. Dann gilt

$$\begin{aligned} \text{fl}(x \cdot y) &= \text{fl}(\tilde{x} \cdot \tilde{y} + x_n y_n) \\ &= (\text{fl}(\tilde{x} \cdot \tilde{y}) + \text{fl}(x_n y_n))(1 + \epsilon_n) \\ &= ((\tilde{x} + \tilde{f}) \cdot \tilde{y} + x_n y_n (1 + \delta_n))(1 + \epsilon_n) \\ &= (\tilde{x} + \tilde{f} + \epsilon_n \tilde{x} + \epsilon_n \tilde{f}) \cdot \tilde{y} + (x_n + (\epsilon_n + \delta_n) x_n + \epsilon_n \delta_n x_n) y_n \\ &= (x + f) \cdot y \end{aligned}$$

mit

$$f = \begin{bmatrix} \tilde{f} + \epsilon_n \tilde{x} + \epsilon_n \tilde{f} \\ (\epsilon_n + \delta_n) x_n + \epsilon_n \delta_n x_n \end{bmatrix}.$$

Für die Abschätzung gilt

$$|\tilde{f} + \epsilon_n \tilde{x} + \epsilon_n \tilde{f}| \leq (n - 1) \text{eps} |\tilde{x}| + \text{eps} |\tilde{x}| + O(\text{eps}^2) = n \text{eps} |\tilde{x}| + O(\text{eps}^2)$$

und

$$|(\epsilon_n + \delta_n) x_n + \epsilon_n \delta_n x_n| \leq 2 \text{eps} |x_n| + O(\text{eps}^2).$$

Wegen $n \geq 2$ gilt somit $|f| \leq n \text{eps} |x| + O(\text{eps}^2)$ wie behauptet. \square

Dieses Lemma zeigt, dass sich die Rundungsfehler im Skalarprodukt im schlimmsten Fall alle addieren. Dies ist allerdings sehr pessimistisch.

Satz 9.2. Es seien $\hat{x} \in \mathbb{F}^n$ bzw. $\hat{y} \in \mathbb{F}^n$ die numerischen Lösungen der unteren bzw. oberen Dreieckssysteme $Lx = b$ bzw. $Uy = c$ in Fließkommaarithmetik. Dann gibt es Matrizen F bzw. G so dass gilt

$$\begin{aligned} (L + F)\hat{x} &= b, & |F| &\leq n \text{eps} |L| + O(\text{eps}^2), \\ (U + G)\hat{y} &= c, & |G| &\leq n \text{eps} |U| + O(\text{eps}^2). \end{aligned}$$

Beweis. Wir beweisen mittels Induktion über die Größe der Matrizen n . Dabei betrachten wir nur das vorwärts einsetzen, also L . Der Beweis für das System mit U geht analog.

$n = 1$. In diesem Fall gilt $l_{1,1} x_1 = b_1$, also $\hat{x}_1 = \text{fl}(b_1 / l_{1,1}) = (b_1 / l_{1,1})(1 + \epsilon_1)$. Dies können wir nun mittels Taylorentwicklung schreiben als

$$\begin{aligned} \frac{l_{1,1}}{1 + \epsilon_1} \hat{x}_1 &= b_1 \\ \Leftrightarrow (1 - \epsilon_1 + R(\epsilon_1^2)) l_{1,1} \hat{x}_1 &= b_1 \\ \Leftrightarrow (l_{1,1} - \epsilon_1 l_{1,1} + l_{1,1} R(\epsilon_1^2)) \hat{x}_1 &= b_1. \end{aligned}$$

Somit gilt die Darstellung mit $f_1 = -\epsilon_1 l_{1,1} + l_{1,1} R(\epsilon_1^2)$ und damit

$$|f_1| \leq \text{eps} |l_{1,1}| + O(\text{eps}^2).$$

$n \geq 2$. Wir schreiben das untere Dreieckssystem in Blockform:

$$\begin{bmatrix} L_1 & 0 \\ v^T & \alpha \end{bmatrix} \begin{bmatrix} x_1 \\ w \end{bmatrix} = \begin{bmatrix} b_1 \\ \beta \end{bmatrix}$$

mit der $n-1 \times n-1$ -Matrix L_1 sowie den $n-1$ -Vektoren v , x_1 und b_1 . Sei nun \hat{x}_1 die numerische Lösung von $L_1 x_1 = b_1$ für die die Induktionsvoraussetzung $(L_1 + F_1)\hat{x}_1 = b_1$ gilt. Dann berechnen wir \hat{w} als

$$\begin{aligned}\hat{w} &= \text{fl}((\beta - v^T \hat{x}_1)/\alpha) \\ &= (\text{fl}(\beta - v^T \hat{x}_1)/\alpha)(1 + \epsilon_n) \\ &= ((\beta - \text{fl}(v^T \hat{x}_1))(1 + \delta_n)/\alpha)(1 + \epsilon_n) \\ &= ((\beta - (v + f)^T \hat{x}_1)/\alpha)(1 + \delta_n + \epsilon_n + \delta_n \epsilon_n).\end{aligned}$$

wobei wir im letzten Schritt den Hilfssatz 9.1 benutzt haben. Wie oben benutzen wir die Taylorentwicklung

$$\frac{1}{1 + \delta_n + \epsilon_n + \delta_n \epsilon_n} = 1 - (\delta_n + \epsilon_n) + R((\delta_n + \epsilon_n + \delta_n \epsilon_n)^2)$$

und erhalten

$$\begin{aligned}(1 - (\delta_n + \epsilon_n) + R((\delta_n + \epsilon_n + \delta_n \epsilon_n)^2)) \hat{w} &= (\beta - (v + f)^T \hat{x}_1)/\alpha \\ \Leftrightarrow (v + f)^T \hat{x}_1 + (1 - (\delta_n + \epsilon_n) + R((\delta_n + \epsilon_n + \delta_n \epsilon_n)^2)) \alpha \hat{w} &= \beta.\end{aligned}$$

Mit Hilfe der Induktionsvoraussetzung erhalten wir dann

$$\begin{bmatrix} L_1 + F_1 & 0 \\ (v + f)^T & (1 - (\delta_n + \epsilon_n) + R((\delta_n + \epsilon_n + \delta_n \epsilon_n)^2)) \alpha \end{bmatrix} \begin{bmatrix} \hat{x}_1 \\ \hat{w} \end{bmatrix} = \begin{bmatrix} \hat{b}_1 \\ \beta \end{bmatrix}$$

und daraus identifizieren wir

$$F = \begin{bmatrix} F_1 & 0 \\ f^T & -(\delta_n + \epsilon_n) + R((\delta_n + \epsilon_n + \delta_n \epsilon_n)^2) \alpha \end{bmatrix}.$$

Für die Abschätzung erhalten wir nach Induktionsvoraussetzung

$$|F_1| \leq (n-1) \text{ eps } |L_1| + O(\text{eps}^2)$$

und für die beiden anderen Teile bekommen wir

$$\begin{aligned}|f^T| &\leq (n-1) \text{ eps } |v|^T + O(\text{eps}^2) \\ | -(\delta_n + \epsilon_n) + R((\delta_n + \epsilon_n + \delta_n \epsilon_n)^2) \alpha | &\leq 2 \text{ eps } |\alpha| + O(\text{eps}^2)\end{aligned}$$

und somit insgesamt wegen $n \geq 2$ dann $|F| \leq n \text{ eps } |L| + O(\text{eps}^2)$ wie behauptet. \square

9.3 LU-Zerlegung und Lösen eines LGS

Nach diesen Vorarbeiten können wir uns nun dem Hauptresultat zuwenden.

Satz 9.3. Es sei $A \in \mathbb{F}^{n \times n}$ regulär. Wir nehmen an die exakte LU-Zerlegung $A = LU$ sei ohne Zeilentausch möglich. Dann gilt für die numerisch berechneten Faktoren \hat{L} und \hat{U} die Darstellung

$$A + H = \hat{L} \hat{U} \quad \text{mit} \quad |H| \leq 3(n-1) \text{ eps } (|A| + |\hat{L}| |\hat{U}|) + O(\text{eps}^2)$$

Beweis. Wieder beweisen wir mittels Induktion über n .

$n = 1$. In diesem Fall gilt $l_{1,1} = 1$ und damit auch in Fließkommaarithmetik $1 \odot u_{1,1} = a_{1,1}$, also $H = 0$.

$n \geq 2$. Wir schreiben A in 2×2 Blockform:

$$A = \begin{bmatrix} \alpha & w^T \\ v & B \end{bmatrix}.$$

Dann geht die LU -Zerlegung folgendermaßen vor:

- Berechne den $n - 1$ -Vektor $\hat{z} = \text{fl}(v/\alpha)$.
- Berechne die Modifikation als Rang-1-Update: $\hat{A}_1 = \text{fl}(B - \hat{z}w^T)$
- Berechne *rekursiv* die LU -Zerlegung von \hat{A}_1 , hier können wir die Induktionsvoraussetzung nutzen.

Diese Schritte arbeiten wir nun ab

$$a) \hat{z} = \text{fl}(v/\alpha) = v/\alpha + f \text{ mit } |f| \leq \text{eps } |v|/|\alpha|.$$

b)

$$\begin{aligned} \hat{A}_1 &= \text{fl}(B - \hat{z}w^T) \\ &= B - \text{fl}(\hat{z}w^T) + G \quad |G| \leq \text{eps } |B - \text{fl}(\hat{z}w^T)| \\ &= B - (\hat{z}w^T + G') + G \quad |G'| \leq \text{eps } |\hat{z}w^T| \leq \text{eps } |\hat{z}||w^T| \\ &= B - \hat{z}w^T + F \quad |F| = |-G' + G| \leq |G'| + |G| \\ &\quad \leq \text{eps } |\hat{z}||w^T| + \text{eps } |B - \text{fl}(\hat{z}w^T)| \\ &\leq 2 \text{ eps } (|B| + |\hat{z}||w^T|) + O(\text{eps}^2). \end{aligned}$$

c) Nun wird \hat{A}_1 LU -zerlegt und es gilt die Induktionsannahme

$$\hat{A}_1 + H_1 = \hat{L}_1 \hat{U}_1 \quad \text{mit} \quad |H_1| \leq 3(n-2) \text{ eps } (|\hat{A}_1| + |\hat{L}_1||\hat{U}_1|) + O(\text{eps}^2).$$

Die LU -Zerlegung, welche der numerische Algorithmus zurückliefert, lautet dann in Blockform

$$\begin{aligned} \hat{L} \hat{U} &= \begin{bmatrix} 1 & 0 \\ \hat{z} & \hat{L}_1 \end{bmatrix} \begin{bmatrix} \alpha & w^T \\ 0 & \hat{U}_1 \end{bmatrix} = \begin{bmatrix} \alpha & w^T \\ \alpha \hat{z} & \hat{z}w^T + \hat{L}_1 \hat{U}_1 \end{bmatrix} \\ &= \begin{bmatrix} \alpha & w^T \\ \alpha \left(\frac{v}{\alpha} + f\right) & \hat{z}w^T + \hat{A}_1 + H_1 \end{bmatrix} \\ &= \begin{bmatrix} \alpha & w^T \\ v + \alpha f & \hat{z}w^T + B - \hat{z}w^T + F + H_1 \end{bmatrix} \\ &= \begin{bmatrix} \alpha & w^T \\ v & B \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ \alpha f & F + H_1 \end{bmatrix} = A + H. \end{aligned}$$

Nun ist noch $|H|$ abzuschätzen. Dazu rechnen wir mit b) von oben

$$\begin{aligned} |\hat{A}_1| &= |B - \hat{z}w^T + F| \leq |B| + |\hat{z}||w^T| + |F| \\ &\leq |B| + |\hat{z}||w^T| + 2 \text{ eps } (|B| + |\hat{z}||w^T|) + O(\text{eps}^2) \\ &= (1 + 2 \text{ eps}) (|B| + |\hat{z}||w^T|) + O(\text{eps}^2) \end{aligned}$$

und

$$\begin{aligned}
 |F + H_1| &\leq |F| + |H_1| \\
 &\leq 2 \text{ eps} (|B| + |\hat{z}| |w^T|) + 3(n-2) \text{ eps} (|\hat{A}_1| + |\hat{L}_1| |\hat{U}_1|) + O(\text{eps}^2) \\
 &\leq 2 \text{ eps} (|B| + |\hat{z}| |w^T|) \\
 &\quad + 3(n-2) \text{ eps} ((1+2 \text{ eps}) (|B| + |\hat{z}| |w^T|) + |\hat{L}_1| |\hat{U}_1|) + O(\text{eps}^2) \\
 &\leq 3(n-1) \text{ eps} (|B| + |\hat{z}| |w^T| + |\hat{L}_1| |\hat{U}_1|) + O(\text{eps}^2)
 \end{aligned}$$

da $3(n-2) + 2 = 3n-4 \leq 3(n-1)$. Damit nun zum letzten Streich:

$$\begin{aligned}
 |H| &= \begin{bmatrix} 0 & 0 \\ |\alpha f| & |F + H_1| \end{bmatrix} \\
 &\leq \begin{bmatrix} 0 & 0 \\ \text{eps} |v| & 3(n-1) \text{ eps} (|B| + |\hat{z}| |w^T| + |\hat{L}_1| |\hat{U}_1|) \end{bmatrix} + O(\text{eps}^2) \\
 &\leq 3(n-1) \text{ eps} \begin{bmatrix} 0 & 0 \\ |v| & (|B| + |\hat{z}| |w^T| + |\hat{L}_1| |\hat{U}_1|) \end{bmatrix} + O(\text{eps}^2) \\
 &\leq 3(n-1) \text{ eps} \left(\begin{bmatrix} |\alpha| & |w^T| \\ |v| & |B| \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ |\hat{z}| & |\hat{L}_1| \end{bmatrix} \begin{bmatrix} |\alpha| & |w|^T \\ 0 & |\hat{U}_1| \end{bmatrix} \right) + O(\text{eps}^2) \\
 &= 3(n-1) \text{ eps} (|A| + |\hat{L}| |\hat{U}|) + O(\text{eps}^2).
 \end{aligned}$$

wobei einige Terme hinzugefügt wurden.

□

Nun können wir alle Resultate zusammen nutzen um den Rundungsfehler bei der Lösung eines linearen Gleichungssystems abzuschätzen.

Satz 9.4. $A \in \mathbb{F}^{n \times n}$ und $b \in \mathbb{F}^n$ seien gegebene Matrix und rechte Seite eines linearen Gleichungssystems. \hat{L} und \hat{U} seien die in Fließkommaarithmetik berechneten Faktoren in der LU-Zerlegung von A . Weiter sei \hat{y} die in Fließkommaarithmetik berechnete Lösung von $\hat{L}y = b$ und \hat{x} die in Fließkommaarithmetik berechnete Lösung von $\hat{U}x = \hat{y}$. Dann gilt für \hat{x} :

$$(A + E)\hat{x} = b, \quad |E| \leq n \text{ eps} (3|A| + 5|\hat{L}| |\hat{U}|) + O(\text{eps}^2).$$

Beweis. Nach Satz 9.2 gilt

$$\begin{aligned}
 (\hat{L} + F)\hat{y} &= b, & |F| &\leq n \text{ eps} |\hat{L}| + O(\text{eps}^2), \\
 (\hat{U} + G)\hat{x} &= \hat{y}, & |G| &\leq n \text{ eps} |\hat{U}| + O(\text{eps}^2),
 \end{aligned}$$

also

$$(\hat{L} + F)(\hat{U} + G)\hat{x} = (\hat{L}\hat{U} + F\hat{U} + \hat{L}G + FG)\hat{x} = b.$$

Nach Satz 9.3 gilt für die numerisch berechnete LU-Zerlegung von A

$$A + H = \hat{L}\hat{U} \quad |H| \leq 3(n-1) \text{ eps} (|A| + |\hat{L}| |\hat{U}|) + O(\text{eps}^2)$$

und somit

$$(A + E)\hat{x} = b \quad E = H + F\hat{U} + \hat{L}G + FG.$$

Bleibt noch $|E|$ abzuschätzen:

$$\begin{aligned} |E| &= |H + F\hat{U} + \hat{L}G + FG| \\ &\leq |H| + |F||\hat{U}| + |\hat{L}||G| + O(\text{eps}^2) \\ &\leq 3(n-1) \text{ eps } (|A| + |\hat{L}||\hat{U}|) + n \text{ eps } |\hat{L}||\hat{U}| + |\hat{L}|n \text{ eps } |\hat{U}| + O(\text{eps}^2) \\ &\leq n \text{ eps } (3|A| + 5|\hat{L}||\hat{U}|) + O(\text{eps}^2). \end{aligned}$$

□

9.4 Pivotisierung

Wir wollen uns nun der Frage zuwenden in wie weit die Gaußelimination ein numerisch stabiles Verfahren zur Lösung eines linearen Gleichungssystems ist. Dazu erinnern wir uns: ein Verfahren heißt numerisch stabil falls die Verstärkungsfaktoren aus der Rundungsfehleranalyse die Verstärkungsfaktoren aus der Konditionsanalyse nicht übersteigen.

Fassen wir die Situation nochmal zusammen:

- 1) $A \in \mathbb{F}^{n \times n}$ und $b \in \mathbb{F}^n$ sind eine gegebene Matrix und eine rechte Seite in bereits in Fließkommazahlen. Die exakte Lösung bezeichnen wir $x \in \mathbb{R}^n$, also $Ax = b$.
- 2) Die numerisch bestimmte, also mit Rundungsfehlern behaftete, Lösung \hat{x} des Gleichungssystems bezeichnen wir mit $\hat{x} \in \mathbb{F}^n$. Für diese gilt nach Satz 9.4 dann $(A + E)\hat{x} = b$ mit $|E| \leq n \text{ eps } (3|A| + 5|\hat{L}||\hat{U}|) + O(\text{eps}^2)$.
- 3) Nun betrachten wir A und b zusätzlich als *gerundete* Varianten von $\check{A} \in \mathbb{R}^{n \times n}$ und $\check{b} \in \mathbb{R}^n$. Damit gilt $A = \check{A} - \Delta A$ und $b = \check{b} - \Delta b$ mit $|\Delta A| \leq \text{eps } A$ und $|\Delta b| \leq \text{eps } b$. Mit \check{x} bezeichnen wir die exakte Lösung des ungerundeten Systems, also $\check{A}\check{x} = \check{b} \Leftrightarrow (A + \Delta A)\check{x} = b + \Delta b$.

In der nun folgenden Analyse benutzen wir die Norm $\|\cdot\|_\infty$ für Vektoren und die zugeordnete Matrixnorm. Für diese (und auch die 1-Norm) gelten $\|A\|_\infty = \|\|A\|\|_\infty$ bzw. $\|b\|_\infty = \|\|b\|\|_\infty$.

Bei der numerischen Stabilität müssen wir Konditions- und Rundungsfehleranalyse vergleichen. Betrachten wir daher zunächst die Konditionsanalyse, d.h. welchen Einfluss die Rundungsfehler bei der Eingabe auf die Lösung haben. Mit dem Störungssatz 6.5 und den oben getroffenen Vereinbarungen gilt

$$\begin{aligned} \frac{\|\check{x} - x\|_\infty}{\|x\|_\infty} &\leq \frac{\text{cond}_\infty(A)}{1 - \text{cond}_\infty(A) \frac{\|\Delta A\|_\infty}{\|A\|_\infty}} \left(\frac{\|\Delta A\|_\infty}{\|A\|_\infty} + \frac{\|\Delta b\|_\infty}{\|b\|_\infty} \right) \\ &\leq \frac{\text{cond}_\infty(A)}{1 - \text{cond}_\infty(A) \frac{\|\Delta A\|_\infty}{\|A\|_\infty}} 2 \text{ eps}. \end{aligned} \tag{9.1}$$