PROLOG – WIESO NUMERIK?

S. BARTELS, 14.12.2024

1. Ziele und Konzepte

Die *numerische Mathematik* oder kurz *Numerik* ist mit der praktischen Umsetzung mathematischer Konzepte befasst, um zum Beispiel reale Vorgänge zu berechnen. Dies kann das Infektionsgeschehen einer Pandemie sein, die Auswertung und Visualisierung einer medizinischen Computertomographie, die Realisierung von Suchalgorithmen im Internet, die Nutzung einer Internetplattform, das Trainieren eines neuronalen Netzes, die Vorhersage des Wetters, die Berechnung von Ozeanströmungen, die Belastbarkeit von Brücken und Gebäuden, die Simulation eines Crashtests oder die Kompression von Daten zur schnellen Übertragung von Informationen. Da in der Regel große Datenmengen auftreten, erfolgt die Umsetzung typischerweise mit Hilfe des Computers, was zu zusätzlichen Besonderheiten führt.

Computer können lediglich einfache arithmetische Operationen durchführen und dies auch nur approximativ, das heißt mit *Rundungsfehlern*. Jede mathematische Aufgabenstellung muss daher auf einfache Probleme reduziert werden. Sehr effizient und robust lassen sich das Lösen linearer Gleichungssysteme sowie die Auswertung expliziter Rechenvorschriften realisieren. Mit diesen zwei Konzepten können viele Aufgaben wie Eigenwertprobleme, restringierte Optimierungsaufgaben, nichtlineare Gleichungen und Datenkompressionsprobleme approximativ gelöst werden.

Bei der Entwicklung von Verfahren können jedoch unerwartete Effekte auftreten. So können beispielsweise äquivalente Formeln zu unterschiedlichen Ergebnissen bei ihrer Umsetzung am Computer führen, verschiedene Folgen mit demselben Grenzwert können unterschiedlich schnell konvergieren und Rundungsfehler können sich im Laufe einer Berechnung aufsummieren. Da Rundungsfehler ohnehin unvermeidbar sind, ist es zudem weder notwendig noch sinnvoll, exakte Lösungen von Problemen zu bestimmen.

Der erste Teil widmet sich der schnellen und robusten *Lösung linearer Gleichungssysteme* mit regulären Matrizen $A \in \mathbb{R}^{n\times n}$, das heißt zu einem gegebenen Vektor $b \in \mathbb{R}^n$ die Bestimmung von $x \in \mathbb{R}^n$ mit

$$Ax = b.$$

Dabei ist es besonders wichtig zu verstehen, wie sich Störungen von Daten auf die Lösung auswirken. Darauf aufbauend werden überbestimmte Gleichungssysteme beziehungsweise Ausgleichsprobleme, Eigenwertprobleme sowie lineare Optimierungsprobleme betrachtet.

Der Kern des zweiten Teils ist die *Approximation von Funktionen* mit einfach darstellbaren Funktionen wie beispielsweise stückweise poynomiellen Funktionen s_h, sodass eine vorgegebene Genauigkeit $\varepsilon > 0$ erzielt wird, das heißt

$$\|f - s_h\|_{C^0(I)} \leq \varepsilon.$$

Dies kann verwendet werden, um die Berechnung von Ableitungen und Integralen auf einfache Probleme zurückzuführen. Weitere Aspekte sind die Berechnung von Null- und Minimalstellen.
Im dritten Teil wird die numerische *Approximation gewöhnlicher Differenzialgleichungen* beziehungsweise von Anfangswertproblemen untersucht, die die allgemeine Gestalt

$$y'(t) = f\big(t, y(t)\big), \quad y(0) = y_0$$

besitzen. Sie bilden die Grundlage der Simulation zeitabhängiger Probleme. Bereits der einfache Fall $y' = \alpha y$ mit Lösung $y(t) = y_0 e^{\alpha t}$ führt auf Erkenntnisse, die sich auf große Klassen von Problemen übertragen lassen. Mit den Methoden können Flugbahnen von Körpern, Hamiltonsche Systeme zur Beschreibung von Sonnensystemen und eindimensionale Randwertprobleme numerisch approximativ gelöst werden.

2. Schwierigkeiten und Ideen

Wir betrachten einige typische und teilweise überraschende Phänomene der direkten algorithmischen Umsetzung mathematischer Konzepte.

2.1. **Rundungsfehler.** Da binäre Computer nur endlich viele Zahlen darstellen können, sind Rundungsfehler unvermeidbar. Auch wenn moderne Computer mit hoher Genauigkeit rechnen, kann dies leicht zu Schwierigkeiten führen. Erhält beispielsweise eine Partei $n_P = 2\,099\,580$ von insgesamt $n_G = 42 \cdot 10^6$ abgegebenen Stimmen, so liefert ein Computer den Anteil

```
n_P/n_G = 0.0500
```

also vermeintlich $5,00\%$ der Stimmen. Die gesetzliche Fünfprozenthürde sieht jedoch keine Rundung vor und eine exaktere Darstellung des Quotienten zeigt das Ergebnis

$$\frac{n_P}{n_G} = 0.04999000,$$

sodass die Partei nicht die erforderlichen Stimmen erhalten hat. Hier entsteht ein irreführendes Ergebnis durch Rundung bei der visuellen Darstellung der Zahl. Eine weiterer Fehler entsteht durch rundungsbehaftete arithmetische Operationen des Rechners. Die relative Rechengenaunigkeit eines Computers lässt sich bestimmen, indem die Zahl $x = 1$ so lange halbiert wird, bis der Ausdruck $1 + x$ vom Computer nicht mehr von 1 unterschieden wird, siehe Abbildung 1. Eine typische Genauigkeit liegt bei $1 \cdot 10^{-16}$, sodass man

von fünfzehn korrekten Dezimalstellen ausgehen kann. Statt die Rechengenauigkeit in Bezug zu einer einzelnen Stimme zu setzen, lässt sich die Fünfprozenthürde einfacher mit der Ungleichung $n_P/n_G \geq 1/20$ beziehungsweise $20n_P \geq n_G$ prüfen.

```
% machine_precision.m
x = 1;
while 1+x > 1
   x = x/2;
end
disp(2*x);
```

```
>> machine_precision
   2.2204e-16
```

ABBILDUNG 1. Bestimmung der Maschinengenauigkeit (links) und Ergebnis der Berechnung (rechts).

2.2. **Konvergenzgeschwindigkeit.** Die Zahl $\sqrt{2}$ lässt sich durch sukzessives Bestimmen der Dezimalstellen konstruieren. Ausgehend von $r_0 = 1$ werden Dezimalstellen hinzugefügt, um Zahlen r_k mit k Nachkommastellen zu erhalten, die maximal mit der Eigenschaft $r_k^2 < 2$ sind. Im ersten Schritt wird also $r_1 = 1,4$ gesetzt, da $(1,5)^2 > 2$ gilt. Im k-ten Schritt gelte $r_{k-1}^2 < 2$ und

$$r_k = r_{k-1} + \ell \cdot 10^{-k}$$

wobei $\ell \in \{0, 1, \ldots, 9\}$ maximal gewählt wird, sodass wiederum $r_k^2 < 2$ gilt. Man erhält also in jedem Schritt eine weitere korrekte Dezimalstelle und entsprechend gilt für den Fehler

$$\delta_k = \left|\sqrt{2} - r_k\right| < 10^{-k}.$$

Der Fehler wird in jedem Schritt um den Faktor $q = 1/10$ reduziert. Mit einem Trick erhält man Approximationen, bei denen sich die Anzahl korrekter Dezimalstellen in jedem Schritt verdoppelt. Dazu betrachten wir allgemeiner die Berechnung von $\sqrt{a}$ für eine positive Zahl $a > 0$. Die Gleichung $x^2 = a$ ist offensichtlich äquivalent zu

$$x^2 = \frac{1}{2}x^2 + \frac{1}{2}a \quad \Longleftrightarrow \quad x = \frac{1}{2}\Big(x + \frac{a}{x}\Big).$$

Die zweite Identität charakterisiert die Lösung als Fixpunkt x^* einer Funktion $x \mapsto \Phi(x)$ und diese Beobachung kann man zur Definition der *Fixpunktiteration*

$$x_{k+1} = \Phi(x_k) = \frac{1}{2}\Big(x_k + \frac{a}{x_k}\Big)$$

mit einem geeigneten Startwert $x_0 > 0$ verwenden, was als *Verfahren von Heron* bezeichnet wird. Hierbei kann man sogenannte quadratische Konvergenz der Fehler $e_k = |\sqrt{a} - x_k|$ nachweisen, das heißt

$$e_{k+1} \leq c e_k^2$$

beziehungsweise die Verdopplung korrekter Dezimalstellen in jedem Schritt, sofern $ce_k^2 < 1$ gilt. Eine Realisierung findet sich in Abbildung 2. Die Konvergenzgeschwindigkeit einer Fixpunktiteration lässt sich mit einer Taylor-Approximation quantifizieren. Gilt $\Phi'(x_*) = 0$, so folgt

$$x_{k+1} - x_* = \Phi(x_k) - \Phi(x_*) = \frac{1}{2}\Phi''(\xi)(x_k - x_*)^2,$$

was eine *lokale, quadratische Konvergenz* impliziert. Gilt $\Phi'(x_*) \neq 0$, so folgt analog die *lokale, lineare Konvergenz* $e_{k+1} \leq qe_k$, falls $|\Phi'(x)| \leq q < 1$ für alle $x \in B_\varepsilon(x_*)$ gilt. Typische Verläufe entsprechender Fixpunktiterationen sind in Abbildung 3 dargestellt.

```
% heron.m
a = 2.0; delta = 1.0e-15;
x = a/2; e = abs(x-sqrt(a));
while e > delta
    x = (x+a/x)/2;
    e = abs(x-sqrt(a));
    disp([x,e]);
end
```

```
>> format shortE
>> heron
   1.5000e+00   8.5786e-02
   1.4167e+00   2.4531e-03
   1.4142e+00   2.1239e-06
   1.4142e+00   1.5947e-12
   1.4142e+00   2.2204e-16
```

Abbildung 2. Berechnung der Quadratwurzel nach Heron (links) und Resultate der Berechnung (rechts).

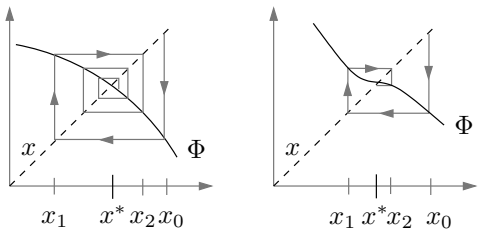
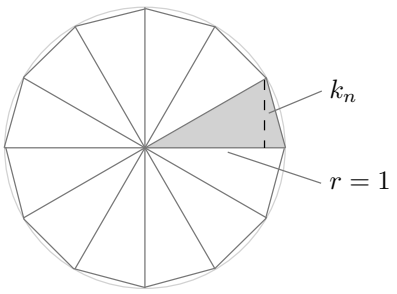


Abbildung 3. Lineare (links) und quadratische (rechts) Konvergenz von Fixpunktiterationen.

2.3. **Instabilitäten.** Rundungsfehler können sich bei Problemen mit gewissen schlechten Eigenschaften stark bemerkbar machen. Als Beispiel betrachten wir die Approximation der Kreiszahl π durch den Flächeninhalt des Einheitskreises. Dazu wird der Kreis wie in Abbildung 4 mit n kongruenten Dreiecken approximiert, deren Höhen mit k_n bezeichnet werden, sodass durch $A_n = nk_n/2$ die Fläche $A = \pi$ angenähert wird. Dieses Vorgehen wurde bereits von Archimedes im dritten Jahrhundert vor Christus verwendet.



j	A_n	e_n
0	3.000000000000000	1.4159×10^{-1}
1	3.105828541230250	3.5764×10^{-2}
2	3.132628613281237	8.9640×10^{-3}
11	3.141592618640789	3.4949×10^{-8}
12	3.141592645321216	8.2686×10^{-9}
15	3.141592645321216	8.2686×10^{-9}
16	3.141593669849427	1.0163×10^{-6}
19	3.141586839655041	5.8139×10^{-6}
23	3.159806164941135	1.8214×10^{-2}
27	6.000000000000000	2.8584

ABBILDUNG 4. Approximation der Einheitskreisfläche mit n Dreiecken (links) und numerisch bestimmte Flächeninhalte A_n sowie Fehler $e_n = |A_n - \pi|$ mit $n = 2^j \cdot 12$ (rechts).

Es gilt $k_n = \sin(2\pi/n)$, es sollen jedoch nur Grundoperationen und die Quadratwurzel verwendet werden. Mit der Identität $\sin\alpha = 2\sin(\alpha/2)\cos(\alpha/2)$ und der pq-Formel ergibt sich die *Rekursionsformel*

$$2k_{2n}^2 = 1 - \sqrt{1 - k_n^2}.$$

Aus $\sin(\pi/6) = 1/2$ erhält man den Startwert $k_{12} = 1/2$ und kann damit eine Folge von Höhen bestimmen. Die mit dem in Abbildung 5 dargestellten Programm erzeugten und in Abbildung 4 aufgeführten Ergebnisse zeigen, dass die Approximationen von π zunächst besser werden, dann stagnieren und schließlich völlig unbrauchbar werden. Nutzt man jedoch eine binomische Formel, so erhält man die äquivalente Darstellung

$$2k_{2n}^2 = \big(1 - \sqrt{1 - k_n^2}\big)\frac{1 + \sqrt{1 - k_n^2}}{1 + \sqrt{1 - k_n^2}} = \frac{k_n^2}{1 + \sqrt{1 - k_n^2}}.$$

Mit dieser Formel lässt sich π bis auf Maschinengenauigkeit annähern. Es zeigt sich, dass allgemein die Subtraktion nahezu gleichgroßer Zahlen vermieden werden sollte.

```
% pi_approx.m
n = 12; k = 0.5; J = 30;
for j = 1:J
   n = 2*n;
   k = sqrt((1-sqrt(1-k^2))/2);
   A = n*k/2; e = abs(pi-A);
   disp([j,A,e]);
end
```

```
% pi_approx_mod.m
n = 12; k = 0.5; J = 30;
for j = 1:J
   n = 2*n;
   k = k/sqrt(2*(1+sqrt(1-k^2)));
   A = n*k/2; e = abs(pi-A);
   disp([j,A,e]);
end
```

ABBILDUNG 5. Approximation der Kreiszahl π mit direkter (links) und modifizierter (rechts) Berechnung der Höhen k_n.

2.4. **Rechenaufwand.** Die Berechnung der Determinante einer quadratischen Matrix $A \in \mathbb{R}^{n\times n}$ lässt sich mit dem Laplaceschen Entwicklungssatz durchführen. Mit der Rekursionsformel

$$\det A = \sum_{j=1}^{n} (-1)^{1+j} a_{1j} \det \widehat{A}_{1j},$$

wobei $\widehat{A}_{1j}$ die Teilmatrix ist, die durch Streichen der ersten Zeile und j-ten Spalte entsteht, kann die Berechnung so lange auf die Bestimmung von Determinanten kleinerer Matrizen zurückgeführt werden, bis schließlich Matrizen mit nur einem Eintrag auftreten, Abbildung 6 zeigt eine praktische Umsetzung. Der Rechenaufwand wächst allerdings dramatisch, beim Übergang von $n = 8$ auf $n = 10$ steigt die Rechenzeit um einen Faktor $90 = 9 \cdot 10$ und für Matrizen der Dimension $n \geq 12$ ist das Verfahren kaum noch in vertretbarer Zeit durchführbar. Praktisch und theoretisch sieht man, dass $n!$ Rechenoperationen notwendig sind. Alternativ dazu liefert das Gaußsche Eliminationsverfahren eine Faktorisierung $A = LU$ mit Dreiecksmatrizen L und U, wobei für die Diagonaleinträge von L insbesondere $\ell_{ii} = 1$ gefordert werden kann. Damit folgt mit den Rechenregeln für die Determinante, dass

$$\det A = \det L \det U = (\ell_{11}\ell_{22}\dots\ell_{nn})(u_{11}u_{22}\dots u_{nn}) = u_{11}u_{22}\dots u_{nn},$$

wobei ausgenutzt wurde, dass die Determinante einer Dreiecksmatrix durch das Produkt der Diagonaleinträge beziehungsweise Eigenwerte gegeben ist. Ist also die Faktorisierung gegeben, so lässt sich die Determinante mit $n-1$ Rechenoperationen bestimmen. Eine Überprüfung des Eliminationsverfahrens zeigt, dass die Faktorisierung mit n^3 Rechenoperationen bestimmt werden kann, erforderliche Zeilenvertauschungen können in die Überlegungen einbezogen werden.

2.5. **Robustheit bei Störungen.** Rundungsfehler lassen sich als Störungen auffassen und so kann man abstrakt beurteilen, ob sich ein Problem überhaupt und unabhängig von speziellen Algorithmen approximieren beziehungsweise numerisch lösen lässt. Zur Veranschaulichung betrachten wir die Bestimmung der Nullstellen eines Polynoms. Konkret wählen wir

$$p(x) = (x-a)^n - 0$$

mit einer gegebenen Zahl a, die dann die n-fache Nullstelle des Polynoms ist. Wir stören den Subtrahenden 0 und subtrahieren stattdessen eine kleine Zahl $\varepsilon > 0$, das heißt wir betrachten das Polynom

$$p_\varepsilon(x) = (x-a)^n - \varepsilon.$$

Die in Abbildung 7 dargestellten komplexen Nullstellen sind gegeben durch $\widetilde{x}_k = a + s_k \varepsilon^{1/n}$ mit den n-ten Einheitswurzeln $s_k = e^{\mathrm{i}2\pi k/n}$, $k = 1, 2, \dots, n$, die gleichmäßig auf dem Rand des Einheitskreises in der komplexen Ebene verteilt sind, im Fall $n = 2$ sind es $s_1 = -1$ und $s_2 = 1$. Der Fehler zwischen den korrekten Nullstellen $x_k = a$ und denen des gestörten Polynoms beträgt

```
% det_laplace.m
function val = det_laplace(A)
n = size(A,1); val = 0;
if n == 1
    val = A(1,1);
else
   for j = 1:n
      I = 2:n;
      J = [1:j-1,j+1:n];
      hat_A_1j = A(I,J);
      val = val+(-1)^(1+j)...
          *A(1,j)...
          *laplace(hat_A_1j);
    end
end
```

```
% det_laplace_hilb.m
for n = 4:2:10
   A = hilb(n);
   tic; d = det_laplace(A); toc
end
```

```
>> det_laplace_hilb
Elapsed time is 0.000814 seconds.
Elapsed time is 0.002340 seconds.
Elapsed time is 0.108463 seconds.
Elapsed time is 9.220774 seconds.
```

ABBILDUNG 6. Berechnung der Determinante mit dem Laplaceschen Entwicklungssatz (links) und Laufzeiten für Matrixgrößen $n = 4, 6, 8, 10$ (rechts).

$e_k = |x_k - \widetilde{x}_k| = \varepsilon^{1/n}$ und dieser wird kleiner, wenn ε kleiner wird. Problematisch ist jedoch, dass das Verhältnis vom Ausgabe- zum Eingabefehler also

$$\frac{\max_{k=1,\dots,n} |x_k - \widetilde{x}_k|}{\|p - p_\varepsilon\|_{C^0(\mathbb{R})}} = \frac{\varepsilon^{1/n}}{\varepsilon} = \varepsilon^{(1-n)/n}$$

unbeschränkt ist für $\varepsilon \to 0$ und $n \geq 2$. Kleine Störungen in den Daten des Problems wirken sich also überproportional im Ergebnis aus. Man bezeichnet daher die Nullstellenbestimmung von Polynomen als *schlecht konditioniertes Problem*. Unmittelbar damit verbunden ist die schlechte Konditionierung der Bestimmung von Eigenwerten einer Matrix. Ein schlecht konditioniertes Problem des realen Lebens ist das senkrechte Aufstellen eines Stifts.

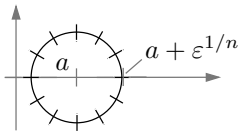


ABBILDUNG 7. Die (komplexen) Nullstellen des gestörten Polynoms $p_\varepsilon(x) = (x - a)^n - \varepsilon$ liegen auf der Kreislinie um a mit Radius $r = \varepsilon^{1/n}$.

2.6. **Inexaktes Lösen.** Das Gaußsche Eliminationsverfahren zur Lösung eines linearen Gleichungssystems führt auf einen Aufwand von n^3 Rechenoperationen. Ein im Sinne der Rechnerarithmetik exaktes Lösen ist aber selten erforderlich, da nicht nur Rundungsfehler das Ergebnis beeinflussen,

sondern die Daten auch durch Mess- und Modellfehler nicht als exakt angesehen werden können. Diese Beobachtung führt auf die Idee, dass man durch lediglich näherungsweises Lösen des linearen Gleichungssystems den Rechenaufwand erheblich reduzieren kann. Ein Ansatz basiert auf der Zerlegung der Matrix A in ihren Diagonalanteil D und den Rest $R = A - D$. Sofern D regulär ist, ist die Gleichung $Ax = b$ damit äquivalent zu den Gleichungen

$$Dx = b - Rx \quad \Longleftrightarrow \quad x = D^{-1}(b - Rx).$$

Die zweite Gleichung lässt sich dabei als Fixpunktgleichung $x = \Phi(x)$ interpretieren und führt auf die *Iteration*

$$x_{k+1} = D^{-1}(b - Rx_k)$$

mit einem Startvektor $x_0 \in \mathbb{R}^n$. In einigen Fällen ergeben sich in wenigen Schritten gute Approximationen. Die Auswertung der rechten Seite erfordert allgemein einen Aufwand von n^2 Rechenoperationen, in vielen Fällen hat A beziehungsweise R jedoch viele verschwindende Einträge und der Aufwand ist nur ein moderates Vielfaches cn von n. Wenn die Iteration schnell konvergiert, wird somit der Aufwand zur Lösung des Systems von n^3 auf $\widetilde{c}n$ reduziert, was bei typischen Größen von n im Bereich $[10^2, 10^7]$ enorm ist. Um diesen Aspekt auszunutzen, muss in dem in Abbildung 8 dargestellten Programm die Definition von A modifiziert werden, damit unnötige Multiplikationen mit Null vermieden werden. Ein besseres Konvergenzverhalten erzielt man mit der Iteration $x_{k+1} = (D + U)^{-1}(b - Lx_k)$, wobei U und L die Teilmatrizen von A oberhalb beziehungsweise unterhalb der Diagonalen sind und in jedem Schritt ein Gleichungssystem mit Dreiecksmatrix $D + U$ gelöst werden muss. Rundungsfehler sind hier unproblematisch, da konvergente Fixpunktiterationen einen selbststabilisierenden Effekt haben in dem Sinne, dass jede Iterierte als neuer Startwert angesehen werden kann.

```
% jacobi_iteration.m
n = 10^2; b = ones(n,1);
e = ones(n,1); e_s = ones(n-1,1);
A = diag(4*e,0)-diag(e_s,1)-diag(e_s,-1);
% A = spdiags([-e,4*e,-e],[-1,0,1],n,n);
D = diag(A); D_inv = D.^(-1); R = A-diag(D);
x = zeros(n,1); tol = 1.0e-3; ctr = 0;
while norm(A*x-b) > tol
    x = D_inv.*(b-R*x); ctr = ctr+1; disp(ctr);
end
```

ABBILDUNG 8. Lösen eines Gleichungssystems mit der Jacobi-Iteration, die alternative Definition der Bandmatrix A vermeidet unnötige Multiplikationen mit Nulleinträgen.

$$A = \begin{bmatrix} 4 & -1 & & \\ -1 & \ddots & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 4 \end{bmatrix}$$

n	A vollbesetzt	A dünnbesetzt
10^2	0.005273 s	0.047754 s
10^3	0.028120 s	0.009399 s
10^4	1.042249 s	0.023457 s
10^5	—	0.106429 s
10^6	—	0.512903 s

ABBILDUNG 9. Werden überflüssige Multiplikationen bei Bandmatrizen vermieden, so führt das iterative Verfahren auch bei sehr großen Matrizen zu geringem Speicherbedarf und kurzen Rechenzeiten.

2.7. **Approximation mit Polynomen.** Ein Satz von Weierstraß besagt, dass sich jede stetige Funktion auf einem kompakten Intervall beliebig gut durch Polynome approximieren lässt. Allerdings zeigen diese Resultate nicht, wie man die Polynome findet beziehungsweise welchen Polynomgrad man benötigt, um eine vorgegebene Genauigkeit zu erzielen. Zur Berechnung solcher Polynome können paarweise verschiedene Punkte $x_0, x_1, \ldots, x_n$ im Intervall $[a, b]$ gewählt und ein Polynom p durch die Forderung

$$p(x_i) = f(x_i), \quad i = 0, 1, \ldots, n,$$

definiert werden. Um diese $n+1$ *Interpolationsbedingungen* zu erfüllen, muss das Polynom mindestens den Grad n besitzen. Aus dem Hauptsatz der Algebra folgt, dass ein Polynom mit diesem Grad eindeutig definiert ist. Mit einer Basis $(p_j)_{j=0,\ldots,n}$ wie beispielsweise den Monomen $p_j(x) = x^j$, ergibt sich der Koeffizientenvektor $c \in \mathbb{R}^{n+1}$ von p aus dem Gleichungssystem $Ac = f$ mit $A_{ij} = p_j(x_i)$ und $f_i = f(x_i)$, $i, j = 0, 1, \ldots, n$. Bei gewissen Funktionen f und gleichmäßig verteilten Punkten $x_0, x_1, \ldots, x_n$ beobachtet man jedoch, dass die Polynome für größer werdende Zahlen n nicht uniform konvergieren, siehe Abbildungen 10 und 11. Mit Hilfe des Satzes von Rolle sieht man, dass die Abstände zwischen den Stützstellen am Rand kleiner gewählt werden sollten, was durch sogenannte *Tschebyscheff-Knoten* optimal realisiert wird. Neben diesem Effekt ist zu beachten, dass die Monombasis zu einer Matrix A mit ungünstigen Eigenschaften hinsichtlich kleiner Störungen führt.

2.8. **Wahl geeigneter Basen.** Jeder Vektor $x \in \mathbb{R}^n$ lässt sich bezüglich der kanonischen Basis $e_1, e_2, \ldots, e_n$ darstellen, das heißt es gilt

$$x = \sum_{k=1}^{n} \alpha_k e_k,$$

wobei die Koeffizienten α_k gerade den Komponenten des Vektors entsprechen. Hat der Vektor x besondere Eigenschaften, ist er beispielsweise als abgetastetes Audiosignal zu Zeitpunkten $t_1, t_2, \ldots, t_n$ gegeben, so ist es sinnvoll, eine Basis $v_1, v_2, \ldots, v_n$ zu wählen, die diese Eigenschaft berücksichtigt. In diesem Fall haben viele Koeffizienten in der Linearkombination die Eigenschaft betragsmäßig sehr klein beziehungsweise vernachlässigbar zu sein,

```
% interpolation.m
f = @(x) 1./(1+25*x.^2);
delta = 0.01; X = (-1:delta:1); Y = f(X); n = 11;
x_eq = zeros(n+1,1); y_eq = zeros(n+1,1);
x_ch= zeros(n+1,1); y_ch = zeros(n+1,1);
dx = 2/n; dtheta = pi/(2*(n+1));
for k = 1:n+1
    x_eq(k) = -1+(k-1)*dx; y_eq(k) = f(x_eq(k));
    x_ch(k) = cos((2*k-1)*dtheta); y_ch(k) = f(x_ch(k));
end
p_eq = polyfit(x_eq,y_eq,n); p_ch = polyfit(x_ch,y_ch,n);
plot(X,Y,'--',x_eq,y_eq,'o',X,polyval(p_eq,X));
title('equidistant'); pause
plot(X,Y,'--',x_ch,y_ch,'o',X,polyval(p_ch,X));
title('chebyshev');
```

ABBILDUNG 10. Berechnung und Darstellung eines Interpolationspolynoms mit gleichmäßig und ungleichmäßig verteilten Stützstellen.

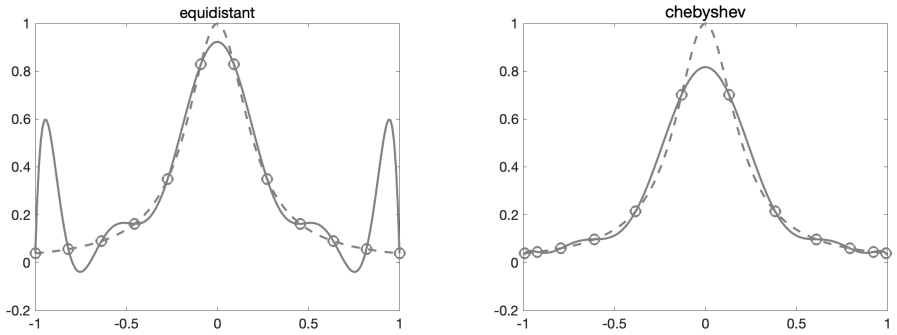


ABBILDUNG 11. Polynominterpolation mit äquidistanten Stützstellen (links) und Tschebyscheff-Knoten (rechts).

sodass gilt

$$x = \sum_{k=1}^{n} \beta_k v_k \approx \sum_{\ell=1}^{m} \beta_{k_\ell} v_{k_\ell}, \quad m \ll n.$$

Ist zum Beispiel $n = 10^4$, so kann der Vektor x oft mit $m \sim 10^2$ relevanten Informationen gut dargestellt werden. Dies bezeichnet man als *Datenkompression*, was die Basis des digitalen Zeitalters ist. Die mathematische Herausforderung besteht dabei in der effizienten Umsetzung des Basiswechsels. Werden die Vektoren $(v_k)_{k=1,\dots,n}$ als Grundschwingungen gewählt, so ermöglicht die *schnelle Fouriertransformation* einen nahezu optimalen Basiswechsel. Ein Beispiel ist in Abbildung 12 dargestellt.

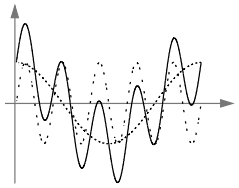


ABBILDUNG 12. Funktionen lassen sich häufig als Summe von Sinus-Schwingungen darstellen.

2.9. **Große Zwischenergebnisse.** Zeilenvertauschungen sind beim Gaußschen Eliminationsverfahren nur dann erforderlich, wenn sogenannte Pivot-Elemente, mit denen die Elimination von Einträgen unterhalb der Diagonale durchgeführt wird, identisch Null sind. Zur Vermeidung von Instabilitäten beziehungsweise starken Auswirkungen von Rundungsfehlern sollten Zeilenvertauschungen auch dann durchgeführt werden, wenn Pivot-Elemente klein im Vergleich zu anderen Einträgen sind. Andernfalls treten betragsmäßig große Zwischenergebnisse auf, wie man am Beispiel

$$\begin{bmatrix} \varepsilon & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

mit Lösung $(x_1, x_2) \approx (1, 1)$ für $0 < \varepsilon \ll 1$ prüft. Dass diese Zwischenergebnisse auf große Rechenfehler führen können, zeigt die Störungsrechnung für die Summe $s = y_1 + y_2 + \cdots + y_n$ mit exakten Summanden y_i und gestörten Werten $\widetilde{y}_i = (1 + \sigma_i\varepsilon_i)y_i$, mit $\sigma_i \in \{\pm 1\}$ und $\varepsilon_i \geq 0$, sodass für die gestörte Summe $\widetilde{s}$ gilt

$$\widetilde{s} = \sum_{i=1}^{n} \widetilde{y}_i = \sum_{i=1}^{n} (1 + \sigma_i\varepsilon_i)y_i = s + \sum_{i=1}^{n} \sigma_i\varepsilon_i y_i.$$

Für den relativen Fehler im Ergebnis $|s - \widetilde{s}|/|s|$ ergibt sich mit der Dreiecksungleichung und den relativen Fehlern $|\widetilde{y}_i - y_i|/|y_i| = \varepsilon_i$ der Daten, dass

$$\varepsilon_s = \frac{|s - \widetilde{s}|}{|s|} \leq \frac{1}{|s|} \sum_{i=1}^{n} |\varepsilon_i||y_i| \leq \Big(\frac{\sum_{i=1}^{n} |y_i|}{|s|} \Big) \max_{i=1,\ldots,n} \varepsilon_i = \kappa\varepsilon_y.$$

Es kann also eine große Verstärkung des relativen Fehlers auftreten, wenn $|s|$ klein ist im Vergleich zu den absoluten Summanden $|y_i|$. Die erste Ungleichung ist eine Gleichung, wenn die Störungen dasselbe Vorzeichen wie die Summanden haben, und die zweite Ungleichung ist eine Gleichung, wenn alle Störungen gleich groß sind. Das in Abbildung 13 gezeigte Programm berechnet den Wert

$$s = \sqrt{x} + k \exp(x - 1) + k \sin(x3\pi/2)$$

für eine Störung $\widetilde{x} = (1 + \varepsilon_p)x$ von $x = 1$, was auf eine Verstärkung der relativen Fehler um den Faktor $\kappa \approx k$ führt und somit das Resultat bestätigt.

```
% intermediate_vals.m
x = 1.0; s = 1.0;
eps_p = 10^(-5); k = 10^3;
x_p = (1+eps_p)*x;
s_p = sqrt(x_p)+k*exp(x_p-1)...
    +k*sin(3*pi*x_p/2);
e_rel_s = abs((s_p-s)/s);
e_rel_x = abs(eps_p);
kappa_rel = e_rel_s/e_rel_x;
disp(kappa_rel);
```

```
>> intermediate_vals
   1.0006e+03
```

ABBILDUNG 13. Sind Zwischenergebnisse oder Summanden betragsmäßig größer als das Endergbnis, kann es zu einer großen Verstärkung relativer Fehler kommen.

Ein Spezialfall der obigen Abschätzung ist die Subtraktion nahezu gleichgroßer Zahlen, was dem Fall $y_1 \approx -y_2$ entspricht und auf sogenannte *Auslöschungseffekte* führt. Haben beispielsweise zwei Stäbe die Längen $\ell_1 = 101.51$ und $\ell_2 = 100.49$ in Zentimetern und wurden diese approximativ mit $\widetilde{\ell}_1 = 102.00$ und $\widetilde{\ell}_2 = 100.00$ gemessen, so sind die relativen Fehler ε_i, $i = 1, 2$, kleiner als 0.5%, der relative Fehler der Differenzen $\delta = 1,02$ und $\widetilde{\delta} = 2,00$ ist jedoch fast 100.0%, was einer Fehlerverstärkung von $\kappa \approx 200$ entspricht.

2.10. **Abstiegsverfahren.** Um ein (lokales) Minimum einer differenzierbaren Funktion $g : \mathbb{R}^n \to \mathbb{R}$ zu bestimmen, ist es sinnvoll, wie beim Abstieg in einer Gebirgslandschaft, die Funktionswerte schrittweise zu reduzieren. Damit man möglichst rasch zu einem Minimum gelangt, sollte für den jeweils nächsten Iterationsschritt die Richtung mit der lokal größten Reduktion des Funktionswerts gewählt werden. Diese ist gegeben durch den negativen Gradienten der Funktion. Ausgehend von einem Startwert $x_0 \in \mathbb{R}^n$ wird damit eine Folge von Iterierten x_k, $k = 0, 1, \dots$, durch die Vorschrift

$$x_{k+1} = x_k - \alpha_k \nabla g(x_k)$$

bestimmt. Die *Schrittweite* α_k sollte dabei sinnvoll gewählt werden, sodass man nicht tatsächlich wieder einen größeren Funktionswert erhält. Abbildung 14 zeigt einen typischen resultierenden Pfad auf dem Funktionsgraphen. Neben der Optimierung der Schrittweiten ist bei einer Klasse von Minimierungsproblemen auch die Optimerung der Suchrichtungen von Interesse. Bei quadratischen Minimierungsproblemen der Gestalt $g(x) = (1/2)\|Ax - b\|^2$ lässt sich damit sicherstellen, dass die Abstiegsrichtungen in einem geeigneten Sinne orthogonal zueinander sind und man mit maximal n Schritten zum Minimum gelangt.

2.11. **Implizite und explizite Verfahren.** Approximationslösungen des Anfangswertproblems $y' = f(t, y)$, $y(0) = y_0$, erhält man, indem man die

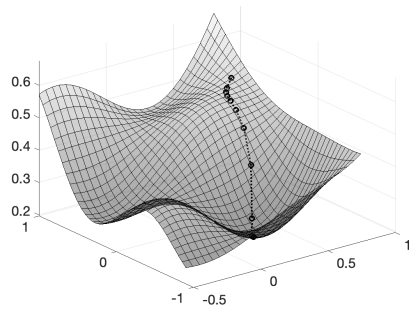


ABBILDUNG 14. Illustration des Abstiegsverfahrens zur Bestimmung eines Minimums einer Funktion.

Ableitung von y durch eine Sekantensteigung

$$y'(t) \approx \frac{y(t+\tau) - y(t)}{\tau}$$

approximiert, was im Fall dieses rechtsseitigen Differenzenquotienten mit Schrittweite $\tau > 0$ und Zeitschritten $t_k = k\tau$ auf das Euler-Verfahren

$$y_{k+1} = y_k + \tau f(t_k, y_k)$$

mit Startwert y_0 führt. Man erhält also eine Folge von Approximationen $(y_k)_{k=0,\dots,K}$ durch einfaches sukzessives oder *explizites* Auswerten der rechten Seite. Experimente im Fall $f(t,y) = \alpha y$ mit $\alpha < 0$ und exakter, beschränkter Lösung $y(t) = y_0 e^{\alpha t}$ zeigen jedoch, dass die Approximationen nur für hinreichend kleine Schrittweiten beschränkt bleiben, siehe Abbildung 15. Dies wird verbessert, wenn man statt des rechtsseitigen Differenzenquotienten einen linksseitigen nimmt, was auf das Verfahren

$$y_k = y_{k-1} + \tau f(t_k, y_k)$$

mit Startwert y_0 führt, siehe Abbildung 16. Der Preis für die besseren Stabilitätseigenschaften des Verfahrens ist jedoch die erforderliche Lösung eines möglicherweise nichtlinearen Gleichungssystems in jedem Iterationsschritt. Man bezeichnet dieses Verfahren daher als *implizites Verfahren*.

2.12. **Mehrtermrekursion.** Um bessere Approximationen von Ableitungen zu erhalten, ist es naheliegend mehr als nur zwei Zeitpunkte zu verwenden, beispielsweise also mit einer Kombination von drei Werten y_{k+2}, y_{k+1}, y_k die Ableitung $y'(t_r)$ zu approximieren, das heißt

$$y'(t_r) \approx \beta_0 y_k + \beta_1 y_{k-1} + \beta_2 y_{k-2}.$$

Mögliche Koeffizienten α_ℓ, $\ell = 0, 1, 2$, ergeben sich aus Taylor-Approximationen, allerdings sind nicht alle so gefundenen Werte gut geeignet. Ein Kriterium findet man durch Betrachten der trivialen Differenzialgleichung $y'(t) = 0$ mit Startwert y_0 und konstanter Lösung $y(t) = y_0$. Ein sinnvolles Verfahren

```
% euler_expl.m
alpha = -2; y_0 = 1; T = 10;
f = @(t,s) alpha*s;
K = 8; tau = T/K;
y = zeros(K+1,1); y(1) = y_0;
for k = 1:K
    t_k = (k-1)*tau;
    y(k+1) = y(k)+tau*f(t_k,y(k));
end
plot(tau*(0:K),y,'o-');
title('expl. Euler');
```

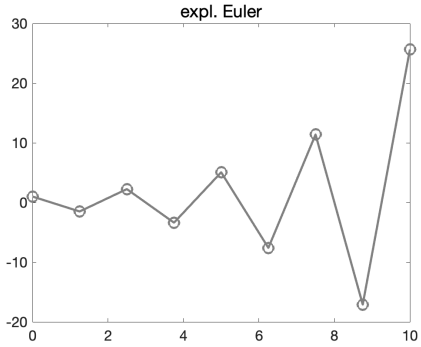


ABBILDUNG 15. Das explizite Euler-Verfahren ist ein einfach realisierbares Verfahren (links), welches jedoch zu unbeschränkten Approximationen führen kann (rechts).

```
% euler_impl.m
alpha = -2; y_0 = 1; T = 10;
K = 16; tau = T/K;
y = zeros(K+1,1); y(1) = y_0;
for k = 2:K+1
   t_k = (k-1)*tau;
   % y(k) = y(k-1)+tau*f(t_k,y(k));
   y(k) = (1-alpha*tau)^(-1)*y(k-1);
end
plot(tau*[0:K],y,'o-');
title('impl. Euler');
```

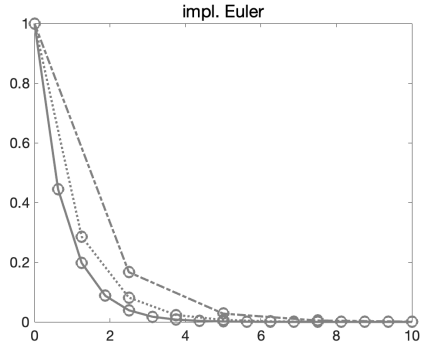


ABBILDUNG 16. Das implizite Euler-Verfahren (links) besitzt bessere Stabilitätseigenschaften als das explizite Verfahren (rechts).

sollte garantieren, dass in diesem Fall Approximationen beschränkt bleiben. Diese erfüllen für gegebene Startwerte y_0, y_1 die Dreitermrekursion

$$\alpha_2 y_{k+2} + \alpha_1 y_{k+1} + \alpha_0 y_k = 0$$

beziehungsweise in Matrixdarstellung mit $\gamma_\ell = -\alpha_\ell/\alpha_2$, $\ell = 0, 1$, die Relation

$$\begin{bmatrix} y_{k+1} \\ y_{k+2} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ \gamma_0 & \gamma_1 \end{bmatrix} \begin{bmatrix} y_k \\ y_{k+1} \end{bmatrix} = A \begin{bmatrix} y_k \\ y_{k+1} \end{bmatrix}.$$

Eine Überführung der Iterationsmatrix A in Jordansche Normalform liefert

$$\text{(i)} \quad J = T^{-1}AT = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}, \qquad \text{(ii)} \quad J = T^{-1}AT = \begin{bmatrix} \lambda_1 & 1 \\ 0 & \lambda_1 \end{bmatrix},$$

mit einer orthogonalen Matrix T und geometrisch einfachen beziehungsweise mehrfachen Eigenwerten $\lambda_1, \lambda_2 \in \mathbb{C}$. Relevant für die Stabilität des numerischen Verfahrens ist nun, ob die Matrix J nicht-expansiv ist, das heißt ob $\|Jz\|_* \leq \|z\|_*$ für alle $z \in \mathbb{C}^2$ mit einer geeigneten Vektornorm $\|\cdot\|_*$ gilt. Im ersten Fall ist dies gegeben, falls $|\lambda_1|, |\lambda_2| \leq 1$, und im zweiten, falls $|\lambda_1| < 1$ gilt. Ein Beispiel einer instabilen Dreitermrekursion ist durch die Koeffizienten $(\alpha_2, \alpha_1, \alpha_0) = (1, 4, -5)$ gegeben, stabil hingegen ist die Wahl $(\alpha_2, \alpha_1, \alpha_0) = (3, -4, 1)$. Die Resultate einer instabilen Iteration sind in Abbildung 17 gezeigt.

```
% multistep_stab.m
a = [-5,4,1];        % unstable
% a = [1,-4,3];      % stable
g_2 = -a(2)/a(3);
g_1 = -a(1)/a(3);
K = 8; delta = .01;
y = zeros(K+1,1);
y(1) = 1; y(2) = 1+delta;
for k = 2:K
    y(k+1) = g_2*y(k)+g_1*y(k-1);
end
plot((0:K),y,'o-');
title('multistep');
```

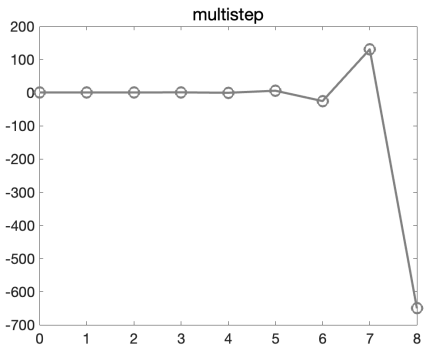


ABBILDUNG 17. Mehrschrittverfahren (links) können auf oszillierende, schnell wachsende Approximationen führen (rechts).

2.13. **Fehlerquellen.** Bei der numerischen Lösung einer mathematischen Aufgabenstellung ergeben sich zahlreiche, meist unvermeidbare Fehler.

- *Modellfehler:* Hierunter versteht man den Fehler der oft vereinfachten Darstellung eines realen Problems mittels mathematischer Gleichungen sowie Messfehler bei der Bestimmung spezifischer Problemeigenschaften.
- *Verfahrensfehler:* Der verwendete Algorithmus zur Lösung eines Problems führt auf Fehler, die durch Approximationen kontinuierlicher Größen wie Ableitungen oder Abbruchkriterien bei iterativen Verfahren verursacht werden.
- *Rundungsfehler:* Sämtliche arithmetische Operationen des Computers müsssen als fehlerbehaftet angesehen werden.

Es stellt sich heraus, dass *relative Fehler* besser zur Beurteilung eines Verfahrens geeignet sind als *absolute Fehler*. Unter der *Konditionierung* eines Problems versteht man die (vom numerischen Verfahren unabhängige) Anfälligkeit des Problems auf Störungen, unter der *Stabilität* eines Verfahrens die

durch die Rechenschritte verursachte Fehlerverstärkung, und unter *Konvergenz* die Verringerung des Verfahrensfehlers, wenn Approximationen verbessert und Abbruchkriterien verringert werden.

3. Vorgehensweise der Numerik

Die Entwicklung numerischer Verfahren zur approximativen Lösung einer mathematischen Aufgabenstellung sollte die folgenden Aspekte berücksichtigen:

- Unerwartete Phänomene beobachten und verstehen
- Methoden entwickeln, die Probleme vermeiden
- Geeignete Fixpunktgleichungen finden
- Dominante Fehlerquellen identifizieren
- Sinnvolle Konvergenzbegriffe verwenden
- Besondere Eigenschaften von Problemklassen nutzen
- Problemangepasste Basen konstruieren
- Bedingungen für Konvergenz kritisch diskutieren

4. Aufgaben

Die folgenden Aufgaben können experimentell oder theoretisch bearbeitet werden.

(a) Bestimmen Sie für $\ell = 1, 2, \ldots, 10$ die kleinste Maschinenzahl x, sodass der Vergleich $10^\ell + x > 10^\ell$ vom Computer als korrekt ausgewertet wird. Interpretieren Sie Ihre Ergebnisse.

(b) Prüfen Sie die Bedingung $\Phi'(x^*) = 0$ für quadratische Konvergenz des Heron-Verfahrens und konstruieren Sie geeignete Startwerte für die Approximation von $\sqrt{a}$, $a > 0$.

(c) Approximieren Sie die Ableitung der Exponentialfunktion am Punkt $x = 1$ durch Sekantensteigungen $(f(x+h)-f(x))/h$ sowie $(f(x+h)-f(x-h))/(2h)$ mit verschiedenen Schrittweiten $h > 0$ und kommentieren Sie Ihre Ergebnisse.

(d) Geben Sie ein möglichst stabiles Verfahren zur approximativen Berechnung von $\exp(x)$ für beliebiges $x \in \mathbb{R}$ an.

(e) Wie lange dauert die Berechnung der Lösung eines linearen Gleichungssystems der Größe $n = 10^\ell$, $\ell = 3, 4, 5, 6$, mit dem Gaußschen Eliminationsverfahren, wenn der Rechner 10^9 Operationen pro Sekunde durchführen kann, und wieviel Speicherplatz wird benötigt?

(f) Testen Sie das Jacobi-Verfahren $x_{k+1} = D^{-1}(b - Rx_k)$ und das Gauß-Seidel-Verfahren $x_{k+1} = (D + U)^{-1}(b - Lx_k)$ für Bandmatrizen $A \in \mathbb{R}^{n\times n}$ mit Nebendiagonaleinträgen -1 und Hauptdiagonaleinträgen $a_{ii} = 2$ beziehungsweise $a_{ii} = 4$ für $i = 1, 2, \ldots, n$.

(g) Testen Sie die Polynominterpolation mit äquidistanten Stützstellen und Tschebyscheff-Knoten im Fall $f(x) = \cos(x)$. Berechnen Sie Extrema einiger Ableitungen der Funktionen $f(x) = \cos(x)$ und $f(x) = (1 + 25x^2)^{-1}$ im Intervall $[-1, 1]$.

(g) Bestimmen Sie eine Basis des Polynomraums vom maximalen Grad 3, sodass sich ein Polynom q mit den Eigenschaften $q(0) = v_0$, $q(1) = v_1$ und $q'(0) = v_2$, $q'(1) = v_3$ mit den Koeffizienten v_0, v_1, v_2, v_3 darstellen lässt.

(h) Bestimmen Sie experimentell Fehlerverstärkungen der Gauß-Elimination ohne Pivotsuche für das Gleichungssystem $Ax = b$, wobei $a_{11} = \varepsilon$, $a_{12} = a_{21} = a_{22} = 1$ und $b_1 = 1 + \varepsilon$ sowie $b_2 = 2$ für ein $\varepsilon > 0$ seien.

(i) Liegt im Falle des Terminierens des Abstiegsverfahrens mit Abbruchkriterium $\|\nabla g(x_k)\| \le \varepsilon$ mit einer gegebenen Zahl $0 < \varepsilon \ll 1$ stets eine Approximation eines globalen Minimums vor?

(j) Diskutieren Sie Fehlerquellen bei der Berechnung der Flugbahn eines Körpers mit der aus den Newtonschen Gesetzen resultierenden Gleichung $x(t) = x_0 + tv_0 + (t^2/2)(0, -g)$ mit $g = 9,81\mathrm{m/s}^2$ und gegebenen $x_0, v_0 \in \mathbb{R}^2$.

(k) Welche besondere Eigenschaft zeigt sich beim impliziten Euler-Verfahren für die Gleichung $y'(t) = 1$, $y(0) = y_0$, auf einem großen Zeitintervall $[0, T]$?

(l) Untersuchen Sie die Stabilität der Dreitermrekursion $y_{k+2} = -2y_{k+1} + y_k$ und testen Sie diese mit den Anfangswerten $y_0 = 1$ und $y_1 = \sqrt{2} - 1$.