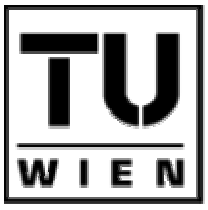


TECHNISCHE UNIVERSITÄT WIEN

VIENNA UNIVERSITY OF TECHNOLOGY

DIPLOMARBEIT

ITERATIONSVERFAHREN FÜR LINEARE UND NICHTLINEARE GLEICHUNGSSYSTEME

ausgeführt am Institut für
Analysis und Scientific Computing
der Technischen Universität Wien

unter der Anleitung von
Ao.Univ.Prof. Mag.rer.nat. Dr.rer.nat. Gabriela Schranz-Kirlinger

durch

Eva Roth
1220 Wien, Am Kaisermühlendamm 101/23

_____ Datum

_____ Unterschrift (Studentin)

Inhaltsverzeichnis

1 Einleitung

Einleitend möchte ich einerseits die Gründe der Wahl meines Diplomarbeitsthemas „Iterationsverfahren für lineare und nichtlineare Gleichungssysteme“ näher beleuchten und andererseits meine Schwerpunktsetzungen bzw. die einzelnen Kapitel dieser Arbeit kurz erläutern.

Die Verweise auf die verwendete Literatur sind in eckiger Klammer und Großbuchstaben in Anlehnung an die Anfangsbuchstaben des/der AutorInnen und/oder dem Inhalt der entsprechenden Literaturquelle angegeben (vgl. Literaturverzeichnis). Auch Seitenangaben habe ich angeführt, wenn ich sie als wichtig erachtet habe, z.B. im Zuge eines Hinweises für ein vertiefendes Nachlesen.

Ich möchte mich an dieser Stelle bei Fr. Ao.Univ.Prof. Mag.rer.nat. Dr.rer.nat. Gabriela Schranz-Kirlinger für ihre Unterstützung und ihre umfangreiche Betreuung meiner Diplomarbeit bedanken.

1.1 Gründe für die Wahl des Diplomarbeitsthemas

Für die Themenwahl meiner Diplomarbeit war für mich als zukünftige Mathematiklehrerin in erster Linie ein Bezug zur Schulmathematik wichtig. Das Newton-Verfahren war mir aus meinem Mathematik-Schulunterricht als Iterationsverfahren zur Nullstellenbestimmung der 1. Ableitung im Rahmen von Extremwertaufgaben nur als Formel im Sinne eines „Kochrezeptes“ bekannt. In zwei Lehrveranstaltungen im Laufe meines Studiums wurde mir das Newton-Verfahren als Tangenten-Verfahren sehr anschaulich vorgestellt. Einerseits im Zuge der **Lehrveranstaltung „Anwendungen der Mathematik für LehramtskandidatInnen: Modelle – Verfahren - Beispiele“** unter der Leitung von Hrn. Prof. Dorninger, in der vielfältige und anspruchsvolle Beispiele der angewandten, also praxisnahen Mathematik für den Schulunterricht vorgestellt wurden. Andererseits innerhalb der **Lehrveranstaltung „Numerische Mathematik für LehramtskandidatInnen“** unter der Leitung von Fr. Prof. Schranz-Kirlinger, in der die theoretischen Grundlagen hinsichtlich des Newton-Verfahrens verständlich aufbereitet wurden.

Für die oben erwähnte Lehrveranstaltung „Numerische Mathematik für LehramtskandidatInnen“ soll die hier vorliegende Arbeit betreffend „Iterations-verfahren für lineare und nichtlineare Gleichungssysteme“ eine theoretische und praktische Vertiefung mit dem Fokus auf Anschauung mittels computerunterstützten Beispielen in Maple darstellen. Die Schwerpunktsetzungen bzw. Kapitel dieser Diplomarbeit habe ich im Hinblick auf Vernetzungsmöglichkeiten mit der Schulmathematik gewählt. Im Folgenden werde ich nun etwas weiter ausholen, um die mir wesentlich erscheinenden Vernetzungen deutlich machen zu können.

Laut dem AHS-**Lehrplan** der Unter- und Oberstufe in Mathematik (siehe [LPU] und [LPO]) werden innerhalb des Stoffbereichs „Arbeiten mit Variablen“ Verfahren zum Lösen von *linearen Gleichungssystemen* in Verbindung mit geometrischer Interpretation behandelt. Im Falle von zwei Gleichungen mit zwei (bzw. drei) Variablen wird die Lagebeziehung von zwei Geraden (bzw. Ebenen) betrachtet, im Falle von drei Gleichungen mit drei Variablen wird die Lagebeziehung von drei Ebenen betrachtet. In diversen Mathematikschulbüchern ([REICHEL_5, S. 210, 224], [REICHEL_6, S. 113-116], [Malle_6, S. 42-43] und [Malle_O2, S. 244-247]) wird das

Gaußsche Eliminationsverfahren als ein zentrales Verfahren zum Lösen von linearen Gleichungssystemen mit m Gleichungen und n Variablen (Unbekannten) vorgestellt. Auch der Begriff Matrix (Koeffizientenmatrix) wird im Zuge dessen erwähnt (vgl. [REICHEL_6, S. 113]).

Jedes lösbare lineare Gleichungssystem kann mit Hilfe des Gaußschen Eliminationsverfahrens gelöst werden. Bei *sehr vielen* Gleichungen mit *sehr vielen* Variablen wachsen der Rechenaufwand und die numerischen Probleme allerdings auf ein unerträgliches Maß an[1]. In solchen Fällen verwendet man in der Praxis daher oft *Näherungsverfahren zum Lösen linearer Gleichungssysteme*. Eines davon ist das sogenannte *Gauß-Seidel-Verfahren*, das in manchen Schulbüchern (vgl. [REICHEL_6, S. 116]) thematisiert wird (eine detaillierte Ausarbeitung dazu wird in Kapitel 3.2 vorgestellt) – ebenso wie die Themen Matrix, Fixpunkt und Vektorraum (vgl. [REICHEL_6, S. 144-161]), welche im Rahmen meines Diplomarbeitsthemas eine grundlegende Rolle spielen.

Auch *nichtlineare Gleichungen* werden im Rahmen des Mathematikschulunterrichts behandelt, beispielsweise in Form von quadratischen Gleichungen oder allgemein bei der Berechnung von algebraischen Gleichungen höheren Grades.

Im Schulbuch [REICHEL_8, S. 190, 176-178] wird erläutert, dass man die Lösungsmenge eines Systems von quadratischen Gleichungen mit zwei (oder drei) Variablen mittels Computerunterstützung geometrisch durch die Durchschnittsmenge der entsprechenden Kegelschnittlinien (oder entsprechenden Flächen 2. Grades, wozu Ellipsoid (insbesondere die Kugel), Paraboloid und Hyperboloid zählen) erhält.

Die einfachste Form einer *quadratischen Gleichung* ist $x^2 = a$. Die Berechnung der Quadratwurzel liefert die Lösung $(x = \pm\sqrt{a})$. Im Schulbuch [NOVAK_O2, S. 109-110] wird ein Verfahren zur Berechnung einer Näherung der Quadratwurzel einer Zahl angegeben: $x_{k+1} = \frac{1}{2}(x_k + \frac{a}{x_k})$, wobei a für jene Zahl steht, deren Quadratwurzel bestimmt werden soll. Hierbei handelt es sich um das *Heron-Verfahren* oder *babylonische Wurzelziehen* (vgl. [WIKI]). In Kapitel 4.6.1 wird näher auf das Heron-

[1] Eine detaillierte Ausarbeitung dazu ist in [FETZ, S. 327-333] nachzulesen.

Verfahren eingegangen (als Spezialfall des Newton-Verfahrens und mit geometrischer Veranschaulichung). Meiner Meinung nach können SchülerInnen an diesem Beispiel sehr gut erkennen, was numerisches Lösen bedeutet: Es wird nicht nur stur das Wurzelzeichen in den Taschenrechner getippt, sondern das numerische Verfahren bzw. Iterationsverfahren, das dahinter steckt, kennengelernt, wobei anschauliche Darstellungen zum besseren Verständnis dienen sollen.

Nicht nur für Wurzelberechnungen benötigt man Näherungsverfahren. Viele Fragestellungen in der Praxis führen auf Gleichungen höheren Grades (vgl. Kapitel 2.2.3). Bei der näherungsweisen Berechnung von Lösungen von Gleichungen höheren Grades werden im Schulbuch [NOVAK_O2, S. 110-113] und [NOVAK_O3, S. 108-118] diverse Verfahren vorgestellt: Das *Regula-falsi-Verfahren*, das *Newton-Verfahren* und das *Bisektionsverfahren*. Für diese Näherungsverfahren sollen die SchülerInnen ein Programm erstellen mit einem Schleifen(-Wiederholungs-)zähler. Mit gleichen Funktionsgleichungen und gleicher Genauigkeit soll getestet werden, welches Verfahren weniger Wiederholungen benötigt, also schneller „arbeitet“ bzw. schneller konvergiert.

Weiters ist noch zu erwähnen, dass bez. des im Lehrplan verankerten Stoffbereichs „Differentialrechnung“ u. a. das Lösen von Extremwertaufgaben im Vordergrund steht. Der Fokus liegt auf der Nullstellenberechnung der 1. Ableitung einer Funktion, womit Iterationsverfahren wie das Newton-Verfahren oder das Sekanten-Verfahren in den Blickpunkt des Interesses kommen.

Diese soeben beschriebene Problemstellung - also ein Vergleich diverser numerischen Verfahren, die im Schulunterricht thematisiert werden (können) - spiegelt den Grundgedanken meiner Diplomarbeit wider. Ich verzichte auf eine strikt fachdidaktische Aufbereitung meiner behandelten Themen. Die in meiner Arbeit aufgegriffenen Themen sind als anregende Ergänzung zu diversen Schulbüchern gedacht und zwar in folgendem Sinne: Sie sollen im Rahmen eines **projekt-orientierten Mathematikunterrichts** oder im Mathematik-Wahlpflichtfach in der Oberstufe (7. oder 8. Klasse) als Grundlage dienen für

- eine theoretische Vertiefung hinsichtlich Iterationsverfahren für nichtlineare (siehe Kapitel 4) und lineare (siehe Kapitel 3) Gleichungssysteme
- mit entsprechender Veranschaulichung durch
- computerunterstützte Beispiele mittels dem Computeralgebrasystems **Maple**.

„Lernen mit **technologischer Unterstützung**“ ist einer der im Lehrplan verankerten didaktischen Grundsätze. Nicht nur der Taschenrechner, sondern verschiedene Technologien (wie Computeralgebrasysteme (CAS), dynamische Geometriesoftware oder Tabellenkalkulationsprogramme) sollen im Mathematikunterricht zum Einsatz kommen, da sie eine wesentliche Rolle beim Erarbeiten und Anwenden gewisser Inhalte spielen können. Meines Erachtens bietet diese hier vorliegende Arbeit Ansätze für einen sinnvollen Technologieeinsatz mittels dem CAS Maple hinsichtlich der Thematik „Iterationsverfahren linearer und nichtlinearer Gleichungssysteme“ im Rahmen eines projektorientierten Mathematikunterrichts.

Zusammenfassend haben also folgende Aspekte **meine Wahl zu diesem Diplomarbeitsthema begründet**: Einerseits haben mich die beiden oben erwähnten Lehrveranstaltungen zu einer theoretische Vertiefung inspiriert. Andererseits wollte ich den Lehrplan für Mathematik verknüpft mit der Forderung nach projektorientiertem Unterricht am Beispiel eines Technologieeinsatzes (mittels Maple) berücksichtigen. Auf dieser Basis habe ich die hier vorliegende Arbeit verfasst. Ich möchte nochmals betonen, dass ich auf eine fachdidaktische Aufbereitung (mit Tafelbild, Arbeitszettel und Übungsaufgaben) verzichtet habe. Diese Arbeit soll für einen projektorientierten Mathematikunterricht vorrangig grundlegende Anregungen bieten - vor allem im Hinblick auf die Visualisierung.

Abschließend möchte ich noch darauf hinweisen, dass in den Kapiteln 2.1.3 und 2.2.3 auf mögliche Anwendungen kurz hingewiesen wird. Denn eine wichtige Aufgabe einer Lehrperson ist in meinen Augen, auch den SchülerInnen den Sinn und Nutzen der Mathematik im Alltag zu verdeutlichen. Das gerät in den Klassenzimmern leider häufig in den Hintergrund, was Mathematik als langweilige „Kochrezept-sammlung“ den Kindern isoliert von ihrer Lebenswelt erscheinen lässt.

1.2 Kurzer Überblick über die einzelnen Kapitel

Kapitel 2 befasst sich mit jenen Grundlagen, die für den weiteren Aufbau dieser Arbeit von Bedeutung sind. Ergänzend zu der Lehrveranstaltung „Numerische Mathematik für LehramtskandidatInnen“ bei Fr. Prof. Schranz-Kirlinger, insbesondere dem Kapitel 2 „Lineare Gleichungssysteme“ und Kapitel 3 „Nichtlineare Gleichungssysteme“, werden grundlegende Definitionen wiederholt, Verweise gegeben und neue Begrifflichkeiten behandelt.

Die theoretische Aufbereitung, Programmierung in Maple (der Input ist in roter, der Output in blauer Farbe ersichtlich) und ein Vergleich von diversen Iterationsverfahren bilden das Herzstück dieser Arbeit.

Zwei prominente Verfahren betreffend den linearen Fall von Gleichungssystemen werden in **Kapitel 3** behandelt: Gauß-Seidel- und Jacobi-Verfahren.

Vier prominente Verfahren betreffend den nichtlinearen Fall von Gleichungssystemen werden in **Kapitel 4** behandelt: Bisektionsverfahren, Regula-falsi-Verfahren, Sekantenverfahren und Newton-Verfahren. Das Newton-Verfahren sowie einige Modifikationen werden in IR bzw. IR^n betrachtet.

2 Grundlagen

In diesem Kapitel erfolgt die Definition grundlegender Begriffe. Es wird auf das Vorlesungsskriptum zur „Numerische Mathematik für LehramtskandidatInnen“ (siehe [SCHRA]) verwiesen, insbesondere auf zwei darin vorkommenden Kapitel, nämlich Kapitel 2 „Lineare Gleichungssysteme“ (u. a. mit den Themen *Grundtatsachen aus der linearen Algebra, Lösungstheorie*, *Konditionsabschätzungen* und *Gaußelimination*) und Kapitel 3 „Nichtlineare Gleichungssysteme“ (u. a. mit Themen wie *diverse Fixpunktsätze* und *Newton-Verfahren*).

Da die hier vorliegende Arbeit eine Vertiefung hinsichtlich der beiden soeben genannten Kapitel darstellt, lehnt sich die im Folgenden verwendete Notation stark daran an. Einige Grundlagen werden wiederholt bzw. in einer etwas veränderten Notation dargestellt (z. B. bei der Vektordarstellung wird die Pfeil-Symbolik $\vec{x}$ anstelle von x verwendet). Weiters werden auch neue Begrifflichkeiten erläutert, die essentiell für den weiteren Aufbau dieser Arbeit sind.

Anmerkend sei erwähnt, dass in [WIKI] und [NEUN, S. 38-39] eine detaillierte *Liste numerischer Verfahren* nachgelesen werden kann und die in dieser Arbeit behandelten Verfahren dort ihre entsprechende Einordnung finden.

2.1 Lineare Gleichungssysteme

2.1.1 Allgemeines

In Anlehnung an [ŞAN, S. 72-73] und [BAR, S. 214] ist eine *lineare Gleichung* mit n Unbekannten folgendermaßen definiert:

$$c_1 \cdot x_1 + c_2 \cdot x_2 + ... + c_n \cdot x_n = b$$

$c_1, c_2, ..., c_n \in IR$ sind die skalaren *Koeffizienten* der Gleichung, $b \in IR$ ist ebenfalls ein Skalar (ein Skalar ist entweder eine Zahl oder ein Symbol, dem ein skalarer Wert zugeordnet ist, z. B. λ). Die Variablen $x_1, x_2, ..., x_n$ sind die *Unbekannten*.

Ein *lineares m x n - Gleichungssystem* besteht aus m linearen Gleichungen mit n Unbekannten:

$$a_{11} \cdot x_1 + a_{12} \cdot x_2 + ... + a_{1n} \cdot x_n = b_1$$

$$a_{21} \cdot x_1 + a_{22} \cdot x_2 + ... + a_{2n} \cdot x_n = b_2$$

$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$

$$a_{m1} \cdot x_1 + a_{m2} \cdot x_2 + ... + a_{mn} \cdot x_n = b_m$$

oder in Kurzschreibweise mit Matrizen und Vektoren unter der Benutzung der Matrix-Vektor-Multiplikation:

$$A \cdot \vec{x} = \vec{b}$$

Die Zahlen $a_{11}, a_{12}, ..., a_{mn}$ heißen *Koeffizienten* des Gleichungssystems, $x_1, x_2, ..., x_n$ sind die *Unbekannten*. Die Zahlen $b_1, b_2, ..., b_m$ bilden die *rechte Seite* oder *Inhomogenität* des Gleichungssystems. Die Bestimmung aller unbekannten Variablen wird als *Lösen* des Gleichungssystems bezeichnet.

$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \ldots & a_{mn} \end{pmatrix} \in IR^{m \times n}$ heißt *Koeffizientenmatrix bzw. Systemmatrix*.

$\vec{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \in IR^n$ bezeichnet den Vektor der Unbekannten.

Hier sei angemerkt, dass es sich bei $\vec{x}$ um eine n x 1 - Matrix (bzw. ein n-Tupel reeller Zahlen) handelt und man daher auch auf die Pfeilsymbolik hätte verzichten können. Trotzdem wird die Pfeilsymbolik gewählt, da sie in den Mathematikschulbüchern gebräuchlich ist.

$\vec{b} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix} \in IR^m$ bezeichnet den Vektor der rechten Seite.

$(A \mid \vec{b}) = \left(\begin{array}{ccc|c} a_{11} & \cdots & a_{1n} & b_1 \\ \vdots & \ddots & \vdots & \vdots \\ a_{m1} & \cdots & a_{mn} & b_m \end{array}\right)$ bezeichnet die *erweiterte Koeffizientenmatrix*.

$\vec{x}$ heißt *Lösungsvektor*, wenn seine Komponenten jede Gleichung des Systems erfüllen.

Es gibt verschiedene *Arten* von linearen Gleichungssystemen (vgl. [BAR, S. 214-216]):

- *Homogenes* lineares Gleichungssystem: $A \cdot \vec{x} = \vec{0}$, alle $b_i = 0$
- *Inhomogenes* lineares Gleichungssystem: $A \cdot \vec{x} = \vec{b}$, nicht alle $b_i = 0$
- *Quadratisches* (n x n -) lineares Gleichungssystem: m = n
- *Überbestimmtes* lineares Gleichungssystem: m > n
- *Unterbestimmtes* lineares Gleichungssystem: m < n

2.1.2 Lösbarkeit linearer Gleichungssysteme

Die allgemeine Lösungsbedingung für lineare Gleichungssysteme lautet gemäß [BAR, S. 214]:

$$r(A) = r(A \mid \vec{b}) = r \,.$$

$r(A)$ bezeichnet hierbei den Rang der Matrix A. Er ist als Zeilenrang bzw. Spaltenrang definiert, also als die Anzahl der linear unabhängigen Zeilen bzw. Spalten von A (vgl. [SCHRA, S. 39]).

Bezüglich der Lösbarkeit linearer Gleichungssysteme gibt es drei Möglichkeiten:

- *genau eine Lösung/eindeutig lösbar*,
- *unendlich viele Lösungen/nicht eindeutig lösbar* oder
- *keine Lösung/nicht lösbar*.

Eine übersichtliche Darstellung dieser Fälle findet sich in [BAR, S. 214-216].

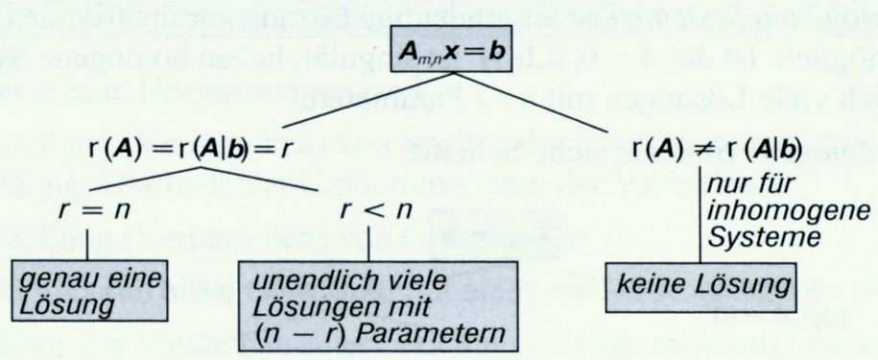


Abbildung 1: Lösbarkeit linearer Gleichungssysteme [BAR, S. 215]

In [KALT, S. 31] wird erläutert, dass jede der m Gleichungen eine Beziehung zwischen den n Unbekannten darstellt. Bei zu wenig vorhandener Information in einem linearen Gleichungssystem $A \cdot \vec{x} = \vec{b}$ sind nicht alle Unbekannten bestimmbar (d. h. es gibt keine eindeutige Lösung), man spricht von einem *unterbestimmten* linearen Gleichungssystem. Bei *überbestimmten* linearen Gleichungssystemen gibt es mehr Gleichungen als Unbekannte.

In Anlehnung an [WIKI] wird folgende Kurzzusammenfassung gegeben: Die Form der Lösungsmenge lässt sich grundsätzlich mit Hilfe der erweiterten Koeffizientenmatrix $(A|\vec{b})$ bestimmen, indem diese mit Hilfe der elementaren Zeilenumformungen auf eine Block-Dreiecksform - d. h. soweit wie möglich auf eine Dreiecksform - gebracht wird:

$$(A \mid \vec{b}) = \left(\begin{array}{ccc:c:c} c_{11} & \cdots & c_{1r} & \cdots & d_1 \\ 0 & \ddots & \vdots & \cdots & \vdots \\ 0 & 0 & c_{rr} & \cdots & d_r \\ \hdashline 0 & \cdots & \cdots & 0 & d_{r+1} \\ \vdots & & & \vdots & \vdots \\ 0 & \cdots & \cdots & 0 & d_m \end{array}\right)$$

- Ist mindestens ein d_i mit $i \geq r+1$ von Null verschieden, so gibt es *keine Lösungen*.
- Ist $d_{r+1} = d_{r+2} = ... = d_m = 0$ und $r = n$, so ist das Gleichungssystem *eindeutig lösbar*.
- Ist $d_{r+1} = d_{r+2} = ... = d_m = 0$ und $r < n$, so gibt es *unendlich viele Lösungen*.

Es gibt 2 Arten von *Lösungsverfahren für lineare Gleichungssysteme* (vgl. [BAR, S. 217-225]):

1.) ***Direkte*** *Lösungsverfahren* liefern die exakte Lösung, sofern diese existiert, abgesehen von Rundungsfehlern in endlich vielen Schritten.

Dies erfolgt durch schrittweises Transformieren von $A \cdot \vec{x} = \vec{b}$ in ein einfacher zu lösendes Problem $R \cdot \vec{x} = \vec{c}$. Hierbei enthält R eine reguläre obere Block-Dreiecksmatrix[2], wobei alle Zwischenschritte die gleiche Lösungsmenge wie das Ausgangsproblem haben.

Wesentlich dabei ist, dass äquivalente, ranginvariante Umformungen die Lösungsmenge eines linearen Gleichungssystems und den Rang ihrer Koeffizientenmatrix nicht verändern.

Das Gaußsche Eliminationsverfahren ist das prominenteste Beispiel eines direkten Lösungsverfahrens.

2.) ***Iterative*** *Lösungsverfahren* verbessern den Startvektor $\vec{x}^{(0)}$ der Lösung schrittweise: $\vec{x}^{(0)} \rightarrow \vec{x}^{(1)} \rightarrow ... \rightarrow \vec{x}^{(n)} \rightarrow ...$

$\vec{x}^{(n)}$ konvergiert für n → ∞ gegen die gesuchte Lösung.

[2] *Regulär* bedeutet invertierbar, d. h. es existiert die zu R inverse Matrix R^{-1}; obere Dreiecksmatrix bedeutet, dass alle Koeffizienten r_{ii} der Hauptdiagonale von 0 verschieden sind und unterhalb der Hauptdiagonale nur Nullen stehen.

Iterative Lösungsverfahren sind beispielsweise das *Gauß-Seidel-* und *Jacobi-Verfahren,* die zur Klasse der Splitting-Verfahren (siehe Kapitel 3.1.1) gehören.

Iterative Methoden liefern niemals eine exakte Lösung, sondern immer nur eine Näherung der Lösung, da es sich um einen unendlichen Prozess handelt, der bei gewünschtem Genauigkeitsniveau beendet wird (in Anlehnung an [KALT, S. 51]).

Gemäß [HUCK, S. 74] stellt sich bei den direkten Lösungsverfahren die Frage nach den benötigten Rechenoperationen und der Rechengenauigkeit, während bei den iterativen Lösungsverfahren interessiert, wie hoch die Konvergenzgeschwindigkeit und wie aufwändig ein Iterationsschritt ist.

Während man bei nichtlinearen Gleichungen oder Gleichungssystemen in der Regel keine Alternative zu einem Iterationsverfahren hat, stehen diese bei linearen Gleichungssystemen in Konkurrenz mit dem Eliminationsverfahren. Dabei hat es sich gezeigt, dass bei gewissen, in den Anwendungen sehr häufig vorkommenden linearen Gleichungssystemen die Iterationsverfahren den Eliminationsverfahren überlegen sein können, wenn es darum geht, eine Lösung mit einer vorgeschriebenen Genauigkeit bei geringstem Rechenaufwand zu berechnen. Dazu kommen Vorteile hinsichtlich des Speicherbedarfs, die in der Vergangenheit eine große Rolle spielten, als schnelle Arbeitsspeicher teuer und deshalb klein waren. Allerdings schwankt die Einschätzung der Verfahren mit der verwendeten Hardware. Wie so oft liegt die Wahrheit in der Mitte: Bei einer bestimmten Problemklasse, die für physikalische Anwendungen (z.B. auch Wettervorhersage) eine zentrale Rolle spielt, sind die (derzeit) schnellsten Methoden sogenannte *Hybrid-Verfahren*, also Misch-Verfahren, die eine Mischung von Iterations- und Eliminationsverfahren darstellen, wie [LOCH, S. 302] betont.

2.1.3 Anwendungen

Nun erfolgt ein kurzer Exkurs hinsichtlich einiger möglicher Anwendungen von linearen Gleichungssystemen, die im Alltag eine bedeutende Rolle spielen und somit an die Lebenswelt der SchülerInnen anknüpfen (wie in Abschnitt 1.1 gefordert). Für

eine weitere Beschäftigung mit den nun folgenden Themen wird auf die angegebenen Literaturquellen verwiesen.

(1) Bei [KALT] wurde unter dem Abschnitt II „Moderne Verfahren zur numerischen Lösung linearer Gleichungssysteme“ das Kapitel 3 *Eine Anwendung: Die Computertomographie (CT)* verfasst.

(2) Im Bereich der Elektrotechnik, der im Zusammenhang mit der *Entwicklung von Mikrochips* auch für die Informatik wichtig ist, wählt [HUCK, S.74-76] ein Beispiel zur *Berechnung der Stromstärkenverteilung* eines Stromkreises mit Ohmschen Widerständen in Reihen- und Parallelschaltung.

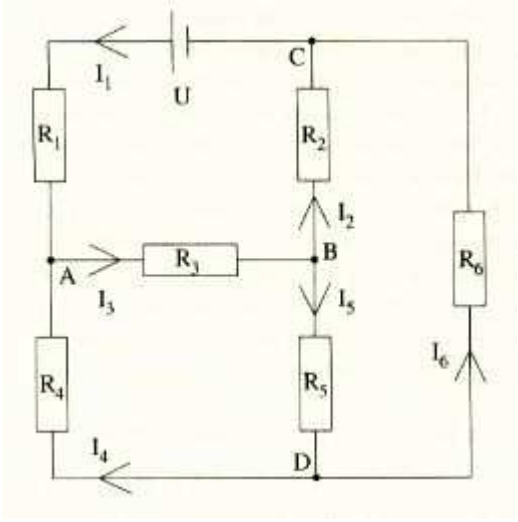


Abbildung 2: Stromkreis mit Ohmschen Widerständen in Reihen- und Parallelschaltung [HUCK, S.74]

Uns interessieren die Stromstärken $I_1, I_2, ..., I_6$, wenn an den Stromkreis mit den Widerständen $R_1, R_2, ..., R_6$ die Spannung U angelegt wird. Um die Stromstärkenverteilung berechnen zu können, stellt man zunächst ein lineares Gleichungssystem auf. Dazu verwendet man einerseits das Ohmsche Gesetz $U = R \cdot I$ und andererseits die zwei Kirchoffschen Regeln.

(3) Gemäß [MUNZ, S. 377] treten bei Näherungsverfahren für *partielle Differentialgleichungen* oft große lineare Gleichungssysteme (LGS) auf, welche zur Ermittlung der Werte der Näherungslösung gelöst werden müssen. Diese LGS bzw. deren Koeffizientenmatrizen nennt man schwachbesetzt oder dünnbesetzt, da nur sehr wenige Elemente der jeweiligen Matrix von Null verschieden sind. Schwachbesetzte Matrizen treten beispielsweise bei Finite-Elemente-Verfahren (vgl. das folgende Beispiel (4)) auf. Die effiziente Lösung von schwachbesetzten LGS ist eine wichtige Grundaufgabe bei der numerischen Lösung von partiellen Differentialgleichungen.

Iterative Methoden haben bei schwach besetzten LGS wesentliche Vorteile. Die Matrix des LGS mit den vielen Nullen muss nicht abgespeichert werden, es werden nur die Elemente ungleich Null benötigt.

Am Beispiel einer elliptischen partiellen Differentialgleichung, die die *Verbiegung einer Membran* beschreibt (nachzulesen in [MUNZ, S. 245]), erfolgt ein Vergleich zweier Iterationsverfahren (Jacobi-Verfahren versus Gauß-Seidel-Verfahren) (nachzulesen in [MUNZ, S. 285]). Für die Umsetzung dieses Beispieles wurde das Programm Maple gewählt.

(4) Die *Finite-Elemente-Methode*, kurz FEM, oder auch „Methode der finiten Elemente“, ist ein numerisches Verfahren zur näherungsweisen Lösung, insbesondere elliptischer partieller Differentialgleichungen mit Randbedingungen. Sie ist auch ein weit verbreitetes modernes Berechnungsverfahren im Ingenieurwesen (vgl. [WIKI]).

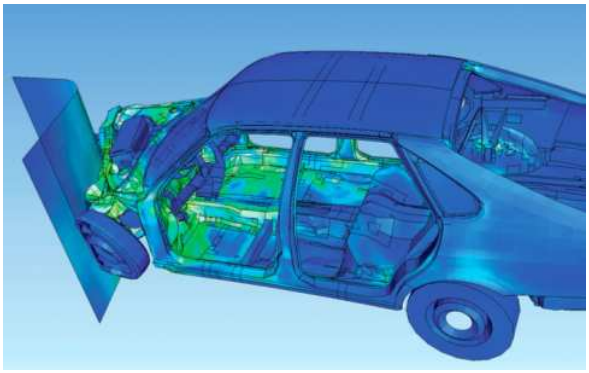


Abbildung 3: Visualisierung einer FEM-Simulation der Verformung eines Autos bei asymmetrischem Frontalaufprall [WIKI]

In [ŞAN, S. 604] findet sich ein aus einer FEM-Modellierung stammendes lineares Gleichungssystem, welches mit dem Gauß-Seidel-Verfahren in Maple gelöst wird.

2.2 Nichtlineare Gleichungssysteme

2.2.1 Allgemeines

Der Begriff *linear* im Bezug auf ein Gleichungssystem bedeutet, dass jede Gleichung des Systems linear ist, d. h. die Unbekannten nur linear vorkommen (vgl. die Definition in Kapitel 2.1.1).
Sobald eine der Gleichungen nichtlinear ist, spricht man von einem *nichtlinearen* Gleichungssystem. Eine nichtlineare Gleichung erkennt man daran, dass sie eben *nicht* linear ist und somit in der Gleichung ein Term der Form $x \cdot y$, x^2, $\sin x$, $\ln x$ oder dergleichen auftritt (in Anlehnung an [KALT, S.10-11]).

Die Bedeutung *linear* im Zusammenhang mit einer Gleichung (bzw. mit einem Gleichungssystem) ist allerdings nicht zu verwechseln mit der Bedeutung linear im Zusammenhang mit einer Funktion bzw. Abbildung. Im letzteren Fall bedeutet Linearität nämlich Folgendes:

Definition: (In Anlehnung an [SCHRA, S. 35])

f ist ***lineare Abbildung*** : $\Leftrightarrow$

$f : IR^n \to IR^m, \forall \vec{x}, \vec{y} \in IR^n, \lambda \in IR$

1.) $f(\vec{x} + \vec{y}) = f(\vec{x}) + f(\vec{y})$

2.) $f(\lambda \cdot \vec{x}) = \lambda \cdot f(\vec{x})$

Wesentlich ist, dass sich *jede* lineare Abbildung $f : IR^n \to IR^m$ als $f(\vec{x}) = A \cdot \vec{x}$ schreiben lässt und umgekehrt *jede* Matrix A eine lineare Abbildung darstellt (gemäß [SCHRANZ, S. 36-37]). A steht in diesem Kontext für die Koeffizientenmatrix.

Die Gleichung $\lambda \cdot x + \mu = \nu$ mit $\nu = f(x)$ ist zwar eine lineare Gleichung, aber das zugehörige $f : IR^n \to IR^m$ mit n = m = 1 ist keine lineare Abbildung, da $f(x + y) = \lambda \cdot (x + y) + \mu \neq \lambda \cdot x + \mu + \lambda \cdot y + \mu = f(x) + f(y)$. Damit ist der erste Definitionspunkt verletzt. Somit gibt es keine (einer linearen Abbildung

entsprechenden) Koeffizientenmatrix A und demnach auch kein lineares Gleichungssystem $A \cdot \vec{x} = \vec{b}$ mit $\vec{b} = f(\vec{x})$.
Unser Beispiel $\lambda \cdot x + \mu = \nu$ mit $\nu = f(x)$ und $f : IR \to IR$ für *eine einzige* nichtlineare Gleichung beschreibt den **skalaren bzw. eindimensionalen Fall** eines nichtlinearen Gleichungssystems. Beim **vektoriellen bzw. mehrdimensionalen Fall** eines nichtlinearen Gleichungssystems betrachtet man Funktionen der Form $f : IR^n \to IR^m$, wobei m = n > 1 gilt.

Bei der Gleichung $f(x) = \lambda \cdot x + \mu$ mit $f : IR \to IR$ handelt es sich um einen Spezialfall, besser gesagt um den linearen Fall, im Rahmen von nichtlinearen Gleichungssystemen. Die Lösbarkeit (entspricht dem Finden einer Nullstelle, vgl. Abschnitt 2.2.2) richtet sich nach den Skalaren λ und μ:

- eindeutige Lösung bzw. regulärer Fall $(\lambda \neq 0)$,
- keine Lösung $(\lambda = 0, \mu \neq 0)$ oder
- unendlich viele Lösungen $(\lambda = 0, \mu = 0)$ (vgl. [SCHRANZ, S. 68]).

2.2.2 Lösbarkeit nichtlinearer Gleichungssysteme

Laut [ŞAN, S. 579] ist das Lösen einer nichtlinearen Gleichung $f(x) = y$ gleichbedeutend damit, die Nullstelle r zu finden, wo $f(r) = 0$ ist. Wird die Funktionskurve $y = f(x)$ im x-y-Koordinatensystem wie in Abbildung 4 graphisch dargestellt, entspricht die Nullstelle dem Schnittpunkt zwischen der Kurve und der x-Achse. In einfachen Fällen lässt sich eine nichtlineare Gleichung auch mit Hilfe von Formelsammlungen (z.B. quadratische oder kubische Polynome) oder auf Erfahrungsgrundlage lösen. Beispielsweise weiß man i. d. R. auch ohne lange Rechnung, dass $x = 1$ die Lösung bzw. Nullstelle von $f(x) = \ln x$ ist. Im Allgemeinen wird es aber nicht möglich sein, eine nichtlineare Gleichung so einfach zu lösen. In diesen Fällen ist der Einsatz numerischer Lösungsverfahren sinnvoll bzw. notwendig.

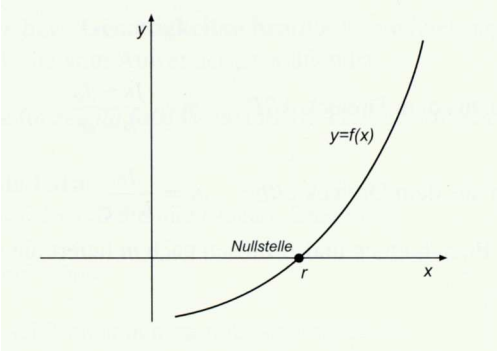


Abbildung 4: Nullstelle einer nichtlinearen Funktion [ŞAN, S. 579]

2.2.3 Anwendungen

Nun erfolgt ein kurzer Exkurs hinsichtlich einiger möglicher Anwendungen von nichtlinearen Gleichungssystemen, die im Alltag eine bedeutende Rolle spielen und somit an die Lebenswelt der SchülerInnen anknüpfen (wie in Abschnitt 1.1 gefordert).

(1) Räuber-Beute-Modell
(2) Grippeausbreitung im Kindergarten

Die Beispiele (1) und (2) werden im Kapitel 2.3.3 im Rahmen von Anwendungen der Fixpunktiteration im *nichtlinearen Fall* detailliert vorgestellt.

2.3 Iterationsverfahren: Fixpunktiteration

2.3.1 Allgemeines

In Anlehnung an [WIKI], [SCHRA, S.72-81], [PLATO, S.81, 84-84] und [DORN, S. 11-15] werden im Folgenden die Themen

- Fixpunkt,
- Fixpunktsätze,
- Fixpunktiteration sowie
- Konvergenz der Ordnung p

näher betrachtet.

In der Geometrie versteht man unter einem **Fixpunkt** einen Punkt, der durch eine gegebene Abbildung auf sich abgebildet wird.[3]

Allgemein wird als Fixpunkt in der Mathematik ein Punkt x bezeichnet, der die Gleichung $f(x) = x$ erfüllt. Anschaulich bedeutet dies, dass man die Funktion auf den Punkt anwenden kann, ohne ihn zu verändern.

Ein Fixpunkt x^*, der die Gleichung $f(x) = x$ erfüllt, ist geometrisch im IR^1 der Schnittpunkt der Funktion $f(x)$ mit der 1. Mediane $y = x$.

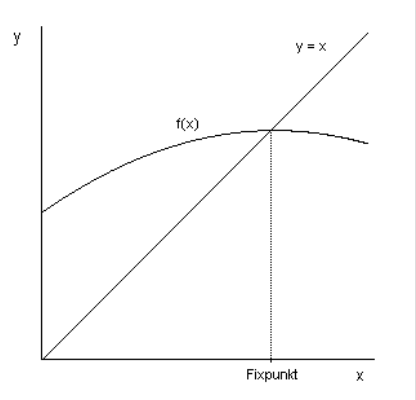


Abbildung 5: Darstellung eines Fixpunktes [WIKI]

[3] Die Fixpunkte einer Achsenspiegelung sind die Punkte der Spiegelachse. Eine Punktspiegelung hat nur einen Fixpunkt, nämlich deren Zentrum.

Jede lineare oder nichtlineare Gleichung $f(x) = y$ lässt sich in eine Fixpunktform $\varphi(x) = x$ umwandeln:

$$f(x) = y \text{ (insbesondere } f(x) = 0 \text{ (}\textit{Nullstellenproblem}\text{))}$$
$$\Leftrightarrow f(x) - y = 0$$
$$\Leftrightarrow \underbrace{f(x) - y + x}_{=:\varphi(x)} = x \text{ (}\textit{Fixpunktproblem}\text{).}$$

Die Existenz und Eindeutigkeit von Fixpunkten ist Gegenstand einiger wichtiger mathematischer Sätze, sogenannter **Fixpunktsätze** (z. B. von Banach, Brouwer oder Schauder).

Der ***Fixpunktsatz von Banach*** besagt, dass eine Kontraktion[4] eines vollständigen metrischen Raumes[5] in sich *genau einen* Fixpunkt besitzt (Details dazu siehe Kapitel 2.3.2).

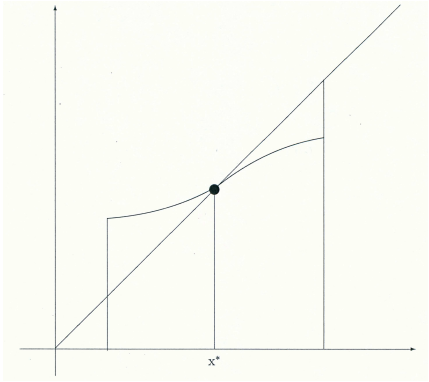


Abbildung 6:
Eindeutigkeit bei Kontraktion [SCHRA, S. 74]

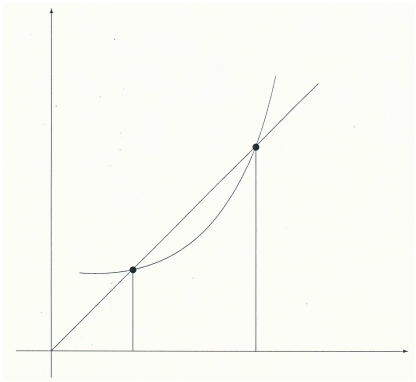


Abbildung 7:
Mehrdeutigkeit bei Selbstabbildung [SCHRA, S. 75]

Wenn statt einer Kontraktion nur eine stetige Selbstabbildung[6] betrachtet wird, muss der Fixpunkt *nicht eindeutig* sein und entsprechende Fixpunktsätze zeigen dann nur die *Existenz*. Der ***Fixpunktsatz von Schauder*** zeigt etwa die *Existenz (mindestens)*

[4] In Kapitel 2.3.1.1 ist die Definition einer *Kontraktion* nachzulesen.
[5] Ein *vollständiger metrischer Raum* ist ein Raum, in dem eine Abstandsfunktion (Metrik) existiert und jede Cauchy-Folge konvergiert.
[6] Eine Abbildung, die eine Menge in sich selbst abbildet, wird als *Selbstabbildung* bezeichnet. Anders formuliert (in Anlehnung an [LENZ, S. 27]): Urbilder und Bilder dieser Funktion liegen stets in ein und demselben Bereich.

eines Fixpunktes in einer kompakten, konvexen Teilmenge eines Banachraums[7]. Dieser Satz ist eine Verallgemeinerung des ***Fixpunktsatzes von Brouwer***, der besagt, dass jede stetige Abbildung einer abgeschlossenen und konvexen Menge in sich selbst (mindestens) einen Fixpunkt besitzt, wobei die Bildmenge beschränkt ist. Im Gegensatz zu den beiden anderen Sätzen gilt der Fixpunktsatz von Brouwer allerdings nur in endlichdimensionalen Räumen, also z. B. im IR^n oder im $\mathbb{C}^n$.

In der numerischen Mathematik bezeichnet ***Iteration*** eine Methode, sich der exakten Lösung eines Rechenproblems schrittweise, aber zielgerichtet anzunähern (sukzessive Approximation). Sie besteht in der wiederholten Anwendung desselben Rechenverfahrens. Meistens iteriert man mit Rückkopplung: Die Ergebnisse eines Iterationsschrittes werden als Ausgangswerte des jeweils nächsten Schrittes genommen – bis das Ergebnis zufrieden stellt. Ein Beispiel dafür ist das Newton-Verfahren. Manchmal benutzt man für einen Schritt die Ergebnisse der letzten beiden Schritte, zum Beispiel beim Sekanten-Verfahren[8]. Es muss anschließend noch bewiesen werden, dass die Iterationsfolge konvergiert und dass der Grenzwert mit der gesuchten Lösung übereinstimmt. Die Geschwindigkeit der Konvergenz ist ein Maß dafür, wie effizient die Iterationsmethode ist.

Jedes **Fixpunktiterationsverfahren** hat die Form

$$x_{n+1} = \varphi(x_n) \text{ für n = 0, 1, 2, …} \qquad (2.1)$$

Mit jeder weiteren Iteration soll sich x_{n+1} der exakten Lösung x^* annähern. Das Ziel ist, die Iterationsvorschrift φ so zu konstruieren, dass sie genau einen Fixpunkt x^* besitzt, dass also schließlich gilt:

$$x^* = \varphi(x^*) .$$

[7] Ein *Banach-Raum*, benannt nach dem Mathematiker Stefan Banach, ist ein vollständiger normierter Vektorraum. Banach-Räume gehören zu den zentralen Studienobjekten der Funktionalanalysis. Die interessantesten Banach-Räume sind unendlich-dimensionale Funktionenräume.

[8] Für die Berechnung von x_{n+1} werden die beiden vorhergehenden Werte x_n und x_{n-1} benötigt (vgl. Kapitel 4.3)

Anzumerken ist, dass das Auffinden eines geeigneten Startwerts x_0 oft nicht leicht ist.

In [PLATO, S. 81] findet sich eine detaillierte Darstellung dieses Sachverhalts: Im Folgenden sei $f: IR^n \to IR^n$ eine gegebenen Funktion und $x^* \in IR^n$ eine Nullstelle von f,

$$f(x^*) = 0,$$

die es zu bestimmen gilt. Wenn f nichtlinear ist, lässt sich ein solches Gleichungssystem typischerweise nur approximativ lösen[9], was im Folgenden mittels Iterationsverfahren der Form (2.1) geschehen soll mit einer geeigneten stetigen Iterationsfunktion $\varphi: IR^n \to IR^n$. Dabei soll die Abbildung φ so beschaffen sein, dass Konvergenz im folgenden Sinne vorliegt:

Definition: (In Anlehnung an [PLATO, S. 81])

Sei $\varphi: IR^n \to IR^n$ eine Iterationsfunktion. Das Verfahren (2.1) zur Bestimmung von $x^* \in IR^n$ heißt (lokal) ***konvergent***, wenn ein $\delta > 0$ existiert, so dass für alle Startwerte $x_0 \in U(x^*; \delta) := \{y \in IR^n : \| y - x^* \| < \delta\}$ [10] gilt

$$\| x_n - x^* \| \to 0 \text{ für } n \to \infty. \qquad (2.2)$$

Hier bezeichnet $\| \cdot \|: IR^n \to IR^n$ eine nicht näher spezifizierte Vektornorm.

Da die Iterationsfunktion $\varphi: IR^n \to IR^n$ als stetig[11] in x^* vorausgesetzt ist, handelt es sich aufgrund der Konvergenz (2.2) bei $x^* \in IR^n$ notwendigerweise um einen Fixpunkt von φ,

$$\varphi(x^*) = x^*,$$

denn $x^* \overset{(2.2)}{=} \lim_{n\to\infty} x_{n+1} \overset{(2.1)}{=} \lim_{n\to\infty} \varphi(x_n) \overset{\varphi \text{ stetig}}{=} \varphi(\lim_{n\to\infty} x_n) \overset{(2.2)}{=} \varphi(x^*)$.

Daher bezeichnet man das Verfahren (2.1) als Fixpunktiteration.

---

[9] Für lineares f sind neben iterativen (= approximativen) auch direkte Lösungsverfahren möglich (vgl. Kapitel 2.1.2).

[10] $x_0 \in U(x^*; \delta)$ bedeutet, dass x_0 in einer Umgebung/offenen Kugel um x^* mit Radius δ liegt.

[11] Bei einer stetigen Funktion ist die Funktionsbildung mit der Limesbildung vertauschbar.

Die Konvergenz von Fixpunktiterationen wird etwa mittels des Fixpunktsatzes nach Banach untersucht. In vielen Fällen divergiert die Iteration, sodass kein Fixpunkt ermittelt werden kann. Dabei gibt es zwei Möglichkeiten:

- Die Funktion f hat tatsächlich keine Nullstelle oder
- die Wahl des Startwerts x_0, der Iterationsfunktion φ oder der Darstellung von $f(x) = 0$ führen zu Divergenz, obwohl eine Nullstelle existiert (Ein Beispiel dazu findet sich in Kapitel 2.3.1.1).

Zusätzlich zur Konvergenz ist wünschenswert, dass das Verfahren (2.1) eine möglichst hohe **Konvergenzordnung** im Sinne der folgenden Definition besitzt:

Definition: (In Anlehnung an [PLATO, S. 81])
Sei $\varphi : IR^n \to IR^n$ eine Iterationsfunktion mit Fixpunkt $x^* \in IR^n$. Das Verfahren (2.1) heißt (lokal) ***konvergent von (mindestens) der Ordnung*** $p \geq 1$, wenn ein $\delta > 0$ existiert, so dass für alle Startwerte $x_0 \in U(x^*; \delta)$ gilt

$$\| x_{n+1} - x^* \| \leq C \| x_n - x^* \|^p \text{ für n = 0, 1, 2, …}$$

mit einer Konstanten $0 \leq C = C(x_0) < \infty$, wobei im Fall $p = 1$ noch $C < 1$ gefordert wird. Bei Konvergenz der Ordnung p = 1 bzw. p = 2 spricht man von (mindestens) *linearer* bzw. *quadratischer* Konvergenz.

Das Verfahren (2.1) heißt ***konvergent von genau der Ordnung*** p, wenn es konvergent von der Ordnung p ist und keine höhere Konvergenzordnung besitzt.

Unter dem Begriff Konvergenzordnung (auch Konvergenzgeschwindigkeit) versteht man also die Geschwindigkeit, mit der sich die Glieder einer konvergenten Folge (x_n) dem Grenzwert x^* nähern. Dieser Begriff ist in der Numerik wichtig, wo eine Näherung des Grenzwertes eines Iterationsverfahrens i. A. durch Berechnung einer kleinen Anzahl von Iterationen geschieht. Konvergenz der Ordnung p bedeutet dann, dass in jedem Iterationsschritt die Anzahl der genauen Dezimalstellen ver-p-facht wird, also beispielsweise bei quadratischer Konvergenz verdoppelt.

Die schnellere Konvergenz von Verfahren höherer Ordnung wird meist mit größerem Aufwand pro Iteration bezahlt (vgl. gemäß [SCHRA, S. 20] die Faustformel: „Verfahren höherer Ordnung lohnen sich nur bei strengem Genauigkeitsniveau).

2.3.1.1 Ein einleitendes einfaches Beispiel

Nun wird ein **einleitendes einfaches Beispiel** erläutert, das sich für die Überleitung zum nächsten Abschnitt (Fixpunktsatz nach Banach) gut eignet und sich mit folgender Thematik befasst (in Anlehnung an [DORN, S. 11-15]): Ob das ***Iterationsverfahren konvergiert, hängt von Folgendem ab:***

1.) Startwert x_0
2.) Wahl der Iterationsfunktion φ
3.) Darstellung von $f(x) = 0$

Dass die Darstellung der Gleichung $f(x) = 0$ bzw. die Wahl der Iterationsfunktion φ Auswirkung auf die Konvergenz des Iterationsverfahrens haben kann, wird durch folgende Betrachtungen ersichtlich:

Ist $f(x) = x^2 - 3x + 2$, dann führen die folgenden drei äquivalenten (Fixpunkt-) Darstellungen (in der Form $\varphi(x) = x$) von $f(x) = 0$ (Nullstellendarstellung) zu ein und demselben Startwert $x_0 = -\frac{1}{2}$ zu völlig verschiedenen Ergebnissen:

a) $\varphi(x) = x = -x^2 + 4x - 2$

b) $\varphi(x) = x = \frac{x^2 + 2}{3}$

c) $\varphi(x) = x = 3 - \frac{2}{x}$ mit der Zusatzbedingung $x \neq 0$

Zuerst einmal lassen wir die Funktion $f(x) = x^2 - 3x + 2$ in Maple zeichnen, um uns einen ersten Überblick über die Nullstellen dieser Funktion zu verschaffen.

- **Pakete laden**

Wenn man mit einem neuen Maple-Worksheet zu arbeiten beginnt, ist es sinnvoll, den Befehl *restart* auszuführen. Damit wird Maple initialisiert, d. h. alle Bindungen

(bereits gemachte benutzerdefinierte Belegungen von Symbolen/Variablen) werden gelöscht und Maple verhält sich anschließend wie nach einem Neustart.

```
> restart:
```

Die benötigten Pakete laden: Das Paket *plots* bietet Graphikmöglichkeiten.

```
> with(plots):
```

➢ Erster Überblick über die Nullstellen

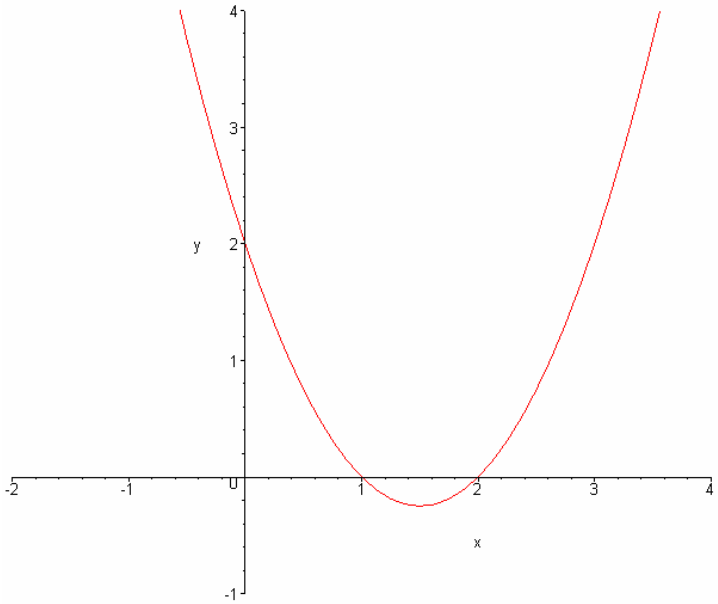
Wir suchen die Nullstellen von $f(x) = x^2 - 3x + 2$, d. h. die Lösungen der quadratischen Gleichung $f(x) = 0$:

```
> f(x):=x^2-3*x+2;
```

$$f(x) := x^2 - 3x + 2$$

Der Graph der Funktion $f(x)$ sieht so aus[12]:

```
> plot(f(x), x=-2..4, y=-1..4, scaling = constrained);
```



[12] *scaling* = *constrained* bewirkt ein unverzerrtes Bild (die Koordinatenachsen werden im selben Maßstab dargestellt)

Zur Kontrolle verwenden wir den *Solve*-Befehl, der die angegebene Gleichung $f(x)=0$ nach der Variable x löst und wir erhalten als Maple-Output die beiden Nullstellen 2 und 1.

```
> solve(f(x)=0,x);
```

$$2, 1$$

Mit $x_0 = -\frac{1}{2}$ wurde der Startwert wie üblich in der Nähe der vermuteten Nullstelle(n) festgelegt.

- **a)**

```
> x0:=-1/2:                    #[13] Der Startwert wird definiert
for i from 1 to 8 do           # i=1,2,...,8 (8 Näherungen sollen
                               berechnet werden)
  x1:=-x0^2+4*x0-2:
  printf("x%d=%g\n",i,x1):
  x0:=x1:
end do:

x1=-4.25
x2=-37.0625
x3=-1523.88
x4=-2.32830e+06
x5=-5.42101e+12
x6=-2.93874e+25
x7=-8.63617e+50
x8=-7.45834e+101
```

D. h. Das Iterationsverfahren divergiert.

- **b)**

```
> x0:=-1/2:
for i from 1 to 8 do
  x1:=1/3*(x0^2+2):
  printf("x%d=%g\n",i,x1):
  x0:=x1:
end do:

x1=0.75
x2=0.854167
x3=0.909867
```

[13] In der Maple-Syntax erlaubt das Symbol # das Einfügen von Kommentaren innerhalb des Programmiercodes.

```
x4=0.942619
x5=0.962844
x6=0.975689
x7=0.98399
x8=0.989412
```

D. h. Das Iterationsverfahren konvergiert gegen 1.

- **c)**

```
> x0:=-1/2:
for i from 1 to 8 do
  x1:=3-2/x0:
  printf("x%d=%g\n",i,x1):
  x0:=x1:
end do:
```

```
x1=7
x2=2.71429
x3=2.26316
x4=2.11628
x5=2.05494
x6=2.02674
x7=2.01319
x8=2.00655
```

D. h. Das Iterationsverfahren konvergiert gegen 2.

Darstellung a) führt nicht zum Erfolg, da sich Divergenz ergibt. Umformungen der Darstellung a) können jedoch zu Konvergenz führen, wie die Darstellung b) oder c) zeigt. Anmerkend sei festgehalten, dass bei Konvergenz nur eine Nullstelle ermittelt wird, die nicht die einzige sein muss (wie in unserem Fall).

Abschließend stellt sich nun die allgemeine Frage: Welche Iterationsfunktionen φ sind wünschenswert? Bei φ soll es sich um eine kontrahierende Abbildung handeln, denn dann existiert genau ein Fixpunkt und die Iterationsfunktion φ konvergiert wie gewünscht (vgl. Fixpunktsatz nach Banach im Kapitel 2.3.2).

Definition: (In Anlehnung an [DORN, S. 11] und [LENZ, S. 28-29])

Sei φ eine Iterationsfunktion $\varphi:[a;b]\subseteq IR\to IR$

(1) $x\in[a;b]\Rightarrow\varphi(x)\in[a;b]$ (d. h. φ ist *Selbstabbildung*)

(2) Es gibt ein $L\in IR$ mit $0\le L<1$[14] (*Lipschitzkonstante* oder *Kontraktionszahl*): $x_1,x_2\in[a;b]\ \Rightarrow\ \|\varphi(x_1)-\varphi(x_2)\|\le L\,\|x_1-x_2\|$ (*Lipschitzbedingung*)

Unter den Bedingungen (1) und (2) heißt φ ***kontrahierende Abbildung*** bzw. ***Kontraktion*** auf [a;b].

Diese hier angegebene Definition gilt ebenso für die Verallgemeinerung, wenn IR^n sowie $\vec{x}\in[\vec{a};\vec{b}]:=\{\vec{x}\in IR^n\mid a_i\le x_i\le b_i,1\le i\le n\}$ anstelle von IR und $x\in[a;b]$ betrachtet werden. (vgl. [LENZ, S. 75])

In Darstellung a) hat die Funktion φ durch das in ihr vorkommende x^2 keine Chance, kontrahierend zu sein – bei großen x-Werten „explodieren“ die Abstände der Bilder.

Wie in [LENZ, S. 29] betont wird, ist es oft nicht einfach zu überprüfen, ob eine gegebene Abbildung eine Kontraktion ist. Falls die Abbildung jedoch differenzierbar ist und eine stetige Ableitung besitzt (eine derartige Abbildung nennt man auch kurz eine *stetig differenzierbare Funktion*), kann man eine sehr einfach zu überprüfende hinreichende Kontraktionsbedingung angeben.

Satz (***Hinreicheichende Kontraktionsbedingung***): (In Anlehnung an [LENZ, S. 29])
$\varphi:[a;b]\subseteq IR\to IR$, φ ist eine stetig differenzierbare Abbildung auf $[a;b]$ und es gilt

$$\max\{|\varphi'(x)|\mid x\in[a;b]\}=:L<1$$

$\Rightarrow\varphi$ ist eine kontrahierende Abbildung auf $[a;b]$.

Beweis: Nachzulesen in [LENZ, S. 29-30]

[14] Dadurch ist der Abstand zweier Bilder der Funktion *echt* kleiner als der Abstand der zugehörigen Urbilder (in Anlehnung an [LENZ, S. 27]), d. h. die Bilder werden zusammengezogen („kontrahiert“) und dadurch wird Konvergenz hergestellt.

2.3.2 Fixpunktsatz nach Banach

Ein klassischer Satz unter den Fixpunktsätzen (vgl. Kapitel 2.3.1), bei dem die Klasse der *stetigen Funktionen* betrachtet wird, ist der Fixpunktsatz nach Banach. Dieser spielt für den weiteren Aufbau dieser Arbeit eine wesentliche Rolle und wird nun eigens in diesem Kapitel vorgestellt.

Der Fixpunktsatz nach Banach wurde im Jahre 1922 von Stefan Banach (1892-1945), einem polnischen Mathematiker, erstmals formuliert und bewiesen. Auf ihm beruhen direkt oder indirekt nahezu alle iterativen Verfahren, die als Fixpunktproblem interpretierbar sind (in Anlehnung an [LENZ, S. 27]).

Fixpunktsatz nach Banach in IR^n: (In Anlehnung an [SCHRA, S. 80], [PLATO, S. 84-85] und [LENZ, S. 74-76])

$\varphi: D \subset IR^n \to IR^n$, D abgeschlossen. Weiters gelte $\varphi(D) \subset D$ (φ ist Selbstabbildung) und φ sei kontrahierend auf D bez. einer beliebigen Vektornorm $\|\cdot\|: IR^n \to IR$, das heißt, für eine Konstante (Lipschitzkonstante) $0 < L < 1$ sei $\| \varphi(\vec{x}_1) - \varphi(\vec{x}_2) \| \leq L \| \vec{x}_1 - \vec{x}_2 \| \quad \forall \vec{x}_1, \vec{x}_2 \in D$ erfüllt. Dann gilt Folgendes:

(a) φ besitzt genau einen Fixpunkt $\vec{x}^* \in D$, also es gilt $\varphi(\vec{x}^*) = \vec{x}^*$.

(b) Für jeden beliebig gewählten Startwert $\vec{x}_0 \in D$ liefert die Fixpunktiteration $\vec{x}_{n+1} = \varphi(\vec{x}_n)$ für n = 0, 1, 2, … eine gegen $\vec{x}^*$ konvergierende Folge, d. h. $\lim_{n \to \infty} \vec{x}_n = \vec{x}^*$.

Weiters gilt

$$\| \vec{x}_n - \vec{x}^* \| \leq \frac{L}{1-L} \| \vec{x}_n - \vec{x}_{n-1} \| \quad \text{(a-posteriori Fehlerabschätzung),}$$

$$\| \vec{x}_n - \vec{x}^* \| \leq \frac{L^n}{1-L} \| \vec{x}_1 - \vec{x}_0 \| \quad \text{(a-priori Fehlerabschätzung).}$$

Beweis: Nachzulesen in [SCHRA, S. 80-81].

2.3.3 Anwendungen

2.3.3.1 Beispiel 1: Fixpunktsatz nach Banach in IR

(In Anlehnung an [LENZ, S. 27-33])

Hier wird ein konkretes Beispiel einer Anwendung zum Fixpunktsatz nach Banach in IR^n mit n=1 vorgestellt. Bevor eine Umsetzung in Maple erfolgt, werden die dafür notwendigen Grundlagen wie Selbstabbildung und kontrahierende Abbildung (bzw. die einfacher zu überprüfende hinreichende Kontraktionsbedingung) betrachtet.

- φ ist eine Selbstabbildung

Die Funktion $\varphi: IR \to IR$ mit $x \mapsto \varphi(x) := \frac{5}{2}x(1-x)$ ist eine Selbstabbildung bezüglich des Intervalls $\left[\frac{7}{20};\frac{13}{20}\right]$ (in Abbildung 8 durch schwarze Balken markiert).

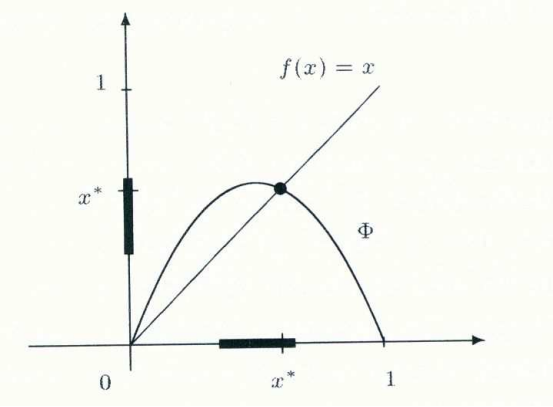


Abbildung 8: Beispielfunktion mit Fixpunkt [LENZ, S. 28]

Als nach unten geöffnete Parabel mit Scheitelpunkt in $\frac{1}{2}$ nimmt sie nämlich ihre Extremwerte im Intervall $\left[\frac{7}{20};\frac{13}{20}\right]$ sowie am Scheitelpunkt $\frac{1}{2}$ an, und diese

Funktionswerte liegen wegen $\varphi\left(\frac{7}{20}\right)=\varphi\left(\frac{13}{20}\right)=\frac{91}{160}=0.56875$ sowie $\varphi\left(\frac{1}{2}\right)=\frac{5}{8}=0.625$ alle wieder im Intervall $\left[\frac{7}{20}=0.35;\frac{13}{20}=0.65\right]$.

- φ ist eine kontrahierende Abbildung

Die Funktion φ ist eine kontrahierende Abbildung bez. des Intervalls $\left[\frac{7}{20};\frac{13}{20}\right]$. Dies folgt mit der *hinreichenden Kontraktionsbedingung* (siehe Kapitel 2.3.1.1) sofort aus der stetigen Differenzierbarkeit von φ sowie der Abschätzung

$$\max\left\{\left|\varphi'(x)\right|=\left|\frac{5}{2}-5x\right| \mid x\in\left[\frac{7}{20};\frac{13}{20}\right]\right\}=\frac{3}{4}=:L<1.$$

- Anwendung zum Fixpunktsatz nach Banach in IR^n mit n=1

Die Funktion φ ist – gemäß der beiden zuvor behandelten Punkte - eine *kontrahierende Selbstabbildung* bez. des Intervalls $\left[\frac{7}{20};\frac{13}{20}\right]$ mit der Lipschitzkonstante $L:=\frac{3}{4}$. Ihr somit eindeutig bestimmter Fixpunkt $x^*\in\left[\frac{7}{20};\frac{13}{20}\right]$ gemäß dem Fixpunktsatz nach Banach in IR^n mit n=1 ist z. B. durch Auflösen der quadratischen Fixpunktgleichung berechenbar und lautet $x^*=\frac{3}{5}$. [15]

---

[15] $x^*=\varphi(x^*)=\frac{5}{2}x^*(1-x^*)=\frac{5}{2}x^*-\frac{5}{2}(x^*)^2$

$\frac{5}{2}(x^*)^2-\frac{5}{2}x^*+x^*=0$

$\frac{5}{2}(x^*)^2-\frac{3}{2}x^*=0 \quad /\cdot 2$

$5(x^*)^2-3x^*=0$

$x_{1,2}{}^*=\frac{3\pm\sqrt{9-4.5.0}}{2.5}=\frac{3\pm 3}{10}$

$x_1{}^*=\frac{6}{10}=\frac{3}{5},\ x_2{}^*=\frac{0}{10}=0$ (trivialer Fixpunkt)

- Maple-Code

Ein Maple-Code zur Berechnung von x^* ausgehend vom Startwert $x_0 := \frac{\frac{7}{20}+\frac{13}{20}}{2} = \frac{1}{2} = 0.5$ [16] unter Ausnutzung der a-posteriori Fehlerabschätzung könnte z. B. wie folgt aussehen:

➢ Grundeinstellungen

Durch die Angabe *Digits:=n* wird die Genauigkeit der Rechnung auf n Stellen erhöht. Standardmäßig wird in Maple mit 10 Stellen gerechnet, daher ist die Angabe Digits:= 10 überflüssig. Trotzdem wird sie hier angeführt, um bei einem anders gewählten n eine sofortige Korrektur im Maple-Code zu ermöglichen.

```
> restart:with(plots):Digits:=10:
```

➢ Prozedur *FixpunktBanach* in Maple programmieren

Fixpunkt-Verfahren (Fixpunktsatz nach Banach)

Parameter der Prozedur FixpunktBanach (phi,L,xs0,tol)

phi:	Funktion, bei der ein Fixpunkt gesucht ist
L:	Lipschitzkonstante
xs0:	Startwert
tol:	Genauigkeit (Toleranz)

Ergebnis: Fixpunkt

```
> FixpunktBanach:=proc(phi,L,xs0,tol)
local x0,x1,i:      # i...Iterationsschritt
i:=0:x0:=xs0:       # Initialisierung von i bei 0 und x0 bei
                    Startwert xs0
```

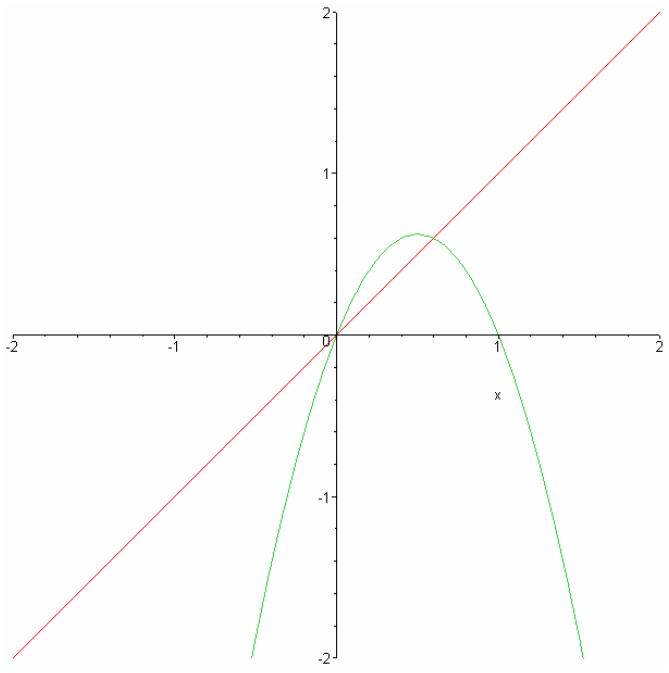
[16] Üblicherweise wird als Startwert x_0 das arithmetische Mittel aus den Intervallgrenzen gewählt.

```
while (i<1000) do   # Solange i kleiner 1000 ist, soll
                    folgendermaßen vorgegangen werden
  i:=i+1:                 # Betrachte den nächsten
                          Iterationsschritt
  x1:=evalf(phi(x0)):     # Formel des
                          Fixpunktiterationsverfahrens (2.1) zur
                          Berechnung des nächsten
                          Iterationsschrittes. evalf wertet den
                          Ausdruck in Gleitkommadarstellung aus
printf("x%d=%f\n",i,x1): # Ausgabe der Lösung für den eben
                          berechneten Iterationsschritt
  if (L/(1-L)*abs(x1-x0)<tol) then     # Ausnutzung der
                                       a-posteriori
                                       Fehlerabschätzung
    break:
  end if:
  x0:=x1:                 # Der x-Wert wird aktualisiert
end do:
return(x1):               # Ausgabe des Ergebnisses mit
                          gewünschter Genauigkeit
end proc:
```

- **Graphische Darstellung**

```
> phi:=x->5/2*x*(1-x);    # Definition der Funktion phi
f:=x->x:                  # 1.Mediane
plot({phi(x),f(x)},x=-2..2,view=[-2..2,2..-2]);
```

$$\phi := x \to \frac{5}{2} x\,(1 - x)$$



- **Berechnung des Fixpunktes**

```
> FixpunktBanach(phi,0.75,0.5,10^(-6));

x1=0.625000
x2=0.585938
x3=0.606537
x4=0.596625
x5=0.601659
x6=0.599164
x7=0.600416
x8=0.599791
x9=0.600104
x10=0.599948
x11=0.600026
x12=0.599987
x13=0.600007
x14=0.599997
x15=0.600002
x16=0.599999
x17=0.600000
x18=0.600000
x19=0.600000
                    0.6000001018
```

Der gesuchte Fixpunkt der Funktion phi ist somit 0.6 mit einer Genauigkeit von 10^(-6).

Anmerkung: Der triviale Fixpunkt 0 ist in der Graphik erkennbar.

2.3.3.2 Beispiel 2: Räuber-Beute-Modell

(In Anlehnung an [LENZ, S. 21-27])

Eine schöne Motivation für Iterationsverfahren bilden *diskrete dynamische Systeme*. Um ein konkretes Beispiel vor Augen zu haben, betrachte man das folgende, stark vereinfachte *Räuber-Beute-Modell*.

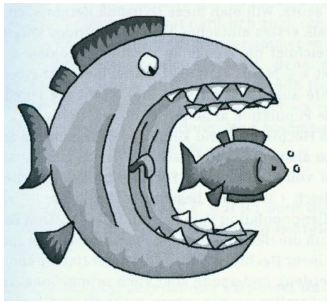


Abbildung 9: Räuber-Beute-Modell [LENZ, S. 21]

In einem See leben Hechte und Forellen. Sind zu Beginn nur einige Hechte im See und hinreichend viele Forellen, dann steht den Hechten ein praktisch unbegrenzter Vorrat an nachwachsenden Forellen zur Verfügung. Die Hechte vermehren sich auf Grund der guten Versorgungslage also sehr schnell. Das hat zur Folge, dass sich die Zahl der Forellen dadurch erheblich vermindert. Auf diese Art zerstören die Hechte ihre wesentliche Lebensgrundlage, nämlich ihre Hauptnahrungsquelle. Jetzt dezimiert die Überbevölkerung der Hechte ihren eigenen Bestand und bedroht ihre Population. Da die Population der Hechte stark abgenommen hat, wächst die Zahl der Forellen wieder. Dem stark reduzierten Hechtbestand geht es nun wieder gut. Er besitzt reichlich Futter und kann sich so wieder vergrößern.

- Modellbildung

Es entsteht auf diese Art eine Dynamik zwischen der Zahl der Hechte und der Zahl der Forellen, also zwischen Räuber und Beute. Will man diese Dynamik beschreiben, so bietet sich als erstes *einfaches Modell* folgendes Vorgehen an. Man bezeichnet den relativen Anteil der Hechte zu einem diskreten Zeitpunkt $n \in IN$ mit $x_n \in [0;1]$. Dabei bedeutet $x_n = 0$, dass die Hechte ausgestorben sind und damit die Forellen

ihre maximale Population annehmen können, und $x_n = 1$, dass es nur noch Hechte gibt und keine Forellen mehr im See sind. Geht man also stark vereinfachend davon aus, dass die Gesamtzahl (genauer: der Anteil) von Hechten und Forellen im See zu jedem Zeitpunkt gleich 1 ist, dann beschreibt die Größe $1 - x_n$ genau die Forellenpopulation zum Zeitpunkt $n \in IN$.[17] Zum Zeitpunkt $n+1$ ist nun die Hechtpopulation x_{n+1} einerseits proportional zu x_n (je mehr Hechte es gab, umso mehr Hechte konnten geboren werden), andererseits aber auch proportional zu $1 - x_n$ (je mehr Forellen es gab, umso mehr Hechte konnten satt werden).[18] Bezeichnet man die Proportionalitätskonstante mit $\alpha \in (0;\infty)$, dann ergibt sich folgender Zusammenhang

$$x_{n+1} = \alpha x_n (1 - x_n),\ n \in IN,$$

der auch als *logistische Wachstumsgleichung* bezeichnet wird. Auf der rechten Seite dieser Gleichung stehen zwei miteinander konkurrierende Faktoren. Wird x_n größer, so wird $1 - x_n$ kleiner und umgekehrt. Die Faktoren beschränken also wechselseitig das Wachstum von x_n. Der Parameter α ist ein Steuerungsparameter, der das qualitative Verhalten der Populationsfolge $(x_n)_{n \in IN}$ entscheidend beeinflusst und z. B. Konvergenz oder Divergenz der Folge implizieren kann[19]. Im Fall des Hecht-Forelle-Modells könnte α näherungsweise durch Messung bestimmt werden, wenn man zu gewissen Zeitpunkten $n \in IN$ die Anzahl von Hechten und Forellen im See kennen würde. Nimmt man also z. B. an, dass $\alpha = 2.5$ ist und die Hechtpopulation zu Beginn $x_0 = 0.35$ beträgt, so ergeben sich folgende Populationszahlen:

n		0	1	2	…	1000	…	$\to \infty$
x_n		0.35	0.56875	0.613184	…	0.6	…	$\frac{6}{10} = \frac{3}{5}$

```
> restart:
> x0:=0.35:alpha:=2.5:
```

[17] In diesem stark vereinfachten Modell wird also für jeden gestorbenen Hecht eine Forelle geboren und umgekehrt. Dies entspricht nicht der Realität, denn Hecht- und Forellenbestände entwickeln sich eher unabhängig voneinander (siehe im Folgenden das verbesserte/allgemeinere Modell).
[18] Eine mehr ins Detail gehende Betrachtung bez. der Proportionalität findet sich in Kapitel 2.3.3.3.
[19] Nähere Ausführungen diesbez. finden sich in Kapitel 2.3.3.3.

```
for i from 1 to 1000 do
  x1:=alpha*x0*(1-x0):
  printf("x%d=%g\n",i,x1):
  x0:=x1:
end do:
```

```
x1=0.56875
x2=0.613184
x3=0.592974
x4=0.60339
x5=0.598276
x6=0.600854
x7=0.599571
x8=0.600214
x9=0.599893
x10=0.600054
x11=0.599973
x12=0.600013
x13=0.599993
x14=0.600003
x15=0.599998
x16=0.600001
x17=0.6
x18=0.6
x19=0.6
x20=0.6
```

⋮ usw.

```
x999=0.6
x1000=0.6
```

Die Hechtpopulation und damit natürlich auch die Forellenpopulation münden also unter den angenommenen Anfangsbedingungen in einen stabilen Zustand, denn die Folge $(x_n)_{n\in IN}$ konvergiert gegen $x^* = 0.6$: Es wird 60% Hechte und 40% Forellen im See geben. Zu diesem Ergebnis hätte man aber auch auf einem völlig anderen Weg kommen können:

Man betrachte die Funktion $\varphi_\alpha : IR \to IR$,

$$\varphi_\alpha(x) := \alpha\, x(1-x),\ x \in IR,$$

so bedeutet ein stabiler Zustand $x^* \in IR$ im Sinne des Räuber-Beute-Modells, dass er die Fixpunktgleichung

$$\varphi_\alpha(x^*) = x^*$$

erfüllen muss, also ein Fixpunkt der Modellfunktion φ_α ist. Diesen Fixpunkt kann man im vorliegenden einfachen Fall nicht nur als Grenzwert der Iterationsfolge

$$x_{n+1} = \varphi_\alpha(x_n),\ n \in IN,$$

bestimmen (wie oben vorgeführt), sondern auch durch Auflösung der Fixpunktgleichung direkt berechnen (der triviale Fall $x^* = 0$ kann ausgeschlossen werden):

$$\varphi_\alpha(x^*) = x^* = \alpha x^*(1-x^*) \overset{x^* \neq 0}{\Leftrightarrow} x^* = \frac{\alpha - 1}{\alpha} \text{ [20]}. \qquad (2.3)$$

In unserem Beispiel ist $x^* = \frac{2.5-1}{2.5} = 0.6$.

- Allgemeineres Modell

In einem weiteren Schritt soll das betrachtete Populationsproblem dahingehend *verallgemeinert* werden, dass sich die Hecht- und Forellenbestände etwas unabhängiger voneinander entwickeln dürfen. Man bezeichnet den relativen Anteil der Hechte zu einem diskreten Zeitpunkt $n \in IN$ wieder mit $x_n \in [0;1]$ und den der Forellen mit $y_n \in [0;1]$. Es werde angenommen, dass die Hechtpopulation x_{n+1} zum Zeitpunkt $(n+1) \in IN$ positiv durch den Hecht- und Forellenbestand zum Zeitpunkt $n \in IN$ bestimmt wird gemäß

$$x_{n+1} = 0.6x_n + 0.6y_n,\ n \in IN.$$

Weiters werde angenommen, dass die Forellenpopulation y_{n+1} zum Zeitpunkt $(n+1) \in IN$ negativ durch den Hecht- und positiv durch den Forellenbestand zum Zeitpunkt $n \in IN$ bestimmt wird gemäß

$$y_{n+1} = -0.2x_n + 1.3y_n,\ n \in IN.$$

Bezeichnet man mit dem Symbol ~ die noch nicht erfolgte Normierung auf Gesamtpopulation 1, dann erhält man die Iterationsvorschrift

$$\tilde{x}_{n+1} = 0.6x_n + 0.6y_n,$$

$$\tilde{y}_{n+1} = -0.2x_n + 1.3y_n$$

$$x_{n+1} = \frac{\tilde{x}_{n+1}}{\tilde{x}_{n+1} + \tilde{y}_{n+1}}$$

$$y_{n+1} = \frac{\tilde{y}_{n+1}}{\tilde{x}_{n+1} + \tilde{y}_{n+1}}$$

[20] $x^* = \alpha x^*(1-x^*) \overset{x^* \neq 0}{\Leftrightarrow} 1 = \alpha(1-x^*) \Leftrightarrow 1 = \alpha - \alpha x^* \Leftrightarrow x^* = \frac{\alpha - 1}{\alpha}$

In Matrix-Vektor-Notation lässt sich das auch schreiben als

$$\begin{pmatrix} \tilde{x}_{n+1} \\ \tilde{y}_{n+1} \end{pmatrix} = \begin{pmatrix} 0.6 & 0.6 \\ -0.2 & 1.3 \end{pmatrix} \cdot \begin{pmatrix} x_n \\ y_n \end{pmatrix},$$

$$\begin{pmatrix} x_{n+1} \\ y_{n+1} \end{pmatrix} = \frac{1}{\tilde{x}_{n+1} + \tilde{y}_{n+1}} \cdot \begin{pmatrix} \tilde{x}_{n+1} \\ \tilde{y}_{n+1} \end{pmatrix}.$$

Man erhält also zu jedem Zeitpunkt $n \in IN$ einen *Populationsvektor* und die entstehende Folge ist eine *Folge von Vektoren*. Nimmt man z. B. wieder an, dass die Hechtpopulation zu Beginn $x_0 = 0.35$ beträgt und damit die Forellenpopulation $y_0 = 1 - x_0 = 0.65$, so ergeben sich folgende Populationsvektoren:

n		0	1	2	…	1000	…	$\to \infty$
x_n		0.35	0.436364	0.481752	…	0.6	…	$\frac{6}{10} = \frac{3}{5}$
y_n		0.65	0.563636	0.518248	…	0.4	…	$\frac{4}{10} = \frac{2}{5}$

```
> x0:=0.35:y0:=0.65:
  for i from 1 to 1000 do
  x1:=(0.6*x0+0.6*y0)/(0.6*x0+0.6*y0+(-0.2*x0+1.3*y0)):
  y1:=(-0.2*x0+1.3*y0)/(0.6*x0+0.6*y0+(-0.2*x0+1.3*y0)):
  printf("(x%d,y%d)=(%g,%g)\n",i,i,x1,y1):
  x0:=x1:y0:=y1:
end do:
```

```
(x1,y1)=(0.436364,0.563636)
(x2,y2)=(0.481752,0.518248)
(x3,y3)=(0.509609,0.490391)
(x4,y4)=(0.528362,0.471638)
(x5,y5)=(0.541781,0.458219)
(x6,y6)=(0.551812,0.448188)
(x7,y7)=(0.559554,0.440446)
(x8,y8)=(0.565681,0.434319)
(x9,y9)=(0.570625,0.429375)
(x10,y10)=(0.574678,0.425322)
```

⋮ usw.

```
(x109,y109)=(0.599999,0.400001)
(x110,y110)=(0.6,0.4)
(x111,y111)=(0.6,0.4)
(x112,y112)=(0.6,0.4)
(x113,y113)=(0.6,0.4)
```

⋮ usw.

```
(x999,y999)=(0.6,0.4)
(x1000,y1000)=(0.6,0.4)
```

Die Hechtpopulation und die Forellenpopulation münden also auch in diesem Modell unter den angenommenen Anfangsbedingungen in einen stabilen Zustand, denn die Vektorfolge $((x_n, y_n)^T)_{n\in IN}$ [21] konvergiert gegen den Populationsvektor $(x^*, y^*)^T = (0.6,0.4)^T$: Es wird also auch in diesem Fall 60% Hechte und 40% Forellen im See geben. Zu diesem Ergebnis hätte man wieder auf einem völlig anderen Weg kommen können:

Betrachtet man die Funktion $\varphi: IR^2 \to IR^2$,

$$\varphi((x,y)^T) := \begin{pmatrix} 0.6 & 0.6 \\ -0.2 & 1.3 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix},$$

so bedeutet ein stabiler *Zustandvektor* $(x^*, y^*)^T \in IR^2$ im Sinne des Räuber-Beute-Modells, dass er die Fixpunktgleichung

$$\varphi((x^*, y^*)^T) = (x^*, y^*)^T$$

erfüllen muss, also wieder ein Fixpunkt der Modellfunktion φ ist, die hier allerdings Vektoren auf Vektoren abbildet. Diesen Fixpunkt kann man im vorliegenden einfachen Fall nicht nur als Grenzwert der Iterationsfolge der Vektoren bestimmen (wie oben vorgeführt), sondern auch durch Auflösung der Fixpunktgleichung direkt berechnen:

$$\varphi((x^*, y^*)^T) = (x^*, y^*)^T = \begin{pmatrix} 0.6 & 0.6 \\ -0.2 & 1.3 \end{pmatrix} \cdot \begin{pmatrix} x^* \\ y^* \end{pmatrix} \Rightarrow$$

$$x^* = 0.6x^* + 0.6y^* \quad /- x^*$$

$$y^* = -0.2x^* + 1.3y^* \quad /- y^*$$

$$0 = -0.4x^* + 0.6y^* \quad \text{(I)}$$

$$0 = -0.2x^* + 0.3y^* \quad \text{(II)}$$

[21] Aus einem Zeilenvektor wird durch *Transposition* ein Spaltenvektor: $(x_n, y_n)^T = \begin{pmatrix} x_n \\ y_n \end{pmatrix}$

Die Gleichung (II) ist die Hälfte von Gleichung (I), somit linear abhängig und kann ignoriert werden. Es gilt

$$0.4x^* = 0.6y^* .$$

Durch die Normierung gilt außerdem

$$x^* + y^* = 1 \Rightarrow y^* = 1 - x^* .$$

Nun werden x^* und y^* berechnet:

$$0.4x^* = 0.6 \cdot (1 - x^*)$$

$$0.4x^* = 0.6 - 0.6x^*$$

$$x^* = 0.6$$

$$y^* = 1 - 0.6 = 0.4$$

In der linearen Algebra bedeutet $\varphi((x^*, y^*)^T) = (x^*, y^*)^T$, dass $(x^*, y^*)^T$ ein *Eigenvektor*[22] der Iterationsmatrix $\begin{pmatrix} 0.6 & 0.6 \\ -0.2 & 1.3 \end{pmatrix}$ zum *Eigenwert* 1 ist. Es handelt sich also um ein *Eigenwert-Eigenvektor-Problem.*

Definition: (in Anlehnung an [WIKI])
Ist V ein Vektorraum über einem Körper K (IR oder $\mathbb{C}$) und $f : V \to V$ eine lineare Abbildung von V in sich selbst (Endomorphismus), so bezeichnet man als *Eigenvektor* einen Vektor $\vec{v} \neq \vec{0}$, der durch f auf ein Vielfaches $\lambda \in K$ von sich selbst abgebildet wird:

$$f(\vec{v}) = \lambda \vec{v} .$$

Den Faktor λ nennt man dann den zugehörigen *Eigenwert*.

Anders formuliert: Hat für ein $\lambda \in K$ die Gleichung $f(\vec{v}) = \lambda \vec{v}$ eine Lösung $\vec{v} \neq \vec{0}$ (der Nullvektor ist natürlich immer eine Lösung), so heißt λ *Eigenwert* von f. Jede Lösung $\vec{v} \neq \vec{0}$ heißt *Eigenvektor* von f zum Eigenwert λ.

[22] Ein *Eigenvektor* einer Abbildung ist in der linearen Algebra ein vom Nullvektor verschiedener Vektor, dessen Richtung durch die Abbildung nicht verändert wird. Ein Eigenvektor wird also nur gestreckt (bzw. gestaucht), und man bezeichnet den Streckungsfaktor (bzw. Stauchfaktor) als *Eigenwert* der Abbildung (vgl. [WIKI]).

Ist der Vektorraum endlichdimensional, so kann jeder Endomorphismus f durch eine quadratische Matrix A beschrieben werden. Die obige Gleichung lässt sich dann als Matrizengleichung schreiben:

$$A \cdot \vec{x} = \lambda \cdot \vec{x} \, .$$

Man nennt eine Lösung $\vec{x} \neq \vec{0}$ und λ in diesem Fall Eigenvektor bzw. Eigenwert der Matrix A.

- Überblick

Nachdem im Rahmen des obigen kleinen Beispiels einige erste Szenarien für Iterationsverfahren und Fixpunktgleichungen sowohl im ein- als auch im mehrdimensionalen Fall skizziert wurden, geht es im Folgenden um die Vorstellung ausgewählter wichtiger Verfahren dieses Typs aus dem Umfeld der numerischen Mathematik.

Folgende „Wissensbausteine“, wie [LENZ, S. 26] betont, beschäftigen sich dabei mit *Techniken aus der Analysis* und behandeln *eindimensionale* und *mehrdimensionale Fragestellungen*:

- Fixpunktsatz nach Banach in IR (siehe Kapitel 2.3.2)
- Newton-Verfahren in IR und IR^n (siehe Kapitel 4.4 und 4.5), im Speziellen das Heron-Verfahren (siehe Kapitel 4.6.1)
- Sekanten-Verfahren (siehe Kapitel 4.3), im Speziellen das Regula-falsi-Verfahren (siehe Kapitel 4.2)

Folgende Wissensbausteine haben Techniken aus der *linearen Algebra* zum Gegenstand und spielen sich in *mehrdimensionalen Räumen* ab:

- Fixpunktsatz nach Banach in IR^n (siehe Kapitel 2.3.2)
- Gesamtschritt-Verfahren (siehe Kapitel 3.1)
- Einzelschritt-Verfahren (siehe Kapitel 3.2)

2.3.3.3 Beispiel 3: Grippeausbreitung im Kindergarten (Parabeliteration)

(In Anlehnung an [HUCK, S. 198-201] und [LOCH, S. 302-309])

Beispiel 3 lässt sich in analoger Weise wie Beispiel 2 lösen. Allerdings wird hier ein weiteres Anwendungsbeispiel näher vorgestellt und interessante Details ergänzend angeführt.

Bei der einfach gebauten nichtlinearen Funktion $\varphi_\alpha : x \to \alpha x(1-x) = \alpha x - \alpha x^2$ verhalten sich die Iterationsfolgen völlig unterschiedlich, je nachdem wie der Parameter $\alpha \in IR$ gewählt wird (siehe das folgende Beispiel).

In einem Kindergarten ist die Grippe ausgebrochen. Das Ziel ist es, ein mathematisches Modell für den epidemiologischen Verlauf dieser Krankheit zu entwickeln. Dazu führt man folgende Bezeichnungen ein:

P: Gesamtzahl der Kinder im Kindergarten (Population),
t_n: Zeitpunkt t_n (z. B. der n. Tag nach Ausbruch der Epidemie),
$K_n := K(t_n)$: Anzahl der erkrankten Kinder zum Zeitpunkt t_n,
$k_n := \frac{K_n}{P}$: relativer Anteil der zum Zeitpunkt t_n erkrankten Kinder,
α: Infektionsrate

In den ersten Tagen nach Ausbruch der Epidemie stellt die Kindergartenleiterin fest, dass die Zahl der erkrankten Kinder von einem Tag zum nächsten ziemlich proportional anwächst, d. h. es gilt eine Rekursion der Gestalt

$$k_{n+1} = \alpha k_n \text{ mit } \alpha > 1.$$

Man bezeichnet diese *lineare* Rekursion als *lineares Wachstumsmodell*. Verfolgt man diese Rekursion rückwärts, so ergibt sich

$$k_{n+1} = \alpha k_n = \alpha^2 k_{n-1} = \cdots = \alpha^{n+1} k_0 .$$

Anmerkung:

> Rekursion und Iteration sind im Wesentlichen gleichmächtige Sprachmittel. *Rekursion* (lat. *recurrere* „zurücklaufen“) betont, dass für die Berechnung von k_{n+1} auf den Wert k_n (und dann auf $k_{n-1},...,k_0$) *zurück*gegriffen wird. Bei der *Iteration* (lat. *iterare* „wiederholen“) findet für die Berechnung *von* k_{n+1} ausgehend von k_0 (und dann $k_1,...,k_{n-1}$) die wiederholte (schrittweise *voran*gehende) Anwendung desselben Rechenverfahrens statt.

Bei $k_0 > 0$, $\alpha > 1$ wächst die Folge $(k_n)_{n\in IN}$ [23] monoton, und ist unbeschränkt. Daraus folgt, dass ab einem gewissen Zeitpunkt t_m alle Werte k_n mit $n \geq m$ größer als 1 sind im Widerspruch zu ihrer Definition als relative Anteile. Es gibt natürlich nicht nur diesen mathematischen Widerspruch. Auch die Kindergartenleiterin stellt („experimentell“) fest, dass die Formel $k_{n+1} = \alpha k_n$ zwar in den ersten Tagen nach Ausbruch der Epidemie deren Verlauf gut wiedergibt, dass aber danach die Anzahl der Neuerkrankungen stark zurückgeht und nach einiger Zeit auf Null absinkt. Man versucht deshalb, das offensichtlich zu einfache Modell der Realität besser anzupassen. Es scheint vernünftig zu sein, den Anteil der Infizierten auch proportional zum Anteil $1-k_n$ der Gesunden anzusetzen. Denn nur solange gesunde Kinder vorhanden sind, können sich Neuerkrankungen ergeben, und deren Anzahl wird umso höher sein, je mehr Gesunde vorhanden sind. Insgesamt hat man dann eine *nichtlineare* Rekursion

$$k_{n+1} = \alpha k_n (1-k_n) .$$

Falls erst wenige Kinder erkrankt sind, also für k_n nahe bei Null, gilt

$$k_{n+1} \approx \alpha k_n .$$

Um den Verlauf der Epidemie analytisch und graphisch besser darstellen zu können, führt man die beiden Iterationsfunktionen

$$\varphi_\alpha : x \mapsto \alpha x ,$$

$$\psi_\alpha : x \mapsto \alpha x(1-x) ,$$

[23] In dieser hier vorliegenden Arbeit gilt $IN := \{0,1,2,3,...\}$.

ein, sodass im einfachen linearen Modell

$$k_{n+1} = \varphi_\alpha(k_n)$$

und im verbesserten Modell der **Parabeliteration**

$$k_{n+1} = \psi_\alpha(k_n)$$

gilt. Graphisch erhält man die Werte k_n, indem man den Graphen von φ_α bzw. ψ_α und die 1. Mediane zeichnet und ausgehend von $k_0 > 0$ die „Iterationstreppe“ durchläuft.

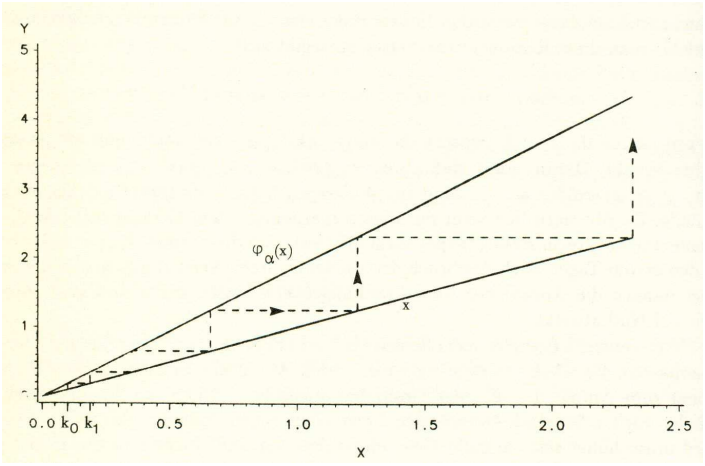


Abbildung 10: Graph von φ_α für $\alpha = \frac{15}{8}$ [LOCH, S. 304]

Man sieht, dass sich die Iterationstreppe von φ_α in Abbildung 10 öffnet, d. h. es gilt $k_n \to \infty$ für $n \to \infty$, während sich für ψ_α in Abbildung 11 ein von α abhängiger Grenzwert $k_\alpha^* := k^* = \lim_{n\to\infty} k_n$ einstellt ($k^* = \frac{\alpha - 1}{\alpha}$ in Analogie zu (2.3)). Die Parabeliteration wird nun etwas eingehender untersucht.

Es gelte $1 < \alpha \leq 2$, was eine realistische Annahme für den zu einem Epidemiemodell gehörenden Parameter α ist. Dann folgt

$$0 < k^* \leq \frac{1}{2},$$

d. h. k^* liegt in dem Teilintervall von [0;1], in dem ψ_α monoton wächst.

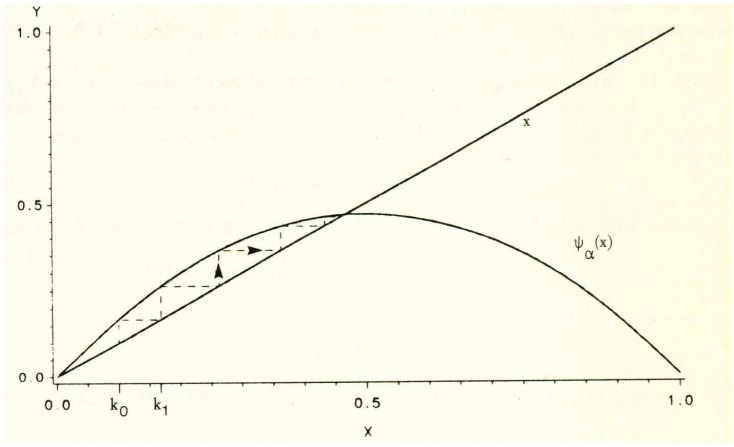


Abbildung 11: Graph von ψ_α für $\alpha = \frac{15}{8}$ [LOCH, S. 305]

Satz: (In Anlehnung an [LOCH, S. 306])

Es gelte $1 < \alpha \leq 2$. Dann konvergiert für jeden Startwert $0 < k_0 \leq k^* := \frac{\alpha - 1}{\alpha}$ die Iteration des logistischen Wachstumsmodells

$$k_{n+1} = \alpha k_n (1 - k_n) \text{ , n = 0,1,2,…,}$$

gegen k^*, und die Folge $(k_n)_{n \in IN}$ ist monoton.

(Man sagt auch, dass die Folgenglieder *monoton* gegen k^* *konvergieren*.)

Beweis: Nachzulesen in [LOCH, S. 306].

In [LOCH, S. 304-309] wird ein allgemeinerer Zugang bez. der Parabeliteration vorgestellt. Daraus werden im Folgenden einige interessante Inhalte skizziert, deren Details in der genannten Literaturquelle nachgelesen werden können.

Die Parabeliteration zeigt, dass sich eine nichtlineare Rekursion qualitativ sehr unterschiedlich verhalten kann, je nachdem, welchen Wert ein „Steuerparameter“ (in unserem Beispiel ist es die Größe α) hat.

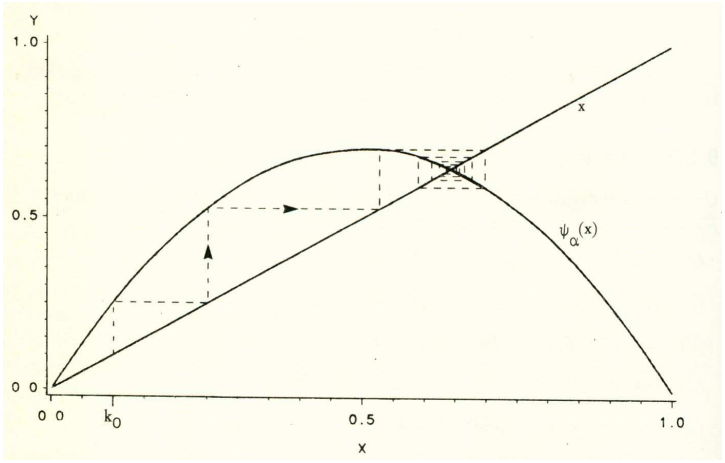


Abbildung 12: Graph von ψ_α für $\alpha = 2.8$: alternierende Konvergenz [LOCH, S. 306]

Für $\alpha > 2$ liegt der Fixpunkt $k^* = \frac{\alpha - 1}{\alpha}$ im Intervall $\left(\frac{1}{2};1\right]$, in dem ψ_α monoton fällt. Dies bedeutet, dass dann sicher keine *monotone* Konvergenz vorliegen kann, falls die Iteration überhaupt konvergiert. Man wird vermuten, dass bis zu einem gewissen $\alpha_0 > 2$ noch Konvergenz eintritt, während für alle α jenseits dieses kritischen Wertes α_0 - wobei $\alpha_0 = 3$ gilt gemäß [LOCH, S. 307] - Divergenz zu erwarten ist. Für $\alpha \in (2;\alpha_0 = 3)$ liegt *alternierende Konvergenz* vor, d. h. es gilt $(k_n - k^*)(k_{n+1} - k^*) < 0$.

Wählt man aber α nur wenig größer als $\alpha_0 = 3$, so ergibt sich ein überraschender Effekt.

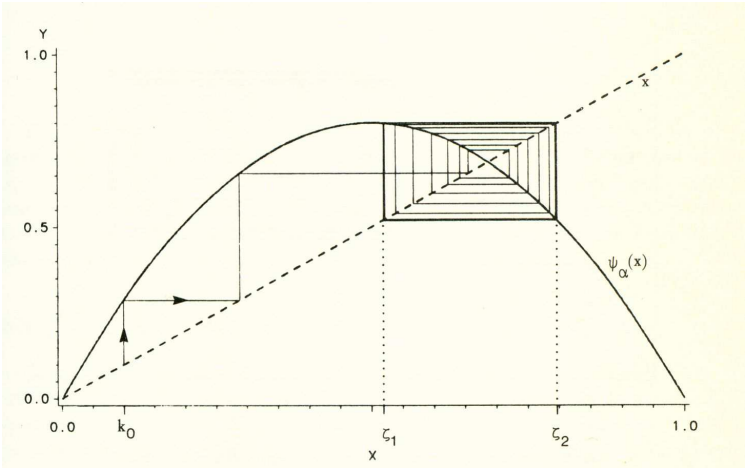


Abbildung 13: Graph von ψ_α für $\alpha = 3.2$: zweipunktiger Attraktor [LOCH, S. 307]

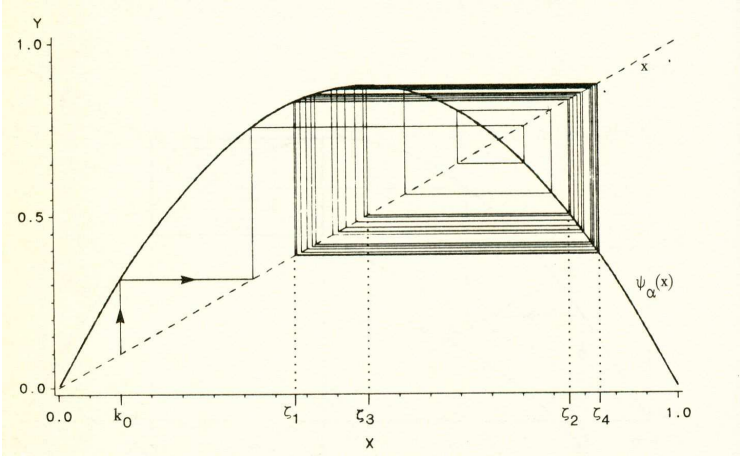


Abbildung 14: Graph von ψ_α für $\alpha = 3.5$: vierpunktiger Attraktor [LOCH, S. 308]

Im Fall von Abbildung 13 wird ein Startwert $k_0 \in (0;k^*)$ gewählt und die Iterationsfolge hat zwei Häufungspunkte ζ_1 und ζ_2, zwischen denen sie „hin und her springt". Man bezeichnet die Menge $\{\zeta_1, \zeta_2\}$ anschaulich als *Attraktor* der Iterationsfunktion ψ_α, da sie unabhängig vom gewählten Startwert $0 < k_0 < k^*$ die Iterationsfolge $(k_n)_{n\in IN}$ „anzieht". Allgemein bezeichnet man die Teilmenge $X \subseteq [0;1]$ als *Attraktor* von ψ_α, wenn für fast alle Startwerte $k_0 \in [0;1]$ jeder Punkt von X Häufungspunkt der Iterationsfolge $(k_n)_{n\in IN}$ ist.

In Abbildung 11 und Abbildung 12 ist der Attraktor einpunktig (Konvergenz bei $\alpha \in (1;\alpha_0 = 3)$), in Abbildung 13 zweipunktig und in Abbildung 14 vierpunktig. Erhöht man α weiter, so ergeben sich allgemein 2^m-punktige Attraktoren. Schließlich ergeben sich Attraktoren, die n-punktig mit jedem $n \in IN$ sind, d. h. es kann jede beliebige Zykluslänge auftreten. Allerdings sind diese Attraktoren (Zyklen der Länge n) wegen der begrenzten Auflösung von Graphik-Ausgabegeräten oft nicht mehr graphisch darstellbar.

Jenseits von $\alpha > 3.569...$ tritt *Chaos* ein, ein Phänomen, das MathematikerInnen, PhysikerInnen, BiologInnen, ChemikerInnen, ja sogar AstronomInnen, ÖkologInnen und VolkswirtInnen gleichermaßen fasziniert, weil es in ihren Arbeitsbereichen Systeme gibt, die sich chaotisch verhalten. Beispiele hierfür sind Populationsmodelle, chemische Reaktionen, das Wetter oder volkswirtschaftliche Modelle. Während sich bei einem n-punktigen Attraktor jede Iterationsfolge $(k_n)_{n\in IN}$ schließlich asymptotisch

wie ein Zyklus der Länge n verhält, ist im chaotischen Bereich überhaupt keine Regelmäßigkeit mehr vorhanden. Die Iterierten springen völlig „wirr“ hin und her; es stellt sich kein vorhersehbarer asymptotischer Grenzzustand ein (vgl. Abbildung 15).

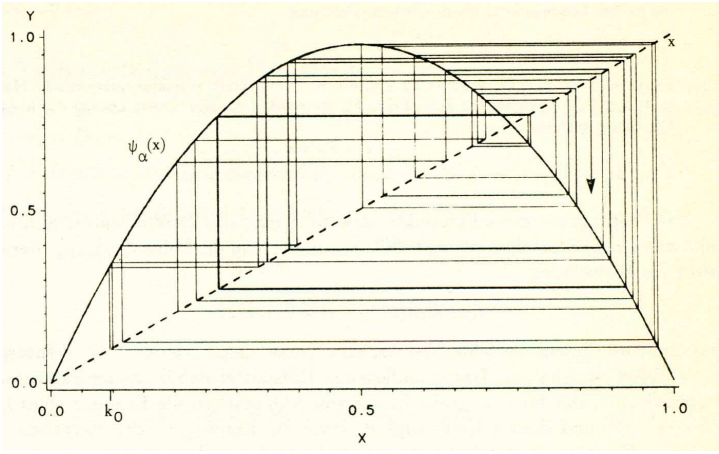


Abbildung 15: Graph von ψ_α für $\alpha = 3.9$: Chaos [LOCH, S. 309]

Im Rahmen dieser Diplomarbeit liegt das wesentliche Interesse darin, ob eine gegebene Iteration konvergiert oder divergiert. Die genaue Analyse des Konvergenz-Divergenzverhaltens und die Bestimmung von Attraktoren geht weit über die thematischen Zielsetzungen dieser Arbeit hinaus.

2.4 Fehleranalyse: Kondition, Stabilität und Konsistenz

(In Anlehnung an [WIKI])

Ein wichtiger Teil der Analyse von Algorithmen in der Numerik ist eine **Fehleranalyse**. Bei einer numerischen Berechnung kommen verschiedene Typen von Fehlern zum Tragen: Beim Rechnen mit Gleitkommazahlen treten unvermeidlich ***Rundungs- und Rechenfehler*** auf. Diese Fehler lassen sich zwar durch eine Erhöhung der Stellenzahl verkleinern, ganz beseitigen kann man sie aber nicht, da jeder Computer prinzipiell nur mit endlich vielen Stellen rechnen kann.

Wie die Lösung eines Problems auf Störungen in den Eingangsdaten (***Datenfehler***) reagiert, wird mit der **Kondition** gemessen. Hat ein Problem eine große bzw. schlechte Kondition, so hängt die Lösung des Problems empfindlich von den Eingangsdaten ab, was eine numerische Lösung in Frage stellt. Man spricht von einem *schlecht gestellten* Problem, das nach Möglichkeit durch eine Umformulierung umgangen werden sollte.

Gemäß [SCHRA, S. 6] versteht man unter *guter Kondition*, dass sich die Lösung nur wenig ändert, wenn die Daten des Problems verändert werden, während *schlechte Kondition* bedeutet, dass sich kleine Fehler in den Daten extrem stark („katastrophal“) auf die Lösung auswirken[24].

Das numerische Verfahren ersetzt das kontinuierliche mathematische Problem durch ein diskretes, also endliches Problem. Dabei tritt der sogenannte *Diskretisierungsfehler* bzw. ***Verfahrensfehler*** auf, der im Rahmen der **Konsistenzanalyse** abgeschätzt und bewertet wird. Dies ist notwendig, da ein numerisches Verfahren im Regelfall nicht die exakte Lösung liefert. Wie sich solche Fehler beim Weiterrechnen vergrößern, wird mit Hilfe der **Stabilitätsanalyse** bewertet.
Konsistenz und Stabilität des Algorithmus führen im Regelfall zu Konvergenz der (numerischen) Lösung gegen die analytische.

[24] In [SCHRA, S. 5-7] ist hierzu ein Beispiel nachzulesen.

Konzepte wie Konvergenzgeschwindigkeit oder Stabilität sind beim Rechnen (Berechnen numerischer Lösungen per Hand) sehr wichtig. So lässt beispielsweise eine hohe Konvergenzgeschwindigkeit darauf hoffen, schnell mit der Berechnung fertig zu werden. Und schon Gauß bemerkte, dass sich seine Rechenfehler beim Gaußschen Eliminationsverfahren manchmal desaströs auf die Lösung auswirkten und sie so komplett unbrauchbar machten. Er zog deswegen das Gauß-Seidel-Verfahren vor, bei dem Fehler durch das Ausführen eines weiteren Iterationsschrittes leicht ausgeglichen werden können.

Im Gegensatz zu den direkten Methoden sind iterative Verfahren weitgehend unempfindlich gegenüber Rundungsfehlern. Dies liegt daran, dass jede neue Näherungslösung als Ausgangsnäherung für die Iteration angesehen werden kann und sich somit Rundungsfehler i. A. nicht anhäufen können, sondern eher kompensiert werden (vgl. [KALT, S. 51]).

Die drei in der Numerik entscheidenden Fehlerbewertungsmechanismen sind Kondition, Stabilität und Konsistenz. Alle drei Größen analysieren die Entstehung oder Fortpflanzung von Fehlern, unterscheiden sich aber in der "Auswahl" der Fehlerquellen.

- Kondition

Die Konditionsbewertung geht davon aus, dass der Algorithmus genau funktioniert (ist somit unabhängig vom konkreten Lösungsverfahren), jedoch die Eingabedaten gestört sind. Die *Konditionszahl* stellt ein Maß für die Abhängigkeit der Lösung eines Problems von der Störung der Eingangsdaten dar. Sie beschreibt den Faktor, um den der Eingangsfehler im ungünstigsten Fall verstärkt wird.

- Stabilität

Die Stabilität vergleicht das Ergebnis des numerischen Verfahrens mit dem des exakten Verfahrens unter gestörten Eingabedaten. Ein Verfahren heißt *stabil*, wenn es gegenüber kleinen Störungen der Daten unempfindlich ist. Insbesondere bedeutet dies, dass sich Rundungsfehler nicht zu stark auf die Berechnung auswirken.

Die Beziehung zwischen Kondition und Stabilität lässt sich wie folgt beschreiben:

Es sei $f(x)$ das mathematische Problem in Abhängigkeit von der Eingabe x, und es sei $\tilde{f}$ der numerische Algorithmus, sowie $\tilde{x}$ die gestörten Eingabedaten. So möchte man den folgenden Fehler abschätzen:

$$\| f(x) - \tilde{f}(\tilde{x}) \|.$$

Mit der Dreiecksungleichung gilt:

$$\| f(x) - \tilde{f}(\tilde{x}) \| = \| f(x) - f(\tilde{x}) + f(\tilde{x}) - \tilde{f}(\tilde{x}) \| \leq \| f(x) - f(\tilde{x}) \| + \| f(\tilde{x}) - \tilde{f}(\tilde{x}) \|.$$

Hierbei bezeichnet

$\| f(x) - f(\tilde{x}) \|$ die **Kondition** und

$\| f(\tilde{x}) - \tilde{f}(\tilde{x}) \|$ die **Stabilität**.

Kurz zusammengefasst: Stabilität ist eine Eigenschaft des Algorithmus und die Kondition eine Eigenschaft des Problems.

- Konsistenz

Im Rahmen einer Konsistenzanalyse wird beispielsweise der entstehende Fehler in Abhängigkeit von einem gewählten Gitter oder einer gewählten Schrittweite betrachtet.

3 Iterationsverfahren zur Lösung linearer Gleichungssysteme

Das Gaußsche Eliminationsverfahren hat einen iterativen Teilaspekt, da sich etwas wiederholt (es soll immer eine weitere Stufe erzeugt werden), jedoch handelt es sich nicht um ein iteratives sondern um ein direktes/exaktes Verfahren. Das Gaußsche Eliminationsverfahren wird nur *einmal* durchgeführt und besteht aus *endlich vielen* Schritten. Iteration hingegen bedeutet, dass man *beliebig viele* Schritte bzw. Wiederholungen des Verfahrens (z.B. Gauß-Seidel-Verfahrens) machen kann, wie es für die Genauigkeit sinnvoll erscheint. Anders formuliert (vgl. [KALT, S. 51]): Man versucht ausgehend von einem Startvektor eine schrittweise Verbesserung der Lösung zu finden; eigentlich ist dies ein unendlicher Prozess, welchen man bei einem gewissen Genauigkeitsniveau der gefundenen Lösung beendet.

Das Gaußsches Eliminationsverfahren als klassisches direktes Lösungsverfahren für lineare Gleichungssysteme ist für große Matrizen allerdings zu aufwändig (vgl. Kapitel 1.1), daher werden in diesem Kapitel iterative Lösungsverfahren vorgestellt. Es handelt sich dabei um zwei lineare Fixpunkt-Verfahren, die zur Klasse der **Splitting-Verfahren** gehören:

- **Jacobi-Verfahren[25] oder Gesamtschritt-Verfahren**
- **Gauß-Seidel-Verfahren[26] oder Einzelschritt-Verfahren**

[25] Benannt nach Carl Gustav Jacob **Jacobi** (1804-1851) (gemäß [WIKI]).
[26] Das Verfahren wurde zuerst von Carl Friedrich **Gauß** (1777-1855) entwickelt, aber erst später durch Ludwig **Seidel** (1821-1896) veröffentlicht (gemäß [WIKI]).

3.1 Jacobi-Verfahren (Gesamtschritt-Verfahren)

3.1.1 Theoretische Grundlagen

(In Anlehnung an [MUNZ, S. 377-382])

Eine iterative Methode wird durch eine Iterationsvorschrift definiert. Auf der linken Seite steht die Iterierte im neuen Iterationsschritt, auf der rechten Seite eine Berechnungsvorschrift, in welche die alten Iterierten eingehen. Durch wiederholtes Ausführen werden sukzessive Näherungen der Lösung erhalten.

Man betrachte ein lineares Gleichungssystem (LGS) der Form $A \cdot \vec{x} = \vec{b}$ mit der n x n – Matrix A und dem gesuchten Lösungsvektor $\vec{x}$. Gesucht sind iterative Lösungen dieses LGS mit n Unbekannten:

$$a_{11} \cdot x_1 + a_{12} \cdot x_2 + ... + a_{1n} \cdot x_n = b_1$$
$$a_{21} \cdot x_1 + a_{22} \cdot x_2 + ... + a_{2n} \cdot x_n = b_2$$
$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$
$$a_{n1} \cdot x_1 + a_{n2} \cdot x_2 + ... + a_{nn} \cdot x_n = b_n$$

Wir lösen jeweils die k. Gleichung nach x_k auf (d. h. die 1. Gleichung nach x_1, die 2. Gleichung nach x_2,…, die n. Gleichung nach x_n):

$$x_k = -\frac{1}{a_{kk}} \left(\sum_{i=1}^{k-1} a_{ki} x_i + \sum_{i=k+1}^{n} a_{ki} x_i - b_k \right) \text{ mit } k = 1,2,...,n$$

und lösen diese Gleichung iterativ: Wir geben uns eine Anfangsschätzung bzw. einen Startwert $(x_1^{(0)}, x_2^{(0)}, ..., x_n^{(0)})$ für die Lösung $(x_1, x_2, ..., x_n)$ vor und bestimmen im (m+1). Schritt eine Lösung durch die Vorschrift

$$x_k^{(m+1)} = -\frac{1}{a_{kk}} \left(\sum_{i=1}^{k-1} a_{ki} x_i^{(m)} + \sum_{i=k+1}^{n} a_{ki} x_i^{(m)} - b_k \right) \text{ für } k = 1,2,...,n. \qquad (3.1)$$

(Jacobi-Verfahren oder Gesamtschritt-Verfahren).

Nach dieser Iterationsvorschrift wird das LGS gelöst.

Beim Jacobi-Verfahren beginnt man bei der (m+1). Iteration mit der 1. Gleichung, um $x_1^{(m+1)}$ durch die „alten“ Werte $x_1^{(m)}, x_2^{(m)}, ..., x_n^{(m)}$ zu berechnen. Anschließend wählt man die 2. Gleichung und berechnet $x_2^{(m+1)}$ aus den „alten“ Daten $x_1^{(m)}, x_2^{(m)}, ..., x_n^{(m)}$. Dabei könnte man im Prinzip schon benutzen, dass $x_1^{(m+1)}$ zu diesem Zeitpunkt schon berechnet wurde und voraussichtlich auch schon eine bessere Approximation darstellt. Man sollte also ein verbessertes Verfahren erhalten, indem man bei der Berechnung von $x_i^{(m+1)}$ die schon aktualisierten Werte $x_1^{(m+1)}, x_2^{(m+1)}, ..., x_{i-1}^{(m+1)}$ berücksichtigt:

$$x_k^{(m+1)} = -\frac{1}{a_{kk}}\left(\sum_{i=1}^{k-1} a_{ki} x_i^{(m+1)} + \sum_{i=k+1}^{n} a_{ki} x_i^{(m)} - b_k\right) \text{ für } k = 1,2,...,n. \qquad (3.2)$$

(Gauß-Seidel-Verfahren oder Einzelschritt-Verfahren)

Bemerkung zur ***Namensgebung*** des Gesamtschritt-Verfahrens von Jacobi und des Einzelschritt-Verfahrens von Gauß-Seidel:
Die Namen dieser Verfahren rühren daher, dass das ***Gesamtschritt-Verfahren*** *zunächst alle Komponenten* von $\vec{x}^{(m+1)}$ (approximierter Lösungsvektorvektor $\vec{x}$ nach dem (m+1). Iterationsschritt) *berechnet*, bevor diese in der weiteren Rechnung verwendet werden, während beim ***Einzelschritt-Verfahren*** in die Berechnung von $x_i^{(m+1)}$ bez. $\vec{x}^{(m+1)}$ die *zuvor berechneten Werte* $x_j^{(m+1)}$, $j = 1,2,...,i-1$ *sofort eingehen* (gemäß [OPF, S. 182]).

Den beiden Iterationsverfahren ist eines gemeinsam: Bei Erreichen der gewünschten Genauigkeit wird die Iteration abgebrochen. Um ein Abbruchkriterium zu erhalten, bestimmt man nach jeder Iteration das Residuum. Im m. Iterationsschritt ist es durch

$$R_k^{(m)} = \sum_{i=1}^{n} a_{ki} x_i^{(m)} - b_k \quad \text{für } k = 1,2,...,n$$

gegeben.

Anmerkung: (In Anlehnung an [WIKI])

> Als *Residuum* (lat. „das Zurückgebliebene“) bezeichnet man speziell in der numerischen Mathematik die Abweichung vom gewünschten Ergebnis, welche

entsteht, wenn etwa in eine Gleichung Näherungslösungen eingesetzt werden. Angenommen, es sei eine Funktion f gegeben und man möchte ein x finden, so dass $f(x) = b$. Mit einer Näherung x_0 an x ist das Residuum $r = b - f(x_0)$, der Fehler hingegen $x_0 - x$.

Der Fehler ist in der Regel unbekannt, da x unbekannt ist, weswegen dieser als Abbruchkriterium in einem numerischen Verfahren nicht benutzt werden kann. Das Residuum dagegen ist stets verfügbar.

Wenn das Residuum klein ist, folgt in vielen Fällen, dass die Näherung nahe bei der Lösung liegt, das heißt $| x_0 - x | << 1$.

Vom Residuumvektor nehmen wir die maximale Komponente

$$R^{(m)} = \max_{k=1,\ldots,n} | R_k^{(m)} | . \qquad (3.3)$$

Als ***Abbruchkriterium*** fordert man in der Regel, dass sowohl

1.) das Residuum (3.3) kleiner als eine gewisse Vorgabe ist ($R^{(m)} < \delta_1$) als auch
2.) die Differenz der Werte zweier aufeinander folgender Iterationen kleiner als eine vorgegebene Schranke sind ($\max_{k=1,\ldots,n} \| \vec{x}_k^{(m)} - \vec{x}_k^{(m-1)} \| < \delta_2$).

- Herleitung des Jacobi- und Gauß-Seidel-Verfahrens

Allgemeine theoretische Aussagen und Informationen über die Struktur dieser beiden Iterationsverfahren erhält man über die Matrix-Formulierung: Dazu wird die ursprüngliche Matrix A in die Summe

$$A = L + D + U$$

zerlegt bzw. „aufgesplittet“, worauf die Bezeichnung **Splitting-Verfahren** zurückführbar ist (vgl. [WIKI] und [KANZ, S. 133-134]). Dabei ist L (lower) die Matrix, welche die Elemente von A im unteren Dreieck enthält, U (upper) ist die Matrix, welche die Elemente im oberen Dreieck enthält und D die Matrix mit den Diagonalelementen. Die Matrizen haben somit die Gestalt

$$L = \begin{pmatrix} 0 & \cdots & \cdots & 0 \\ * & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ * & \cdots & * & 0 \end{pmatrix}, \quad D = \begin{pmatrix} * & & & \\ & \ddots & & \\ & & \ddots & \\ & & & * \end{pmatrix}, \quad U = \begin{pmatrix} 0 & * & \cdots & * \\ \vdots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & * \\ 0 & \cdots & \cdots & 0 \end{pmatrix},$$

wobei das Symbol * für ein Matrixelement steht, welches ungleich Null sein darf. Es wird hier vorausgesetzt, dass die Diagonalmatrix D regulär ist und somit die Inverse D^{-1} existiert.

Wir setzen diese Aufspaltung in das lineare Gleichungssystem $A\vec{x} = \vec{b}$ ein und erhalten

$$(L + D + U)\vec{x} = \vec{b}\,.$$

Behält man hier D auf der linken Seite und schafft alles andere auf die rechte Seite, ergibt sich

$$D\vec{x} = -(L + U)\vec{x} + \vec{b}\,.$$

Mit der inversen Matrix D^{-1} multipliziert folgt

$$\vec{x} = -D^{-1}(L + U)\vec{x} + D^{-1}\vec{b}$$

und damit der Ausgangspunkt für eine Iterationsvorschrift

$$\vec{x}^{(m+1)} = -D^{-1}(L + U)\vec{x}^{(m)} + D^{-1}\vec{b}$$

mit $m = 0,1,2,\ldots$. Schaut man sich diese Gleichung komponentenweise an, dann sieht man leicht, dass dies gerade die Matrix-Schreibweise des Jacobi-Verfahrens (3.1) ist.

Allgemein können wir ein lineares Iterationsverfahren damit durch die Vorschrift

$$\vec{x}^{(m+1)} = M\vec{x}^{(m)} + N\vec{b} \ , \ m = 0,1,2,... \qquad (3.4)$$

definieren.

Durch die Wahl

$$M_J := -D^{-1}(L+U) \, , \ N_J := D^{-1}$$

erhält man das *Jacobi-Verfahren*.

Beim *Gauß-Seidel-Verfahren* werden die schon berechneten Werte im neuen Iterationsschritt benutzt. Die zugehörigen Koeffizienten stehen in der unteren Dreiecksmatrix. Man erhält somit für dieses Verfahren zunächst

$$(L+D)\vec{x} = -U\vec{x} + \vec{b}$$ [27]

und daraus die Iterationsvorschrift

$$\vec{x}^{(m+1)} = -(L+D)^{-1}U\vec{x}^{(m)} + (L+D)^{-1}\vec{b}$$

mit den Iterationsmatrizen

$$M_{GS} := -(L+D)^{-1}U \ , \ N_{GS} := (L+D)^{-1} .$$

Gemäß [WIKI] sind Iterationsverfahren der Form (3.4)

- *linear*, d. h. $\vec{x}^{(m+1)}$ hängt nur linear von $\vec{x}^{(m)}$ ab,
- *stationär*, d.h. M und $N\vec{b}$ sind unabhängig von der Schrittnummer der Iteration und
- *einstufig*, d.h. nur der letzte und nicht noch weitere Näherungsvektoren werden verwendet.

Beim ersten Iterationsschritt hat m den Wert Null. Das Ergebnis der Rechnung ist ein erster Näherungswert $\vec{x}^{(1)}$ für den gesuchten Lösungsvektor $\vec{x}$. Diesen Näherungswert kann man seinerseits in die Iterationsvorschrift einsetzen und gewinnt einen besseren Näherungswert $\vec{x}^{(2)}$, den man wieder einsetzen kann. Wiederholt man diesen Vorgang, gewinnt man eine Folge von Werten, die sich dem Lösungsvektor immer mehr annähern, wenn die Konvergenzbedingungen (siehe folgender Satz) erfüllt sind: $\vec{x}^{(0)}, \vec{x}^{(1)}, \vec{x}^{(2)}, ... \to \vec{x}$ (gemäß [WIKI]).

[27] Die schon berechneten Werte werden auf die rechte Seite des LGS gebracht.

- Konvergenz des Jacobi- und Gauß-Seidel-Verfahrens

Satz: (In Anlehnung an [MUNZ, S. 382])

Die Iteration (3.4) ist genau dann für jeden Startvektor $\vec{x}^{(0)}$ und jede rechte Seite $\vec{b}$ ***konvergent***, wenn

$$\rho(M) < 1 \text{ mit } \rho(M) = \max_{k=1,\ldots,n} |\lambda_k(M)|$$

gilt. Dabei ist $\rho(M)$ der *Spektralradius* der Iterationsmatrix M, welcher der betragsmäßig größte Eigenwert von M ist. λ_k bezeichnet den k. *Eigenwert*[28] von M.

$\rho(M)$ sollte möglichst klein sein, da dadurch die Konvergenzgeschwindigkeit bestimmt wird (gemäß [WIKI]).

Aus diesem Satz lassen sich verschiedene Bedingungen ableiten, bei denen die Konvergenz der Iterationsverfahren unabhängig vom Startvektor und der rechten Seite gesichert ist. Ein Kriterium dieser Art liefert z. B. der folgende Satz:

Satz: (In Anlehnung an [MUNZ, S. 382] und [WIKI])

Ist A ***strikt diagonaldominant***, das heißt es gilt

$$|a_{kk}| > \sum_{\substack{k=1 \\ i \neq k}}^{n} |a_{ki}| \text{ mit } k = 1,2,\ldots,n \text{ und } i = 1,2,\ldots,n,$$[29]

dann ist das Jacobi- und das Gauß-Seidel-Verfahren ***konvergent***.

Splitting-Verfahren besitzen eine lineare Konvergenzgeschwindigkeit (gemäß [WIKI]). Also besitzen das Jacobi- und Gauß-Seidel-Verfahren die Konvergenzordnung p=1.

Gemäß [LENZ, S. 81-85] kann sowohl das Jacobi- als auch das Gauß-Seidel-Verfahren konvergieren, wenn die Ausgangsmatrix A das starke Zeilensummenkriterium (bzw. die strikte Diagonaldominanz) nicht erfüllt. Das

[28] Die Definiton für *Eigenwert* ist in Kapitel 2.3.3.2 nachzulesen.

[29] In Worten formuliert (in Anlehnung an [WIKI]): Eine Matrix $A \in IR^{n \times n}$ heißt *strikt diagonaldominant*, falls die Beträge ihrer Diagonalelemente a_{kk} jeweils größer sind als die Summe der Beträge der restlichen jeweiligen Zeileneinträge a_{ki}. Dieses Kriterium wird auch als ***starkes Zeilensummen-kriterium*** bezeichnet.

Kriterium ist also lediglich hinreichend, jedoch nicht notwendig. Ferner konvergiert das Gauß-Seidel-Verfahren aufgrund der unmittelbaren Benutzung bereits berechneter Näherungen i. A. schneller als das Jacobi-Verfahren. Dies ist am folgenden Beispiel in Maple ersichtlich (siehe Kapitel 3.3).

- Definition Bandmatrix

(In Anlehnung an [WIKI] und [PLATO, S. 61])

Bandmatrizen sind *dünnbesetzte* Matrizen spezieller Struktur. Sie entstehen häufig bei der Diskretisierung von gewöhnlichen oder partiellen Differentialgleichungen. (vgl. Kapitel 2.1.3)

Mit *Bandmatrix* wird eine Matrix bezeichnet, bei der neben der Hauptdiagonale nur eine bestimmte Anzahl Nebendiagonalen Elemente ungleich Null aufweist. Sind nur eine untere und eine obere Nebendiagonale ungleich Null, so spricht man von *Tridiagonalmatrizen*[30].

Definition: (In Anlehnung an [WIKI] und [PLATO, S. 61])

$A \in IR^{n\times n}$ heißt **Bandmatrix**, falls für ihre Elemente a_{ij} gilt:

$$a_{ij} = 0 \text{ für } i > p + j \text{ bzw. } j > q + i \text{ mit gewissen } p, q \in IN .$$

Neben der Hauptdiagonale sind also nur p untere und q obere Nebendiagonalen besetzt.

$$A = \begin{pmatrix} a_{11} & \cdots & a_{1,q+1} & 0 & \cdots & \cdots & \cdots & 0 \\ \vdots & \ddots & & \ddots & \ddots & & & \vdots \\ a_{p+1,1} & & \ddots & & \ddots & \ddots & & \vdots \\ 0 & \ddots & & \ddots & & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & & \ddots & & \ddots & 0 \\ \vdots & & \ddots & \ddots & & \ddots & & a_{n-q,n} \\ \vdots & & & \ddots & \ddots & & \ddots & \vdots \\ 0 & \cdots & \cdots & \cdots & 0 & a_{n,n-p} & \cdots & a_{nn} \end{pmatrix}$$

[30] In Kapitel 3.1.2 wird eine Matrix $A = \begin{pmatrix} 2 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 2 \end{pmatrix}$ betrachtet. Bei A handelt es sich um eine spezielle Bandmatrix (Tridiagonalmatrix).

3.1.2 Ein Beispiel in Maple

(In Anlehnung an [MUNZ, S.378])

Anhand eines konkreten Beispiels wird die direkte Umsetzung des Jacobi-Verfahrens in Maple behandelt.

Wir betrachten dabei das folgende lineare Gleichungssystem:

$$\begin{aligned} 2x_1 + 1x_2 + 0x_3 &= 10 \\ 1x_1 + 2x_2 + 1x_3 &= 10 \\ 0x_1 + 1x_2 + 2x_3 &= 10 \end{aligned}$$

Somit ist $A = \begin{pmatrix} 2 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 2 \end{pmatrix}$ die Koeffizientenmatrix (eine Bandmatrix laut Fußnote 30) und $\vec{b} = \begin{pmatrix} 10 \\ 10 \\ 10 \end{pmatrix}$ der Vektor der rechten Seite.

Hinweise:

- Durch die Angabe *Digits:=n* wird die Genauigkeit der Rechnung auf n Stellen erhöht. Standardmäßig wird in Maple mit 10 Stellen gerechnet.
- In [LENZ, S. 81] und [LOCH, S. 330] wird darauf hingewiesen, dass in der Praxis die Implementierung des Jacobi-Verfahrens bzw. Gauß-Seidel-Verfahrens auf dem Computer i. A. komponentenweise vorgenommen wird (Verwendung von (3.1) bzw. (3.2)), während man bei Handrechnung der Matrixschreibweise den Vorzug gibt (Verwendung von (3.4)).

Parameter:

eq[1]…eq[n]:	lineare Gleichungen
x1…xn:	Variablen der Gleichungen, nach denen aufgelöst werden soll

Der geringfügige Unterschied der Maple-Codierung des Gauß-Seidel-Verfahrens gegenüber dem Jacobi-Verfahren ist durch einen grauen Hintergrund zur Verdeutlichung hervorgehoben.

Im Folgenden wird das Jacobi-Verfahren (3.1) direkt umgesetzt, ohne die Unterstützung von speziellen Maple-Befehlen, die die Umsetzung vereinfachen würden. Zuerst werden die Koeffizientenmatrix und der Vektor der rechten Seite aufgebaut. In der Hauptschleife, die n_iter Mal durchlaufen wird, wird der Algorithmus (3.1) umgesetzt. Nach jedem Iterationsschritt wird für die drei Unbekannten x_1 , x_2 und x_3 der neu errechnete Wert ausgegeben.

```
> restart:
> with(linalg):          # Das Paket "Lineare Algebra" laden

> n_iter:= 100:          # Anzahl der Iterationsschritte
> N:=3:                  # Dimension der NxN-
                         Koeffizientenmatrix A (Bandmatrix),
                         des Vektors b der rechten Seite und
                         des Start- bzw. Lösungsvektors x

> A:=band([1,2,1],N);
> b:=[seq(10,i=1..N)];   # Vektor b als Folge (engl. sequence)
                         darstellen
> x:=[seq(0.0,i=1..N)];  # Lösungsvektor x initialisiert bei
                         Startvektor = Nullvektor. Eingabe von
                         0.0 statt 0, damit das gesuchte
                         Ergebnis für x1, x2 und x3 als
                         Dezimalzahl und nicht als Bruch
                         ausgegeben wird
```

$$A := \begin{bmatrix} 2 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 2 \end{bmatrix}$$

$$b := [10, 10, 10]$$

$$x := [0., 0., 0.]$$

```
> for j from 1 to n_iter do        # j=1,2,...,n_iter
                                   Hauptschleife

>    for k from 1 to N do          # Zeilenindex in (3.1)

>       e:=0: f:=0:                    # Summen e und f
                                       initialisiert bei 0
```

```
>       for i from 1 to k-1 do          # Spaltenindex in (3.1)
>         e := e + A[k,i] * x[i]:       # e beginnt bei Null
                                        und addiert einen
                                        Summanden nach dem
                                        anderen, bis
                                        die erste Summe im
                                        Jacobi-Verfahren (3.1)
                                        herauskommt
>       end do:

>       for i from k+1 to N do
>         f := f + A[k,i] * x[i]:  # f beginnt bei Null und
                                   berechnet die zweite Summe
                                   im Jacobi-Verfahren (3.1)
>       end do:

>       x_neu[k] := -(1/(A[k,k]))*(e+f-b[k]):     # Umsetzung
                                                  von(3.1)

>       end do:

>       for k from 1 to N do  # Man betrachte die k. Komponente
                              des Lösungsvektors
                              x=[x1,x2,...,xN] mit k=1,2,...,N
>         x[k] := x_neu[k]:   # Das alte xk wird aktualisiert
>       end do:

>       print(`Die Iterationslösung für x1, x2, x3 lautet
        nach`, j,`Iterationen `, x[1],  x[2], x[3]);

> end do:
```

Die Iterationslösung für x1, x2 , x3 lautet nach , 1, *Iterationen* , 5.000000000, 5.000000000, 5.000000000

Die Iterationslösung für x1, x2 , x3 lautet nach , 2, *Iterationen* , 2.500000000, -0., 2.500000000

Die Iterationslösung für x1, x2 , x3 lautet nach , 3, *Iterationen* , 5.000000000, 2.500000000, 5.000000000

Die Iterationslösung für x1, x2 , x3 lautet nach , 4, *Iterationen* , 3.750000000, -0., 3.750000000

Die Iterationslösung für x1, x2 , x3 lautet nach , 5, *Iterationen* , 5.000000000, 1.250000000, 5.000000000

Die Iterationslösung für x1, x2 , x3 lautet nach , 6, *Iterationen* , 4.375000000, -0., 4.375000000

Die Iterationslösung für x1, x2 , x3 lautet nach , 7, *Iterationen* , 5.000000000, 0.625000000, 5.000000000

Die Iterationslösung für x1, x2 , x3 lautet nach , 8, *Iterationen* , 4.687500000, -0., 4.687500000

Die Iterationslösung für x1, x2 , x3 lautet nach , 9, *Iterationen* , 5.000000000, 0.312500000, 5.000000000

Die Iterationslösung für x1, x2 , x3 lautet nach , 10, *Iterationen* , 4.843750000, -0., 4.843750000

⋮ usw.

Die Iterationslösung für x1, x2 , x3 lautet nach , 64, *Iterationen* , 4.999999999, -0., 4.999999999

Die Iterationslösung für x1, x2 , x3 lautet nach , 65, *Iterationen* , 5.000000000, 0.100000000010^{-8}, 5.000000000

Die Iterationslösung für x1, x2 , x3 lautet nach , 66, *Iterationen* , 5.000000000, -0., 5.000000000

Die Iterationslösung für x1, x2 , x3 lautet nach , 67, *Iterationen* , 5.000000000, -0., 5.000000000

⋮ usw.

Die Iterationslösung für x1, x2 , x3 lautet nach , 100, *Iterationen* , 5.000000000, -0., 5.000000000

3.2 Gauß-Seidel-Verfahren (Einzelschritt-Verfahren)

3.2.1 Theoretische Grundlagen

Siehe Kapitel 3.1.1.

3.2.2 Ein Beispiel in Maple

(In Anlehnung an [MUNZ, S. 379])

Es wird dasselbe konkrete lineare Gleichungssystem wie in Kapitel 3.1.2 betrachtet. Grund dafür ist, dass dann in Kapitel 3.3 ein Vergleich der beiden Verfahren (Jacobi versus Gauß-Seidel) anhand dieses Beispiels erläutert werden kann.

Im Folgenden wird das Gauß-Seidel-Verfahren (3.2) direkt umgesetzt, ohne die Unterstützung von speziellen Maple-Befehlen, die die Umsetzung vereinfachen würden. Zuerst werden die Koeffizientenmatrix und der Vektor der rechten Seite aufgebaut. In der Hauptschleife, die n_iter Mal durchlaufen wird, wird der Algorithmus (3.2) umgesetzt. Nach jedem Iterationsschritt wird für die drei Unbekannten x_1 , x_2 und x_3 der neu errechnete Wert ausgegeben.

```
> restart:
> with(linalg):

> n_iter:=40:
> N:=3:
> A:=band([1,2,1],N):
> b:=[seq(10,i=1..N)]:
> x:=[seq(0.0,i=1..N)]:

> for j from 1 to n_iter do

>    for k from 1 to N do

>      e:=0: f:=0:

>      for i from 1 to k-1 do
>        e := e + A[k,i] * x[i]:
>      end do:
```

```
>       for i from k+1 to N do
>         f := f + A[k,i] * x[i]:
>       end do:

>       x[k] := -(1/(A[k,k]))*(e+f-b[k]): # Umsetzung von (3.2)

>     end do:

>     print(`Die Iterationslösung x1, x2, x3 lautet
      nach `, j,`Iterationen `,x[1],x[2],x[3]);

> end do:
```

Die Iterationslösung x1, x2, x3 lautet nach , 1, *Iterationen* , 5.000000000, 2.500000000, 3.750000000

Die Iterationslösung x1, x2, x3 lautet nach , 2, *Iterationen* , 3.750000000, 1.250000000, 4.375000000

Die Iterationslösung x1, x2, x3 lautet nach , 3, *Iterationen* , 4.375000000, 0.6250000000, 4.687500000

Die Iterationslösung x1, x2, x3 lautet nach , 4, *Iterationen* , 4.687500000, 0.3125000000, 4.843750000

Die Iterationslösung x1, x2, x3 lautet nach , 5, *Iterationen* , 4.843750000, 0.1562500000, 4.921875000

Die Iterationslösung x1, x2, x3 lautet nach , 6, *Iterationen* , 4.921875000, 0.07812500000, 4.960937500

Die Iterationslösung x1, x2, x3 lautet nach , 7, *Iterationen* , 4.960937500, 0.03906250000, 4.980468750

Die Iterationslösung x1, x2, x3 lautet nach , 8, *Iterationen* , 4.980468750, 0.01953125000, 4.990234375

Die Iterationslösung x1, x2, x3 lautet nach , 9, *Iterationen* , 4.990234375, 0.009765625000, 4.995117188

Die Iterationslösung x1, x2, x3 lautet nach , 10, *Iterationen* , 4.995117188, 0.004882812000, 4.997558594

⋮ usw.

Die Iterationslösung x1, x2, x3 lautet nach , 32, *Iterationen* , 4.999999999, 0.100000000010^{-8}, 5.000000000

Die Iterationslösung x1, x2, x3 lautet nach , 33, *Iterationen* , 5.000000000, -0., 5.000000000

Die Iterationslösung x1, x2, x3 lautet nach , 34, *Iterationen* , 5.000000000, -0., 5.000000000

⋮ usw.

Die Iterationslösung x1, x2, x3 lautet nach , 40, *Iterationen* , 5.000000000, -0., 5.000000000

Anmerkung:

Beim Gauß-Seidel-Verfahren verzichtet man auf eine *alle Komponenten betreffende* (wegen x[k]:=x_neu[k] für k=1,2,…,N=3) Aktualisierung des alten Vektors $\vec{x}^{(j)}$ mit dem neuen Vektor $\vec{x}^{(j+1)}$, wie es das Jacobi-Verfahren verlangt. Bereits aktualisierte Komponenten sollen nämlich sofort in der Rechnung berücksichtigt werden.

3.3 Jacobi- und Gauß-Seidel-Verfahren im Vergleich: Welches Verfahren konvergiert schneller?

Bei dem in Kapitel 3.1.2 und 3.2.2 behandelten Beispiel besitzt das Gauß-Seidel-Verfahren gegenüber dem Jacobi-Verfahren eine **schnellere Konvergenz**. Das Jacobi-Verfahren kommt auf die richtige Lösung im 66. Iterationsschritt, das Gauß-Seidel-Verfahren bereits im 33. Iterationsschritt.

4 Iterationsverfahren zur Lösung nichtlinearer Gleichungen/Gleichungssysteme

Ganz allgemein betrachten wir in diesem Kapitel die numerische Lösung einer Gleichung der Form $f(x) = 0$. Gesucht ist somit eine Nullstelle der i. A. nichtlinearen Funktion $f = f(x)$. Ist die nichtlineare Funktion nicht gerade eine quadratische Funktion, dann ist man darauf angewiesen, dies näherungsweise auszuführen. Vorausgesetzt sei im Folgenden immer, dass die Funktion $f = f(x)$ im betrachteten Bereich zumindest *stetig* ist. (In Anlehnung an [MUNZ, S. 369])

Im Folgenden werden verschiedene Verfahren zur Lösung **nichtlinearer Gleichungen** vorgestellt, die zur Klasse der nichtlinearen Fixpunkt-Verfahren in IR gehören:

- **Bisektionsverfahren** (Intervallhalbierungsverfahren)
- **Regula-falsi-Verfahren**[31] (als Spezialfall des Sekanten-Verfahrens)
- **Sekanten-Verfahren**
- **Newton-Verfahren**[32] (Tangenten-Verfahren) in IR
- **Modifizierte Newton-Verfahren** in IR bzw. IR^n, insbesondere das **Heron-Verfahren (Babylonisches Wurzelziehen)**

Zur Lösung **nichtlinearer Gleichungssysteme** wird das **Newton-Verfahren** in IR^n erläutert.

[31] *Regula falsi* (lat. „Regel des Falschen“) ist auch bekannt unter der von Fibonacci (1180-1241) verwendeten Bezeichnung *regula duarum falsarum positionum* (lat. „Regel vom zweifachen falschen Ansatz“) oder *lineares Eingabeln* (gemäß [WIKI] und [HERM, S. 199]).

[32] Auch bekannt unter dem Namen Newtonsches Näherungsverfahren oder Newton-Raphsonsche Methode, benannt nach Sir Isaac **Newton** (1643-1727) und Joseph Raphson (1648-1715) (gemäß [WIKI]).

4.1 Bisektionsverfahren

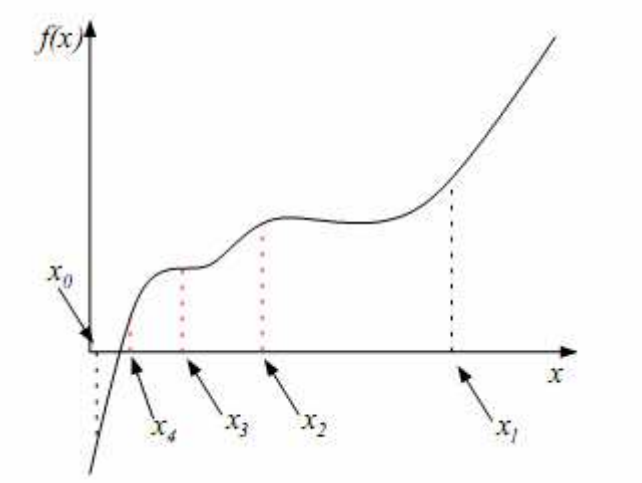


Abbildung 16: Prinzip der Bisektionsverfahren [MUNZ, S. 370]

(In Anlehnung an [MUNZ, S. 370-372])

Aus der Analysis kennt man den *Zwischenwertsatz*, der besagt, dass eine stetige Funktion, die an einer Stelle einen positiven und an einer anderen Stelle einen negativen Wert hat, dazwischen mindestens eine Nullstelle besitzt. Diese Tatsache kann man als Ausgangspunkt eines numerischen Verfahrens zum Auffinden von Nullstellen benutzen.

Gibt es zwei Punkte x_0, x_1 mit der Eigenschaft, dass

$$f(x_0)\,f(x_1) < 0\,,$$

d.h. die beiden zugehörigen Funktionswerte $f(x_0)$ und $f(x_1)$ haben unterschiedliche Vorzeichen, dann gibt es nach dem Zwischenwertsatz zwischen x_0 und x_1 mindestens eine Nullstelle von f .

Wir nehmen $x_0 < x_1$ an, halbieren das Intervall $[x_0, x_1]$, nehmen den Mittelpunkt des Intervalls

$$x_{neu} = \frac{x_0 + x_1}{2} \qquad (4.1)$$

und prüfen das Vorzeichen des Funktionswerts an dieser Stelle. Ist es das gleiche Vorzeichen wie das des Funktionswertes bei x_0, so ersetzen wir x_0 durch den neuen Punkt x_{neu}. Wir haben erneut ein Intervall, jetzt $[x_{neu}, x_1]$, welches die gleichen Eigenschaften wie das Ausgangsintervall $[x_0, x_1]$ hat. Es befindet sich somit eine Nullstelle in diesem Intervall.

Hat im anderen Fall die Funktion bei x_{neu} das gleiche Vorzeichen wie der Funktionswert bei x_1, so ersetzen wir x_1 durch den neuen Punkt. Das neue Intervall ist dann $[x_0, x_{neu}]$. Wir sehen, wie dieses Verfahren weiter läuft. Man fängt wieder von vorne an. Das betrachtete Intervall wird sukzessive halbiert[33] und die Hälften weiter untersucht, in welcher der Vorzeichenwechsel stattfindet. War der Funktionswert schon Null oder aber sein Betrag so klein, dass er uns als numerische Näherung genügt, dann sind wir bereits fertig.

Die einzelnen Schritte lassen sich im folgenden Algorithmus formulieren:

- Berechne $x_{neu} = \frac{x_0 + x_1}{2}$.
- Falls $|f(x_{neu})|$ klein genug ist, brich ab und verwende x_{neu} als numerische Näherung der Nullstelle.
- Sonst:
  - Falls $f(x_0) f(x_{neu}) < 0$, ersetze x_1 durch x_{neu} ($x_1 := x_{neu}$).
  - Andernfalls ersetze x_0 durch x_{neu} ($x_0 := x_{neu}$).
  - Beginne von vorne.

In Abbildung 16 sind die Punkte ihrer Entstehung nach nummeriert.

Anmerkung:

Haben beim Start des Verfahrens die Funktionswerte an den Intervallgrenzen x_0 und x_1 die gleichen Vorzeichen, so muss man das Intervall entsprechend vergrößern bis unterschiedliche Vorzeichen auftreten.

[33] Daher nennt man das Bisektionsverfahren auch **Intervallhalbierungsverfahren**. Anmerkung: Das Wort Bisektion setzt sich zusammen aus *Bi* „Zwei“ und *Sektion* „Schnitt“. Es steht also für Zwei-Teilung. (In Anlehnung an [WIKI])

4.2 Regula-falsi-Verfahren

(In Anlehnung an [MUNZ, S. 372-373])

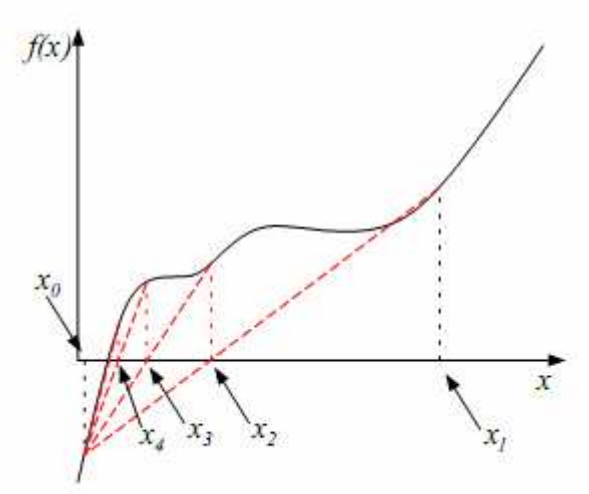


Abbildung 17: Prinzip des Regula-Falsi-Verfahrens [MUNZ, S. 372]

Schon Adam Ries(e) kannte eine Verbesserung des Bisektionsverfahrens. Die Wahl des neuen Punktes in der Mitte erscheint oft nicht optimal gewählt, und das Verfahren kann selbst bei sehr einfachen Funktionen (wie z.B. linearen) sehr lange dauern. Da liegt es nahe, statt der Halbierung eine Gerade durch die Punkte $(x_0, f(x_0))$ und $(x_1, f(x_1))$ zu legen. Dort, wo diese die x–Achse schneidet, wählt man den neuen Wert x_{neu}. Damit hat man bei linearen Funktionen in einem Schritt die exakte Lösung. Aber auch bei nichtlinearen Funktionen sollte dieses Verfahren schneller zum Ziel kommen. In Abbildung 17 zeigt sich dies an unserem Beispiel deutlich.

Nun benötigen wir noch die zugehörige Formel, nach der sich x_{neu} berechnen lässt. Dazu setzen wir eine allgemeine lineare Funktion an, die in x_0 den Wert $f(x_0)$ hat, in x_1 den Wert $f(x_1)$ und in x_{neu} verschwindet. Nach dem Lagrangeschen Interpolationspolynom[34] lautet die Geradengleichung

$$p(x) = f(x_0)\frac{x - x_1}{x_0 - x_1} + f(x_1)\frac{x - x_0}{x_1 - x_0}.$$

[34] Details zur Lagrange-Interpolation sind in [SCHRA, S.98-101] nachzulesen.

Setzen wir $x = x_{neu}$ in diese Gerade ein, ergibt sich als Funktionswert Null. Löst man weiter nach x_{neu} auf, so ergibt sich die gewünschte Formel:[35]

$$x_{neu} = \frac{x_0 f(x_1) - x_1 f(x_0)}{f(x_1) - f(x_0)} \tag{4.2}$$

Der Algorithmus für das Regula-falsi-Verfahren lautet also:

- Berechne x_{neu} nach Formel (4.2).
- Falls $|f(x_{neu})|$ klein genug ist, brich ab und verwende x_{neu} als numerische Näherung der Nullstelle.
- Sonst:
  - Falls $f(x_0) f(x_{neu}) < 0$, ersetze x_1 durch x_{neu} ($x_1 := x_{neu}$).
  - Andernfalls ersetze x_0 durch x_{neu} ($x_0 := x_{neu}$).
  - Beginne von vorne.

In Abbildung 17 sind die Punkte in der Reihenfolge ihrer Entstehung nummeriert.

[35] $0 = f(x_0)\frac{x_{neu} - x_1}{x_0 - x_1} + f(x_1)\frac{x_{neu} - x_0}{x_1 - x_0} \quad / \cdot (x_0 - x_1)$

$0 = f(x_0) \cdot (x_{neu} - x_1) - f(x_1) \cdot (x_{neu} - x_0)$

$- f(x_0) \cdot x_{neu} + f(x_1) \cdot x_{neu} = -f(x_0) \cdot x_1 + f(x_1) \cdot x_0$

$x_{neu} = \frac{-f(x_0) \cdot x_1 + f(x_1) \cdot x_0}{-f(x_0) + f(x_1)}$

4.3 Sekanten-Verfahren

(In Anlehnung an [MUNZ, S. 373-374])

War das Regula-falsi-Verfahren eine Variation des Bisektionsverfahrens, so ist das Sekanten-Verfahren eine Variation des Regula-falsi-Verfahrens. Es ergibt sich, wenn man sich von Anfang an nicht darum kümmert, ob die jeweiligen zwei Punkte die Nullstelle einschließen oder nicht. Man verwirft einfach den ältesten Punkt, nachdem man den neuen Punkt ausgerechnet hat. Sind x_0 und x_1 gegeben, so lautet die Iterationsvorschrift:

$$x_{k+1} = \frac{x_{k-1} f(x_k) - x_k f(x_{k-1})}{f(x_k) - f(x_{k-1})} \qquad k = 1, 2, \ldots. \tag{4.3}$$

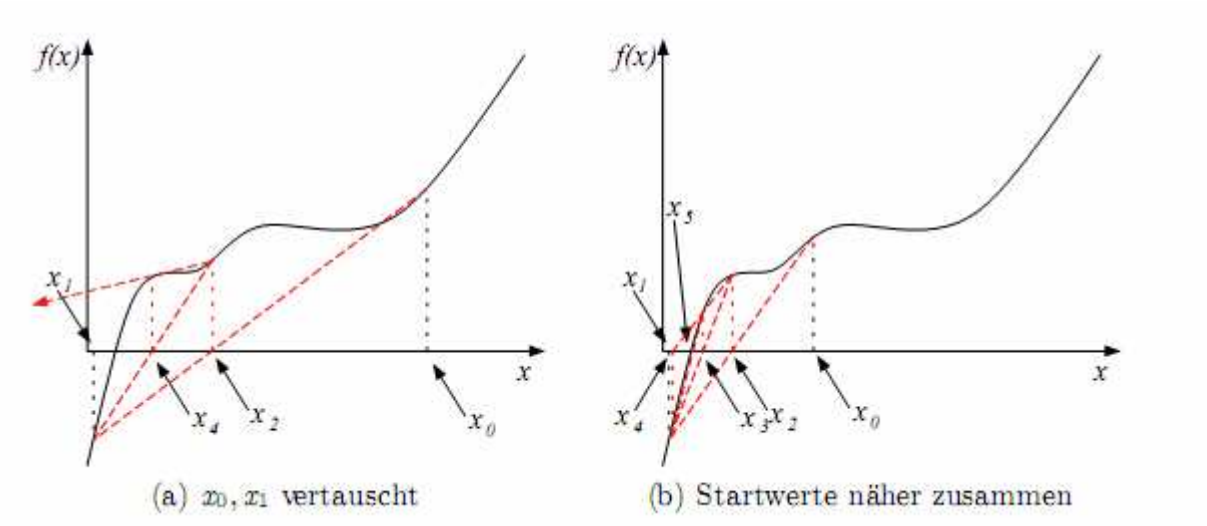


Abbildung 18: Prinzip des Sekanten-Verfahrens [MUNZ, S. 374]

Abbildung 18 zeigt, wo die Stärken und Schwächen liegen. Werden die Startwerte wie im Beispiel für das Bisektionsverfahren und das Regula-falsi-Verfahren gewählt, führt das Sekanten-Verfahren sehr schnell aus dem Definitionsbereich von f heraus. Auch das Vertauschen der beiden Startwerte (Fall (a)) kann dieses Problem nicht lösen. Sind dagegen die Startwerte wie in Fall (b) näher beieinander, so kann man sehr schnelle Konvergenz erzielen.

Möchte man dieses Verfahren zur Approximation einer Nullstelle verwenden, so muss man sicherstellen, dass das Verfahren abbricht, sobald man in eine Situation wie in Fall (a) kommt, und mit zwei anderen Werten neu startet. Diese kann man durch ein paar Schritte des Bisektionsverfahrens erhalten.

Eine wichtige Anwendung des Sekanten-Verfahrens ist die Suche nach einer Anfangseinschließung für das Bisektionsverfahren oder das Regula-falsi-Verfahren. Hat man nämlich am Anfang zwei Punkte, deren Funktionswerte die gleichen Vorzeichen haben, so kann man einige Schritte des Sekanten-Verfahrens durchführen, bis man auf einen Punkt kommt, dessen Funktionswert das umgekehrte Vorzeichen besitzt.

Fazit:

Beim Sekanten-Verfahren werden jeweils zwei x-Werte zur Berechnung benötigt. Im Regula-falsi-Verfahren erfolgt die Auswahl dieser Werte aus den vorangegangenen derart, dass die zugehörigen Funktionswerte verschiedene Vorzeichen haben ($f(x_k) \cdot f(x_{k-1}) < 0$, d. h. die gesuchte Nullstelle wird immer von x_k und x_{k-1} eingeschlossen). Dadurch ist die Konvergenz des Regula-falsi-Verfahrens stets garantiert. Der Vorteil des Regula-falsi-Verfahrens im Vergleich zum Sekanten-Verfahren liegt in diesem schrittweisen Einschluss der gesuchten Nullstelle. Der Nachteil besteht darin, dass man eine zusätzlich Abfrage ($f(x_k) \cdot f(x_{k-1}) < 0$) in jedem Iterationsschritt investieren muss. (In Anlehnung an [DORN, S. 12] und [LENZ, S. 45])

4.4 Newton-Verfahren (Tangenten-Verfahren) in IR

Gemäß [HERM, S. 186] ist das Newton-Verfahren eine der bekanntesten Techniken zur Lösung nichtlinearer Gleichungen. Seine Herleitung kann auf mindestens drei unterschiedlichen Wegen erfolgen, die in der Literatur auch tatsächlich beschritten werden:

- *geometrisch* als Tangentenmethode,
- *analytisch* als spezielles Taylorpolynom oder
- *rechentechnisch* als einfaches Verfahren zur Beschleunigung der Konvergenz der Fixpunkt-Iteration.

Eine detaillierte Ausarbeitung dieser drei Wege ist in [HERM, S. 186-191] nachzulesen.

(In Anlehnung an [MUNZ, S. 374-375] und [DORN, S. 11-15])

Für differenzierbare Funktionen kann man aus dem Sekanten-Verfahren eine Methode herleiten, die oft noch um einiges schneller konvergiert: Das Newton-Verfahren. Die Iterationsvorschrift des Sekanten-Verfahrens (4.3) lässt sich umschreiben in

$$x_{k+1} = x_k - f(x_k)\frac{x_k - x_{k-1}}{f(x_k) - f(x_{k-1})} \qquad \text{k= 1, 2, 3, ...} \quad [36] \qquad (4.4)$$

Für differenzierbares f kann man hier den Grenzübergang $x_{k-1} \to x_k$ durchführen. Der Bruch auf der rechten Seite in (4.4) ist gerade der Kehrwert des

[36] Genauer formuliert: Das Sekanten-Verfahren besitzt die Iterationsfunktion $\varphi(x) = x - f(x) \cdot \frac{1}{\frac{\Delta f}{\Delta x}}$ mit der Iterationsvorschrift

$$x_{k+1} = \varphi(x_k) = x_k - f(x) \cdot \frac{1}{\frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}}} = x_k - \frac{x_k - x_{k-1}}{f(x_k) - f(x_{k-1})} \cdot f(x_k) \qquad \text{k = 1, 2, 3, ...}$$

Differenzenquotienten[37]. Durch den Grenzübergang geht er demnach in den Kehrwert des *Differentialquotienten*[38] über. Damit ergibt sich als Iterationsvorschrift für das Newton-Verfahren:

$$x_{k+1} = x_k - f(x_k)\frac{1}{f'(x_k)} = x_k - \frac{f(x_k)}{f'(x_k)} \qquad \text{k= 0, 1, 2, ...} \qquad [39] \qquad (4.5)$$

Man sucht nun nicht mehr nach dem Schnittpunkt der *Sekante* durch die Punkte $(x_{k-1}, f(x_{k-1}))$ und $(x_k, f(x_k))$ mit der x-Achse (im Fall des *Sekanten*-Verfahrens), sondern nach dem Schnittpunkt der *Tangente* durch den Punkt $(x_k, f(x_k))$ mit der x–Achse (im Fall des Newton- bzw. *Tangenten*-Verfahrens). (In Anlehnung an [BÄR, S. 156])

Anmerkung: (In Anlehnung an [KNORR, S. 33-34])

> Ein Grundprinzip vieler numerischer Verfahren für nichtlineare Probleme ist die sogenannte „*Linearisierung*“ von Funktionen. Dies bedeutet, dass die zu untersuchende Funktion durch eine lineare Funktion (plus eine additive Konstante d) ersetzt wird, also durch eine Funktion, deren Graph eine Gerade ist. Mit dieser Ersatzfunktion lässt es sich natürlich viel einfacher rechnen. Man löst das zu Grunde liegende Problem also für die Ersatzfunktion und hofft, dass diese Lösung der Lösung der ursprünglichen Problemstellung nahe kommt. Häufig benutzt man auch mehrere Ersatzfunktionen dieses Typs, um die erzielten Näherungen noch zu verbessern.

---

[37] bzw. der Sekantensteigung $\frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}}$, daher hat sich die Bezeichnung **Sekanten-Verfahren** ergeben.

[38] bzw. der Tangentensteigung $f'(x_k)$, daher heißt das Newton-Verfahren auch **Tangenten-Verfahren.**

[39] Genauer formuliert: Das Newton-Verfahren besitzt die Iterationsfunktion
$\varphi(x) = x - f(x) \cdot \frac{1}{f'(x)} = x - \frac{f(x)}{f'(x)}$ mit der Iterationsvorschrift
$x_{k+1} = \varphi(x_k) = x_k - f(x_k)\frac{1}{f'(x_k)} = x_k - \frac{f(x_k)}{f'(x_k)}$ k = 0, 1, 2, … .

Rechnerisch bedeutet die Linearisierung im Fall des Tangenten-Verfahrens

$$f(x) \approx \underbrace{f'(x_0)}_{\text{k...Steigung derTangente}} \cdot x + \underbrace{f(x_0) - f'(x_0) \cdot x_0}_{\text{d...Abstand der Tangente bez. der y-Achse}} = \underbrace{f(x_0) + f'(x_0)(x - x_0) = y}_{(*)},$$

d. h. die Funktion f wird durch die Geradengleichung (*) angenährt. Bei dieser Gerade handelt es sich um die Tangente an den Graph der Funktion im Punkt. Wenn x nahe bei x_0 liegt, ist die Abweichung der Funktion f von ihrer Tangente klein. Sie wird in der Regel umso größer, je weiter x von x_0 entfernt ist.

Wenn man die Tangente mit der x-Achse schneidet, also $y = 0$ setzt, ergibt sich $x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$. Dies führt auf die Iterationsvorschrift (4.5) des Newton-Verfahrens.

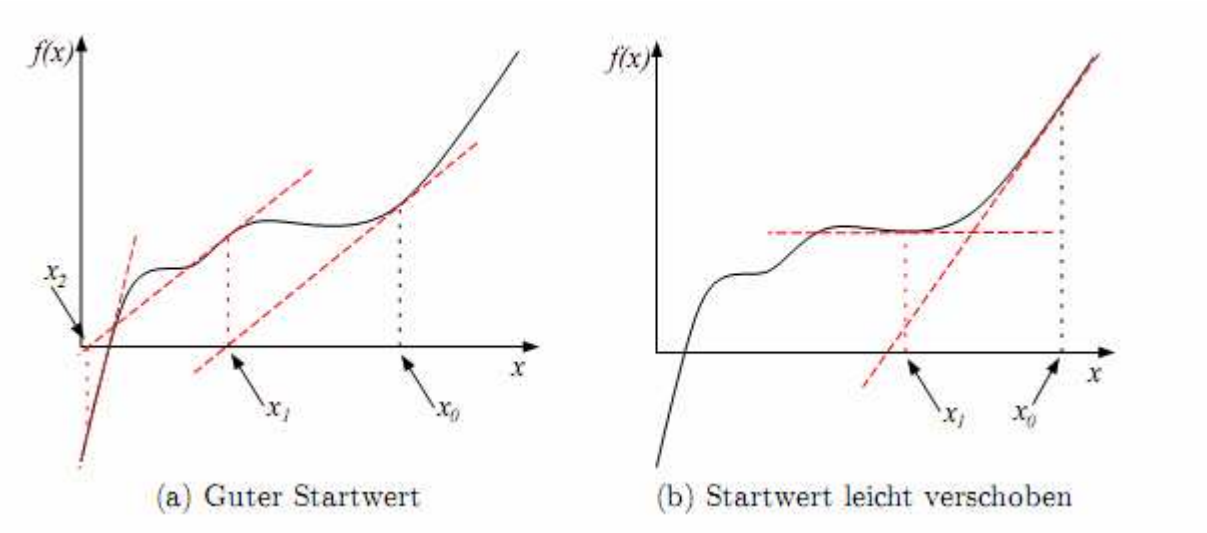


Abbildung 19: Prinzip des Newton-Verfahrens [MUNZ, S. 375]

In Graphik (b) in Abbildung 19 findet man ein Beispiel dafür, dass dies nicht immer funktionieren muss. Wird die Ableitung Null, so ist die Tangente parallel zur x-Achse und schneidet sie nicht. In der Formel (4.5) drückt sich das darin aus, dass der Nenner des Bruchs Null wird. Hier versagt das Verfahren und bricht ab. Hat man jedoch wie in Graphik (a) in Abbildung 19 einen guten Startwert erwischt, so bekommt man eine besonders schnelle Konvergenz. Man muss also auch hier Abfragen einbauen, die kontrollieren, ob der Startwert gut war. Man kann z. B. dann,

wenn der Betrag des Funktionswertes der Iterierten $|f(x_k)|$ steigt statt zu fallen, für ein paar Schritte auf das Regula-falsi-Verfahren übergehen und mit dem dort erreichten Wert wieder in die Newton-Iteration einsteigen.

4.5 Newton-Verfahren in IR^n

Das in Kapitel 4.4 vorgestellte Newton-Verfahren betrachtet Funktionen in IR (eindimensionaler Fall). In diesem Kapitel soll die Verallgemeinerung des Newton-Verfahrens auf höherdimensionale Räume (IR^n) kurz erläutert werden.

(In Anlehnung an [KNORR, S. 65-68])

Das Newton-Verfahren zur näherungsweisen Bestimmung von Nullstellen von $f : IR \to IR$ lautet (4.5). Es entstand aus der Linearisierung von f an der Stelle x_k: $f(x) \approx f(x_k) + f'(x_k)(x - x_k)$.

Eine entsprechende Linearisierung ist ebenfalls für $F : IR^n \to IR^n$ möglich. Wir bezeichnen im Folgenden den schon berechneten Näherungswert mit $\vec{x}^{(k)}$. In der Linearisierung muss die Ableitung $f'(x_k)$ durch die sogenannte *Jacobi-Matrix*[40] von F an der Stelle $\vec{x}^{(k)}$ ersetzt werden. Die Jacobi-Matrix ist die Matrix der partiellen Ableitungen von F, sie hat die allgemeine Form

$$DF(\vec{x}) = F'(\vec{x}) = J := \begin{pmatrix} \frac{\partial F_1}{\partial x_1}(\vec{x}) & \frac{\partial F_1}{\partial x_2}(\vec{x}) & \cdots & \frac{\partial F_1}{\partial x_n}(\vec{x}) \\ \frac{\partial F_2}{\partial x_1}(\vec{x}) & \frac{\partial F_2}{\partial x_2}(\vec{x}) & \cdots & \frac{\partial F_2}{\partial x_n}(\vec{x}) \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial F_n}{\partial x_1}(\vec{x}) & \frac{\partial F_n}{\partial x_2}(\vec{x}) & \cdots & \frac{\partial F_n}{\partial x_n}(\vec{x}) \end{pmatrix},$$

wobei $F(\vec{x}) = F(x_1, x_2, \cdots, x_n) = \begin{pmatrix} F_1(x_1, x_2, \cdots, x_n) \\ \vdots \\ F_n(x_1, x_2, \cdots, x_n) \end{pmatrix}$ (gemäß [HUCK, S. 212]).

Beispiel: Im Fall von $F(x_1, x_2) = \begin{pmatrix} 2x_1 + 4x_2 \\ 4x_1 + 8x_2^3 \end{pmatrix}$ wäre $DF(x_1, x_2) = \begin{pmatrix} 2 & 4 \\ 4 & 24x_2^2 \end{pmatrix}$.

[40] Benannt nach dem deutschen Mathematiker Carl Gustav Jacob Jacobi (1804-1851) (gemäß [WIKI]).

Unter Benutzung der Jacobi-Matrix können wir dann F an der Stelle $\vec{x}^{(k)}$ linearisieren: $F(\vec{x}) \approx F(\vec{x}^{(k)}) + F'(\vec{x}^{(k)})(\vec{x} - \vec{x}^{(k)})$. Wir suchen nun eine Nullstelle $\vec{x}$ der rechten Seite, d. h. $F(\vec{x}^{(k)}) + F'(\vec{x}^{(k)})(\vec{x} - \vec{x}^{(k)}) = 0$. Die Idee ist, dass dieser Vektor $\vec{x}$ eine genauere Näherung für die exakte Nullstelle von F ist als $\vec{x}^{(k)}$. Die Iterationsvorschrift lautet demnach:

$$\vec{x}^{(k+1)} = \vec{x}^{(k)} - (F'(\vec{x}^{(k)}))^{-1} F(\vec{x}^{(k)}).$$

Das Newton-Verfahren konvergiert quadratisch, wenn der Startvektor nahe genug an einer Nullstelle $\vec{x}$ liegt, $F'(\vec{x})$ regulär und F dreimal stetig differenzierbar ist.
Beweis: In [PLATO, S. 88-89] nachlesbar.

Das Newton-Verfahren kann als ein *Minimierungsverfahren* der Euklidischen Norm[41] von $F(\vec{x})$ angesehen werden. Daher kann es passieren, dass anstelle einer Nullstelle von F ein lokales Minimum $\vec{x}_{\min}$ der Norm von F gefunden wird, die aber nicht 0 ist. In diesem Fall ist stets $F'(x_{\min})$ nicht regulär.

Eine detaillierte Ausführung zum Newton-Verfahren in IR^n zur **Berechnung von Nullstellen** sowie zum Newton-Verfahren in IR^n zur **Berechnung eines Minimums** ist in [HUCK, S. 212-214] nachzulesen.

Zum Newton-Verfahren in IR^2 sind konkrete Beispiele mit Angabe der Lösungsschritte in folgenden Literaturquellen nachzulesen:

- [KNORR, S. 66-67]
- [AUZI_2008, S.40]
- [SANNS, S. 30-34] mit Visualisierung mittels des CAS Mathematica
- [KNORR, S.67-68] verwendet das Newton-Verfahren zur Berechnung eines Minimums

[41] Euklidische Norm (Länge) $\|.\|_2$ eines Vektors $\vec{x} \in IR^n$: $\|\vec{x}\|_2 := \sqrt{\sum_{i=1}^{n} x_i^2}$ (gemäß [SCHRA, S. 42]).

4.6 Einige modifizierte Newton-Verfahren in IR bzw. IR^n

In der Literatur finden sich diverse modifizierte Newton-Verfahren. Einige werden nun kurz erläutert, wobei das Heron-Verfahren in einem eigenen Kapitel detaillierter behandelt wird.

Anzumerken ist, dass es das Newton-Verfahren nicht nur für Anwendungen in IR (eindimensionaler Fall), sondern auch in IR^n (mehrdimensionaler Fall) gibt - Details dazu finden sich in Kapitel 4.5. Selbiges gilt auch für die hier angeführten Modifikationen.

(In Anlehnung an [KNORR, S. 34-35])

Das Newton-Verfahren ist ein sehr beliebtes und schnelles Verfahren. Es hat aber den Nachteil, dass man in jedem Schritt eine Ableitung berechnen muss. Wenn man also die Nullstellen einer Funktion sucht, deren Ableitung man nicht kennt, ist das Newton-Verfahren nicht anwendbar. Dann kann man zu verschiedenen Varianten/Modifikationen greifen. (In Anlehnung an [KNORR, S. 34])

- Vereinfachtes Newton-Verfahren

Statt in jedem Schritt $f'(x_k)$ auszurechnen, kann man immer wieder $f'(x_0)$ verwenden. Das damit entstandene Verfahren

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_0)} \qquad k = 0, 1, 2, \ldots$$

heißt *vereinfachtes Newton-Verfahren*. Man kann erwarten, dass es nicht so gut wie das Original-Newton-Verfahren ist, was praktisch bedeutet, dass die damit erzeugten Werte i. A. nicht so schnell gegen eine Nullstelle von f konvergieren. (In Anlehnung an [KNORR, S. 34] und [LENZ, S. 37])

- Gedämpftes Newton-Verfahren

Die Motivation für diese Modifikation des Newton-Verfahren sowie theoretische Grundlagen sind in [SCHRA, S. 90-91] nachzulesen. Der grundlegende Idee besteht darin, den Newton-Schritt mit Hilfe des Faktors λ zu „verkürzen/dämpfen“:

$$x_{k+1} = x_k - \lambda \cdot \frac{f(x_k)}{f'(x_k)} \qquad 0 < \lambda < 1 \qquad \text{k = 0, 1, 2, …}$$

Durch diese Strategie kann man den Konvergenzbereich des Newton-Verfahrens wesentlich vergrößern.

- Sekanten-Verfahren (gehört zur Klasse der Quasi-Newton-Verfahren[42])

Das Sekanten-Verfahren hat gegenüber dem Newton-Verfahren den Vorteil, dass keine Ableitungen (Tangentensteigungen) benötigt werden. Dafür braucht es zwei Startwerte, was aber keine wesentliche Einschränkung darstellt. In jedem Schritt wird eine Funktionsauswertung benötigt (nicht deren zwei, denn $f(x_{k-1})$ hat man schon im vorigen Schritt berechnet). Auch wenn das Sekanten-Verfahren nicht so schnell wie das Newton-Verfahren konvergiert, ist es mit geringerem Aufwand durchzuführen und daher konkurrenzfähig. (In Anlehnung an [KNORR, S. 34-35])

[42] Das Sekantenverfahren in IR oder das Broyden-Verfahren in IR^n gehören zur Klasse der Quasi-Newton-Verfahren, die keine Ableitungswerte von f benötigen, also ableitungsfreie Iterationsverfahren darstellen (In Anlehnung an [AUZI_2007, S. 184-196]). Eine detaillierte Beschreibung des Broyden-Verfahrens findet sich in [HERM, S. 256-261].

4.6.1 Heron-Verfahren (Babylonisches Wurzelziehen)

(In Anlehnung an [WIKI] und [DORN, S. 14])

Das Heron-Verfahren[43] ist ein Verfahren zur Berechnung einer Näherung der Quadratwurzel einer Zahl. Es ist ein Spezialfall des Newton-Verfahrens. Die Iterationsvorschrift lautet:

$$x_{k+1} = \frac{x_k + \frac{a}{x_k}}{2} \qquad \text{k = 0, 1, 2, …} \qquad a > 0 \qquad x_0 > 0 \qquad (4.6)$$

Hierbei steht $a > 0$ für die Zahl, deren Quadratwurzel bestimmt werden soll. Der Startwert x_0 der Iteration kann beliebig festgesetzt werden, solange er positiv ist.

Beweis:
Historisch gesehen ist das Heron-Verfahren wesentlich älter als das Newton-Verfahren, kann aber formal sehr elegant auf dieses zurückgeführt werden. In diesem Sinne besteht das Heron-Verfahren einfach aus der Anwendung des Newton-Verfahrens zur Nullstellenbestimmung der Funktion, genauer des quadratischen Polynoms $f : IR \to IR$ mit $f(x) := x^2 - a$ und $a > 0$. Für die Suche nach der Nullstelle von f ergibt sich mit dem Newton-Verfahren (4.5) wegen $f'(x) = 2x$ bei beliebig vorgegebenem $x_0 > 0$ nämlich das folgende algorithmische Vorgehen:

$$x_{k+1} := x_k - \frac{f(x_k)}{f'(x_k)} = x_k - \frac{x_k^2 - a}{2x_k} = x_k - \frac{1}{2}x_k + \frac{a}{2x_k} = \frac{x_k + \frac{a}{x_k}}{2} \qquad \text{k = 0, 1, 2, ….}$$

(In Anlehnung an [LENZ, S. 37, 38] und [LOCH S.333]). Eine detaillierte Darstellung bzw. Beweisführung wird in [LOCH, S.333-338] gegeben.

[43] Benannt nach **Heron** von Alexandria (um 100). Das Verfahren ist auch unter der Bezeichnung **Babylonisches Wurzelziehen** bekannt, da es bereits vor mehr als 3000 Jahren von den Babyloniern verwendet wurde. (gemäß [LENZ, S. 37] und [DORN, S. 14])

- Beispiel

Im Folgenden wird ein einfaches Beispiel für die Wurzel aus 9 und die Annäherung nach vier Berechnungsschritten an den wahren Wert $\sqrt{9} = 3$ angeführt:

$a = 9$ und $x_0 = 9$

$$x_1 = \frac{9 + \frac{9}{9}}{2} = \frac{10}{2} = 5$$

$$x_2 = \frac{5 + \frac{9}{5}}{2} = \frac{\frac{34}{5}}{2} = \frac{34}{10} = 3{,}4$$

$$x_3 = \frac{\frac{34}{10} + \frac{9}{\frac{34}{10}}}{2} = \frac{\frac{34}{10} + \frac{90}{34}}{2} = \frac{257}{85} \approx 3{,}024$$

$$x_4 = \frac{\frac{257}{85} + \frac{9}{\frac{257}{85}}}{2} = \frac{\frac{257}{85} + \frac{765}{257}}{2} = \frac{65537}{21854} \approx 3{,}00009$$

- Geometrische Veranschaulichung des Heron-Verfahrens

Dem Heron-Verfahren liegt die Idee zu Grunde, dass ein Quadrat mit Flächeninhalt A eine Seitenlänge von $\sqrt{A}$ hat. Ausgangspunkt des Verfahrens ist ein beliebiges Rechteck mit Flächeninhalt A. Schritt für Schritt wird das Seitenverhältnis des Rechtecks so geändert, dass sich seine Form immer mehr der eines Quadrats annähert, während der Flächeninhalt gleich bleibt. Die Seitenlängen des Rechtecks sind die Näherungswerte für $\sqrt{A}$.

Im ersten Schritt wird eine beliebige Seitenlänge x_0 für das Rechteck gewählt. Damit dieses den gewünschten Flächeninhalt hat, wird die zweite Seitenlänge mit der Formel

$$y_0 = \frac{A}{x_0}$$

berechnet. Als Beispiel soll die Wurzel aus 9 berechnet werden. Für die eine Seitenlänge wird der Wert 9 (= x_0) gewählt, sodass sich die andere Seitenlänge zu 1 (= y_0) berechnet. Das erste Rechteck hat deshalb die folgende Form:

Abbildung 20 (gemäß [WIKI])

Die Ähnlichkeit dieses Rechteck mit einem Quadrat ist gering. Das kommt auch dadurch zum Ausdruck, dass die Seitenlängen 1 und 9 sehr schlechte Näherungen für $\sqrt{9} = 3$ sind.

Um eine bessere Annäherung an ein Quadrat zu erhalten, muss die lange Seite gekürzt und die kurze Seite verlängert werden. Als neue Länge der langen Seite wird der Mittelwert

$$x_1 = \frac{x_0 + y_0}{2}$$

der beiden bisherigen Seitenlängen genommen. Die Länge der anderen Seite berechnet sich wie oben zu

$$y_1 = \frac{A}{x_1}.$$

Im Beispiel ergibt sich als Mittelwert die Seitenlänge 5 (= x_1). Die dazugehörige kurze Seite hat eine Länge von $\frac{9}{5} = 1{,}8$ (= y_1).

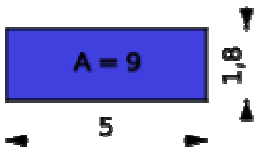


Abbildung 21 (gemäß [WIKI])

Auch hier ist die Ähnlichkeit zu einem Quadrat noch gering. Allerdings ist das neue Rechteck im Vergleich zum vorhergehenden „quadratischer“.

Der beschriebene Ablauf wird in jedem weiteren Schritt des Heron-Verfahrens wiederholt. Der Mittelwert der Seitenlängen eines Rechtecks entspricht der Länge der langen Seite des neuen Rechtecks und die Länge der kurzen Seite lässt sich daraus jeweils wie oben beschrieben berechnen. Im Beispiel entstehen so in den nächsten zwei Schritten die folgenden beiden Rechtecke.

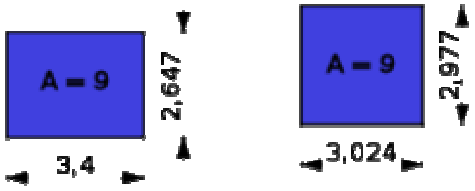


Abbildung 22 (gemäß [WIKI])

Das letzte Rechteck ist schon annähernd quadratisch. Die Seitenlänge 3,024 ($\approx x_3$) liegt entsprechend nahe bei 3, dem exakten Wert von $\sqrt{9}$.

- Verallgemeinerung des Verfahrens

Das Verfahren (4.6) lässt sich verallgemeinern, sodass die n. Wurzel von a berechnet wird. Je größer n ist, desto mehr Schritte werden benötigt, um die Wurzel genau zu berechnen. Dazu wird die n-dimensionale Entsprechung der oben genannten geometrischen Herleitung benutzt.

$$x_{k+1} = \frac{(n-1)x_k^n + a}{n\,x_k^{n-1}} \qquad \text{k = 0, 1, 2, …} \qquad a > 0 \qquad x_0 > 0 \qquad (4.7)$$

- Konvergenz

Beim Heron-Verfahren (4.6) gelten folgende Beziehungen:

$$x_k \geq x_{k+1} \geq \sqrt{a} \qquad \text{k = 0, 1, 2, …,}$$

d. h. der Näherungswert geht von oben gegen die gesuchte Wurzel, und

$$\lim_{k\to\infty} x_k = \sqrt{a}\ ,$$

d. h. das Verfahren konvergiert gegen die Wurzel.

Beweis: (In Anlehnung an [DORN, S.14-15])

Bereits vor mehr als 3000 Jahren verwendeten die Babylonier zur Berechnung von $\sqrt{a}$ die Iteration $x_{k+1} = \dfrac{x_k + \dfrac{a}{x_k}}{2} = \varphi(x_k)$ mit k = 0, 1, 2, … und $a > 0$. Man zeige, dass diese Iteration für jeden beliebigen Startwert $x_0 > 0$ auf eine monoton fallende und konvergente Folge $(x_k)_{k\in IN}$ mit $\lim_{k\to\infty} x_k = \sqrt{a}$ führt.

a und x_0 sind positiv. Durch wiederholtes Einsetzen in φ können keine negativen Zahlen herauskommen. Daher sind alle hier auftretenden Größen stets positiv.

(1) z. z.: $\varphi(x_k) \geq \sqrt{a}$

$(x_k - \sqrt{a})^2 \geq 0$

$x_k^2 - 2x_k\sqrt{a} + a \geq 0$

$x_k^2 + a \geq 2x_k\sqrt{a} \quad / : x_k > 0$

$x_k + \dfrac{a}{x_k} \geq 2\sqrt{a}$

$\dfrac{x_k + \dfrac{a}{x_k}}{2} \geq \sqrt{a}$

$\varphi(x) \geq \sqrt{a}$

Das bedeutet, dass $x_{k+1} \geq \sqrt{a}$ für k = 0, 1, 2, … .

(2) z. z.: $x_{k+1} \leq x_k$ für k = 1, 2, 3, …

Für k = 1, 2, 3, … gilt:

$x_k \geq \sqrt{a}$ laut (1)

$x_k^2 \geq a$

$$x_{k+1} = \varphi(x_k) = \frac{x_k + \frac{a}{x_k}}{2} \leq \frac{x_k + \frac{x_k^2}{x_k}}{2} = x_k$$

Das bedeutet, dass x_k für k = 1, 2, 3, … monoton fallend ist.

(3) Demnach ist $x_1, x_2, \ldots$ eine monoton fallende mit $\sqrt{a}$ nach unten beschränkte Folge. Daraus folgt, dass sie auch konvergent ist. Der Grenzwert sei c.

$$x_{k+1} = \frac{x_k + \frac{a}{x_k}}{2}$$

$$\lim_{k\to\infty} x_{k+1} = \lim_{k\to\infty} \frac{x_k + \frac{a}{x_k}}{2}$$

$$c = \frac{c + \frac{a}{c}}{2} \quad /\cdot 2c$$

$$2c^2 = c^2 + a$$

$$c^2 = a$$

$$c = \pm\sqrt{a}$$

Da eine positive Folge keinen negativen Grenzwert haben kann folgt: $\lim_{k\to\infty} x_k = c = \sqrt{a}$.

□

4.7 Die Verfahren im Vergleich

4.7.1 Entscheidungshilfen

(In Anlehnung an [MUNZ, S. 376])

Das **Newton-Verfahren** ist sicherlich ein schnelles Verfahren. Allerdings muss man es, wie oben erwähnt, oft mit einem anderen Verfahren kombinieren, weil man einem gegebenen Startwert nicht direkt ansehen kann, ob er zur Konvergenz führt oder nicht. Hierbei ist anzumerken, dass das Auffinden eines Startwertes oft nicht leicht ist. Ist die Funktion nicht differenzierbar oder die Ableitung nicht durch eine geeignete Formel gegeben, so ist das Verfahren nicht anwendbar. Manchmal ist die Berechnung der Ableitung auch so aufwändig, dass das die schnellere Konvergenz im Vergleich zum Sekantenverfahren wieder aufwiegt. In einem solchen Fall wird auch ein **modifiziertes Newton-Verfahren** (siehe *vereinfachtes Newton-Verfahren* in Kapitel 4.6) angewandt, bei dem die Ableitung nicht in jedem Iterationsschritt neu berechnet wird.

Ein Vorteil des **Sekanten-Verfahrens** gegenüber dem Newton-Verfahren ist der wesentlich geringere Rechenaufwand für einen Iterationsschritt. Das Problem, dass man den Startwerten die Konvergenz oder Divergenz nicht direkt ansehen kann, teilt es allerdings mit diesem. So muss man beide Verfahren oft in Kombination mit dem Bisektionsverfahren oder Regula-falsi-Verfahren benutzen.

Bisektionsverfahren und **Regula-falsi-Verfahren** garantieren, dass das Verfahren konvergiert, sobald man eine Einschließung einer Nullstelle durch Punkte mit Funktionswerten unterschiedlichen Vorzeichens hat. Allerdings erkauft man sich diese Sicherheit durch eine – vor allem beim Bisektionsverfahren – langsamere Konvergenz. Oft ist diese aber ausreichend, insbesondere dann, wenn keine allzu hohe Genauigkeit gefordert wird. Für die Anfangseinschließung brauchen Bisektionsverfahren und Regula-falsi-Verfahren oft ein paar Schritte des Sekanten-Verfahrens.

4.7.2 Konvergenz

Das **Bisektionsverfahren** ist stets konvergent (man spricht von globaler Konvergenz). Allerdings ist dafür die Konvergenzgeschwindigkeit nur linear (p=1) (In Anlehnung an [HUCK, S. 209]).
Beweis: In [AUZI_2007, S. 184] nachzulesen.

Das **Regula-falsi-Verfahren** besitzt die Konvergenzordnung p=1 (gemäß [NEUN, S. 298).
Beweis: In [LOCH, S. 322] nachzulesen.

(In Anlehnung an [KNORR, S. 36])

Für *einfache Nullstellen* von f konvergiert

- das **Sekanten-Verfahren** mit der Ordnung des Goldenen Schnitts $p = \frac{1+\sqrt{5}}{2} = 1{,}618...$ [44]
  (Beweis: In [CAR, S. 175-177] nachzulesen.)
- das **vereinfachte Newton-Verfahren** linear (p=1)
  (Beweis: In [HEU, S. 412-416] nachzulesen.)
- das **Newton-Verfahren** quadratisch (p=2)
  (Beweis: In [HUCK, S. 206-207] oder [PLATO, S. 83-84] nachzulesen.)

Im Falle einer *mehrfachen Nullstelle* von f konvergiert das **Newton-Verfahren** nur noch linear (p=1).
Beweis: In [HUCK, S. 207] oder [PLATO, S. 83-84] nachzulesen.
Man kann aber die quadratische Konvergenz des Newton-Verfahrens aufrecht erhalten, indem man die Iteration wie folgt modifiziert:

[44] Das Sekanten-Verfahren ist also deutlich schneller als das Bisektionsverfahren, verwendet aber die Information $f(x_k)$ und $f(x_{k-1})$, wohingegen das Bisektionsverfahren nur die Vorzeichen von $f(x_k)$ benötigt. (In Anlehnung an [CAR, S. 177])

$$x_{k+1} = x_k - m\frac{f(x_k)}{f'(x_k)} \qquad k = 0, 1, 2, \ldots$$

wobei $m \in IN$ die Vielfachheit der Nullstelle ist.
Beweis: In [HERM, S. 205] nachzulesen.

Die rasche Konvergenz des Newton-Verfahren ergibt sich für Startwerte hinreichend nahe an der Nullstelle. Global (d. h. für beliebige Startwerte) muss das Verfahren nicht konvergieren, wie Abbildung 19 zeigt (in Anlehnung an [SCHRA, S. 90]).[45]

Das **gedämpfte Newton-Verfahren** ist linear konvergent (p=1) (gemäß [AUZI_2007, S. 190]).
Beweis: In [AUZI_2007, S. 190-192] nachzulesen.

Da sich das **Heron-Verfahren** aus dem Newton-Verfahren ableiten lässt (siehe Kapitel 4.6.1), ist die Konvergenzordnung ebenfalls wie beim Newton-Verfahren p=2.

[45] Eine detaillierte Betrachtung dieses Sachverhalts mit Graphiken zur Veranschaulichung ist in [SCHRA, S. 89-90] nachzulesen.

4.7.3 Newton-Verfahren und Regula-falsi-Verfahren im Vergleich: Ein Beispiel in Maple

Die in diversen Schulbüchern oft vorkommenden nichtlinearen Iterationsverfahren zur Nullstellenberechnung einer gegebenen Funktion sind das Newton-Verfahren und das Sekanten-Verfahren. Das folgende Beispiel in Maple zeigt anhand einer konkreten Aufgabenstellung aus dem Mathematikschulunterricht einen Vergleich der beiden Algorithmen hinsichtlich ihrer Effizienz.

(In Anlehnung an [DORN, S. 8])

Gegeben: Funktion $y = f(x)$ mit Definitionsbereich I, wobei I ein Intervall ist.
Gesucht: Absolute Extrema von $f(x)$ auf I.

Falls die absoluten Extrema von $f(x)$ auf I existieren, findet man diese unter den relativen Extrema im Inneren von I und den Werten von $f(x)$ an den Rändern von I. Gehört ein Randpunkt von I nicht zu I, so ist der entsprechende Limes zu untersuchen.

Aufgabenstellung:

Finde die absoluten Extrema der Funktion $f(x) = 3 - \frac{1}{1+x}\left(3 - e^{\frac{-x}{10}}\right) + e^{\frac{-x}{10}}$ für $x \geq 0$.[46]

- **Grundeinstellungen**

```
> restart:with(plots):Digits:=10:
```

[46] Bei $f(x)$ ist nur eine numerische Lösung möglich, weil x sowohl im Exponenten als auch außerhalb des Exponenten vorkommt.

- **Prozedur *Newton* und Prozedur *Sekante* in Maple programmieren**

Newton-Verfahren

Parameter der Prozedur Newton (f,xs0,tol)

f: Funktion, bei der eine Nullstelle gesucht ist

xs0: Startwert

tol: Genauigkeit (Toleranz)

Ergebnis: Nullstelle

```
> Newton:=proc(f,xs0,tol)
local x0,x1,i:                  # i...Iterationsschritt
i:=0:x0:=xs0:                   # Initialisierung von i bei 0 und
                                x0 bei Startwert xs0
while (i<1000) do               # Solange i kleiner 1000 ist,
                                soll folgendermaßen vorgegangen
                                werden
  i:=i+1:                       # Betrachte den nächsten
                                Iterationsschritt
  x1:=evalf(x0-f(x0)/D(f)(x0)):    # Formel des Newton-
                                   Verfahrens (4.5) zur
                                   Berechnung des nächsten
                                   Iterationsschrittes.
                                   evalf wertet den Ausdruck in
                                   Gleitkommadarstellung aus
  printf("x%d=%f\n",i,x1):      # Ausgabe der Lösung für den
                                eben berechneten
                                Iterationsschritt
  if (abs(x0-x1)<tol) then      # Falls der Absolutbetrag
                                zwischen zwei
                                aufeinanderfolgenden Näherungen
                                kleiner der vorgegebenen Toleranz
                                ist, soll das Verfahren abbrechen
    break:
  end if:
  x0:=x1:                       # Der x-Wert wird aktualisiert
end do:
return(x1):                     # Ausgabe des Ergebnisses mit
                                gewünschter Genauigkeit
end proc:
```

Sekanten-Verfahren

Parameter der Prozedur Sekante(f,xs0,xs1,tol)

f:	Funktion, bei der eine Nullstelle gesucht ist
xs0:	erster Startwert
xs1:	zweiter Startwert
tol:	Genauigkeit (Toleranz)

Ergebnis: Nullstelle

```
> Sekante:=proc(f,xs0,xs1,tol)
local x0,x1,x2,i:
i:=1:x0:=xs0:x1:=xs1:
while (i<1000) do
  i:=i+1:
  x2:=evalf(x1-f(x1)*(x1-x0)/(f(x1)-f(x0))):# Formel des
                                              Sekanten-
                                              Verfahrens (4.4)
  printf("x%d=%f\n",i,x2):
  if (abs(x1-x2)<tol) then
    break:
  end if:
  x0:=x1:
  x1:=x2: # Die Werte werden aktualisiert
end do:
return(x2):
end proc:
```

- **Berechnung der Randwerte**

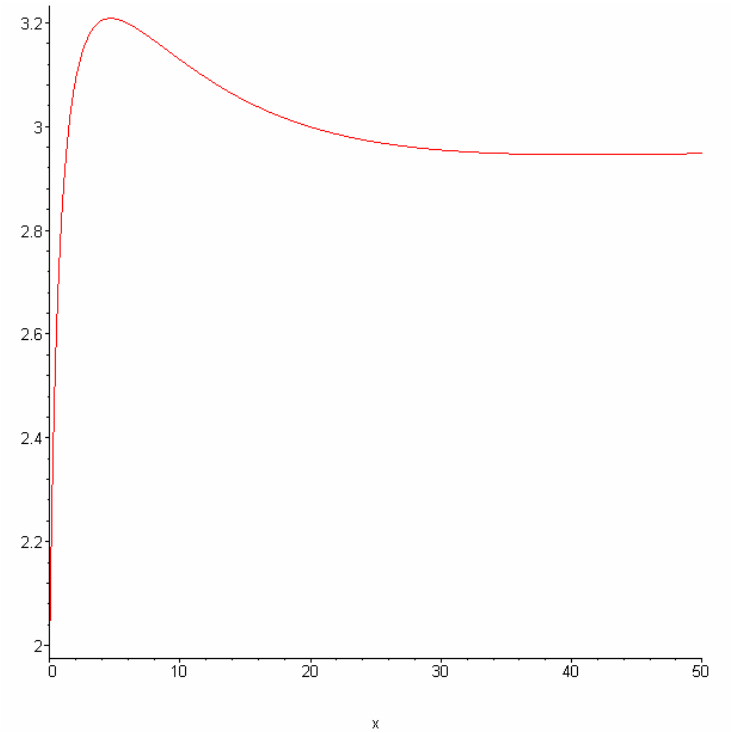
$$f(0) = 3 - 2 + 1 = 2$$

$$\lim_{x\to\infty} f(x) = \lim_{x\to\infty}\left(3 - \frac{1}{1+x}\left(3 - e^{\frac{-x}{10}}\right) + e^{\frac{-x}{10}}\right) = \lim_{x\to\infty}\left(3 - \frac{3 - \frac{1}{e^{\frac{x}{10}}}}{1+x} + \frac{1}{e^{\frac{x}{10}}}\right) = 3 - 0 + 0 = 3$$

- **Darstellung der Funktion** $f(x)$

```
> f:=x->3-1/(1+x)*(3-exp(-x/10))+exp(-x/10);
plot(f(x),x=0..50);
```

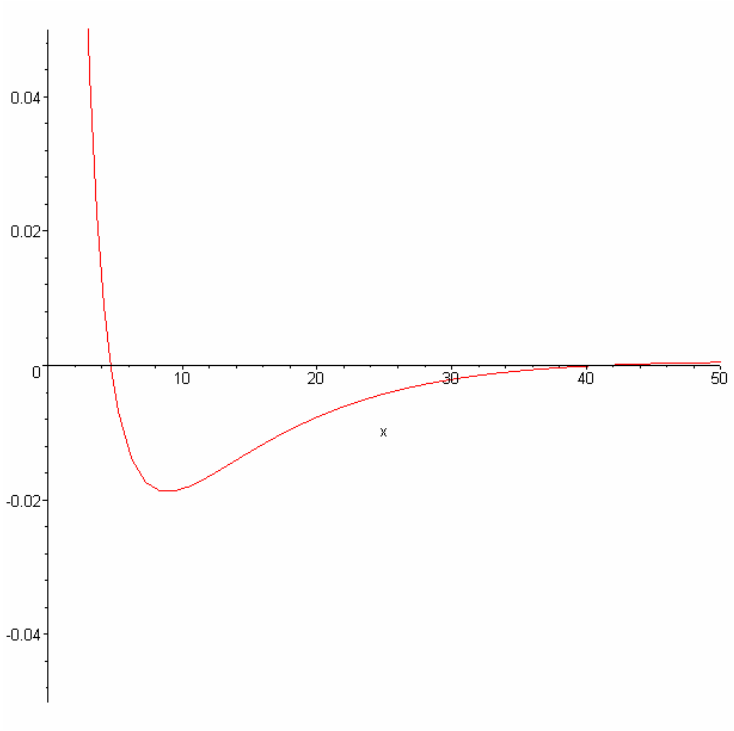
$$f := x \to 3 - \frac{3 - \mathbf{e}^{\left(-\frac{1}{10}x\right)}}{1+x} + \mathbf{e}^{\left(-\frac{1}{10}x\right)}$$



- **Darstellung der Ableitung** $f'(x)$

```
> df:=D(f);     # Differentialoperator D auf f anwenden
plot(df(x),x=0..50,view=[0..50,-0.05..0.05]);
```

$$df := x \to \frac{3 - \mathbf{e}^{\left(-\frac{1}{10}x\right)}}{(1+x)^2} - \frac{1}{10}\frac{\mathbf{e}^{\left(-\frac{1}{10}x\right)}}{1+x} - \frac{1}{10}\mathbf{e}^{\left(-\frac{1}{10}x\right)}$$



- **Berechnung der relativen Extrema**

```
> Newton(df,1,10^(-6));
```

```
x1=1.834591
x2=2.827050
x3=3.793050
x4=4.442915
x5=4.656850
x6=4.674800
x7=4.674914
x8=4.674914
```

4.674914189

```
> evalf(f(%));  # Mit dem Symbol % wird auf den vorhergehenden
                letzten Maple-Output zugegriffen
```

3.208340601

```
> x0:=1:x1:=5:
Sekante(df,x0,x1,10^(-6));
```

```
x2=4.956399
x3=4.640503
x4=4.678411
x5=4.674956
x6=4.674914
x7=4.674914
```

4.674914191

```
> evalf(f(%));
```

3.208340601

```
> Newton(df,50,10^(-6));

x1=30.746050
x2=36.840681
x3=40.145859
x4=41.009227
x5=41.059847
x6=41.060011
x7=41.060011
```

41.06001105

```
> evalf(f(%));
```

2.945538529

```
> x0:=10:x1:=50:
Sekante(df,x0,x1,10^(-6));

x2=49.015434
x3=32.364641
x4=45.022984
x5=42.906913
x6=40.499381
x7=41.128661
x8=41.062439
x9=41.060000
x10=41.060011
x11=41.060011
```

41.06001105

```
> evalf(f(%));
```

2.945538529

- **Zusammenfassung der Ergebnisse**

Im Inneren von I hat $f(x)$ bei $x_1 = 4{,}6749...$ ein relatives Maximum und bei $x_2 = 41{,}0600...$ ein relatives Minimum, wobei $f(x_1) = 3{,}2083...$ und $f(x_2) = 2{,}9455...$.

Randwertberechnung bez. I: $f(0) = 2$ und $\lim_{x \to \infty} f(x) = 3$.

$\Rightarrow$ absolutes Maximum bei x_1, absolutes Minimum bei 0.

- **Welches der beiden Verfahren konvergiert schneller bei der Genauigkeitsvorgabe von tol =** 10^{-6} **?**

Bei x_1 konvergieren das Newton- und Sekanten-Verfahren in etwa gleich schnell. Bei x_2 konvergiert das Newton-Verfahren deutlich schneller.

5 Literaturverzeichnis

[AUZI_2007] Auzinger, W. & Praetorius, D. (2007). Vorlesungsskriptum zu *Numerische Mathematik für Technische Mathematik*. Wien: Technische Universität, Institut für Analysis und Scientific Computing.

[AUZI_2008] Auzinger, W. & Weinmüller, E. (2008). Vorlesungsskriptum zu *Analysis 2 für Technische Physik.* Wien: Technische Universität.

[BAR] Barsch, H.-J. (2007). *Taschenbuch mathematischer Formeln.* München: Carl Hanser.

[BÄR] Bärwolff, G. (2007). *Numerik für Ingenieure, Physiker und Informatiker.* München: Elsevier.

[CAR] Carstensen, C. (2002). Vorlesungsskriptum zu *Numerische Mathematik 2*. Wien: Technische Universität.

[DORN] Dorninger, D. (2004). Vorlesungsskriptum zu *Anwendungen der Mathematik für LehramtskandidatInnen: Modelle – Verfahren – Beispiele.* Wien: Technische Universität, Institut für Diskrete Mathematik und Geometrie.

[FETZ] Fetzer, A. & Fränkel, H. (2008). *Mathematik 1. Lehrbuch für ingenieurwissenschaftliche Studiengänge.* Berlin: Springer.

[HERM] Hermann, M. (2006). *Numerische Mathematik*. München: Oldenbourg.

[HEU] Heuser, H. (2002). *Lehrbuch der Analysis Teil 2.* Stuttgart: Teubner.

[HUCK] Huckle, T. & Schneider, S. (2002). *Numerik für Informatiker.* Berlin: Springer.

[KALT] Kaltenecker, F. (2005). *Die numerische Lösung linearer Gleichungssysteme von der Antike bis in die Gegenwart.* Wien: Technische Universität Wien, Institut für Analysis und Scientific Computing. Diplomarbeit.

[KANZ] Kanzow, C. (2005). *Numerik linearer Gleichungssysteme: Direkte und iterative Verfahren*. Berlin: Springer.

[KNORR] Knorrenschild, M. (2005). *Numerische Mathematik. Eine beispielorientierte Einführung.* München: Carl Hanser.

[LENZ] Lenze, B. (2007). *Basiswissen Angewandte Mathematik. Numerik, Grafik, Kryptik.* Witten: W3L.

[LOCH] Locher, F. (1993). *Numerische Mathematik für Informatiker.* Berlin: Springer.

[MUNZ] Munz, C.-D. & Westermann, T. (2009). *Numerische Behandlung gewöhnlicher und partieller Differentialgleichungen. Ein interaktives Lehrbuch für Ingenieure.* Berlin: Springer.

[NEUN] Neundorf, W. (2002). *Numerische Mathematik. Vorlesungen, Übungen, Algorithmen und Programme.* Aachen: Shaker.

[OPF] Opfer, G. (2008). *Numerische Mathematik für Anfänger. Eine Einführung für Mathematiker, Ingenieure und Informatiker.* Wiesbaden: Vieweg+Teubner.

[PLATO] Plato, R. (2000). *Numerische Mathematik kompakt. Grundlagenwissen für Studium und Praxis.* Braunschweig/Wiesbaden: Vieweg.

[ŞAN] Şanal, Z. (2009). *Mathematik für Ingenieure. Grundlagen, Anwendungen in Maple und C++.* Wiesbaden: Vieweg+Teubner.

[SANNS] Sanns, W. & Schuchmann, M. (2001). *Praktische Numerik mit Mathematica*. Eine Einführung. Stuttgart: Teubner.

[SCHRA] Schranz-Kirlinger, G. & Frank, R. (2006). Vorlesungsskriptum zu *Numerische Mathematik für LehramtskandidatInnen.* Wien: Technische Universität, Institut für Analysis und Scientific Computing.

Schulbücher:

[MALLE_O2] Malle, G., Bürger, H., Fischer, R., Kronfellner, M., Mühlgassner, T. & Schlöglhofer, F. (2000). Mathematik Oberstufe 2. Wien: öbc&hpt.

[MALLE_6] Malle, G., Ramharter, E., Ulovec, A. & Kandl, S. (2005). *Mathematik verstehen 6.* Wien: öbv&hpt.

[NOVAK_O2] Novak, J. (1990). *Mathematik Oberstufe Band II.* Wien: Reniets.

[NOVAK_O3] Novak, J. (1991). *Mathematik Oberstufe Band III.* Wien: Reniets.

[REICHEL_5] Reichel, H.-C. & Müller, R. (2002). *Lehrbuch der Mathematik 5.Klasse.* Wien: öbv&hpt.

[REICHEL_6] Reichel, H.-C., Müller, R. & Hanisch, G. (2002). *Lehrbuch der Mathematik 6.Klasse.* Wien: öbv&hpt.

[REICHEL_8] Reichel, H.-C., Müller, R. & Hanisch, G. (1999). *Lehrbuch der Mathematik 8.* Wien: öbv&hpt.

Internetquellen:

[LPO] http://www.bmukk.gv.at/schulen/unterricht/lp/lp_ahs_oberstufe.xml (Zugriff am 16. Februar 2010)

[LPU] http://www.bmukk.gv.at/schulen/unterricht/lp/lp_ahs_unterstufe.xml (Zugriff am 16. Februar 2010)

[WIKI] http://de.wikipedia.org (Zugriff am 13. April 2010)