Werner-von-Siemens-Gymnasium Weißenburg/Bay.

Kollegstufe Abiturjahrgang 1992/1994

F A C H A R B E I T

aus der Mathematik

Das Gaußsche Eliminationsverfahren

Theoretische Grundlagen und programmierte Realisierung

Verfasser: Florian Michahelles

Leistungskurs: M31
Kursleiter: H. Weber
Bearbeitungszeitraum: 1. 2. 1993 bis
Abgabetermin: 1. 2. 1994

Inhaltsverzeichnis

Anhang

I) Theoretische Grundlagen

1. Einführung

Diese Facharbeit behandelt drei Verfahren zur Lösung linearer Gleichungssysteme. Im ersten werden zunächst die theoretischen Grundlagen der Verfahren dargelegt, im zweiten Teil folgt dann die Umsetzung der Verfahren in Computerprogramme.
Zunächst soll allerdings zuerst einmal der Aufbau der linearen Gleichungssysteme erklärt werden.

1.1 Aufbau einer linearen Gleichung

Lineare Gleichungssysteme sind ihrerseits aus linearen Gleichungen aufgebaut.
Lineare Gleichungen bestehen aus Summen von Termen wie:

$$ax + by + cz = d$$

a,b,c und d sind Konstanten, x, y und z dagegen Variablen in dieser Gleichung. Der auf der rechten Seite stehende Teil der Gleichung wird als Freiglied bezeichnet.
Ein Term ist eine Multiplikation von einer Konstanten einer Variable, wobei diese nur in der ersten Potenz auftreten darf.

1.2 Lineare Gleichungssysteme

Faßt man mehrere lineare Gleichungen zusammen, so erhält man ein lineares Gleichungssystem:

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

Die a_{ij} mit i,j=1..n sind Konstanten und x_i mit i=1..n Variablen. b_i mit i=1..n stellen die Freiglieder dar.
Obiges Gleichungssystem ist außerdem noch ein Spezialfall, da die Anzahl der Unbekannten der Anzahl der Gleichungen entspricht. Für diesen Spezialfall sind eindeutige Lösungen möglich. Um diese Lösungen zu finden, bedient man sich verschiedener Lösungsverfahren, z.B. Cramer'sche Regel, Gaußsches Eliminationsverfahren.
Letzteres soll später genauer untersucht werden.

1.3 Lineare Gleichungssysteme als Matrizen

Man kann das gerade vorgestellte Gleichungssystem auch noch auf eine andere Weise darstellen:

$$\begin{pmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ a_{31} & a_{32} & \ldots & a_{3n} \\ \ldots & \ldots & \ldots & \ldots \\ a_{n1} & a_{n2} & \ldots & a_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \ldots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ \ldots \\ b_n \end{pmatrix}$$

noch kürzer:

$$Ax = b$$

A repräsentiert die Koeffizientenmatrix, x die Unbekannten und b die Freiglieder des Gleichungssystems.

Bei Matrizen sind drei Umformungen erlaubt, ohne daß sich das Ergebnis der Matrix ändert:
1. Vertauschen von Zeilen
2. Ersetzen von einer Zeile durch eine Linearkombination aus dieser Zeile und einer anderen.
3. Vertauschen von Spalten

2. Gaußsches Eliminationsverfahren

Carl Friedrich Gauß war einer der größten Mathematiker überhaupt. Neben seinen großen Entdeckungen, wie z.B. der Gauß'schen Normalverteilung, der Gauß'schen Fehlerfunktion oder der Gauß'schen Zahlenebene, wird das Gauß'sche Eliminationsverfahren oft übersehen.
Der Gaußsche Eliminationsalgorithmus gehört zu den wichtigsten numerischen Verfahren der Mathematik. Er ist bereits 150 Jahre alt und hat sich in dieser Zeit kaum verändert. Dieses Verfahren wurde in den letzten zwanzig Jahren ausgiebig erforscht, was für die Effizienz dieses Algorithmuses spricht.
Zu dem ist das Gaußsche Eliminationsverfahren relativ einfach.
Der Algorithmus besteht aus zwei Etappen, aus der Vorwärtselimination und dem

Rückwärtseinsetzen.

2.1 Vorwärtselimination

Ziel dieser ersten Etappe ist es, das Gleichungssystem auf eine Dreiecksform zu bringen, wobei unterhalb der Hauptdiagonalen Nullen stehen und oberhalb neue Koeffizienten stehen sollen. Das Gleichungssystem soll nach dieser Etappe also wie folgt aussehen:

$$\begin{aligned} a'_{11}x_1 + a'_{12}x_2 + ... + a'_{1n}x_n &= b'_1 \\ 0 + a'_{22}x_2 + ... + a'_{2n}x_n &= b'_2 \\ ... & \\ 0 + 0 + 0 + a'_{nn}x_n &= b'_n \end{aligned}$$

Ein Gleichungssystem, das auf diese Form gebracht wurde, wird auch als gestaffeltes Gleichungssystem bezeichnet.
Die erste Zeile im Gleichungssystem enspricht bereits der Zeile im gestaffelten Gleichungssystem. In allen weiteren Zeilen muß man nun mit geeigneten Operationen in der ersten Spalte Nullen erzeugen. Dazu benutzt man die zweite Elementaroperation bei Matrizen. Man multipliziert dazu die erste Zeile mit einem geeigneten Faktor c und subtrahiert dieses Produkt dann von der zweiten Zeile.
Der Faktor c wird so berechnet:

$$c = a_{21} : a_{11}$$

Nach diesem Schritt hat man die erste Null erzeugt:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + ... + a_{1n}x_n &= b_1 \\ 0 + a'_{22}x_2 + ... + a'_{2n}x_n &= b'_2 \\ ... & \\ a_{n1}x_1 + a_{n2}x_2 + ... + a_{nn}x_n &= b_n \end{aligned}$$

Im zweiten Schritt multipliziert man wieder die erste Zeile mit einem Faktor c, allerdings dieses Mal mit $c = a_{31} : a_{11}$ und subtrahiert diese von der dritten Zeile. Dann wird das Produkt aus $c = a_{41} : a_{11}$ und erster Zeile von der vierten Zeile subtrahiert. So verfährt man weiter bis zur letzten Zeile. Nach diesem Vorgang sind alle a_{i1} mit i=2..n eliminiert:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + ... + a_{1n}x_n &= b_1 \\ 0 + a'_{22}x_2 + ... + a'2_nx_n &= b'_2 \\ ... & \\ 0 + a'_{n2}x_2 + ... + a'_{nn}x_n &= b'_n \end{aligned}$$

Jetzt stimmt auch schon die zweite Zeile mit der der gewünschten Endform überein. Man muß nun die zweite Spalte ab der dritten Zeile eliminieren. Dazu schreibt man beim Faktor c jeweils a'_{22} in den Nenner. Im Zähler steht dann wieder der jeweilige Koeffizient, der Null werden soll. Außerdem multipliziert man nun nicht mehr die erste Zeile mit diesem Faktor, sondern die zweite Zeile. Wir befinden uns also in der zweiten Eliminationsstufe. Sonst verfährt man wie bei der ersten Eliminationsstufe. Bei der Eliminationsstufe k wird der Faktor c also immer aus einem Quotienten mit dem ersten Koeffizient der k-ten Zeile im Nenner und dem zu eliminierenden Koeffizienten im Zähler gebildet. Mit diesem Faktor wird nun die k-te Zeile multipliziert und von der Zeile mit dem zu eliminierendem Koeffizient subtrahiert.
Der Faktor c allgemein:

$$c_{ik} = a_{ik}:a_{kk}$$

mit
n = Anzahl der Gleichungen und Unbekannten
k = 1..n-1; Eliminationsstufe
i = 1..n-1; Zeile

Nachdem man n-1 Eliminationen durchgeführt hat, ist die Dreiecksform erreicht:

$$w_{11}x1 + w12x2 +...+ w1nxn = v1$$
$$w22x2 +...+ w2nxn = v2$$
$$.....$$
$$wnnxn = vn$$

Die neuen Koeffizienten heißen jetzt wij mit i,j=1..n, da die Koeffizienten aij mit i=j=1..n durch die bisherigen Umformungen andere Werte wij mit i=j=1..n angenommen haben. Dasselbe gilt für die Freigelieder bi mit i=1..n und vi mit i=1..n.
Aus dieser Dreiecksform lassen sich nun die Unbekannten xi mit i=1..n durch Rückwärtseinsetzen leicht errechnen.

2.2 Rückwärtseinsetzen oder Rücksubstitution

Die Lösung der letzten Gleichung steht nun schon fast da. Man muß nur noch nach xn auflösen:

$$xn = vn:wnn$$

Betrachtet man die vorletzte Gleichung, so erkennt man, daß xn-1 wieder durch Auflösen nach xn-1 und mit bekanntem xn berechnet werden kann:

$$xn-1 = (vn-1 - wnnxn):wn-1n$$

Auf diese Weise kann man sich im ganzen gestaffelte Gleichungssystem hocharbeiten, indem man immer wieder mit bisher bekannten Lösungen eine neue ausrechnet.
Damit sind nun alle xi mit i=1..n bestimmt.

2.3 Verbesserung durch Pivotierung

Durch das nachfolgend erklärte Verfahren kann die Effektivität der Vorwärtselimination verbessert werden.
Als Spitzen- oder Pivotelement bezeichnet man den Koeffizienten, der bei der Berechnung des Faktors c im Nenner steht.
Man kann nun durch Zeilenvertauschen erreichen, daß das Pivotelement das betragsgrößte in der jeweiligen Spalte ist, z.B. bei Eliminationsstufe k=1 und |a11| < |a51|:

nach Zeilvertauschen:

$$a51x1 + a52x2 +...+ a5nxn = b5$$
$$a21x1 + a22x2 +...+ a2nxn = b2$$
$$a31x1 + a32x2 +...+ a3nxn = b3$$

$$
\begin{aligned}
a41x1 + a42x2 +...+ a4nxn &= b4\\
a11x1 + a12x2 +...+ a1nxn &= b1\\
a61x1 + a62x2 +...+ a6nxn &= b6\\
&.....\\
an1x1 + an2x2 +...+ annxn &= bn
\end{aligned}
$$

Für den Faktor c bedeutet dies, daß er nun höchstens den Wert 1 annehmen kann, da das Pivotelement bei der Berechnung von c im Nenner steht und ist durch die Pivotierung der Nenner stets größer als der Zähler.
Durch dieses Verfahren kann man die Koeffizienten klein halten. Andernfalls könnten einige Koeffizienten sehr groß werden und andere sehr klein, denn wenn der Pivotwert aii mit i=1..n sehr klein wäre, würde der Faktor c sehr groß.
Darauffolgend würde aii nun mit dem Faktor c multipliziert werden. Rechenmaschinen haben nun einmal große Probleme damit, aus Produkten mit sehr großen und sehr kleinen Faktoren vernünftige Ergebnisse zu berechnen. Ein Gleichungsystem könnte nun derart schlecht konditioniert sein, daß sich solche Rechenfehler häufen und die Lösungen verzerrt würden. Dies wird mit der Pivotierung umgangen.
Werden nur Zeilenausgetauscht, so bezeichnet man dieses Vorgehen als partielle Pivotierung. Vollständige Pivotierung wäre, wenn man zusätzlich noch Spalten vertauschen würde. In der Praxis hat sich jedoch gezeigt, daß das weitaus aufwendigere vollständige Pivotieren gegenüber dem partiellen Pivotieren keinen Vorteil hat. Die Genauigkeit der Lösung wird nicht verbessert.
Die partielle Pivotierung darf bei der Anwendung des Gauß'schen Eliminationsverfahren noch aus einem anderen Grund keineswegs fehlen. Es könnte nämlich sein, daß das aktuelle Pivotelement den Wert 0 hat. In diesem Fall würde man bei der Berechnung des Faktors c auf eine Division durch Null stoßen.
Mit der Pivotierung wird das umgangen, sie darf also auf keinen Fall fehlen.

3. Gauß-Jordan Algorithmus

Der Gauß-Jordan Algorithmus unterscheidet sich vom Gauß'schen-Eliminationsverfahren nur in der zweiten Etappe. Die erste Etappe ist bei beiden Verfahren die Vorwärtselimination. Nachdem nun das Gleichungssystem in ein gestaffeltes System doch Vorwärtseliminieren überführt worden ist, wird nun nicht rücksubstitutiert, sondern man versucht oberhalb der Hauptdiagonale ebenfalls Nullen zu erzeugen. Zum Schluß bleibt also nur noch die Hauptdiagonale stehen, und die Werte für die xi mit i=1..n sind damit gegeben:

$$
\begin{aligned}
s11x1 &= t1\\
s22x2 &= t2\\
&.....\\
&. snnxn = tn
\end{aligned}
$$

Das angestrebte Ziel erreicht man durch das sogenannte Gauß-Jordan'sche Reduktionsverfahren. Man geht von einem gestaffelten Gleichungsystem, das durch Vorwärtselemination erstellt wurde:

$$
\begin{aligned}
w11x1 + w12x2 +.....+ w1nxn &= v1\\
w22x2 +.....+ w2nxn &= v2\\
&.....\\
wn-1,n-1xn-1 + wn-1,nxn &= vn\\
wnnxn &= vn
\end{aligned}
$$

Die eben durchgeführte Vorwärtselimination muß nun genau in anderer Richtung durchgeführt werden, d.h. es gilt nun in der ersten Reduktionsstufe die letzte Spalte ab der vorletzten Zeile zu

eliminieren, denn die letzte Zeile entspricht bereits der gewünschten Endform. Die letze Zeile wird also wieder mit einem Faktor c=wn-1,n:wnn multipliziert und dann von der vorletzen Zeile subtrahiert werden. Die vorletzte Zeile lautet dann:

w'n-1,n-1x'n-1 = v'n-1

Ähnlich behandelt man die (n-2)-te Zeile. Das Produkt aus c=wn-2,n:wnn und wnn wird von der (n-2)-ten Zeile abgezogen, zurück bleibt:

w'n-2,n-2xn-2 + w'n-2,n-1xn-1 = v'n-2

Ist man schließlich in der ersten Zeile angelangt, so geht man über zur zweiten Reduktionsstufe. Bei der Berechnung des Faktors c steht nun w'n-1,n-1 im Nenner und der Subtrahend ist nun das Produkt aus c und der vorletzten Zeile. Dies führt man ebenfalls wieder solange durch, bis man die erste Zeile erreicht hat. Dann geht man über zur dritten Reduktionsstufe.
Letztendlich erreicht man die (n-1)-te Reduktionsstufe und die Hauptdiagonalen-Form des Gleichungssystems ist erreicht.
Der Faktor c bei dem Reduktionsverfahren wird allgemein wie folgt berechnet:

cik = an-i,n-k+1:an-k-1,n-k+1

mit
n = Anzahl der Gleichungen und Unbekannten
k = 1..n-1; Reduktionsstufe
i = 1..n-1; Zeile

Um die Lösungen für die xis mit i=1..n zu bekommen, muß man lediglich die einzelnen Gleichungen nach x auflösen:

x1 = t1:s11
x2 = t2:s22
.....
xn-1 = tn-1:sn-1,n-1
xn = tn:snn

Hiermit ist die Lösung für das Gleichungssystem mit Hilfe des Gauß-Jordan-Verfahrens gefunden worden.

4. Gauß-Seidel-Verfahren

Dieses Verfahren weist keine Gemeinsamkeit mit den beiden vorherigen auf.
Dieses Verfahren enthält in seinem Namen den Namen Gauß deshalb, weil bereits Carl Friedrich Gauß dieses Verfahren anwendet. Allerdings veröffentlichte er dieses Verfahren nicht.
Viele Jahre stieß der Mathematiker Phillip L. von Seidel erneut auf diesen Algorithmus, deshalb Gauß-Seidel-Verfahren.
Das Gauß-Seidel-Verfahren ist ein iteratives Verfahren, d.h. die Lösung des Gleichungssystems wird durch Iterationen genähert.
Gegeben sei wieder folgendes Gleichungssystem:

$$
\begin{aligned}
a11x1 + a12x2 +...+ a1nxn &= b1\\
a21x1 + a22x2 +...+ a2nxn &= b2\\
&.....\\
an1x1 + an2x2 +...+ annxn &= bn
\end{aligned}
$$

Man löst nun nach xi für i=1..n auf:

$$
\begin{aligned}
x1 &= (b1 - a12x2 -...- anxn):a11\\
x2 &= ...\\
&.....\\
xn &= ...
\end{aligned}
$$

Nun wählt man sich für xi mit i=1..n einen Startwert. Erstaunlicherweise ist dieser Startwert beliebig, da in den folgenden Iterationen die xis gegen die Lösung konvergieren. Einfachhalber wählt man für alle xi Null als Startwert:

$$
\begin{aligned}
x1 &= 0\\
x2 &= 0\\
&.....\\
xn &= 0
\end{aligned}
$$

Im zweiten Durchgang berechnet man nun ein neues x1 mit den vorherigen anderen xis:

$$x'1 = (b1 - a120 -...- an0):a11$$

Darauf werden die übrigen x'is jeweils mit den vorherigen Lösungen berechnet. Führt man diesen Prozeß weiter, so hat man nach genügend Durchläufen brauchbare Lösungen.
Es kann jedoch passieren, daß der Prozeß nicht konvergiert, sondern divergiert. Man kann dies manchmal umgehen, indem man wieder die oben besprochene partielle Pivotierung anwendet.
Meistens divergiert der Iterationsprozeß mit Pivotieren aber trotzdem. Eine Konvergenz tritt nämlich nur dann ein, wenn die Koeffizienten in der Hauptdiagolen (Pivotelemente) gegenüber den anderen Koeffizienten vom Betrag stark überwiegen. Die Zahl der Gleichungsysteme, die für das Gauß-Seidel-Verfahren geeignet sind, sinkt damit natürlich erheblich.
Man muß vor Anwendung dieses Verfahren nach dem Pivotieren also immer überprüfen, ob alle Pivotelemente aii mit i=1..n vom Betrag her größer als die Summe der aij j=1..n sind. Nur dann nämlich kann man eine schnelle Konvergenz garantieren. Eine Konvergenz könnte aber trotzdem eintreten, die dann aber so langsam wäre, daß ein anderes Verfahren die Lösung schneller finden würde.
Weiterhin hat es sich als günstig erwiesen, die Schrittweite zu halbieren, wenn sich das neue xi vom vorherigen x'i im Vorzeichen unterscheidet:

$$xi = (xi + x'i):2$$

Ferner sollte man sich noch überlegen, wann die Iterierung abgebrochen werden soll. Dies hängt von der gewünschten Genauigkeit der Lösung ab. Man kann den Iterationsprozeß für beendet erklären, wenn sich xi vom vorherigen x'i nur noch um eine Toleranzgröße unterscheidet. Allerdings kann man den Rechenprozeß auch erst dann stoppen, wenn xi mit dem vorherigen x'i übereinstimmt, die Toleranz Null ist.
Das Gauß-Seidel-Verfahren eignet sich vor allem für sehr große Matrizen. Vorteilhaft ist zudem, daß die Näherung immer nur von der vorherigen abhängt, d.h. Rundungsfehler können sich nicht

anhäufen.
Außerdem eignet sich das Gauß-Seidel-Verfahren auch zur Lösung nicht linearer Gleichungen.

5. Probleme mit den Algorithmen

Grundsätzlich arbeiten die oben beschriebenen Algorithmen nur mit nicht singulären Gleichungssystemen. Wendet man einen der Algorithmen auf ein singuläres Gleichungssystem an, so stößt man dann meist auf eine Nulldivision. Dies ist dann der Hinweis dafür, daß keine eindeutige Lösung existiert.
Probleme kann es mit Gleichungssystemen geben, deren Determinanten fast Null sind. Aufgrund von Rundungsfehlern stößt man dann vielleicht auf eine Division durch Null und schließt daraus, daß für dieses Gleichungssystem keine Lösung existiert. Tatsächlich existiert aber doch eine Lösung.
Ein weiteres Problem ist es, wenn in einem Gleichungssystem eine Gleichung eine komplizierte Linearkombination aus anderen Gleichungen dieses Gleichungssystems ist. Derartige Linearkombinationen sind schwer zu finden. Man könnte also eine Lösung erhalten, die aber nicht eindeutig ist.
Glücklicherweise stehen einem aber mehrere Lösungsverfahren zur Verfügung. Man sollte deshalb ein Gleichungssystem generell mehreren Verfahren unterziehen. Errechnen nun verschiedene Verfahren unterschiedliche Lösungen, so ist dies ein Hinweis darauf, daß keine eindeutige Lösung existiert.
Zusätzlich ist es ratsam, die lineare Unabhängigkeit von Gleichungen innerhalb eines Gleichungssystems mit Hilfe der Determinante zu überprüfen. Besonders bei großen Gleichungssystemen ist dies aber oft sehr aufwendig.
Speziell beim Gauß-Seidel-Verfahren hat man das Problem, daß dieser Algorithmus nur in sehr begrenztem Maße einsetzbar ist, da nur wenige Gleichungssysteme für dieses Verfahren geeignet sind.

II) Programmumsetzungen

Die drei obigen Algorithmen per Hand auszuführen wäre sehr zeitaufwendig und mühselig. Ständig diesselben Rechenoperationen zu betreiben, wären für einen Menschen zu ermüdend. Ein einziger Rechenfehler würde die ganze Arbeit wertlos machen, da dieser Rechenfehler das Ergebnis völlig verfälschen würde.
Mehr an Bedeutung gewinnen die Algorithmen, wenn man sie von Computern ausführen läßt. So lassen sich Rechenfehler ausschließen, man muß lediglich noch mit Rundungsfehlern zurechtkommen. Außerdem ist die Ausführungszeit fast vernachlässigbar, in Bruchteilen von Sekunden wird das Ergebnis vom Computer berechnet.
Weil der Computer einen Algorithmus nicht einfach bearbeiten kann, muß man diesen erst in eine für den Computer verständliche Form umsetzten. Die Umsetzung ist dann ein Computerprogramm.
Die Entwicklungen für Computerprogramme der drei beschriebenen Algorithmen soll nachfolgend vollzogen werden. Die Programme werden in PASCAL geschrieben und sollten systemunabhängig sein. Im folgenden Verlauf wird zunächst das Grundprinzip und dann erst das fertige Programm, das, wie das zugehörige Ablaufprotokoll, im Anhang zu finden ist. Es empfiehlt bei der Programmbeschreibung, den Programmquelltext vor sich liegen zu haben, um die einzelnen Schritte besser verfolgen zu können.
Bei der Programmierung wurde darauf geachtet, daß sich die einzelnen Programme sehr ähneln. Es wurde versucht ähnliche Programmteile wie z.B. Ein- und Ausgabe einheitlich zu gestalten, folglich werden gleiche Programmteile auch nur einmal im Text erklärt.

1. Gauß'sches Eliminationsverfahren

1.1 Grundprinzip

Dem Computer muß zu Beginn erst einmal das Gleichungssystem übergeben werden, auf das das Gauß'sche Eliminationsverfahren angwendet werden soll. Die Befehlsanweisungen sollte man dazu sinnvollerweise in einer Prozedur mit dem Namen "Eingabe" zusammenfassen. Jetzt ist es wieder nützlich, sich das Gleichungssystem in Matrix-Schreibweise vorzustellen. Lediglich Koeffizientenmatrix und den Freigliedervektor muß dem Computer übergeben werden. Es bietet sich an, diese Werte in einem zwei-dimensionalen nx(n+1) Array vom Typ Real abzulegen. In der Spalte n+1 legt man die Freiglieder ab. Diese Feld muß ebenso global definiert sein wie die Dimension n des Gleichungssystems.
Die Eingaberoutine könnte dann wie folgt lauten:

```
PROCEDURE Eingabe;
BEGIN
   Write('Anzahl der Gleichungen: ');
   ReadLn(n);
   FOR i:=1 TO n DO
     FOR j:=1 TO n+1 DO
     BEGIN
       Write('Koeffizient[',i,',',j,'] ');
       ReadLn(Koeffizient[i,j]);
     END;
END;
```

Dem Computer ist das Gleichungssystem nun bekannt, der Eliminationsvorgang kann gestartet werden. Die Prozedur "Eliminieren" soll die Befehlsanweisungen dafür in sich vereinen.
Insgesamt benötigt man drei Laufschleifen. Die äußerste Schleife, die k-Schleife, führt die k-Eliminationsstufen und Pivotierungen durch. Die k-Schleife läuft von eins bis n-1, da in einem n-dimensionalen Gleichungssystem n-1 Eliminationsstufen notwendig sind, um dieses in ein gestaffeltes Gleichungsystem zu überführen. Vor jeder Eliminationsstufe muß zunächst der Pivotierungsvorgang durchgeführt werden. Dieser ist unbedingt notwendig, da das Programm sonst eventuell mit "Division by Zero" abbrechen könnte.
Die Pivotierung gestaltet sich ziemlich einfach. Für das Pivotelement nimmt man zu Anfang Null an, sucht dann mit einer Schleife die k-te Spalte nach vom Betrag her größeren Werten ab und ermittelt dann die Zeile des größten Wertes.
Diese Zeile vertauscht man dann in einer anderen Laufschleife mit der k-ten Zeile. Ist das Pivotelement trotzdem null, so muß der Eliminationsvorgang abgebrochen werden, da das Gleichungssystem in diesem Fall keine eindeutige Lösung besitzt.
Erst jetzt kann eliminiert werden. Dazu benötigt man zwei verschachtelte Laufschleifen. Die äußere sei die i-Schleife, sie gibt an, in welcher Zeile gerade eliminiert wird. In dieser wird der Faktor c ermittelt, mit dem dann die Zeile k multipliziert und von der Zeile i subtrahiert wird. Dieser Vorgang wird in der innersten Schleife, der j-Schleife, ausgeführt.
Die Prozedur "Eliminieren" ist damit komplett:

```
PROCEDURE Eliminieren;
VAR Pivotelement, Faktor: REAL;
    Pivotzeile, i, j, k: INTEGER;
BEGIN
  FOR k:= 1 TO n - 1 DO {* Gesamtelimination}
    BEGIN
```

```
      Pivotelement:= 0;
      Pivotzeile:= k;
      FOR i:= k TO n DO  {* Pivotelement suchen}
        IF Abs(Koeffizient[i, k])  Pivotelement THEN
        BEGIN
          Pivotelement:= Abs(Koeffizient[i, k]);
          Pivotzeile:= i;
        END;
      IF k <> Pivotzeile THEN {* wenn k=i, Pivotelement bereits richtig}
        FOR j:= k TO n + 1 DO {* Zeilen vertauschen}
          Exchange(Koeffizient[k, j], Koeffizient[Pivotzeile, j]);

      IF Koeffizient[k, k] = 0 THEN Abbruch;
      FOR i:= k + 1 TO n DO {* Elimination auf Stufe k}
        BEGIN
          Faktor:= Koeffizient[i, k] / Koeffizient[k, k];
          FOR j:= k TO n + 1 DO
            Koeffizient[i,j]:= Koeffizient[i,j]-Faktor*Koeffizient[k,j];
        END;
    END;
END;
```

Die Vorwärtselimination ist beendet, das Rücksubstituieren kann gestartet werden. Die dazugehörige Prozedur soll "Rückwärtseinsetzen" heißen. Man benötigt zwei Laufschleifen, eine äußere i- und eine innere j-Schleife. Die i-Schleife durchläuft rückwärts die Werte von n bis eins.
Um den folgenden Schritt besser verstehen zu können, ruft man sich am besten noch einmal das gestaffelte Gleichungssystem aus I.2.1 ins Gedächtnis:

$$
\begin{aligned}
a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n &= a_{1,n+1} \\
a_{22}x_2 + \ldots + a_{2n}x_n &= a_{2,n+1} \\
\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots & \\
a_{nn}x_n &= a_{n,n+1}
\end{aligned}
$$

Löst man nun jeweils die i-te Gleichung nach xi auf, so erhält man folgendes:

$$
\begin{aligned}
x_1 &= (a_{1,n+1} - (a_{12}x_2 + \ldots + a_{1n}x_n)) : a_{11} \\
x_2 &= (a_{2,n+1} - (a_{23}x_3 + \ldots + a_{2n}x_n)) : a_{22} \\
\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots & \\
x_n &= (a_{n,n+1} - 0) : a_{nn}
\end{aligned}
$$

Die j-Schleife berechnet nun für die Zeile i die Summe in der innersten Klammer (oben kursiv). Die j-Schleife durchläuft die Werte i+1 bis n.
Man könnte denken, es gäbe ein Problem, wenn i=n ist, weil dann j=n+1 sein muß und somit auch xn+1 herangezogen würde, das natürlich nicht existiert. Doch löst sich dieses Problem in Wohlgefallen auf, da die j-Schleife in diesem Fall überhaupt nicht mehr durchlaufen wird, da dann der Startwert j=n+1 größer als der Endwert n ist.
Im weiteren Verlauf der i-Schleife wird nun noch geprüft, ob ann=0 und gegebenenfalls wieder ein "Division by Zero" Fehler abgefangen.
Letztendlich wird die Lösung xi berechnet.
Als Programmtext sieht dies wie folgt aus:

```
PROCEDURE RückwärtsEinsetzen;
VAR i, j, NullPos: INTEGER;
    Term: REAL;
BEGIN
```

```
  FOR i:= n DOWNTO 1 DO
    BEGIN
      Term:= 0;
      FOR j:= i + 1 TO n DO
        Term:= Term + Koeffizient[i, j] * Lösung[j];

      IF Koeffizient[i, i] = 0 THEN Abbruch;
      Lösung[i] := (Koeffizient[i, n + 1] - Term) / Koeffizient[i, i];
    END;
END;
```

Jetzt steht die simultane Lösung in dem Array "Lösung". Dieses kann man sich am Ende noch mit einer Prozedur "DruckeLösung" ausgeben lassen. Das komplette Programm liegt im Anhang als Quelltext vor.

1.2 Programmbeschreibung

Die Programmbeschreibung wird in die verschiedenen Unterpunkte Variablen und Konstanten, Prozeduren und Funktionen und Hauptprogramm unterteilt.
Das unten beschriebene Programm gibt zusätzlich auf Wunsch noch Rechenzwischenschritte aus, in der Programmbeschreibung wird darauf allerdings nicht eingegangen. Anzumerken ist noch, daß diese Zwischenschrittausgabe sehr einfach gehalten wurde, d.h. es kann zu Ausgaben kommen, die mathematisch nicht korrekt sind (z.b. zwei aufeinanderfolgende Verknüpfungszeichen).
Die einzelnen Schritte werden in der Reihenfolge beschrieben, in der sie auch im Programmtext zu finden sind.

Globale Variablen und Konstanten

In der Konstante MaxGleichungen wird die maximale Anzahl der Gleichungen, d.h. die maximale Dimension des Gleichungssystems festgelegt. Dies ist notwendig, da in PASCAL die Größe von Arrays nur über Konstanten definiert werden kann.
Die Konstante ZeichenBreite enthält die Breite des Bildschirms in Zeichen. Dieser Wert wird verwendet, um Daten zentriert ausgeben zu können.
Koeffizient ist eine Variable vom Typ Matrix in diesem zweidimensionalen Feld wird die Koeffizientenmatrix zusammen mit den Freigliedern gespeichert.
Lösung soll am Ende des Programms die simultane Lösung des Gleichungssystems enthalten. Lösung kann genauso wie Koeffizient Zahlen vom Typ Real speichern.
n beeinhaltet die tatsächliche Dimension des Gleichungssystems, allerdings darf n MaxGleichungen nicht überschreiten.

Prozeduren

Die Prozedur Eingabe liest, wie der Name bereits ausdrückt, die vom Benutzer gewünschten Daten ein und speichert diese dem entsprechend in n und Koeffizient.
DruckeGleichung druckt die Gleichung Nr auf den Bildschirm. Diese Prozedur wird von DruckeGleichungssystem verwendet, um das komplette Gleichungssystem in einem ansehnlichen Format auszugeben.
Die Prozedur Abbruch übernimmt die Fehlerausgabe, falls das eingegebene Gleichungssystem keine eindeutige Lösung besitzt.

Eliminieren und RückwärtsEinsetzen sind die Prozeduren, die vorher bereits ausführlich entwickelt wurden. Sie wurden jedoch noch um einige Zeilen erweitert, damit die jeweiligen Zwischenschritte mit ausgegeben werden können, sonst wurde nichts verändert.
DruckeLösung übernimmt die Ausgabe der simultanen Lösung.

Hauptprogramm:

Das Hauptprogramm konnte sehr kurz gehalten werden, da die überwiegende Arbeit bereits von den

Prozeduren übernommen wird.
Zu Beginn des Hauptprogramms wird mittels der Prozedur Eingabe die Eingabe des Benutzers bearbeitet, darauf wird der Bildschirm gelöscht und die eingegebenen Koeffizienten als Gleichungssystem mit DruckeGleichungssystem dargestellt. Schließlich übernehmen die Prozeduren Eliminieren und RückwärtsEinsetzen das Lösen des Gleichungssystems. Diese Lösung wird dann zum Schluß mit DruckeLösung ausgegeben.

2. Gauß-Jordan Verfahren

2.1 Grundprinzip

Das PASCAL-Programm für das Gauß-Jordan-Verfahren läßt sich sehr gut aus dem gerade besprochenen entwickeln. Da sich der Gauß-Jordan Algorithmus vom Gauß'schen Eliminationsvefahren erst ab dem Jordan'schen Reduktionsvefahren unterscheidet, lassen sich die Prozeduren "Eingabe" und "Eliminieren" unverändert übernehmen.
Man muß lediglich eine "GaussJordanReduktion"-Prozedur programmieren. Dies gestaltet sich auch relativ einfach, da die "Eliminieren"-Prozedur dafür nur etwas umgeschrieben werden muß:

```
PROCEDURE GaussJordanReduktion;
VAR i, j, k: INTEGER;
    Faktor: REAL;
BEGIN
  FOR k:= n DOWNTO 1 DO
    BEGIN
      IF Koeffizient[k, k] = 0 THEN Abbruch;
      FOR i:= k - 1 DOWNTO 1 DO
        BEGIN
          Faktor:= Koeffizient[i, k] / Koeffizient[k, k];
          FOR j:= n + 1 DOWNTO i DO
            Koeffizient[i,j]:=Koeffizient[i,j]-Faktor*Koeffizient[k, j];
        END;
      Lösung[k] := Koeffizient[k, n + 1] / Koeffizient[k, k];
    END;
END;
```

Die Pivotierung kann völlig entfallen. In dem gestaffelten Gleichungssystem existiert pro Spalte jeweils nur ein Pivotelement, so daß es also gar keine Alternative zum Tauschen gibt. Totales Pivotieren wäre möglich, würde das Programm aber nur unnötig verlängern.
Die weiteren Änderungen sind ebenfalls sehr gering. Es müssen lediglich die Grenzen und die Laufrichtung der k-, i- und j-Schleife geändert werden (kursiv hervorgehoben).

2.2 Programmbeschreibung

Auch dieses Programm liegt wieder im Anhang als Source vor, wobei auch dieses Programm wieder Zwischenschritte ausgeben kann. Auf die Dokumentation dieser Programmteile wird wie oben verzichtet.
Die Programmbeschreibung kann sehr kurz gehalten werden, da dieser Quelltext mit dem Programm Gauß in den Punkten Variablen und Konstanten, Funktionen und Hauptprogramm fast übereinstimmt. Es wurde lediglich die Prozedur Rückwärtseinsetzen durch die Prozedur GaussJordanReduktion ausgetauscht, sonst blieb alles beim Alten.

3. Gauß-Seidel Verfahren

3.1 Grundprinzip

Das Programm für dieses Verfahren unterscheidet sich, genauso wie das Verfahren selbst, von den beiden vorherigen fast völlig. Folglich läßt sich von den beiden vorherigen, abgesehen von der Ein- und Ausgaberoutine und Pivotierung, nichts übernehmen.
Bevor das Iterationsverfahren starten kann, muß zuerst das beste Pivotelement gesucht werden, um ein divergieren des Iterationsprozesses zu vermeiden. Im Gegensatz zu den beiden vorherigen Verfahren wird beim Gauß-Seidel Verfahren in einem Zug durchpivotiert, d.h. die Pivotierung steht getrennt vor den Iterationsanweisungen.
Zur Pivotierung benötigt man drei Schleifen, wobei zwei in einer verschachtelt sind. Die äußerste Schleife ist die k- Schleife, die von 1 bis n-1 läuft und die Spalten durchläuft, aus welchen das Pivotelement ermittelt werden soll. Der nun folgende Programmteil aus i- und j-Schleife ist aus den beiden vorher beschriebenen Programmen bekannt.
Nun kommt die Überprüfung des Gleichungssystems. Die äußere i-Schleife durchläuft die zeilenweise das Gleichungssystem. Stößt die innere j-Schleife nur auf eine Zeile, in der das Pivotelement nicht überwiegt, ist das Gleichungssystem bereits ungeeignet, bricht die i-Schleife ab und eine Meldung wird ausgegeben. An dieser Stelle könnte man auch gleich das Programm abbrechen.
Die Variable SZeile enthält immer die Summe der Beträge der Koeffizienten der Zeile i ohne das Diagonalelement.
Aber nun zur Umsetzung des Iterationsprozesses. Zuerst einmal muß man die initiale Näherung bestimmen, d.h. alle Felder des Arrays "Lösung" auf Null setzen, dann kann man den Iterationsprozeß starten. Man benötigt zur Programmierung drei ineinander verschachtelte Schleifen. Die äußerste bestimmt die Dauer des Iterationsprozesses. Dieser soll nämlich, nach folgenden Kriterien abgebrochen werden. Entweder wenn die Lösung gefunden wurde, d.h. wenn sich die Näherung gegenüber der vorherigen nicht mehr verbessert, oder wenn die Zahl der Iterationen über einen vorgegebenen Wert hinaus gewachsen ist. Hierzu bietet sich eine REPEAT-DO-UNTIL-Schleife an, da man die Anzahl der Durchläufe vorher nicht kennt. Zu Beginn dieser Schleife wird die Laufvariable um
eins erhöht, dann startet die nächste Schleife, die i-Schleife. Diese beginnt bei dem Wert eins und endet bei n. In dieser Schleife werden jeweils die xi mit i=1..n berechnet. Jetzt folgt bereits die innerste Schleife, die j-Schleife, die ebenfalls die Werte von eins bis n durchläuft. Mit dieser Schleife wird jeweils der Zähler aus den vorherigen Iterationen bzw. der initialen Näherung und den dazugehörigen Koeffizienten berechnet. Nach dieser Schleife kann nun das neue xi berechnet werden, wobei vorher wieder ein "Division by Zero"-Fehler abgefangen werden muß. Nun wird noch verglichen, ob sich Genauigkeit der neuberechneten Lösung gegenüber der alten verbessert hat. Ist dies nicht der Fall, so wird der Iterationsvorgang beendet. Die andere Möglichkeit wäre, daß die Maximalzahl der Iterationen überschritten wurde und somit keine genaue Lösung gefunden wurde. Diesen Fall mit einzubeziehen ist wichtig, da das Programm vom Computer bis in die Ewigkeit weitergeführt würde, ohne eine Lösung zu finden. Hier der Programmtext:

```
PROCEDURE GaussSeidel;
VAR Pivotelement, Zähler, NeueLösung, SZeile: REAL;
    Pivotzeile, i, j, k, Iteration: INTEGER;
    Fertig, geeignet: BOOLEAN;
BEGIN
  FOR k:= 1 TO n - 1 DO
    BEGIN
      Pivotelement:= 0;
      FOR i:= 1 TO n DO  {* Pivotelement suchen}
        IF Abs(Koeffizient[i, k])  Pivotelement THEN BEGIN
```

```
          Pivotelement:= Abs(Koeffizient[i, k]);
          Pivotzeile:= i;
        END;
      IF k <> Pivotzeile THEN {* wenn k=i, Pivotelement bereits günstig}
      FOR j:= 1 TO n + 1 DO {* Zeilen vertauschen}
        Exchange(Koeffizient[k, j], Koeffizient[Pivotzeile, j]);
    END;

  geeignet:= TRUE;       {* überprüfen, ob Gleichungssystem geeignet}
  i:= 1;
  REPEAT
    SZeile:= 0.0;
    FOR j:= 1 TO n DO
      IF i <> j THEN SZeile:= SZeile + Abs(Koeffizient[i, j]);
    IF SZeile = Abs(Koeffizient[i, i]) THEN geeignet:= FALSE;
    i:= i + 1;
  UNTIL (i  n) OR NOT geeignet;
  IF NOT geeignet THEN
  WriteLn('Gleichungssystem ungeeignet');
  WriteLn;

  FOR i:= 1 TO n DO
    Lösung[i] := 0;  {* erste Lösungsnäherung}

  Iteration:= 0;

  REPEAT
    Iteration:= Iteration + 1;
    Fertig:= TRUE;
    FOR i:= 1 TO n DO
      BEGIN
        Zähler:= Koeffizient[i, n + 1];
        FOR j:= 1 TO n DO
          IF i <> j THEN Zähler:= Zähler - Koeffizient[i, j]*Lösung[j];
        IF Koeffizient[i, i] = 0 THEN Abbruch;
        NeueLösung:= Zähler / Koeffizient[i, i];
        IF NeueLösung <> Lösung[i] THEN Fertig:= FALSE;
        Lösung[i] := NeueLösung;
      END;
  UNTIL Fertig OR (Iteration = MaxIterationen);
END;
```

3.2 Programmbeschreibung

Auch das letzte Programm befindet sich wieder im Anhang. Beim Betrachten des Sourcecodes erkennt man wieder den bekannte Programmaufbau und die Struktur von Gauss. Allerdings ersetzt in diesem Fall die Prozedur GaussSeidel gleich die zwei Prozeduren Eliminieren und RückwärtsEinsetzen. Außerdem gibt es noch eine Konstante MaxIterationen. Diese Konstante speichert die maximal zulässige Anzahl der Iterationen.
Der Rest ist mit den beiden vorherigen Programmtexten identisch.

3.3 Schlußbemerkung

Damit ist nun auch der letzte Teil der Facharbeit abgehandelt. Es wurde versucht die Algorithmen so knapp und genau wie möglich vorzustellen. Mit diesen Algorithmen sollten alle linearen NxN-Gleichungssysteme zu lösen sein.
Zuletzt soll noch geklärt werden, wann welches Verfahren eingesetzt werden soll.
Die Verfahren von Gauß und Jordan sind universell einsetzbar, d.h. sie finden immer eine Lösung. Besonders bei großen Gleichungssystemen steigt die Bearbeitungszeit allerdings rapide an, da beim Gauß'schen Eliminationsverfahren die Anzahl Arbeitsschritte für ein Gleichungssytem der Dimension

N mit $N^3/3$ ansteigt. Das Gauß-Jordan Verfahren benötigt sogar noch mehr Arbeitschritte, es arbeitet ungefähr anderthalb mal langsamer als das Verfahren von Gauß. Das Gauß'sche Verfahren ist eines der schnellsten Lösungsverfahren, es kann höchstens von iterativen, z.B. Gauß-Seidel (N^2-3), übertroffen werden.
Das Gauß-Seidel-Verfahren ist bei sehr großen Gleichungssystemen meistens das schnellste, wenn es funktioniert.
Neben diesen Algorithmen existieren natürlich noch viele weitere, wie z.B. Cramer'sche Regel oder LU-Dekomposition, die ihrerseits auch wieder Vor- und Nachteile haben.

Anhang

Anhang A: Programmtexte

Hier stehen die Programmtexte, die oben entwickelt wurden. Zusätzlich ist noch das Programm "GaußVergleich" aufgeführt, das in Anhang C zum Vergleichen der drei Verfahren verwendet wird. Dieses Programm verwendet die Unit Timer, die nicht aufgeführt ist. Diese Unit enthält lediglich Befehle zum Stoppen der Ausführungszeiten.

Gauss.p
GaussJordan.p
GaussSeidel.p
GaussVergleich.p

Anhang B: Programmablaufprotokolle

Es folgen einige Ablaufprotokolle der Programme Gauss, GaussJordan und GaussSeidel.
Die Eingaben des Benutzers werden doppelt unterstrichen gedruckt, alle anderen Ausgaben normal.
Das Löschen des Bildschirms wird durch ClrScr< angedeutet.
Außerdem wird die Koeffizienteneingabe nur im ersten Protokoll wiedergegeben, da dies auf Dauer zu viel Seiten verschwenden würde. Die Eingabe wird dann durch Koeffizienteneingabe< angedeutet.

```
Protokoll Gauss 1:

ClrScr
Anzahl der zu lösenden Gleichungen:2
ClrScr
Koeffizient[1,1]: 1
ClrScr
Koeffizient[1,2]: 2
ClrScr
Koeffizient[1,3]: 3
ClrScr
Koeffizient[2,1]: 4
ClrScr
Koeffizient[2,2]: 5
ClrScr
Koeffizient[2,3]: 6


Sollen die Rechenschritte angezeigt werden (J/N):n
ClrScr
Start-Gleichungen:
```

```
| (1) 1*x1 +  2*x2 =  3
| (2) 4*x1 +  5*x2 =  6
```

I. Vorwärtselimination

II. Rückwärtseinsetzen

```
| (1) x1 = -1
| (2) x2 =  2
```

Protokoll Gauss 2:

ClrScr
Anzahl der zu lösenden Gleichungen:4
ClrScr
Koeffizienteneingabe

Sollen die Rechenschritte angezeigt werden (J/N):j
ClrScr

Start-Gleichungen:

```
| (1) 20*x1 +  10*x2 +  3*x3 +  4*x4 =  5
| (2) 3*x1 +  5*x2 +  8*x3 +  2*x4 = 4
| (3) 3*x1 +  5*x2 +  9*x3 +  2*x4 =  8
| (4) 3*x1 +  10*x2 +  13*x3 +  12*x4 =  25
```

I. Vorwärtselimination

(2) - 0.15*(1):

```
| (1) 20*x1 +  10*x2 +  3*x3 +  4*x4 =  5
| (2) 0*x1 +  3.5*x2 +  7.55*x3 +  1.4*x4 =  3.25
| (3) 3*x1 +  5*x2 +  9*x3 +  2*x4 =  8
| (4) 3*x1 +  10*x2 +  13*x3 +  12*x4 =  25
```

(3) - 0.15*(1):

```
| (1) 20*x1 +  10*x2 +  3*x3 +  4*x4 =  5
| (2) 0*x1 +  3.5*x2 +  7.55*x3 +  1.4*x4 =  3.25
| (3) 0*x1 +  3.5*x2 +  8.55*x3 +  1.4*x4 =  7.25
| (4) 3*x1 +  10*x2 +  13*x3 +  12*x4 =  25
```

(4) - 0.15*(1):

```
| (1) 20*x1 +  10*x2 +  3*x3 +  4*x4 =  5
| (2) 0*x1 +  3.5*x2 +  7.55*x3 +  1.4*x4 =  3.25
| (3) 0*x1 +  3.5*x2 +  8.55*x3 +  1.4*x4 =  7.25
| (4) 0*x1 +  8.5*x2 +  12.55*x3 +  11.4*x4=  24.25
```

(2) mit (4) vertauscht:

| (1) 20*x1 + 10*x2 + 3*x3 + 4*x4 + = 5
| (2) 0*x1 + 8.5*x2 + 12.55*x3 + 11.4*x4 = 24.25
| (3) 0*x1 + 3.5*x2 + 8.55*x3 + 1.4*x4 = 7.25
| (4) 0*x1 + 3.5*x2 + 7.55*x3 + 1.4*x4 = 3.25
———————————————————————————————————————

(3) - 0.411765*(2):
| (1) 20*x1 + 10*x2 + 3*x3 + 4*x4 = 5
| (2) 0*x1 + 8.5*x2 + 12.55*x3 + 11.4*x4 = 24.25
| (3) 0*x1 + 0*x2 + 3.38235*x3 + -3.29412*x4 = -2.73529
| (4) 0*x1 + 3.5*x2 + 7.55*x3 + 1.4*x4 = 3.25
———————————————————————————————————————

(4) - 0.411765*(2):
| (1) 20*x1 + 10*x2 + 3*x3 + 4*x4 = 5
| (2) 0*x1 + 8.5*x2 + 12.55*x3 + 11.4*x4 = 24.25
| (3) 0*x1 + 0*x2 + 3.38235*x3 + -3.29412*x4 = -2.73529
| (4) 0*x1 + 0*x2 + 2.38235*x3 + -3.29412*x4 = -6.73529
———————————————————————————————————————

(4) - 0.704348*(3):
| (1) 20*x1 + 10*x2 + 3*x3 + 4*x4 = 5
| (2) 0*x1 + 8.5*x2 + 12.55*x3 + 11.4*x4 = 24.25
| (3) 0*x1 + 0*x2 + 3.38235*x3 + -3.29412*x4 = -2.73529
| (4) 0*x1 + 0*x2 + 0*x3 + -0.973913*x4 = -4.8087
———————————————————————————————————————

II. Rückwärtseinsetzen

| (1) 20*x1 + 10*x2 + 3*x3 + 4*x4 = 5
| (2) 0*x1 + 8.5*x2 + 12.55*x3 + 11.4*x4 = 24.25
| (3) 0*x1 + 0*x2 + 3.38235*x3 + -3.29412*x4 = -2.73529
| (4) x4 = (-4.80870) : -0.973913
———————————————————————————————————————

| (1) 20*x1 + 10*x2 + 3*x3 + 4*x4 = 5
| (2) 0*x1 + 8.5*x2 + 12.55*x3 + 11.4*x4 = 24.25
| (3) x3 = (-2.73529 - -3.29412* 4.9375) : 3.38235
| (4) x4 = 4.9375
———————————————————————————————————————

| (1) 20*x1 + 10*x2 + 3*x3 + 4*x4 = 5
| (2) x2 = (24.25 - 12.55* 4 - 11.4* 4.9375) : 8.5
| (3) x3 = 4
| (4) x4 = 4.9375
———————————————————————————————————————

| (1) x1 = (5 - 10*-9.675 - 3* 4 - 4* 4.9375) : 20
| (2) x2 = -9.675
| (3) x3 = 4
| (4) x4 = 4.9375
———————————————————————————————————————

| (1) x1 = 3.5
| (2) x2 = -9.675
| (3) x3 = 4
| (4) x4 = 4.9375
———————————————————————————————————————

Protokoll Gauss 3:

ClrScr
Anzahl der zu lösenden Gleichungen:3
ClrScr
Koeffizienteingabe

Sollen die Rechenschritte angezeigt werden (J/N):j
ClrScr
Start-Gleichungen:

| (1) 5*x1 + 8*x2 + 10*x3 = 7
| (2) 3*x1 + 5*x2 + 8*x3 = 2
| (3) 10*x1 + 16*x2 + 20*x3 = 4
_____

I. Vorwärtselimination

II. Rückwärtseinsetzen

Die Determinante ist 0 = es existiert keine eindeutige Lösung

Protokoll GaussJordan 1:

ClrScr
Anzahl der zu lösenden Gleichungen:4
ClrScr
Koeffizienteneingabe

Sollen die Rechenschritte angezeigt werden (J/N):j
ClrScr
  Start-Gleichungen:
| (1) 1*x1 + 2*x2 + 3*x3 + 4*x4 = 30
| (2) 2*x1 + 1*x2 + 4*x3 + 3*x4 = 28
| (3) 3*x1 + 4*x2 + 1*x3 + 2*x4 = 24
| (4) 4*x1 + 3*x2 + 2*x3 + 1*x4 = 20
_____

I. Vorwärtselimination

(1) mit (4) vertauscht:
| (1) 4*x1 + 3*x2 + 2*x3 + 1*x4 = 20
| (2) 2*x1 + 1*x2 + 4*x3 + 3*x4 = 28
| (3) 3*x1 + 4*x2 + 1*x3 + 2*x4 = 24
| (4) 1*x1 + 2*x2 + 3*x3 + 4*x4 = 30
_____

(2) - 0.5*(1):
| (1) 4*x1 + 3*x2 + 2*x3 + 1*x4 = 20
| (2) 0*x1 + -0.5*x2 + 3*x3 + 2.5*x4 = 18

```
| (3) 3*x1 +  4*x2 +  1*x3 +  2*x4 =  24
| (4) 1*x1 +  2*x2 +  3*x3 +  4*x4 =  30
```

```
(3) -  0.75*(1):
| (1) 4*x1 +  3*x2 +  2*x3 +  1*x4 =  20
| (2) 0*x1 + -0.5*x2 +  3*x3 +  2.5*x4 =  18
| (3) 0*x1 +  1.75*x2 + -0.5*x3 +  1.25*x4 =  9
| (4) 1*x1 +  2*x2 +  3*x3 +  4*x4 =  30
```

```
(4) -  0.25*(1):
| (1) 4*x1 +  3*x2 +  2*x3 +  1*x4 =  20
| (2) 0*x1 + -0.5*x2 +  3*x3 +  2.5*x4 =  18
| (3) 0*x1 +  1.75*x2 + -0.5*x3 +  1.25*x4 =  9
| (4) 0*x1 +  1.25*x2 +  2.5*x3 +  3.75*x4 =  25
```

```
(2) mit (3) vertauscht:
| (1) 4*x1 +  3*x2 +  2*x3 +  1*x4 =  20
| (2) 0*x1 +  1.75*x2 + -0.5*x3 +  1.25*x4 =  9
| (3) 0*x1 + -0.5*x2 +  3*x3 +  2.5*x4 =  18
| (4) 0*x1 +  1.25*x2 +  2.5*x3 +  3.75*x4 =  25
```

```
(3) - -0.285714*(2):
| (1) 4*x1 +  3*x2 +  2*x3 +  1*x4 =  20
| (2) 0*x1 +  1.75*x2 + -0.5*x3 +  1.25*x4 =  9
| (3) 0*x1 +  0*x2 +  2.85714*x3 +  2.85714*x4 =  20.5714
| (4) 0*x1 +  1.25*x2 +  2.5*x3 +  3.75*x4 =  25
```

```
(4) -  0.714286*(2):
| (1) 4*x1 +  3*x2 +  2*x3 +  1*x4 =  20
| (2) 0*x1 +  1.75*x2 + -0.5*x3 +  1.25*x4 =  9
| (3) 0*x1 +  0*x2 +  2.85714*x3 +  2.85714*x4 =  20.5714
| (4) 0*x1 +  0*x2 +  2.85714*x3 +  2.85714*x4 =  18.5714
```

```
(4) -  1*(3):
| (1) 4*x1 +  3*x2 +  2*x3 +  1*x4 =  20
| (2) 0*x1 +  1.75*x2 + -0.5*x3 +  1.25*x4 =  9
| (3) 0*x1 +  0*x2 +  2.85714*x3 +  2.85714*x4 =  20.5714
| (4) 0*x1 +  0*x2 +  0*x3 +  0*x4 = -2
```

II. Gauß-Jordan-Reduktion

Die Determinante ist 0 = es existiert keine eindeutige Lösung

Protokoll GaussJordan 2:

ClrScr
Anzahl der zu lösenden Gleichungen:4
ClrScr
Koeffizienteneingabe

Sollen die Rechenschritte angezeigt werden (J/N):n

```
ClrScr
  Start-Gleichungen:
| (1) 2*x1 +  3*x2 +  5*x3 +  7*x4 =  95
| (2) 3*x1 +  4*x2 +  10*x3 +  1*x4 =  132
| (3) 4*x1 +  2*x2 +  3*x3 +  10*x4 =  79
| (4) 7*x1 +  3*x2 +  2*x3 +  10*x4 =  82
```

```
I. Vorwärtselimination
```

```
II. Gauß-Jordan-Reduktion
```

```
| (1) x1 =  1
| (2) x2 =  9
| (3) x3 =  9
| (4) x4 =  3
```

```
Protokoll GaussJordan 3:

ClrScr
Anzahl der zu lösenden Gleichungen:2
ClrScr
Koeffizienteneingabe


Sollen die Rechenschritte angezeigt werden (J/N):Ja
ClrScr
  Start-Gleichungen:
| (1) 3*x1 +  27*x2 =  4
| (2) 2*x1 +  26*x2 =  0
```

```
I. Vorwärtselimination

(2) -  0.666667*(1):
| (1) 3*x1 +  27*x2 =  4
| (2) 0*x1 +  8*x2 = -2.66667
```

```
II. Gauß-Jordan-Reduktion

(1) -  3.375*(2):
| (1) 3*x1 +  0*x2 =  13
| (2) 0*x1 +  8*x2 = -2.66667
```

```
| (1) x1 =  13 :  3
| (2) x2 = -2.66667 :  8
```

```
| (1) x1 =  4.33333
| (2) x2 = -0.333333
```

```
_____________________________________________

Protokoll GaussSeidel 1:

ClrScr
Anzahl der zu lösenden Gleichungen:2
ClrScr
Koeffizienteneingabe


Sollen die Rechenschritte angezeigt werden (J/N):n
ClrScr
  Start-Gleichungen:
| (1) 6*x1 +  5*x2 =  4
| (2) 3*x1 +  2*x2 =  1
_____________________________________________

Gleichungssystem für Gauß-Seidel ungeeignet - wahrscheinlich keine Konvergenz


 Nach 100 Iterationen:
| (1) x1 =  6.54547E+09
| (2) x2 = -9.81820E+09
_____________________________________________

Protokoll GaussSeidel 2:

ClrScr
Anzahl der zu lösenden Gleichungen:2
ClrScr
Koeffizienteneingabe


Sollen die Rechenschritte angezeigt werden (J/N):j
ClrScr
  Start-Gleichungen:
| (1) 10*x1 +  2*x2 =  9
| (2) 3*x1 +  7*x2 =  6
_____________________________________________


  Iteration 1:
x1 = ( 9 -  10* 0 -  2* 0) :  10  =  0.9
x2 = ( 6 -  3* 0.9 -  7* 0) :  7  =  0.471429
```

```
  Iteration 2:
x1 = ( 9 -  10* 0.9 -  2* 0.471429) :  10  =  0.805714
x2 = ( 6 -  3* 0.805714 -  7* 0.471429) :  7  =  0.511837

  Iteration 3:
x1 = ( 9 -  10* 0.805714 -  2* 0.511837) :  10  =  0.797633
x2 = ( 6 -  3* 0.797633 -  7* 0.511837) :  7  =  0.5153

  Iteration 4:
x1 = ( 9 -  10* 0.797633 -  2* 0.5153) :  10  =  0.79694
x2 = ( 6 -  3* 0.79694 -  7* 0.5153) :  7  =  0.515597

  Iteration 5:
x1 = ( 9 -  10* 0.79694 -  2* 0.515597) :  10  =  0.796881
x2 = ( 6 -  3* 0.796881 -  7* 0.515597) :  7  =  0.515623

  Iteration 6:
x1 = ( 9 -  10* 0.796881 -  2* 0.515623) :  10  =  0.796876
x2 = ( 6 -  3* 0.796876 -  7* 0.515623) :  7  =  0.515625

  Iteration 7:
x1 = ( 9 -  10* 0.796876 -  2* 0.515625) :  10  =  0.796875
x2 = ( 6 -  3* 0.796875 -  7* 0.515625) :  7  =  0.515625

  Iteration 8:
x1 = ( 9 -  10* 0.796875 -  2* 0.515625) :  10  =  0.796875
x2 = ( 6 -  3* 0.796875 -  7* 0.515625) :  7  =  0.515625
```

```
Protokoll GaussSeidel 3:

ClrScr
Anzahl der zu lösenden Gleichungen:5
ClrScr
Koeffizienteneingabe


Sollen die Rechenschritte angezeigt werden (J/N):n
ClrScr

  Start-Gleichungen:
| (1) 60*x1 +  2*x2 +  3*x3 +  4*x4 +  5*x5 =  80
| (2) 3*x1 +  45*x2 +  3*x3 +  4*x4 +  6*x5 = -10
| (3) 3*x1 +  8*x2 +  5*x3 +  65*x4 +  4*x5 =  16
| (4) 2*x1 +  4*x2 +  49*x3 +  3*x4 + -4*x5 =  69
| (5) 2*x1 +  4*x2 +  9*x3 +  3*x4 +  96*x5 =  0.5
-------------------------------------------------


 Nach 7 Iterationen:
| (1) x1 =  1.28025
| (2) x2 = -0.39275
| (3) x3 =  1.36824
| (4) x4 =  0.138629
| (5) x5 = -0.137704
```

Anhang C: Vergleich der Algorithmen

In diesem Abschnitt werden die drei Algorithmen miteinander verglichen. Dazu werden mit GaussVergleich jeweils die Zeiten ermittelt, die die drei Algorithmen für das Lösen eines eingebenen Gleichungssystems benötigen.
Dabei ist zu beachten, daß nur solche Gleichungssysteme zum Test herangezogen werden können, die auch für das Gauß-Seidel-Verfahren geeignet sind.
Bei den Zeiten muß man berücksichtigen, daß sie mit der internen Uhr eines COMMODORE AMIGA 500 gemessen wurden, der mit einem 68000er Prozessor bestückt und mit 7.14 MHz getaktet war. Bei anderen Computern können sich natürlich andere Zeiten ergeben.
Wichtig sind allerdings nicht die absoluten Zeitwerte, sondern die Verhältnisse zueinander.
In der folgenden Tabelle sind einige Testwerte zusammengefaßt:

Dimension	Gauß/s	GaußJordan/s	GaußSeidel/s
1	0	0	0.04
5	0.02	0.02	0.06
10	0.14	0.22	0.26
15	0.44	0.72	0.56
20	0.94	1.64	0.96
25	1.78	3.08	1.48
30	3.02	5.22	2.3
50	13.02	23.46	5.64
70	34.74	62.4	10.94
100	99.46	182.36	26.7

Man kann also sehr deutlich erkennen, daß das Gauß'sche Verfahren bei Gleichungssystemen bis etwa zur Dimension 20 das schnellste ist. Während der Gauß-Seidel-Algorithmus bis dahin immer der langsamste war, setzt er sich nun von anderen deutlich ab. Bei der Dimension 100 benötigt das Iterationsverfahren ein fünftel weniger Zeit als das Eliminationsverfahren, und etwa ein neuntel der Zeit als das Reduktionsverfahren.
Das Gauß-Jordan Verfahren ist von Anfang an langsamer als das Gauß'sche, allerdings vergrößter sich der Abstand auch noch erheblich.
Bei großen Matrizen ist also das Gauß-Seidel Verfahren zu empfehlen, allerdings wird dieses Verfahren viele Gleichungssystem aufgrund mangelnder Konvergenz nicht lösen können. Folglich kommt man in diesem Falle am Gauß'schen Algorithmus nicht vorbei. Das Gauß-Jordan Verfahren ist ledig
lich sinnvoll, um die Ergebnisse eines anderen Algorithmuses zu überprüfen.
Im folgendem Diagramm werden die Zeiten in y-, die Dimensionen n in x-Richtung angetragen:

Aus diesem Diagramm erkennt man nun, daß das Gauß-Seidel-Verfahren, den Gauß-Jordan Algorithmus etwa bei der Dimension n=10, das Gauß'sche Verfahren erst bei n=20 einholt.
Bei n=50 ist das iterative Verfahren fast um das doppelte schneller als das Eliminationsverfahren, und mehr als dreimal schneller als das Reduktionsverfahren.
Weiter kann man in dem Diagramm an dem unruhigen Verlauf der Kurve des Gauß-Seidel-Verfahrens, erkennen, daß für dieses Verfahren die Zeiten sehr stark schwanken, da die Konvergenz und somit

die Geschwindigkeit sehr stark von der Konditionierung des Gleichungssystems abhängt. Bei den beiden anderen Verfahren dagegen hängt die Arbeitszeit fast nur von der Dimension des Gleichungsverfahren ab, d.h. die Arbeitszeit für verschiedene Gleichungssysteme der Dimension n ist etwa gleich.

Anhang D: Bibliographische Daten

Miller, A. R.: PASCAL PROGRAMME. Mathematik Statistik Informatik,
Berkley; Düsseldorf,
SYBEX-Verlag 19863,
S. 73-94 u. S. 129-138

Prey, W.; Flannery, B.; Tuckolsky, S.; Veterling, W.:
Numerical Recipes. The Art of scientific computing,
Cambridge University Press 1986,
S. 24-31

Sedgewick, R.: Algorithmen,
Bonn; München; Reading,
Addision-Wesley 1991,
S. 607-616

Zurmühl, R.: Praktische Mathematik für Ingenieure und Physiker,
Berlin,
Springer-Verlag 19655,
S. 106-112 u. S. 157-161

Ich erkläre hiermit, daß ich die Facharbeit ohne fremde Hilfe angefertigt und nur die im Literaturverzeichnis angeführten Quellen und Hilfsmittel benützt habe.

_____ , den _____
Ort Datum

_____
Unterschrift des Schülers

_____, den _____ _____
Ort Datum Unterschrift des Schülers