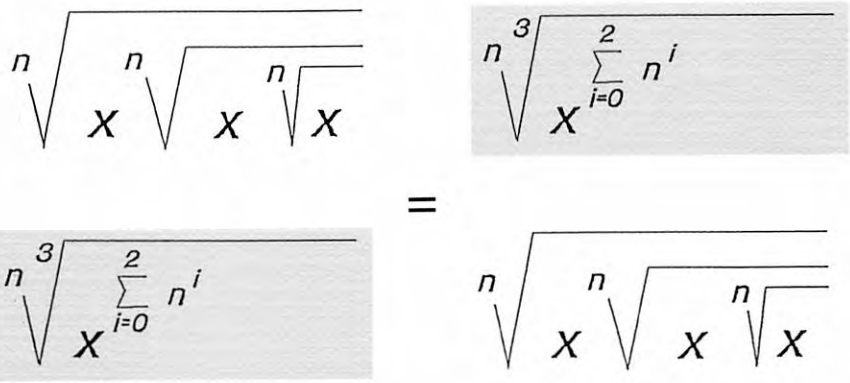
COMPUTER-PRAXIS MATHEMATIK

Wolfgang Drabe

Numerische Mathematik

● Grundlagen ● Algorithmen



● Maclaurinsche Reihen ● Interpolationspolynome
● Numerisches Differenzieren und Integrieren
● Differentialgleichungen ● Regula falsi
● Newtonsches Iterationsverfahren

DÜMMLER

COMPUTER-PRAXIS MATHEMATIK
Herausgegeben von StD Pohlmann

Wolfgang Drabe

Numerische Mathematik

● Grundlagen ● Algorithmen

● Maclaurinsche Reihen ● Interpolationspolynome
● Numerisches Differenzieren und Integrieren
● Differentialgleichungen ● Regula falsi
● Newtonsches Iterationsverfahren

Unter Mitarbeit von Thomas Reuther.
Mit zahlreichen Aufgaben und
deren Lösungen.
Dümmlerbuch 4542

FERD. DÜMMLERS VERLAG · BONN

Zu diesem Lehr- und Arbeitsbuch für die Sekundarstufe II gibt es bei Dümmler folgende 3,5"-HD-Diskette für IBM-PC und Kompatible (MS-DOS-Rechner)
mit Quelltexten der Beispiele und Lösungen in Turbo-Pascal (Version 4.0 bis 7.0) 3,5"-HD-Diskette. DM 36,– (Dümmlerbuch 45432)

KROLL[S] Analysis Alternative für Grund- und Leistungskurse

Von Prof. W. KROLL und Dr. J. VAUPEL.

- **Dieses völlig neu konzipierte Lehr- und Arbeitsbuch vermehrt nicht die lange Liste ähnlicher Bücher, sondern bringt einen neuen, schülergerechten Ansatz. KROLL-VAUPEL[S] neue Schülerarbeitsbücher zeichnen sich durch besondere Vorzüge inhaltlicher, methodischer und organisatorischer Art aus.**
- Eine Fülle neuer Ideen, Inhalte und Aufgaben sind in beiden Bänden verwirklicht, die wichtige meth.-didakt. Weiterentwicklung bedeuten.
- Beide Bände bringen einen Kernstoff für Grundkurse sowie Auswahlstoffe für Leistungskursniveau (im Vorwort genau beschrieben u. begründet).
- Die Probleme werden ausführlich erklärt und Lösungsmethoden schrittweise entwickelt. Varianten und verschiedene Präzisierungsstufen sind in den Text eingearbeitet oder werden in Aufgabenform angeboten.
- Beispiele und Aufgaben sind in großer Anzahl vorhanden. Viele Aufgaben sind neuartig, anwendungsbezogen, fächerübergreifend.
- Das Layout ist ruhig und großzügig. Die verschiedenen Textelemente (Lehrtext, Merksätze, Aufgaben usw.) heben sich wohltuend voneinander ab und bestechen durch Übersichtlichkeit. Eine Fülle großformatiger, instruktiver Abbildungen trägt wesentlich zur Veranschaulichung der Stoffe bei.
- Kurzum – ein äußerlich ansprechendes und inhaltlich abgerundetes Buch. Einführungsreif, auch an Ihrer Schule.
- All das reizte die Lehrer zunächst zur unterrichtsbegleitenden Lektüre, dann zu probeweisen Einführungen.
- Diese geistige Auseinandersetzung, jede für sich allein, im Fachkollegium oder im Fachseminar, ist zugunsten des KROLL-VAUPEL gelaufen:
- Inzwischen läuft die Welle der Einführungen. Beide Bände liegen schon in 2., durchgesehener Auflage vor, voll kompatibel zur 1. Auflage.

Band 1: Differentialrechnung 1

224 Seiten. 135 Abb. Format 18,2 x 25,3 cm. DM 32,80. 2. Aufl. 1988. (Dümmlerbuch 4281)

Band 2: Integralrechnung und Differentialrechnung 2

248 Seiten. 212 Abb. Format 18,2 x 25,3 cm. DM 32,80. 2. Aufl. '89. (Dümmlerbuch 4282)

Analytische Geometrie / Lineare Algebra

Von W. KROLL / H. REIFFERT / J. VAUPEL.
204 Seiten. 183 Abbildungen, 547 Übungs- u. z.T. neuartige Anwendungs-Aufgaben. Format 18,2 x 25,3 cm. DM 34,80. 1997. (Dümmlerbuch 4285)

Lösungsbände

Sie enthalten neben meth.-didakt. Hinweisen vor allem die vollständigen Lösungen fast aller Aufgaben, jeweils beider Auflagen der Schülerbände, passen also jeweils zur 1. und 2. Auflage.

- **Lehrerband 1.** IV, 215 Seiten. 225 Abb. 85 Tab. 2. Aufl. 1994. DIN A5. DM 32,80 (Dümmlerbuch 4283)
- **Lehrerband 2.** 240 Seiten. 208 Abb. 34 Tab. 2. Auflage. DIN A5. 1997. DM 32,80 (Dümmlerbuch 4284)
- **Lehrerband 3.** 272 Seiten. 102 Abb. DIN A5. 1998. DM 36,80 (D'buch 4286)

Weitere Titel und Informationen zu Mathematik sowie den Reihen COMPUTER-PRAXIS MATHEMATIK und BAUSTEINE INFORMATIK enthält unser Prospekt (9900 P), den wir Ihnen gerne auf Anforderung zusenden.

ISBN 3-427-**45421**-5 (Preis-)Änderungen vorbehalten.

Printed in Germany by WB-Druck, 87669 Rieden

Vorwort

Das vorliegende Buch ist gedacht für Schüler der gymnasialen Oberstufe, für Studienanfänger im mathematisch-naturwissenschaftlichen, technischen oder wirtschaftswissenschaftlichen Bereich sowie für Personen, die sich im Selbststudium Kenntnisse zu den hier angesprochenen Themenbereichen verschaffen wollen. Die Inhalte dieses didaktisch aufbereiteten, auf Unterrichtserfahrungen aufbauenden Buches orientieren sich an den Rahmenrichtlinien und Lehrplänen der gymnasialen Oberstufe für das Fach Mathematik. Dabei wird häufig im Zusammenhang mit der Analysis als Erweiterung vorgeschlagen:

- Folgen und Reihen (insbesondere Taylorreihen)
- numerische Differentiation und Integration
- näherungsweise Bestimmung von Nullstellen, Iterationsverfahren etc.

Ferner sei erwähnt, daß die exakten Lösungen mathematischer Probleme nicht in jedem Falle im Vordergrund stehen, denn

- numerische Ergebnisse gehören häufig zur Menge der irrationalen Zahlen, deren praktisch verwertbare Größen nur (mit Digitalrechnern beliebig genau errechenbare) Näherungswerte sein können,
- inzwischen weiterentwickelte Verfahren der numerischen Mathematik „ersetzen" häufig aufwendige und zeitraubende Herleitungen exakter Lösungen.

Dieses Buch will auch Antworten auf häufig gestellte Schülerfragen geben wie z.B. die Frage

- nach der prinzipiellen Arbeitsweise von Taschenrechnern bei der Ermittlung von Funktionswerten
- auf welche Weise bei exakt nicht lösbaren oder ungelösten Problemen wie etwa unbekannte Stammfunktionen von Integralen oder Differentialgleichungen Ergebnisse erzielt werden können.

Zu diesem Lehrbuch ist auch eine Diskette erhältlich, die die Lösungswege der Problemstellungen in Form von Turbo-Pascal-Programmen beinhaltet.

Die Autoren

Inhaltsverzeichnis

1. Polynomdarstellung analytischer Funktionen

Die vor allem in den letzten beiden Jahrzehnten erzielten Fortschritte auf dem Gebiet der Elektronik haben es ermöglicht, dickleibige Tabellenbücher durch Taschenrechner zu ersetzen. In diesem Zusammenhang erhebt sich die Frage, in welcher Weise Taschenrechner zu Funktionswerten gelangen. Hierzu bieten sich zwei grundsätzlich verschiedene Verfahren an:

- Es werden die abrufbaren Funktionswerte nach einer vom Hersteller vorgenommenen Eingabe dauerhaft gespeichert. Gegen dieses Verfahren spricht vor allem der enorme Speicherbedarf eines Taschenrechners, der dennoch handlich sein soll.
- Der jeweils gewünschte Funktionswert wird nach Eingabe des Arguments berechnet. In der Tat liegt dieses Verfahren in der Regel der Arbeitsweise eines Taschenrechners zu Grunde. Allerdings hat dazu jede Firma ihre Geheimnisse. Wir werden im folgenden allgemeine Prinzipien und Möglichkeiten dazu aufzeigen.

1.1 Berechnung von Funktionswerten über Polynome

Der englische Mathematiker B. Taylor (1685 - 1731) entwickelte eine nach ihm benannte Reihe, die dazu geeignet ist Funktionen in Form von Potenzreihen darzustellen:

$$f(a+x) = f(a) + \frac{f'(a)}{1!}x + \frac{f''(a)}{2!}x^2 + \ldots + \frac{f^{(n)}(a)}{n!}x^n + R_n$$

R_n stellt dabei das Restglied dar. Die Reihe konvergiert für den Fall, daß der Grenzwert $\lim_{n\to\infty} R_n = 0$ wird. Die Taylorsche Reihe läßt sich immer dann darstellen, wenn die Funktion beliebig oft differenzierbar ist.

Wird a=0 gesetzt, dann bezeichnet man eine solche Taylorsche Reihe als Maclaurinsche Reihe (nach C. Maclaurin [1698 - 1748]):

$$f(x) = f(0) + \frac{f'(0)}{1!}x + \frac{f''(0)}{2!}x^2 + \ldots + \frac{f^{(n)}(0)}{n!}x^n + \ldots$$

Die Herleitung der Maclaurinschen Reihe beginnt mit der Potenzreihe als mögliche Darstellungsform einer Funktion. Die Koeffizienten gewinnt man in der nachfolgend angegebenen Weise. Ausgehend von

$$f(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n = \sum_{i=0}^{n} a_i x^i$$

erhält man für

$$\begin{aligned} f'(x) &= a_1 + 2a_2 x + 3a_3 x^2 + \ldots + n a_n x^{n-1} \\ f''(x) &= 2a_2 + 6a_3 x + \ldots + (n-1) n a_n x^{n-2} \\ f'''(x) &= 6a_3 + 24a_4 x + \ldots + (n-2)(n-1) n a_n x^{n-3} \\ &\vdots \end{aligned}$$

Daraus folgt für $f(0) = a_0$, $f'(0) = a_1$, $f''(0) = 2a_2$, ...

$$f^{(n)}(0) = [\prod_{i=1}^{n} i]\, a_n = n!\, a_n$$

$$\Longleftrightarrow \quad a_n = \frac{f^{(n)}(0)}{n!}$$

Nach entsprechender Ersetzung der Koeffizienten erhält man

$$f(x) = f(0) + f'(0)x + \frac{f''(0)}{2!}x^2 + \ldots + \frac{f^{(n)}(0)}{n!}x^n$$

Allgemein ausgedrückt also

$$\boxed{f(x) = \sum_{i=0}^{n} \frac{f^{(i)}(0)}{i!}\, x^i}$$

Anhand einiger Beispiele soll die Überführung von Funktionen in Maclaurinsche Reihen demonstriert werden.

1. Beispiel: Ausgehend von der ganzrationalen Funktion

$$y = 5x^4 + 2x^3 - 6x - 10$$

ergeben sich folgende Ableitungen:

$$\begin{aligned} y' &= 20x^3 + 6x^2 - 6 \\ y'' &= 60x^2 + 12x \\ y''' &= 120x + 12 \\ y^{(4)} &= 120 \\ y^{(5)} &= 0 = y^{(6)} = y^{(n)} \end{aligned}$$

und somit folgende Funktionswerte: $f(0) = -10$, $f'(0) = -6$, $f''(0) = 0$, $f'''(0) = 12$, $f^{(4)}(0) = 120$, $f^{(5)}(0) = 0, \ldots, f^{(n)}(0) = 0$. Eingesetzt in das Polynom

$$y = f(0) + \frac{f'(0)}{1!}x + \frac{f''(0)}{2!}x^2 + \frac{f'''(0)}{3!}x^3 + \frac{f^{(4)}(0)}{4!}x^4 + \ldots$$

resultiert daraus $y = -10 - 6x + 2x^3 + 5x^4$.

2. Beispiel: Ausgehend von der gebrochenrationalen Funktion

$$y = \frac{x^3 - 8}{x - 2}$$

ergeben sich folgende Ableitungen

$$\begin{aligned} y' &= \frac{3x^2(x-2) - (x^3-8)^2}{(x-2)} = \frac{3x^3 - 6x^2 - x^3 + 8^2}{(x-2)} \\ &= \frac{2x^3 - 6x^2 + 8}{(x-2)^2} \\ y'' &= \frac{(6x^2 - 12x)(x-2)^2 - 2(x-2)(2x^3 - 6x^2 + 8)}{(x-2)^4} \end{aligned}$$

$$= \frac{6x^3 - 12x^2 - 12x^2 + 24x - 4x^3 + 12x^2 - 16}{(x-2)^3}$$

$$= \frac{2x^3 - 12x^2 + 24x - 16}{x^3 - 6x^2 + 12x - 8} = 2$$

$$y^{(n)} = 0 \quad \text{für} \quad n > 2 \quad \wedge \quad n \in N$$

und somit folgende Funktionswerte: $f(0) = 4$, $f'(0) = 2$, $f''(0) = 2$, $f^{(n)}(0) = 0$ für $n > 2 \wedge n \in N$

Eingesetzt in das Polynom

$$f(x) = f(0) + \frac{f'(0)}{1!}x + \frac{f''(0)}{2!}x^2$$

resultiert daraus für $x \neq 2$

$$\boxed{y = 4 + 2x + x^2}$$

Der Beweis für die Gültigkeit des ermittelten Polynoms kann durch die Polynomdivision des Funktionsterms erbracht werden:

$$\begin{array}{l} (x^3 - 8) : (x-2) = x^2 + 2x + 4 \\ \underline{-(x^3 - 2x^2)} \\ \quad -8 + 2x^2 \\ \underline{-(2x^2 - 4x)} \\ \quad -8 + 4x \\ \underline{-(-8 + 4x)} \\ \quad 0 \end{array}$$

3. Beispiel: Ausgehend von der Funktion $y = sin\ x$ ergeben sich folgende Ableitungen und Funktionswerte:

$$\begin{array}{llll} y & = & sin\ x & f(0) = 0 \\ y' & = & cos\ x & f'(0) = 1 \\ y'' & = & -sin\ x & f''(0) = 0 \\ y''' & = & -cos\ x & f'''(0) = -1 \\ y^{(4)} & = & sin\ x & f^{(4)}(0) = 0 \\ y^{(5)} & = & cos\ x & f^{(5)}(0) = 1 \text{ etc.} \end{array}$$

Eingesetzt in das Maclaurinsche Polynom resultiert daraus

$$sin\ x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + - \ldots$$

4. Beispiel: Ausgehend von der Funktion $y = ln(x+1)$ ergeben sich folgende Ableitungen

$$y' = \frac{1}{x+1} \qquad y'' = -\frac{1}{(x+1)^2} \qquad y''' = \frac{2}{(x+1)^3}$$

$$y^{(4)} = -\frac{3!}{(x+1)^4} \qquad y^{(5)} = \frac{4!}{(x+1)^5} \qquad \text{etc.}$$

und somit folgende Funktionswerte: $f(0) = 0$, $f'(0) = 1$, $f''(0) = -1$, $f'''(0) = 2$, $f^{(4)}(0) = -3!$, $f^{(5)}(0) = 4!$ etc. Eingesetzt in das Maclaurinsche Polynom resultiert daraus

$$ln(x+1) = 0 + x - \frac{1!}{2!}x^2 + \frac{2!}{3!}x^3 - \frac{3!}{4!}x^4 + \frac{4!}{5!}x^5 - + \ldots$$

$$ln(x+1) = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + - \ldots$$

Diese Reihe konvergiert für das halboffene Intervall]-1 ; 1].

1.1.1 Rekursiv definierte Verfahren

Vergleicht man die Glieder einer der letzten beiden Reihen für $sin\ x$ bzw. $ln(x+1)$ miteinander, so erkennt man eine gewisse „Regelmäßigkeit". Bei genauerer Betrachtung stellt man fest, daß sich das jeweils nachfolgende Glied aus dem vorherigen nach einer bestimmten Vorschrift berechnen läßt. Somit handelt es sich bei den Gliedern dieser Reihen um rekursive Folgen. Auf diese Weise lassen sich z.B. die Glieder der Reihe für

$$sin\ x = x \sum_{i=1}^{n} q_{(2i-1)}$$

durch folgenden Term als rekursive Bildungsvorschrift

$$q_{(2i+1)} = -\frac{x^2}{2i(2i+1)} q_{(2i-1)}$$

mit $i \epsilon N$ und $q_1 = 1$ als Anfangswert ermitteln. Allgemein ergibt sich für die Glieder einer Maclaurinschen Reihe als rekursives Bildungsgesetz folgender Term:

$$q_i = q_{i-1} \frac{\frac{f^{(i)}(0)}{i!} x^i}{\frac{f^{(i-1)}(0)}{(i-1)!} x^{i-1}} \qquad \text{bzw.}$$

$$\boxed{q_i = q_{i-1} \frac{f^{(i)}(0)}{i\, f^{(i-1)}(0)} x}$$

Ergibt sich für q_i ein negativer Wert, dann resultiert daraus eine alternierende Reihe. Ferner ist zu beachten, daß falls jedes zweite Glied Null wird, diese Glieder bei der Ermittlung des Terms für q_i übergangen werden. Aus den Gliedern der Folge $< q_i >$ ergibt sich schließlich die Reihe

$$\boxed{f(x) = f(0) + \sum_{i=1}^{n} q_i}$$

mit $i \epsilon N$ und $q_0 = f(0)$ als Anfangswert.

Die algorithmische Darstellung dieses Sachverhalts sieht folgendermaßen aus, wobei für das Argument a der (angenäherte) Funktionswert f(a) ermittelt werden soll:

Anfangswerte:

$x := a$, $q_0 := f(0)$, $s_0 := f(0)$

Wiederhole von $i = 1$ bis $n \epsilon N$:

$$q_i := q_{i-1} \frac{f^{(i)}(0)\, a}{i\, f^{(i-1)}(0)}$$

$$s_i := s_{i-1} + q_i$$

Der Funktionswert ist dann schließlich $f(a) \approx s_n$. Zu den (in einfacher Weise) rekursiv berechenbaren elementaren Funktionen gehören Exponentialfunktionen, einige logarithmische Funktionen (z.B. $ln(1+x)$), einige trigonometrische Funktionen ($sin\, x$ und $cos\, x$), zyklometrische Funktionen, Hyperbelfunktionen und Areafunktionen.

Bei den Maclaurinschen Reihen für $tan\, x$ und $cot\, x$ lassen sich die Koeffizienten unter Einbeziehung der rekursiv berechenbaren Bernoullischen Zahlen ebenfalls rekursiv berechnen.

Als Beispiel für einen Algorithmus rekursiv berechenbarer Funktionswerte sei die Berechnung von $sin\, a$ genannt:

Anfangswerte (im Bogenmaß): $q_0 := a$, $s_0 := a$

Wiederhole von $i = 1$ bis $n \epsilon N$:

$$q_i := -q_{i-1} \frac{q^2}{2i\,(2i+1)}$$

$$s_i := s_{i-1} + q_i$$

Hierbei erhält man bereits bei n=35 für $sin\, a$ 10 signifikante Stellen hinter dem Komma!

In Turbo-Pascal lassen sich Rekursionen mit Hilfe von Funktionen oder Prozeduren realisieren, die sich selber aufrufen. Diese Tatsache wird im Folgenden an zwei einfachen Beispielen demonstriert:

Rekursive Berechnung der Fakultät:

$$0! = 1$$
$$n! = (n-1)!\,n$$

Turbo-Pascal- Programm:

```
PROGRAM fakultaet; (* auf der Diskette als fak.pas *)
uses crt;
VAR n : REAL; (* REAL, weil Wertebereichüberschreitung
                 bei INTEGER *)

  FUNCTION fak (n: REAL): REAL;
    BEGIN
      IF n = 0 THEN fak := 1
      ELSE fak := fak(n-1) * n
    END;

  BEGIN
  CLRSCR;
  WRITELN('Fakultätsberechnung');
  WRITE('Bitte n eingeben: ');
    READLN(n);
  WRITELN('n! = ',fak(n):4:0)
  END.
```

Rekursive Berechnung der Binomialkoeffizienten:

$$\binom{n}{0} = 1$$

$$\binom{n}{i} = \binom{n}{i-1}\frac{n-i+1}{i}$$

Turbo-Pascal- Programm:

```
PROGRAM binomialkoeffizient;
(* auf Diskette als binomi.pas *)
uses crt;
VAR n,i : REAL; (* REAL, weil Wertebereichüberschreitung
                   bei INTEGER *)
```

```
FUNCTION binkoeff(i:REAL):REAL
  BEGIN
    IF i=0 THEN binkoeff := 1
    ELSE binkoeff := binkoeff(i-1) * (n-i+1)/i
  END;

BEGIN
  CLRSCR;
  WRITELN('Berechnung von Binomialkoeffizienten: ');
  WRITE('Bitte n eingeben: '); READLN(n);
  WRITE('Bitte i eingeben: '); READLN(i);
  WRITELN(binkoeff(i):4:0)
END.
```

Ein weiteres Beispiel hierfür, das jedoch auf einem komplizierteren Algorithmus beruht, ist die in Abschnitt 2.3.2 enthaltene Koeffizientenberechnung mit Hilfe des Vietaschen Wurzelsatzes.

1.1.2 Das Hornerschema als alternatives Berechnungsverfahren

Läßt sich für die Entwicklung eines Polynoms keine oder keine einfach zu entwickelnde Rekursionsvorschrift ermitteln, dann besteht die Möglichkeit zumindest mit Hilfe der Hornerzerlegung die Berechnung der Potenzen zu umgehen. Voraussetzung hierfür ist, daß die Koeffizienten bis zum (je nach gewünschter Genauigkeit des Funktionswertes) n-ten Glied der Reihe entweder bekannt oder berechnet worden sind. Dieses Verfahren beginnt im Gegensatz zu den zuvor erwähnten Rekursionen immer mit dem Koeffizienten der höchsten Potenz. Im Folgenden wird zunächst das sogenannte Hornerschema für den Fall n=4 vorgestellt.

$$\begin{aligned} f(x) &= a_0 + a_1x + a_2x^2 + a_3x^3 + a_4x^4 \\ &= a_0 + x(a_1 + a_2x + a_3x^2 + a_4^3) \\ &= a_0 + x(a_1 + x(a_2 + a_3x + a_4x^2)) \\ &= a_0 + x(a_1 + x(a_2 + x(a_3 + a_4x))) \end{aligned}$$

Wenn wir im Inneren des Klammernsystems beginnen, ergibt sich hieraus der Zyklus von $k = n$ bis 1 und $k \epsilon N$:

a_k mit x multiplizieren

a_{k-1} zu $x\,a_k$ hinzuaddieren

Der Algorithmus des Hornerschemas für $y = f(x)$ lautet bei $x = x_0$ wie folgt:

Anfangswerte: $x := x_0$, $y := a_n$

Wiederhole von $k = n - 1$ bis 0 mit $k \epsilon N_0$:

$$y := y * x + a_k$$

Als Beispiel diene die angenäherte Berechnung von e mittels des Grenzwertes

$$\lim_{n\to\infty}(1 + \frac{1}{n})^n$$

Anfangswert:

$$e := \binom{n}{n} = 1$$

Wiederhole von k = n-1 bis 0 mit $k \epsilon N_0$

$$e := e\frac{1}{n} + \binom{n}{k}$$

Durch die Anwendung des binomischen Lehrsatzes ergibt sich für $(1+\frac{1}{n})^n$ die Reihe

$$\binom{n}{0} + \binom{n}{1} * \frac{1}{n} + \binom{n}{2} * (\frac{1}{n})^2 + ... + \binom{n}{n} * (\frac{1}{n})^n .$$

Die Berechnung der Potenzen von $\frac{1}{n}$ läßt sich übrigens mit Hilfe des Hornerschemas umgehen:

	$\binom{n}{n}$	$\binom{n}{n-1}$	...	$\binom{n}{0}$
		$\frac{1}{n} * \binom{n}{n}$	...	...
$x = \frac{1}{n}$	$\binom{n}{n}$	$\binom{n}{n-1} + \frac{1}{n}\binom{n}{n}$	...	... = e

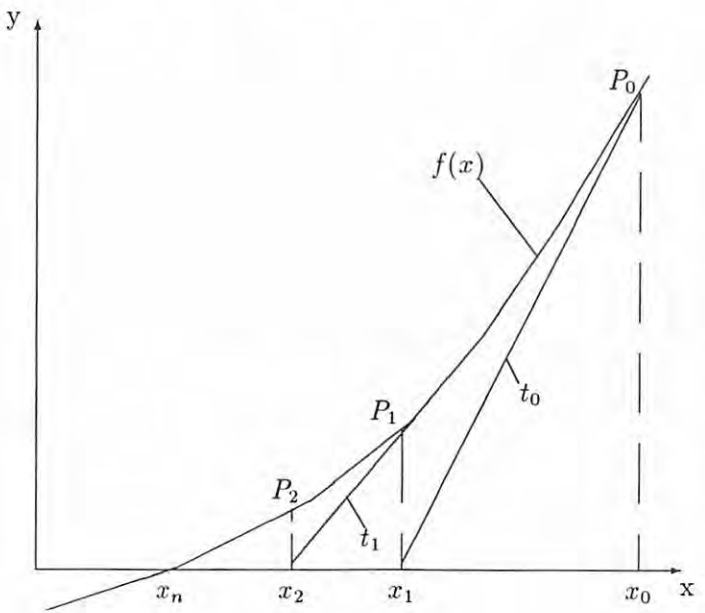
Das zugehörige Programm mit der Bezeichnung e_zahl_1.pas ist auf der Begleitdiskette enthalten.

1.2 Gegenüberstellung von Rekursionen und Iterationen

Es sei noch erwähnt, daß Rekursionen von Iterationen zu unterscheiden sind. Obwohl in beiden Fällen ein- und derselbe Rechenvorgang auf den jeweils zuvor berechneten Wert angewendet wird, handelt es sich jedoch bei Iterationen um ein Verfahren der wiederholten Verbesserung eines Näherungswertes, (Verfahren der sukzessiven Approximation). Hierbei wird eine Näherungsfolge erzeugt, wobei jede Annäherung an den gesuchten Wert als Ausgangspunkt für eine weitere Näherung dient.

1.2.1 Das Newtonsche Iterationsverfahren

Obwohl nach dem Fundamentalsatz der Algebra (C.F. Gauß, 1799) algebraische Gleichungen n-ten Grades ($n \epsilon N$) n Lösungen besitzen, lassen sich, von Sonderfällen abgesehen, Gleichungen 5. und höheren Grades nachgewiesenermaßen (H. Abel, 1824) nicht mehr exakt lösen. Um dennoch zumindest angenäherte Lösungen zu erhalten, verwendet man für diesen Zweck häufig das Newtonsche Iterationsverfahren. Zur Erläuterung des Verfahrens diene folgende Abbildung:



x_0 ... Anfangswert der Iteration
x_i ... Annäherung an die Nullstelle
x_n ... Nullstelle
t_i ... Tangente in P_i

Die Iterationsvorschrift, die sich in einfacher Weise aus der Gleichung

$$f'(x_i) = \frac{y_{i+1} - y_i}{x_{i+1} - x_i} = \frac{0 - f(x_i)}{x_{i+1} - x_i}$$

herleiten läßt, lautet:

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} .$$

1.2.1.1 Iterative Ermittlung reeller Wurzeln

Die Ermittlung reeller Wurzeln $x = \sqrt[n]{a}$ ($n \epsilon N$ und $a \geq 0$) ist gleichbedeutend mit der Berechnung reeller Nullstellen von Graphen mit der Funktionsgleichung $y = x^n - a$.

Zwecks Annäherung an den gesuchten Wurzelwert dient die Iterationsvorschrift:

$$x_{i+1} = x_i - \frac{x_i^n - a}{n * x_i^{n-1}}$$

bzw.

$$x_{i+1} = \frac{1}{n}\left[(n-1)\,x_i + \frac{a}{x_i^{n-1}}\right] .$$

Betrachtet man x_{i+1} als abhängige Variable $f(x_i)$, dann erhält man für die erste Ableitung der entsprechenden Funktion $f'(x_i) = (1 - \frac{1}{n})(1 - \frac{a}{x_i^n})$ mit $x_i = \sqrt[n]{a}$ als Extremwert.

Aus der 2. Ableitung $\frac{(n-1)a}{x_i^{n+1}}$ ergibt sich durch Einsetzen von $x_i = \sqrt[n]{a}$:

$$f''(\sqrt[n]{a}) = \frac{n-1}{\sqrt[n]{a}} > 0$$

für $n > 1$, d.h. der Extremwert ist ein Minimum. Hieraus resultiert, daß bei einem beliebigen positiven Startwert x_0 das Verfahren konvergiert.

Dazu folgendes Beispiel:

Es soll der Wurzelwert $x = \sqrt[5]{5}$ auf 6 Stellen hinter dem Komma genau ermittelt werden, d.h. die Iteration endet, wenn $|x_{i+1} - x_i| < 10^{-6}$ ist! Die entsprechende Iterationsvorschrift lautet:

$$x_{i+1} = \frac{1}{5}(4x_i + \frac{5}{x_i^4}) = 0,8x_i + \frac{1}{x_i^4}$$

a) Für $x_0 = 1 < x_n$ ergeben sich die Iterationsschritte:

$x_1 = 2,250000$

$x_2 = 1,839018$

$x_3 = 1,558643$

$x_4 = 1,416353$

$x_5 = 1,381575$

$x_6 = 1,379734$

$x_7 = 1,379729$

b) Für $x_0 = a = 5 > x_n$ ergeben sich die Iterationsschritte:

$x_1 = 4,001600$

$x_2 = 3,205180$

$x_3 = 2,573619$

$x_4 = 2,081689$

$x_5 = 1,718603$

$x_6 = 1,489512$

$x_7 = 1,394763$

$x_8 = 1,380050$

$x_9 = 1,379729$

1.2.1.2 Iterative Reziprokwertermittlung

Als weiteres Beispiel für die Newton-Iteration soll die Reziprokwertermittlung nach diesem Verfahren erläutert werden.

Ausgehend von $x = \frac{1}{a}$, $(a > 0)$ ist für die Formulierung der Iterationsvorschrift die Funktion

$$f(x) = \frac{1}{x} - a$$

und deren erste Ableitung

$$f'(x) = -\frac{1}{x^2}$$

erforderlich. Danach lautet die Iterationsvorschrift:

$$x_{i+1} = x_i - \frac{\frac{1-ax_i}{x_i}}{-\frac{1}{x_i^2}} = x_i + x_i - ax_i^2$$

$$x_{i+1} = x_i(2 - ax_i)$$

Dieses Verfahren konvergiert im Intervall $0 < x_0 < \frac{2}{a}$.

Es sei noch erwähnt, daß die Kriterien zur Ermittlung des Konvergenzbereiches von Newton-Iterationen in diesem Zusammenhang nicht behandelt werden. Das genannte Verfahren wird an folgendem Beispiel demonstriert:

Es ist der Reziprokwert von 8, also $x = \frac{1}{8}$ zu ermitteln!

Die Iterationsvorschrift lautet:

$$x_{i+1} = x_i(2 - 8x_i) = 2x_i - 8x_i^2$$

Der Anfangswert $x_0 = 0,1$ liegt im Konvergenzbereich $0 < 0,1 < \frac{2}{8}$.

Iterationsschritte:

$$x_1 = 0,12$$

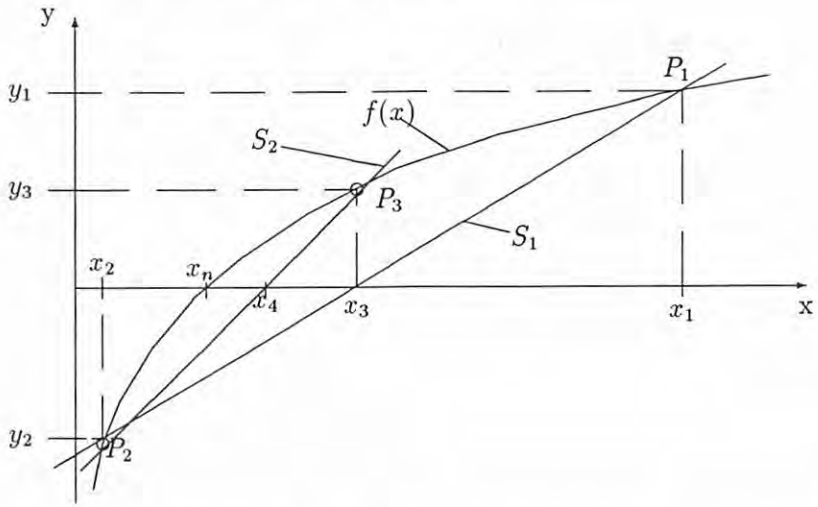
$$x_2 = 0,1248$$

$$x_3 = 0,12499968$$

$$x_4 = 0,125$$

1.2.2 Regula falsi

Die Regula falsi, auch als Sekantennäherungsverfahren bezeichnet, soll anhand folgender Skizze erläutert werden:



Die Regula falsi kommt ohne den Funktionswert der Ableitungsfunktion für den jeweiligen Näherungswert $f'(x_i)$ aus. Das Verfahren beginnt mit der Ermittlung zweier in der Nähe der Nullstelle liegenden Punkte, deren Ordinaten unterschiedliche Vorzeichen besitzen müssen. Der Schnittpunkt der durch P_1 und P_2 verlaufenden Sekante S_1 mit der x-Achse ergibt den Näherungswert x_3. Ausgehend von der Funktionsgleichung der Sekante

$$\frac{y - y_1}{x - x_1} = \frac{y_2 - y_1}{x_2 - x_1}$$

erhält man für $y = 0$ den Wert für x_3:

$$x_3 = x_1 - \frac{f(x_1)}{f(x_2) - f(x_1)}(x_2 - x_1)$$

bzw.

$$x_3 = \frac{x_1 f(x_2) - x_2 f(x_1)}{f(x_2) - f(x_1)} .$$

Für die Fortsetzung des Verfahrens ist es wichtig zu wissen, welches Vorzeichen $f(x_3)$ besitzt.

a) Im Falle $f(x_1) > 0$, $f(x_2) < 0$ und $f(x_3) < 0$ oder $f(x_1) < 0$, $f(x_2) > 0$ und $f(x_3) > 0$ gilt die Iterationsvorschrift:

$$x_{i+3} = \frac{x_1 f(x_{i+2}) - x_{i+2} f(x_1)}{f(x_{i+2}) - f(x_1)}$$

mit $i \epsilon N$

b) Im Falle $f(x_1) > 0$, $f(x_2) < 0$ und $f(x_3) > 0$ oder $f(x_1) < 0$, $f(x_2) > 0$ und $f(x_3) < 0$ gilt die Iterationsvorschrift:

$$x_{i+3} = \frac{x_2 f(x_{i+2}) - x_{i+2} f(x_2)}{f(x_{i+2}) - f(x_2)}$$

mit $i \epsilon N$

Das folgende Beispiel soll dieses Verfahren veranschaulichen.

Aufgabe: Es ist eine Nullstelle der Funktion $f : x \mapsto 2x^3 + 4x^2 - 30x$ zu ermitteln. Für $x_1 = 2$ ist $f(x_1) = -28$ und für $x_2 = 4$ ist $f(x_2) = 72$. Mit diesen Werten ist

$$x_3 = \frac{x_1 f(x_2) - x_2 f(x_1)}{f(x_2) - f(x_1)} = 2,56$$

und $f(x_3) = -17,03$.

Die Fortsetzung des Verfahrens erfolgt nach der unter b) genannten Iterationsvorschrift:

$$x_4 = \frac{x_2 f(x_3) - x_3 f(x_2)}{f(x_3) - f(x_2)} = 2,835$$

$f(x_4) = -7,31$

$$x_5 = \frac{x_2 f(x_4) - x_4 f(x_2)}{f(x_4) - f(x_2)} = 2,943$$

etc.

Der genaue Wert der Nullstelle lautet $x_n = 3$.

1.3 Übungen zur Vertiefung

Aufgabe 1:

Entwickeln Sie für

$$cos\, x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + - \ldots$$

a) das rekursive Bildungsgesetz für die Glieder dieser Reihe,

b) den Algorithmus der rekursiv berechenbaren Funktionswerte,

c) ein aus b) resultierendes Turbo-Pascal-Programm mit dem Dateinamen **cos_funk.pas**.

Aufgabe 2:

Erstellen Sie ein Turbo-Pascal-Programm mit dem Dateinamen **polynom.pas**, das nach Eingabe der Koeffizienten einer ganzrationalen Funktion und eines Intervalls den Graphen der Funktion bei optimaler Ausnutzung des Bildschirmes darstellt. Benutzen Sie zur Berechnung der Funktionswerte das Hornerschema und speichern Sie die Funktionswerte vor der grafischen Darstellung in einem Array.

Beispiel:

Gegeben sei die Funktion mit der Gleichung

$$y = -x^4 + 5\,x^2 - 3\,x + 4$$

und das Intervall $[-2,7; 2,5]$. Darzustellen sei der Graph der Funktion in dem angegebenen Intervall unter optimaler Ausnutzung des Bildschirmes.

Anmerkungen:

- Programmieren Sie übersichtlich, da das Programm später noch in anderen Aufgaben gebraucht wird.
- Mit Hilfe der Turbo-Pascalbefehle **getmaxx** und **getmaxy** lassen sich die Koordinaten des maximal darstellbaren Pixels ermitteln. Das Programm ist so zu schreiben, daß der dargestellte Teil des Funktionsgraphen den Bildschirm ganz ausfüllt.

Aufgabe 3:

Ermitteln Sie die Zahl e mit einer Genauigkeit von 10 Stellen hinter dem Komma mittels der Reihe

$$e^x = \sum_{i=0}^{\infty}[(i!)^{-1}x^i] = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \cdots + \frac{x^n}{n!}$$

für $x = 1$ unter Einbeziehung des Hornerschemas. Nennen Sie das zugehörige Turbo-Pascal-Programm e_zahl_2.pas.

Aufgabe 4:

Ermitteln Sie den Schnittpunkt der Graphen $f_1 : x \mapsto cos\ x$ und $f_2 : x \mapsto tan\ x$ mit Hilfe des Newtonschen Näherungsverfahrens. Dazu muß die Nullstelle der Funktion $f_3 : x \mapsto cos\ x - tan\ x$ bestimmt werden.

Beginnen Sie daher zunächst mit der Herleitung einer geeigneten Iterationsvorschrift und berechnen Sie danach den Abzissenwert im Bogenmaß mit einer Genauigkeit von 10 Stellen hinter dem Komma.

Bei der Entwicklung eines entsprechenden Turbo-Pascal-Programms mit dem Namen new_iter.pas empfiehlt es sich Standardfunktionen zu verwenden.

Überprüfen Sie das Ergebnis durch Lösen der goniometrischen Gleichung $cos\ x - tan\ x = 0$ resp. $sin^2\ x + sin\ x - 1 = 0$.

Aufgabe 5:

Es gilt eine Nullstelle der Funktion $f : x \mapsto x^3 - 3,8x^2 + 7,2$ mit Hilfe der Regula falsi zu ermitteln. Eine der 3 Nullstellen liegt im Intervall [-1;-2], denn es ist $f(-1) = 2,4$ und $f(-2) = -24$.

Das entsprechende Turbo-Pascal- Programm soll den Namen reg_fals.pas erhalten.

2. Von Stützstellen zu Interpolationspolynomen

Sind von einer Funktion mit $y = f(x)$ die Argumente $x = x_0$, $x = x_1$, $x = x_n$ bekannt, dann werden diese in der Mathematik als Stützstellen bezeichnet. Die zugehörigen Funktionswerte $y_0 = f(x_0)$, $y_1 = f(x_1)$, ..., $y_n = f(x_n)$ heißen Stützwerte. Besteht nun die Aufgabe darin, einen Funktionswert $y = f(x)$ für einen beliebigen x-Wert zu berechnen, der zwischen zwei benachbarten Stützstellen liegt, dann muß der gesuchte Funktionswert y aus den bekannten Funktionswerten y_0, y_1, ..., y_n näherungsweise berechnet werden. Dieses nennt man interpolieren.

In der Praxis tritt dieses Problem immer dann auf, wenn man es mit Meßpunkten zu tun hat, die man während eines Versuchs aufgenommen hat. Dabei gilt es eine Funktion zu suchen, die durch die Meßpunkte verläuft. Eine solche Funktion läßt sich am einfachsten durch ein Polynom, genauer gesagt durch ein Interpolationspolynom approximieren.

Aus der Algebra ist bekannt, daß sich für $(n + 1)$ vorgegebene Punkte $(x_0; y_0), ...(x_n; y_n)$ genau ein Polynom n-ten Grades $P_n(x)$ ermitteln läßt. Dieses Polynom dient dann als Näherungsfunktion für $y = f(x)$. Um dieses Polynom zu bestimmen gibt es mehrere Möglichkeiten. Einige grundlegende Verfahren sollen nun in den folgenden Kapiteln erläutert werden. Das Interpolationspolynom stellt eine Annäherung an den tatsächlichen Verlauf der Funktion nur in dem Intervall dar, in dem die Meßpunkte liegen.

2.1 Der Gaußalgorithmus zur Ermittlung von Polynomkoeffizienten

Der Gaußalgorithmus soll zunächst an einem Beispiel demonstriert werden. Im Anschluß daran wird der Algorithmus allgemein erläutert.

Beispiel:

Für die grafische Darstellung einer Funktion seien folgende Stützstellen gegeben:

x	1	2	3
y	2	-1	3

Zu ermitteln ist die Funktionsgleichung dieses Graphen als Interpolationspolynom im Intervall [1;3].

Lösung:

Da es sich in dem Beipiel um 3 Stützstellen handelt, ist die einfachste Lösung ein Polynom 2. Grades. Es sind also die Koeffizienten des Polynomes $y = a_2\,x^2 + a_1\,x + a_0$ zu ermitteln.

Nun werden die Stützstellenpaare der Reihe nach in die Parabelgleichung eingesetzt.

Der erste Punkt lautet $(x/y) = (1/2)$: Nach Einsetzen in die Gleichung erhält man:

$$2 = a_2\,1^2 + a_1\,1 + a_0$$

oder anders geschrieben:

$$a_2 + a_1 + a_0 = 2\,.$$

Nach dieser Methode setzt man alle Stützstellenpaare in die Funktionsgleichung ein und erhält daraufhin folgendes Gleichungssystem:

$$\begin{aligned} I)\quad & a_2 + a_1 + a_0 = 2 \\ II)\quad & 4a_2 + 2a_1 + a_0 = -1 \\ III)\quad & 9a_2 + 3a_1 + a_0 = 3 \end{aligned}$$

Auf diese Weise ist ein Gleichungssystem mit 3 Gleichungen für 3 Variable entstanden. Dieses kann mit Hilfe des Additionsverfahrens in ein gestaffeltes Gleichungssystem umgeformt werden.

Dazu multipliziert man z.B. die erste Gleichung mit $\frac{4}{1}$ und addiert diese zur zweiten Gleichung hinzu. Die erste Gleichung bleibt bestehen, die dritte hingegen vorerst unverändert:

$$\begin{aligned} I)\quad & a_2 + a_1 + a_0 = 2 \quad |\tfrac{-4}{1} \\ II)\quad & 4a_2 + 2a_1 + a_0 = -1 \\ III)\quad & 9a_2 + 3a_1 + a_0 = 3 \end{aligned}$$

Daraus resultiert

$$\begin{array}{lrrrrl} I) & a_2 + & a_1 & + & a_0 & = 2 \quad \left|\frac{-9}{1}\right. \\ II) & & +(-2a_1) & + & (-3a_0) & = -9 \\ III) & 9a_2 + & 3a_1 & + & a_0 & = 3 \end{array}$$

Nun multipliziert man die erste Gleichung mit $\frac{-9}{1}$ und addiert diese zur dritten Gleichung hinzu. Die erste und zweite Gleichung bleibt unverändert:

$$\begin{array}{lrrrrl} I) & a_2 + & a_1 & + & a_0 & = 2 \\ II) & & +(-2a_1) & + & (-3a_0) & = -9 \quad \left|-\frac{-6}{-2}\right. \\ III) & & +(-6a_1) & + & (-8a_0) & = -15 \end{array}$$

Schließlich multipliziert man die zweite Gleichung mit $-\frac{-6}{2}$ und addiert diese zur dritten Gleichung hinzu. Die Gleichungen (I) und (II) bleiben unverändert:

$$\begin{array}{lrrrrl} I) & a_2 + & a_1 & + & a_0 & = 2 \\ II) & & +(-2a_1) & + & (-3a_0) & = -9 \\ III) & & & & +(-8a_0) & = 12 \end{array}$$

Damit ist das gestaffelte Gleichungssystem erstellt. Als nächstes wären die Koeffizienten a_2 bis a_0 durch Einsetzen in die Gleichungen zu ermitteln.

Gleichung (III) ergibt die Lösung für a_0 an. Aus den übrigen beiden Gleichungen ergeben sich für $a_1 = -13,5$ und für $a_2 = 3,5$.

Setzt man nun die Lösungen des Gleichungssystems a_2 bis a_0 in die ursprüngliche Polynomgleichung $y = a_2 x^2 + a_1 x + a_0$ ein, so erhält man das gesuchte Interpolationspolynom:

$$y = 3,5\,x^2 - 13,5\,x + 12\,.$$

Damit ist der Gaußalgorithmus beispielhaft erläutert worden. Für die allgemeine Lösung gilt:

Ersetzt man in dem Interpolationspolynom

$$y = a_{n-1}x^{n-1} + \ldots + a_2\,x^2 + a_1\,x + a_0$$

die x- und y-Werte durch Wertepaare (x_i/y_i) der Stützstellen, so erhält man für die Berechnung der Koeffizienten a_i ein lineares Gleichungssystem mit n Variablen, wobei n der Anzahl der Stützstellen entspricht.

Gegeben seien allgemein die Stützstellen:

x	x_1	x_2	x_3	...	x_n
y	y_1	y_2	y_3	...	y_n

Als Grundlage gilt zunächst folgendes Gleichungssystem:

$$x_1^{n-1}\, a_{n-1} + \ldots + x_1^2\, a_2 + x_1\, a_1 + a_0 = y_1$$

$$x_2^{n-1}\, a_{n-1} + \ldots + x_2^2\, a_2 + x_2\, a_1 + a_0 = y_2$$

$$x_2^{n-1}\, a_{n-1} + \ldots + x_3^2\, a_2 + x_3\, a_1 + a_0 = y_3$$

$$\vdots$$

$$x_m^{n-1}\, a_{n-1} + \ldots + x_m^2\, a_2 + x_m\, a_1 + a_0 = y_m$$

Nach Einsetzen der Wertepaare (x_1/y_1) bis (x_n/y_n) erhält man folgendes lineares Gleichungssystem:

$$k_{1n-1}\, a_{n-1} + \ldots + k_{12}\, a_2 + k_{11}\, a_1 + k_{10} a_0 = y_1$$

$$k_{2n-1}\, a_{n-1} + \ldots + k_{22}\, a_2 + k_{21}\, a_1 + k_{20} a_0 = y_2$$

$$k_{3n-1}\, a_{n-1} + \ldots + k_{32}\, a_2 + k_{31}\, a_1 + k_{30} a_0 = y_3$$

$$\vdots$$

$$k_{mn-1}\, a_{n-1} + \ldots + k_{m2}\, a_2 + k_{m1}\, a_1 + k_{m0} a_0 = y_m$$

Dieses Gleichungssystem läßt sich mit Hilfe des Gaußalgorithmus lösen:

Analog zu dem am Anfang des Kapitels aufgeführten Beispiels multipliziert man die erste Gleichung mit $-\frac{k_{2n-1}}{k_{1n-1}}$ und addiert sie zur zweiten Gleichung. Das Ergebnis der Addition wird in die zweite Gleichung übernommen. Die erste Gleichung bleibt immer unverändert. Somit erhält man am Anfang der zweiten Gleichung eine Null:

$$k_{1n-1}\, a_{n-1} + \ldots + k_{12}\, a_2 + k_{11}\, a_1 + k_{10} a_0 = y_1 \quad \left| * \frac{-k_{2n-1}}{k_{1n-1}} \right.$$

$$k_{2n-1}\, a_{n-1} + \ldots + k_{22}\, a_2 + k_{21}\, a_1 + k_{20} a_0 = y_2$$

$$k_{3n-1}\, a_{n-1} + \ldots + k_{32}\, a_2 + k_{31}\, a_1 + k_{30} a_0 = y_3$$

$$\vdots$$

$$k_{mn-1}\, a_{n-1} + \ldots + k_{m2}\, a_2 + k_{m1}\, a_1 + k_{m0} a_0 = y_m$$

Nach der Addition erhält man:

$$
\begin{aligned}
k_{1n-1}\,a_{n-1} \quad &+\ldots+ \quad k_{12}\,a_2 + k_{11}\,a_1 + k_{10}a_0 = y_1 \\
k_{2n-1}\,a_{n-1} \quad &+\ldots+ \quad k_{22}\,a_2 + k_{21}\,a_1 + k_{20}a_0 = y_2 \qquad *) \\
k_{3n-1}\,a_{n-1} \quad &+\ldots+ \quad k_{32}\,a_2 + k_{31}\,a_1 + k_{30}a_0 = y_3 \\
&\vdots \\
k_{mn-1}\,a_{n-1} \quad &+\ldots+ \quad k_{m2}\,a_2 + k_{m1}\,a_1 + k_{m0}a_0 = y_m
\end{aligned}
$$

*) Anmerkung: Durch das Multiplizieren und Addieren haben sich die Werte von k_{2n-1}, ... , k_{22}, k_{21}, k_{20}, und y_2 verändert. Man beläßt jedoch den Variablen denselben Namen, da der eigentliche Variablenwert für die Betrachtung unerheblich ist. Dies gilt analog auch für die folgenden Schritte bei der allgemeinen Darstellung des Algorithmus.

Als nächstes multipliziert man die erste Gleichung mit $\frac{-k_{3n-1}}{k_{1n-1}}$ und addiert sie zur dritten Gleichung. Das Ergebnis der Addition wird in der dritten Gleichung übernommen. Die erste Gleichung bleibt wie immer unverändert. Somit erhält man auch am Anfang der dritten Gleichung eine Null:

$$
\begin{aligned}
k_{1n-1}\,a_{n-1} \quad &+\ldots+ \quad k_{12}\,a_2 + k_{11}\,a_1 + k_{10}a_0 = y_1 \qquad \left| * \tfrac{-k_{3n-1}}{k_{1n-1}} \right. \\
0\,a_{n-1} \quad &+\ldots+ \quad k_{22}\,a_2 + k_{21}\,a_1 + k_{20}a_0 = y_2 \\
k_{3n-1}\,a_{n-1} \quad &+\ldots+ \quad k_{32}\,a_2 + k_{31}\,a_1 + k_{30}a_0 = y_3 \\
&\vdots \\
k_{mn-1}\,a_{n-1} \quad &+\ldots+ \quad k_{m2}\,a_2 + k_{m1}\,a_1 + k_{m0}a_0 = y_m
\end{aligned}
$$

Nach der Addition erhält man:

$$
\begin{aligned}
k_{1n-1}\,a_{n-1} \quad &+\ldots+ \quad k_{12}\,a_2 + k_{11}\,a_1 + k_{10}a_0 = y_1 \\
0\,a_{n-1} \quad &+\ldots+ \quad k_{22}\,a_2 + k_{21}\,a_1 + k_{20}a_0 = y_2 \\
0\,a_{n-1} \quad &+\ldots+ \quad k_{32}\,a_2 + k_{31}\,a_1 + k_{30}a_0 = y_3 \\
&\vdots \\
k_{mn-1}\,a_{n-1} \quad &+\ldots+ \quad k_{m2}\,a_2 + k_{m1}\,a_1 + k_{m0}a_0 = y_m
\end{aligned}
$$

Nach diesem Prinzip wird so lange verfahren, bis in der ersten Spalte (außer in der ersten Gleichung) nur noch Nullen stehen. Daraus resultiert folgende Darstellung:

$$\begin{aligned}
k_{1n-1}\,a_{n-1} &+\ldots+ k_{12}\,a_2 + k_{11}\,a_1 + k_{10}a_0 = y_1\\
0\,a_{n-1} &+\ldots+ k_{22}\,a_2 + k_{21}\,a_1 + k_{20}a_0 = y_2\\
0\,a_{n-1} &+\ldots+ k_{32}\,a_2 + k_{31}\,a_1 + k_{30}a_0 = y_3\\
&\vdots\\
0\,a_{n-1} &+\ldots+ k_{m2}\,a_2 + k_{m1}\,a_1 + k_{m0}a_0 = y_m
\end{aligned}$$

Weil in diesem Gleichungssystem jede Gleichung als Reihe dargestellt wird, kann man auch schreiben:

$$\begin{aligned}
k_{1n-1}\ a_{n-1} + k_{1n-2}\ a_{n-2} +\ldots+ k_{12}\ a_2 + k_{11}\,a_1 + k_{10}\,a_0 &= y_1\\
0\ \ a_{n-1} + k_{2n-2}\ a_{n-2} +\ldots+ k_{22}\ a_2 + k_{21}\,a_1 + k_{20}\,a_0 &= y_2\\
0\ \ a_{n-1} + k_{3n-2}\ a_{n-2} +\ldots+ k_{32}\ a_2 + k_{31}\,a_1 + k_{30}\,a_0 &= y_3\\
&\vdots\\
0\ \ a_{n-1} + k_{mn-2}\ a_{n-2} +\ldots+ k_{m2}\ a_2 + k_{m1}\,a_1 + k_{m0}\,a_0 &= y_m
\end{aligned}$$

Da in der ersten Spalte (bis auf die erste Gleichung) überall Nullen erzeugt wurden, fällt der erste Teil jeweils heraus. Man erhält nun folgendes Schema:

$$\begin{aligned}
k_{1n-1}\ a_{n-1}\ \ k_{1n-2}\ a_{n-2} +\ldots+ k_{12}\ a_2 + k_{11}\,a_1 + k_{10}\,a_0 &= y_1\\
k_{2n-2}\ a_{n-2} +\ldots+ k_{22}\ a_2 + k_{21}\,a_1 + k_{20}\,a_0 &= y_2\\
k_{3n-2}\ a_{n-2} +\ldots+ k_{32}\ a_2 + k_{31}\,a_1 + k_{30}\,a_0 &= y_3\\
&\vdots\\
k_{mn-2}\ a_{n-2} +\ldots+ k_{m2}\ a_2 + k_{m1}\,a_1 + k_{m0}\,a_0 &= y_m
\end{aligned}$$

Vergleicht man diese Darstellung mit der 1. Darstellung des linearen Gleichungssystems, so stellt man fest, daß sich jene bis auf die erste Gleichung kaum von dieser unterscheidet. Lediglich die Variablenbezeichnungen sind anders. Das heißt: Man kann das neu entstandene System auf die gleiche Art und Weise lösen wie das ursprüngliche, wenn man von der ersten Gleichung absieht. Man multipliziert die zweite Gleichung mit $\frac{-k_{3n-2}}{k_{1n-2}}$ und addiert sie dann zur dritten Gleichung. Das Ergebnis der Addition wird in der dritten Gleichung übernommen. Die zweite Gleichung bleibt hierbei immer unverändert. Somit erhält man am Anfang der dritten Gleichung eine Null:

$$
\begin{array}{rrrrrrrrrrrrrrr}
k_{1n-1} & a_{n-1} & k_{1n-2} & a_{n-2} & + \ldots + & k_{12} & a_2 & + & k_{11} a_1 & + & k_{10} a_0 & = & y_1 \\
 & & k_{2n-2} & a_{n-2} & + \ldots + & k_{22} & a_2 & + & k_{21} a_1 & + & k_{20} a_0 & = & y_2 \\
 & & 0 & a_{n-2} & + \ldots + & k_{32} & a_2 & + & k_{31} a_1 & + & k_{30} a_0 & = & y_3 \\
 & & \vdots & & & & & & & & & & \\
 & & k_{mn-2} & a_{n-2} & + \ldots + & k_{m2} & a_2 & + & k_{m1} a_1 & + & k_{m0} a_0 & = & y_m
\end{array}
$$

Man setzt das Verfahren fort, bis man an das Ende der zweiten Spalte gelangt und überall (bis auf die zweite Gleichung) eine Null erzeugt hat. Man erhält dann folgende Darstellung:

$$
\begin{array}{rrrrrrrrrrrrrrr}
k_{1n-1} & a_{n-1} & k_{1n-2} & a_{n-2} & + \ldots + & k_{12} & a_2 & + & k_{11} a_1 & + & k_{10} a_0 & = & y_1 \\
 & & k_{2n-2} & a_{n-2} & + \ldots + & k_{22} & a_2 & + & k_{21} a_1 & + & k_{20} a_0 & = & y_2 \\
 & & 0 & a_{n-2} & + \ldots + & k_{32} & a_2 & + & k_{31} a_1 & + & k_{30} a_0 & = & y_3 \\
 & & \vdots & & & & & & & & & & \\
 & & 0 & a_{n-2} & + \ldots + & k_{m2} & a_2 & + & k_{m1} a_1 & + & k_{m0} a_0 & = & y_m
\end{array}
$$

Wegen der Reihenschreibweise läßt sich das System erweitern, um letztlich nach dem bekannten Verfahren wieder Nullen zu erzeugen. Dies könnte man nun bei einer allgemeingültigen Darstellung bis ins Unendliche fortführen. Wir brechen daher an dieser Stelle ab. Da jedoch immer nur endliche Gleichungssysteme gelöst werden, erhält man schließlich folgende Darstellung:

$$
\begin{array}{rrrrrrrrrrrrrrr}
k_{1n-1} & a_{n-1} + & k_{1n-2} & a_{n-2} & + \ldots + & k_{12} & a_2 + & & k_{11} a_1 + & & k_{10} a_0 & = & y_1 \\
 & & k_{2n-2} & a_{n-2} & + \ldots + & k_{22} & a_2 + & & k_{21} a_1 + & & k_{20} a_0 & = & y_2 \\
 & & & & \ldots + & k_{32} & a_2 + & & k_{31} a_1 + & & k_{30} a_0 & = & y_3 \\
 & & & & & & & & \vdots & & & & \\
 & & & & & & & & k_{(m-1)1} a_1 + & & k_{(m-1)0} a_0 & = & y_{(m-1)} \\
 & & & & & & & & & & k_{m0} a_0 & = & y_m
\end{array}
$$

Diese Darstellung wird als gestaffeltes Gleichungssystem bezeichnet. Daraus lassen sich die Koeffizienten a_0 bis a_{n-1} des gesuchten Interpolationspolynoms berechnen. Zur Veranschaulichung der hierbei angewandten Methode sei auf das vorangegangene Beispiel verwiesen.

Wichtiger Hinweis:

Wie bereits erwähnt, werden die Variablenwerte schrittweise verändert. Man darf deshalb keinesfalls aus der Herleitung zum gestaffelten Gleichungssystem folgern, daß sich z.B. der Koeffizient a_0 so berechnen läßt, indem man den Quotienten von $\frac{y_m}{k_{m0}}$ bildet und dabei für y_m die letzte Stützstelle, sowie für k_{m0} den Wert aus dem zu Anfang dargestellten

linearen Gleichungssystem einsetzt! Man kann sich die Berechnung des gestaffelten Gleichungssystems nicht ersparen!

Da die Koeffizientendeterminante eine Vandermondesche Determinante

$$D_{Van} = \begin{vmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^{n-1} \\ 1 & x_2 & x_2^2 & \cdots & x_2^{n-1} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 1 & x_n & x_n^2 & \cdots & x_n^{n-1} \end{vmatrix} = \prod(x_i - x_k)$$

mit $i, k, n \epsilon N$ für alle $i > k$ von 1 bis n ist, hat dieses Gleichungssystem immer eine eindeutige Lösung, wenn $D_{Van} \neq 0$ ist. Um zu gewährleisten, daß $D_{Van} \neq 0$ ist, muß überprüft werden, ob jede Stützstelle nur einmal eingegeben und die Stützstelle mit dem Abszissenwert Null ans Ende gesetzt wurde.

2.1.1 Erstellen einer Matrix nach Eingabe von Stützstellen

Die Aufgabenstellung in Kapitel 2.1 beinhaltet eine Grobgliederung zur Lösung der gesamten Aufgabe. In diesem Kapitel wird der Aufgabenteil a) erläutert, also das Erstellen einer geeigneten Matrix mit der später die Koeffizienten des Interpolationspolynoms berechnet werden können.

Um diese Aufgabe lösen zu können, sei nochmals auf das Kapitel 2.1 verwiesen. Darin wird erklärt, daß

- von der allgemeinen Form der Polynomgleichung auszugehen ist.
- der Grad des Polynoms (Interpolationspolynoms) von der Anzahl der Stützstellen abhängt.
- der Grad des Interpolationspolynoms immer um eins kleiner ist als die Anzahl der Stützstellen.
- das gesuchte Gleichungssystem immer aus so vielen Gleichungen besteht wie Stützstellen vorhanden sind.
- die einzelnen Gleichungen des Gleichungssystems nach der allgemeinen Form der Polynomgleichung erzeugt werden, ohne linear abhängig zu sein. Wenn man davon ausgeht, daß keine Stützstelle doppelt eingegeben wird, ist diese Auflage erfüllt.

- das Interpolationspolynom immer der Anzahl der Stützstellen entsprechend viele unbekannte Koeffizienten besitzt.

Damit sind die wichtigsten Bedingungen genannt worden, die bei der Entwicklung des Algorithmus berücksichtigt werden müssen.

Zunächst benötigt man ein zweidimensionales ARRAY. Dieses stellt das Gerüst für die Matrix dar. Es ist wegen der y-Werte $n+1$ Spalten breit und entsprechend der Anzahl der Stützstellen n Zeilen lang. Die Anzahl der unbekannten Koeffizienten entspricht der Anzahl der Stützstellen. Damit ist die Größe des ARRAY limitiert worden.

Da nach Einsetzen der Stützstellen in die Polynomgleichung dem Koeffizient a_0 kein Wert zugewiesen wird, bleibt diese Spalte zunächst frei. In der Spalte a_1 stehen die x-Werte der Stützstellen. Zum besseren Verständnis wird das ARRAY $a[n+1,n]$ im Folgenden wiedergegeben:

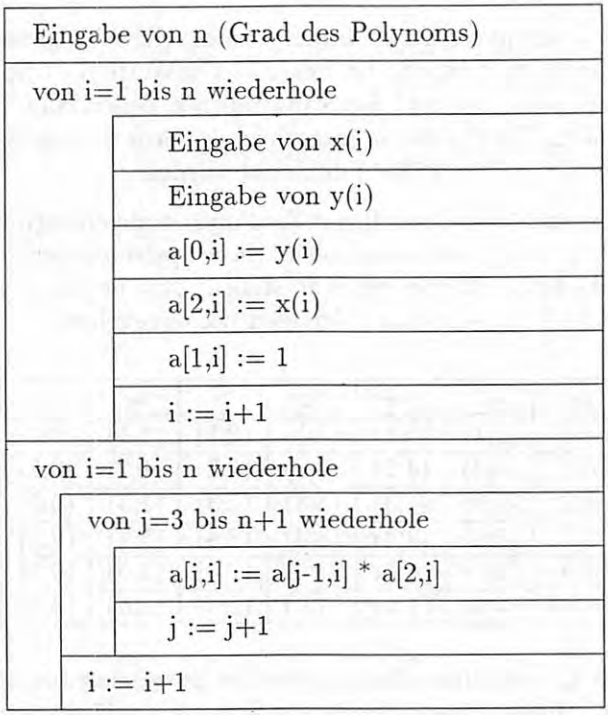
a_n	a_4	a_3	a_2	a_1	a_0	y
(n;1)	(5;1)	(4;1)	(3;1)	(2;1)	(1;1)	(0;1)
(n;2)	(5;2)	(4;2)	(3;2)	(2;2)	(1;2)	(0;2)
(n;3)	(5;3)	(4;3)	(3;3)	(2;3)	(1;3)	(0;3)
(n;4)	(5;4)	(4;4)	(3;4)	(2;4)	(1;4)	(0;4)
(n;5)	(5;5)	(4;5)	(3;5)	(2;5)	(1;5)	(0;5)
(n;n)	(5;n)	(4;n)	(3;n)	(2;n)	(1;n)	(0;n)

Damit läßt sich deutlicher erkennen wie die Zuweisung bei der Eingabe erfolgen muß. Mittels einer Schleife, die von 1 bis n läuft, werden die y-Werte innerhalb des ARRAYs in [0;n]; und die x-Werte in [2;n] eingelesen. Die Konstante n muß zuvor eingegeben werden. In der Spalte a_0 erhält jedes Feld den Wert 1, weil $x^0 = 1$ ist. Es gilt also die Zuweisung innerhalb der Schleife von 1 bis n für die Spalte $a_0[1;n] = 1$.

Diese Teilschritte lassen sich in einer einzigen Schleife realisieren.

Um die Potenzen des Gleichungssystems zu erhalten, multipliziert man die Werte in der Spalte a_1 mit sich selbst. Ein Wert in der Spalte a_3 wird so erzeugt, indem man den zugehörigen Wert in der Spalte a_2 mit a_1 multipliziert. Dieses Verfahren ist bis n zu wiederholen. Danach ist eine neue Schleife unumgänglich. Diese Schleife durchläuft die Zeilen 3 bis n. Um für alle Spalten die zugehörigen Werte berechnen zu können, benötigt man eine weitere, äußere Schleife, mit (bei 1 beginnend) n Durchläufen.

Damit ist die Reihenfolge festgelegt in der die Matrix erstellt wird. Außerdem sind bereits die Anzahl der benötigten Schleifen und die erforderlichen Abbruchkriterien festgelegt worden. Zur Veranschaulichung des Algorithmus dient folgendes Struktogramm:



n	:	Grad der Funktion vom Datentyp Byte
i,j	:	Schleifenvariable vom Datentyp Byte
a	:	Koeffizientenmatrix vom Datentyp ARRAY[0..n],[1..n]

Nachdem das Struktogramm erstellt ist, kann man mit dem Programmieren des Quelltextes beginnen:

```
PROGRAM Gauss_1; (* Werteeingabe und Erstellen der Matrix *)
uses crt;

CONST   nmax = 10; (* Maximale ARRAY- Größe / veränderbar *)

VAR   i,j,n : Byte;
        a : ARRAY[0..nmax],[1..nmax] OF REAL;

BEGIN
  CLRSCR;
  WRITE('Wie viele Stützstellen sollen eingegeben werden ?');
  READLN(n);

  FOR i := 1 TO n DO
    BEGIN
      WRITE(' x',i,' = ');
        READLN(a[2,i]);
      WRITE(' y',i,' = ');
        READLN(a[0,i]);
      a[1,i] := 1;
    END;
              (* Eingabe der Stützstellen beendet *)
              (* Zuweisung von 1 in Spalte a0 ist erfolgt *)

  FOR i := 1 TO n DO
    FOR j := 3 TO n DO
      a[j,i] := a[j-1,i] * a[2,i];
              (* Matrix fertig *)

  CLRSCR;
    FOR i := 1 TO n DO
      FOR j := n DOWNTO 1 DO
        BEGIN
          WRITE(a[j,i]:6:2);
          WRITE(' || ');
          WRITELN(a[0,i]:6:2);
        END;
             (* Ausgabe der gesamten Matrix auf den Bildschirm *)
END.
```

Der Quelltext wäre damit erstellt und muß lediglich auf seine semantische Korrektheit hin überprüft werden. Dazu könnte das Beispiel aus Kapitel 2.1 und ein weiteres mit mindestens 3 Stützstellen dienen.

2.1.2 Algorithmus zur Erstellung des gestaffelten Gleichungssystems

Nachdem im Kapitel 2.1.1 das Gleichungssystem erstellt wurde, wäre nun der in Kapitel 2.1 beschriebene Aufgabenteil b) zu lösen. Es gilt die entstandene Matrix aus Aufgabenteil a) in eine Dreiecksmatrix eines gestaffelten Gleichungssystems umzuwandeln. Das soll mit Hilfe des Gaußalgorithmus erfolgen. Betrachtet man die Herleitung aus Kapitel 2.1 eingehend, so stellt man fest, daß sich das gesamte Problem mit drei geschickt verknüpften Schleifen lösen läßt:

- Schleife 1 ist für das zeilenweise Multiplizieren und Aufaddieren mit dem negativen Kehrwert erstellt worden.
- Schleife 2 bedingt in der jeweiligen Spalte, daß von der obersten bis zur untersten Gleichung Nullen erzeugt werden. Schleife 2 beeinflußt Schleife 1.
- Schleife 3 verweist Schleife 2 in die Spalte, in der noch nicht ausreichend Nullen erzeugt worden sind.

Nachdem alle Schleifen erfolgreich durchlaufen wurden, ist das gestaffelte Gleichungssystem erzeugt worden. Um das Zusammenwirken der einzelnen Schleifen darstellen zu können, geht man zweckmäßigerweise von einem ARRAY aus.

Schleife 1:
Schleife 1 beginnt bei dem Feld [0;2] (Anfangswert bestimmt Schleife 3) und endet beim Feld [n;2]. [1] Ist die Schleife durchlaufen, findet derselbe Vorgang eine Zeile tiefer statt usw. Diese Steuerung übernimmt Schleife 2. Schleife 1 läuft daher von 0 bis n. Den erfaßten Bereich der Schleife zeigt die folgende Darstellung:

[1] Dies mag zunächt verwundern, weil sich nach dem mathematischen Algorithmus der Wert der Schleifenvariablen von n bis 0 veringern müßte. In dem Falle würde man allerdings im ARRAY vorzeitig Werte überschreiben, was letztlich zu falschen Ergebnissen führt.

a_n	a_4	a_3	a_2	a_1	a_0	y
(n;1)	(5;1)	(4;1)	(3;1)	(2;1)	(1;1)	(0;1)
(n;2)	**(5;2)**	**(4;2)**	**(3;2)**	**(2;2)**	**(1;2)**	**(0;2)**
(n;3)	(5;3)	(4;3)	(3;3)	(2;3)	(1;3)	(0;3)
(n;4)	(5;4)	(4;4)	(3;4)	(2;4)	(1;4)	(0;4)
(n;5)	(5;5)	(4;5)	(3;5)	(2;5)	(1;5)	(0;5)
(n;n)	(5;n)	(4;n)	(3;n)	(2;n)	(1;n)	(0;n)

Schleife 2: Schleife 2 beginnt bei dem Feld [n;2] (Anfangswert bestimmt Schleife 3) und endet beim Feld [n;n]. Ist die Schleife durchlaufen, findet derselbe Vorgang eine Spalte weiter rechts statt usw. Diese Steuerung übernimmt Schleife 3. Schleife 2 läuft daher von 2 bis n. Den erfaßten Bereich der Schleife zeigt die folgende Darstellung:

a_n	a_4	a_3	a_2	a_1	a_0	y
(n;1)	(5;1)	(4;1)	(3;1)	(2;1)	(1;1)	(0;1)
(n;2)	(5;2)	(4;2)	(3;2)	(2;2)	(1;2)	(0;2)
(n;3)	(5;3)	(4;3)	(3;3)	(2;3)	(1;3)	(0;3)
(n;4)	(5;4)	(4;4)	(3;4)	(2;4)	(1;4)	(0;4)
(n;5)	(5;5)	(4;5)	(3;5)	(2;5)	(1;5)	(0;5)
(n;n)	(5;n)	(4;n)	(3;n)	(2;n)	(1;n)	(0;n)

Schleife 3: Schleife 3 beginnt bei dem Feld [n;1] Sie bestimmt die Anfangswerte für die Schleifen 1 und 2. Schleife 3 endet beim Feld [2;n-1]. Ist die Schleife durchlaufen, dann ist das gestaffelte Gleichungssystem erstellt. Schleife 3 läuft daher von 1 bis n-1. Den erfaßten Bereich der Schleife zeigt die folgende Darstellung:

a_n	a_4	a_3	a_2	a_1	a_0	y
(n;1)	(5;1)	(4;1)	(3;1)	(2;1)	(1;1)	(0;1)
(n;2)	**(5;2)**	(4;2)	(3;2)	(2;2)	(1;2)	(0;2)
(n;3)	(5;3)	**(4;3)**	(3;3)	(2;3)	(1;3)	(0;3)
(n;4)	(5;4)	(4;4)	**(3;4)**	(2;4)	(1;4)	(0;4)
(n;5)	(5;5)	(4;5)	(3;5)	**(2;5)**	(1;5)	(0;5)
(n;n)	(5;n)	(4;n)	(3;n)	(2;n)	(1;n)	(0;n)

Nachdem nun die Schleifen und die benötigen Bedingungen wie Anfangswert und Abbruchkriterium festgelegt sind, sollen diese Überlegungen an einem Beispiel nachvollzogen werden. Dazu möge das Beispiel aus Kapitel 2.1 dienen.

a_2	a_1	a_0	y	$\Rightarrow$	a_2	a_1	a_0	y
(3;1)	(2;1)	(1;1)	(0;1)	$\Rightarrow$	1	1	1	2
(3;2)	(2;2)	(1;2)	(0;2)	$\Rightarrow$	0	-2	-3	-9
(3;3)	(2;3)	(1;3)	(0;3)	$\Rightarrow$	0	-6	-8	-15

Nach dem Gaußalgorithmus ergibt sich: Berechnung der zweiten Zeile:

$$[0;2] = [0;2] - \frac{[0;1][3;2]}{[3;1]} \Rightarrow -1 - \frac{2*4}{1} = -9$$

usw. Insgesamt ergibt sich:

a_2	a_1	a_0	y	$\Rightarrow$	a_2	a_1	a_0	y
(3;1)	(2;1)	(1;1)	(0;1)	$\Rightarrow$	1	1	1	2
(3;2)	(2;2)	(1;2)	(0;2)	$\Rightarrow$	0	-2	-3	-9
(3;3)	(2;3)	(1;3)	(0;3)	$\Rightarrow$	0	-6	-8	-15

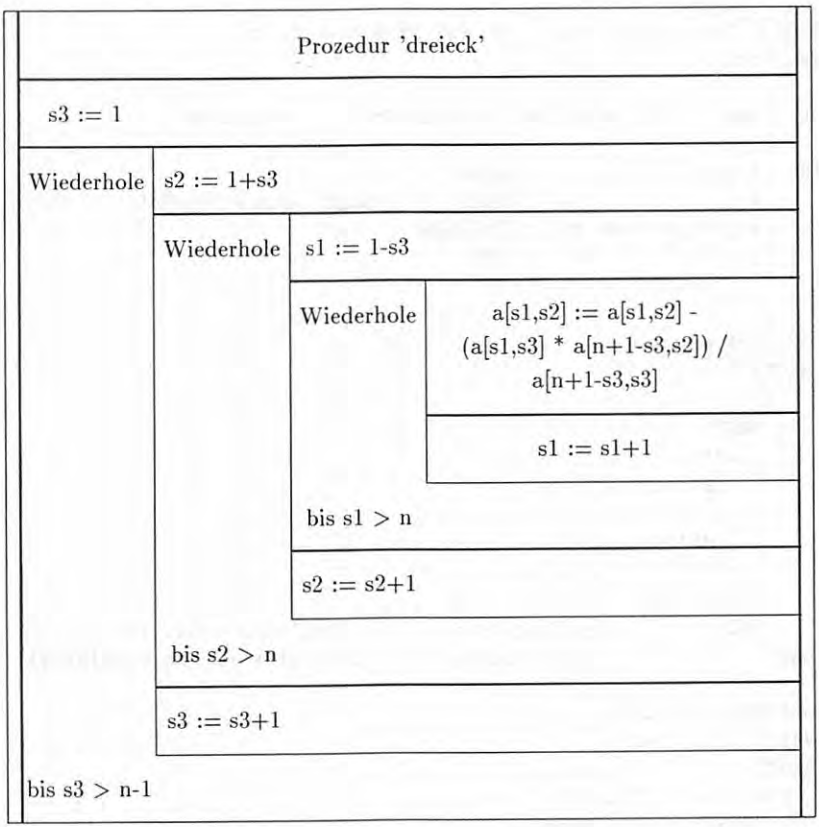
Zweite Berechnung der dritten Zeile:

$$[0;2] = [0;2] - \frac{[0;2][2;3]}{[2;2]} \Rightarrow -15 - \frac{(-9)(-6)}{-2} = 12$$

usw. Insgesamt ergibt sich:

a_2	a_1	a_0	y	$\Rightarrow$	a_2	a_1	a_0	y
(3;1)	(2;1)	(1;1)	(0;1)	$\Rightarrow$	1	1	1	2
(3;2)	(2;2)	(1;2)	(0;2)	$\Rightarrow$	0	-2	-3	-9
(3;3)	(2;3)	(1;3)	(0;3)	$\Rightarrow$	0	0	1	12

Vergleicht man die Ergebnisse mit dem Beispiel aus Kapitel 2.1, dann stellt man fest, daß alles richtig berechnet wurde. Damit ist die Grundlage für das Erstellen des Struktogramms geschaffen worden:



Verwendete Variablen:

s1,s2,s3	:	Schleifenvariablen vom Datentyp Integer
a	:	Bereits in Aufgabenteil (a) deklariert
n	:	Bereits in Aufgabenteil (a) deklariert

Nachdem das Struktogramm erstellt wurde, wird es in den Turbo-Pascal Quelltext umgesetzt:

```
PROGRAM Gaussalgorithmus; (* auf Diskette gauss_2.pas *)
uses crt;

CONST nmax = 50; (* Maximale ARRAY-Größe / veränderbar *)

VAR   i,j,n              : Byte;
      a                  : ARRAY[0..nmax,1..nmax] OF REAL;
      s1,s2,s3,srek,sp   : INTEGER;
      h                  : REAL;


PROCEDURE eingabe;
  BEGIN
    FOR i := 1 TO n DO
      BEGIN
        WRITE(' x',i,' = ');
          READLN(a[2,i]);
        WRITE(' y',i,' = ');
          READLN(a[0,i]);
        a[1,i] := 1;
        WRITELN;
      END;          (* Eingabe der Stützstellen beendet *)
  END;              (* Zuweisung von 1 in Spalte a0 ist erfolgt *)

PROCEDURE nullsuche;
  VAR hilf : REAL;
  BEGIN
    FOR i := 1 TO n DO
      IF a[2,i] = 0 THEN
        BEGIN
          hilf := a[2,n];
          a[2,n] := a[2,i];
          a[2,i] := hilf;
          hilf := a[0,n];
          a[0,n] := a[0,i];
          a[0,i] := hilf;
        END;
  END;
```

```
  PROCEDURE matrix;
  BEGIN
    FOR i := 1 TO n DO
      FOR j := 3 TO n DO
        a[j,i] := a[j-1,i] * a[2,i];
  END;          (* Matrix fertig *)


  PROCEDURE dreieck;
  BEGIN
    s3 := 1;
      REPEAT
        s2 := s3+1;
          REPEAT
            s1 := 1-s3;
            REPEAT
              a[s1,s2] := a[s1,s2] - ( (a[s1,s3] * a[n+1-s3,s2])
              / a[n+1-s3,s3] );
              s1 := s1+1;
            UNTIL s1 > n;
            s2 := s2+1;
          UNTIL s2 > n;
          s3 := s3+1;
      UNTIL s3 > n-1;
  END; (* gestaffeltes Gleichungssystem ist erstellt *)

PROCEDURE ausgabe;
  BEGIN
    CLRSCR;
      FOR i := 1 TO n DO
        BEGIN
          FOR j := n DOWNTO 1 DO
            BEGIN
              WRITE(a[j,i]:6:2);
              WRITE(' ');
            END;
          WRITELN(a[0,i]:6:2);
        END;
      READLN; (* Ausgabe der gesamten Matrix auf den Bildschirm *)
  END;
```

```
BEGIN
  CLRSCR;
  WRITELN(' Algorithmus für das gestaffelte Gleichungssystem ');
  WRITELN;
  WRITE(' Wie viele Stützstellen sollen eingegeben werden ? ');
    READLN(n);
  WRITELN; WRITELN;
  eingabe;      (* liest die Stützstellenwerte ein *)
  nullsuche;    (* schreibt die Stützstelle x=0 an das Ende, da *)
                (* der Algorithmus sonst versagt *)
  matrix;       (* erstellt aus den Stützstellen ein *)
                (* Gleichungssystem in Matrixform *)
  dreieck;      (* erstellt eine Dreiecksmatrix *)
  ausgabe;      (* gibt die Dreiecksmatrix auf dem Bildschirm aus *)
END.
```

2.1.3 Rekursive Berechnung der Polynomkoeffizienten

Um dieses Problem lösen zu können, möge dieser Vorgang zunächst an einem Beispiel verdeutlicht werden. Nach dem bisherigen Stand bricht das Programm nach Ausgabe der Dreiecksmatrix ab. Am Beispiel von 3 Stützstellen könnte dies so aussehen:

a_2	a_1	a_0	y	$\Rightarrow$	a_2	a_1	a_0	y
(3;1)	(2;1)	(1;1)	(0;1)	$\Rightarrow$	1	1	1	2
(3;2)	(2;2)	(1;2)	(0;2)	$\Rightarrow$	0	-2	-3	-9
(3;3)	(2;3)	(1;3)	(0;3)	$\Rightarrow$	0	0	1	12

Für die rekursive Berechnung der Polynomkoeffizienten müßte man nun wie folgt verfahren:

$$[1;3] = \frac{[0;3]}{[1;3]} = \frac{12}{1} = 12$$

$$[2;3] = \frac{[0;2] - [1;2] * [1;3]}{[2;2]} = \frac{-9 - (-3) * 12}{-2} = 12$$

$$[3;3] = \frac{[0;1] - ([1;1] * [1;3] + [2;1] * [2;3])}{[3;1]} = \ldots = 3.5$$

Vorsicht! Beim Einsetzen der Werte muß man beachten, daß immer nur die aktuellen Werte einsetzt werden. Das könnten bereits diejenigen sein, die gerade zuvor berechnet wurden!

Man erkennt, daß sich das Problem mit zwei Schleifen lösen läßt. Die erste Schleife steuert dabei die eigentliche Rekursion, während in der zweiten Schleife die zunehmend längerwerdende „Produktensumme" berechnet wird. Das Endergebnis dieser „Produktensumme" soll vorübergehend in einer Hilfsvariablen h gespeichert werden, auf die dann die erste Schleife zugreifen kann. Nach dem obigen Beispiel ergibt sich:

$$[1;3] = \frac{[0;3]}{[1;3]} = \frac{12}{1} = 12$$

$$[2;3] = \frac{[0;2] - \overbrace{[1;2] * [1;3]}^{h}}{[2;2]} = \frac{-9 - (-3) * 12}{-2} = 12$$

$$[3;3] = \frac{[0;1] - (\overbrace{[1;1] * [1;3] + [2;1] * [2;3]}^{h})}{[3;1]} = \ldots = 3.5$$

Somit reduziert sich das Problem auf zwei Schritte:

1. Die Berechnung der Hilfvariablen h
2. Die Berechnung der restlichen Rekursion

1. Die Berechnung der Hilfvariablen h:

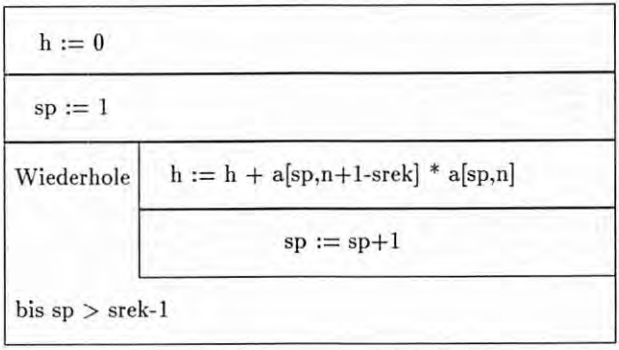
Der Wert der Schleifenvariablen der Rekursion srek minus eins gibt die Anzahl der Summen von Produkten in h an, d.h.:

- bei der ersten Berechnung der Koeffizienten ist der Rekursionsschleifenwert $srek = 1$. Die Produkte, die gebildet werden müssen um h zu bestimmen, sind Null. Dies berechnet sich aus $srek - 1 = 1 - 1 = 0$.
- bei der zweiten Berechnung der Koeffizienten ist der Rekursionsschleifenwert $srek = 2$. Die Produkte, die gebildet werden müssen um h zu bestimmen, sind eins. Dies berechnet sich aus $srek - 1 = 2 - 1 = 1$.

- bei der dritten Berechnung der Koeffizienten ist der Rekursionsschleifenwert $srek = 3$. Die Produkte, die gebildet werden müssen um h zu bestimmen, sind zwei. Dies berechnet sich aus $srek - 1 = 3 - 1 = 2$.

Die Anzahl der Durchläufe werden in der Variablen sp gespeichert.

Damit kann für die Berechnung von h ein Struktogramm erstellt werden:



2. Die Berechnung der restlichen Rekursion:

Für die restlichen Berechnungen diene nochmals das genannte Beispiel. Dabei ist das bereits berechnete h als fester Wert zu übernehmen:

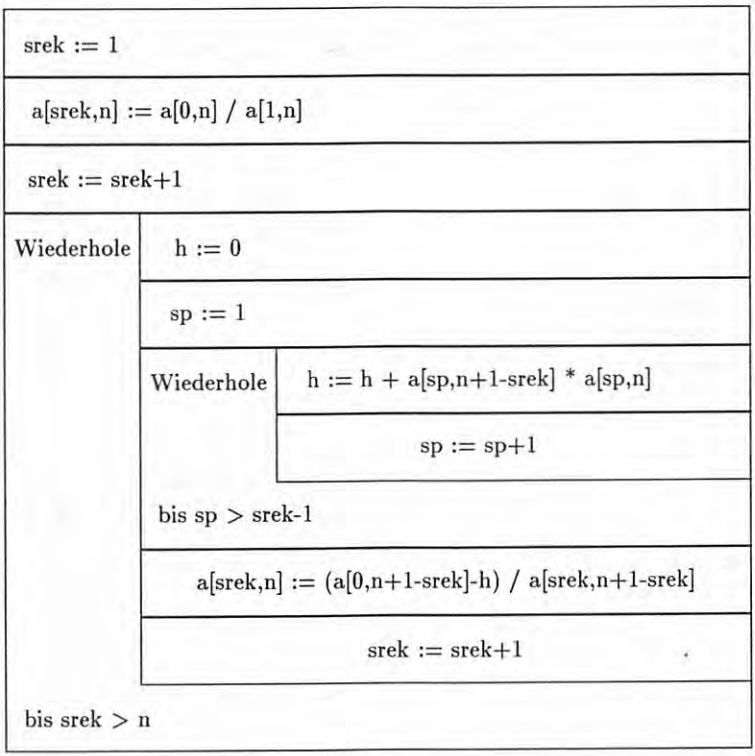
$$[1;3] = \frac{[0;3]}{[1;3]} \quad [2;3] = \frac{[0;2] - h}{[2;2]} \quad [3;3] = \frac{[0;1] - h}{[3;1]}$$

Betrachtet man daraufhin die Rekursion, die sich inzwischen sehr vereinfacht hat, so stellt man fest, daß eine Schleife benötigt wird, die von 1 bis n läuft. Somit ist der von $srek$ erfaßte Bereich ermittelt worden.

Die Berechnung des ersten Koeffizienten kann immer direkt erfolgen, ohne daß h berechnet zu werden braucht, denn h ist Null. Für alle weiteren Koeffizientenberechnungen läßt sich nun leicht folgende Zuweisung finden:

$$a[srec; n] = \frac{a[0; n + 1 - srek] - h}{a[srek; n + 1 - srek]}$$

Damit kann das vollständige Struktogramm erstellt werden. Dabei ist zu beachten, daß die erwähnte Zuweisung h bereits berechnet sein muß.



Am Ende des Algorithmus befinden sich die Koeffizienten des Interpolationspolynoms in der letzten Zeile des ARRAYs. Beim o.a. Beispiel würde das folgendermaßen aussehen:

a_2	a_1	a_0	y	⇒	a_2	a_1	a_0	y
(3;1)	(2;1)	(1;1)	(0;1)	⇒	1	1	1	2
(3;2)	(2;2)	(1;2)	(0;2)	⇒	0	-2	-3	-9
(3;3)	(2;3)	(1;3)	(0;3)	⇒	**3.5**	**-13.5**	**12**	12

Aus dem Struktogramm ergibt sich der folgende Turbo-Pascal-Quelltext.

```
PROGRAM Gaussalgorithmus; (* auf Diskette gauss_3.pas *)
uses crt;

CONST nmax = 50; (* Maximale ARRAY-Größe / veränderbar *)

VAR   i,j,n              : Byte;
      a                  : ARRAY[0..nmax,1..nmax] OF REAL;
      s1,s2,s3,srek,sp   : INTEGER;
      h                  : REAL;


PROCEDURE eingabe;
  BEGIN
    FOR i := 1 TO n DO
      BEGIN
        WRITE(' x',i,' = ');
          READLN(a[2,i]);
        WRITE(' y',i,' = ');
          READLN(a[0,i]);
        a[1,i] := 1;
        WRITELN;
      END;          (* Eingabe der Stützstellen beendet *)
  END;              (* Zuweisung von 1 in Spalte a0 ist erfolgt *)

PROCEDURE nullsuche;
  VAR hilf : REAL;
  BEGIN
    FOR i := 1 TO n DO
      IF a[2,i] = 0 then
        BEGIN
          hilf := a[2,n];
          a[2,n] := a[2,i];
          a[2,i] := hilf;
          hilf := a[0,n];
          a[0,n] := a[0,i];
          a[0,i] := hilf;
        END;
  END;
```

```
PROCEDURE matrix;
  BEGIN
    FOR i := 1 TO n DO
      FOR j := 3 TO n DO
        a[j,i] := a[j-1,i] * a[2,i];
  END;          (* Matrix fertig *)



PROCEDURE dreieck;
  BEGIN
    s3 := 1;
      REPEAT
        s2 := s3+1;
          REPEAT
            s1 := 1-s3;
            REPEAT
              a[s1,s2] := a[s1,s2] - ( (a[s1,s3] * a[n+1-s3,s2])
              / a[n+1-s3,s3] );
              s1 := s1+1;
            UNTIL s1 > n;
            s2 := s2+1;
          UNTIL s2 > n;
          s3 := s3+1;
      UNTIL s3 > n-1;
  END; (* gestaffeltes Gleichungssystem ist erstellt *)

PROCEDURE ausgabe;
  BEGIN
    CLRSCR;
      FOR i := 1 TO n DO
        BEGIN
          FOR j := n DOWNTO 1 DO
            BEGIN
              WRITE(a[j,i]:6:2);
              WRITE(' ');
            END;
          WRITELN(a[0,i]:6:2);
        END;
      READLN; (* Ausgabe der gesamten Matrix auf den Bildschirm *)
  END;
```

```
PROCEDURE koeffrek;
  BEGIN
    srek := 1;
    a[srek,n] := a[0,n]/a[1,n];
    srek := srek+1;

    REPEAT
      h := 0;
      sp:= 1;

        REPEAT
          h := h+ a[sp,n+1-srek] * a[sp,n];
          sp := sp+1;
        UNTIL sp > srek-1;

      a[srek,n] := (a[0,n+1-srek]-h) / a[srek,n+1-srek];
      srek := srek +1;
    UNTIL srek > n;

    FOR i := 1 TO n DO
      BEGIN
        FOR j := n DOWNTO 1 DO
          BEGIN
            WRITE(a[j,i]:6:2);
            WRITE(' ');
          END;
        WRITELN(a[0,i]:6:2);
      END;
      READLN;
  END;


BEGIN
  CLRSCR;
  WRITELN(' Algorithmus zum Erstellen des gestaffelten Gleichuns-');
  WRITELN(' systems und zur Berechnung der Koeffizienten des ');
  WRITELN(' Interpolationspolynoms');
  WRITELN;
  WRITE(' Wie viele Stützstellen sollen eingegeben werden ? ');
    READLN(n);
  WRITELN; WRITELN;
  eingabe;   (* liest die Stützstellenwerte ein *)
  nullsuche; (* schreibt die Stützstelle x=0 an das Ende, da *)
             (* der Algorithmus sonst versagt *)
  matrix;    (* erstellt aus den Stützstellen ein *)
             (* Gleichungssystem in Matrixform *)
```

```
  dreieck;    (* erstellt eine Dreiecksmatrix *)
  ausgabe;    (* gibt die Dreiecksmatrix auf dem Bildschirm aus *)
  koeffrek;   (* errechnet aus der Dreiecksmatrix die Koeffizienten *)
              (* des Interpolationspolynoms und gibt diese aus *)
END.
```

2.1.4 Übungen zur Vertiefung

Aufgabe 1:

Verbinden Sie die Programme gauss-3.pas und polynom.pas so, daß die berechneten Koeffizienten an das bereits erstellte Programm polynom.pas aus Kapitel 1.2 übergeben werden, und somit der Graph des Interpolationspolynoms bei optimaler Ausnutzung des Bildschrimes dargestellt werden kann. Nennen Sie das neu entstandene Programm gauss.pas.

Aufgabe 2:

Geben Sie für das erstellte Programm gauss.pas folgende Stützstellen ein:

x	-8	-6	-4	-2	0	2	4	6	8
y	2	4	0	2	0	-2	0	-4	-2

Ermitteln Sie die Koeffizienten des Interpolationspolynoms und den zugehörigen Graph im Intervall [-8; 8]!

Aufgabe 3:

Geben Sie für das erstellte Programm gauss.pas folgende Stützstellen ein:

x	-3	1	7	-9	2
y	-2.5	-1.5	22.5	38.5	0

Welches Ergebnis läßt sich aus den berechneten Koeffizienten des Interpolationspolynoms und dem zugehörigen Graph im Intervall [-3; 2] gewinnen?

2.2 Das Interpolationspolynom nach Lagrange

Nach Lagrange lassen sich Polynome allgemein in folgender Form darstellen:

$$f(x) = L_0(x) * f(x_0) + L_1(x) * f(x_1) + \ldots + L_n(x) * f(x_n)$$

bzw.:

$$y = L_0(x) * y_0 + y = L_1(x) * y_1 + \ldots + y = L_n(x) * y_n \,.$$

Daraus ergibt sich die Reihenschreibweise:

$$y = \sum_{i=0}^{n} L_i(x) * y_i \,.$$

$L_i(x)$ mit $i \in \mathcal{N}_0$ sind sogenannte Lagrangesche Fundamentalpolynome der Form:

$$L_i(x) = \frac{(x - x_0) * \ldots * (x - x_{i-1}) * (x - x_{i+1}) * (x - x_n)}{(x_i - x_0) * \ldots * (x_i - x_{i-1}) * (x_i - x_{i+1}) * (x_i - x_n)}$$

oder in Produktreihenschreibweise:

$$L_i(x) = \prod_{k=0 \wedge k \neq i}^{n} \frac{x - x_k}{x_i - x_k} \,.$$

Diese allgemeine Darstellung soll nun anhand eines mathematischen Beispiels erläutert werden:

Beispiel: Gegeben seien folgende Stützstellen:

x	1	2	3
y	2	-1	3

Zu ermitteln ist nach der Methode von Lagrange das Interpolationspolynom für die oben angegebenen Stützstellen.

Lösung: Aus der am Anfang dargestellten allgemeinen Form des Lagrangeschen Interpolationspolynoms ergibt sich folgende Gleichung:

$$y = y_0 \frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)} + y_1 \frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)} + y_2 \frac{(x - x_0)(x - x_1)}{(x_2 - x_0)(x_2 - x_1)}$$

Setzt man nun die o.a. Stützstellen ein, die sich allgemein wie folgt darstellen lassen,

x	x_0	x_1	x_2
y	y_0	y_1	y_2

dann erhält man folgenden Ausdruck:

$$y = 2\frac{(x-2)(x-3)}{(1-2)(1-3)} - 1\frac{(x-1)(x-3)}{(2-1)(2-3)} + 3\frac{(x-1)(x-2)}{(3-1)(3-2)} .$$

Nach dem Vereinfachen der Nenner (Berechnung der Quotienten) erhält man den Ausdruck:

$$y = 1(x-2)(x-3) + 1(x-1)(x-3) + 1,5(x-1)(x-2) .$$

Durch Ausmultiplizieren der Linearfaktoren kommt man zu:

$$\begin{aligned} y &= 1(x^2 - 3x - 2x + 6) + 1(x^2 - 3x - x + 3) \\ &\quad +1,5(x^2 - 2x - x + 2) \\ &= x^2 - 5x + 6 + x^2 - 4x + 3 + 1,5x^2 - 4,5x + 3 \end{aligned}$$

und erhält schließlich

$$y = 3,5x^2 - 13,5x + 12 .$$

Das Ziel ist jedoch, dieses mathematische Verfahren in einen Algorithmus und danach in ein Turbo-Pascal-Programm umzusetzen. Dabei erweist es sich als schwierig das Ausmultiplizieren der Linearfaktoren zu programmieren. Es ist daher erforderlich das oben dargestellte mathematische Verfahren so abzuwandeln, daß es programmierbar wird.

Computergerechter Algorithmus:

Analog zum mathematischen Algorithmus soll für die Stützstellen

x	1	2	3
y	2	-1	3

das entsprechende Interpolationspolynom ermittelt werden. Dabei wird jedoch der abgewandelte Algorithmus verwendet und an diesem Beispiel vorgestellt. Die Verallgemeinerung dieses Algorithmus findet im Anschluß an das Beispiel statt.

Lösung: Zunächst werden die Abzissenwerte der Stützstellen nochmals gesondert hervorgehoben:

$x_0 = 1 \; ; \; x_1 = 2 \; ; \; x_2 = 3 \; .$

Dem mathematischen Algorithmus entnimmt man die Quotienten der Lagrangeschen Interpolationspolynome. Das Programmieren dieser Quotienten dürfte keine Schwierigkeiten bereiten.
$q_0 = 1 \; ; \; q_1 = 1 \; ; \; q_2 = 1,5$
Nun werden Felder generiert, deren Anzahl und Größe von der Anzahl der eingegebenen Stützstellen abhängt. Es werden so viele Felder wie Stützstellen gebraucht. Jedes Feld ist so breit wie Stützstellen vorhanden sind und jedes Feld ist so lang wie die um Eins verminderte Anzahl von Stützstellen beträgt. Demnach benötigt man bei diesem Beispiel 3 Felder. Jedes der 3 Felder ist 3 Stellen breit und 2 Stellen lang!

Nun trägt man, den Lagrangeschen Fundamentalpolynomen entsprechend, die Abzissenwerte ein:

2	3			1	3			1	2	

Danach potenziert man die Abzissenwerte spaltenweise durch Multiplizieren mit sich selbst:

2	3			1	3			1	2	
4	9			1	9			1	4	

Schließlich werden zeilenweise die Summen gebildet:

2	3	5		1	3	4		1	2	3
4	9	13		1	9	10		1	4	5

Anschließend ist es notwendig, für jedes Feld die Koeffizienten zu berechnen. Dabei gilt folgende Berechnungsvorschrift. Zum Beispiel beim Vorhandensein von vier Summen:

$$\begin{aligned} a_4 &= 1 \; (Anfangswert) \\ a_3 &= -1 * (a_4 * s_1) \\ a_2 &= -\frac{1}{2} * (a_3 * s_1 + a_4 * s_2) \end{aligned}$$

$$a_1 = -\frac{1}{3} * (a_2 * s_1 + a_3 * s_2 + a_4 * s_3)$$
$$a_0 = -\frac{1}{4} * (a_1 * s_1 + a_2 * s_2 + a_3 * s_3 + a_4 * s_4)$$

Da im betrachteten Beispiel insgesamt pro Feld zwei Summen gebildet wurden, reduziert sich das System auf:

$$a_2 = 1\ (Anfangswert)$$
$$a_1 = -1 * (a_2 * s_1)$$
$$a_0 = -\frac{1}{2} * (a_1 * s_1 + a_2 * s_2)$$

Nach diesem Schema müssen nun für alle Felder die Koeffizienten berechnet werden:

Berechnung zu Feld 1:

$$a_2 = 1\ (Anfangswert)$$
$$a_1 = -1 * (1 * 5) = -5$$
$$a_0 = -\frac{1}{2} * ((-5) * 5 + 1 * 13) = 6$$

Berechnung zu Feld 2:

$$a_2 = 1\ (Anfangswert)$$
$$a_1 = -1 * (1 * 4) = -4$$
$$a_0 = -\frac{1}{2} * ((-4) * 4 + 1 * 10) = 3$$

Berechnung zu Feld 3:

$$\begin{aligned} a_2 &= 1\ (Anfangswert) \\ a_1 &= -1 * (1 * 3) = -3 \\ a_0 &= -\frac{1}{2} * ((-3) * 3 + 1 * 5) = 2 \end{aligned}$$

Als nächstes werden die Koeffizienten zur weiteren Berechnung zusammengestellt und mit den Quotienten multipliziert:

	a_2	a_1	a_0		
Feld 1	1	-5	6	*1	$[q_0]$
Feld 2	1	-4	3	*1	$[q_1]$
Feld 3	1	-3	2	*1,5	$[q_2]$

Daraus ergibt sich:

	a_2	a_1	a_0
Feld 1	1	-5	6
Feld 2	1	-4	3
Feld 3	1,5	-4,5	3

Danach wären spaltenweise die Summen zu berechnen:

	a_2	a_1	a_0
Feld 1	1	-5	6
Feld 2	1	-4	3
Feld 3	1,5	-4,5	3
Summe	3,5	-13,5	12

Die Summen sind nun die Koeffizienten des Lagrangeschen Interpolationspolynoms (Vergleiche mathematischen Algorithmus!). Mit diesem Algorithmus ist man also in der Lage die entsprechenden Koeffizienten zu berechnen. Es dürfte nun auch keine unüberwindlichen Probleme mehr bereiten, diesen abgewandelten Algorithmus in ein Turbo-Pascal-Programm umzusetzen. Es ist jedoch auf eine geschickte Wahl der ARRAYs zu achten, da sonst der Speicherplatz rasch belegt ist oder berechnete Werte in falsche ARRAYs gelangen und man so zu falschen Ergebnissen kommt.

Das soeben erläuterte Potenzsummenverfahren läßt sich in mathematischer Kurzschreibweise folgendermaßen zusammenfassen:

$$a_n = 1 \; (Anfangswert)$$

$$a_{n-i} = -\frac{1}{i} \sum_{j=1}^{i} (a_{n-i+j*S_j}) \; (Rekursionsvorschrift)$$

mit

$$S_j = \sum_{k=1}^{n} x_k^j$$

als Potenzsummen.

Dabei gilt: $n \mathrel{\widehat{=}}$ Anzahl der Stützstellen
$i, j, k \epsilon N$
$i \leq n$
$x_k \ldots$ Abszissenwerte der Stützstellen

Die Indizes $(n - i)$ mit $0 \leq i \leq n$ sind zugleich auch die Exponenten der dazugehörenden unabhängigen Variablen x. Es sei noch erwähnt, daß die sich durch das Ausmultiplizieren von Linearfaktoren ergebenden Koeffizienten a_i auch mit Hilfe des Satzes von Vieta als Summe der Kombinationen der n Abszissenwerte der Stützstellen zur i-ten Klasse ermittelt werden können. Die jeweilige Anzahl der Kombinationen beträgt dann

$$C_i(n) = \binom{n}{i} = \frac{n!}{i!(n-i)!}$$

Zur Veranschaulichung dieses Verfahrens seien folgende Abszissenwerte gegeben:

$x_1 = -1; x_2 = 1; x_3 = 2; x_4 = 3$.

Das entsprechende Polynom lautet:

$$(x+1)(x-1)(x-2)(x-3) = a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0 \; .$$

Die Koeffizienten a_i lassen sich nach Vieta folgendermaßen berechnen:

$$a_4 = 1$$

$$a_3 = -(x_+x_2 + x_3 + x_4) = -5$$

$$a_2 = x_1x_2 + x_1x_3 + x_1x_4 + x_2x_3 + x_2x_4 + x_3x_4 = 5$$

$$a_1 = -(x_1x_2x_3 + x_1x_2x_4 + x_1x_3x_4 + x_2x_3x_4) = 5$$

$$a_0 = x_1x_2x_3x_4 = -6$$

Also:

$$(x+1)(x-1)(x-2)(x-3) = x^4 - 5x^3 + 5x^2 + 5x - 6$$

2.2.1 Ermittlung der Teilpolynome mit Hilfe von Potenzsummen

In Kapitel 2.2 ist der abgewandelte Algorithmus bereits am Beispiel von drei Stützstellen vorgestellt worden. Dieser soll nun allgemeiner gefasst werden, damit das Algorithmisieren erleichtert wird.

a) Sicherung der Eingaben:

Zunächst werden die Abzissenwerte der Stützstellen nach der Eingabe in einem ARRAY gespeichert:

$$\begin{aligned} x_0 &= (Zahlenwert) \\ x_1 &= (Zahlenwert) \\ x_2 &= (Zahlenwert) \\ &\vdots \\ x_n &= (Zahlenwert) \end{aligned}$$

b) Berechnung der Produkte:

Aus den Lagrangeschen Fundamentalpolynomen resultiert im Nenner jeweils eine Produktreihe ohne Variable. Es ist daher als nächstes erforderlich diese Produktreihe zu berechnen. Der Algorithmus dafür dürfte schnell gefunden sein.

c) Berechnung der Quotienten:

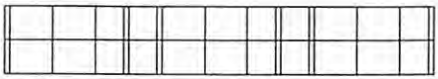
Sind die Produktreihen ermittelt worden, so ist es erforderlich die Quotienten aus den y-Werten der Stützstellen und den Produktreihen zu berechnen.

d) Nachdem die Quotienten gespeichert worden sind, ist mit dem Erstellen der Lagrangeschen Fundamentalpolynome in dafür vorbereiteten ARRAYs zu beginnen. Da das Beispiel in Kapitel 2.2 nur auf 3 Stützstellen beruht, wird das Verfahren noch einmal allgemein erläutert:

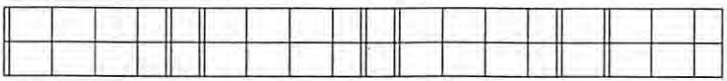
e) Allgemein gilt:

Es werden so viele Felder wie Stützstellen gebraucht. Jedes Feld ist so viele Stellen breit wie Stützstellen vorhanden sind. Jedes Feld ist so viele Stellen lang wie die um Eins verminderte Anzahl der Stützstellen beträgt.

Bei 3 Stützstellen:



Bei 4 Stützstellen:



Nun werden, den Lagrangeschen Fundamentalpolynomen entsprechend, die Abzissenwerte eingetragen:

Bei 3 Stützstellen:

x_1	x_2			x_0	x_2			x_0	x_1	

Bei 4 Stützstellen:

x_1	x_2	x_3			x_0	x_2	x_3			x_0	x_1	x_3			x_0	x_1	x_2	

Anmerkung:

Der jeweils mit der Ordnungszahl eines Fundamentalpolynoms indizierte x- Wert fehlt. Z.B.: Beim 0. Fundamentalpolynom fehlt x_0, beim 1. Fundamentalpolynom fehlt x_1, beim 2. fehlt x_2 usw.

Anschließend potenziert man die Abzissenwerte spaltenweise durch Multiplizieren mit sich selbst und bildet zeilenweise die Summen (Vergleiche Beispiel in 2.2)

2.2.2 Ermittlung der Koeffizienten des Interpolationspolynoms

Auch im Kapitel 2.2.2 wird mit der Verallgemeinerung des in Kapitel 2.2 vorgestellten, abgewandelten Algorithmus zur Berechnung des Lagrangeschen Interpolationspolynoms fortgefahren. Verallgemeinert man

den Algorithmus zur Berechnung der Linearfaktorprodukte (in Kapitel 2.2 Berechnung der einzelnen Felder genannt), kommt man auf folgende Darstellung:

$$
\begin{aligned}
a_n &= 1\ (Anfangswert) \\
a_{n-1} &= -1 * (a_n * s_1) \\
a_{n-2} &= -\frac{1}{2} * (a_{n-1} * s_1 + a_n * s_2) \\
a_{n-3} &= -\frac{1}{3} * (a_{n-2} * s_1 + a_{n-1} * s_2 + a_n * s_3) \\
a_{n-4} &= -\frac{1}{4} * (a_{n-3} * s_1 + a_{n-2} * s_2 + a_{n-1} * s_3 + a_n * s_4) \\
&\vdots \\
a_{n-n} &= -\frac{1}{n} * (a_1 * s_1 + a_2 * s_2 + a_3 * s_3 + a_4 * s_4 + a_n * s_n)
\end{aligned}
$$

Im Anschluß daran werden die berechneten Koeffizienten zur weiteren Berechnung nochmals zusammengestellt und dann mit den gespeicherten Quotienten multipliziert:

	a_n	...	a_2	a_1	a_0	
Feld 1	1	...				* $[q_0]$
Feld 2	1	...				* $[q_1]$
Feld 3	1	...				* $[q_2]$
⋮	⋮	⋮	⋮	⋮	⋮	⋮
Feld n	1	...				* $[q_n]$

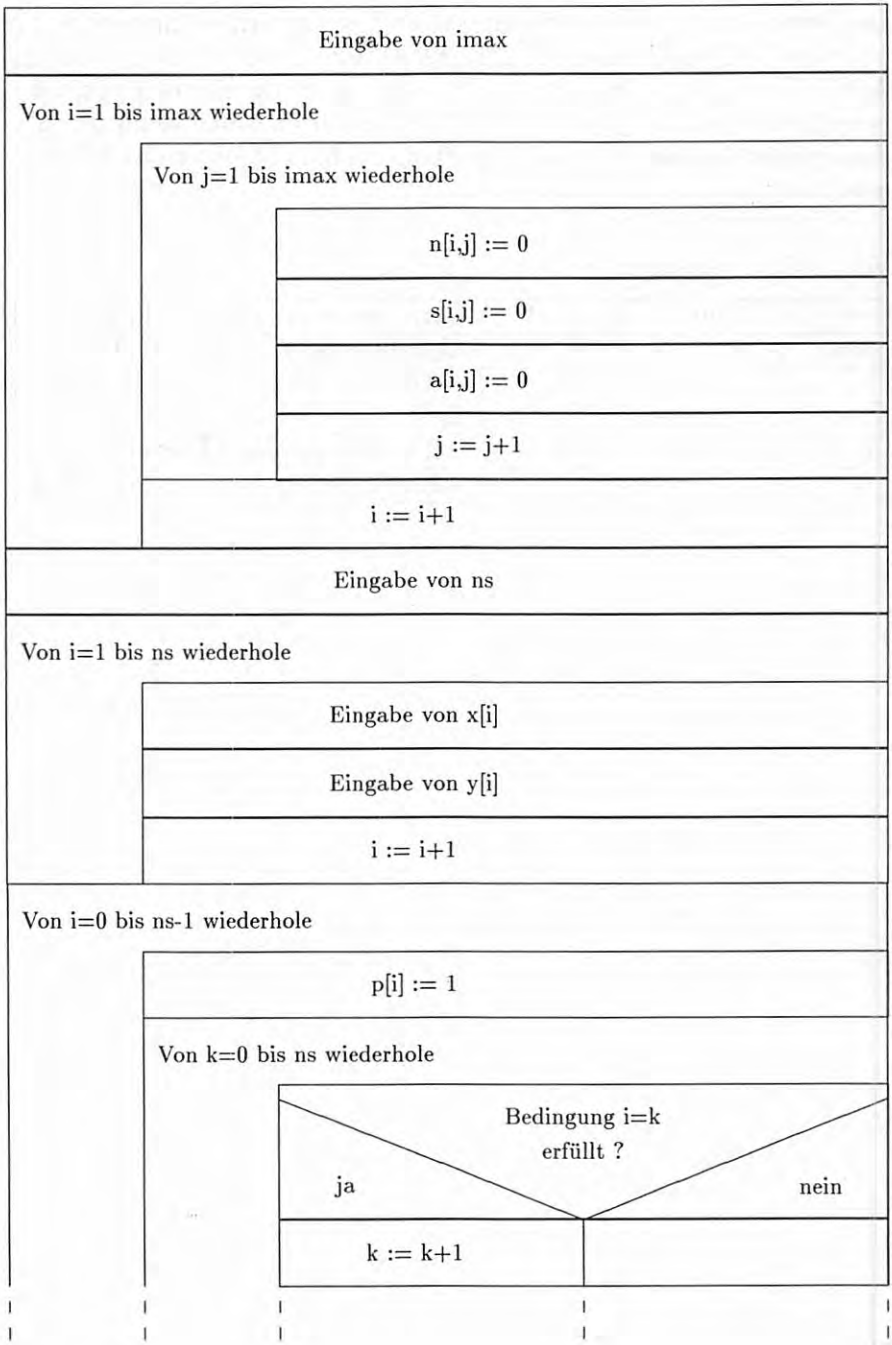
Nun werden spaltenweise die Summen berechnet (Vgl. Kap. 2.2):

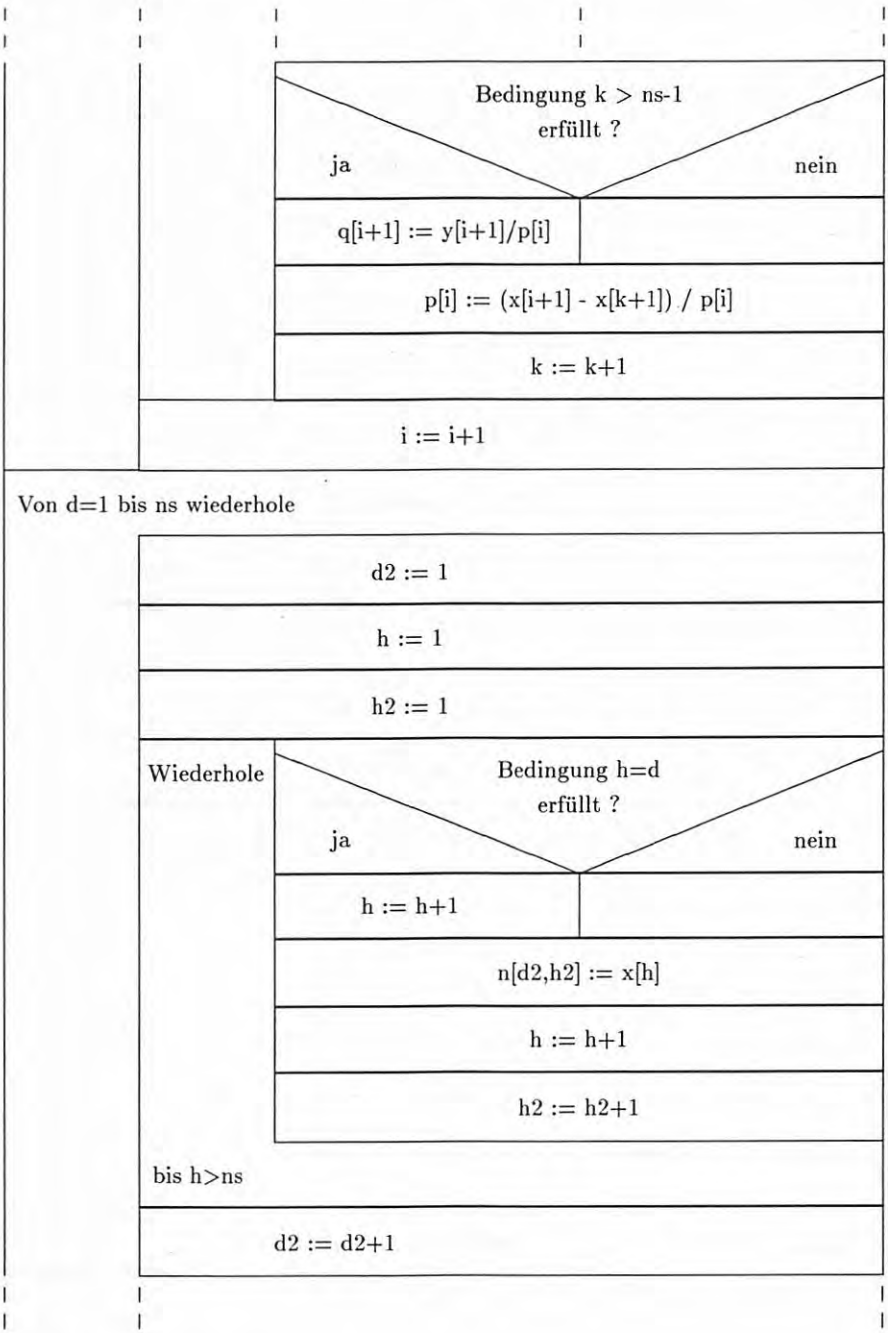
	a_n	...	a_2	a_1	a_0	
Feld 1	q_0	...				* $[q_0]$
Feld 2	q_1	...				* $[q_1]$
Feld 3	q_2	...				* $[q_2]$
⋮	⋮	⋮	⋮	⋮	⋮	⋮
Feld n	q_n	...				* $[q_n]$
Summe						

Die Summen sind die Koeffizienten des Lagrangeschen Interpolationspolynoms (Vergleiche Kapitel 2.2). Es ist auf eine geschickte Wahl der ARRAYs zu achten, da sonst der Speicherplatz schnell belegt ist, oder berechnete Werte in falsche ARRAYs gelangen und man so zu falschen Ergebnissen kommt. Diesen Algorithmus veranschaulicht das folgende Struktogramm:

Verwendete Variablen:

sw, z, d, d2, ns, i, j, k, h, h2	:	Variablen vom Datentyp Integer
wert	:	Variable vom Datentyp Real
n, s, a	:	ARRAY [1..50,1..50] vom Datentyp Real
q, x, y, koef	:	ARRAY [1..50] vom Datentyp Real
p	:	ARRAY [0..50] vom Datentyp Real


Eingabe von imax
Von i=1 bis imax wiederhole
Von j=1 bis imax wiederhole
n[i,j] := 0
s[i,j] := 0
a[i,j] := 0
j := j+1
i := i+1
Eingabe von ns
Von i=1 bis ns wiederhole
Eingabe von x[i]
Eingabe von y[i]
i := i+1
Von i=0 bis ns-1 wiederhole
p[i] := 1
Von k=0 bis ns wiederhole
Bedingung i=k
erfüllt ?
ja
nein
k := k+1


Bedingung k > ns-1
erfüllt ?
ja
nein
q[i+1] := y[i+1]/p[i]
p[i] := (x[i+1] - x[k+1]) / p[i]
k := k+1
i := i+1
Von d=1 bis ns wiederhole
d2 := 1
h := 1
h2 := 1
Wiederhole
Bedingung h=d
erfüllt ?
ja
nein
h := h+1
n[d2,h2] := x[h]
h := h+1
h2 := h2+1
bis h>ns
d2 := d2+1

```
Von z=1 bis ns wiederhole
    Von i=2 bis ns-1 wiederhole
        n[i,z] := n[i-1,z] * n[1,z]
        i := i+1
    z := z+1
Von z=1 bis ns wiederhole
    Von i=1 bis ns wiederhole
        s[z,d] := s[z,d] + n[z,i]
        i := i+1
    z := z+1
a[ns,d] := 1
Von sw=1 bis ns wiederhole
    j := 0
    wert := 0
    Wiederhole
        wert := wert + a[ns-sw+1+j,d]*s[j+1,d]
        j := j+1
    bis j+1 > sw
    a[ns-sw,d] := -1 / sw * wert
    sw := sw+1
```

```
                    d := d+1
Von i=1 bis ns+1 wiederhole
                    Von j=1 bis ns+1 wiederhole
                                        a[j,i] := a[j,i] * q[i]
                                        j := j+1
                    i := i+1
                    i := ns-1
Wiederhole          koef[i] := 0
                    Von j=1 bis ns+1 wiederhole
                                        koef[i] := koef[i] + a[i+1,j]
                                        j := j+1
                    Ausgabe von koef[i]
                    i := i-1
bis i<-1
```

2.2.3 Übungen zur Vertiefung

Aufgabe:

a) Ermitteln Sie zunächst ohne PC nach der Methode von Lagrange die Funktionsgleichung zu folgender Wertetabelle:

x	1	4	6
y	2	5	3

b) Entwickeln Sie aus dem Struktogramm ein geeignetes Turbo-Pascal-Programm mit dem Dateinamen lagra_b.pas. Überprüfen Sie die semantische Korrektheit des Programms durch die bei a) berechneten Werte.

c) Verknüpfen Sie das Programm lagra_b.pas mit dem bereits erstellten Programm polynom.pas so, daß die errechneten Koeffizienten übergeben werden und der Graph der Funktion dargestellt werden kann. Nennen Sie das so entstandene Programm lagrange.pas.

2.3 Das Interpolationspolynom nach Newton

Für (n+1) Stützstellen lautet das Newtonsche Interpolationspolynom allgemein

$$y = p_0 + \sum_{i=0}^{n} [\, p_i \prod_{k=0}^{i-1} (x - x_k) \,]$$

mit $i,\ n \in N_0 \wedge k \in N_0$. Dabei lassen sich die Koeffizienten p_k aus dem linearen Gleichungssystem $y_0 = p_0$ und

$$y_m = p_0 + \sum_{i=1}^{m} [\, p_i \prod_{k=0}^{i-1} (x_m - x_k) \,]$$

rekursiv ermitteln.

Der Vorteil der Newton-Methode besteht darin, daß bei Hinzufügen weiterer Stützstellen lediglich das Gleichungssystem sowie das Polynom erweitert werden müssen. Bei der Lagrange-Methode müßte in diesem Fall der gesamte Vorgang wiederholt werden.

Beispiel für das Newtonsche Interpolationspolynom:

Aufgabe: Gegeben sind folgende Stützstellen

x	0	1	2	3
y	6	2	-2	6

Zu ermitteln ist das Interpolationspolynom nach der Methode von Newton!

Lösung:

Aus der obigen Gleichung folgt:

$$y = p_0 + p_1\,(x - x_0) + p_2\,(x - x_0)(x - x_1) + p_3\,(x - x_0)(x - x_1)(x - x_2)$$

Dabei sind die Koeffizienten p_i (hier p_0 bis p_3) dadurch zu bestimmen, indem die Wertepaare der Stützstellen in die obige Gleichung eingesetzt werden:

$$\begin{aligned}
y_0 &= p_0 \\
y_1 &= p_0 + p_1\,(x_1 - x_0) \\
y_2 &= p_0 + p_1\,(x_2 - x_0) + p_2\,(x_2 - x_0)(x_2 - x_1) \\
y_3 &= p_0 + p_1\,(x_3 - x_0) + p_2\,(x_3 - x_0)(x_3 - x_1) \\
&\quad + p_3\,(x_3 - x_0)(x_3 - x_1)(x_3 - x_2)
\end{aligned}$$

bzw.

$$\begin{aligned}
6 &= p_0 \\
2 &= p_0 + p_1 * 1 \\
-2 &= p_0 + p_1 * 2 + p_2 * 2 * 1 \\
6 &= p_0 + p_1 * 3 + p_2 * 3 * 2 + p_3 * 3 * 2 * 1
\end{aligned}$$

bzw.

$$\begin{aligned}
6 &= p_0 \\
2 &= p_0 + 1 * p_1 \\
-2 &= p_0 + 2 * p_1 + 2 * p_2 \\
6 &= p_0 + 3 * p_1 + 6 * p_2 + 6 * p_3
\end{aligned}$$

Hierbei handelt es sich um ein gestaffeltes lineares Gleichungssystem, aus dem man die Koeffizienten (vergleichbar mit der Berechnung der Variablenwerte beim Gaußalgorithmus) rekursiv ermitteln kann:

$$
\begin{aligned}
p_0 &= 6 \\
p_1 &= \frac{2 - p_0}{1} = -4 \\
p_2 &= \frac{-2 - p_0 - 2p_1}{2} = 0 \\
p_3 &= \frac{6 - p_0 - 3p_1 - 6p_2}{6} = 2
\end{aligned}
$$

Auf diese Weise erhält man das Interpolationspolynom:

$$
\begin{aligned}
y &= 6 - 4\,(x-0) + 0\,(x-0)(x-1) + 2\,(x-0)(x-1)(x-2) \\
y &= 6 - 4x + 2x^3 - 4x^2 - 2x^2 + 4x \\
y &= 6 - 4x + 2x(x-1)(x-2) \\
y &= 2x^3 - 6x^2 + 6
\end{aligned}
$$

2.3.1 Ermittlung der Koeffizienten der Newtonschen Linearfaktorprodukte mit Hilfe dividierter Differenzen

Die Koeffizienten der Linearfaktorproduke des Newtonverfahrens lassen sich (rekursiv) einfacher durch die Bildung dividierter Differenzen berechnen. Letztere werden folgendermaßen gebildet:

$$
\begin{aligned}
\frac{\Delta_{01}y}{\Delta_{01}x} &= \frac{y_1 - y_0}{x_1 - x_0} = y_{11} \;;\; \frac{\Delta_{02}y}{\Delta_{02}x} = \frac{y_2 - y_1}{x_2 - x_1} = y_{12}; \ldots \\
\frac{\Delta_{11}y}{\Delta_{11}x} &= \frac{y_{12} - y_{11}}{x_2 - x_0} = y_{21} \;;\; \frac{\Delta_{12}y}{\Delta_{12}x} = \frac{y_{13} - y_{12}}{x_3 - x_1} = y_{22}; \ldots \\
\frac{\Delta_{21}y}{\Delta_{21}x} &= \frac{y_{22} - y_{21}}{x_3 - x_0} = y_{31} \;;\; \frac{\Delta_{22}y}{\Delta_{22}x} = \frac{y_{23} - y_{22}}{x_4 - x_1} = y_{22}; \ldots
\end{aligned}
$$

Wendet man diese dividierten Differenzen auf das Beispiel aus Kapitel 2.3 an, so erhält man:

$$
\begin{aligned}
y_{11} &= \frac{y_1 - y_0}{x_1 - x_0} = \frac{2-6}{1-0} = -4 \\
y_{12} &= \frac{y_2 - y_1}{x_2 - x_1} = \frac{-2-2}{2-1} = -4 \\
y_{13} &= \frac{y_3 - y_2}{x_3 - x_2} = \frac{6-(-2)}{3-2} = 8 \\
y_{21} &= \frac{y_{12} - y_{11}}{x_2 - x_0} = \frac{-4-(-4)}{2-0} = 0 \\
y_{22} &= \frac{y_{13} - y_{12}}{x_3 - x_1} = \frac{8-(-4)}{3-1} = 6 \\
y_{31} &= \frac{y_{22} - y_{21}}{x_3 - x_0} = \frac{6-0}{3-0} = 2
\end{aligned}
$$

Ordnet man die so gewonnenen y-Werte in eine Matrix ein, dann stellen die Diagonalenelemente die gesuchten Koeffizienten dar:

x_i	y_i	y_{0i}	y_{1i}	y_{2i}	y_{3i}
0	6	**$6 = p_0$**	—	—	—
1	2	—	**$-4 = p_1$**	—	—
2	-2	—	-4	**$0 = p_2$**	—
3	6	—	8	6	**$2 = p_3$**

Daraus resultiert folgendes Struktogramm:

```
Eingabe der Anzahl der Stützstellen n
Von i=0 bis n-1 wiederhole
    Eingabe von x[i]
    Eingabe von y[i]
    i := i+1
ya[0,1] := y[0]
Von i=1 bis ns-1 wiederhole
    ya[1,i] := (y[i]-y[i-1]) / (x[i]-x[i-1])
    i := i+1
Von z=2 bis n-1 wiederhole
    Von i=1 bis n-z wiederhole
        ya[z,i] := (ya[z-1,i+1]-ya[z-1,1])/(x[z-1+i]-x[i-1])
        i := i+1
    z := z+1
```

Variablen:

ya, x, y	:	ARRAY vom Datentyp Real
n,i,z	:	Variablen vom Datentyp Integer

Verwendet man außerdem noch äquidistante Stützstellen mit: $x_{i+1} - x_i = c$, dann braucht man die Differenzen der y-Werte lediglich durch ein ganz-

zahliges Vielfaches von c zu dividieren:

$$\frac{\Delta_{ij}y}{\Delta_{ij}x} = \frac{\Delta_{ij}y}{(i+1)*c}$$

Da in unserem Beispiel bereits äquidistante Stützstellen verwendet wurden, soll diese Vereinfachung im folgenden verdeutlicht werden:

$$c = x_{i+1} - x_i$$

zum Beispiel: $c = x_1 - x_0 = 1 - 0 = 1$

$$\begin{aligned}
y_{11} &= \frac{y_1 - y_0}{1*c} = \frac{2-6}{1} = -4 \\
y_{12} &= \frac{y_2 - y_1}{1*c} = \frac{-2-2}{1} = -4 \\
y_{13} &= \frac{y_3 - y_2}{1*c} = \frac{6-(-2)}{1} = 8 \\
y_{21} &= \frac{y_{12} - y_{11}}{2*c} = \frac{-4-(-4)}{2} = 0 \\
y_{22} &= \frac{y_{13} - y_{12}}{2*c} = \frac{8-(-4)}{2} = 6 \\
y_{31} &= \frac{y_{22} - y_{21}}{3*c} = \frac{6-0}{3} = 2
\end{aligned}$$

2.3.2 Algorithmus zur Berechnung der Polynomkoeffizienten bei äquidistanten Stützstellen unter Anwendung des Vietaschen Wurzelsatzes

Bevor die endgültige Form des Polynoms angegeben werden kann, müssen noch die sich beim Ausmultiplizieren der Linearfaktoren ergebenden Koeffizienten ermittelt werden. Diese lassen sich mit Hilfe des Vietaschen Wurzelsatzes bestimmen. Ausgehend von

$$(x-x_1)(x-x_2)(x-x_3)*\ldots*(x-x_n) = q_n x^n + q_{n-2}x^{n-2} + \ldots + a_0 x^0$$

folgt nach Vieta daraus

$$
\begin{aligned}
q_n &= 1 \\
q_{n-1} &= -(x_1 + x_2 + x_3 + \ldots + x_n) \\
q_{n-2} &= x_1x_2 + x_1x_3 + \ldots + x_2x_3 + x_2x_4 + \ldots + x_{n-1}x \\
q_{n-1} &= x_1x_2x_3 + x_1x_2x_4 + \ldots + x_2x_3x_4 + x_2x_3x_5 + \ldots + x_{n-2}x_{n-1}x_n \\
&\vdots \\
q_0 &= (-1)^n x_1x_2x_3 * \ldots * x_n
\end{aligned}
$$

Allgemein ausgedrückt sind die Koeffizienten q_{n-1} Kombinationen von n Elementen zur i-ten Klasse mit alternierenden Vorzeichen. Der hierfür geeignete Algorithmus bei Vorhandensein äquidistanter Stützstellen ist im Programm newton_c.pas als Sourcefile wiedergegeben worden.

```
PROGRAM newton_c;
uses crt;

VAR
  i,anzahl,p : INTEGER;
  anfang,deltax,res: REAL ;

FUNCTION vietabasis(anfang: REAL; anzahl,p: INTEGER): REAL;
  VAR
    k : INTEGER;
    r : REAL;
  BEGIN
    r := 0;
    IF p <> 0 THEN
      BEGIN
        FOR k := 0 TO anzahl-p DO
          r := r + (anfang + k*deltax) *
          vietabasis(anfang + (k+1)*deltax,anzahl - (k+1),p-1);
        vietabasis:= r
      END
    ELSE
        vietabasis:= 1
  END;
```

```
BEGIN
  CLRSCR;
  WRITELN('Programm Vieta: Dieses Programm ermittelt die ');
  WRITE('Koeffizienten des sich ');
  WRITELN('aus den Linearfaktorprodukten ergebenden Polynoms ');
  WRITELN('bei Verwendung äquidistanter Stützstellen.');
  WRITELN;
  WRITE('Nullstelle des ersten Linearfaktors (x-x(0)): x(0) =');
    READLN(anfang);
  WRITE('Abstand der Nullstellen der Linearfaktoren (Äquidistanz):');
  WRITE('delta x = ');
    READLN(deltax);
  WRITE('Anzahl der Linearfaktoren : n = ');
    READLN(anzahl);
  WRITELN;
  WRITELN('Die Koeffizienten lauten:');
  FOR i := 0 TO anzahl DO
    BEGIN
      IF i <> anzahl THEN
        BEGIN
          p := anzahl - i;
          res := vietabasis(anfang,anzahl,p);
          IF p MOD 2 = 1 THEN res := -res
        END
      ELSE
        res := 1;
      WRITELN('q(',i,') = ',res:3:2)
    END
END.
```

Da sich dieser Algorithmus als Struktogramm nur in umständlicher Weise darstellen läßt, wird dafür der entsprechende Quelltext wiedergegeben. Die Realisierung der relevanten Kombinationen erfolgt mit Hilfe einer Rekursion, wobei nicht nur der bei rekursiven Funktionen übliche Rückgriff auf sich selbst erfolgt, sondern auch während der Funktionswertberechnung auf die Rekursion selbst zurückgegriffen wird. Damit ähnelt dieser Algorithmus der bekannten Ackermann- Funktion. Im Folgenden soll diese Rekursion, die durch die „function" in newton_c.pas realisiert wird, am Beispiel der Berechnung von q_0 - der Absolutwert von q_0 entspricht

in der „function" der Variablen r für i gleich Null - bei 3 Linearfaktoren („anzahl" = 3), einen Anfangswert 2 („anfang" = 2) und der Äquidistanz („deltax" = 1), erläutert werden. Das Produkt der Linearfaktoren $(x-2)(x-3)(x-4)$ ergibt unter Einbeziehung des Vietaschen Wurzelsatzes

$$q_3x^3 + q_2x^2 + q_1x - q_0 = x^3 - 9x^2 + 26x - 24$$

und damit für $q_0 = -24$.

Verfolgt man die Änderung der Variablen innerhalb der „function" zu Beginn und am Ende der FOR-Schleife, so erhält man die Wertetabelle:

anfang	anzahl	p	k	r	Bemerkungen
2.00	3	3	0	0.00	Schleifenanfang
3.00	2	2	0	0.00	
4.00	1	1	0	0.00	
4.00	1	1	0	4.00	Schleifenende
3.00	2	2	0	12.00	
2.00	3	3	0	24.00	

Dieser rekursive Vorgang läßt sich folgendermaßen veranschaulichen:

```
r=vietabasis(2,3,3)
   =2*vietabasis(3,2,2)
      =3*vietabasis(4,1,1)
         =4*vietabasis(5,0,0)
            =1
         = 4*1
      =3*4
   =2*12
```

Also: $r = 24 = |q_0|$

Entsprechende Wertetabellen ließen sich auch für die übrigen Koeffizienten q_1, q_2 und q_3 erstellen. Vorzeichengerechte Koeffizienten erhält man durch Anwendung der Bedingung $p \; mod \; 2 = 1$. Immer dann, wenn diese Bedingung erfüllt ist, wird der entsprechende Koeffizient mit einem Minuszeichen versehen.

Nachdem die Koeffizienten q_i mit Hilfe eines rekursiven Algorithmus bestimmt worden sind, werden diese geordnet in einen ARRAY geschrieben. Anschließend multipliziert man zeilenweise die Koeffizienten q_i mit den

Koeffizienten der Liniearfaktorprodukte p_i und addiert dann spaltenweise die Koeffizienten gleichwertiger Potenzen von q_i. Man erhält dann die Koeffizienten a des gesuchten Interpolationspolynoms. Um das Verfahren zu veranschaulichen diene als Beispiel folgende Darstellung mit 3 Linearfaktoren:

Zunächst werden die Koeffizienten q_i in ein ARRAY eingelesen:

q_{33}	q_{32}	q_{31}	q_{30}
	q_{22}	q_{21}	q_{20}
		q_{11}	q_{10}
			q_{00}

Danach multipliziert man zeilenweise mit den Koeffizienten p_i:

q_{33}	q_{32}	q_{31}	q_{30}	$*p_3$
	q_{22}	q_{21}	q_{20}	$*p_2$
		q_{11}	q_{10}	$*p_1$
			q_{00}	(hier keine Multiplikation)

Nun addiert man spaltenweise und erhält die Koeffizienten des Interpolationspolynoms:

	q_{33}	q_{32}	q_{31}	q_{30}
+		q_{22}	q_{21}	q_{20}
+			q_{11}	q_{10}
+				q_{00}
Σ	$\mathbf{a_3}$	$\mathbf{a_2}$	$\mathbf{a_1}$	$\mathbf{a_0}$

Somit kann man aus der letzten Zeile des ARRAYs die Koeffizienten des Interpolationspolynoms entnehmen.

Für das eingangs erwähnte Beispiel bedeutet dies, daß sich letztlich folgendes Polynom ergibt:

$$y = 2s^3 - 6x^2 + 6$$

Nachfolgend wird der Einfachheit halber der vollständige Quelltext des in Kapitel 2.3.2 erläuterten Algorithmus wiedergegeben.

```
PROGRAM newton_d;
uses crt;

CONST max = 50;
VAR
  anzahl,i : INTEGER;
  anfang,deltax: REAL;
  y : ARRAY [0..max,0..max] OF REAL;

PROCEDURE eingabe;
  BEGIN
    WRITELN('Interpolationspolynom nach Newton: Dieses Programm erzeugt');
    WRITELN('aus äquidistanten Stützstellen nach dem Verfahren von');
    WRITELN('Newton ein Interpolationspolynom!');
    WRITELN;
    WRITELN;
      WRITE('Anzahl der Stützstellen (max. ',max,') : n = ');
        READLN(anzahl);
      WRITE('Startwert für x : x(0)= ');
        READLN(anfang);
      WRITE('Äquidistanz (Schrittweite) : delta x = ');
        READLN(deltax);
    FOR i:=0 TO anzahl-1 DO
      BEGIN
        WRITE('Y-Wert an der Stelle x = ',anfang+i*deltax:3:2,' : ');
        READLN(y[0,i])
      END
  END;

PROCEDURE matrix;
  VAR i,k : INTEGER;
  BEGIN
    FOR i :=1 TO anzahl - 1 DO
      FOR k := i TO anzahl -1 DO
        y[i,k] := (y[i-1,k] - y[i-1,k-1] ) / (deltax*i)
  END;
```

```
FUNCTION vietabasis(anfang: REAL; k,m: INTEGER) :REAL;
  VAR
    n : INTEGER;
    s : REAL;
  BEGIN
    s:=0;
      IF m <> 0 THEN
        BEGIN
          FOR n := 0 TO k-m DO
            s := s + (anfang + n*deltax) *
            vietabasis(anfang + (n+1)*deltax,k - (n+1),m - 1);
          vietabasis := s
        END
      ELSE
        vietabasis := 1
  END;
FUNCTION vieta(anfang : REAL; i,k : INTEGER) : REAL;
  VAR
    m : INTEGER;
    r : REAL;

  BEGIN
    IF i <> k THEN
      BEGIN
        m := k-i;
        r := vietabasis(anfang,k,m);
          IF m MOD 2 = 1 THEN r := -r;
        vieta := r
      END
    ELSE
      vieta := 1
  END;

FUNCTION polynom_koeff(i : INTEGER) : REAL;
  VAR
    k : INTEGER;
    ergebnis: REAL;
  BEGIN
    ergebnis := y[i,i];
      FOR k:=i+1 TO anzahl-1 DO
        ergebnis := ergebnis + y[k,k] * vieta(anfang,i,k);
    polynom_koeff := ergebnis
  END;
```

```
  PROCEDURE ausgabe;
  BEGIN
    WRITELN('Ausgabe der Koeffizienten:');
    WRITELN;
      FOR i:= 0 TO anzahl - 1 DO
        WRITELN('Koeffizient a(',i,') = ',polynom_koeff(i):4:3)
  END;
BEGIN
  CLRSCR;
  eingabe;
  matrix;
  CLRSCR;
  ausgabe
END.
```

2.3.3 Übungen zur Vertiefung

Aufgabe:

a) Ermitteln Sie zunächst ohne PC nach der Methode von Newton ohne und mit dividierter Differenzen das Interpolationspolynom mit folgenden Stützstellen:

x	0	1	2	3
y	-1	2	-5	4

b) Entwickeln Sie ein Turbo-Pascal-Programm mit dem Dateinamen newton_b.pas zur Ermittlung der Koeffizienten der Newtonschen Linearfaktorprodukte mit Hilfe von dividierten Differenzen bei allgemeinen Stützstellen. Überprüfen Sie die Richtigkeit des Programmes durch die bei a) berechneten Werte.

c) Überprüfen Sie die bei a) berechneten Werte mit Hilfe des unter dem Dateinamen newton_c.pas dargestellten Algorithmus. Entwickeln Sie für q1 - vergleichbar mit dem angegebenen Beispiel - die Wertetabelle der FOR-Schleife und veranschaulichen Sie diesen rekursiven Vorgang!

d) Überprüfen Sie die Korrektheit des gesamten in newton_d.pas wiedergegebenen Algorithmus anhand des Aufgabenteils a)!

e) Verknüpfen Sie Programm newton_d.pas mit dem bereits erstellten Programm polynom.pas, so daß die berechneten Koeffizienten übergeben werden und der Graph dargestellt werden kann. Nennen Sie das neu entstandene Programm newton.pas.

3. Numerisches Differenzieren

In diesem Kapitel soll erläutert werden, wie der Graph einer Funktion und die zugehörigen Ableitungsfunktionen dargestellt werden können. Voraussetzung hierfür ist, daß Stützstellen gegeben sind. Dieses Problem läßt sich mit Hilfe des modifizierten Hornerschemas lösen, das im folgenden Kapitel erläutert wird. Anschließend wird erwähnt, wie der Graph einer analytischen Funktion und die zugehörigen Ableitungsfunktionen dargestellt werden können.

3.1 Modifiziertes Hornerschema unter Einbeziehung der Linearfaktorprodukte des Newtonverfahrens

Um mit Hilfe des Computers Ableitungsfunktionen berechnen zu können, muß man zunächst auf die Linearfaktorprodukte des Newtonverfahrens zurückgreifen. Sobald diese ermittelt worden sind, kann man mit Hilfe eines modifizierten Hornerschemas die gewünschten Werte berechnen. Zunächst soll anhand eines Beispiels die Vorgehensweise veranschaulicht werden.

Beispiel:

Gegeben seien die Stützstellen

x	1	2	3	4
y	3	1	7	-9

Zu bestimmen sind der Funktionswert sowie sämtliche Ableitungsfunktionswerte an der Stelle $x = 2{,}5$ des zugehörigen Interpolationspolynoms!

Lösung:

a) Linearfaktorprodukte des Newtonverfahrens: [2]

Nach dem Newtonverfahren erhält man für die oben aufgeführten Stützstellen folgende Koeffizienten der Newtonschen Linearfaktorprodukte:

[2] Da das Newtonverfahren bereits ausführlich behandelt wurde, sei im folgenden lediglich die Lösung der gesuchten Linearfaktorprodukte zu den obigen Stützstellen wiedergegeben.

$$
\begin{aligned}
p_3 &= -5 \\
p_2 &= 4 \\
p_1 &= -2 \\
p_0 &= 3
\end{aligned}
$$

Daraus ergibt sich das folgende Newtonsche Interpolationspolynom:

$$y = 3 - 2(x-1) + 4(x-1)(x-2) - 5(x-1)(x-2)(x-3)$$

b) modifiziertes Hornerschema:

Zum besseren Verständnis sei zunächst das gesamte Rechenschema dargestellt. Anschließend wird das Verfahren detailliert erläutert.

+	-5 0	4 (2.5-3)*(-5)	-2 (2.5-2)*6.5	3 (2.5-1)*1.25
x=2.5 +	-5 0	6.5 (2.5-2)*(-5)	1.25 (2.5-1)*4	$4.875 = \frac{f(2.5)}{0!}$
x=2.5 +	-5 0	4 (2.5-1)*(-5)	$7.25 = \frac{f'(2.5)}{1!}$	
x=2.5 +	-5 0	$-3.5 = \frac{f''(2.5)}{2!}$		
x=2.5	$-5 = \frac{f'''(2.5)}{3!}$			

Demnach lauten die gesuchten Werte für $x = 2,5$:

$$
\begin{aligned}
f(2,5) &= 4,875 * 0! = 4,875 * 1 = 4,875 \\
f'(2,5) &= 7,25 * 1! = 7,25 * 1 = 7,25 \\
f''(2,5) &= -3,5 * 2! = -3,5 * 2 = -7 \\
f'''(2,5) &= -5 * 3! = -5 * 6 = -30
\end{aligned}
$$

Erläuterung des Verfahrens:

In die erste Zeile werden geordnet, beginnend mit dem Koeffizienten des längsten Linearfaktorprodukts, die Linearfaktorprodukte des Newtonverfahrens eingetragen. Nun berechnet man nach dem schon bekannten Verfahren für das Hornerschema den Funktionswert $f(2,5)$. Um den Funktionswert der 1. Ableitung zu erhalten, rechnet man vom Ansatz her

mit demselben Algorithmus. Es wird jedoch auf die zuvor berechneten Zwischenergebnisse des Funktionswertes zurückgegriffen. Die folgenden Ableitungen werden analog zu diesem Schema berechnet. Am Ende sind die berechneten Werte mit der jeweils zugehörigen Fakultät zu multiplizieren. Mit Hilfe des modifizierten Hornerschemas lassen sich somit schnell die gewünschen Ableitungsfunktionswerte berechnen. Es soll nun anhand eines Beispiels überprüft werden, inwieweit die näherungsweise bestimmten Funktionswerte von den exakt berechneten abweichen.

Gegeben seien die Stützstellen

x	-1	0	3	4
y	2	-2	3	1

Daraus ergeben sich nach dem Newtonverfahren folgende Koeffizienten der Newtonschen Linearfaktorprodukte:

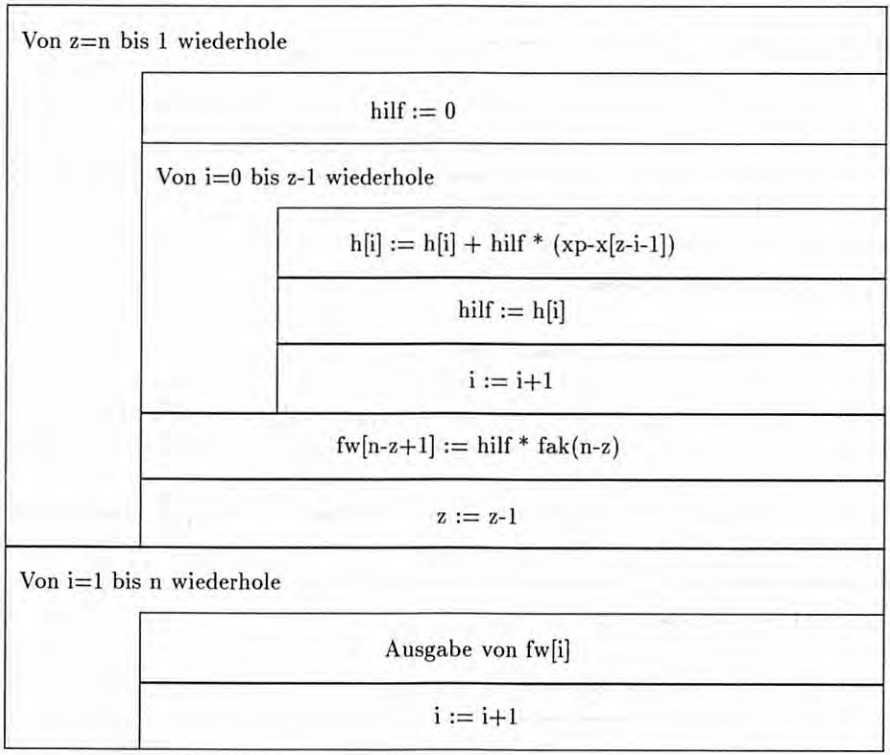
$$
\begin{aligned}
p_3 &= 0,358 \\
p_2 &= 1,24 \\
p_1 &= -4 \\
p_0 &= 2
\end{aligned}
$$

Berechnet man die Ableitungsfunktionswerte an der Stelle $x = 2$ mit Hilfe des modifizierten Hornerschemas, so erhält man:

	0.358	1.24	-4	2
+	0	(2-3)*0.358	(2-0)*1.062	(2+1)*(-1.876)
x=2	0.358	1.062	-1.876	$-3.628 = \frac{f(2)}{0!}$
+	0	(2-0)*0.36	(2+1)*1.78	
x=2	0.36	1.78	$3.46 = \frac{f'(2)}{1!}$	
+	0	(2+1)*0.36		
x=2	0.36	$2.86 = \frac{f''(2)}{2!}$		
+	0			
x=2	$0.36 = \frac{f'''(2)}{3!}$			

$f(2)$	=	-3.628	*0!	=	-3.628	*1	=	-3.628
$f'(2)$	=	3.46	*1!	=	3.46	*1	=	3.46
$f''(2)$	=	2.86	*2!	=	2.86	*2	=	5.72
$f'''(2)$	=	0.36	*3!	=	0.36	*6	=	2.16

```
Eingabe der Anzahl der Stützstellen
Von i=0 bis n-1 wiederhole
    Eingabe von x[i] und y[i]
    i := i+1
ya[0,1] := y[0]
Von i=1 bis n-1 wiederhole
    ya[1,i] := (y[i]-y[i-1]) / (x[i]-x[i-1])
    i := i+1
Von z=2 bis n-1 wiederhole
    Von i=1 bis n-z wiederhole
        ya[z,i] := (ya[z-1,i+1]-ya[z-1,i]) / (x[z-1+i]-x[i-1])
        i := i+1
    z := z+1
Von i=1 bis n+1 wiederhole
    h[n-i] := ya[i-1,1]
    i := i+1
Eingabe von xp
```

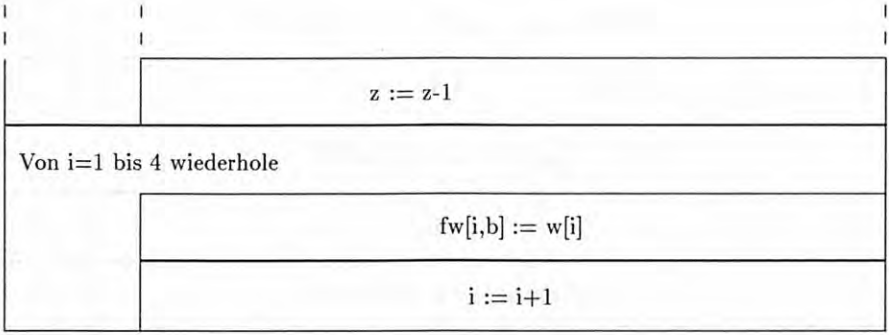


Variablen:

ya	:	Zweidimensionaler ARRAY vom Datentyp Real
x, y, h, fw, a	:	Eindimensionaler ARRAY von Datentyp Real
i, n ,z	:	Variablen vom Datentyp Integer
xp, hilf	:	Variablen vom Datentyp Real

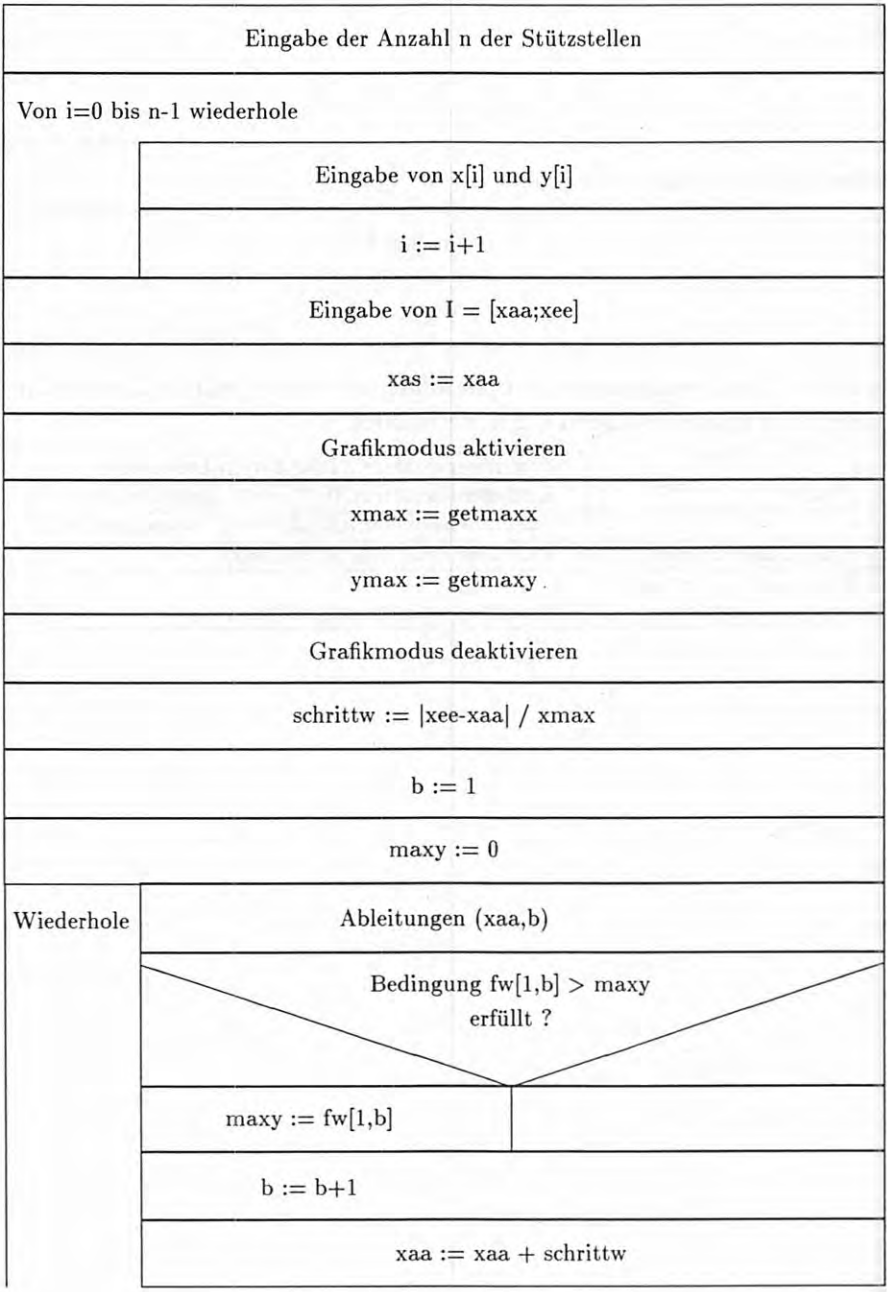
Soll ein Algorithmus erstellt werden, der es ermöglicht nach Eingabe von Stützstellen und einem Intervall den Graphen der Stammfunktion, sowie der 1. und 2. Ableitung darzustellen, kann man dafür folgendes Struktogramm verwenden. (Die „function" zur Berechnung der Fakultät ist hierbei ebenfalls einzubinden). Der gesamte Algorithmus besteht aus einer „Procedure" und einem Hauptprogramm. Zunächst wird das Struktogramm für die Procedure mit der Vereinbarung PROCEDURE Ableitungen (xa : Real ; b : Integer) ; angegeben.

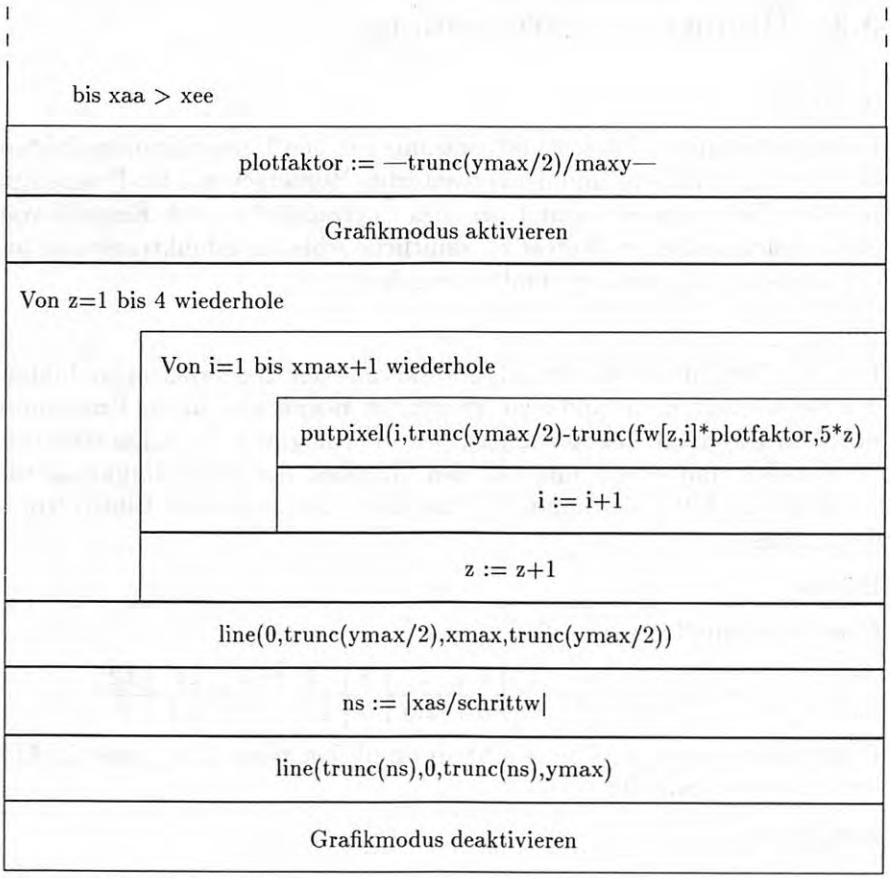
```
ya[0,1] := y[0]
Von i=1 bis n-1 wiederhole
    ya[1,i] := (y[i]-y[i-1]) / (x[i]-x[i-1])
    i := i+1
Von z=2 bis n-1 wiederhole
    Von i=1 bis n-z wiederhole
        ya[z,i] := (ya[z-1,i+1]-ya[z-1,i])/(x[z-1+i]-x[i-1])
        i := i+1
    z := z+1
Von i=1 bis n+1 wiederhole
    h[n-i] := ya[i-1,1]
    i := i+1
Von z=n bis 1 wiederhole
    hilf := 0
    Von i=0 bis z-1 wiederhole
        h[i] := h[i]+hilf*(xa-x[z-i-1])
        hilf := h[i]
        i := i+1
    w[n-z+1] := hilf * fak(n-z)
```



Für das Hauptprogramm kann man folgendes Struktogramm verwenden. Folgende Variablen wurden dafür verwendet:

ya	:	Zweidimensionaler ARRAY vom Datentyp Real
x,y,h,a,w	:	Eindimensionaler ARRAY vom Datentyp Real
fw	:	Zweidimensionaler ARRAY vom Datentyp Real
i, n, z, b, xmax, ymax	:	Variablen vom Datentyp Integer
hilf, xaa, xee, schrittw, maxy, xas, plotfaktor, ns	:	Variablen vom Datentyp Real


Eingabe der Anzahl n der Stützstellen
Von i=0 bis n-1 wiederhole
Eingabe von x[i] und y[i]
i := i+1
Eingabe von I = [xaa;xee]
xas := xaa
Grafikmodus aktivieren
xmax := getmaxx
ymax := getmaxy
Grafikmodus deaktivieren
schrittw := |xee-xaa| / xmax
b := 1
maxy := 0
Wiederhole
Ableitungen (xaa,b)
Bedingung fw[1,b] > maxy
erfüllt ?
maxy := fw[1,b]
b := b+1
xaa := xaa + schrittw


bis xaa > xee
plotfaktor := —trunc(ymax/2)/maxy—
Grafikmodus aktivieren
Von z=1 bis 4 wiederhole
Von i=1 bis xmax+1 wiederhole
putpixel(i,trunc(ymax/2)-trunc(fw[z,i]*plotfaktor,5*z)
i := i+1
z := z+1
line(0,trunc(ymax/2),xmax,trunc(ymax/2))
ns := |xas/schrittw|
line(trunc(ns),0,trunc(ns),ymax)
Grafikmodus deaktivieren

3.2 Übungen zur Vertiefung

Aufgabe 1:

Erstellen Sie anhand des Struktogramms aus den Linearfaktorprodukten des Newtonverfahrens und dem erweiterten Hornerschema ein Programm mit dem Dateinamen ableit_1.pas, das es ermöglicht, nach Eingabe von Stützstellen und eines Wertes x_p, sämtliche Ableitungsfunktionswerte an der Stelle x_p zu berechnen und auszugeben.

Aufgabe 2:

Erstellen Sie anhand des Struktogramms aus den Linearfaktorprodukten des Newtonverfahrens und dem erweiterten Hornerschema ein Programm mit dem Dateinamen ableit_2.pas, das es ermöglicht, nach Eingabe von Stützstellen und einem Intervall den Graphen der Stammfunktion, sowie der 1. und 2. Ableitung bei maximaler Ausnutzung des Bildschirmes darzustellen.

Beispiel:

Gegeben seien die Stützstellen:

x	0	1	2	3	4	5	6	7	8	9	10	11	12
y	0	8,2	0	-5,5	0	3,7	0	-2,5	0	1,7	0	-1,1	0

Ermitteln Sie die Graphen der Stammfunktion sowie der 1. und 2. Ableitung im Intervall [0; 12] !

Aufgabe 3:

Ermitteln Sie mit einem bereits bekannten Verfahren (Gauß, Lagrange oder Newton) die Koeffizienten des Interpolationspolynoms zu den Stützstellen von Aufgabe 2.

Aufgabe 4:

Erstellen Sie mit Hilfe der Standardfunktionen EXP(x) und SIN(x) ein Turbo-Pascal-Programm mit dem Dateinamen ged_schw.pas, das nach Eingabe eines Intervalls den Graphen der Funktion:

$$f:\ y = 10 * e^{-\frac{x}{5}} * sin\frac{\pi}{2} * x$$

bei maximaler Ausnutzung des Bildschirmes darstellt. Vergleichen Sie den Graphen im Intervall I = [0; 12] mit der Stammfunktion des Beispiels von Aufgabe 2.

Aufgabe 5:

Erstellen Sie ein Programm mit dem Dateinamen abl_allg.pas, das es ermöglicht, nach Eingabe einer analytischen Funktion in den Programmquelltext sowie nach Eingabe eines Intervalls den Graphen der Stammfunktion sowie die der 1. und 2. Ableitung bei maximaler Ausnutzung des Bildschirmes darzustellen.

Anleitung:

- Generieren Sie Stützstellen aus der im Quelltext stehenden Funktion
- Lösen Sie den Rest der Aufgabe wie bei Aufgabe 2.

Beispiel:

Gegeben Sei die Funktion:

$$f: \; y = e^x * sinx$$

und das Intervall I = $[-\pi;\ \pi]$. Darzustellen sind die Graphen der angegebenen Funktion sowie der 1. und 2. Ableitung bei optimaler Ausnutzung des Bildschirmes.

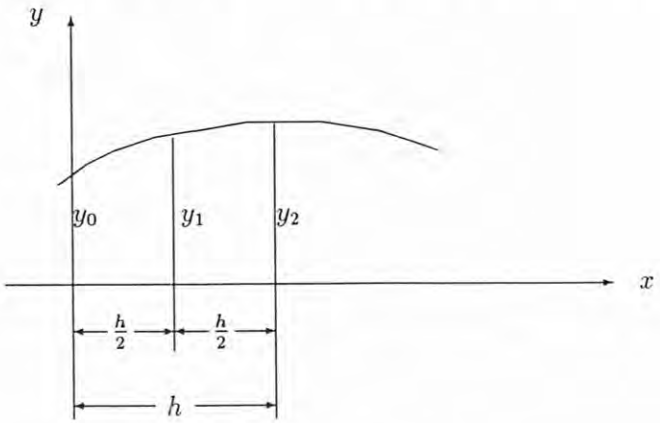
4. Numerisches Integrieren

In diesem Kapitel soll erläutert werden, wie mit Hilfe der numerischen Mathematik bestimmte Integrale gelöst werden können. Für die näherungsweise Berechnung eines bestimmten Integrals $A = \int_a^b f(x)dx$ werden häufig folgende Verfahren verwendet:

- Sehnen-Trapez-Verfahren $A \approx \frac{f}{2}[(y_0 + y_n) + 2(y_1 + y_2 + \ldots + y_{n-1})]$
- Tangenten-Trapez-Verfahren $A \approx 2k[y_1 + y_3 + \ldots + y_{n-1}]$
- Simpson-Verfahren. Da es sich hierbei um ein gängiges Verfahren handelt, soll es im Folgenden näher erläutert werden. Ferner sei noch angemerkt, daß n die geradzahlige Anzahl von Teilintervallen der Länge $k = \frac{b-a}{n}$ bedeutet.

4.1 Das Simpsonverfahren bei empirisch ermittelten und analytischen Funktionen

Dabei werden beispielsweise 3 benachbarte äquidistante Stützstellen [4] innerhalb eines Intervalls von der Breite h durch einen Parabelbogen miteinander verbunden:



[4] Stützstellen mit gleichem Abstand zueinander

Zunächst werden die 3 Stützstellen in einer Tabelle dargestellt:

x	0	$\frac{h}{2}$	h
y	y_0	y_1	y_2

Danach wendet man die Parabelgleichung an, um auf die Koeffizienten der oben dargestellten Funktion zu kommen:

$$f(x) = a_2x^2 + a_1x + a_0 .$$

Die Stützstellen in die Parabelgleichung eingesetzt ergibt folgendes Gleichungssystem:

$$\begin{array}{lll} (1) & y_2 = h^2a_2 + ha_1 + a_0 \\ (2) & y_1 = \frac{h^2}{4}a_2 + \frac{h}{2}a_1 + a_0 \\ (3) & y_0 = a_0 \end{array}$$

Um die Koeffizienten zu erhalten, multipliziert man Gleichung (2) mit -4 und addiert sie zu Gleichung (3):

$$\begin{array}{lrcl} (3) & y_2 & = h^2a_2 + ha_1 + a_0 \\ (2) & -4y_1 & = -h^2a_2 - 2ha_1 - 4a_0 \\ \hline (2)+(3): & y_2 - 4y_1 & = -ha_1 - 3a_0 \end{array}$$

Danach kann man die Gleichung nach a_1 umformen:

$$a_1 = -\frac{3a_0 + y_2 - 4y_1}{h} .$$

Außerdem ergibt sich aus Gleichung (1):

$$a_0 = y_0 .$$

Durch Umformen erhält man schließlich:

$$a_1 = \frac{1}{h}(-3y_0 + 4y_1 - y_2) .$$

Setzt man die berechnete Formel von a_1 in Gleichung (3) ein, so erhält man folgenden Ausdruck:

$$y_2 = h^2a_2 + h\left[\frac{1}{h}(-3y_0 + 4y_1 - y_2)\right] + y_0 .$$

Dann wird dieser Ausdruck nach a_2 umgeformt:

$$\begin{aligned} 0 &= h^2 a_2 - 3y_0 + 4y_1 - 2y_2 + y_0 \\ h^2 a_2 &= 3y_0 - 4y_1 + 2y_2 - y_0 \\ h^2 a_2 &= 2y_0 - 4y_1 + 2y_2 = 2(y_0 - 2y_1 + y_2) \\ a_2 &= \frac{2}{h^2}(y_0 - 2y_1 + y_2) \\ y_2 &= h^2 a_2 - 3y_0 + 4y_1 - y_2 + y_0 \end{aligned}$$

Die Fläche des obigen Intervalls läßt sich berechnen durch:

$$A_h = \int_0^h f(x)dx \ \wedge \ f(x) = a_2x^2 + a_1x + a_0$$

$$\Longrightarrow A_h = \int_0^h (a_2x^2 + a_1x + a_0)\,dx = \left[\frac{a_2}{3}x^3 + \frac{a_1}{2}x^2 + a_0x\right]_0^h$$

$$A_h = \frac{a_2}{3}h^3 + \frac{a_1}{2}h^2 + a_0h\ .$$

Setzt man nun die zuvor berechneten Koeffizienten ein, so erhält man folgenden Ausdruck:

$$A_h = \frac{h^3}{3}\left[\frac{2}{h^2}(y_0 - 2y_1 + y_2)\right] + \frac{h^2}{2}\left[\frac{1}{h}(-3y_0 + 4y_1 - y_2)\right] + y_0 h \,.$$

Durch Vereinfachen ergibt sich:

$$A_h = \frac{2}{3}h(y_0 - 2y_1 + y_2) + \frac{1}{2}h(-3y_0 + 4y_1 - y_2) + y_0 h \,.$$

Zwecks weiterer Vereinfachung teilt man durch h. Die Bedingung h ungleich 0 ist dadurch erfüllt, indem das Intervall immer größer sein muß als Null.

$$\frac{A_h}{h} = \frac{2}{3}(y_0 - 2y_1 + y_2) + \frac{1}{2}(-3y_0 + 4y_1 - y_2) + y_0 \,.$$

Nach dem Ausmultiplizieren erhält man:

$$\frac{A_h}{h} = \frac{2}{3}y_0 - \frac{4}{3}y_1 + \frac{2}{3}y_2 - \frac{3}{2}y_0 + 2y_1 - \frac{1}{2}y_2 + y_0 \,.$$

Um die Formel zusammenfassen zu können, müssen die Brüche erweitert werden:

$$\frac{A_h}{h} = \frac{4}{6}y_0 - \frac{8}{6}y_1 + \frac{4}{6}y_2 - \frac{9}{6}y_0 + \frac{12}{6}y_1 - \frac{3}{6}y_2 + \frac{6}{6}y_0 \,.$$

Durch weitere Vereinfachung erhält man schließlich den Ausdruck für die gesuchte Fläche im Intervall von 0 bis h:

$$\frac{A_h}{h} = \frac{1}{6}y_0 + \frac{4}{6}y_1 + \frac{1}{6}y_2$$

$$A_h = \frac{h}{6}(y_0 + 4y_1 + y_2) \,.$$

Überträgt man das oben aufgeführte Beispiel mit 3 äquidistanten Stützstellen auf beliebig viele Stützstellen, so erhält man die Simpsonsche Regel. Soll eine Fläche im Intervall $[a;\ b]$ berechnet werden, dann teilt man

zunächst nach dem Simpsonverfahren dieses Intervall in eine gerade Anzahl n von Teilintervallen auf. Die Breite k eines Teilintervalls berechnet sich dann nach:

$$k = \frac{b-a}{n} .$$

k : Breite des Teilintervalls

b : obere Intervallgrenze

a : untere Intervallgrenze

n : gerade Anzahl von Teilintervallen;

bei gerader Anzahl n von Stützstellen ist $n = 2i$ mit $i \in \mathcal{N}$;

bei ungerader Anzahl von Stützstellen ist $m = 2i - 1$ mit $i \in \mathcal{N}$

Dabei ist die letzte Stützstelle eines Teilintervalls k zugleich auch die erste des nachfolgenden Teilintervalls.

Für die Fläche

$$A = \int_a^b f(x)dx$$

erhält man angenähert nach der Simpsonschen Regel:

$$A = \frac{k}{3}[(y_0 + y_n) + 2(y_2 + y_4 + \ldots + y_{n-2}) + 4(y_1 + y_3 + \ldots + y_{n-1})]$$

mit $y_0 = y_a$ und $y_n = y_b$.

Der bei der numerischen Integration nach dem Simpsonverfahren entstehende Fehler läßt sich dadurch verringern, indem man folgendermaßen verfährt:

Man berechnet das Integral einmal mit der Schrittweite h und ein weiteres Mal mit der Schrittweite $\frac{h}{2}$. Danach ergeben sich die entsprechenden Näherungswerte des Integrals A_1 und A_2. Den verbesserten Integralwert (A_3) erhält man mit der Summe $A_3 = A_2 + \frac{A_2 - A_1}{15}$.

Beispiel:

Gegeben seien folgende äquidistante Stützstellen:

x	3	4	5	6	7	8	9
y	6	5	4	4,5	5	3,5	2

Zu berechnen ist nach dem Simpsonverfahren

$$A = \int_3^9 f(x)dx \ .$$

Lösung:

$$k = \frac{9-3}{6} = 1$$

$$A = \frac{1}{3}[(6+2) + 2(4+5) + 4(5+4,5+3,5)] \ = \ 26$$

Sollte anstelle von Stützstellen ein Funktionsterm gegeben sein, dann lassen sich mit diesem Term Stützstellen berechnen. Zu beachten ist dabei nur, daß die Stützstellen äquidistant sein müssen.

Beispiel:

Gegeben sei die Funktion

$$f(x) = x^2 \ .$$

Zu berechnen ist nach dem Simpsonverfahren das Integral

$$A = \int_0^3 f(x)dx \ .$$

Lösung:

(1) Generieren von äquidistanten Stützstellen:

Zum Erzeugen der Stützstellen ist das vorgegebene Integrationsintervall von Bedeutung. In diesem Beispiel ist I= [0; 3]. Dies sind gleichzeitig auch die Anfangs- bzw. Endstützstellen. Die Schrittweite Δx kann frei gewählt werden. Bei diesem Beispiel sei $\Delta x = 0,5$. Somit ergibt sich nach Einsezten in den obigen Funktionsterm folgende Wertetabelle:

x	0	0,5	1	1,5	2	2,5	3
y	0	0,25	1	2,25	4	6,25	9

(2) Anwendung des Simpsonverfahrens:

Nachdem genügend viele Stützstellen vorliegen, kann man das Simpsonverfahren anwenden. Dazu gilt es zunächst, die Breite des Teilintervalls k auszurechnen:

$$k = \frac{3-0}{6} = \frac{1}{2}.$$

Nach dem Einsetzen in die Simpsonsche Regel erhält man

$$A = \frac{1}{6}[(0+9) + 2(1+4) + 4(0,25 + 2,25 + 6,25)].$$

Bei der Berechnung mittels Simpsonscher Regel ist in jedem Falle zu berücksichtigen:

- Die Stützstellen müssen äquidistant sein
- Die Abzissenwerte müssen ansteigend angeordnet sein
- Die Integrationsgrenzen werden durch erste und letzte Stützstelle bestimmt
- Die Werte für die Integrationsgrenzen müssen gleiches Vorzeichen besitzen
- Die Anzahl der bekannten Stützstellen beeinflußt das Ergebnis. Dieses ist um so genauer je mehr Stützstellen bekannt sind.

Zur Kontrolle des angewandten Verfahrens soll nun das Beispiel algebraisch gelöst werden:

$$A = \int_0^3 x^2 = \left[\frac{1}{3}x^3\right]_0^3 = \frac{1}{3}3^3 - \frac{1}{3}0^3 = 9.$$

Das soeben am Beispiel erläuterte Simpson-Verfahren wird im Folgenden in einem Struktogramm dargestellt:

```
Ungerade Anzahl der Stützstellen i eingeben
Von n=0 bis i-1 wiederhole
    Eingabe von x[n] und y[n]
    n := n+1
Eingabe der Grenzen a und b
s1 := 0
s2 := 0
k := i
h := (b-a) / (i-1)
m := 0
s1 := y[0] + y[k-1]
Von m=2 bis k-2 wiederhole
    s2 := s2+y[m]
    m := m+2
s1 := s1+2*s2
s2 := 0
Von m=1 bis k-1 wiederhole
    s2 := s2+y[m]
    m := m+2
```

s1 := s1+4*s2
flaeche := s1*h/3
Ausgabe von flaeche

Variablen:

a, b, h, s1, s2, fläche	:	Variablen vom Datentyp Real
i, k, m, n	:	Variablen vom Datentyp Integer
y, x	:	Eindimensionaler ARRAY vom Datentyp Real

4.2 Grafische Darstellung von Stammfunktionen mit Hilfe von Polynomen

Nachdem im Kapitel 4.1 mit Hilfe des Simpsonverfahrens bestimmte Integrale berechnet wurden, soll in diesem Kapitel erläutert werden, wie Stammfunktionen mit Hilfe des Rechners berechnet und grafisch dargestellt werden können.

Die Lösung dieses Problems gelingt dadurch, daß durch Stützstellen Interpolationspolynome entwickelt und grafisch dargestellt werden können. Bei den Interpolationspolynomen handelt es sich immer um ganzrationale Funktionen, die mit Hilfe der Potenzregel leicht integriert werden können:

Beispiel:

Gegeben seien folgende Stützstellen:

x	1	3	6	4
y	1	9	36	16

Ermittelt man hierfür das zugehörige Interpolationspolynom, so erhält man folgende Funktionsgleichung:

$$f(x) = x^2 .$$

Nach der Potenzregel der Integralrechnung gilt:

$$\int x^n dx = \frac{x^{n+1}}{n+1} + c ,$$

wobei c die Integrationskonstante ist.

Für

$$\int f(x) dx$$

erhält man danach

$$\int x^2 dx = \frac{1}{3} x^3 + c .$$

Somit ist die Stammfunktion wiederum ein Polynom. Dieses kann bei Eingabe einer Integrationskonstanten auch graphisch dargestellt werden.

Allgemein gilt:

- Soll eine Stammfunktion grafisch auf dem Bildschirm dargestellt werden, so müssen die Stützstellen, das Intervall in dem dargestellt werden soll und die Integrationskonstante bekannt sein.
- Man berechnet mit einem geeigneten Verfahren das Interpolationspolynom zu den gegebenen Stützstellen.
- Man integriert das berechnete Interpolationspolynom mit Hilfe der Potenzregel der Integralrechnung
- Die entstandene Polynomfunktion läßt sich in einfacher Weise grafisch auf dem Bildschirm darstellen, da sie sich im Prinzip nicht von einem Interpolationspolynom unterscheidet.

Dieses Verfahren gilt für Funktionen allgemein, indem mit Hilfe einer Schleife Stützstellen erzeugt werden. Diese können dann nach dem obigen Verfahren weiter verarbeitet werden. Somit lassen sich Stammfunktionen annähernd berechnen und grafisch darstellen. Im Folgenden wird der hierbei zusätzlich für die Integration mittels Potenzregel erforderliche Algorithmus im Struktogramm wiedergegeben. Vorraussetzung ist, daß die Koeffizienten ermittelt und in einem ARRAY mit dem Namen Koef gespeichert sind.

Eingabe der Integrationskonst. intconst	
Von i=ns-1 bis 1 wiederhole	
	Koef[i+1] := koef[i] / (i+1)
	i := i+1
koef[1] := koef[0]	
koef[0] := intconst	

Variablen:

intconst	:	Variable vom Datentyp Real
i	:	Variable vom Datentyp Integer
koef	:	Eindimensionaler ARRAY vom Datentyp Real
ns	:	Variable vom Datentyp Integer, die die Anzahl der Stützstellen angibt

Nach dem Algorithmus sind die integrierten Koeffizienten ähnlich wie beim Anfang in dem ARRAY koef enthalten und können mit Hilfe einer Schleife ausgegeben bzw. weiterbearbeitet werden.

4.3 Übungen zur Vertiefung

Aufgabe 1:

Erstellen Sie anhand des entsprechenden Struktogramms ein Programm, mit dem Dateinamen integr_1.pas, das nach der Simpsonmethode anhand von Stützstellen und Integrationsgrenzen die Fläche ermittelt.

Beispiel:

Gegeben seien die Stützstellen:

x	1	3	5	7	9
y	2	1	9	4	1

Zu ermitteln ist:

$$\int_1^9 y dx$$

Aufgabe 2:

Erstellen Sie ein Programm mit dem Dateinamen integr_2.pas, das annähernd das Integral

$$a = \int_0^\infty \frac{sin\ x}{x} dx$$

berechnet, wobei zunächst die Stützstellen generiert werden müssen. Ersetzen Sie unendlich durch 100 und erhöhen Sie auf 1000 usw.

Die exakte Lösung ergibt:

$$a = \int_0^\infty \frac{sin\ x}{x} dx = \frac{\pi}{2}$$

Aufgabe 3:

Erstellen Sie unter Zuhilfenahme des zuletzt wiedergegebenen Struktogramms ein Programm mit dem Dateinamen integr_3.pas, das nach Eingabe von Stützstellen, eines Intervalls und der Integrationskonstanten den Graphen der Stammfunktion bei optimaler Ausnutzung des Bildschirmes darstellt.

Beispiel:

Gegeben seien die Stützstellen:

x	1	4	7	5
y	1	16	49	25

die Integrationskonstante $k = 1$, und das Intervall $I = [-5;\ 5]$.

Darzustellen ist die Stammfunktion des Interpolationspolynoms $F(x)$ bei optimaler Ausnutzung des Bildschirmes im angegebenen Intervall.

Aufgabe 4:

Erstellen Sie ein Programm mit dem Dateinamen integr_4.pas, das nach Eingabe einer analytischen Funktion in den Quelltext, eines Intervalls und der Integrationskonstanten den Graphen der Stammfunktion bei optimaler Ausnutzung des Bildschirmes darstellt.

Beispiel:

Gegeben sei die analytische Funktion

$$y = 2x^3 - 2x + 5$$

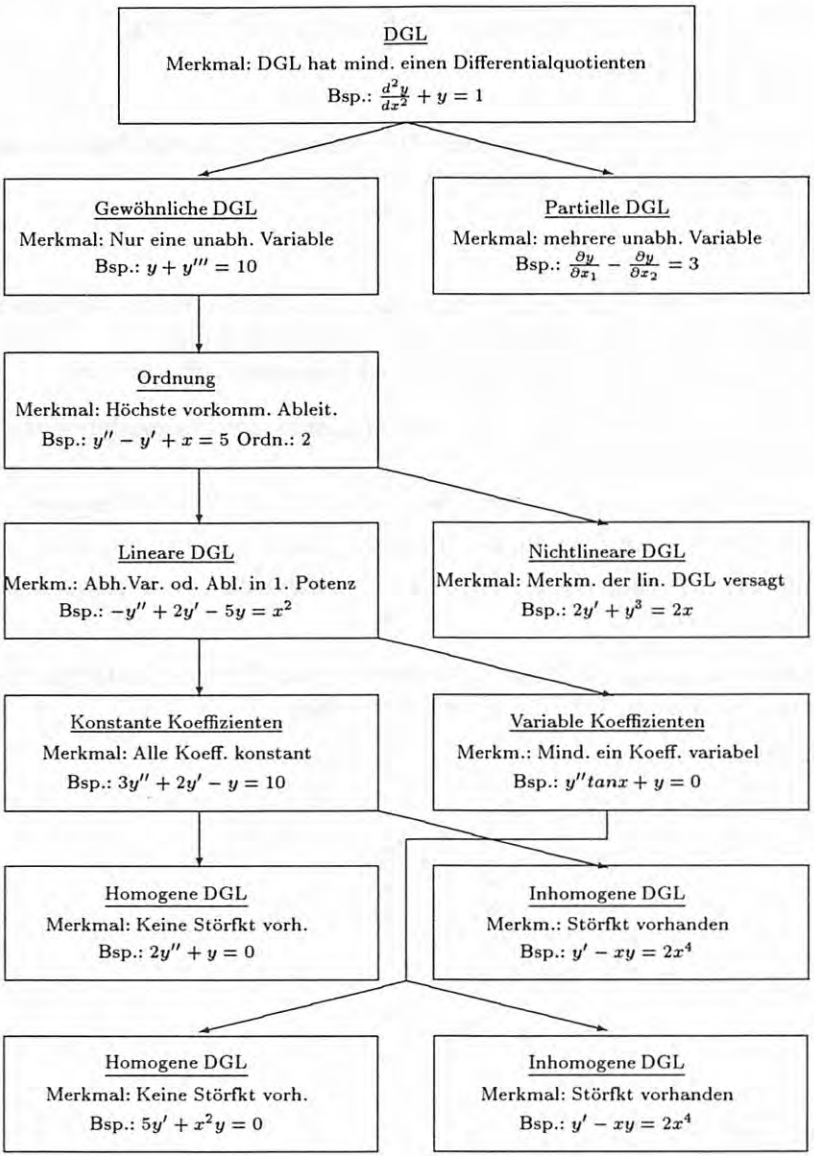
die Integrationskonstante $k = -50$ und das Intervall $I = [-5;\ 5]$.

Darzustellen ist die Stammfunktion des Interpolationspolynoms $F(x)$ bei optimaler Ausnutzung des Bildschirmes im angegebenen Intervall.

5. Differentialgleichungen

In diesem Kapitel sollen mit Hilfe eines numerischen Verfahrens Differentialgleichungen 1. und 2. Ordnung gelöst werden. Durch Anwendung eines Rechners können die Lösungskurven der Differentialgleichungen grafisch dargestellt werden.

Bevor jedoch das Verfahren beschrieben wird, sei zunächst ein Überblick über die gewöhnlichen Differentialgleichungen bzw. eine Klassifizierung derselben dargestellt:


DGL
Merkmal: DGL hat mind. einen Differentialquotienten
Bsp.: $\frac{d^2y}{dx^2} + y = 1$
Gewöhnliche DGL
Merkmal: Nur eine unabh. Variable
Bsp.: $y + y''' = 10$
Partielle DGL
Merkmal: mehrere unabh. Variable
Bsp.: $\frac{\partial y}{\partial x_1} - \frac{\partial y}{\partial x_2} = 3$
Ordnung
Merkmal: Höchste vorkomm. Ableit.
Bsp.: $y'' - y' + x = 5$ Ordn.: 2
Lineare DGL
Merkm.: Abh.Var. od. Abl. in 1. Potenz
Bsp.: $-y'' + 2y' - 5y = x^2$
Nichtlineare DGL
Merkmal: Merkm. der lin. DGL versagt
Bsp.: $2y' + y^3 = 2x$
Konstante Koeffizienten
Merkmal: Alle Koeff. konstant
Bsp.: $3y'' + 2y' - y = 10$
Variable Koeffizienten
Merkm.: Mind. ein Koeff. variabel
Bsp.: $y''tanx + y = 0$
Homogene DGL
Merkmal: Keine Störfkt vorh.
Bsp.: $2y'' + y = 0$
Inhomogene DGL
Merkm.: Störfkt vorhanden
Bsp.: $y' - xy = 2x^4$
Homogene DGL
Merkmal: Keine Störfkt vorh.
Bsp.: $5y' + x^2y = 0$
Inhomogene DGL
Merkmal: Störfkt vorhanden
Bsp.: $y' - xy = 2x^4$

5.1 Das Runge-Kutta-Verfahren für Dgln. 1. Ordnung

In diesem Kapitel wird die grafische Darstellung der Lösungskurven von Differentialgleichungen 1. Ordnung vom Typ

$$y'(x) = f(x; y)$$

erläutert. Für die numerische Lösung kann das Runge Kutta Verfahren verwendet werden, mit dessen Hilfe man Punkt für Punkt die Lösungskurve ausrechnen kann. Dafür muß jedoch folgendes bekannt sein:

- die Differentialgleichung von der der Graph der Lösungskurve dargestellt werden soll
- ein Punkt $P_0 = (x_0; y_0)$ der Lösungskurve (z.B. Randbedingung)
- das Intervall, in dem die Lösungskurve dargestellt werden soll
- die Schrittweite, die den Abstand zwischen den zu berechnenden Punkten angibt.

Zunächst einmal wird das Lösungsverfahren nach Runge- Kutta für Differentialgleichungen. 1. Ordnung wiedergegeben.

1. Durchlauf:

$$\begin{aligned} k_1 &= h * f(x_0; y_0) \\ k_2 &= h * f(x_0 + \frac{h}{2}; y_0 + \frac{k_1}{2}) \\ k_3 &= h * f(x_0 + \frac{h}{2}; y_0 + \frac{k_2}{2}) \\ k_4 &= h * f(x_0 + h; y_0 + k_3) \\ y(x_1) \approx y_1 &= y_0 + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \end{aligned}$$

2. Durchlauf:

$$\begin{aligned} k_1 &= h * f(x_1; y_1) \\ k_2 &= h * f(x_1 + \frac{h}{2}; y_1 + \frac{k_1}{2}) \\ k_3 &= h * f(x_1 + \frac{h}{2}; y_1 + \frac{k_2}{2}) \end{aligned}$$

$$
\begin{aligned}
k_4 &= h * f(x_1 + h; y_1 + k_3) \\
y(x_2) \approx y_2 &= y_1 + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \\
&\vdots
\end{aligned}
$$

Nach dieser Rechenvorschrift ist es nun möglich, Punkt für Punkt die Lösungskurve der Differentialgleichung zu ermitteln. Anfangspunkt ist der Punkt $P_0 = (x_0; y_0)$. Von hier aus wird im Abstand der Schrittweite Punkt für Punkt hochgerechnet. Soll der Algorithmus umgekehrt ablaufen, d.h soll vom Punkt P_0 aus heruntergerechnet werden, muß eine negative Schrittweite gewählt werden.

Beispiel für das Runge Kutta Verfahren für Dgl. 1. Ordnung:

Es soll nun anhand eines Beispiels die oben angegebene Rechenvorschrift angewendet werden. Dazu dient folgende Differentialgleichung:

$$y' = x^2 - 2y$$

mit der Schrittweite $h = 0,5$, dem Punkt $P_0 = (0; 1)$ und dem Intervall I= [0; 5].

Beginn der Berechnung:

$$
\begin{aligned}
k_1 &= h * f(x_0; y_0) \\
k_1 &= 0,5[0^2 - 2] = -1 \\
k_2 &= h * f(x_0 + \frac{h}{2}; y_0 + \frac{k_1}{2}) \\
k_2 &= 0,5[(0 + \frac{0,5}{2})^2 - 2(1 + \frac{-1}{2})] = -0,4688 \\
k_3 &= h * f(x_0 + \frac{h}{2}; y_0 + \frac{k_2}{2}) \\
k_3 &= 0,5[(0 + \frac{0,5}{2})^2 - 2(1 - \frac{0,4688}{2})] = -0,7344 \\
k_4 &= h * f(x_0 + h; y_0 + k_3) \\
k_4 &= 0,5[(0 + 0,5)^2 - 2(1 - 0,7344)] = -0,1406 \\
y(x_1) \approx y_1 &= y_0 + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \\
y_1 &= 1 + \frac{1}{6}(-1 - 2 * 0,4688 - 2 * 0,7344 - 0,1406)
\end{aligned}
$$

$$
\begin{aligned}
&= 0,4089 \\
\Longrightarrow P_1 &= (x_1; y_1) = (0,5; 0,4089)
\end{aligned}
$$

Der soeben berechnete Punkt P_1 nimmt nun den Stellenwert des obigen Punktes P_0 ein. Fortsetzung der Berechnung:

$$
\begin{aligned}
k_1 &= h * f(x_1; y_1) \\
k_1 &= 0,5[0,5^2 - 2 * 4089] = -0,2839 \\
k_2 &= h * f(x_1 + \frac{h}{2}; y_1 + \frac{k_1}{2}) \\
k_2 &= 0,5[(0,5 + \frac{0,5}{2})^2 - 2(0,4089 + \frac{-0,2839}{2})] \\
&= -0,0143 \\
k_3 &= h * f(x_1 + \frac{h}{2}; y_1 + \frac{k_2}{2}) \\
k_3 &= 0,5[(0,5 + \frac{0,5}{2})^2 - 2(0,4089 + \frac{0,0143}{2})] \\
&= -0,1348 \\
k_4 &= h * f(x_1 + h; y_1 + k_3) \\
k_4 &= 0,5[(0,5 + 0,5)^2 - 2(0,4089 - 0,1348)] = 0,2259 \\
y(x_2) \approx y_2 &= y_1 + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \\
y_2 &= 0,4089 + \frac{1}{6}(-0,2839 + 2 * 0,0143 - 2 * 0,1348 \\
&\quad +0,2259) = 0,35 \\
\Longrightarrow P_2 &= (x_2; y_2) = (1; 0,3590)
\end{aligned}
$$

Mit dem soeben berechneten Punkt P_2 rechnet man nach diesem Verfahren weiter und erhält noch folgende Punkte:

$P_3 = (1,5/0,6685)$
$P_4 = (2,0/1,2689)$
$P_5 = (2,5/2,1347)$
$P_6 = (3,0/3,2562)$
$P_7 = (3,5/4,6299)$
$P_8 = (4,0/6,2545)$
$P_9 = (4,5/8,1293)$
$P_{10} = (5,0/10,2542)$

Diese Punkte ergeben schließlich den Lösungsgraph. Das soeben am Beispiel erläuterte Runge-Kutta-Verfahren für Differentialgleichungen 1. Ordnung wird nun in Form eines Struktogramms dargestellt:

Verwendete Variablen:

x0,y0,xa,xe,h,x1,y1	:	Variablen vom Datentyp Real
i	:	Variable vom Datentyp Integer
x, y, k	:	Eindimensionaler ARRAY vom Datentyp Real

Am Ende des Algorithmus befinden sich die Ordinatenwerte der berechneten Punkte in dem ARRAY y.

```
Eingabe von x0
Eingabe von y0
Eingabe von xa
Eingabe von xe
h := |xe-xa| / Maximale Auflösung in x- Richtung
i := 1
x[i] := x0
x1 := x0
y[i] := y0
y1 := x0
Wiederhole
        k[1] := h*f(x1,y1)
        k[2] := h*f(x1+h/2,y1+k[1]/2)
        k[3] := h*f(x1+h/2,y1+k[2]/2)
        k[4] := h*f(x1+h,y1+k[3])
        i := i+1
        x[i] := x1+h
        y[i] := y1+ 1/6*(k[1]+2*k[2]+2*k[3]+k[4])
        x1 := x[i]
        y1 := y[i]
bis i> |xe-xa| / h
```

5.2 Das Runge-Kutta-Verfahren für Dgln. 2. Ordnung

Im Folgenden soll die grafische Darstellung der Lösungskurven von Differentialgleichungen 2. Ordnung vom Typ

$$y''(x) = f(x; y; y')$$

erörtert werden. Für die numerische Lösung kann ebenfalls das Runge-Kutta-Verfahren verwendet werden, mit dessen Hilfe man Punkt für Punkt die Lösungskurve berechnen kann. Dafür muß jedoch auch hier einiges bekannt sein:

- die Differentialgleichung von der der Graph der Lösungskurve dargestellt werden soll
- ein Punkt $P_0 = (x_0; y_0)$ der Lösungskurve (z.B. eine der Randbedingungen)
- ein weiteren Punkt $Q_0 = (x_0; y_0')$ der Lösungskurve (z.B. eine weitere Randbedingung)
- das Intervall, in dem die Lösungskurve dargestellt werden soll
- die Schrittweite, die den Abstand zwischen den zu berechnenden Punkten angibt.

Zunächst jedoch wird das Lösungsverfahren nach Runge- Kutta für Differentialgleichungen 2. Ordnung wiedergegeben.

1. Durchlauf:

$$\begin{aligned}
k_1 &= h * y_0' \\
m_1 &= h * f(x_0; y_0; y_0') \\
k_2 &= h(y_0' + \frac{m_1}{2}) \\
m_2 &= h * f(x_0 + \frac{h}{2}; y_0 + \frac{k_1}{2}; y_0' + \frac{m_1}{2}) \\
k_3 &= h(y_0' + \frac{m_2}{2})
\end{aligned}$$

$$
\begin{aligned}
m_3 &= h * f(x_0 + \frac{h}{2}; y_0 + \frac{k_2}{2}; y_0' + \frac{m_2}{2}) \\
k_4 &= h * (y_0' + m_3) \\
m_4 &= h * f(x_0 + h; y_0 + k_3; y_0' + m_3) \\
y_1 &= y_0 + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \\
y_1' &= y_0' + \frac{1}{6}(m_1 + 2m_2 + 2m_3 + m_4)
\end{aligned}
$$

2. Durchlauf:

$$
\begin{aligned}
k_1 &= h * y_1' \\
m_1 &= h * f(x_1; y_1; y_1') \\
k_2 &= h(y_1' + \frac{m_1}{2}) \\
m_2 &= h * f(x_1 + \frac{h}{2}; y_1 + \frac{k_1}{2}; y_1' + \frac{m_1}{2}) \\
k_3 &= h(y_1' + \frac{m_2}{2}) \\
m_3 &= h * f(x_1 + \frac{h}{2}; y_1 + \frac{k_2}{2}; y_1' + \frac{m_2}{2}) \\
k_4 &= h * (y_1' + m_3) \\
m_4 &= h * f(x_1 + h; y_1 + k_3; y_1' + m_3) \\
y_2 &= y_1 + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \\
y_2' &= y_1' + \frac{1}{6}(m_1 + 2m_2 + 2m_3 + m_4) \\
&\vdots
\end{aligned}
$$

Mit Hilfe dieser Rechenvorschrift kann man Punkt für Punkt die Lösungskurve der Differentialgleichung berechnen. Anfangspunkte sind die Punkte $P_0 = (x_0; y_0)$ und $Q_0 = (x_0; y_0')$. Hiervon ausgehend wird im Abstand der Schrittweite Punkt für Punkt hochgerechnet. Soll der Algorithmus

umgekehrt ablaufen, d.h soll vom Anfangspunkt aus heruntergerechnet werden, muß eine negative Schrittweite gewählt werden.

Beispiel für das Runge Kutta Verfahren für Dgl. 2. Ordnung:

Es soll nun anhand eines Beispiels die oben angegebene Rechenvorschrift angewendet werden. Dazu dient folgende Differentialgleichung:

$$y'' = \frac{1}{x} * y'$$

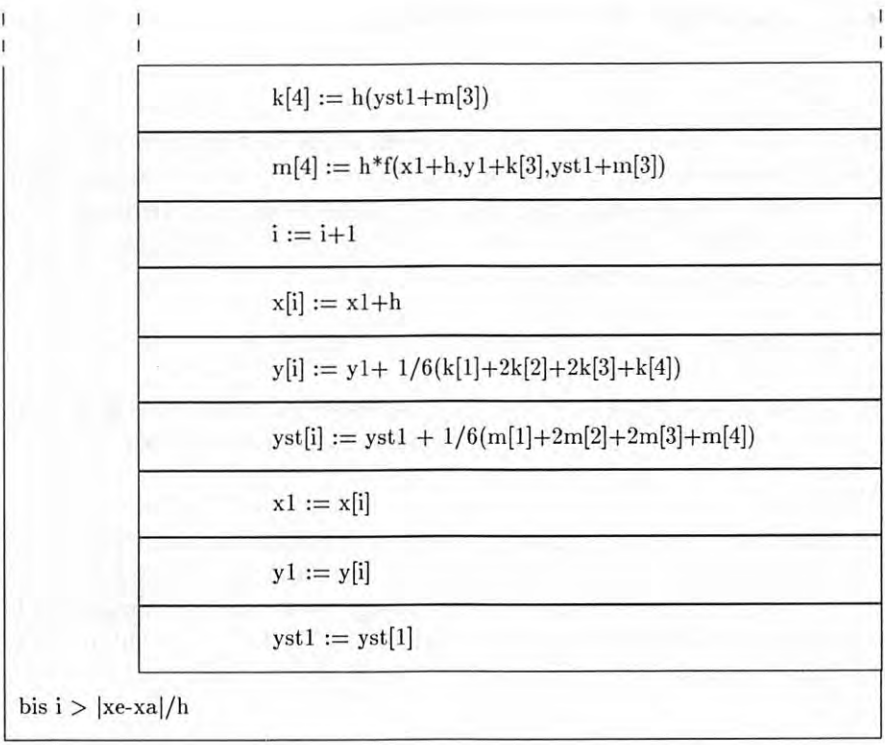
mit der Schrittweite $h = 0,2$; dem Punkt $P_0 = (1;1)$, dem Punkt $Q_0 = (1;1)$ und dem Intervall I = [1; 5].

Wenn man mit Hilfe des obigen Schemas rechnet, erhält man die folgenden Punkte:

$P_5 = (2,0/2,50)$
$P_6 = (2,2/2,92)$
$P_7 = (2,4/3,38)$
$P_8 = (2,6/3,88)$
$P_9 = (2,8/4,42)$
$P_{10} = (3,0/5,00)$
$P_{11} = (3,2/5,62)$
$P_{12} = (3,4/6,28)$
$P_{13} = (3,6/6,98)$
$P_{14} = (3,8/7,72)$
$P_{15} = (4,0/8,50)$
$P_{16} = (4,2/9,32)$
$P_{17} = (4,4/10,18)$
$P_{18} = (4,6/11,08)$
$P_{19} = (4,8/12,02)$
$P_{20} = (5,0/13,00)$

Diese errechneten Punkte ergeben schließlich näherungsweise den Lösungsgraph. Das soeben am Beispiel erläuterte Runge-Kutta-Verfahren für Differentialgleichungen 2. Ordnung wird nun anhand eines Struktogramms dargestellt:

	Eingabe von x0
	Eingabe von y0
	Eingabe von yst0
	Eingabe von xa
	Eingabe von xe
	h := |xe-xa| / Max. Auflösung in x- Richtg.
	i := 1
	x[i] := x0
	x1 := x0
	y[i] := y0
	y1 := y0
	yst[i] := yst0
	yst1 := yst0
Wiederhole	k[1] := h * yst1
	m[1] := h * f(x1,y1,yst1)
	k[2] := h * f(yst1+m[1]/2)
	m[2] := f*f(x1+h/2,y1+k[1]/2,yst1+m[1]/2)
	k[3] := f * f(ys+1+m[2]/2)
	m[3] := h*f(x1+h/2,y1+k[2]/2,yst1+m[2]/2)



Variablen

x0, y0, yst1, yst0, xa, xe, h, x1, y1	:	Variablen vom Datentyp Real
i	:	Variable vom Datentyp Integer
x, y, yst, k, m	:	Eindimensionaler ARRAY vom Datentyp Real

Am Ende des Algorithmus befinden sich die Ordinatenwerte der berechneten Punkte in dem ARRAY y.

5.3 Übungen zur Vertiefung

Aufgabe 1:

Erstellen Sie, ausgehend von dem vorgegebenen Struktogramm, ein Turbo-Pascal-Programm mit dem Dateinamen dgl_1ord.pas zur graphischen Darstellung von gewöhnlichen Differentialgleichungen 1. Ordung nach dem Runge-Kutta-Verfahren. Berücksichtigen Sie hierbei, daß die Dgl. in den Quelltext eingegeben werden muß. Z. B.

$$y' + 2xy = 0 \, .$$

Testen Sie dieses Programm mit der Randbedingung $y(-2) = 0,037$ im Intervall $I = [-2;\ 2]$ mit h bei maximaler Bildschirmauflösung.

Aufgabe 2:

Erstellen Sie, ausgehend von dem vorgegebenen Struktogramm, ein Turbo-Pascal-Programm mit dem Dateinamen dgl_2ord.pas zur graphischen Darstellung von gewöhnlichen Differentialgleichungen 2. Ordnung nach dem Runge-Kutta-Verfahren. Berücksichtigen Sie hierbei ebenfalls, daß die Dgl. in den Quelltext eingegeben werden muß. Z. B.

$$2y + 2y'' = x + 1 \, .$$

Testen Sie dieses Programm mit den Randbedingungen $y(0) = 1,5$ und $y'(0) = 0,5$ im Intervall I= $[0;\ 6,5]$ und mit h bei maximaler Bildschirmauflösung.

Aufgabe 3:

Erstellen Sie ein Turbo-Pascal-Programm mit dem Dateinamen dgl_gra1.pas, das es ermöglicht, die Funktion

$$y = 2e^{-x^2}$$

im Intervall I= $[-2;\ 2]$ unter optimaler Ausnutzung des Bildschirmes darzustellen. Vergleichen Sie dann den entstandenen Graphen mit dem von Aufgabe 1.

Aufgabe 4:

Erstellen Sie ein Turbo-Pascal-Programm mit dem Dateinamen dgl_gra2.pas, das es ermöglicht, die Funktion

$$y = \frac{1}{2} + \frac{1}{2}x + cosx$$

im Intervall $I = [0;\ 6,5]$ unter optimaler Ausnutzung des Bildschirmes darzustellen. Vergleichen Sie dann den entstandenen Graphen mit dem von Aufgabe 2.

6. Literaturhinweise

Ade, H.: Numerische Mathematik in der Schule. MNU [5] 5/79 S. 266-272

Ade/Schell: Numerische Mathematik, Klett, Stuttgart 1975

Degen, W.: Anwendung der Linearen Algebra in der numerischen Mathematik. PM [6] 10/80 S. 296-307

Deuflhard/Hohmann: Numerische Mathematik, De Gruyter, Berlin 1991

Fedtke, S.: Pascal Algebra - Numerik - Computergraphik, Vieweg, Braunschweig 1987

Hui, E.: Eine Erweiterung des Verfahrens von Newton. Teil 1, PM 7/89 S. 392-397

Kose, K. u.a.: Numerik sehen ... und verstehen, Vieweg, Braunschweig 1992

Kroll, W.: Integration numer. Methoden .. Unterricht Teil 1, PM 8/84 S. 225-234

Kroll, W.: Integration numer. Methoden .. Unterricht Teil 2, PM 9/84 S. 270-280

Kroll, W.: Integration numer. Methoden .. Unterricht Teil 3, PM 10/84 S. 308-312

Maess, G.: Vorlesungen über num. Mathematik I : Lin. Algebra, Birkhäuser, Basel 1985

Pfahl, M.: Numerische Mathematik in der Gymnas. Oberstufe, Bibliographisches Institut, Mannheim 1990

Rasfeld, P.: Num. Behandlung gew. Differentialgleichungen. Teil 1, PM 5/87 S. 281-291

Rasfeld, P.: Num. Behandlung gew. Differentialgleichungen. Teil 2, PM 6/87 S. 355-362

Richenhagen, G.: Carl Runge (1856-1927): Von der reinen Mathematik zur Numerik, Vandenhoeck u. Ruprecht, Göttingen 1985

Stetter, H.J.: Numerik für Informatiker, Computergerechte numerische Verfahren, Oldenbourg, München 1976

[5] Der mathematische und naturwissenschaftliche Unterricht, Dümmler Verlag, Bonn
[6] Praxis der Mathematik, Aulis Verlag, Köln

Ulshöfer, K.: Zum Taschenrechner im Mathematikunterricht. PM 4/94 S. 241-249

Willers, A.: Methoden der praktischen Analysis, De Gruyter, Berlin 1971

Witthinrich, P.: Splineintegration, PM 5/86 S. 294-296

Wolgast, H.: Mathematik u. Informatik: Zur Anwendung von Computern im Unterricht. MNU 8/80 S. 450-454

7. Stichwortverzeichnis

Dieses Lehr- und Übungsbuch

- ist gedacht für
  - Schüler der Sekundarstufe II an allgemein- und berufsbildenden Schulen;
  - Studenten der Anfangssemester an Hochschulen und Universitäten;
  - Teilnehmer an entsprechenden Kursen in Fort- und Weiterbildungsmaßnahmen (wie z.B. Volkshochschulen);
  - alle diejenigen, die sich (unter Zuhilfenahme eines PC) im Selbststudium in grundlegende numerische Verfahren der Mathematik einarbeiten möchten.

- beinhaltet u.a.
  - Struktogramme und Programmlistings von im Buch erwähnten Beispielen. Sämtliche Quelltexte (einschließlich der der Übungen) sind für MS-DOS-Rechner auf einer 3,5"-Diskette verfügbar und ab Turbo Pascal 4.0 bis 7.0 lauffähig (Dümmlerbuch 45432), vgl. Seite 2.
  - eine Verbindung von Mathematik und Informatik. Die Behandlung von Folgen und Reihen, der numerischen Differentation und Integration sowie der näherungsweisen Bestimmung von Nullstellen etc. spielen auch in der Informatik bei der Entwicklung entsprechender Algorithmen eine bedeutsame Rolle, so daß dieses Buch einen interessanten Ansatz für fächerübergreifende Lösungen darbietet.