

# Skript zum Erstellen von Tabellen und Matrizen

Xenia Rendtel

16. März 2010

## Inhaltsverzeichnis

<b>1</b>	<b>Motivation</b>	<b>2</b>
<b>2</b>	<b>Umsetzung</b>	<b>2</b>
2.1	Einlesen von csv-Dateien . . . . .	2
2.1.1	Beispiel . . . . .	2
2.1.2	Weitere Optionen . . . . .	3
2.1.3	Beispiel . . . . .	4
2.2	Eine Wertetabelle setzen . . . . .	4
2.2.1	Weiteres Beispiel . . . . .	5
2.3	Eine Tabelle für die lokale Änderungsrate setzen . . . . .	5
2.3.1	Differentialquotient und lokale Änderungsrate . . . . .	5
2.3.2	Weiteres Beispiel . . . . .	6
2.4	Matrizenoperationen . . . . .	6
2.4.1	Theoretische Grundlagen . . . . .	7
2.4.2	Programmeingaben und Beispiele . . . . .	7
<b>3</b>	<b>Der Programmcode</b>	<b>9</b>
3.1	Die Hauptdateien . . . . .	9
3.2	Eine Beispieleingabedatei . . . . .	35

## Tabellenverzeichnis

1	Eine erste Tabelle . . . . .	3
2	Eine Tabelle mit zwei Nachkommastellen und der <code>tabularx</code> -Umgebung . . . . .	4
3	Eine erste Wertetabelle . . . . .	5
4	Eine zweite Wertetabelle . . . . .	5
5	Eine Tabelle mit der lokalen Änderungsrate . . . . .	6
6	Eine weitere Tabelle mit der lokalen Änderungsrate . . . . .	6

## 1 Motivation

Im Laufe meiner Arbeit sind mir immer wieder Wertetabellen über den Weg gelaufen, die ich schön gesetzt haben wollte, um diese in meine  $\LaTeX$ -Dokumente mit einzufügen. Dazu habe ich früher das Latable-Skript von Alex A. Denisov<sup>1</sup> genutzt. Da ich aber mittlerweile hauptsächlich unter Linux arbeite wollte ich dies automatisieren und nicht immer zwischendurch Windows aufrufen.

In dem Buch von Anselm Lingnau ist ein sehr schönes Kapitel über Tabellen in  $\LaTeX$  enthalten.<sup>2</sup> Mithilfe dessen habe ich mir einiges neu angeeignet. In diesem Kapitel beschreibt der **Hack 40: CSV-Dateien als Tabellen setzen**, wie man csv-Dateien verarbeiten kann. Dieser Hack funktioniert sehr schön, aber ich wollte mich danach noch intensiver mit der Materie selbst auseinandersetzen. Außerdem kamen für mich weitere Anwendungsgebiete hinzu, die ich gleich automatisch gesetzt haben wollte:

- Einlesen von csv-Dateien
- Die Ausrichtung und die Tabellenart soll vorgegeben werden können
- Eine Wertetabelle soll erstellt werden
- Eine Tabelle zur lokalen Änderungsrate soll erstellt werden können
- Aus der Tabelle sollen Wertepaare für eine Zeichenfunktion entnommen werden können.
- Matrizenoperationen sollen automatisch durchgeführt werden
  - Matrizenaddition und -subtraktion
  - Skalarmultiplikation
  - Matrizenmultiplikation
  - Transpositionsollen implementiert werden

Dies wurde im Laufe der Zeit zu meinem Anforderungskatalog.

## 2 Umsetzung

### 2.1 Einlesen von csv-Dateien

Aus einer Tabellenkalkulation heraus kann man sich sehr einfach csv-Dateien speichern. Diese werden mithilfe des Skriptes in eine  $\LaTeX$ -Datei verwandelt.

#### 2.1.1 Beispiel

Im Folgenden einmal ein Eingabe und ein Ausgabebeispiel von Messwerten zur Bestimmung des Planckschen-Wirkungsquantums.

Beispiel für eine csv-Datei

1	\$U\$ in V ;1,2;1,4;1,5;1,56;1,6;1,62;1,64;1,66;1,68;1,7;1,72
2	\$I\$ in mA ;0;0,01;0,03;0,1;0,2;0,35;0,55;0,86;1,25;1,8;2,5

---

<sup>1</sup>Siehe (DENISOV, 2009)

<sup>2</sup>Siehe (LINGNAU, 2007, S. 106 - 139)

Die Ausgabe als L<sup>A</sup>T<sub>E</sub>X-Datei:

erste Ausgabe

```

1 \begin{tabular}{ lllllllllll }
2 $U$ in V & 1,2 & 1,4 & 1,5 & 1,56 & 1,6 & 1,62 & 1,64 & 1,66 & 1,68 & 1,7 & 1,72 \\
3 $I$ in mA & 0 & 0,01 & 0,03 & 0,1 & 0,2 & 0,35 & 0,55 & 0,86 & 1,25 & 1,8 & 2,5 \\
4 \end{tabular}

```

$U$ in V	1,2	1,4	1,5	1,56	1,6	1,62	1,64	1,66	1,68	1,7	1,72
$I$ in mA	0	0,01	0,03	0,1	0,2	0,35	0,55	0,86	1,25	1,8	2,5

**Tab. 1:** Eine erste Tabelle

Eingelesen wurde die csv-Datei auf der Kommandozeile mit

```
./tabellen.pl datei.csv
```

Die Ausgabe kann dann in eine l<sup>A</sup>T<sub>E</sub>X-Datei umgeleitet werden mit

```
./tabellen.pl datei.csv > datei.ltx
```

### 2.1.2 Weitere Optionen

Möchte man nun noch Linien und den vollständigen Rumpf einer L<sup>A</sup>T<sub>E</sub>X-Datei haben, so muss man die zwei Zeilen `kopf` und `linie` in die csv-Datei mit eingeben.

Datei mit L<sup>A</sup>T<sub>E</sub>X-Rumpf und Linien

```

1 %% Autor: X. Rendtel
2 %% Letzte Aenderung: 2010
3
4 \documentclass[10pt, a4paper]{article}
5 \usepackage[utf8]{inputenc}
6 \usepackage{array,longtable,colortbl,multirow,hhline,tabularx,multicol,xcolor}
7 \pagestyle{empty}
8 \begin{document}
9 \begin{tabular}{ lllllllllll }
10 \hline
11 $U$ in V & 1,2 & 1,4 & 1,5 & 1,56 & 1,6 & 1,62 & 1,64 & 1,66 & 1,68 & 1,7 & 1,72 \\
12 $I$ in mA & 0 & 0,01 & 0,03 & 0,1 & 0,2 & 0,35 & 0,55 & 0,86 & 1,25 & 1,8 & 2,5 \\
13 \end{tabular}
14 \end{document}

```

Zusätzlich gibt es noch die Möglichkeit die Tabellenart und die Ausrichtung zu ändern. Dazu gibt es noch die folgenden Befehle:

```

ausrichtung="r, r, r, r"
tabellenkopf="tabellenart", "breite"
nachkommastellen=anzahl

```

Bei dem Befehl `ausrichtung` werden die einzelnen Spalten definiert. Dabei wird die einzelne Spaltenausrichtung mit einem Komma getrennt. Stehen hier weniger Spalten, als die Tabelle hat, wird der Rest mit links ausgerichteten Spalten aufgefüllt. Standardmäßig werden alle Spalten links ausgerichtet, wenn im `tabellenkopf` nicht `tabularx` gesetzt ist. Bei dieser Umgebung werden die Spalten mit `X` gesetzt. Ein weiterer Befehl ist der `tabellenkopf`. Hier wird die Tabellenart und Breite gesetzt. Standardmäßig wird die `tabular`-Umgebung genutzt. Ist der Befehl `nachkommastellen` gesetzt, so werden alle Zahlen mit dieser Anzahl an Nachkommastellen dargestellt.

Nicht zu vergessen sind noch die beiden folgenden Befehle

math

und

text

[Hier steht was drin]

ende

Ist `math` gesetzt, so wird der gesamte Inhalt der Datei auf den  $\text{\LaTeX}$ -Mathemodus gesetzt. Nach dem `text`-Befehl folgt Text, der in die Datei nicht in eine Tabelle gesetzt werden soll. Mit `ende` wird der Befehl beendet.

### 2.1.3 Beispiel

Ein Funktionsaufruf sieht z.B. so aus.

ausrichtung, tabellenkopf und nachkommastellen gesetzt

```

1 linien
2 ausrichtung="|c,|r,|r,|r"
3 tabellenkopf="tabularx","1"
4 nachkommastellen=2
5 $U$ in V ;1,2;1,4;1,5;1,56;1,6;1,62;1,64;1,66;1,68;1,7;1,72
6 $I$ in mA ;0;0,01;0,03;0,1;0,2;0,35;0,55;0,86;1,25;1,8;2,5
    
```

ausrichtung, tabellenkopf und nachkommastellen gesetzt

```

1 \begin{tabularx}{1\linewidth}{|c|l|r|r|X|X|X|X|X|X|X|}
2 \hline
3 $U$ in V & 1,2 & 1,4 & 1,5 & 1,56 & 1,6 & 1,62 & 1,64 & 1,66 & 1,68 & 1,7 & 1,72 \\ \hline
4 $I$ in mA & 0,00 & 0,01 & 0,03 & 0,1 & 0,2 & 0,35 & 0,55 & 0,86 & 1,25 & 1,8 & 2,5 \\ \hline
5 \end{tabularx}
    
```

$U$ in V	1,2	1,4	1,5	1,56	1,6	1,62	1,64	1,66	1,68	1,7	1,72
$I$ in mA	0,00	0,01	0,03	0,1	0,2	0,35	0,55	0,86	1,25	1,8	2,5

**Tab. 2:** Eine Tabelle mit zwei Nachkommastellen und der `tabularx`-Umgebung

## 2.2 Eine Wertetabelle setzen

Es kommt ja häufiger vor, dass man eine Wertetabelle benötigt. Dies kann man natürlich auch mit einer Tabellenkalkulation machen. Da ich aber diese einheitlich und schnell gesetzt haben möchte, habe ich in mein Skript den folgenden Befehl mit eingebaut.

Eine erste Wertetabelle

```

1 linien
2 graphtabelle x=1,10 p=1,5 f(x)=x^2+3
    
```

Eine erste Wertetabelle

```

1 \begin{tabular}{|1|1|1|1|1|1|}
2 \hline
3 $x$ & $f(x)$ & $x$ & $f(x)$ \\ \hline
4 1 & 4 & 6 & 39 \\ \hline
5 2 & 7 & 7 & 52 \\ \hline
6 3 & 12 & 8 & 67 \\ \hline
7 4 & 19 & 9 & 84 \\ \hline
8 5 & 28 & 10 & 103 \\ \hline
9 \end{tabular}
    
```

$x$	$f(x)$	$x$	$f(x)$
1	4	6	39
2	7	7	52
3	12	8	67
4	19	9	84
5	28	10	103

Tab. 3: Eine erste Wertetabelle

Dabei steht  $x=1, 10$  für den Definitionsbereich,  $p=1, v, 5$  gibt die Schrittweite an und ob eine Tabelle horizontal oder vertikal ausgerichtet ist. Die 5 steht in diesem Fall dafür, wie viele Werte pro Zeile oder Spalte eingetragen werden. Vor dem Gleichheitszeichen gibt man den Funktionsnamen und Variablennamen an.

### 2.2.1 Weiteres Beispiel

Hier nun ein Beispiel mit einer horizontal ausgerichteten Tabelle.

Eine zweite Wertetabelle

```

1 linien
2 nachkommastellen=1
3 graphtabelle x=1,10 p=1,h,5 h(t)=x^3+x^2
    
```

$t$	1,0	2,0	3,0	4,0	5,0
$h(t)$	2,0	12,0	36,0	80,0	150,0
$t$	6,0	7,0	8,0	9,0	10,0
$h(t)$	252,0	392,0	576,0	810,0	1100,0

Tab. 4: Eine zweite Wertetabelle

## 2.3 Eine Tabelle für die lokale Änderungsrate setzen

In der Analysis braucht man auch häufig eine Tabelle für die lokale Änderungsrate einer Funktion, wenn man in das Thema der Differenzialrechnung einsteigt. Hierzu schaut man sich die folgende Definition an:

### 2.3.1 Differentialquotient und lokale Änderungsrate

Die Funktion  $f : x \rightarrow f(x)$  sei an der Stelle  $x_0$  und in einer Umgebung von  $x_0$  definiert. Dann bezeichnet man im Falle seiner Existenz den Grenzwert

$$\lim_{x \rightarrow x_0} \frac{f(x) - f(x_0)}{x - x_0} \tag{1}$$

als **Differentialquotient** bzw. als **lokale Änderungsrate** von  $f$  an der Stelle  $x_0$ .<sup>3</sup>

In der Schule verwendet man häufig hierfür zunächst eine Näherungstabelle und geht dann auf den Grenzwertbegriff ein. Da ich zu faul bin die Tabelle per Hand zu rechnen habe ich mir die folgende Funktion geschrieben:

Die lokale Änderungsrate

```

1 linien
2 tabellelokal x=3 p=10,v,5 f(x)=x^2
    
```

<sup>3</sup>Aus (BIGALKE und KÖHLER, 2007)

Die lokale Änderungsrate

```

1 \begin{tabular}{|l|l|l|l|l|}
2 \hline
3 $x$ & $\frac{f(x)-f(3)}{x-3}$ & $x$ & $\frac{f(x)-f(3)}{x-3}$ \\ \hline
4 4 & 16 & 3,0001 & 9,00060001 \\ \hline
5 3,5 & 12,25 & 3,00001 & 9,0000600001 \\ \hline
6 3,1 & 9,61 & 3,000001 & 9,000006000001 \\ \hline
7 3,01 & 9,0601 & 3,0000001 & 9,00000060000001 \\ \hline
8 3,001 & 9,006001 & 3,00000001 & 9,00000006 \\ \hline
9 \end{tabular}

```

$x$	$\frac{f(x)-f(3)}{x-3}$	$x$	$\frac{f(x)-f(3)}{x-3}$
4	16	3,0001	9,00060001
3,5	12,25	3,00001	9,0000600001
3,1	9,61	3,000001	9,000006000001
3,01	9,0601	3,0000001	9,00000060000001
3,001	9,006001	3,00000001	9,00000006

Tab. 5: Eine Tabelle mit der lokalen Änderungsrate

Hierbei gibt  $x=3$  die Stelle  $x_0 = 3$  an,  $p=10$ ,  $v, 5$  gibt an, wie viele Werte vertikal ausgerichtet werden sollen und das in einer Spalte fünf Werte stehen sollen.

### 2.3.2 Weiteres Beispiel

Hier ein weiteres Beispiel:

Die lokale Änderungsrate horizontal ausgerichtet

```

1 linien
2 nachkommastellen=10
3 tabellelokal x=3 p=10,h,5 f(x)=x^2

```

$x$	4,0000000000	3,5000000000	3,1000000000	3,0100000000	3,0010000000
$\frac{f(x)-f(3)}{x-3}$	16,0000000000	12,2500000000	9,6100000000	9,0601000000	9,0060010000
$x$	3,0001000000	3,0000100000	3,0000010000	3,0000001000	3,0000000100
$\frac{f(x)-f(3)}{x-3}$	9,0006000100	9,0000600001	9,0000060000	9,0000006000	9,0000000600

Tab. 6: Eine weitere Tabelle mit der lokalen Änderungsrate

## 2.4 Matrizenoperationen

Nachdem ich nun schon häufiger das Themengebiet der Linearen Algebra und Analytischen Geometrie in der Oberstufe unterrichtet habe, wollte ich mir einen Automatismus basteln, der mir die Arbeit abnimmt. Natürlich hätte ich auch Derive oder Maple dafür zur Verfügung<sup>4</sup>, aber wie wohl Einige bemerkt haben macht mir die Programmierung Spaß!

<sup>4</sup>Siehe (GLYNN, 1995) und (WESTERMANN, 2005)

### 2.4.1 Theoretische Grundlagen

**Matrizenaddition** Seien  $A$  und  $B$  zwei  $m \times n$  Matrizen aus  $K^{m \times n}$ , dann ist die Summe von  $A$  und  $B$  ist definiert als

$$A + B := (a_{ij} + b_{ij})_{i=1, \dots, m; j=1, \dots, n} \quad (2)$$

d.h. man addiert zwei Matrizen, indem man die einzelnen Einträge addiert. Zu beachten ist, dass man nur Matrizen von gleichen Zeilen- und Spaltenrang addieren kann.

**Matrizensubtraktion** Seien  $A$  und  $B$  zwei  $m \times n$  Matrizen aus  $K^{m \times n}$ , dann ist die Differenz von  $A$  und  $B$  definiert als

$$A - B := (a_{ij} - b_{ij})_{i=1, \dots, m; j=1, \dots, n} \quad (3)$$

d.h. man subtrahiert zwei Matrizen, indem man die einzelnen Einträge subtrahiert. Zu beachten ist, dass man nur Matrizen von gleichen Zeilen- und Spaltenrang subtrahieren kann.

**Skalarmultiplikation** Eine Matrix  $A \in K^{m \times n}$  wird mit einem Skalar  $r \in \mathbb{R}$  multipliziert, indem alle Einträge mit dem Skalar multipliziert werden.

$$r \cdot A = (r \cdot a_{ij})_{i=1, \dots, m; j=1, \dots, n} \quad (4)$$

**Matrixmultiplikation** Seien  $A \in K^{m \times n}$  und  $B \in K^{n \times q}$  zwei Matrizen. Das Produkt  $A \cdot B$  ist definiert als

$$A \cdot B = (c_{ij})_{i=1, \dots, m; j=1, \dots, q} \quad (5)$$

mit

$$(c_{ij})_{i=1, \dots, m; j=1, \dots, q} = \sum_{k=1}^n a_{ik} b_{kj} \quad (6)$$

**Transponierte Matrix** Wenn in einer  $m \times n$  Matrix  $A$  die Zeilen mit den entsprechenden Spalten vertauscht werden, so entsteht eine Matrix  $A^T$ , die sogenannte **transponierte Matrix**.

### 2.4.2 Programmeingaben und Beispiele

Matrizen werden in mein Programm mit dem Befehl `ende` beendet. Hier eine Beispieleingabe:

Eingabe einer Matrix	Ausgabe von Matrizen
1 1;2;3	1 \begin{tabular}{lll}
2 4;5;6	2 1 & 2 & 3 \\\
3 7;8;9	3 4 & 5 & 6 \\\
4 ende	4 7 & 8 & 9 \\\
5 1;2;3	5 \end{tabular}
6 5;6;7	6 \begin{tabular}{lll}
7 8;9;10	7 1 & 2 & 3 \\\
8 ende	8 5 & 6 & 7 \\\
	9 8 & 9 & 10 \\\
	10 \end{tabular}

Setzt man als `tabellenkopf` die `pmatrix` ein, so erhält man auch gleich die Matrixklammern. Wie wird aber nun gerechnet? Hierfür gibt es verschiedene Eingabemöglichkeiten. Zum einen kann man zwischen die Matrizen in eine neue Zeile die Operationen  $+$ ,  $-$  oder  $*$  schreiben und erhält dann die passende Ausgabe, oder man nutzt die folgenden Befehle

1. `matrixadd [matrix1] [matrix2]`
2. `matrixsub [matrix1] [matrix2]`
3. `matrixmul [matrix1] [matrix2]`
4. `transponiert [matrix1]`

Hierbei sind die `matrix1` und `matrix2` optionale Argumente. Sind diese nicht gesetzt, so werden jeweils die erste und zweite Matrix aus der Datei gelesen. Bei der Transposition wird standardmäßig die erste Matrix genutzt. Man kann aber z.B. mit `matrixmul 2 2` das Quadrat der zweiten Matrix berechnen lassen.

Für die folgenden Beispiele werden jeweils die obigen Matrizen als Beispiel genutzt und zusätzlich `math` und `tabellenkopf="pmatrix", ""` in die Dateien mit eingefügt.

### Addition Ein Beispiel

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} + \begin{pmatrix} 1 & 2 & 3 \\ 5 & 6 & 7 \\ 8 & 9 & 10 \end{pmatrix} = \begin{pmatrix} 1+1 & 2+2 & 3+3 \\ 4+5 & 5+6 & 6+7 \\ 7+8 & 8+9 & 9+10 \end{pmatrix} \\ = \begin{pmatrix} 2 & 4 & 6 \\ 9 & 11 & 13 \\ 15 & 17 & 19 \end{pmatrix}$$

### Subtraktion Ein Beispiel

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} - \begin{pmatrix} 1 & 2 & 3 \\ 5 & 6 & 7 \\ 8 & 9 & 10 \end{pmatrix} = \begin{pmatrix} 1-1 & 2-2 & 3-3 \\ 4-5 & 5-6 & 6-7 \\ 7-8 & 8-9 & 9-10 \end{pmatrix} \\ = \begin{pmatrix} 0 & 0 & 0 \\ -1 & -1 & -1 \\ -1 & -1 & -1 \end{pmatrix}$$

### Multiplikation Ein Beispiel:

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \cdot \begin{pmatrix} 1 & 2 & 3 \\ 5 & 6 & 7 \\ 8 & 9 & 10 \end{pmatrix} \\ = \begin{pmatrix} 1 \cdot 1 + 2 \cdot 5 + 3 \cdot 8 & 1 \cdot 2 + 2 \cdot 6 + 3 \cdot 9 & 1 \cdot 3 + 2 \cdot 7 + 3 \cdot 10 \\ 4 \cdot 1 + 5 \cdot 5 + 6 \cdot 8 & 4 \cdot 2 + 5 \cdot 6 + 6 \cdot 9 & 4 \cdot 3 + 5 \cdot 7 + 6 \cdot 10 \\ 7 \cdot 1 + 8 \cdot 5 + 9 \cdot 8 & 7 \cdot 2 + 8 \cdot 6 + 9 \cdot 9 & 7 \cdot 3 + 8 \cdot 7 + 9 \cdot 10 \end{pmatrix} \\ = \begin{pmatrix} 35 & 41 & 47 \\ 77 & 92 & 107 \\ 119 & 143 & 167 \end{pmatrix}$$

### Skalarmultiplikation

$$2 \cdot \begin{pmatrix} 1 & 2 & 3 \\ 5 & 6 & 7 \\ 8 & 9 & 10 \end{pmatrix} = \begin{pmatrix} 2 \cdot 1 & 2 \cdot 2 & 2 \cdot 3 \\ 2 \cdot 5 & 2 \cdot 6 & 2 \cdot 7 \\ 2 \cdot 8 & 2 \cdot 9 & 2 \cdot 10 \end{pmatrix} \\ = \begin{pmatrix} 2 & 4 & 6 \\ 10 & 12 & 14 \\ 16 & 18 & 20 \end{pmatrix}$$

Bei der Skalarmultiplikation ist zu beachten, dass ein Skalar wie eine Tabelle gesetzt wird, aber hier nur ein Eintrag erstellt wird.

## Skalarmultiplikation

```

1 tabellenkopf="pmatrix",""
2 math
3
4 2;
5 ende
6 *
7 1;2;3
8 5;6;7
9 8;9;10
10 ende

```

## Transponierte Matrix

Die Eingabe ist dabei die folgende:

$$\begin{pmatrix} 1 & 2 & 3 \\ 5 & 6 & 7 \\ 8 & 9 & 10 \end{pmatrix}^T = \begin{pmatrix} 1 & 5 & 8 \\ 2 & 6 & 9 \\ 3 & 7 & 10 \end{pmatrix}$$

## Transponierte Matrix

```

1 tabellenkopf="pmatrix",""
2 math
3
4 1;2;3
5 5;6;7
6 8;9;10
7 ende
8 text
9 ^T
10 =
11 ende
12
13 transponiert

```

### 3 Der Programmcode

Das gesamte Skript gliedert sich in drei Dateien. In `tabellen.pl` wird die Eingabedatei ausgelesen. `tabelle.pm` beinhaltet alle Funktionen für das Tabellenskript. In `hilfsfunktionen.pm` sind Funktionen definiert, die auch für weitere Skripte benötigt werden.

#### 3.1 Die Hauptdateien

## tabellen.pl

```

1 #!/usr/bin/perl -w
2 #####
3 # Skript um csv-Dateien in Latex zu konvertieren
4 # Copyright (C) 2008 Xenia Rendtel
5 #
6 # VERSION 1
7 # Letzte Aenderung: 15.04.2009
8 #
9 # Dieses Programm ist freie Software. Sie koennen es unter den Bedingungen der
10 # GNU General Public License, wie von der Free Software Foundation
11 # veröffentlicht, weitergeben und/oder modifizieren, entweder gemaess Version 2
12 # der Lizenz oder (nach Ihrer Option) jeder spaeteren Version.
13 #
14 # Die Veroeffentlichung dieses Programms erfolgt in der Hoffnung, dass es Ihnen

```

```

15 # von Nutzen sein wird, aber OHNE IRGENDNEINE GARANTIE, sogar ohne die implizite
16 # Garantie der MARKTREIFE oder der VERWENDBARKEIT
17 # ZWECK. Details finden Sie in der GNU General Public License.
18 #
19 # Sie sollten ein Exemplar der GNU General Public License zusammen mit diesem
20 # Programm erhalten haben. Falls nicht, schreiben Sie an die Free Software
21 # Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110, USA.
22 #
23 # Bei Fehlern schreiben Sie diese einfach an
24 # webmaster@rendtel.de
25 #####
26 ###Weitere Pakete
27 use POSIX qw /floor ceil/;
28 use Math::Trig;
29 use strict ;
30 use warnings;
31 use FindBin;
32 use lib "$FindBin::Bin";
33
34 use lib '/home/xenia/Schule/Programme/Tabellen';
35
36 use tabelle ;
37 use hilfsmfunktionen;
38
39 my (
40     $kommandodatei, $texdatei, $i,
41
42     $leer,           $eingabedateizeilen, @eingabedatei,
43     @ausgabedatei, @ausgabedateizeilen, $kommandodateinummer,
44     $datei, $zeile, $layer, $xmin, $xmax, $ausrichtung, $bezeichnung, $func,
45     $variable,
46     $schritt, $spalten, $kopf,
47     $nachkommastellen, $spaltenausrichtung, $liste, $addition, $multiplikation,
48     $subtraktion,    $math
49 );
50
51 $spaltenausrichtung = "";
52 $kopf                = 0;
53 $liste               = 0;
54 $addition            = 0;
55 $math                = 0;
56
57 $multiplikation = 0;
58 $subtraktion    = 0;
59
60 # Arrays und Layer
61 $eingabedateizeilen = 0;
62 @eingabedatei       = ();
63 @ausgabedatei       = ();
64 @ausgabedateizeilen = ();
65 $kommandodateinummer = 0;
66
67 $datei = 0;
68
69 for ( $i = 0 ; $i <= 150 ; $i++ ) {
70     $ausgabedateizeilen[$i] = 0;
71 }
72
73 # Die Eingabedatei wird gelesen
74 sub leseingabedatei {
75     my $dateiname = shift(@_);

```

```

76 my $zeile;
77 open( my $EINGABEDATEI, $dateiname ) || die $dateiname . ": $!";
78 while ( $zeile = <$EINGABEDATEI> ) {
79
80     # Zeilenenden beseitigen, Kommentare und Leerzeilen ignorieren
81     $zeile =~ s/[\r\n]//g;
82     $zeile =~ s/ +/ /g;
83     $zeile =~ s/ +$/ /g;
84     $zeile =~ s/^ +//g;
85     next if ( $zeile =~ /^^\#/ );
86     next if ( $zeile =~ /^$/ );
87     if ( $zeile =~ /^lesedatei (.*)$/i ) {
88         leseingabedatei($1);
89     }
90     else { #Einzelne Zeilen in ein Array speichern
91         $eingabedatei[$eingabedateizeilen] = $zeile;
92         $eingabedateizeilen++;
93     }
94 }
95 close($EINGABEDATEI);
96 }
97
98 # In ein array wird geschrieben
99 sub schreibeinarray {
100     my ( $layer, $zeile ) = @_ ;
101     $ausgabedatei[$layer][$ausgabedateizeilen[$layer] ] = $zeile;
102     $ausgabedateizeilen[$layer]++;
103 }
104
105 # Der Kopf der Latex-Datei wird geschrieben
106 sub kopfdatei {
107
108     if ( $kopf == 1 ) {
109         kommentar( 1, "Autor: X. Rendtel" );
110         kommentar( 1, "Letzte Aenderung: 2010 \n" );
111         latex( 1, "\\documentclass[10pt, a4paper]{article}" );
112         usepackage( "utf8", "inputenc" );
113         usepackage( "",
114             "array,longtable,colortbl,multitrow,hhline,tabularx,multicol,xcolor" );
115         latex( 3, "\\pagestyle{empty} \n\\begin{document}" );
116     }
117     if ( $math == 1 ) { latex( 9, "\\$" ); }
118 }
119
120 sub fussdatei {
121     if ( $math == 1 ) { latex( 11, "\\$" ); }
122     if ( $kopf == 1 ) { latex( 11, "\\end{document}" ); }
123 }
124
125 # Der Standard fuer die Tabellen wird gesetzt
126 settabelle( "tabular", 1 );
127
128 ### Einlesen der Datei
129 while ( exists $ARGV[$kommandodateinummer] ) {
130     $kommandodatei = $ARGV[$kommandodateinummer];
131     $texdatei = $kommandodatei;
132     $texdatei =~ s/\.ptxt$/ /i;
133     $texdatei = $texdatei . ". xtex ";
134     $kommandodateinummer++;
135     $eingabedateizeilen = 0;
136     leseingabedatei($kommandodatei);

```

```

137 for ( $i = 0 ; $i < $eingabedateizeilen ; $i++ ) {
138     $zeile = $eingabedatei[$i];
139
140     # Eine Wertetabelle
141     if ( $zeile =~
142 /graphtabelle( layer=([0-9]+))? x=([-0-9\\.]+),([-0-9\\.]+) p=([-0-9\\.]+),([a-z0-9!]+),([-0-9\\.]+) ([A-Za-
143     z0-9!]+)\((([A-Za-z0-9!]+)\)=(.*)$/i
144     )
145     {
146         $layer      = 10;
147         $layer      = $2 if ( defined $1 );
148         $xmin       = $3;
149         $xmax       = $4;
150         $schritt    = $5;
151         $ausrichtung = $6;
152         $spalten    = $7;
153         $bezeichnung = $8;
154         $variable   = $9;
155         $func       = $10;
156         funktabelle(
157             $layer, $func, $xmin, $xmax, $schritt,
158             $spalten, $bezeichnung, $variable, $ausrichtung
159         );
160     }
161
162     # Die Nachkommastellen setzen
163     elseif ( $zeile =~ /nachkommastellen=(([-0-9\\.]+)$/i ) {
164         setnachkommastellen($1);
165     }
166
167     # Die Tabellenart umdefinieren
168     elseif ( $zeile =~ /tabellenkopf=\"([^\"]*)\"\\\"([^\"]*)\"$/i ) {
169         settabelle ( $1, $2 );
170     }
171
172     # Eine Tabelle fuer die lokale Aenderungsrage
173     elseif ( $zeile =~
174 /tabellelokal( layer=([0-9]+))? x=([-0-9\\.]+) p=([-0-9\\.]+),([a-zA-Z0-9!]+),([-0-9\\.]+) ([A-Za-z0-9!]+)
175     \((([A-Za-z0-9!]+)\)=(.*)$/i
176     )
177     {
178         $layer      = 10;
179         $layer      = $2 if ( defined $1 );
180         $xmin       = $3;
181         $xmax       = $4;
182         $schritt    = $5;
183         $ausrichtung = $6;
184         $spalten    = $7;
185         $bezeichnung = $8;
186         $variable   = $9;
187         $func       = $10;
188
189         lokaltabelle (
190             $layer, $func, $xmin, $schritt, $spalten, $bezeichnung, $variable,
191             $ausrichtung
192         );
193     }
194
195     # Text mit ; getrennt einlesen
196     elseif ( $zeile =~ /text([^\;]*) ;(.*)$/i ) { tabelle( 10, $zeile ); }

```

```

196 # Die Ausrichtung umdefinieren
197 elif ( $zeile =~ /^ausrichtung=\\"(.*)\"$/i ) {
198     $spaltenausrichtung = $1;
199 }
200 elif ( $zeile =~ /^liste=([-0-9\\.]+)([-0-9\\.]+)(.*)$/i ) {
201     liste ( $1, $2, $3 );
202     $liste = 1;
203 }
204
205 # den Rumpf mitbauen
206 elif ( $zeile =~ /^kopf$/i ) { $kopf = 1; }
207
208 # Mathemodus einstellen
209 elif ( $zeile =~ /^math$/i ) { $math = 1; }
210
211 # Linien zeichnen
212 elif ( $zeile =~ /^linien$/i ) { setlinien (); }
213
214 # Tabelle beenden
215 elif ( $zeile =~ /^ende$/i ) { schreibetabelle (); }
216
217 # Opertaionen fuer die Matrizen
218 elif ( $zeile =~ /^\\+$/i ) {
219     ausgabe("+");
220     $addition = 1;
221 }
222
223 elif ( $zeile =~ /^\\-$/i ) {
224     ausgabe("-");
225     $subtraktion = 1;
226 }
227
228 elif ( $zeile =~ /^\\*$/i ) {
229     ausgabe("\\*");
230     $multiplikation = 1;
231 }
232
233 elif ( $zeile =~ /^matrixadd( ([-0-9\\.]+) ([-0-9\\.]+)?$/i ) {
234     if ( defined $1 ) {
235         matrixaddition( $2, $3, 0 );
236         matrixaddition( $2, $3, 1 );
237     }
238     else {
239         matrixaddition( 1, 2, 0 );
240         matrixaddition( 1, 2, 1 );
241     }
242 }
243 }
244
245 elif ( $zeile =~ /^matrixmul( ([-0-9\\.]+) ([-0-9\\.]+)?$/i ) {
246     if ( defined $1 ) {
247         matrixmultiplikation( $2, $3, 0 );
248         matrixmultiplikation( $2, $3, 1 );
249     }
250     else {
251         matrixmultiplikation( 1, 2, 0 );
252         matrixmultiplikation( 1, 2, 1 );
253     }
254 }
255 }
256

```

```

257 elif ( $zeile =~ /^matrixsub( ([-0-9\.]+) ([-0-9\.]+)?)$/i ) {
258     if ( defined $1 ) {
259         matrixsubtraktion( $2, $3, 0 );
260         matrixsubtraktion( $2, $3, 1 );
261     }
262     else {
263         matrixsubtraktion( 1, 2, 0 );
264         matrixsubtraktion( 1, 2, 1 );
265     }
266 }
267
268
269 elif ( $zeile =~ /^transponiert( ([-0-9\.]+)?)$/i ) {
270     if ( defined $1 ) { transponiert($2); }
271     else { transponiert(1); }
272 }
273
274 elif ( $zeile =~ /^text$/i ) {
275     schreibetabelle ();
276     my $enderreicht = 0;
277     my $hilfszeile;
278     my $inzeile = 0;
279     while ( ( !$enderreicht )
280         && ( $i < $eingabedateizeilen - 1 ) )
281     {
282         $i++;
283         $hilfszeile = $eingabedatei[$i];
284         if ( $hilfszeile =~ /^ende$/i ) {
285             $enderreicht = 1;
286         }
287         elsif ( $hilfszeile ne "" ) {
288             if ( $inzeile == 0 ) {
289                 }
290             $inzeile++;
291             latex( 10, $hilfszeile );
292         }
293         else {
294             printf("Kaputte Zeile\n");
295             $enderreicht = 1;
296         }
297     }
298     if ( !$enderreicht ) { printf("Keine sinnvollen Daten\n"); }
299 }
300
301
302 # Alle anderen Befehle sind Schrott
303 else {
304     printf( "%% Unbekannter Befehl '%s'\n", $zeile );
305 }
306 }
307 }
308
309 if ( $subtraktion == 1 ) {
310     ausgabe("=");
311     matrixsubtraktion( 1, 2, 0 );
312     ausgabe("=");
313     matrixsubtraktion( 1, 2, 1 );
314 }
315
316 if ( $addition == 1 ) {
317     ausgabe("=");

```

```

318 matrixaddition( 1, 2, 0 );
319 ausgabe("=");
320 matrixaddition( 1, 2, 1 );
321 }
322
323 if ( $multiplikation == 1 ) {
324   ausgabe("=");
325   matrixmultiplikation( 1, 2, 0 );
326   ausgabe("=");
327   matrixmultiplikation( 1, 2, 1 );
328 }
329
330 $leer = 1;
331
332 if ( $liste == 0 ) {
333   if ( $spaltenausrichtung ne "" ) {
334     tabellenausrichtung($spaltenausrichtung);
335   }
336   schreibetabelle ();
337 }
338
339 for ( $layer = 0 ; $layer <= 100 ; $layer++ ) {
340   for ( $i = 0 ; $i < $ausgabedateizeilen[$layer] ; $i++ ) {
341     $leer = 0;
342   }
343 }
344 if ( $leer == 0 ) {
345   kopfdatei();
346   fussdatei ();
347
348   if ( $kopf == 1 ) {
349
350     for ( $layer = 0 ; $layer <= 100 ; $layer++ ) {
351       for ( $i = 0 ; $i < $ausgabedateizeilen[$layer] ; $i++ ) {
352         printf( "%s \n", $ausgabedatei[$layer][$i] );
353       }
354     }
355   }
356   else {
357     for ( $layer = 9 ; $layer <= 100 ; $layer++ ) {
358       for ( $i = 0 ; $i < $ausgabedateizeilen[$layer] ; $i++ ) {
359         printf( "%s \n", $ausgabedatei[$layer][$i] );
360       }
361     }
362   }
363 }
364 else {
365   printf("%% Die Datei ist leer \n");
366 }

```

tabelle.pm

```

1 #####
2 # Modul fuer csv-Tabellen einlesen und in Latex umwandeln
3 #
4 # VERSION 1
5 # Letzte Aenderung: 26.02.2009
6 #####
7 use POSIX qw /floor ceil/;
8 use Math::Trig;
9 use strict ;

```

```

10 use warnings;
11
12 # Variablen
13
14 my (
15     $tabelle,      $tabellenkopf,      $tabellenfuss,
16     $spaltenzahl, $zeilenzahl,        $spaltendefinition,
17     $i,            $tabellenausrichtung, $spaltenausrichtung,
18     $nachkommastelle, $matrix1zeilen,  $matrix2zeilen,
19     $matrix1spalten, $matrix2spalten,  $matrixzaehler
20 );
21
22 $zeilenzahl = $spaltenzahl = $tabelle = 0;
23 $spaltendefinition = "";
24 my @zeile      = ();
25 my @spalte    = ();
26 my @matrixzeile = ();
27 my @matrixspalte = ();
28 my @matrixstart = ();
29
30 my @zeileinhalt = ();
31 my @spalteinhalt = ();
32 $tabellenausrichtung = 0;
33 my $tabellenart = "tabular";
34 my @lange      = ();
35 my $linien     = 0;
36
37 for ( $i = 0 ; $i <= 150 ; $i++ ) {
38     $matrixzeile[$i] = $matrixspalte[$i] = $matrixstart[$i] = 0;
39 }
40
41 $matrixzaehler = 1;
42
43 # Funktionen fuer die Tabellendefinition in Latex
44
45 $tabellenkopf = $tabellenfuss = "{ " . $tabellenart . " }";
46
47 sub settabellenart {
48     my ($art) = @_;
49     $tabellenart = $art;
50 }
51
52 sub gettabellenart { return $tabellenart; }
53
54 sub settabelle {
55     my ( $tabellenart, $weite ) = @_;
56     my ( $kopf,        $fuss );
57
58     settabellenart ( $tabellenart );
59
60     if ( $tabellenart eq "tabularx" ) {
61         $kopf = "{" . $tabellenart . "}" . $weite . "\\linewidth";
62         $fuss = "{" . $tabellenart . " }";
63     }
64     else {
65         $kopf = "{" . $tabellenart . " }";
66         $fuss = $kopf;
67     }
68 }
69 settabellenkopf($kopf);
70 settabellenfuss($fuss);

```

```
71 | }
72 |
73 | # Die Anzahl der Nachkommastellen wird gesetzt
74 |
75 | my $kommagesetzt = 0;
76 | $nachkommastelle = -1;
77 |
78 | sub setnachkommastellen {
79 |   my ($komma) = @_;
80 |   $nachkommastelle = $komma;
81 |   $kommagesetzt = 1;
82 | }
83 |
84 | sub setspaltendef {
85 |   my ($spalten) = @_;
86 |   $spaltendefinition = $spalten;
87 | }
88 |
89 | sub getspaltendef { return $spaltendefinition; }
90 |
91 | # Sollen Linien gezeichnet werden?
92 |
93 | sub setlinien { $linien = 1; }
94 |
95 | sub getlinien { return $linien; }
96 |
97 | my $amsmath = 0;
98 |
99 | # Der Tabellenkopf wird geschrieben
100 | sub tabellenkopf {
101 |   my ( $layer, $spalteninhalt ) = @_;
102 |
103 |   if ( $tabellenart eq "pmatrix" ) {
104 |     if ( $amsmath == 0 ) {
105 |       usepackage( "", "amsmath" );
106 |       $amsmath = 1;
107 |     }
108 |     schreibeinarray( $layer, sprintf( "\\begin%s", gettabellenkopf() ) );
109 |   }
110 |   else {
111 |     schreibeinarray( $layer,
112 |       sprintf( "\\begin{s}{%s}", gettabellenkopf(), $spalteninhalt ) );
113 |     if ( getlinien() == 1 ) {
114 |       latex( $layer, "\\hline" );
115 |     }
116 |   }
117 | }
118 |
119 | sub gettabellenkopf { return $tabellenkopf; }
120 |
121 | sub settabellenkopf {
122 |   my ($kopf) = @_;
123 |   $tabellenkopf = $kopf;
124 | }
125 |
126 | sub settabellenfuss {
127 |   my ($fuss) = @_;
128 |   $tabellenfuss = $fuss;
129 | }
130 |
131 | sub tabellenfuss {
```

```

132 my ( $layer, $fuss ) = @_;
133 schreibeinarray( $layer, sprintf( "\\end%", $fuss ) );
134 }
135
136 sub gettabellenfuss { return $tabellenfuss; }
137
138 # Funktion um eine Tabelle fuer die lokale Aenderungsrage zu berechnen
139
140 sub lokaltabelle {
141 my ( $layer, $func, $xlokal, $schritt, $spalten, $bezeichnung, $variable,
142     $ausrichtung )
143     = @_;
144
145 my (
146     @xpunkte, @ypunkte, $i,    $x,    $y,    $steiler, $xzeile,
147     $yzeile,  $j,    $punkte, $zahler, $zeichen, $n
148 );
149 $punkte = 0;
150 $zahler = 0;
151 if ( $schritt >= 1 ) {
152     $x = $xlokal + 1;
153     $y = calcfunk( $func, "x", $x );
154     $xpunkte[$punkte] = nachkommastellen( $x, $nachkommastelle );
155     $ypunkte[$punkte] = nachkommastellen( $y, $nachkommastelle );
156     $punkte++;
157 }
158
159 if ( $schritt >= 2 ) {
160     $x = $x - 0.5;
161     $y = calcfunk( $func, "x", $x );
162     $xpunkte[$punkte] = nachkommastellen( $x, $nachkommastelle );
163     $ypunkte[$punkte] = nachkommastellen( $y, $nachkommastelle );
164     $punkte++;
165 }
166
167 if ( $schritt >= 3 ) {
168     for ( $i = 3 ; $i <= $schritt ; $i++ ) {
169         $x = $xlokal + 0.1 / ( 10** ( $i - 3 ) );
170         $y = calcfunk( $func, "x", $x );
171         $xpunkte[$punkte] = nachkommastellen( $x, $nachkommastelle );
172         $ypunkte[$punkte] = nachkommastellen( $y, $nachkommastelle );
173         $punkte++;
174     }
175 }
176
177 if ( $punkte <= $spalten ) { $spalten = $punkte; }
178
179 if ( $ausrichtung eq "v" ) {
180     if ( ( $punkte % $spalten ) == 0 ) {
181         $steiler = abrunden( $punkte / $spalten, 1 ) - 1;
182     }
183     else { $steiler = abrunden( $punkte / $spalten, 1 ); }
184
185     $xzeile = "";
186     for ( $j = 0 ; $j <= $steiler ; $j++ ) {
187         if ( $j == $steiler ) { $zeichen = " "; }
188         else { $zeichen = " "; }
189         $xzeile =
190             $xzeile . "\$"
191             . $variable . "\$";
192         $xzeile =

```

```

193     . $bezeichnung . "("
194     . $variable . ")—"
195     . $bezeichnung . "("
196     . $xlokal . "){"
197     . $variable . ")—"
198     . $xlokal . "} \ $"
199     . $zeichen;
200
201 }
202 tabelle( $layer, $xzeile );
203
204 for ( $i = 0 ; $i < $spalten ; $i++ ) {
205     $xzeile = "";
206     for ( $j = 0 ; $j <= $teiler ; $j++ ) {
207         if ( $j == $teiler ) { $zeichen = ""; }
208         else { $zeichen = " ; " ; }
209         if ( ( $i + $j * $spalten ) > $punkte ) {
210             $xzeile = $xzeile . " ;" . $zeichen;
211         }
212         else {
213             $xzeile = $xzeile
214                 . $xpunkte[ $i + $j * $spalten ] . "" . " ; "
215                 . $ypunkte[ $i + $j * $spalten ] . ""
216                 . $zeichen;
217         }
218     }
219     tabelle( $layer, $xzeile );
220 }
221
222 }
223 else {
224     $teiler = aufrunden( $punkte / $spalten, 1 );
225
226     for ( $i = 1 ; $i <= $teiler ; $i++ ) {
227         $xzeile = "\ $" . $variable . "\ $";
228         $yzeile = "\ $\frac{"
229             . $bezeichnung . "("
230             . $variable . ")—"
231             . $bezeichnung . "("
232             . $xlokal . "){"
233             . $variable . ")—"
234             . $xlokal . "} \ $";
235
236         for ( $j = 1 ; $j <= $spalten ; $j++ ) {
237             if ( $j == $spalten ) { $zeichen = " " ; }
238             else { $zeichen = " ; " ; }
239             if ( $zahler > $punkte ) {
240                 $xzeile = $xzeile . $zeichen;
241                 $yzeile = $yzeile . $zeichen;
242             }
243         }
244         else {
245             $xzeile = $xzeile . "" . $xpunkte[ $zahler ] . "" . $zeichen;
246             $yzeile = $yzeile . "" . $ypunkte[ $zahler ] . "" . $zeichen;
247         }
248         $zahler++;
249     }
250     tabelle( $layer, $xzeile );
251     tabelle( $layer, $yzeile );
252 }
253 }

```

```

254 }
255
256 # Funktion um eine Wertetabelle fuer eine Funktion anzulegen
257
258 sub funktabelle {
259   my (
260     $layer,      $func,  $xmin,    $xmax,
261     $schritt,   $spalten, $bezeichnung, $variable,
262     $ausrichtung, $schrittzahl
263   )
264   = @_;
265
266   my (
267     @xpunkte, @ypunkte, $i,    $x,    $y,    $teiler, $xzeile,
268     $yzeile, $j,    $punkte, $zahler, $zeichen, $n
269   );
270   $schrittzahl = ( $xmax - $xmin ) / $schritt;
271
272   $zahler = 1;
273   $xzeile = $yzeile = "";
274
275   for ( $n = 0 ; $n <= $schrittzahl ; $n++ ) {
276     $punkte++;
277     $x = $xmin + ( $xmax - $xmin ) * ( $n / ($schrittzahl) );
278     $y = calcfunk( $func, "x", $x );
279     $xpunkte[$punkte] = nachkommastellen( $x, $nachkommastelle );
280     $ypunkte[$punkte] = nachkommastellen( $y, $nachkommastelle );
281   }
282
283   if ( $punkte < $spalten ) { $spalten = $punkte; }
284
285   if ( $ausrichtung eq "v" ) {
286     if ( ( $punkte % $spalten ) == 0 ) {
287       $teiler = abrunden( $punkte / $spalten, 1 ) - 1;
288     }
289     else { $teiler = abrunden( $punkte / $spalten, 1 ); }
290
291     $xzeile = "";
292     for ( $j = 0 ; $j <= $teiler ; $j++ ) {
293       if ( $j == $teiler ) { $zeichen = " "; }
294       else { $zeichen = " "; "; }
295       $xzeile =
296         $xzeile . "\$"
297         . $variable . "\$"; " . "\$"
298         . $bezeichnung . "("
299         . $variable . ")\"
300         . $zeichen;
301
302     }
303     tabelle( $layer, $xzeile );
304
305     for ( $i = 1 ; $i <= $spalten ; $i++ ) {
306       $xzeile = "";
307       for ( $j = 0 ; $j <= $teiler ; $j++ ) {
308         if ( $j == $teiler ) { $zeichen = " "; }
309         else { $zeichen = " "; "; }
310         if ( ( $i + $j * $spalten ) > $punkte ) {
311           $xzeile = $xzeile . " "; " . $zeichen;
312         }
313         else {
314           $xzeile = $xzeile

```

```

315         . $xpunkte[ $i + $j * $spalten ] . "" . " ; "
316         . $ypunkte[ $i + $j * $spalten ] . ""
317         . $zeichen;
318     }
319 }
320     tabelle( $layer, $xzeile );
321 }
322 }
323 else {
324     $teiler = aufrunden( $punkte / $spalten, 1 );
325
326     for ( $i = 1 ; $i <= $teiler ; $i++ ) {
327         $xzeile = "\$" . $variable . "\$ ";
328         $yzeile = "\$" . $bezeichnung . "(" . $variable . ")\"$ ";
329
330         for ( $j = 1 ; $j <= $spalten ; $j++ ) {
331             if ( $j == $spalten ) { $zeichen = " "; }
332             else { $zeichen = " "; }
333             if ( $zahler > $punkte ) {
334                 $xzeile = $xzeile . $zeichen;
335                 $yzeile = $yzeile . $zeichen;
336
337             }
338             else {
339                 $xzeile = $xzeile . "" . $xpunkte[$zahler] . "" . $zeichen;
340                 $yzeile = $yzeile . "" . $ypunkte[$zahler] . "" . $zeichen;
341             }
342             $zahler++;
343         }
344         tabelle( $layer, $xzeile );
345         tabelle( $layer, $yzeile );
346     }
347 }
348 }
349 }
350 }
351
352 # Die Spalten- und Zeilenzahl wird bestimmt
353 sub setspalten {
354     my ( $spalte ) = @_;
355     if ( $spalte > $spaltenzahl ) { $spaltenzahl = $spalte; }
356     else { $spaltenzahl = $spaltenzahl; }
357     if ( $spalte == 0 ) { $spaltenzahl = $spalte; }
358 }
359
360 sub setzeilen {
361     my ( $zeile ) = @_;
362     if ( $zeile > $zeilenzahl ) { $zeilenzahl = $zeile; }
363     else { $zeilenzahl = $zeilenzahl; }
364 }
365
366 sub getspalten { return $spaltenzahl; }
367 sub getzeilen { return $zeilenzahl; }
368
369 # Die maximale Spaltenbreite wird festgestellt
370 sub spaltenbreite {
371     my ( $spalte, $breite ) = @_;
372
373     if ( $länge[$spalte] < length($breite) ) {
374         $länge[$spalte] = length($breite);
375     }

```

```

376   else { $lange[$spalte] = $lange[$spalte]; }
377 }
378
379 # Die Spaltenbreite wird wieder zurueckgesetzt fuer die naechsten Tabellen
380
381 sub spaltenbreiterueck {
382   my $j;
383   for ( $j = 0 ; $j < getspalten() ; $j++) { $lange[$j] = 0; }
384 }
385
386 for ( $i = 0 ; $i <= 1000 ; $i++ ) {
387   $spalte[$i] = $spalteinhalt[$i] = $lange[$i] = 0;
388 }
389
390 # Eintraege in die Tabelle werden vorgenommen
391 sub schreibeintabelle {
392   my ( $zeilennummer, $zelle ) = @_ ;
393   $zeile[$zeilennummer][$spalte[$zeilennummer]] = $zelle;
394   spaltenbreite ( $spalte[$zeilennummer], $zelle );
395   $spalte[$zeilennummer]++;
396 }
397
398 my $leer = 0;
399
400 sub zahlodertext {
401   my ( $layer, $zeile ) = @_ ;
402   $zeileinhalt[$layer][$spalteinhalt[$layer]] = suchezahl($zeile);
403   $spalteinhalt[$layer]++;
404 }
405
406 sub suchezahl {
407   my ($zeile) = @_ ;
408
409   if ( $zeile =~ /([0-9\.]+)/ ) { return 1; }
410   else { return 0; }
411 }
412
413 sub tabellenausrichtung {
414   my ($string) = @_ ;
415
416   my ( $vor, $nach, $zahler, $hilfsstring, $neuerstring, $i ) ;
417   $vor      = "";
418   $hilfsstring = $string;
419   $zahler   = 1;
420   $neuerstring = "";
421   while ( $hilfsstring ne "" ) {
422     if ( $hilfsstring =~ /^(^[^,]*) ,(.*$)/i ) {
423       if ( $zahler <= getspalten() - 1 ) {
424         $vor      = $1;
425         $nach     = $2;
426         $hilfsstring = $2;
427         $zahler++;
428       }
429       else {
430         $vor      = $1;
431         $nach     = "";
432         $hilfsstring = "";
433       }
434     }
435     else {
436       $vor      = $nach;

```

```

437     $hilfsstring = "";
438 }
439
440 $neuerstring = $neuerstring . $vor;
441
442 }
443 $spaltenausrichtung = $zahler + 1;
444
445 $neuerstring =~ tr//,//d;
446 setspaltendef($neuerstring);
447 $tabellenausrichtung = 1;
448
449 }
450
451 ## Eine Tabelle wird eingelesen
452
453 sub tabelle {
454     my ( $layer, $string ) = @_ ;
455
456     # Die Zeilenzahl fuer eine Tabelle wird bestimmt
457     setzeilen( $zeilenzahl + 1 );
458
459     my ( $vor, $nach, $zahler );
460     $zahler = 0;
461
462     while ( $string ne "" ) {
463         if ( $string =~ /^(^[^;]*);(.*)$/i ) {
464             $vor = $1;
465             $string = $2;
466             $nach = $2;
467         }
468         else {
469             $vor = $nach;
470             $string = "";
471         }
472
473         $vor =~ s/\s+$/ /g;
474         $vor =~ s/^\s+//g;
475         $vor =~ tr//,//d; # Fuer Rechnungen wird das Komma in einen Punkt verwandelt
476
477         if ( istzahl($vor) == 1 ) {
478             if ( $vor =~ /^(+[+])([0-9]+)$/i ) { $vor = $2; }
479             elsif ( $vor =~ /^(+)([0-9]+)$/i ) { $vor = $2; }
480             elsif ( $vor =~ /^([-][0-9]+)$/i ) { $vor = $2; }
481             elsif ( $vor =~ /^(+)([0-9]+)$/i ) { $vor = -$2; }
482             elsif ( $vor =~ /^([-][0-9]+)$/i ) { $vor = -$2; }
483         }
484         else { $vor = $vor; }
485
486         zahlodertext( getzeilen(), $vor );
487
488         # Die Nachkommastellen der Tabelle werden gesetzt
489         if ( $kommagesetzt == 0 ) { $vor = $vor; }
490         else {
491             if ( istzahl($vor) == 1 ) {
492                 $vor = nachkommastellen( $vor, $nachkommastelle );
493             }
494             else { $vor = $vor; }
495         }
496
497         schreibeintabelle( getzeilen(), $vor );

```

```
498     $zahler++;
499 }
500
501 setspalten($zahler);
502 }
503
504 my $zeiletabelle = 0;
505
506 sub schreibetabelle {
507
508     my ($layer, $j, $i, $inhalt );
509
510     # Zeilen und Spalten fuer eine Matrix werden bestimmt
511     matrix();
512
513     if ( ( $matrixzeile[ $matrixzaehler - 1 ] == 1 )
514         &&( $matrixspalte[ $matrixzaehler - 1 ] == 1 ) )
515     {
516         ausgabe( $zeile[ $matrixstart[ $matrixzaehler - 1 ] ][0] );
517     }
518     else {
519
520         my $tabart = "1";
521         if ( $tabellenart eq "tabularx" ) { $tabart = "X"; }
522
523         # Diese Funktion wird nur aufgerufen, wenn neue Zeilen hinzu gekommen sind
524         if ( $zeiletabelle != getzeilen() ) {
525
526             # Es wird die Zahl der Spalten bestimmt und wenn die Spaltendefinition nicht gesetzt ist,
527             # werden alle Spalten linksbuendig gesetzt
528             if ( $tabellenausrichtung == 0 ) {
529                 $inhalt = getspaltendef();
530                 for ( $i = 0 ; $i < getspalten() ; $i++ ) {
531                     if ( getlinien() == 1 ) { $inhalt = $inhalt . "|" . $tabart; }
532                     else { $inhalt = $inhalt . $tabart; }
533                 }
534
535                 if ( getlinien() == 1 ) { $inhalt = $inhalt . "|"; }
536                 else { $inhalt = $inhalt; }
537
538             }
539             else {
540                 $inhalt = getspaltendef();
541
542                 if ( $spaltenausrichtung < getspalten() ) {
543                     for ( $i = $spaltenausrichtung ; $i <= getspalten() ; $i++ ) {
544                         if ( getlinien() == 1 ) { $inhalt = $inhalt . "|" . $tabart; }
545                         else { $inhalt = $inhalt . $tabart; }
546                     }
547                 }
548             }
549             if ( getlinien() == 1 ) { $inhalt = $inhalt . "|"; }
550             else { $inhalt = $inhalt . "" ; }
551
552             setspaltendef($inhalt);
553
554             $inhalt = "";
555
556             # Der Tabellenkopf wird gesetzt
557             tabellenkopf( 10, getspaltendef() );
558
```

```

559 # Die Inhalte einer Spalte werden alle so ausgerichtet,
560 # dass die & Zeichen alle untereinander stehen
561 for ( $layer = $zeiletabelle + 1 ; $layer <= getzeilen() ; $layer++ ) {
562
563     for ( $i = 0 ; $i < getspalten() ; $i++ ) {
564
565         if ( $i < $spalte[$layer] ) {
566             if ( $i < 1 ) { $inhalt = $inhalt . $zeile[$layer][$i]; }
567
568             for (
569                 $j = 0 ;
570                 $j < $lange[$i] - ( length( $zeile[$layer][$i] ) - 1 ) ;
571                 $j++
572             )
573             {
574                 $inhalt = $inhalt . " ";
575             }
576
577             if ( $i >= 1 ) {
578                 $inhalt = $inhalt . $zeile[$layer][$i];
579
580                 # Bei Zahlen wird der Punkt durch ein Komma ersetzt
581                 $inhalt =~ tr/./,/d;
582             }
583             if ( $i == getspalten() - 1 ) {
584                 if ( getlinien() == 1 ) { $inhalt = $inhalt . " \\\line"; }
585                 else { $inhalt = $inhalt . " \\\line"; }
586             }
587
588             else { $inhalt = $inhalt . " &"; }
589
590         }
591     else {
592         ausgabe("i" . $layer );
593         if ( defined $zeile[$layer][$i] ) {
594             for (
595                 $j = 0 ;
596                 $j < $lange[$i] - ( length( $zeile[$layer][$i] ) - 1 ) ;
597                 $j++
598             )
599             {
600                 $inhalt = $inhalt . " ";
601             }
602         }
603
604         if ( $i == getspalten() - 1 ) {
605             if ( getlinien() == 1 ) { $inhalt = $inhalt . " \\\line"; }
606             else { $inhalt = $inhalt . " \\\line"; }
607         }
608         else { $inhalt = $inhalt . " &"; }
609
610     }
611 }
612 ausgabe($inhalt);
613 $inhalt = "";
614 }
615 tabellenfuss( 10, gettabellenfuss() );
616 }
617
618 # Die Werte einer Tabelle werden wieder zurueckgesetzt fuer die naechste
619 spaltenbreiterueck();

```

```

620   setspalten(0);
621   setspaltendef("");
622
623   # Der Tabellenfuss wird gesetzt
624
625   }
626   $zeiletabelle = getzeilen();
627
628 }
629
630 sub liste {
631   my ( $element1, $element2, $reiheoderzeile ) = @_ ;
632
633   my ( $i, $inhalt );
634   latex( 11,
635     " listplot layer=90 farbe=red linie=1,solid parameter=\\"plotstyle=curve\\" );
636
637   if ( $reiheoderzeile eq "h" ) {
638
639     if ( $element1 > getzeilen() ) { $element1 = getzeilen() - 1; }
640     if ( $element2 > getzeilen() ) { $element2 = getzeilen() - 1; }
641
642     for ( $i = 1 ; $i < getspalten() ; $i++ ) {
643       if ( ( suchezahl( $zeile[$element1][$i] ) == 1 )
644         || ( suchezahl( $zeile[$element2][$i] ) == 1 ) )
645       {
646         $zeile[$element1][$i] =~ tr /, / . / d;
647         $zeile[$element2][$i] =~ tr /, / . / d;
648         $inhalt = $zeile[$element1][$i] . ", " . $zeile[$element2][$i];
649         latex( 11, $inhalt );
650       }
651     }
652   }
653 }
654 else {
655   if ( $element1 > getspalten() ) { $element1 = getspalten() - 1; }
656   if ( $element2 > getspalten() ) { $element2 = getspalten() - 1; }
657
658   for ( $i = 1 ; $i <= getzeilen() ; $i++ ) {
659     if ( ( suchezahl( $zeile[$i][$element1] ) == 1 )
660       || ( suchezahl( $zeile[$i][$element2] ) == 1 ) )
661     {
662       if ( defined $zeile[$i][$element1] ) {
663         $zeile[$i][$element1] =~ tr /, / . / d;
664       }
665
666       if ( defined $zeile[$i][$element2] ) {
667         $zeile[$i][$element2] =~ tr /, / . / d;
668       }
669       if (
670         ( defined $zeile[$i][$element2] )
671         && ( defined $zeile[$i][$element1] ) ) {
672         $inhalt =
673           $zeile[$i][$element1] . ", " . $zeile[$i][$element2];
674         latex( 11, $inhalt );
675       };
676     }
677   }
678 }
679
680   latex( 11, "ende" );

```

```

681 }
682 }
683
684 ## Matrixoperationen
685
686 sub matrix {
687     $matrixzeile[$matrixzaehler] =
688         getzeilen() - $matrixzeile[ $matrixzaehler - 1 ];
689     $matrixspalte[$matrixzaehler] = getspalten();
690     $matrixstart[$matrixzaehler] =
691         getzeilen() - $matrixzeile[$matrixzaehler] + 1;
692     $matrixzaehler++;
693 }
694
695 # Matrixaddition
696
697 sub matrixaddition {
698     my ( $matrix1, $matrix2, $ausgabe ) = @_ ;
699
700     my ( $i, $j, $text, $text1, $text2 );
701     $text = $text1 = $text2 = "";
702     my $layer = 0;
703     $j = 0;
704     my $ergebnis;
705
706     if ( ( $matrixzeile[$matrix1] == $matrixzeile[$matrix2] )
707         && ( $matrixspalte[$matrix1] == $matrixspalte[$matrix2] ) )
708     {
709
710         if ( $ausgabe == 0 ) {
711             for (
712                 $layer = $matrixstart[$matrix1] ;
713                 $layer <= ( $matrixstart[$matrix1] + $matrixzeile[$matrix1] - 1 ) ;
714                 $layer++
715             )
716             {
717
718                 for ( $i = 0 ; $i < $matrixspalte[$matrix1] ; $i++ ) {
719                     $text1 = $zeile[ $layer ][ $i ];
720                     if ( $zeile[ $matrixstart[$matrix2] + $j ][ $i ] < 0 ) {
721                         $text2 = $zeile[ $matrixstart[$matrix2] + $j ][ $i ];
722                     }
723                     else {
724                         $text2 = "+" . $zeile[ $matrixstart[$matrix2] + $j ][ $i ];
725                     }
726                     $text = $text . $text1 . $text2 . ";";
727
728                 }
729                 $j++;
730
731                 tabelle( 10, $text );
732                 $text = "";
733             }
734             $j = 0;
735             schreibetabelle ();
736         }
737         else {
738             for (
739                 $layer = $matrixstart[$matrix1] ;
740                 $layer <= ( $matrixstart[$matrix1] + $matrixzeile[$matrix1] - 1 ) ;
741                 $layer++

```

```

742     )
743     {
744     $text = "";
745
746     for ( $i = 0 ; $i < $matrixspalte[$matrix1] ; $i++ ) {
747     if ( ( istzahl( $zeile[$layer][$i] ) == 1 )
748     && ( istzahl( $zeile[ $matrixstart[$matrix2] + $j ][ $i ] ) == 1 ) )
749     {
750     $ergebnis =
751     $zeile[$layer][$i] + $zeile[ $matrixstart[$matrix2] + $j ][ $i ];
752     $text = $text . $ergebnis . ";";
753
754     }
755     else {
756     $text = $text
757     . $zeile[$layer][$i] . " + "
758     . $zeile[ $matrixstart[$matrix2] + $j ][ $i ] . ";";
759     }
760     }
761     $j++;
762
763     tabelle( 10, $text );
764     }
765     schreibetabelle ();
766     }
767     }
768 }
769
770 #Subtraktion von Matrizen
771
772 sub matrixsubtraktion {
773     my ( $matrix1, $matrix2, $ausgabe ) = @_;
774
775     my ( $i, $j, $text1, $text2, $text );
776
777     my $layer = 0;
778     $j = 0;
779     my $ergebnis;
780     $text = $text1 = $text2 = "";
781
782     if ( ( $matrixzeile[$matrix1] == $matrixzeile[$matrix2] )
783     && ( $matrixspalte[$matrix1] == $matrixspalte[$matrix2] ) )
784     {
785
786     if ( $ausgabe == 0 ) {
787
788     for (
789     $layer = $matrixstart[$matrix1] ;
790     $layer <= ( $matrixstart[$matrix1] + $matrixzeile[$matrix1] - 1 ) ;
791     $layer++
792     )
793     {
794
795     for ( $i = 0 ; $i < $matrixspalte[$matrix1] ; $i++ ) {
796
797     $text1 = $zeile[$layer][$i] ;
798     if ( istzahl( $zeile[ $matrixstart[$matrix2] + $j ][ $i ] ) == 1 ) {
799
800     if ( $zeile[ $matrixstart[$matrix2] + $j ][ $i ] < 0 ) {
801     $text2 = "+" . abs( $zeile[ $matrixstart[$matrix2] + $j ][ $i ] );
802     }

```

```

803         else { $text2 = "-" . $zeile[ $matrixstart[$matrix2] + $j ][ $i ]; }
804     }
805     }
806     else {
807         $text2 = "-" . $zeile[ $matrixstart[$matrix2] + $j ][ $i ];
808     }
809     $text = $text . $text1 . $text2 . ";";
810 }
811 $j++;
812
813 tabelle( 10, $text );
814 $text = "";
815 }
816 $j = 0;
817 schreibetabelle ();
818 }
819 else {
820
821     for (
822         $layer = $matrixstart[$matrix1] ;
823         $layer <= ( $matrixstart[$matrix1] + $matrixzeile[$matrix1] - 1 ) ;
824         $layer++
825     )
826     {
827
828         for ( $i = 0 ; $i < $matrixspalte[$matrix1] ; $i++ ) {
829             if ( ( istzahl( $zeile[$layer][ $i ] ) == 1 )
830                 && ( istzahl( $zeile[ $matrixstart[$matrix2] + $j ][ $i ] ) == 1 ) )
831             {
832                 $ergebnis =
833                     $zeile[$layer][ $i ] - $zeile[ $matrixstart[$matrix2] + $j ][ $i ];
834                 $text = $text . $ergebnis . ";";
835             }
836             else {
837                 $text = $text
838                     . $zeile[$layer][ $i ] . "-"
839                     . $zeile[ $matrixstart[$matrix2] + $j ][ $i ] . ";";
840             }
841         }
842         $j++;
843
844         tabelle( 10, $text );
845         $text = "";
846     }
847     }
848     schreibetabelle ();
849 }
850 }
851 }
852
853 # Matrizenmultiplikation
854 sub matrixmultiplikation {
855     my ( $matrix1, $matrix2, $ausgabe ) = @_;
856     my ( $i, $ii, $text1, $text2 );
857     my $text = "";
858     my $layer = 0;
859     my $ergebnis;
860     my $semikolon = 1;
861     my $skalar = 0;
862     my $hilfsmatrix = 0;
863

```

```
864 # Hat eine Matrix nur eine Zeile und Spalte, so ist dies ein Skalar.
865
866 if ( ( $matrixzeile[$matrix1] == 1 ) && ( $matrixspalte[$matrix1] == 1 ) ) {
867     $skalar      = $zeile[ $matrixstart[$matrix1] ][0];
868     $hilfsmatrix = $matrix2;
869 }
870
871 if ( ( $matrixzeile[$matrix2] == 1 ) && ( $matrixspalte[$matrix2] == 1 ) ) {
872     $skalar      = $zeile[ $matrixstart[$matrix2] ][0];
873     $hilfsmatrix = $matrix1;
874 }
875
876 # Der Rang wird ueberprueft
877 if ( $matrixzeile[$matrix2] == $matrixspalte[$matrix1] ) {
878
879     if ( $ausgabe == 0 ) {
880         for (
881             $layer = $matrixstart[$matrix1] ;
882             $layer <= ( $matrixstart[$matrix1] + $matrixzeile[$matrix1] - 1 ) ;
883             $layer++
884         )
885         {
886             $text      = "";
887             $ergebnis = 0;
888
889             for ( $i = 0 ; $i < $matrixspalte[$matrix2] ; $i++ ) {
890                 for ( $ii = 0 ; $ii < $matrixspalte[$matrix1] ; $ii++ ) {
891                     if ( istzahl( $zeile[$layer][$ii] ) == 1 ) {
892                         if ( $zeile[$layer][$ii] < 0 ) {
893                             $text1 = "(" . $zeile[$layer][$ii] . ")";
894                         }
895                         else { $text1 = $zeile[$layer][$ii] ; }
896                     }
897                     else { $text1 = $zeile[$layer][$ii] ; }
898                     if ( istzahl( $zeile[ $matrixstart[$matrix2] + $ii ][ $i ] ) == 1 )
899                     {
900                         if ( $zeile[ $matrixstart[$matrix2] + $ii ][ $i ] < 0 ) {
901                             $text2 =
902                                 "(" . $zeile[ $matrixstart[$matrix2] + $ii ][ $i ] . ")";
903                         }
904                         else {
905                             $text2 = $zeile[ $matrixstart[$matrix2] + $ii ][ $i ];
906                         }
907                     }
908                     else {
909                         $text2 = $zeile[ $matrixstart[$matrix2] + $ii ][ $i ];
910                     }
911
912                     if ( $ii == $matrixspalte[$matrix1] - 1 ) {
913                         $text = $text . $text1 . "\\cdot" . $text2 . ";";
914                     }
915                     else { $text = $text . $text1 . "\\cdot" . $text2 . "+"; }
916                 }
917             }
918
919             tabelle( 10, $text );
920             $text = "";
921         }
922     }
923
924     schreibetabelle;
```

```

925 }
926
927 else {
928     for (
929         $layer = $matrixstart[$matrix1] ;
930         $layer <= ( $matrixstart[$matrix1] + $matrixzeile[$matrix1] - 1 ) ;
931         $layer++
932     )
933     {
934         $text = "";
935         $ergebnis = 0;
936
937         for ( $i = 0 ; $i < $matrixspalte[$matrix2] ; $i++ ) {
938             for ( $ii = 0 ; $ii < $matrixspalte[$matrix1] ; $ii++ ) {
939
940                 if ( istzahl( $zeile[$layer][$ii] ) == 1 ) {
941                     if ( $zeile[$layer][$ii] < 0 ) {
942                         $text1 = "(" . $zeile[$layer][$ii] . ")";
943                     }
944                     else { $text1 = $zeile[$layer][$ii]; }
945                 }
946                 else { $text1 = $zeile[$layer][$ii]; }
947                 if ( istzahl( $zeile[ $matrixstart[$matrix2] + $ii ][ $i ] ) == 1 )
948                 {
949                     if ( $zeile[ $matrixstart[$matrix2] + $ii ][ $i ] < 0 ) {
950                         $text2 =
951                             "(" . $zeile[ $matrixstart[$matrix2] + $ii ][ $i ] . ")";
952                     }
953                     else { $text2 = $zeile[ $matrixstart[$matrix2] + $ii ][ $i ]; }
954                 }
955                 else { $text2 = $zeile[ $matrixstart[$matrix2] + $ii ][ $i ]; }
956
957                 if ( $ii == $matrixspalte[$matrix1] - 1 ) {
958
959                     if (
960                         ( istzahl( $zeile[$layer][$ii] ) == 1 )
961                         && (
962                             istzahl( $zeile[ $matrixstart[$matrix2] + $ii ][ $i ] ) == 1 )
963                         )
964                     {
965                         $ergebnis = $ergebnis + $zeile[$layer][$ii] *
966                             $zeile[ $matrixstart[$matrix2] + $ii ][ $i ];
967                         $text = $text . "+" . $ergebnis . ";";
968                         $ergebnis = 0;
969                     }
970                     else {
971
972                         $text = $text
973                             . $zeile[$layer][$ii] . "\\cdot "
974                             . $zeile[ $matrixstart[$matrix2] + $ii ][ $i ] . ";";
975                         $semikolon = 1;
976                     }
977                 }
978
979                 else {
980                     if (
981                         ( istzahl( $zeile[$layer][$ii] ) == 1 )
982                         && (
983                             istzahl( $zeile[ $matrixstart[$matrix2] + $ii ][ $i ] ) == 1 )
984                         )
985                     {

```

```
986
987     $ergebnis = $ergebnis + $zeile[$layer][$ii] *
988     $zeile[ $matrixstart[$matrix2] + $ii ][ $i ];
989 }
990 else {
991     $text = $text
992     . $zeile[$layer][$ii] . "\\cdot "
993     . $zeile[ $matrixstart[$matrix2] + $ii ][ $i ];
994 }
995 }
996 }
997
998 }
999 $semikolon = 1;
1000
1001 tabelle( 10, $text );
1002 $text = "";
1003 }
1004 schreibetabelle;
1005 }
1006 }
1007 elseif ( $hilfsmatrix != 0 ) {
1008
1009     if ( $skalar < 0 ) { $text1 = "(" . $skalar . ")"; }
1010     else { $text1 = $skalar; }
1011
1012     if ( $ausgabe == 0 ) {
1013         for (
1014             $layer = $matrixstart[$hilfsmatrix] ;
1015             $layer <=
1016             ( $matrixstart[$hilfsmatrix] + $matrixzeile[$hilfsmatrix] - 1 ) ;
1017             $layer++
1018         )
1019         {
1020
1021             for ( $ii = 0 ; $ii < $matrixspalte[$hilfsmatrix] ; $ii++ ) {
1022                 if ( istzahl( $zeile[$layer][$ii] ) == 1 ) {
1023                     if ( $zeile[$layer][$ii] < 0 ) {
1024                         $text2 = "(" . $zeile[$layer][$ii] . ")";
1025                     }
1026                     else {
1027                         $text2 = $zeile[$layer][$ii];
1028                     }
1029                 }
1030                 else { $text2 = $zeile[$layer][$ii]; }
1031                 $text = $text . $text1 . "\\cdot " . $text2 . ";";
1032             }
1033             tabelle( 10, $text );
1034             $text = "";
1035         }
1036     }
1037     schreibetabelle;
1038 }
1039 }
1040 }
1041 }
1042 else {
1043     for (
1044         $layer = $matrixstart[$hilfsmatrix] ;
1045         $layer <=
1046         ( $matrixstart[$hilfsmatrix] + $matrixzeile[$hilfsmatrix] - 1 ) ;
```

```

1047     $layer++
1048   )
1049   {
1050     for ( $ii = 0 ; $ii < $matrixspalte[$hilfsmatrix] ; $ii++ ) {
1051       if ( istzahl( $zeile[$layer][$ii] ) == 1 ) {
1052         $text2 = $skalar * $zeile[$layer][$ii] ;
1053
1054       }
1055       else { $text2 = $text1 . "\\cdot " . $zeile[$layer][$ii] ; }
1056       $text = $text . $text2 . " ;"
1057
1058     }
1059     tabelle( 10, $text );
1060     $text = "";
1061
1062   }
1063   schreibetabelle ;
1064 }
1065 }
1066 }
1067
1068 ## Transponierte Matrix
1069
1070 sub transponiert {
1071   my ($matrix1) = @_ ;
1072   my ( $text, $i, $layer ) ;
1073
1074   $text = "" ;
1075   for ( $i = 0 ; $i < $matrixspalte[$matrix1] ; $i++ ) {
1076     for (
1077       $layer = $matrixstart[$matrix1] ;
1078       $layer <= ( $matrixstart[$matrix1] + $matrixzeile[$matrix1] - 1 ) ;
1079       $layer++
1080     )
1081     {
1082       $text = $text . $zeile[$layer][$i] . " ;" ;
1083     }
1084
1085     tabelle( 10, $text );
1086     $text = "" ;
1087   }
1088   schreibetabelle () ;
1089 }
1090
1091 1 ;

```

hilfsfunktionen.pm

```

1 #####
2 # Modul fuer Hilfsfunktionen
3 #
4 # VERSION 1
5 # Letzte Aenderung: 26.02.2009
6 #####
7 use POSIX qw /floor ceil/ ;
8 use Math::Trig ;
9 use strict ;
10 use warnings ;
11
12 ### Der Funktionswert wird berechnet
13

```

```
14 sub calcfunk {
15   my ( $funktionsausdruck, $variable, $wert ) = @_;
16   my $y = $funktionsausdruck;
17   $y =~ s/\^\/**/g;
18   $y =~ s/\b$variable\b/($wert)/g;
19   $y = eval($y);
20   return $y;
21 }
22
23 ## Wie viele Nachkommastellen sollen angegeben werden?
24
25 sub nachkommastellen {
26   my ( $wert, $nachkommastelle ) = @_;
27   my $nachkommastellentext = "%0." . $nachkommastelle . "f";
28   if ( $nachkommastelle >= 0 ) {
29     $wert = sprintf( $nachkommastellentext, $wert );
30   }
31   else { $wert = $wert; }
32   return $wert;
33 }
34
35 sub abrunden {
36   my ( $wert, $schritt ) = @_;
37   return $schritt * floor( $wert / $schritt );
38 }
39
40 sub aufrunden {
41   my ( $wert, $schritt ) = @_;
42   return $schritt * ceil( $wert / $schritt );
43 }
44
45 ### Es wird eine Datei erzeugt
46
47 sub latex {
48   my ( $layer, $pstrickstext ) = @_;
49   schreibeinarray( $layer, sprintf( "%s", $pstrickstext ) );
50 }
51
52 sub ausgabe {
53   my ( $pstrickstext ) = @_;
54   latex( 10, $pstrickstext );
55 }
56
57 sub kommentar {
58   my ( $layer, $text ) = @_;
59   schreibeinarray( $layer, sprintf( "%%%" $text ) );
60 }
61
62 sub usepackage {
63   my ( $parameter, $befehl ) = @_;
64   if ( $parameter eq "" ) {
65     schreibeinarray( 2, sprintf( "\\usepackage{%s}", $befehl ) );
66   }
67   else {
68     schreibeinarray( 2,
69       sprintf( "\\usepackage[%s]{%s}", $parameter, $befehl ) );
70   }
71 }
72
73 sub max {
74   my $a = shift;
```

```

75 my $b = shift;
76 return $a > $b ? $a : $b;
77 }
78
79 sub min {
80 my $a = shift;
81 my $b = shift;
82 return $a < $b ? $a : $b;
83 }
84
85 # ist eine Ausgabe eine Zahl ode nicht
86 sub istzahl {
87 my ($zeile) = @_ ;
88
89 if ( $zeile =~ /([a-zA-Z\.\.]+)/ ) { return 0; }
90 else { return 1; }
91 }
92 1;

```

### 3.2 Eine Beispieleingabedatei

tabelle-befehle.csv

```

1  ### Die Tabellenumgebung wird definiert
2  ### Ist tabularx gesetzt,
3  ### so steht der hintere Befehl fuer die Breite der Tabelle
4  tabellenkopf="longtable","0.5"
5
6  ### Wie soll die Tabelle gesetzt werden?
7  ### Die Spaltendefinitionen sind mit Komma getrennt.
8  ### Stehen hier weniger Eintraege, als die Tabelle breit ist, wird
9  ### der Rest mit 1 aufgefuellt.
10  ausrichtung="r,r,r,r"
11  nachkommastellen=2
12
13  ### Soll der Dokumentenrumpf mit gebaut werden, wenn kopf gesetzt ist, dann wird der Rumpf mitgebaut
14  kopf
15
16  ### Mathemodus wird gesetzt
17  math
18
19  ### Linien werden gesetzt
20  #linien
21
22  ### Eine Funktionstabelle wird gebaut.
23  ### In p steht an erster Stelle die Schrittweite
24  ### an zweiter Stelle ob die Tabelle vertikal oder
25  ### horizontal ausgerichtet werden soll.
26  ### An dritter Stelle steht, wie viele Spalten bzw. Zeilen
27  ### angelegt werden sollen.
28
29  #graphtabelle x=5,20 p=0.5,v,5 h(t)=x^2+3
30
31  ### Eine Liste wird erstellt, um die Daten als Kurve zeichnen zu koennen.
32  ### Ist das letzte Argument ein v, so stehen die ersten beiden Argumente
33  ### fuer die Spaltenelemente.
34  ### Ist das letzte Argument ein h, so stehen die ersten beiden Argumente
35  ### fuer die Zeilenelemente.
36

```

```
37 #liste =2,8,v
38
39 ### Eine Tabelle fuer die lokale Aenderungsrage wird gebaut.
40 ### In p steht an erster Stelle, wie viele Werte berechnet werden sollen.
41 ### An zweiter Stelle ob die Tabelle vertikal oder
42 ### horizontal ausgerichtet werden soll.
43 ### An dritter Stelle steht, wie viele Spalten bzw. Zeilen
44 ### angelegt werden sollen.
45
46 #tabellelokal x=3 p=10,v,5 h(t)=x^2
47
48 ### Eine csv-Datei wird eingelesen
49 ### Trennzeichen ist das Semikolon
50
51 1; 2; 3; 4.4
52 5; 6; 7; 8
53 a;b;c;d
54 aa;bb;cc;dd
55 aaa;bbb;ccc
56 aaaa;bbbb;cccc dd; d
57 test ;
58 2.4; 5; 4 ; 7; 5; 4;
59
60 #kopf
61 #tabellenkopf="pmatrix",""
62 #nachkommastellen=2
63 #linien
64
65 ### Matrixoperationen
66 #matrixadd 1 2
67 #matrixsub 1 2
68 #matrixmul 1 2
69 #transponiert 2
```

---

## Literatur

- [BIGALKE und KÖHLER, 2007] BIGALKE, ANTON und N. KÖHLER (2007). Mathematik Sekundarstufe II - Allgemeine Ausgabe: Mathematik, Sekundarstufe II, Allgemeine Ausgabe, Bd.1 : Analysis. Cornelsen.
- [DENISOV, 2009] DENISOV, ALEX A. (2009). LaTable. <http://ftp.ktug.or.kr/tex-archive/help/Catalogue/entries/latable.html>.
- [ERBRECHT, 2003] ERBRECHT, RÜDIGER (2003). Das große Tafelwerk interaktiv. Formelsammlung für die Sekundarstufen I und II. Westliche Bundesländer: Das große Tafelwerk interaktiv, inkl. CD-ROM: ... Chemie, Biologie - für das Abitur empfohlen.. Cornelsen, 1. Aufl.
- [GLYNN, 1995] GLYNN, JERRY (1995). Mathematik entdecken mit DERIVE, von der Algebra bis zur Differentialrechnung. Birkhäuser Verlag.
- [KRIENKE, 2002] KRIENKE, RAINER (2002). Programmieren in Perl.. Hanser Fachbuchverlag, 2., erw Aufl.
- [LINGNAU, 2007] LINGNAU, ANSELM (2007). LaTeX Hacks. Tipps und Techniken für professionellen Textsatz. O'Reilly, 1 Aufl.
- [VOSS, 2008] VOSS, HERBERT (2008). Tabellen mit LaTeX. Lehmanns Media-Lob.de.
- [WESTERMANN, 2005] WESTERMANN, THOMAS (2005). Mathematische Probleme lösen mit Maple. Ein Kurzeinstieg. Springer, Berlin, 2., neu bearb. u. erw Aufl.