BUNDESOBERSTUFENREALGYMNASIUM PERG

DIRNBERGERSTRASSE 43; 4320 PERG

wxMAXIMA

Ein Computer Algebra System in der AHS-Oberstufe.

FACHBEREICHSARBEIT AUS MATHEMATIK

eingereicht bei Mag. Johann Angsüsser

von Manuel Resch

Bad Zell, am 22.Februar.2010

Inhalt

1 Vorwort

Das erste Mal kam die Idee, eine Fachbereichsarbeit in Mathematik zu schreiben, Ende der sechsten Klasse auf. Durch die Motivation von Professor Angsüsser und aus meiner Vorliebe für Mathematik heraus beschloss ich bald ein geeignetes Thema für meine Arbeit zu suchen.

Mein Dank gilt nun Professor Angsüsser, der mir vielfältige Themen aus der Welt der Mathematik vorgeschlagen hat. Nach knapp einem Jahr Themensuche war dann am Ende der siebten Klasse klar, dass ich mich mit dem Computer Algebra System *wxMaxima* beschäftigen werde. Der Hauptgrund für dieses Thema war, dass man im Mathematikstudium hauptsächlich mit solchen Programmen zu tun hat und ich mich so schon etwas auf meine Laufbahn nach dem Gymnasium vorbereiten konnte.

Die konkrete Zielsetzung kam auch von Professor Angsüsser. Da in unserer Schule mit Anfang 2010 die Umstellung auf Laptopklassen begann, benötigte man nun ein Programm, das den graphischen Rechner *Voyage 200*, der bis dahin im Mathematikunterricht zum Einsatz kam, ersetzen konnte. Maxima war hierfür eine Möglichkeit.

Im Internet findet man zwar viele Einführungen in das Programm *wxMaxima*, aber ich wollte eine Anleitung schreiben, die besser auf den Mathematikunterricht in der AHS-Oberstufe zugeschnitten ist. Deshalb ist meine Arbeit auch nach Schulstufen und nicht, wie sonst üblich, nach Themengebieten aufgebaut. Ich habe versucht alle notwendigen Befehle zusammenzufassen und anhand von Beispielen aus dem Unterricht zu veranschaulichen.

Vielleicht kann meine FBA dazu beitragen, dass sich wxMaxima an unserer Schule durchsetzt.

Bad Zell, am 22.Februar.2010 *Manuel Resch*

2 Einführung

Unter *wxMaxima* versteht man eigentlich die auf *wxWidgets* basierende graphische Benutzeroberfläche des Computer Algebra Systems *Maxima*. Das heißt, das eigentliche Programm, um das es hier geht und mit dem man verschiedenste mathematische Operationen ausführen kann, ist *Maxima*. Da die komplette Bedienung aber über *wxMaxima* erfolgt verwende ich diese beiden Begriffe in meiner Arbeit als Synonyme.

2.1 Geschichte

Die Geschichte von *Maxima* begann in den späten Sechzigern, als am Massachusetts Institute of Technology (MIT) ein Computer Algebra System (CAS) unter dem Namen *Macsyma* entwickelt wurde. Dieses legendäre Mathematikprogramm war damals wirklich revolutionär und diente etwa als Vorbild für das heute weit verbreitete CAS *Mathematica*.

Macsyma wurde mit Unterstützung des amerikanischen Energieministeriums (Department of Energy) noch bis 1999 weitergeführt. Das Projekt *Maxima* wurde jedoch schon 1982 von *Macsyma* abgespalten und blieb so bis heute bestehen. Eine wichtige Persönlichkeit in der Entwicklung war William Schelter von der Universität Texas. Ihm ist es auch zu verdanken, dass *Maxima* 1998 zum Open-Source-Programm wurde, als der Quellcode unter einer GNU General Public Licence[1] veröffentlicht wurde. Seit dem Tod von Schelter 2001 wird Maxima von unabhängigen Entwicklern ständig weiterentwickelt. So entstanden die graphische Benutzeroberfläche und eine Plot-Funktion auf Basis von *gnuplot*.
Aus dem Quellcode von *Maxima* entstanden auch andere verwandte Programme, wie *Imaxima* oder *TeXmacs*, die man mit *wxMaxima* nicht verwechseln sollte.

[1] Nähere Informationen zur Rechtssituation unter http://www.fsf.org/licensing/licenses/gpl.html (Englisch) [Stand: 06-02-2010].

3 Installation

Wie schon in der Einleitung erwähnt ist *Maxima* ein Open-Source-Programm und kann von der Internetseite http://sourceforge.net/projects/maxima/files/[2] gratis heruntergeladen werden. Aufgrund der Programmierung in CommonLisp arbeitet Maxima unter allen gängigen Betriebssystemen. Es gibt Pakete für Windows, Linux oder MacOS X und ab der Version 5.10.0b von *Maxima* ist eine graphische Oberfläche (*wxMaxima*) schon integriert.

Diese Arbeit ist mit *Maxima* 5.19 und *wxMaxima* 0.8.3 unter Windows Vista entstanden. Jetzt gibt es schon eine neuere Version von *Maxima* (5.20.1), aber die grundlegende Verwendung, die ich in dieser Arbeit beschreibe, hat sich nicht geändert.

Nach der Installation kann man die Benutzeroberfläche noch an die jeweiligen Anforderungen anpassen. Zum Beispiel sind die vom Programm verwendeten Schriftarten frei wählbar. Für ein einfaches Arbeiten mit dieser Anleitung sollte man zuerst die gleiche Oberfläche einstellen:

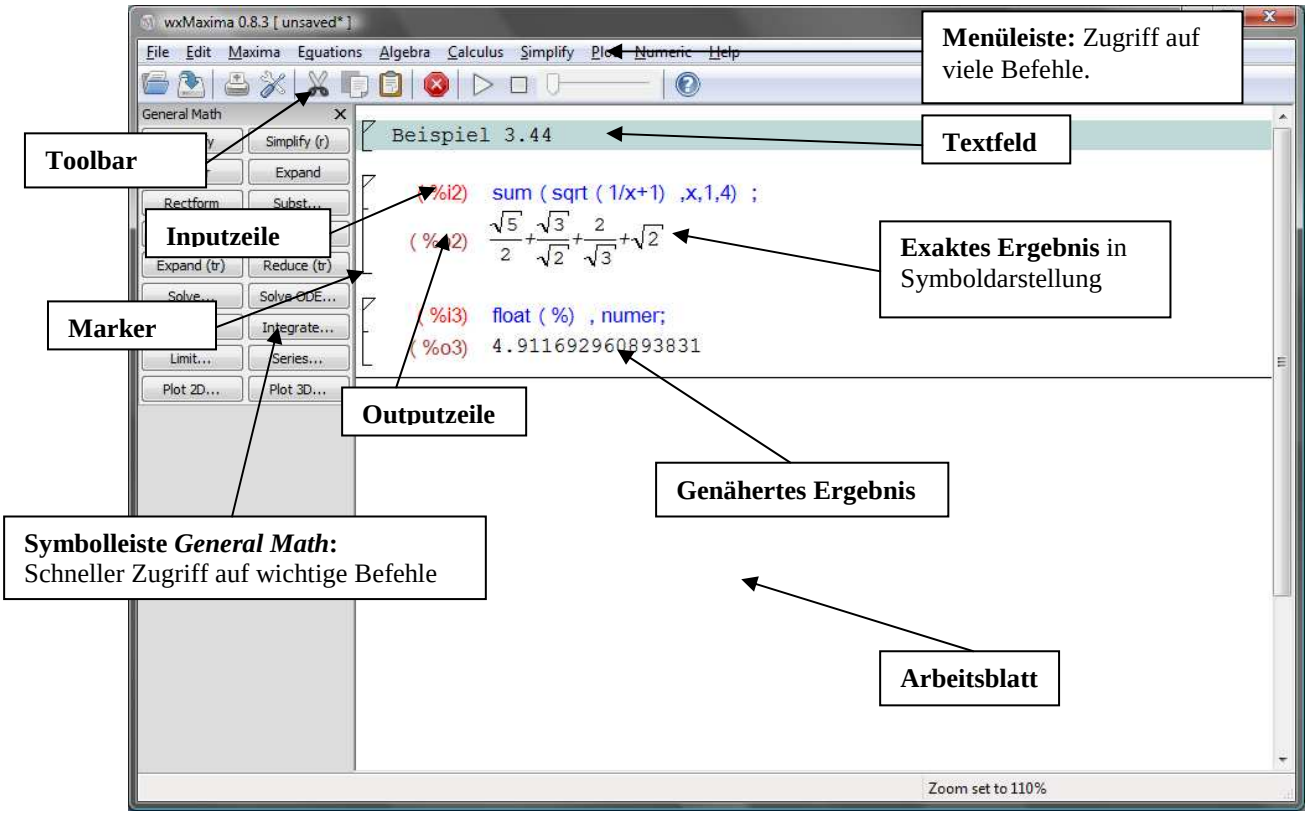
- Menüsprachen Englisch: Um mit den englischen Fachbegriffen vertraut zu werden, wird die englische Menüführung verwendet. Sollte man bei der Installation das deutsche Menü gewählt haben, kann man das unter *Bearbeiten-Einstellungen…* ändern.
- Werkzeugleisten: Über mehrere Symbolleisten kann man Befehle in *Maxima* schneller aufrufen. Die Nützlichsten sind *General Math* und *Toolbar*. Um sie anzuzeigen klickt man sie im Menü *Maxima-Panes* an. Will man diese Oberfläche speichern muss man noch die Option *Save panes layout* in *Edit-Configure…* wählen.

[2] [Stand: 2010-02-21]

4 Erste Schritte

4.1 Maxima als Taschenrechner

4.1.1 Benutzeroberfläche von wxMaxima



wxMaxima bietet im Menü *Edit-Configure-Style* die Möglichkeit die Schriftart individuell einzustellen. Die Schriftgröße kann man auch direkt mit den Tastenkürzeln *Alt+I* (größer) und *Alt+O* (kleiner) verstellen. Vor allem bei Verwendung eines Beamers (z.B. im Unterricht) empfiehlt es sich die Schrift größer einzustellen.

4.1.2 Eingabe und Löschen von Kommandos

Gibt man in Maxima einen Befehl ein, so erscheint dieser in einer Inputzeile (z.B. mit *(%i2)* bezeichnet). Einfaches Drücken der Entertaste führt nur zu einem Zeilenumbruch. Zum Evaluieren geht man so vor:

***Shift+Enter* oder *Strg+Enter* lässt Maxima die Berechnung ausführen.**

Am Ende der Inputzeile erscheint automatisch ein Semikolon ";", wenn dieser nicht schon vorher gesetzt wurde.

```
(%i5) 3*2+4/2;
      (%o5)   8
```

Für Inputzeilen, die auf diese Weise abgeschlossen werden, zeigt Maxima das Ergebnis an. Wenn man mit dem Ergebnis direkt weiterrechnet (s. 4.3), kann es aber von Vorteil sein (sehr lange) Ergebnisse zu unterdrücken. Um das zu erreichen wird die Inputzeile mit einem Dollarzeichen *$* abgeschlossen.

Abschluss der Inputzeile: Ergebnis anzeigen … ; - Ergebnis nicht anzeigen … $

Es können auch mehrere Eingaben in einer Inputzeile berechnet werden, wenn man die Eingaben nur durch Semikolons oder Dollarzeichen und Zeilenumbrüche trennt.

```
(%i2) 2*4;
      3+3$
      27/3;
      (%o2)   8
      (%o4)   9
```

Ganze Inputzeilen mit ihren Outputs löscht man am besten, indem man die Marker auf der linken Seite markiert und dann Entfernen drückt.

4.1.3 Grundrechenarten

Die Grundrechenarten werden in Maxima mit folgenden Operatoren verwendet:

Addition: +	**Subtraktion: -**	**Multiplikation: ***	**Division: /**

Natürlich verwendet auch Maxima alle Rechenregeln in gleicher Form, wie wir sie am Papier anwenden. Ohne Rechenoperator zwischen Zahl und Variable (z. B. 2x statt 2*x) gibt Maxima einen Fehler aus.

4.1.4 Klammersetzung

Maxima kennt drei Arten von Klammern:

1. ( )...werden in Rechenoperationen verwendet

```
(%i6) (3+2)*13;
(%o6)  65
```

Eine Klammerhierarchie in den Rechenoperationen gibt es in Maxima nicht. Bei der Klammersetzung ist deshalb erhöhte Vorsicht geboten, aber Maxima unterstützt den Benutzer, indem es die gerade gültigen Klammern grau hinterlegt und eine geöffnete Klammer bei der Eingabe wieder schließt.

2. [ ]...werden für Listen verwendet

Diese Klammern tauchen zum Beispiel bei der Vektorrechnung auf.

3. { }...werden als Mengenklammern verwendet[3]

4.1.5 Kommapunkt

Vorsicht ist bei Dezimalzahlen geboten, weil Maxima hier einen Punkt verwendet.

```
(%i7) 32.67*3.2;
(%o7)  104.544
```

4.1.6 Vorzeichen-Minus

Anders als bei vielen Taschenrechnern unterscheidet Maxima nicht zwischen dem Minus als Subtraktionsoperator und dem als Vorzeichen. Meist rechnet Maxima mit einem Minus als Vorzeichen zwar richtig, doch eine Klammer ist hier dennoch die sicherere Variante.
Statt:

```
(%i8) 4+-6*3;
(%o8)  -14
```

also besser:

[3] Auf Mengen gehe ich in dieser Arbeit nicht näher ein.

```
(%i9) 4+(-6)*3;
(%o9)  -14
```

4.1.7 Weitere Rechenoperatoren

Am nächsten Beispiel zeigt sich, wie die meisten Befehle in Maxima aussehen: Auf ein Kürzel (hier *sqrt* für „square root“), das immer aus Kleinbuchstaben besteht, folgt in runden Klammern der Input.

$\sqrt{x}$… sqrt(x)

```
(%i10) sqrt(169);
(%o10)  13
```

Potenzieren…^ oder **

```
(%i11) 3^3;
(%o11)  27

(%i12) 16**3;
(%o12)  4096
```

Mit Hilfe des Operators zum Potenzieren kann man auch allgemeine Wurzeln berechnen.

```
(%i13) 64^(1/4);
(%o13)  2^3/2
```

4.1.8 Näherung

Maxima versucht jedes Ergebnis exakt darzustellen. Wird aber eine Kommazahl als Ergebnis gewünscht, gibt es mehrere Möglichkeiten zum Approximieren.

1. Wenn der Input schon in Kommazahlen erfolgt (auch .0), erfolgt auch der Output als Kommazahl. 2. Durch drücken von Strg+F wird das vorherige Ergebnis approximiert. 3. Anhängen von *,numer* an den Input. 4. Über das Menü *Numeric* kann man eine Zahl in eine Gleitkommazahl (To Float) und

in eine große Gleitkommazahl (To Bigfloat) umwandeln. Mit Set Precision stellt man die Länge der großen Gleitkommazahl ein.

```
(%i14) 4/3;
       4/3.0;
       4/3,numer;
(%o14) 4/3
(%o15) 1.333333333333333
(%o16) 1.333333333333333

(%i18) fpprec: 40$
(%i19) bfloat(%pi);
(%o19) 3.141592653589793238462643383279502884197b0
```

ad 1.

ad 3.

ad 4.

Das „b“ am Ende einer *Bigfloat* steht für die Zehnerpotenz: Hier heißt b0 also $*10^0=1$.

4.1.9 Langzahlen

Berechnet man mit Maxima sehr lange Zahlen (z.B. 2^{10000}) werden in der Standardeinstellung Anfang und Ende genau, bei der Mitte aber nur die Anzahl der Stellen angezeigt. Das Darstellungsformat lässt sich so verändern:

set_display(xml)…Standardeinstellungen
set_display(ascii)…komplette Darstellung mit „/“ als Zeilenumbruch
set_display(none)…komplette Darstellung in einer Zeile

4.1.10 Betrag einer Zahl

Sollte man zum Eingeben von Formeln oder Ähnlichem einmal den Betrag einer Zahl benötigen, bietet Maxima folgenden Befehl:

|x|…abs(x)

```
(%i1) abs(-436/23);
(%o1) 436/23
```

4.2 Speichern und Exportieren

Mit Maxima kann man Arbeitsblätter in verschiedenen Formaten speichern. Man muss entscheiden, ob man das Arbeitsblatt weiter in Maxima verwenden möchte (*Speichern*) oder ob man es in anderen Programmen braucht (*Exportieren*).

4.2.1 Speichern eines Arbeitsblatts

Normalerweise werden Arbeitsblätter als *„wxMaxima document“ (*.wxm)* gespeichert. Dieses Format erlaubt es, dass man mit den gespeicherten Inputs später weiterrechnet. Da das Speichern im Prinzip als Text-Datei erfolgt, kann man eine wxm-Datei auch im Editor öffnen. Davor wird aber ausdrücklich gewarnt, weil eine falsche Eingabe in die Text-Datei natürlich das Arbeitsblatt zerstört.

Dann steht noch das Format *„wxMaxima xml document“ (*.wxmx)* zur Verfügung. Hier werden nicht nur Inputs und Kommentare gespeichert, sondern auch Outputs. Dieses Format macht zum Beispiel Sinn, wenn man auf Ergebnisse zugreifen möchte, die eine sehr lange Rechenzeit benötigen. Solchen Rechnungen wird man in der Oberstufe aber nicht begegnen und deshalb verwendet man dieses Format eher selten.

Als dritte Möglichkeit bietet Maxima das Erstellen eines sogenannten *„Maxima batch file“ (*.mac)*. Hierbei handelt es sich um Makro-Dateien, die in Maxima geladen werden können, um seine Funktionen zu erweitern. Einige dieser *Batch Files* sind bei der Installation dabei und auf relevante Files gehe ich in den nächsten vier Kapiteln ein.
Mit diesen *Batch Files* kann man Maxima auch individuell anpassen. Man kann zum Beispiel das Laden anderer, oft verwendeter, Batch Files und selbst definierte Funktionen in eine Makrodatei zusammenfassen und diese beim Start automatisch laden lassen.
Ich habe das zum Beispiel mit meiner Funktion für das Kreuzprodukt (6.6.1) und mit Umwandlungsfunktionen zwischen Grad und Bogenmaß so gemacht:

- Zuerst habe ich die Definitionen (4.4.2) meiner Funktionen ganz normal eingegeben.
- Wählt man dann beim speichern den Dateityp **.mac*, muss man das *Batch File* nur noch in einen Ordner speichern, den Maxima findet. Um zu wissen, welche Ordner das sind, lässt man sich die Pfade mit dem Befehl *file_search_maxima* anzeigen. Wählt man einen beliebigen Datei-Namen muss man das Paket trotzdem jedes Mal selbst laden. Nur wenn man sie *maxima-init.mac* nennt wird sie automatisch geladen.

4.2.2 Exportieren eines Arbeitsblatts

Unter dem Menüpunkt *File-Export…* kann man Arbeitsblätter in ein anderes Format umwandeln und so zum Beispiel auch in Textverarbeitungsprogrammen bearbeiten. Für die Arbeit an meiner FBA habe ich immer *HTML-Dateien* erstellt. Man kann diese Dateien entweder in einem Browser betrachten oder in einem Textverarbeitungsprogramm (*Word*) verändern. Zu beachten ist hierbei das die Outputs in einem extra Ordner als Bilder gespeichert werden und in die HTML-Dateien nur verlinkt sind. Ändert man also irgendwelche Dateien- oder Ordnernamen (auch von Übergeordneten) können die Ergebnisse nicht mehr angezeigt werden. Auf diese Art erstellte HTML-Dateien können natürlich auch ganz einfach auf Websites gestellt werden.

Als Alternative bietet Maxima das Erstellen einer LaTex-Datei. Das funktioniert im Großen und Ganzen ähnlich wie HTML, basiert aber auf einem anderen Format.

4.3 Shortcuts

In dieser Tabelle habe ich einige wichtige Shortcuts gesammelt, die das Arbeiten mit Maxima wesentlich vereinfachen. Natürlich können viele dieser Befehle auch über das Menü aufgerufen werden (dort sind sie jeweils mit den Tastenkürzeln aufgelistet).

Shortcut	Funktion	Beschreibung
F1	Hilfe	Ruft allgemeine Hilfe zu Maxima (nur Englisch) oder zu dem Befehl, den man vorher angeklickt hat, auf.
F5	Inputzeile	Fügt nach der ausgewählten Gruppe (Input und Output) eine neue Inputzeile ein.
F6	Textfeld	Fügt nach der ausgewählten Gruppe ein Textfeld ein (z.B. für Beschriftungen oder Kommentare).
F7	n.m-Aufzählung	Fügt Unteraufzählungspunkt ein (1.1).
F8	n-Aufzählung	Fügt Aufzählungspunkt ein (1.).
F9	Titel	Textfeld mit großer und unterstrichener Schrift.
Strg+R	Neuberechnung	Alle Inputs des Arbeitsblatts werden neu berechnet.
%	ANS	Maxima setzt das zuletzt berechnete Ergebnis ein.
%ix	Input	Zugriff auf x-ten Input
%ox	Output	Zugriff auf x-ten Output
Strg+G	Abbrechen	Unterbricht eine laufende Berechnung

4.4 Weitere Funktionen

4.4.1 Hilfe

Über das Menü *Help-Maxima Help* oder mit *F1* kann man die eingebaute Hilfe von Maxima aufrufen. Diese ist zwar sehr umfangreich, aber es gibt sie nur auf Englisch. Unter anderem ist das auch ein Grund, warum ich die Menüführung auch auf Englisch eingestellt habe.

Für einige Funktionen bietet Maxima eine direkte Eingabehilfe indem man durch mit *example(Funktion)* ein Beispiel zu einer Funktion aufruft:

```
(%i1) example(delete);
(%i2) delete(sin(x),y+sin(x)+x)
 (%o2) y+x
 (%o2) done
```

Ein weiterer Trick um die Hilfe schneller aufzurufen ist es auf das Input-Kürzel der Funktion, bei der man Hilfe benötigt, zu klicken und dann F1 zu drücken. So wird die Maxima-Hilfe gleich nach dieser Funktion durchsucht.

4.4.2 Variablen-Management

Wie bei vielen Taschenrechnern ist es auch bei Maxima möglich Ergebnisse als Buchstaben oder Buchstaben-Zahlen-Kombinationen zu speichern. Außerdem kann man Funktionen definieren und Variablen festlegen. Maxima unterscheidet dabei zwischen einer Zahl, einem einzelnen Ausdruck oder einer Gleichung und einer Funktion.

Festlegen einer Variablen...Bezeichnung: Wert

```
(%i1) a:3$
(%i2) a/6;
```

$$(\%o2)\ \frac{1}{2}$$

Speichern eines Ergebnisses...Bezeichnung: Shortcut zum Output

```
(%i3) b:%o2$
```

Definieren einer Funktion...Bezeichnung(Variable):=Term

```
(%i4) f(x):=x^2$
```

Jetzt kann man die Funktion verwenden:

```
(%i5) f(b);
```

$$(\%o5)\quad \frac{1}{4}$$

Im Gegensatz zu den Befehlen, die immer kleingeschrieben sind, unterscheidet Maxima bei Funktionen und Variablen zwischen Groß- und Kleinschreibung. So können zum Beispiel a und A getrennt festgelegt werden. Verwendet man zweimal denselben Buchstaben ersetzt die zweite Definition die erste. Man kann sogar eine Variable a und gleichzeitig eine Funktion a(x) definieren.

Will man wissen welche Variablen und Funktionen bereits definiert sind, verwendet man die Befehle „values“ beziehungsweise „functions“.

```
(%i6) values;
(%o6)  [a, b]

(%i7) functions;
(%o7)  [f(x)]
```

Die angezeigten Variablen und Funktionen sind immer nur für dieses eine Arbeitsblatt definiert. Nach einem Schließen und Neuöffnen sind sie nicht mehr definiert (Neustart des Programmkerns).

Um definierte Variablen und Funktionen ohne Programm-Neustart zu löschen, verwendet man den Befehl *kill*:

kill(x)…löscht die Variable x und die Funktion x
kill(all)…löscht alle definierten Variablen und Funktionen

5 Neunte Schulstufe

5.1 Vektorrechnung in zwei Dimensionen

5.1.1 Vektoroperationen

Überprüfe rechnerisch, von welchem Typ das Viereck ABCD ist (Quadrat, Raute, Rechteck, Parallelogramm, Trapez, Deltoid oder allgemeines Viereck)![4]

A(-1|-3), B(1|1), C(0|3), D(-2|-1)

Zuerst speichert man die gegebenen Punkte und die Pfeile AB, BC, CD und DA als Spaltenvektoren.

Vektoren werden durch einspaltige Matrizen eingegeben: $\binom{x}{y}$ … matrix([x],[y]).

Einfacher wird das Ganze, wenn man ein Dialogfenster benützt, das man mit *Algebra-Enter Matrix…* aufruft (Einstellungen - Rows:2; Columns:1; Type: General).

```
(%i1) A:matrix([-1],[-3])$
B:matrix([1],[1])$
C:matrix([0],[3])$
D:matrix([-2],[-1])$

(%i5) AB:B-A$
BC:C-B$
CD:D-C$
DA:A-D$
```

Nun kann überprüft werden, ob AB und CD bzw. BC und DA parallel sind. Das sind sie dann, wenn bei der Division der beiden Vektoren beide Ergebniskoordinaten gleich sind.

```
(%i9) AB/CD;
```

$$(\%o9)\quad \begin{bmatrix} -1 \\ -1 \end{bmatrix}$$

[4] Vgl.: Götz, Stefan u.a.: Mathematik. Lehrbuch 5. Wien: öbvhpt, 2004. S.227 - Bsp. 710.

(%i10) BC/DA;

(%o10) $\begin{bmatrix} -1 \\ -1 \end{bmatrix}$

Die gegenüberliegenden Pfeile sind also jeweils parallel. Die verbleibenden Möglichkeiten für dieses Viereck sind Quadrat, Raute, Rechteck oder Parallelogramm.

Als Nächstes berechnet man die Längen der einzelnen Pfeile mit Hilfe des Satzes von Pythagoras.

Falls Vektor in A gespeichert ist, entnimmt A[x] die x-te Koordinate dieses Vektors.

(%i11) sqrt(AB[1]^2+AB[2]^2);

(%o11) $2\sqrt{5}$

(%i12) sqrt(BC[1]^2+BC[2]^2);

(%o12) $\sqrt{5}$

(%i13) sqrt(CD[1]^2+CD[2]^2);

(%o13) $2\sqrt{5}$

(%i14) sqrt(DA[1]^2+DA[2]^2);

(%o14) $\sqrt{5}$

Es ergibt sich, dass gegenüberliegende Pfeile jeweils gleichlang sind. Es verbleiben nur noch die Möglichkeiten Rechteck oder Parallelogramm.
Um zum endgültigen Ergebnis zu gelangen, testet man, ob zwischen zwei angrenzenden Pfeilen ein rechter Winkel liegt, d. h. ob das skalare Produkt der Pfeile verschwindet.

Skalares Produkt der Vektoren A und B... A.B

(%i15) AB.BC;

(%o15) 6

Man findet keinen rechten Winkel und daher ist dieses Viereck ein Parallelogramm.

5.1.2 Normalprojektion

Berechne die Normalprojektion des Vektors $\vec{a}$ (2|-1) auf $\vec{b}$ (9|-1)!

Am besten speichert man die Vektoren zu erst unter a und b ab.

```
(%i1) a:matrix([2],[-1])$
      b:matrix([9],[-1])$
```

Nun berechnet man den Einheitsvektor von b. Man muss für diesen Befehl aber zuerst ein sogenanntes Package, das heißt eine Erweiterung für das Programm, in Maxima laden.

Für das Berechnen von Einheitsvektoren...load(eigen)

```
(%i3) load(eigen)$
```

Nun kann man mit den neuen Befehlen b_0 definieren.

Einheitsvektor von v...uvect(v) oder unitvector(v)

```
(%i4) b0:uvect(b);
```

$$(\%o4)\quad \begin{bmatrix} \frac{9}{\sqrt{82}} \\ -\frac{1}{\sqrt{82}} \end{bmatrix}$$

Jetzt kann man die Normalprojektion mit dem skalaren Produkt durchführen.

```
(%i5) a.b0;
```

$$(\%o5)\quad \frac{19}{\sqrt{82}}$$

5.2 Zahlensysteme

5.2.1 Umrechnen verschiedener Zahlensysteme

Standardmäßig ist Maxima auf das Dezimalsystem eingestellt. Das Programm erlaubt es jedoch verschiedene Zahlensysteme für Input und Output einzustellen. Man kann dabei die Basen zwischen 2 und 36 wählen (es stehen mit den Zahlen von 0 bis 9 und den 26 Buchstaben des Alphabeth auch 36 Ziffern zur Verfügung).

Basis x für den Input…ibase:x
Basis y für den Output…obase:y

Will man nun zum Beispiel Hexadezimalzahlen ins Binärsystem umrechnen geht man wie folgt vor:

```
(%i1) ibase:16$
      obase:2$
```

Zu beachten ist, dass Maxima Zahlen, die mit Buchstaben beginnen nur erkennt, wenn man am Beginn eine Null anhängt.

```
(%i3) [0CF,13,0AB];
  (%o11)  [11001111,10011,10101011]
```

Will man wieder im Dezimalsystem weiterrechen, muss man bei der Rückstellung die aktuelle ibase-Einstellung beachten.

```
(%i4) ibase:0A$
      obase:10$
```

Deshalb ist es einfacher einen Neustart durchzuführen und so wieder auf die Standardeinstellungen zu wechseln (über *Maxima-Restart Maxima*).

5.3 Gleichungen

5.3.1 Lösen von Gleichungen

Zum Lösen der meisten Gleichungen bietet Maxima den Befehl *solve*, den man auch über *General Math* oder das Menü *Equations* aufrufen kann:

Lösen der Geleichung nach x…solve(Gleichung,x)

```
(%i1) solve(x^2+4*x-3*a*x-12*a,x);
  (%o1)  [x=3 a,x=-4]
```

Sollte man in *solve* nur einen Term statt einer Gleichung eingeben, wird dieser nullgesetzt und ebenfalls gelöst.

5.3.2 Gleichungssysteme

Solve kann auch mit mehreren Gleichungen und Variablen umgehen, wenn man diese wie folgt eingibt:

Lösen eines Gleichungssystems nach x und y…solve([Gleichung1,Gleichung2],[x,y])

Löse das folgende Gleichungssystem in RxR![5]

```
(%i1) glg1:x=y-7$
      glg2:x=5*y-23$

(%i3) solve([glg1,glg2],[x,y]);
  (%o3)  [[x=-3,y=4]]
```

Um bei einem größeren Gleichungssystem die Übersichtlichkeit zu wahren, kann man die Eingabe auch über ein Dialogfenster durchführen. Man wählt für lineare Systeme *Equations-Solve Linear System…* und gibt dann zuerst die Anzahl der Gleichungen gefolgt von den Gleichungen und den Variablen ein. Bei anderen als linearen Gleichungssystemen verwendet man *Equations-Solve Algebraic System…*

5.4 Trigonometrie

Maxima verwendet für Winkel immer das Bogenmaß. Benötigt man das Gardmaß muss man von Hand umrechnen oder man definiert sich eigene Funktionen.

π…%pi

5.4.1 Winkelfunktionen

Folgende Befehle verwendet man zum Berechnen der Winkelfunktionen mit Input x im Bogenmaß:

Sinus…sin(x); Cosinus…cos(x); Tangens…tan(x)

[5] Vgl.: Götz: Mathematik. Lehrbuch 5. 2004. S.159 - Bsp. 425a).

```
(%i1) cos(%pi/6);
```

$$(\%o1)\ \frac{\sqrt{3}}{2}$$

Die Umkehrfunktionen mit Ergebnissen im Bogenmaß sehen so aus:

Arcussinus…asin(z); Arcuscosinus…acos(z); Arcustangens…tan(z)

```
(%i2) atan(-sqrt(3));
```

$$(\%o2)\ -\frac{\pi}{3}$$

5.4.2 Berechnungen im schiefwinkeligen Dreieck

Berechne das fehlende Umfangstück, wenn im Dreieck ABC zwei Seiten und der eingeschlossene Winkel gegeben sind![6]

a = 114,3; c = 84,8; β = 25,72°

Um die Formeln übersichtlich zu halten, speichert man zuerst die bekannten Seiten und Winkel ab, wobei man den Winkel auch gleich ins Bogenmaß umrechnet.

```
(%i1) a:114.3$
      c:84.8$
      beta:25.72*(%pi/180)$
```

Man stellt den Cosinussatz für die gesuchte Seite b auf.

```
(%i4) b^2=a^2+c^2-2*a*c*cos(beta)$
```

Daraus berechnet man dann eine Näherung für b.

```
(%i5) solve(%,b)$

(%i6) float(%), numer;
(%o6) [b=-52.82835566419508,b=52.82835566419508]
```

Das gesuchte Umfangstück ist also ca. 52,83 lang.

---

[6] Vgl.: Götz: Mathematik. Lehrbuch 5. 2004. S.201 - Bsp. 628a).

6 Zehnte Schulstufe

6.1 Funktionen

6.1.1 Exponential- und Logarithmusfunktion

Maxima verwendet nur die Eulersche Zahl *e* als eingebaute Basis von Exponentialfunktionen und Logarithmen. So greift man darauf zu:

e^x…exp(x) oder %e^x **Natürliche Logarithmus von x …log(x)**

```
(%i1) exp(2)+%e^3,numer;
  (%o1) 27.47459302211832

(%i2) log(4),numer;
  (%o2) 1.386294361119891
```

Will man andere Logarithmen verwenden, bietet es sich an eine Funktion zum Umrechnen einer beliebigen Basis a zu definieren:

```
(%i3) loga(x,a):=log(x)/log(a)$
```

Zum Beispiel $\log_3 7$ kann man dann berechnen:

```
(%i4) loga(7,3),numer;
  (%o4) 1.771243749161422
```

6.1.2 Exponentielle Wachstumsvorgänge

Angenommen die Energie, die der Mensch auf der Erde erzeugt, nimmt pro Jahr um 5% zu und 10% werden als Strahlung ins Weltall abgegeben. Wann leuchtet die Erde so hell wie die Sonne (5 10^{26}W), wenn der momentane Jahresenergieverbrauch bei 100 Millionen Gigawattstunden liegt?

Für die jährlich abgegebene Strahlung der Erde in Abhängigkeit von der Zeit t gilt (Strahlung in Joule und t in Jahren):

```
(%i1) strahlung(t):=0.1*3.6*10^20*1.05^t$
```

Man rechnet jetzt die Leuchtkraft der Sonne auf Strahlung pro Jahr um:

```
(%i2) sonne:5*10^26*3600*24*365$
```

Mit *solve* berechnet man das Ergebnis:

```
(%i3) solve(sonne=strahlung(t),t),numer$

(%i4) float(%), numer;
  (%o4)  [t=690.9843527738265]
```

Die Erde würde also in etwa 691 Jahren bei unverändertem Energiezuwachs so hell wie die Sonne strahlen.

6.1.3 Logistisches Wachstum

Um mit dem logistischen Wachstum zu rechnen, definiert man sich am besten eine multivariate Funktion für die Population N in Abhängigkeit von der Zeit t, der Zuwachsrate a, der Anfangspopulation N_0 und der oberen Schranke K.

```
(%i1) N(t,a,N0,K):=K*N0*a^t/(N0*a^t+K-N0)$
```

Nimmt man nun das vorherige Beispiel (6.1.2) mit dem Energiezuwachs der Erde und sagt, dass dieses Wachstum ein logistisches sei und aufgrund der natürliche Begebenheiten auf das Doppelte des heutigen Levels beschränkt ist, kann man eine logistische Wachstumsfunktion mit folgenden Parametern aufstellen:

a=1,05; N_0=100 Mio. GWh; K=2 N_0

```
(%i2) NErde(t):=N(t,1.05,100,200)$
```

Nun kann man die Werte für bestimmte Zeitpunkte in der Zukunft mit dem Befehl *evaluate* berechnen.

Berechnung der Bilder eines Funktionsterms...ev(f(x) , x=[Wert1,Wert2])

```
(%i3) ev(NErde(t),t=[5,20,100]);
  (%o3)  [112.137407210581,145.2549405655147,
198.4905803669276]
```

Die Ergebnisse liegen jeweils in Millionen Gigawattstunden vor.

6.2 Ploten von Funktionen (2D)

Zur Veranschaulichung kann man die Funktionen, mit denen man hier rechnet, natürlich auch graphisch darstellen. Maxima bietet viele Möglichkeiten um die Graphen einzufärben und einzelne Punkte anzuzeigen. Außerdem kann man mit Maxima auch Stecken zeichnen und vieles mehr. Da es für solche graphischen Arbeiten aber bessere und vor allem benutzerfreundlichere Programme[7] gibt, gehe ich hier nur auf die Grundzüge ein.

Ploten der Funktionen f(x) und g(x), wobei x ϵ [-2;5] und y ϵ [-4;4]…
… wxplot2d([f(x),g(x)],[x,-2,5],[y,-4,4])

Während für die 1. Achse ein Bereich angegeben werden muss, würde Maxima den Bereich für die 2. Achse anhand der Funktionswerte auch selbstständig wählen.

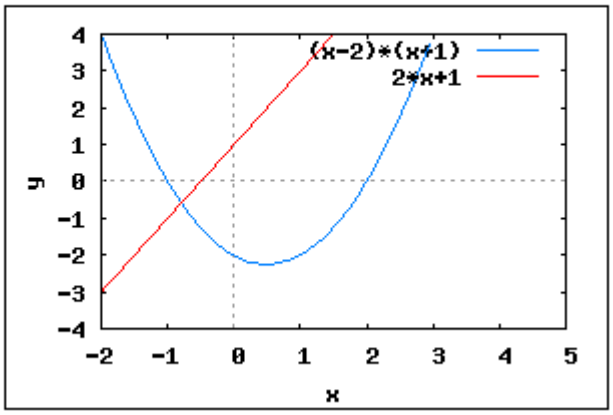
```
(%i1) f(x):=(x+1)*(x-2)$
      g(x):=2*x+1$
```

Wem die standardmäßige Größe des Koordinatensystems nicht passt, kann sie auch verändern:

Gewünschte Größe 300x200 Pixel...wxplot2d(...),wxplot_size=[300,200]

```
(%i3) wxplot2d([f(x),g(x)],[x,-2,5],[y,-4,4]),wxplot_size=[300,200]$
plot2d: some values were clipped.
plot2d: some values were clipped.
```

(%t3)



[7] Ein Beispiel wäre das Programm GeoGebra, das ebenfalls gratis aus dem Internet heruntergeladen werden kann (http://www.geogebra.org/cms/ [Stand: 20-02-2010]).

6.3 Ungleichungen

6.3.1 Lösungsbefehl für Ungleichungen

Mit einer Ergänzung kann Maxima auch Ungleichungen lösen. Dieser Befehl ist aber noch nicht ganz ausgereift. So ist das Programm auf Ungleichungen mit einer Variablen beschränkt. Auch Ungleichungsketten müssen aufgeteilt auf zwei Ungleichungen berechnet werden.

Zum Lösen von Ungleichungen…load(solve_rat_ineq)

Zur Eingabe stehen die folgenden Ungleichungszeichen zur Verfügung:

< > ≤…<= ≥…>=

```
(%i1) load(solve_rat_ineq)$
```

```
(%i2) ineq:-2*x/5+1/10<=1/5;
```

$$(\%o2)\quad -\frac{2x}{5}+\frac{1}{10}\leq\frac{1}{5}$$

```
(%i3) solve_rat_ineq(ineq);
```

$$(\%o3)\quad [[x\geq -\frac{1}{4}]]$$

Der Befehl *solve_rat_ineq* hat auch Probleme, wenn bei der Lösung einer Ungleichung wie im folgenden Beispiel auf einer Seite eine Null auftaucht.

```
(%i4) ineq2:5*(4-y)/3-10/6>=5;
```

$$(\%o4)\quad \frac{5(4-y)}{3}-\frac{5}{3}\geq 5$$

```
(%i5) solve_rat_ineq(ineq2);
(%o5) all
```

Die eigentliche Lösung wäre x≤0, aber Maxima gibt als Lösung alles aus.

6.4 Lineare Gleichungssysteme

Berechne den Term einer Polynomfunktion vierten Grades, die durch die Punkte (0|1), (1|3), (2|25), (-1|1) und (-2|9) geht.

Hier bieten sich zwei Lösungswege an:

a. Zuerst definiert man die allgemeine Polynomfunktion, um die Punkten dann in lin*solve* einzusetzen:

```
(%i1) f(x):=a4*x^4+a3*x^3+a2*x^2+a1*x+a0$

(%i2) linsolve([f(0)=1, f(1)=3, f(2)=25, f(-1)=1, f(-2)=9], [a4,a3,a2,a1,a0]);
  (%o2)  [a4=1,a3=1,a2=0,a1=0,a0=1]
```

Der Term der Polynomfunktion ist also $x^4 + x^3 + 1$.

b. Lineare Gleichungssysteme lassen sich auch mit Matrizenrechnung lösen (die Gleichungen sind vor allem dann schneller einzugeben, wenn die Variablen gleich sortiert sind).

Anlegen der Koeffizientenmatrix für dieses Polynom:

```
(%i6) coef: matrix(
        [0,0,0,0,1],
        [1,1,1,1,1],
        [16,8,4,2,1],
        [1,-1,1,-1,1],
        [16,-8,4,-2,1])$
```

Um zu erfahren ob dieses Gleichungssystems eindeutig lösbar ist, kann man noch die Determinante berechnen.

Determinante der Matrix X...determinant(X)

```
(%i7) determinant(coef);
  (%o7)  -288
```

Eine Determinante ungleich Null bedeutet, dass es eine eindeutige Lösung gibt.

Jetzt definiert man einen Spaltenvektor, der diese Matrix auf eine erweiterte Koeffizientenmatrix ergänzen würde.

```
(%i8) vect: matrix( [1], [3], [25], [1], [9])$
```

Multipliziert man die inverse Koeffizientenmatrix mit diesem Vektor, so erhält man als Produktmatrix den Lösungsvektor.

Inverse Matrix von X...invert(X) oder X^^-1

```
(%i9) invert(coef).vect;
```

$$(\%o9)\quad \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

Für den Term der Polynomfunktion ergibt sich also wieder $x^4 + x^3 + 1$.

6.5 Folgen und Reihen

6.5.1 Grenzwert einer Folge

Maxima bietet auch für Grenzwertberechnungen einen Befehl. Zur Eingabe benötigt man noch ∞ und − ∞:

∞ … inf −∞ … -inf oder minf

$\alpha = \lim_{n\to\infty} a_n$ …limit(a_n, n, inf)

```
(%i1) limit((1+1/n)^n,n,inf);
  (%o1)  %e
```

6.5.2 Summe einer unendlichen geometrischen Reihe

Verwendet man den Summenbefehl kann man eine Reihe näherungsweise berechnen.

$\sum_{i=u}^{o} a(i)$ … sum (a(i), i, u, o)

Bei Summen mit vielen Summanden sollte man das Ergebnis unterdrücken, falls es nicht genau berechnet werden kann, denn sonst gibt Maxima jeden Summanden aus. Durch eine Näherung erhält man ein einzeiliges Ergebnis.

Ein Gummiball fällt aus 1m Höhe auf den Boden, steigt dann 0,8 wieder auf, steigt nach dem nächsten Hinunterfallen 0,64 m auf, usw. Welchen Weg legt der Ball insgesamt zurück.[8]

Für ein ungefähres Ergebnis kann man die ersten 50 Folgenglieder von $a_n=0{,}8^{i-1}$ addieren, um den „heruntergefallenen Weg“ zu berechnen.

```
(%i1) sum(0.8^(i-1),i,1,50),numer;
  (%o1)  4.999928637615367
```

Der Weg wird aber bis auf das erste Folgenglied doppelt zurückgelegt. Daher gilt für den Gesamtweg:

```
(%i2) %o1*2-1;
  (%o2)  8.999857275230735
```

Man erkennt, dass die Summe dieser Reihe wohl gegen 9 geht.

6.6 Vektorrechnung in drei Dimensionen

6.6.1 Kreuzprodukt

Der einzige neue Befehl, der durch die dritte Dimension in der Vektorrechnung dazukommt, ist das Kreuzprodukt. Hier zeigt sich eine der Schwächen von Maxima. Es gibt zwar einen Befehl für das Kreuzprodukt, aber der ist einerseits relativ kompliziert in der Anwendung und funktioniert auch nur bei Zeilenvektoren.

Da man normalerweise in der Geometrie Spaltenvektoren verwendet, habe ich mich entschieden eine eigene Funktion für das Kreuzprodukt anzulegen.

a x b…crossP(a,b)

Und hier die Definition:

```
(%i1) crossP(vect1,vect2):=matrix(vect1[2]*vect2[3]-vect1[3]*vect2[2],
vect1[3]*vect2[1]-vect1[1]*vect2[3],vect1[1]*vect2[2]-vect1[2]*vect2[1])$
```

Und so funktioniert *crossP*:

[8] Vgl.: Malle, Günther u.a.: Mathematik verstehen 6. Wien: öbvhpt, 2006. S.145 - Bsp. 8.09.

```
(%i2) a:matrix([-2],[6],[-1])$
b:matrix([3],[-4],[1])$

(%i4) crossP(a,b);
```

$$(\%o4)\begin{bmatrix} 2 \\ -1 \\ -10 \end{bmatrix}$$

Natürlich muss die Definition von *crossP* bei jedem Neustart wiederholt werden, außer man geht so wie in 4.2.1 beschrieben vor.

6.6.2 Betrag eines Vektors

Im vorherigen Kapitel haben wir den Betrag eines Vektors noch etwas umständlich mit den einzelnen Koordinaten berechnet (5.1.1). Einfacher geht es mit der Wurzel aus dem skalaren Produkt des Vektors mit sich selbst (Quadrieren des Vektors).

Betrag des Vektors $\vec{x}$…sqrt(x^^2)

```
(%i1) vect: matrix( [5], [-3], [-7]);
```

$$(\%o1)\begin{bmatrix} 5 \\ -3 \\ -7 \end{bmatrix}$$

```
(%i2) sqrt(vect^^2);
```

$$(\%o2)\begin{bmatrix} \sqrt{83} \end{bmatrix}$$

Will man mit dem Betrag weiterrechnen, muss man ihn aus der eckigen Klammer herausholen. Entweder man kopiert ihn heraus oder wendet einen Trick an:

```
(%i3) sqrt(vect^^2)[1][1];
```

$$(\%o3)\ \sqrt{83}$$

6.6.3 Winkel zwischen Vektoren

Mit dem skalaren Produkt kann man wie in der zweidimensionalen Geometrie den eingeschlossenen Winkel im Bogenmaß berechnen. Für die Vektoren $\vec{a}$ (1|-3|2) und $\vec{b}$ (5|2|-1) sieht das so aus:

```
(%i1) a: matrix( [1], [-3], [2])$
      b: matrix( [5], [2], [-1])$

(%i3) winkel:acos(a.b/(sqrt(a^^2)[1][1]*sqrt(b^^2)[1][1]))$

(%i4) winkel,numer;
  (%o4) 1.717709247923565
```

6.6.4 Abstand zwischen Punkt und Gerader

Diese Aufgabe dient als Beispiel für die vielen Berechnungen zu einfachen geometrischen Objekten. Sie basieren alle auf denselben Befehlen, die in unterschiedlichen Formeln verpackt sind.

Berechne den Abstand zwischen dem P(0|5|6) und der Geraden g: X=(2|0|1)+μ (-4|1|1)!

Die Formel zur Berechnung lautet: $d = |\vec{a}_0 \times (\vec{P}-\vec{X}_1)|$

```
(%i1) load(eigen)$
      crossP(vect1,vect2):=matrix(vect1[2]*vect2[3]-vect1[3]*vect2[2],
      vect1[3]*vect2[1]-vect1[1]*vect2[3],vect1[1]*vect2[2]-vect1[2]*vect2[1])$
      a:matrix([-4],[1],[1])$
      P:matrix([0],[5],[6])$
     X1:matrix([2],[0],[1])$
     a0:uvect(a)$

(%i7) sqrt(crossP(a0,(P-X1))^^2);
```

$$(\%o7)\quad \left[\sqrt{\left(\frac{5\,2^{3/2}}{3}-\frac{\sqrt{2}}{3}\right)^2+\left(\frac{\sqrt{2}}{3}-\frac{5\,2^{3/2}}{3}\right)^2}\right]$$

```
(%i8) float(%), numer;
  (%o8) [6.000000000000002]
```

Hier zeigt sich eine Schwäche von Maxima. Obwohl das Ergebnis eigentlich exakt 6 sein sollte, taucht weit hinter dem Komma wieder eine Ziffer ungleich Null auf. Das liegt daran, dass Maxima Gleitkommazahlen als Binärzahlen abspeichert. Dieses Problem tritt auch in anderen Softwareprodukten auf. Hier kann man zum Beispiel durch die Umwandlung in eine große Gleitkommazahl (*bfloat)* Abhilfe schaffen.

6.7 Beschreibende Statistik

In diesem Kapitel geht es darum, Listen mit verschiedenen Kennzahlen zu versehen. Maxima bietet im Umgang mit Listen viele Befehle und ich erkläre nur die für den Unterricht direkt relevanten. Natürlich kann man die Hilfe oder das Internet nach weiteren Informationen durchsuchen.

Zuerst muss man wieder eine Erweiterung laden:

Für beschreibende Statistik…load(descriptive)

6.7.1 Zentralmaße

Am besten man speichert eine Liste mit Werten, zum Beispiel den Noten einer Schularbeit, im Vorhinein ab.

```
(%i2) sa:[1,4,5,3,3,2,4,1,1,2,4,3]$
```

Nun kann man arithmetisches Mittel und Median berechnen.

Arithmetisches Mittel der Werteliste x...mean(x)

```
(%i3) mean(sa),numer;
(%o3) 2.75
```

Median der Werteliste x...median(x)

```
(%i4) median(sa);
(%o4) 3
```

6.7.2 Streuungsmaße

Genau wie bei den Zentralmaßen kann man auch Varianz und Standardabweichung berechnen.

Varianz der Werteliste x...var(x)

```
(%i5) var(sa);
(%o5) 27/16
```

Standardabweichung der Werteliste x...std(x)

```
(%i6) std(sa);
```

$$(\%o6)\quad \frac{3^{3/2}}{4}$$

Man erkennt auch, dass die Standardabweichung als Quadratwurzel der Varianz definiert ist.

7 Elfte Schulstufe

7.1 Polynomgleichungen

Für Maxima sind auch Gleichungen höheren Grades kein Problem. Man verwendet ganz normal den *solve*-Befehl.

```
(%i1) solve(x^4-5*x^2+6=0,x);
```
$$(\%o1)\quad [x=-\sqrt{2},x=\sqrt{2},x=-\sqrt{3},x=\sqrt{3}]$$

Probleme ergeben sich dann, wenn eine Lösung nicht mehr exakt sondern nur noch nummerisch möglich ist. Hier versagt *solve*.

```
(%i2) solve(x^5-3*x^3+4*x^2-x+13/3=0,x);
```
$$(\%o2)\quad [0=3x^5-9x^3+12x^2-3x+13]$$

Man kann dann zum Beispiel den Befehl *algsys* verwenden, der auch algebraische Gleichungssysteme löst. Dieser Befehl funktioniert mit den gleichen Argumenten wie *solve* in eckigen Klammern:

Lösen der Geleichung nach x…algsys([Gleichung], [x])

algsys ist ein übergeordneter Lösungsalgorithmus, der bei exakten Lösungen auf *solve* zurückgreift, aber auch nummerische Lösungsverfahren beherrscht.

```
(%i3) algsys([x^5-3*x^3+4*x^2-x+13/3=0],[x]);
(%o3) [[x=1.319967222863467-0.81115972017536 %i],[x=
0.81115972017536 %i+1.319967222863467],[x=-
2.299051117232935],[x=-0.86959861219832 %i-
0.17044166711225],[x=0.86959861219832 %i-0.17044166711225]
]
```

Die Lösung ist jetzt zwar möglich, aber es werden auch komplexe Lösungen berechnet. Bei *algsys* schafft das Anhängen von *‚realonly:true* Abhilfe.

```
(%i4) algsys([x^5-3*x^3+4*x^2-x+13/3=0],[x]),realonly:true;
(%o4) [[x=-2.299051117232935]]
```

Als Alternative stehen für eine Gleichung auch die Befehle *realroots* und *allroots* zur Verfügung:

Grundmenge R ... realroots(Gleichung,Genauigkeit) **Grundmenge C ... allroots(Gleichung)**

```
(%i5) realroots(x^5-3*x^3+4*x^2-x+13/3=0,10^-10);
```

$$(\%o5)\quad [x=-\frac{78994794685}{34359738368}]$$

```
(%i6) float(%), numer;
(%o6)  [x=-2.299051111476729]
```

Bis zur zehnten Stelle hinter dem Komma stimmt diese Lösung mit der von *algsys* überein.

```
(%i7) allroots(x^5-3*x^3+4*x^2-x+13/3=0,10^-10);
(%o7)  [x=0.86959861219832 %i-0.17044166711225,x=-
0.86959861219832 %i-0.17044166711225,x=-2.299051111502442,
x=0.81115972017536 %i+1.319967222863467,x=
1.319967222863467-0.81115972017536 %i]
```

Hier erhält man wieder dieselben Lösungen wie mit *algsys*.

7.2 Differentialrechnung

7.2.1 Ableitungen berechnen

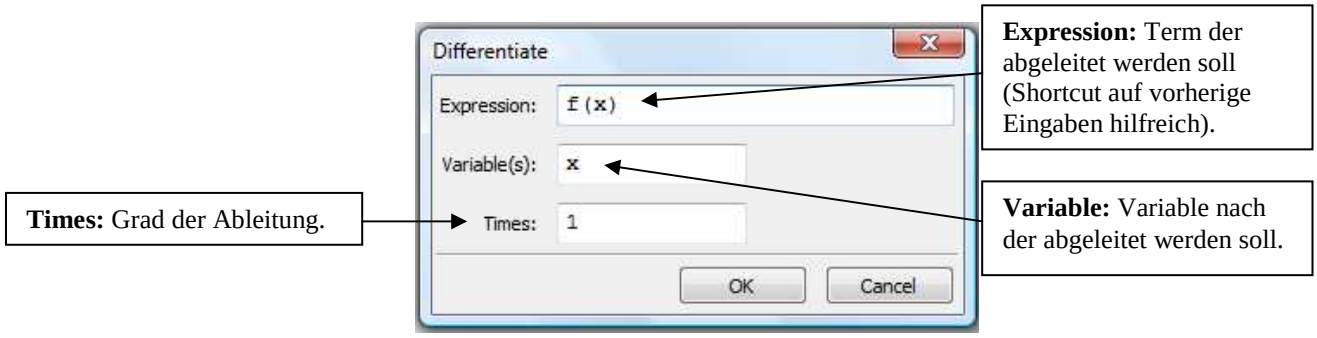
Mit Maxima leitet man eine stetige Funktion durch folgenden Befehl ab:

$\frac{d^n f(x)}{d x^n}$**... diff(f(x),x,n)**

Wird kein n angegeben, erhält man die erste Ableitung.

```
(%i1) f(x):=3*x^2+sin(x)$

(%i2) diff(f(x),x);
(%o2)  cos(x)+6 x

(%i3) diff(f(x),x,2);
(%o3)  6-sin(x)
```

Der Befehl *diff* kann auch über das Menü *Calculus–Differentiate…* oder *General Math* als Dialogfenster aufgerufen werden:



7.2.2 Kurvendiskussion

Ermittle die Null-, Extrem- und Wendestellen der Funktion $f(x) = -x^3 + 3\,x^2 - 3\,x$.

```
(%i1) f(x):=-x^3+3*x^2-3*x;
      diff(f(x),x,1);
      diff(f(x),x,2);
      diff(f(x),x,3);
```

$$(\%o1)\quad f(x):=-x^3+3\,x^2+(-3)\,x$$

$$(\%o2)\quad -3\,x^2+6\,x-3$$

$$(\%o3)\quad 6-6\,x$$

$$(\%o4)\quad -6$$

Beim Arbeiten mit mehreren Ableitungen verschiedenen Grades ist es hilfreich diese als eigene Funktionen zu definieren. Hier muss man aber den Befehl *define* verwenden, weil sonst nur Rückbezüge auf f(x) entstehen und man in die neuen Funktionen nicht einsetzen kann.

Definieren der Funktion f(x) mit dem Term a ... define(f(x),a)

```
(%i5) define(f1(x),diff(f(x),x,1))$
      define(f2(x),diff(f(x),x,2))$
      define(f3(x),diff(f(x),x,3))$
```

Jetzt kann man die Funktion, ihre erste und ihre zweite Ableitung nullsetzen und erhält als Lösungen die gewünschten Stellen.

```
(%i8) Nullstellen:algsys([f(x)=0],[x]),realonly:true;
```

$$(\%o8)\quad [\,[\,x=0\,]\,]$$

```
(%i9) Extremstellen:algsys([f1(x)=0],[x]),realonly:true;
(%o9)  [[x=1]]
```

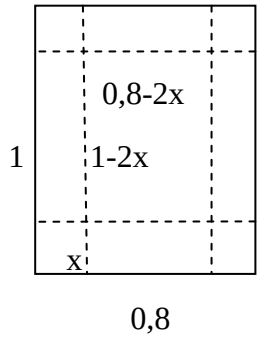
```
(%i10) Wendestellen:algsys([f2(x)=0],[x]),realonly:true;
(%o10)  [[x=1]]
```

Zur Unterscheidung zwischen lokalen Minima, Maxima und einem Terrassenpunkt kann man zum Beispiel die erste Ableitung links und rechts der Extremstelle betrachten.

```
(%i11) f1(0);
       f1(2);
(%o11) -3
(%o12) -3
```

Hier ändert sich das Monotonieverhalten nicht, es liegt also ein Terrassenpunkt vor.

7.2.3 Extremwertaufgaben



Aus einem rechteckigen Kartonstück (1m x 0,8 m) soll durch das Wegschneiden von vier Quadraten an den Ecken das Netz einer Schachtel entstehen. Welche Seitenlänge x müssen diese Quadrate haben, damit das Volumen der Schachtel maximal wird.

In der Realität ergibt sich eine Einschränkung auf x ϵ [0; 0,4].

Zuerst definiert man die Zielfunktion:

```
(%i1) V(x,y,z):=x*y*z$
```

Dann setzt man die aus der Skizze ersichtlichen Nebenbedingungen ein:

```
(%i2) V(x,(1-2*x),(0.8-2*x));
(%o2) (0.8-2 x)(1-2 x)x
```

Nun sucht man nach dem Extremwert wie in 7.2.2 erklärt wurde.

```
(%i3) algsys([diff(V(x,(1-2*x),(0.8-2*x)),x)=0],[x]),realonly:true;
```

$$(\%o3)\quad \left[\left[x=\frac{\sqrt{21}+9}{30}\right],\left[x=-\frac{\sqrt{21}-9}{30}\right]\right]$$

```
(%i4) float(%), numer;
  (%o4)  [[x=0.45275252316519],[x=0.14724747683481]]
```

Durch die anfangs angegebene Einschränkung bleibt nur x = 0.1472... als Lösungsmöglichkeit übrig. Es muss noch ein möglicher Terrassenpunkt ausgeschlossen werden, indem man die erste Ableitung links und rechts vom Ergebnis betrachtet.

```
(%i5) define(V1(x),diff(V(x,(1-2*x),(0.8-2*x)),x))$

(%i6) V1(0.1);
  (%o6)  0.2

(%i7) V1(0.2);
  (%o7)  -0.16
```

Das Monotonieverhalten ändert sich, also wird das Volumen tatsächlich bei x = 0.1472... maximal. Randwertmaxima können hier schon aus logischen Gründen ausgeschlossen werden.

7.3 Kreis und Kugel

Das Rechnen mit Kreis, Kugel und Tangente beschränkt sich im Prinzip auf die Befehle die man für Gleichungen und Funktionen verwendet. Als einzige Ergänzung erkläre ich noch das Ploten eines Kreises.

7.3.1 Ploten eines Kreises

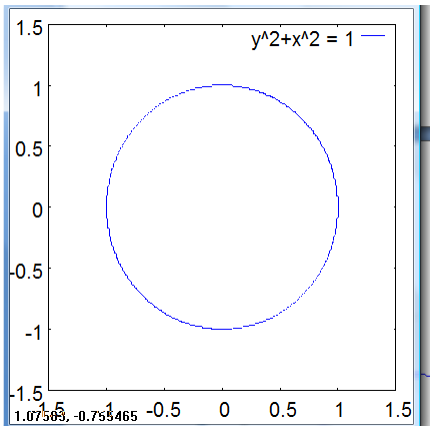
Normalerweise (mit *wxplot2d*) müsste man den Kreis auf zwei Funktionen aufteilen (eine positive und eine negative Wurzelfunktion). Durch eine Erweiterung kann man diesen umständlichen Schritt aber umgehen.

Zur Darstellung von Kreisen…load(implicit_plot)

Gleiches gilt natürlich auch für die Kegelschnitte (7.4).
Der Einheitskreis wird so geplotet (die Darstellung erfolgt bei diesem Befehl immer in einem eigenen Fenster):

```
(%i1) load(implicit_plot)$
```

```
(%i2) implicit_plot([x^2+y^2=1],[x,-1.5,1.5],[y,-1.5,1.5]),wxplot_size=[300,200] ;
  (%o2)  done
```



7.4 Kegelschnitte

Wie schon im vorherigen Kapitel gibt es auch hier keine neuen Befehle. Als Beispiel für die verschiedenen Kegelschnitte, die sich in ihren Formeln nur wenig unterscheiden, berechne ich die Tangente an eine Ellipse.

7.4.1 Tangente an eine Ellipse

Ermittle Gleichungen der Tangenten vom Punkt Q (8|-1) an die Ellipse ell: $x^2 + 6\,y^2 = 10$![9]

Zuerst definiert man die Gleichung der Ellipse.

```
(%i1) ell:x^2+6*y^2=10$
```

Sei der Berührungspunkt der Tangente t an die Ellipse *ell* P ($p_1|p_2$), so gilt für t.

```
(%i2) t: x*p1+6*y*p2=10$
```

Da Q auf t liegt, kann man diesen Punkt für x und y einsetzen.

```
(%i3) ev(t,[x=8],[y=-1]);
  (%o3)  8 p1 - 6 p2 = 10
```

[9] Aus: Malle, Günther u.a.: Mathematik verstehen 7. Wien: öbvhpt, 2007. S.161 - Bsp. 7.65.

P liegt auf der Ellipse und daher kann man x und y mit p1 und p2 substituieren.

```
(%i4) ev(ell,[x=p1],[y=p2]);
```

$$(\%o4)\quad 6\,p2^2+p1^2=10$$

Zwei Gleichungen mit zwei Variablen lassen sich mit *solve* lösen.

```
(%i5) solve([8*p1-6*p2=10, 6*p2^2+p1^2=10], [p1,p2]);
```

$$(\%o5)\quad \left[\left[p1=2,\,p2=1\right],\left[p1=\frac{2}{7},\,p2=-\frac{9}{7}\right]\right]$$

Setzt man die Lösungen in t ein, erhält man die Tangentengleichungen.

```
(%i6) ev(t,[p1=2,p2=1]);
```

$$(\%o6)\quad 6\,y+2\,x=10$$

```
(%i7) ev(t,[p1=2/7,p2=-9/7]);
```

$$(\%o7)\quad \frac{2\,x}{7}-\frac{54\,y}{7}=10$$

Beide Gleichungen kann man noch vereinfachen und es ergeben sich die Lösungen
t_1: x + 3y = 5 und t_2: x - 27y = 35.

7.5 Taylorreihe

Will man die Näherung einer Funktion mit Hilfe der Taylorreihe berechnen, macht es einem Maxima mit einem eigenen Befehl ganz einfach:

Taylor-Entwicklung von f(x) am Punkt x_0 mit dem Grad n … taylor (f(x), *x*, x_0, *n*)

Mit n gleich 1 kann man auch das Differential berechnen.

Berechne näherungsweise den Sinus bei 35° = $\frac{7}{36}\,\pi$ rad mit Hilfe einer Taylorreihe vom Grad 3!

Als x_0 wählt man am besten $\frac{\pi}{6}$.

```
(%i1) taylor (sin(x), x, %pi/6, 3);
```

$$(\%o1)\quad \frac{1}{2}+\frac{\sqrt{3}\left(x-\frac{\pi}{6}\right)}{2}-\frac{\left(x-\frac{\pi}{6}\right)^2}{4}-\frac{\sqrt{3}\left(x-\frac{\pi}{6}\right)^3}{12}+\ldots$$

```
(%i2) ev(%o1,x=7*%pi/36),numer;
  (%o2) 0.57357519191346
```

Der Sinus von 35° beträgt also ungefähr 0,5736.

7.6 Stochastik

7.6.1 Kombinatorik

Eine Permutation ohne Wiederholung berechnet man mit der Fakultät:

x Fakultät …x!

```
(%i1) 23!;
  (%o1) 25852016738884976640000
```

Eine Kombination ohne Wiederholung berechnet man mit *binomial*:

$\binom{n}{k}$ … $binomial(n,k)$

```
(%i2) binomial(10,2);
  (%o2) 45
```

7.6.2 Binomialverteilung

Beim Rechnen mit Verteilungen muss man zuerst eine Erweiterung laden (vgl. 8.2):

Für Verteilungen…load(distrib)

Jetzt kann man die Wahrscheinlichkeit, dass ein Ereignis E k-mal auftritt, berechnen, wenn in einem n-mal wiederholten Zufallsversuch als Ausgänge nur E (mit der Wahrscheinlichkeit p) und sein Gegenereignis (mit der Wahrscheinlichkeit 1-p) zugelassen sind.

Eine Zufallsvariable H ist daher binomialverteilt, wenn sie angibt wie oft E auftritt.

$P(H=k) = b_{n,p}(k) = \binom{n}{k} p^k (1-p)^{n-k}$…pdf_binomial(k,n,p)

Nimmt man als Beispiel die Werte n = 10, p = 0,3 und k = 2 tritt das mit folgender Wahrscheinlichkeit auf:

```
(%i1) load(distrib)$

(%i2) pdf_binomial(2,10,.3);
(%o2) 0.2334744405
```

Will man die Wahrscheinlichkeit berechnen, wenn sich k in einem Intervall bewegt, kombiniert man *pdf_binomial* mit dem Summenbefehl

$P(k_1 \leq H \leq k_2) = \sum_{k=k1}^{k2} b_{n,p}(k)$ = …sum(pdf_binomial(k,n,p),k,k_1,k_2)

Die Berechnung von $P(2 \leq H \leq 6)$ sieht dann so aus:

```
(%i3) sum(pdf_binomial(k,10,0.3),k,2,6);
(%o3) 0.8400995757
```

7.7 Komplexe Zahlen

7.7.1 Eingabe und Ausgabe komplexer Zahlen

Imaginäre Einheit *i*…%i

```
(%i1) komplex:5/2-3/4*%i;
                          5   3 %i
(%o1)                     - - ----
                          2    4
```

Um auf den Imaginär- bzw. den Realteil zuzugreifen, gibt es folgende Funktionen:

Imaginärteil der komplexen Zahl x…imagpart(x)
Realteil der komplexen Zahl x…realpart(x)

```
(%i2) imagpart(komplex);
                          3
(%o2)                     - -
                          4
```

```
(%i3) realpart(komplex);
   (%o3)  5/2
```

Man muss beim Rechnen mit komplexen Zahlen in Maxima beachten, dass sich einige Befehle ändern. Die Wichtigsten, wie etwa die Grundrechnungsarten und *solve*, funktionieren aber unverändert.

7.7.2 Rechnen und Umwandeln

```
(%i1) z1:5+3*%i$
      z2:1-%i$
```

Addition:

```
(%i3) z1+z2;
   (%o3)  2 %i+6
```

Subtraktion:

```
(%i4) z1-z2;
   (%o4)  4 %i+4
```

Multiplikation:

```
(%i5) z1*z2;
   (%o5)  (1-%i)(3 %i+5)
```

Um das Ergebnis in kartesische Koordinaten umzuwandeln, muss man noch einen anderen Befehl anwenden:

Kartesische Koordinaten der komplexen Zahl z...rectform(z)

```
(%i6) rectform(z1*z2);
   (%o6)  8-2 %i
```

Division:

```
(%i7) z1/z2;
```

$$(\%o7)\quad \frac{3\,\%i+5}{1-\%i}$$

Will man das Ergebnis der Division dann in Polarform, geht man so vor:

Polarform der komplexen Zahl z...polarform(z)

```
(%i8) polarform(z1/z2);
```

$$(\%o8)\quad \sqrt{17}\;\%e^{\%i\,\mathrm{atan}(4)}$$

Potenzrechen:

```
(%i9) z1^z2;
```

$$(\%o9)\quad (3\,\%i+5)^{1-\%i}$$

```
(%i10) polarform(z1^z2),numer;
```

Wurzelziehen:

```
(%i11) z1^(1/5);
```

$$(\%o11)\quad (3\,\%i+5)^{1/5}$$

```
(%i12) rectform(z1^(1/5)),numer;
(%o12) 0.15348395747907 %i+1.414510563161848
```

Maxima gibt als Lösung aber nur eine Wurzel an.

7.7.3 Rotationen mit komplexen Zahlen

Rotiere den Punkt A (6|5) um $\frac{\pi}{6}$ mit dem Rotationszentrum Z (2|1)!

Zuerst wandelt man die Koordinaten von A und Z in komplexe Zahlen um:

```
(%i1) A:6+5*%i$
      Z:2+%i$
```

Mit der bekannten Formel kann man nun das Ergebnis berechnen.

```
(%i3) Aneu:rectform((A-Z)*exp(%i*%pi/6)+Z);
```
$$(\%o3)\quad (2\sqrt{3}+3)\,\%i+2\sqrt{3}$$

```
(%i4) x:realpart(Aneu);
      y:imagpart(Aneu);
```
$$(\%o4)\quad 2\sqrt{3}$$
$$(\%o5)\quad 2\sqrt{3}+3$$

Der rotierte Punkt A_{neu} hat also die Koordinaten $(2\sqrt{3} \mid 2\sqrt{3} + 3)$.

7.8 Implementieren einfacher Algorithmen

7.8.1 Newton-Verfahren

Dieses Verfahren dient zum nummerischen Aufsuchen von Nullstellen, wenn die Ableitung der Funktion überall existiert und die Nullstellen ungefähr bekannt sind.

Für diesen Algorithmus ist eine Schleife notwendig. Man könnte diese einfach in den Input schreiben, aber dann sind Variablen, die dieser Algorithmus definiert, global und behindern vielleicht andere Rechnungen. Besser ist es die Schleife in einen sogenannten Block zu schreiben, in dem die Variablen lokal definiert werden.

Block verschiedener lokaler Variablen und Ausdrücke ...
block ([x, y], Ausdruck1, Ausdruck2)

Hier verwende ich eine *While*-Schleife, die folgenden Aufbau hat:

Während eine Bedingung A erfüllt ist, wird der Befehl B ausgeführt...while A do [B]

Um eine Schleife zu verlassen und ein Ergebnis auszugeben verwendet man *return*:

Ausgabe des Wertes x ... return(x)

Jetzt folgt meine Implementierung des Newton-Verfahrens, wobei ich die Genauigkeit fix auf 10^-10 festlege. Als Input ist dann eine Funktion von x und ein Punkt in der Nähe der Nullstelle notwendig.

```
(%i1) newton(Funktion,Punkt):=
        block([ a1:Punkt,a2,d:1,f,a],
         define(f(x),Funktion),
         define(a(x),diff(Funktion,x)),
```

```
 while d>.0000000001
  do [a2:(a1-f(a1)/a(a1)),
      d:abs(a2-a1),
      a1:a2],
return(a1)
)$
```

Man weiß, dass die Nullstellen in der Nähe von 1.5, 3.0 und 6.0 liegen. Das *.0* ist wichtig weil sonst exakt gerechnet wird und ein sehr langer Term entsteht.

Die Nullstellen sind also näherungsweise:

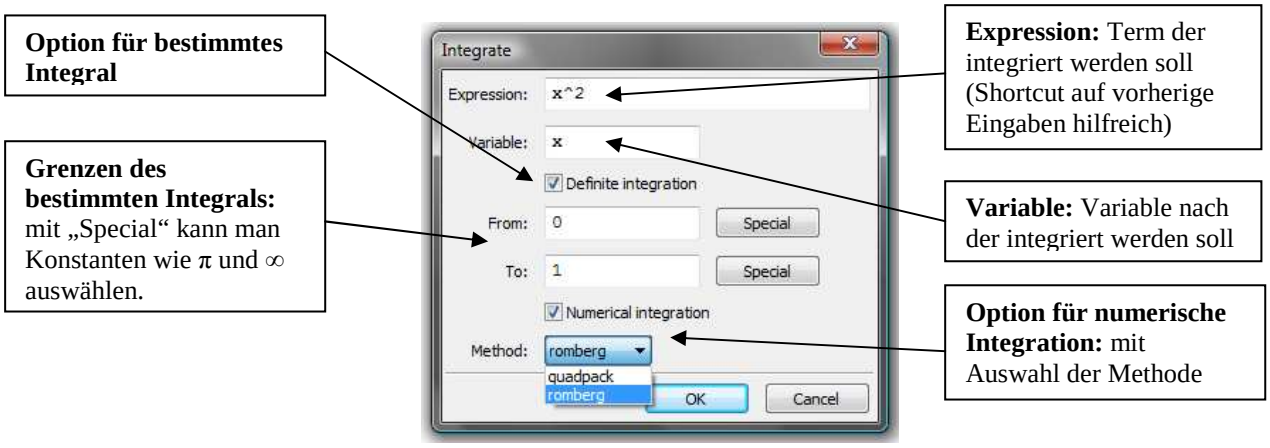
```
(%i2) newton(sin(x)+log(x)-1,1.5);
  (%o2) 1.109956375580268

(%i3) newton(sin(x)+log(x)-1,3.0);
  (%o3) 3.353028471513239

(%i4) newton(sin(x)+log(x)-1,6.0);
  (%o4) 5.500889363469072
```

8 Zwölfte Schulstufe

8.1 Integralrechnung

8.1.1 Integrationsbefehl

Natürlich bietet Maxima auch eine Funktion zum Integrieren. Man gibt den Befehl entweder direkt ein oder besser man ruft ein Dialogfeld über das Menü *Calculus-Integrate...* bzw. über die Symbolleiste *General Math* auf, um bei der Eingabe Tippfehler zu vermeiden.



Der Befehl für ein unbestimmtes Integral nimmt nun folgende Form an (im Dialogfeld werden die Argumente einzeln eingegeben):

$$\int x^5\,dx = \textbf{integrate(x\^5, x)}$$

```
(%i1) integrate(x^5, x);
                              6
                             x
(%o1)                        --
                             6
```

Für ein bestimmtes Integral werden einfach noch die Grenzen angefügt bzw. im Menü die Option "Definite Integration" ausgewählt:

$$\int_0^{\pi} \sin x\,dx = \textbf{integrate(sin(x), x, 0, } \pi\textbf{)}$$

```
(%i2) integrate(sin(x), x, 0, %pi);
(%o2)                         2
```

Maxima gibt eine Integrationskonstante nur an, wenn eine komplette Gleichung (also beidseitig) integriert wird:

```
(%i3) integrate(x=1, x);
```

$$(\%o3)\quad \frac{x^2}{2} = x + \%c1$$

8.1.2 Numerisches Integrieren

Sollte das Integral einer bestimmten Funktion nicht exakt berechenbar sein, gibt Maxima diese nicht integriert (höchstens vereinfacht) wieder aus.

```
(%i7) integrate(sin(x)/x, x,1,2);
```

$$(\%o7)\quad \int_1^2 \frac{\sin(x)}{x}\,dx$$

Sucht man nach einem solchen bestimmten Integral, muss man für ein ungefähres Ergebnis ein numerisches Lösungsverfahren verwenden. Maxima bietet hierfür im Menü (Calculus-Integrate...) eine Option für die Verfahren "quadpack" und "romberg"[10]. Mit "romberg" würde das Ergebnis des obigen Integrals etwa lauten:

```
(%i6) romberg(sin(x)/x, x, 1, 2);
(%o6)  0.65933121094521
```

8.1.3 Ober- und Untersummen

Will man zum Beispiel die Fläche eines Vierteleinheitskreises berechnen, stellt man zuerst den Funktionsterm f(x) für den Teil des Kreises über der x-Achse auf:

Kreisgleichung: $\mathbf{x^2 + y^2 = 1 \rightarrow f(x) = \sqrt{1 - x^2}}$

Nun definiert man die nötigen Parameter und die Funktion f(x): Die Summen sollen je aus 500 Summanden bestehen, daher n=500. Um den Viertelkreis zu berechnen kann man als Grenzen a und b zum Beispiel 0 und 1 verwenden. f(x) folgt aus obiger Überlegung.

```
(%i1) n:500$
      a:0$
      b:1$
      f(x):=sqrt(1-x^2)$
```

[10] Die genauen Funktionsweisen übersteigen den Umfang dieser Arbeit, daher wird nur die Anwendung beschrieben.

Nun berechnet man die Obersumme, wobei man das exakte Ergebnis unterdrücken sollte, weil Maxima sonst alle 500 Summanden anzeigt. Durch Drücken von Strg+F erhält man das ungefähre Ergebnis.

```
(%i5) (b-a)/n*sum(f(a+i*(b-a)/n),i,0,n-1)$

(%i6) float((b-a)/n*sum(f(a+i*(b-a)/n),i,0,n-1)), numer;
      (%o6)  0.78637186925053
```

Analoges gilt nun für die Berechnung der Untersumme.

```
(%i7) (b-a)/n*sum(f(a+i*(b-a)/n),i,1,n)$

(%i8) float((b-a)/n*sum(f(a+i*(b-a)/n),i,1,n)), numer;
      (%o8)  0.78437186925053
```

Zur Überprüfung kann man sich über die Flächenformel des Kreises auch ein exakteres Ergebnis ausrechen:

$A_{Kreis} = r^2 \pi$ mit r=1 → $A_{Viertelkreis} = \frac{\pi}{4}$

```
(%i9) float(%pi/4), numer;
      (%o9)  0.78539816339745
```

Die Ergebnisse der Ober- und Untersummen unterscheiden sich also ab der dritten Nachkommastelle vom exakt berechneten Wert.

8.1.4 Flächenberechnung

Berechnen die Fläche die der Graph von $f(x) = x^3 - 9x^2 + 18x$ mit der x-Achse einschließt![11]
Zuerst werden die Nullstellen der Funktion gesucht, um die Flächen einzeln ausrechnen zu können.

```
(%i1) solve(x^3-9*x^2+18*x=0, x);
      (%o1)  [x=6,x=3,x=0]
```

Jetzt kann man die bestimmten Integrale zwischen den Nullstellen berechnen.

[11] Vgl.: Malle, Günther u.a.: Mathematik verstehen 8. Wien: öbvhpt, 2007. S.28 - Bsp. 2.02.

```
(%i2) integrate(x^3-9*x^2+18*x, x, 0, 3);
```

$$(\%o2)\quad \frac{81}{4}$$

```
(%i3) integrate(x^3-9*x^2+18*x, x, 3, 6);
```

$$(\%o3)\quad -\frac{81}{4}$$

Addiert man die Beträge der Teilflächen, erhält man die Gesamtfläche.

```
(%i4) abs(%o2)+abs(%o3);
```

$$(\%o4)\quad \frac{81}{2}$$

Die Gesamtfläche beträgt also $\frac{81}{2}$.

8.1.5 Oberfläche von Rotationskörpern

Dass für den Integrationsbefehl von Maxima auch Kreisintegrale kein Problem darstellen, zeigt das folgende Beispiel, das man händisch nur durch mehrfaches Substituieren lösen kann.

Berechne die Oberfläche des Ellipsoids, das entsteht, wenn man die Ellipse $x^2 + 4\,y^2 = 4$ um die x-Achse rotiert![12]

Zuerst löst man die Gleichung nach y, um sie in eine Funktion von x zu verwandeln.

```
(%i1) solve(x^2+4*y^2=4,y);
```

$$(\%o1)\quad \left[y=-\frac{\sqrt{4-x^2}}{2},y=\frac{\sqrt{4-x^2}}{2}\right]$$

Man definiert eine Funktion y(x) gleich dem positiven Lösungsterm für y.
Um den Term direkt aus dem Output zu entnehmen, wendet man zwei „Tricks“ an.

%o1[n]...entnimmt das n-te Element der Liste im Output 1.

und

[12] Vgl.: Malle: Mathematik verstehen 8. 2007. S.68 - Bsp. 4.37 b).

Right-hand-side...rhs(a=b)...entnimmt den Ausdruck der in einer Gleichung rechts vom Gleichzeichen steht (b).

bzw.

Left-hand-side...lhs(a=b)...entnimmt den Ausdruck der in einer Gleichung links vom Gleichzeichen steht (a).

```
(%i2) y(x):=rhs(%o1[2])$
```

Im nächsten Schritt werden die Nullstellen gesucht, um die Grenzen des Integrals festlegen zu können.

```
(%i3) solve(y(x)=0,x);
(%o3)  [x = -2, x = 2]
```

Mit der bekannten Formel für die Oberfläche eines Rotationskörpers berechnet man nun das Ergebnis. Zuerst exakt:

```
(%i4) 2*%pi*integrate(y(x)*sqrt(1+(diff(y(x),x,1))^2), x, -2, 2);
(%o4)  (2 %pi (4 sqrt(3) %pi + 9)) / 9
```

Dann näherungsweise:

```
(%i5) float(%), numer;
(%o5)  21.47843532788373
```

8.1.6 Schwerpunkt einer Fläche

Berechne die Koordinaten (s|t) des Schwerpunkts der von der Funktion f mit f(x) = $\sqrt{x}$ im Intervall [0; 4] festgelegten Fläche![13]

Eine Möglichkeit ist es, zuerst die bekannten Formeln für die Schwerpunktkoordinaten allgemein anzuschreiben (wobei die Grenzen mit a und b bezeichnet werden).

[13] Vgl.: Malle: Mathematik verstehen 8. 2007. S.71 - Bsp. 4.39.

```
(%i1) s:integrate(x*f(x), x, a,b)/integrate(f(x), x,a,b);
```

$$(\%o1)\quad \frac{\int_a^b x\,f(x)\,dx}{\int_a^b f(x)\,dx}$$

```
(%i2) t:1/2*integrate(f(x)^2, x, a,b)/integrate(f(x), x,a,b);
```

$$(\%o2)\quad \frac{\int_a^b f(x)^2\,dx}{2\int_a^b f(x)\,dx}$$

Nun braucht man nur noch alle Unbekannten zu definieren und die Berechnungen erneut durchführen.

```
(%i3) f(x):=sqrt(x)$
      a:0$
      b:4$

(%i6) s:integrate(x*f(x), x, a,b)/integrate(f(x), x,a,b);
```

$$(\%o6)\quad \frac{12}{5}$$

```
(%i7) t:1/2*integrate(f(x)^2, x, a,b)/integrate(f(x), x,a,b);
```

$$(\%o7)\quad \frac{3}{4}$$

Die Schwerpunktkoordinaten sind also $(\frac{12}{5} \mid \frac{3}{4})$.

8.2 Stochastik

Um auf die speziellen Befehle, die das Berechnen von Verteilungen beschleunigen, zugreifen zu können, muss zuerst das Package „distrib“ geladen werden. Zur Berechnung von Normalverteilungen muss man mit Maxima nicht selbst auf standardisierte Kurven umrechnen, sondern kann μ und σ einfach eingeben.

Für Verteilungen…load(distrib)

8.2.1 Dichtefunktion der Normalverteilung

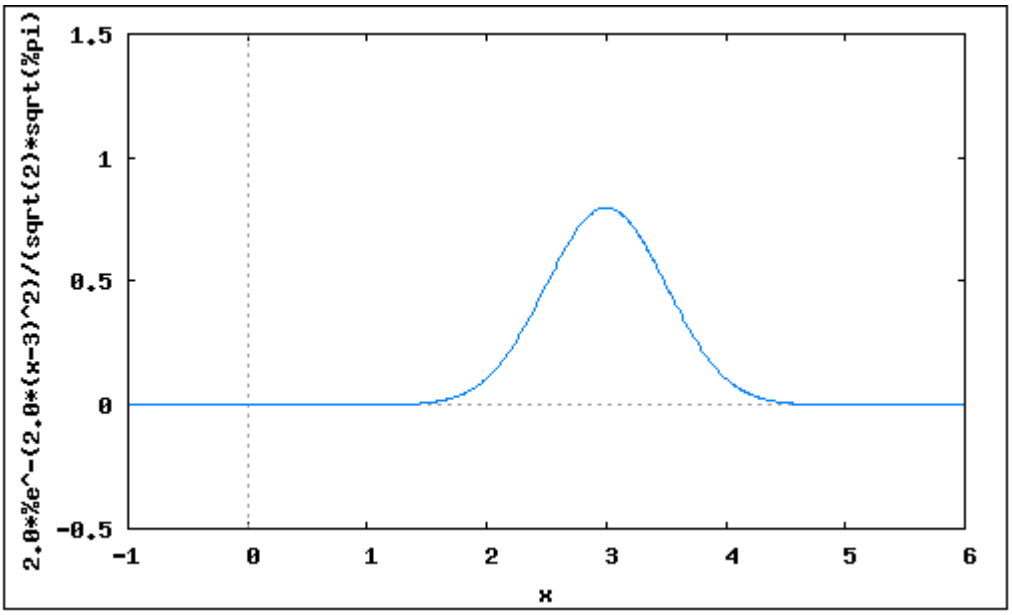
Um Einzelereignisse eines normalverteilten Zufallversuchs zu berechnen, verwendet man die folgende Funktion:

$$\varphi_{\mu,\sigma}(x) := \frac{1}{\sqrt{2\pi}\,\sigma}\, e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \ldots\ldots \text{pdf\_normal}(x,\mu,\sigma)$$

Zur Veranschaulichung kann man die Funktion mit μ=3 und σ=0,5 plotten lassen:

```
(%i2) wxplot2d([pdf_normal(x,3,0.5)], [x,-1,6], [y,-0.5,1.5])$
```

(%t2)



8.2.2 Verteilungsfunktion der Normalverteilung

Zur Berechnung der Wahrscheinlichkeit eines Intervalls verwendet man $\Phi_{\mu,\sigma}(x)$. Dabei berechnet Maxima die Fläche unter der Gauß´schen Glockenkurve zwischen -∞ und x.

$$\Phi_{\mu,\sigma}(x) := \frac{1}{\sqrt{2\pi}\,\sigma} \int_{-\infty}^{x} e^{-\frac{1}{2}z^2}\, dz \ldots\ldots \text{cdf\_normal}(x,\mu,\sigma)$$

Eine Zufallsvariable X sei normalverteilt mit μ=15 und σ=2. Berechne P(X≤ 16)!

```
(%i1) load(distrib)$

(%i2) cdf_normal(16,15,2),numer;
  (%o2)  0.69146246127401
```

Zum Vergleich kann man das Integral der standardisierten Dichtefunktion auch direkt ausrechnen. Das dauert bei einfachen Beispielen nicht länger, doch für komplexere Aufgaben sollte man die speziellen Befehle verwenden. Außerdem wird durch die eigenen Befehle die Eingabe erleichtert.

```
(%i3) 1/sqrt(2*%pi)*integrate(%e^(-1/2*z^2), z,minf,((16-15)/2))$

(%i4) float(%), numer;
  (%o4)  0.69146246127401
```

Die Wahrscheinlichkeit, dass X≤ 16 ist, beträgt ungefähr 69,15%.

8.2.3 Umkehraufgaben

Um von bekannten Wahrscheinlichkeiten zurück zu rechnen, bietet Maxima bei normalverteilten Intervallen eine eigene Funktion:

quantile_normal (cdf_normal(x,μ,σ), μ,σ) = x

Beim Zurückrechnen des vorherigen Beispiels (8.2.2) zeigt sich dann auch die Genauigkeit von Maxima.

```
(%i6) quantile_normal(0.69146246127401,15,2),numer;
  (%o6)  15.99999999999998
```

Aufgrund der gerundeten Eingabe erhält man nicht ganz den Ausgangswert.

8.2.4 Berechnung von μ oder σ bei vorgegebener Wahrscheinlichkeit

Leider bietet Maxima hierfür keine eigene Umkehrfunktion. Man muss also den Umweg über die standardisierte Normalverteilung mit μ = 0 und σ = 1 gehen. Zuerst ermittelt man das z mit *quantile_normal* und dann rechnet man mit folgender Formel entweder μ oder σ aus.

Wenn quantile_normal(Wahrscheinlichkeit, 0, 1) = z, dann gilt z = $\frac{x-\mu}{\sigma}$

8.2.5 Näherung an Binomialverteilungen

Berechne nährungsweise mit der Normalverteilung$\binom{1200}{200}\left(\frac{1}{6}\right)^{200}\left(\frac{5}{6}\right)^{1000}$!

Gegeben sind also n=1200, p=$\frac{1}{6}$ und q=$\frac{5}{6}$. Daraus folgt μ = n*p = 200 und σ²=n*p*q=$\frac{500}{3}$. Da σ² größer 9 ist, kann man hier die Binomialverteilung auf eine Normalverteilung umrechen.

```
(%i2) pdf_normal(200,200,sqrt(500/3));
```

$$(\%o2)\quad \frac{\sqrt{3}}{2^{3/2}\,5^{3/2}\sqrt{\pi}}$$

Strg+F

```
(%i3) float(%), numer;
(%o3) 0.030901936161855
```

Das Ergebnis von $\binom{1200}{200}(\frac{1}{6})^{200}\,(\frac{5}{6})^{1000}$ ist also ungefähr gleich 0,031.

8.2.6 Wahrscheinlichkeit in Intervallen

Eine Maschine erzeugt Platten, deren Dicke mit μ = 2 und σ = 0,02 normalverteilt ist. Eine Abweichung von μ größer als 0,05 ist Ausschuss.
a) Wie viel Prozent Ausschuss sind zu erwarten?
b) Durch Verschleiß hat sich μ auf 2,01 verstellt. Wie viel Ausschuss ist jetzt zu erwarten?

a) Mit der Formel für ein symmetrisches Intervall berechnet man P(|X-μ| ≤ 0,05), also die Wahrscheinlichkeit, dass eine Platte kein Ausschuss ist.

```
(%i2) 2*cdf_normal(2.05,2,0.02)-1,numer;
(%o2) 0.98758066934845
```

Für das Ergebnis berechnet man dann die Gegenwahrscheinlichkeit in Prozent.

```
(%i3) (1-%o2)*100;
(%o3) 1.241933065155276
```

Ungefähr 1,24% aller Platten sind Ausschuss.

b) Da hier kein symmetrisches Intervall mehr vorliegt, muss die Wahrscheinlichkeit anders berechnet werden: P(1,95 ≤ X ≤ 2,05) = $\Phi_{2,01;0,02}(2,05) - \Phi_{2,01;0,02}(1,95)$. Wieder erhält man die Wahrscheinlichkeit, dass eine Platte kein Ausschuss ist.

```
(%i4) cdf_normal(2.05,2.01,0.02)-cdf_normal(1.95,2.01,0.02),numer;
(%o4) 0.97589997002019
```

Nun berechnet man wieder die Gegenwahrscheinlichkeit in Prozent.

```
(%i5) (1-%o4)*100;
  (%o5) 2.410002997980931
```

Etwa 2,41% aller Platten sind durch den Verschleiß Ausschuss.

8.2.7 Hypothesentest

Ein Verkehrsunternehmen ist von der Annahme ausgegangen, dass 10% der Fahrgäste Schwarzfahrer sind (H_0: p=0,1). Ein Kontrollor soll 500 Fahrgäste zufällig überprüfen. Bei welchen Ergebnissen kann diese Annahme verworfen werden (α_0=0,05)?

Hinweis: Eigentlich handelt es sich bei diesem Beispiel um eine hypergeometrische Verteilung. Obwohl Maxima hierfür einen eigenen Befehl[14] hätte, verwende ich aufgrund des Oberstufenlehrplans eine Näherung über Binomial- und Normalverteilung.

Die Parameter dieses Beispiels sind: n=500 und p=0,1. Der σ^2-Test ergibt, dass die Näherung durch die Normalverteilung zulässig ist. Man berechnet nun die Grenzen eines symmetrischen Intervalls, bei dem $\frac{\alpha 0}{2}$ je links und rechts außerhalb der Grenzen liegt.

Hinweis: Maxima kann nur sehr eingeschränkt griechische Buchstaben verwenden, deshalb verwende ich die deutschen Äquivalente.

Man berechnet zuerst μ und σ und dann die untere und obere Grenze, bei denen H_0 verworfen werden kann.

```
(%i2) m:500*0.1$
      s:sqrt(500*0.1*0.9)$

(%i4) quantile_normal(0.025,m,s),numer;
  (%o4) 36.85216189135127

(%i5) quantile_normal(0.975,m,s),numer;
  (%o5) 63.14783810864873
```

[14] pdf_hypergeometric( ) bzw. cdf_hypergeometric( )

Man kann die Hypothese H_0 also bei Stichproben bis 36 oder ab 64 mit α_0=0,05 verwerfen.

8.2.8 Konfidenzintervall

In einer Stichprobe vom Umfang 650 findet man 201 Personen vor, die eine bestimmte Partei wählen wollen. Gib nun ein 95%- Konfidenzintervall (α_0=0,05) für den unbekannten relativen Anteil p der Wähler dieser Partei in der Grundgesamtheit aller Wähler an.[15]

Zuerst berechnet man z mit der standardisierten Verteilungsfunktion: $\Phi_{0,1}(z) = 1 - \frac{\alpha 0}{2}$

```
(%i2) z:quantile_normal(0.975,0,1)$
```

Dann kann man den Abstand ε um $\hat{p} = \frac{201}{650}$ mit der „Worst-Case-Methode“ ausrechnen.

```
(%i3) eps:z*1/2*sqrt(1/650)$
```

Für das Konfidenzintervall ergibt sich so:

```
(%i4) [(201/650)-eps,(201/650)+eps],numer;
  (%o4)  [0.27079271306497,0.34766882539657]
```

95%-Konfidenzintervall: $\hat{p} \in [0{,}27;\ 0{,}35]$

[15] Vgl.: Malle: Mathematik verstehen 8. 2007. S.116 - Bsp. 6.44.

9 Nachwort

Beim Verfassen dieser Arbeit war ich mit dem Problem konfrontiert, dass ich eine Anleitung über etwas schreiben wollte, von dem ich bis vor einem Jahr überhaupt keine Ahnung hatte. Ich musste also zuerst viele kleine und größere Probleme im Umgang mit dem Programm lösen, bevor ich die Beispiele aus den einzelnen Schulstufen relativ gut durchrechnen konnte.

Vielleicht sind einige meiner Berechnungen nicht die elegantesten und Maxima hätte wohl noch den einen oder anderen Spezialbefehl für meine Zwecke bereitgestellt. Durch die vielen Möglichkeiten die Maxima bietet, war es aber von Anfang an auch verwirrend.

Mit dieser Anleitung habe ich es aber hoffentlich geschafft, alle für eine AHS-Oberstufe wichtigen Befehle zu sammeln, und so für Schüler und Lehrer eine Alternative zu langem Suchen in der Hilfe oder im Internet zu schaffen.

Bad Zell, am 22.Februar.2010 *Manuel Resch*

10 Quellenverzeichnis

Einführung:

- http://www.austromath.at/daten/maxima/index.htm [Stand 2010-01-26]
- http://maxima.sourceforge.net/ [Stand: 2010-01-26]
- http://maxima.sourceforge.net/relatedprojects.html [Stand: 2010-01-26]
- http://packages.debian.org/de/squeeze/wxmaxima [Stand: 2010-01-26]

Installation:

- http://www.lehrer-online.de/maxima.php [Stand: 2010-01-26]

Rechenbeispiele:

- Götz, Stefan u.a.: Mathematik. Lehrbuch 5. Wien: öbvhpt, 2004.
- Malle, Günther u.a.: Mathematik verstehen 6. Wien: öbvhpt, 2006.
- Malle, Günther u.a.: Mathematik verstehen 7. Wien: öbvhpt, 2007.
- Malle, Günther u.a.: Mathematik verstehen 8. Wien: öbvhpt, 2007.

Befehle und Details zu Maxima:

- Maxima-Hilfe
- Wegscheider, Walter: Maxima 5.xx. Workshop – Computeralgebrasystem. online im Internet: http://www.austromath.at/daten/maxima/index.htm [Stand: 2010-02-21].

11 Begleitprotokoll

Jun.08	Prof. Angsüsser schlägt vor, dass ich eine FBA schreiben könnte.
Sep.08	Beginn der Themensuche.
Im Lauf des Schuljahres	Diskussionen über verschiedene Themen (unter anderem auch wxMaxima).
13., 20., 28.05.2009	FBA-Vorbereitungskurs mit Prof. Elke Gusenbauer.
Jun.09	Festlegung auf "wxMaxima-Kurs" als Thema.
Sommerferien 2009	Erste Versuche mit Maxima zu rechnen.
	Mailverkehr mit Prof. Angsüsser über Aufbau, Arbeitssprache und Quellen.
16.09.2009	Antrag zur Bewilligung der FBA.
29.09.2009	Treffen mit Prof. Angsüsser - Festlegen des Aufbaus / Tipps zum Export.
18.10.2009	Einrichten der Formatvorlagen, rechnen ausgewählter Beispiele.
20.10.2009	Treffen mit Angsüsser - Besprechen von Zwischenstand.
28.10.2009	Auswahl der Beispiele für die Neunte Schulstufe.
01., 02.11.2009	Durchrechnen der Beispiele für die Neunte Schulstufe.
05., 24.11.2009	Ergänzungen zur Neunten Schulstufe.
25.11.2009	Treffen mit Angsüsser - Beispiele besprechen.
05., 06., 07., 08.12.2009	Auswahl und Rechnen der Beispiele für die Zehnte Schulstufe.
14.12.2009	Rechnen mit Gleichungen und Funktionen.
18.12.2009	Treffen mit Angsüsser - Besprechen einiger Befehle
Weihnachtsferien 09/10	Auswahl und Rechnen der Beispiele für die Elfte Schulstufe und das Kapitel erste Schritte.
03.01.2010	Endgültiges Festlegen von Aufbau und Formatvorlagen.
10.01.2010	Internetrecherche zur Geschichte von Maxima.
25.01.2010	Arbeit an den Kapiteln Maxima als TR, Einführung und 9. Schulstufe
26.01.2010	Fertigstellen des Kapitels Erste Schritte, Arbeit an Einführung und Neunte Schulstufe.
01.02.2010	Arbeit an Integralrechnung.
02.02.2010	Arbeit an Integralrechnung.
06.02.2010	Arbeit an Titelblatt, Einführung, Maxima als TR, Zwölfte Schulstufe.
07.02.2010	Arbeit an Zwölfter Schulstufe.
09.02.2010	Letztes Treffen mit Prof. Angsüsser.
14.02.2010	Arbeit an Neunter Schulstufe.
15.02.2010	Verfassen des Kapitels Exkurse.
17.02.2010	Verfassen von Vorwort und Geschichte.
19.02.2010	Zusammenstellen der einzelnen Kapitel. Verfassen von Installation; Bearbeiten von Erste Schritte, Neunte Schulstufe, Zwölfte Schulstufe.
20.02.2010	Bearbeiten von Zehnter Schulstufe.
21.02.2010	Verfassen von Nachwort; Zusammenstellung der Quellen, des Registers und des Protokolls; Drucken der FBA.
22.02.2010	Abgabe der FBA.

BUNDESOBERSTUFENREALGYMNASIUM PERG
4320 PERG, DIRNBERGERSTR. 43 • TEL: 07262 52257 • FAX: 07262 52257-15
borg.perg@eduhi.at • www.borg-perg.eduhi.at

12 Erklärung zur Fachbereichsarbeit

Manuel Resch	8.d	2009/10
Name der Schülerin/des Schülers	Klasse	Schuljahr

Thema der Fachbereichsarbeit:
Kurs für das Computer Algebra System „wxMaxima“ im Mathematikunterreicht der AHS-Oberstufe.

Ich erkläre, dass ich die Fachbereichsarbeit ohne fremde Hilfe verfasst und dazu nur die angegebene Literatur verwendet habe. Ich habe die Arbeit einer Korrektur unterzogen und Tippfehler ausgebessert.

Perg, am 22.02.2010
Ort und Datum der Abgabe

_____
Unterschrift des Schülers