Jetzt können bereits Schüler CAS nutzen

– und damit die Grundlagen schaffen, um später auch als Student oder gar Professor Mathematik zu betreiben –

Da jetzt schon für wenig Geld Laptops zur Verfügung stehen, können nun Schüler kostenlos MAXMA nutzen und sich für Ihre Zukunft in Mathematik rüsten – und wenn eine einfache Anleitung für den ersten Gebrauch zur Verfügung steht, ist das Bewältigen mit dem System so einfach wie der Gebrauch eines Taschenrechners.

Diese Software ist unter verschiedenen Betriebssystemen nutzbar, nämlich Windows, Mac und Linux. Anfänglich ist die Bedienung etwas gewöhnungsbedürftig, aber wenn diese Anleitung für Schüler (und Studenten) erst einmal die Einstiegshürde beseitigt hat – und diese Anleitung ist einfacher, als und sie komplizierten Abhanlungen und Totorien zunächst glauben machen – dann ist der Startschuss für eine erfolgreiche Mathematik gegeben! Jetzt kann man immer noch die Feinheiten nutzen, die uns das Tutorium und die komplizierte Anleitung ermöglichen.

Alle Rechnungen lassen sich in Arbeitsblätter speichern und wieder abrufen, so dass sie für weitere Rechnungen zur Verfügung stehen. Dieses Programm beherrscht nicht nur alles, was einen Taschenrechner ausmacht, sondern es beherrscht echte Computer-Algebra. Wir können also mit Buchstaben rechnen und Grafiken erzeugen. Das können auch einige andere Mathematik-Programme wie Derive, Mathcad, Maple und Mupad, aber Maxima ist für jeden nutzbar, weil kostenlos.

Gemäß Wikipedia ist Maxima eine Version vom Macyma, einem der ersten Computeralgebra - Systeme überhaupt. Es wurde schon früh, in den 1960er Jahren im Auftrag des US-Energieministeriums (DOS) am MIT entwickelt. Eine Macsyma-Version (DOE Macsyma) wurde von William Schelter von 1982 bis zu seinem Tod 2001 weiterentwickelt.

1998 erhielt Schelter vom Energieministerium die Genehmigung, seine Version unter der GPL zu veröffentlichen. Diese Version wird nun unter dem Namen Maxima von einer unabhängigen Gruppe von Anwendern und Entwicklern gepflegt.

Mit dem Programm wxMaxima ist darüber hinaus eine grafische Benutzeroberfläche verfügbar, die durch Menüs und Dialoge die Nutzung des Programms einfacher macht und eine grafische Formelausgabe besitzt. Ab Version 5.10.0b ist die grafische Benutzeroberfläche in das Installiationspaket von Window integriert. Doch zunächst müssen das Programm erst einmal installieren,

Unter maxima.sourceforge.net

http://sourceforge.net/project/showfiles.php?group_id=4933

findet man die Installationsroutine. Dort solle man z. B. die Windows-Version vom schweizer Server (Lausanne) wählen. Wenn Sie Maxima oder genauer **wxmaxima** auf Ihrem Computer installiert haben, dann sollten Sie erst einmal diese Anleitung lesen, bevor Sie sich ins Tutorium (z. B. unter Documentation, German Totorials,

Maxima 5.12.0 unter der Oberfläche wxMaxima 0.7.2. Workshop - Computeralgebrasystem

von Magister Walter Wegscheider stürzen oder weitergehende Literatur studieren.

Maxima als Taschenrechner

Geben Sie z. B. an:

4x5 und „return“:

(%i1) 4*5;
(%o1) 20

Als Ergebnis erhalten wir 20. Die erste Zeile stellt (%i1) für die Eingabe 4*5 und die zweite Zeile (%o1) für das Ergebnis voran, und zwar $i für input und %o für ouput. Wir können nun mit dem Ergebnis weiterrechnen, in dem wir in die nächste Zeile z. B.

(%i2) %*7;
(%o2) 140

eingeben. Dann wird die Rechnung fortgesetzt. Auch mit **F5** wird die letzte Eingabe in die neue fortgesetzt, oder mit %Zeilennummer setzen wir mit dem letzten Ergebnis fort!

(%i2) 20*7;
(%o2) 140

Wir geben nun einen Bruch ein:

(%i3) 1/5;
(%o3) 1/5

Wir sollen doch statt des Bruches jetzt die Dezimalschreibweise erhalten, dann geben wir ein:

(%i3) float(1/5);
(%o3) 0,2

oder

```
(%i2)  float(1/7);
(%o2)  0.14285714285714
```

log(x) : gibt immer den natürlichen Logarithmus an. Notfalls muss die Umrechnung auf einer andere Basis erfolgen. Die Formel hierfür:

$\mathbf{\log_a(x)} = 1/\log_e a * \log_e(x)$, also

$\log_{10}(2) = 1/\log(10) * \log(2)$

Die Zahl wird „englisch“ präsentiert, das Komma also durch einen Punkt ausgedrückt (und dieser auch eingegeben), wenn wir mehr Stellen benötigen, so können wir mit bfloat eine (und mit dem Menü Genauigkeit eine beliebige Zahl von Stellen festlegen), aber wenn es mehr als 16 werden, kommt so etwas heraus:

```
(%i5)  bfloat(1/7);
(%o5)
1.428571428571428571428571428571[44 digits]428571428571428571428571429b-1
```

Wir können jedoch die Ausgabe festlegen:

(Menüpunkt: Maxima, Formeldarstellung ändern)

xml .. Langzahlen werden aufgeteilt dargestellt; Beginn genau (nur die Anzahl der Stelle wird angegeben), Ende genau

axcii …Langzahlen werden komplett angezeigt, über den Backslash (\) wird ein Zeilenumbruch durchgeführt

none … Langzahlen werden komplett angezeigt, ohne Zeilenumbruch

asci:

(%i12) 100!;

(%o12)
933262154439441526816992388562667004907159682643816214685929638952175 99\

9932299156089414639761565182862536979208272237582511852109168640000000000000 000\

000000000

none:

(%o10)
93326215443944152681699238856266700490715968264381621468592963895217599993 2

und diese Stellen sind nicht zu sehen:

2991560894146397615651828625369792082722375825118521091686400000000000000000 00000000

Probieren Sie es einmal aus!

Mit **Str*g*-*F*** kann der ***bfloat*-Wert** auch gefordert werden.

Wir können alle Operatoren einsetzen! Dabei zählen auch Wurzel (sqrt), wenn jedoch andere Wurzeln gefragt sind, dann müssen wir uns eines Tricks bedienen: 4^(1/5) ist dabei die 5, Wurzel einer Zahl:

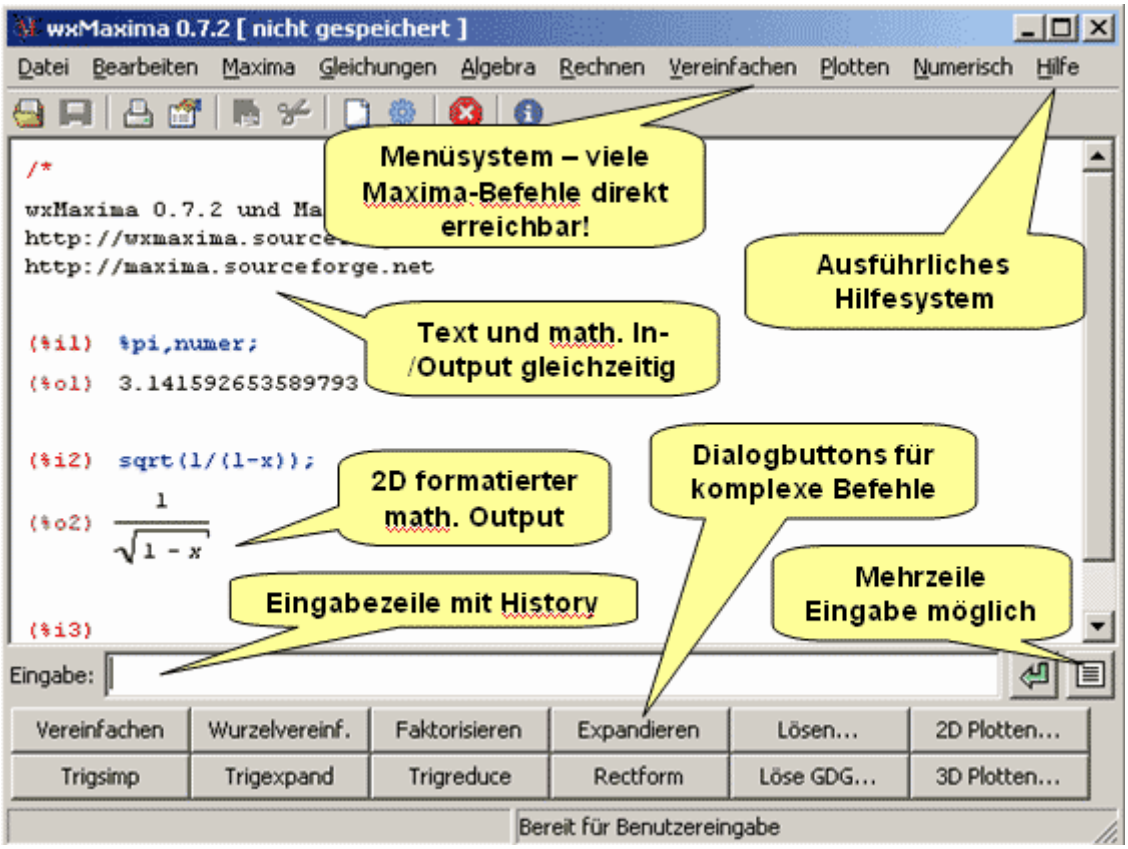
(%i1) float(4^(1/5));
(%o1) 1.319507910772894

Das Korrigieren und das Überschreiben von Eingaben

Eingaben können am einfachsten durch Doppelklick überschrieben werden. Anschließend kann die bisherige das Arbeitsblatt (*Bearbeiten = Strg-Return, re-evaluate all*) oder auch nur die Eingabe (*Bearbeiten, Eingabe neu berechnen)* neu berechnet werden.

Mit *F6* lassen sich nachträglich Text und mit *F7* Ausdrücke einfügen. Mit Shift-F6 läßt sich nachträglich großer und fetter Text einfügen.

So jetzt ist Maxima als Taschenrechner besprochen, was jetzt folgt, kann Maxima außerdem noch.! Doch zunächst einmal die Oberfläche (aus dem Tutorium von Maxima:



Der MAXIMA-Bildschirm enthält (von oben nach unten) folgende Elemente:

die Menüleiste (über die die meisten *MAXIMA*-Befehle direkt angesprochen werden können)

die Symbolleiste (bietet Windows-spezifische Funktionen wie Speichern, Drucken, ...)

das eigentliche Algebrafenster (beim Start leer bis auf einen Hinweistext)

die Eingabezeile mit einem Button, über den mehrzeilige Eingabe aufrufbar ist

die Dialogbuttons für häufig gebrauchte komplexe Maxima-Befehle, die über Dialogboxen gesteuert werden können (Vereinfachung der Eingabesyntax)

die Statuszeile mit ergänzenden Informationen

Mehrzeilige Eingabe

Bei der Eingabe komplexer Terme und vor allem bei Programmierarbeiten ist die einzellige Eingabe unübersichtlich. Hier wird der Übergang zum mehrzeiligen Eingabe-Dialogfenster emfohlen. Es wird entweder über den Menüeintrag (*Bearbeiten, Lange Eingabe*) oder mittels des *Shortcuttes Strg+I* aufgefufen:

wxMaxima speichert Daten entweder im ASCII-Format (die daher in jedem Editor oder Textverarbeitungsprogramm bearbeiten werden können (auch wenn im Format wxm ausdrücklich davor warnt wird), Sie können unter dem Menüpunkt *Bearbeiten, Öffnen ge*öffnet werden, Weitere Möglichkeiten beim Speichern sind:

save <Dateiname>, *all*,
save <Dateiname>, *values*
save <Dateiname>, *functions*

und mit load <Dateiname> werden die entsprechenden Werte wieder zur Verfügung gestellt

Allerdings ist die Speicherung unter Windows nur mit allen Daten unter dem Dateiname möglich.

Ein weiterer Dateityp ist der von mac-Dateien, der einfach geladen oder selbst erstellt wurde. Mac-Dateien sind meist Sammlungen von neuen Funktionen, wie

***vect* (einem Paket für die Vektorrechnung**
***sdnst* (Nested Roots für das Vereinfachen von verschachtelten Wurzelausdrücken)**
***implicit_plot* (2-d Grafik)**
***distrib* (Funktionen zur Normalverteilung, Binomialverteilung usw.)**
***numericalio* (Einlesen von Daten, Textfiles in Variablen)**
usw

Mit *load („vect“)$* lädt *man das* Vektor-Paket automatisch beim Start mit. (Siehe auch das Tutorial)

Variablen verwenden

MAXIMA verarbeitet Variablen in einbuchstabiger Form ebenso wie in Wortform, Groß- und Kleinschreibung sind möglich und werden unterschieden. Beispiele:

X, ya, x,a,b,SARU,SatURn

Variable können über die Eingabezeile durch „:“ mit einem Wert belegt werden und über das Menü *MAXIMA, ZEIGE VARIABLEN* können die belegten Variablen eingesehen werden. Der Befehl dazu lautet *values* (wird automatisch angezeigt). Variablen können mit dem Menü *MAXIMA, VARABLE LÖSCHEN* wieder gelöscht werden.

Auch mit kill (<*varablen*>) oder *kill(all)* können Varable gelöscht werden, dabei löscht *kill(all)* jedoch auch alle anderen Werte, wie Fuktionen usw..

Trigonometrie

wxMaxima kennt die Befehle (und ihre Umkehrbefehle):

tan - atan

cot - acot

sin - asin

cos - acos

und weitere, die hier jedoch nicht interessieren. Wichtig ist, dass die Befehle im Bogenmaß wirken, so dass für Trigonometrie eine Umrechnung erfolgen muss:

Umrechnung von Bogenmaß in Gradmaß

Gradmaß = Bogenmaß * 360 / (2 * pi)

Umrechnung von Grad in Bogenmaß:

Bogenmaß = Gradmaß * 2 * pi/360

Diese Umwandlungen sind im Arbeitsblatt Winkel-Bogenmaß vorgegeben.

Numerik - Teilbarkeit

mod(m, n) = modulo n. der nichtnegative Test von m,/n

floor(a/B) = (positive, a und b < 0) Ganzzahldivision

gdc(m1, m2) = ggT, größter gemeinsamer Teiler (auch über das Menü *Rechnen* abrufbar)

lcm(m1, m2) = kgV, kleinstes gemeinsames Vielfaches (funktioniert jedoch nicht,

vielleicht findet jemand die wirkliche Funktion heraus)

prime(m) überprüft, ob m eine Primzahl ist

next_prime(m) prev_prime(m)

die nächsten zur Zahl m gelegenen Primzahlen

Beispiel:

Berechne den größten gemeinsamen Teilen von 12566 und 833:

(%i1 gcd(432, 12564);
(%o1) 36

Vereinfachen

MAXIMA kennt keinen allemeingültigen Vereinfachungsbebefehl, sondern verwendet verschiedene Vereinfachungfunktionen.

ratsimp(a) vereinfacht den Ausdruck und gibt einen Quotienten von zwei Polynomen zurück

fullratlsimp(a)
wiederholte Ausführung von ratsimp, gefolgt von nicht-reationalen Vereinfachungen – bis keine Veränderung mehr auftritt

radcan(a) vereinfacht Ausdrücke für log-, exp, Radikale, die Differenz gleicher Ausdrücke verschienenen Aussehens die zu Null vereinfacht werden

grind(a) grind(a) stellt eine Variable oder Funktion in einer kompakten Form dar

trigsimp(a) wandelt verschiedene trigonometrische Funktionen in sin und cos Äquivalente um

trigreduce(a) vereinfacht trigonometrische Produkte und Potenzen

trigexpan(a) vereinfacht trigonometrische Produkte und Potenzen

foursimp(a) vereinfacht ganzzahlige Vielfache von Pi in Abhängigkeit verschiedener (Flags)

attsimp(a) vereinfacht algebraische Tensorausdrücke

vectorsimp(a) wendet Vereinfachungen und Expansionen bzgl. Vektoroperationen (abhängig von Flags) an.

factor(a) faktorisiert Polynome rational. Bei natürlichen Zahlen bewirkt der Befehl eine Zerlegung in Primfaktoren

expand(a) Multipliziert einen Ausdruck aus

ratexpand(a) expandiert stärker mit meist besseren Ergebnissen als expand

Besonders heikel sind verschachtelte Wurzelausdrücke, hier muss für eine Vereinfachung die Bibliothek *sgnst* mit *load(sqdnst*) gladen werden. Wie *bei trigexpand* und *logexpand* kann mittels *sqtdenest* die Struktur vor der Vereinfachung in für das CAS einfachere Bestandteile aufgebrochen werden.

Die wichtigsten Vereinfachungregeln sind mit dem Menue oder links unterhalb des Arbeitsblattes abrufbar.

Um Ausdrücke, insbesondere Polynome, weiter zu zerlegen, bietet *MAXIMA* eine Reihe von Funktionen an. Damit kann auf Zähler und Nenner eines Ausdrucks direkt zugegriffen werden bzw. es können Polynomdivisionen durchgeführt werden.

`num(u)` - gibt den Zähler eines Bruchausdrucks zurück

`denom(u)` - gibt den Nenner eines Bruchausdrucks zurück.

`divide(u, v)` - berechnet Quotienten und Rest einer Polynomdivision u/v.

`quotient(u, v)` - berechnet den Quotienten einer Polynomdivision u/v.

`remainder(u, v)` - gibt den Rest einer Polynomdivision u/v zurück

Lösen einer linearen Gleichung

Aufgabenstellung: Die lineare Gleichung a*x + b = c soll nach x aufgelöst werden. *MAXIMA* lässt uns dafür 2 Möglichkeiten offen - wir können die Gleichung mit Hilfe von Äquivalenzumformungen lösen oder mit Hilfe des eingebauten Lösungsbefehls `SOLVE` arbeiten.

Äquivalenzumformungen

```
(%i16)  a*x+b=c;
(%o16)  a x + b = c

(%i17)  (a*x+b=c)-b;
(%o17)  a x = c - b
/*
Achtung! etwas ungewohnte Schreibweise.
Über F5 einfügen der markierten Teile
in die Bearbeitungszeile.

(%i18)  (a*x=c-b)/a;
                 c - b
(%o18)       x = -----
                   a
```

SOLVE-Befehl

Der `SOLVE`-Befehl verlangt nach der zu lösenden Gleichung und der Variablen (den Variablen), nach denen die Gleichung aufgelöst werden soll!

```
(%i1) eq:s=m-k*s/(k+a);
(%o2) s=m-(k*s)/(k+a);
(%i3) solve(eq,k);
(%o4) [k=-(a*s-a*m)/(2*s-m)]
```

Übung:

Berechnen Sie aus der Gleichung $t = n - k \cdot a/(s + a)$ die Variable *s* durch Äquivalenzumformung und über den `SOLVE`-Befehl.

SOLVE und ALGSYS

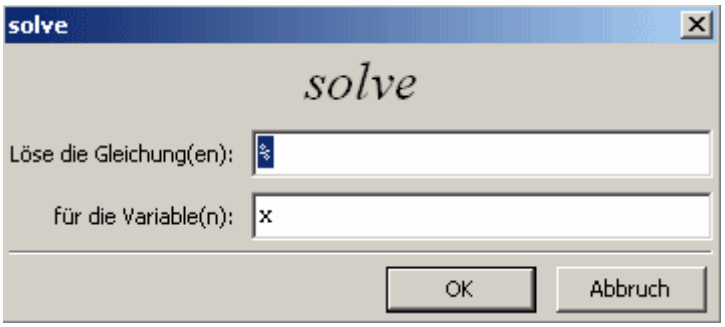
Das erste Argument von *SOLVE* ist die zu lösenden Gleichung (oder Ungleichung). Falls das Argument weder eine Gleichung noch Ungleichung ist, wird das Argument zu einer Gleichung gemacht, indem es gleich Null gesetzt wird. Das zweite Argument ist die Variable, nach der gelöst werden soll.

`SOLVE(u, v)` – ist der eigentliche Lösungsbefehl. u ist die Gleichung und v ist die Variable, nach der aufgelöst werden soll.

`ALGSYS([u], [v])` - der übergeordnete Lösungsbefehl von MAXIMA greift auf *SOLVE* zurück, wenn algebraisch eindeutig zu berechnenden Lösungen existieren mit dem Vorteil, dass hier zusätzliche Regeln wie *REALONLY* berücksichtigt werden, und greift auf numerische Verfahren zurück, wenn keine eindeutigen Lösungen möglich sind. Man beachte den ungewöhnlichen Gebrauch der Klammern.

Hinweis - den Befehl „Lösen“ (*SOLVE*) erhält man auch über das Menü (*GLEICHUNGEN - LÖSEN*) und mit Hilfe der Symbolleiste über den Button *Lösen*. Im Menü gibt es auch einen eigenen Punkt *NUMERISCH LÖSEN*, der auf den Befehl **`find_root()`** zugreift.

Bei Aufruf über das Menü öffnet sich folgender Dialog



Übungen:

Löse folgende Gleichung nach a in den Grundmengen R und C (reelle und komplexe Lösungen): $a^3 + 2 \cdot a^2 + 9 \cdot a + 18 = 0$

Löse folgende Gleichung nach m in den Grundmengen R und C:
$m^2 + 2 \cdot m + 3 = 0$

Löse die Gleichung mit Formvariablen a, b, c nach x: $a \cdot x^2 + b \cdot x + c = 0$

Einschränkung der Lösungsmenge

Die Menge, über der der *SOLVE*-Befehl von *MAXIMA* arbeitet, lässt sich nicht direkt beeinflussen. Einschränkungen über *ASSUME* etc. werden hier ignoriert. Über den Befehl **`algsys`** können aber Einschränkungen vorgenommen werden, indem die Variable **`realonly`** gesetzt wird.

Beispiel:

Löse folgende Gleichung zuerst in der Grundmenge C, anschließend nur für reelle Zahlen: $x^3 - 4x^2 + 6x - 4 = 0$

```
(%i15)  eqn:x^3-4*x^2+6*x-4=0;
(%o15)  x^3 - 4 x^2 + 6 x - 4 = 0

(%i16)  solve(eqn,x);
(%o16)  [ x = 1 - %i , x = %i + 1 , x = 2 ]

(%i25)  algsys([eqn],[x]),realonly:true;
(%o25)  [ [ x = 2 ] ]
```

Gleichungssysteme

Als Schüler ist es vor allem wichtig, lineare Gleichungssysteme zu lösen und zwar mit dem Menue, wer Interesse auf andere Lösungsmethoden hat, soll nachschauen unter dem Tutorium.

Also ***Gleichungen, Löse Lineares System*** und zunächst eingeben, aus wie vielen Gleichungen das System besteht und dann Eingebe der Gleichungen nebst der Variablen nach denen gelöst werden soll. Z.B.:

(%i1) linsolve([x+2*y+z=4, -x+4*y+z=7, 2*x+8*y-2*z=18], [x,y,z]);

(%o1) [x=1/2,y=2,z=-1/2]

Summen

Summe berechnen : sum(n.n,1,100) , wobei sum(Ausdruck, Variable, von, bis) bedeuten.

Als einfache Beispiel wollen wir die Summe von 2*(1 bis 100) berechnen:

sum(2*n, n, 1, 100)
10100

für weitergehende Summen siehe das Tutorium

Grenzwerte

Für Grenzwerte dient das Menue ***Rechnen Grenzwert …***, für spezielle Berechnungen siehe das Tutorium, ansonsten ist das Menue selbstserklärend.

Folgen

Für Folgen seht der Folge – Befehl zur Verfügung.

Definieren, z. B.

folge(n):=(1+1/n)^n
folge(1)
2

Berechnung der Folgen - Werte für n=1..12
%i17) folge(n):=(1+1/n)^n;

(%i18) makelist(folge(n),n,1,10);

(%o18)[2,9/4,64/27,625/256,7776/3125,117649/46656,2097152/823543,43046721/16777216,1000000000/387420489,25937424601/10000000000]

(%i19) float(%)

(%o19)[2.0,2.25,2.37037037037037,2.44140625,2.48832,2.521626371742113,2.546499697040713,2.565784513950348,2.581174791713197,2.5937424601]

(%i20) float(folge(150));
(%o20) 2.709275911334879

Die Zahl "Unendlich" kann mit inf eingegebenwerden,

„Minus Unendlich" mit minf

wir können also auch

limit(folge(n),n,inf); eingeben und erhalten
%e , also die Eulersche Zahl e

Übung: Berechne den Grenzwert der Folge

(2n-3)/(3n+1). Für n gegen unendlich

(%i21) limit((2*n+3)/(3*n-4),n,inf);

(%o21) 2/3(%i22) g(x):=abs(x^2-1);

Berechne die Steigung der Funktion abs(x^2.1) für linksseitige und rechtsseitige Annhäherung an die Spitze:

(%o22) g(x):=abs(x^2-1)

linksseitig:

(%i23) limit((g(1+h)-g(1))/h,h,0,plus);
(%o23) 2

rechtsseitig:

(%i24) limit((g(1+h)-g(1))/h,h,0,minus);
(%o24) -2

Differenzialrechnung

Wenn erst die Funktion festgelegt wird mit z. B. *f(x):=x^2+1*, können die Ableitungen nicht über *diff* festgelegt werden. Zwar kann die Differenzialrechnung auch über diesen Wege durchgeführt werden und über *subst* dann auch die Ableitung berechnet werden, am elegantesten jedoch ist dieser Weg:

Am besten wird zuerst die Funktion definiert, z. B.

define (f(x), x^3-5*x+8);

Und dann die Ableitungen:

define(f1(x),diff(f(x),x));

define(f2(x),diff(f(x),x,2));
define(f3(x),diff(f(x),x,3));

Beispiel: 2 x Diffenzieren, Nullstellen, Extrema (Test auf Relevalz), Wendepunkte (Test auf Relevanz, Symmetrieeigenschaften:

Wir geben die Funktion ein , bestimmen die Ableitungen:

```
(%i7)  define(f(x),-x^2+10*x-23);
(%o7)  f(x):=-x^2+10 x-23

(%i8)  define(f1(x),diff(f(x),x));
(%o8)  f1(x):=10-2 x

(%i9)  define(f2(x),diff(f(x),x,2));
(%o9)  f2(x):=-2

(%i10)  define(f3(x),diff(f(x),x,3));
(%o10)  f3(x):=0
```

wir bestimmen die Nullstellen:

```
(%i21)  solve([f(x)=0], [x]);
(%o21)  [x=5-sqrt(2), x=sqrt(2)+5]
```

Wir setzen die erste 1. Ableitung =0 und schauen, ob es einen möglichen Extremwert gibt:

```
(%i12)  f(x);
(%o12)  -x^2+10 x-23

(%i13)  solve([f1(x)=0], [x]);
(%o13)  [x=5]

(%i14)  f(5);
(%o14)  2
```

Also, wir haben ein Minimum bei x = 5

Wir setzen nun f2 (die zweite Ableitung) 0 und lösen auf:

```
(%i14)  f(5);
(%o14)  2

(%i15)  solve([f2(x)=0], [x]);
(%o15)  []
```

Die Funktion hat also keine Wendepunkte

Jetzt prüfen wir noch, ob Achsen – oder Punktsymmetrie-Eigenschaften vorhanden sind:

```
(%i32)  solve([f(x)=-f(x)], [x]);
(%o32)  [x = 5 - sqrt(2), x = sqrt(2) + 5]

(%i33)  solve(-[f(x)=f(-x)], [x]);
(%o33)  [x = 0]
```

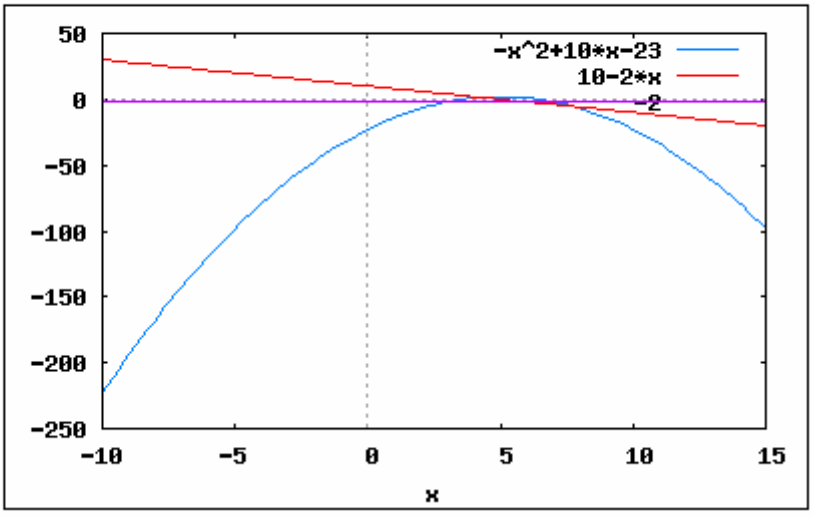
Da nur bestimme Werte gleich sind, liegt keine Symmetrie vor.

Nun wollen wir die Funktion und ihre Ableistungen noch zeichnen

(Menue Plotten 2d und für x noch Begrenzungswerte (-10 bis 15) eingeben, die vorgeschlagenen Werte (-5 bis 5) veranschaulichen die Gleichung nicht, für y belassen wir es bei den vorgeschlagenen Werten.

```
(%i36)  wxplot2d([f(x),f1(x),f2(x)], [x,-10,15])$
```

(%t36)



Oder wir plotten extra:

Gnuplot gibt die Grafik extra aus und *openmath* als speicherfähige (.pdf)-datei.

Integrieren

Wir berechnen die Stammfunktion einer Gleichung f(x) mit:

integrate(f(x).x)

und das Integral mit den Grenzen a und b:

integrate(f(x),x,a,b)

und das ist schon alles, was wir zur Integralrechnung wissen müssen.

Fläche zwischen 2 Kurven:

Beispiel:

f(x) = x^2-4*x+4
g(x)=-x^2+6x-4

erfordert zunächst die Schnittstellenberechnung:

```
(%i9)  f(x):=x^2-4*x+4;
(%o9)  f(x):=x^2-4 x+4

(%i10)  g(x):=-x^2+6*x-4;
(%o10)  g(x):=-x^2+6 x-4

(%i11)  float(solve([f(x)=g(x)], [x]));
(%o11)  [x=1.0, x=4.0]
```

(In diesem Fall kann float auch entfallen, da hier ganzzahlige Ergebnisse vorkommen), und dann die Rechnung:

```
(%i12)  abs(integrate(f(x),x,1,4)-integrate(g(x),x,1,4));
(%o12)  9
```

auch uneigentliche Integrale lassen sich berechnen:

```
(%i1)  integrate(3/(x^2),x,2,inf);
(%o1)  3/2
```

Volumen eines Drehkörpers:

Die Funktion f sei stetig auf [a;b]. Dann erzeugt das um die x-Achse rotierende Schaubild von f einen Drehkörper:

a sei -1, b sei 2. f()x) sei x+2:

```
(%i12)  f(x):=x+2;
(%o12)  f(x):=x+2

(%i13)  untereGrenze:2;
(%o13)  2

(%i14)  obereGrenze:4;
(%o14)  4

(%i15)
V:float(%pi*integrate(f(x)^2,x,untereGrenze,obereGrenze));
(%o15)  159.1740277818828
```

Zufallszahlen

Zufallszahlen werden mit dem Befehl random(z) erzeugt. Dabei gilt für den Fall, dass

z eine natürliche Zahl ist, eine positive ganze Pseudozufallszahl zwischen 0 und z-1 zurück

z eine (positiv ist verpflichtend) Kommazahl ist.

Gibt MAXIMA eine Zufallszahl (Kommazahl) zwischen o und z zurück

Beispiel:

```
(%i1)  i:49;
(%o1)  49

(%i2)  random(i)+1;
(%o2)  44
```

Permutationen ohne Wiederholung

```
perm_ow(n):=n!;
(%o3)  perm_ow(n):=n!
```

Permutationen mit Wiederholung

```
(%i4)  perm_mw(n,k):=n!/(product(k[i]!,i,1,length(k)));
```

$$\text{(\%o4)}\quad \text{perm\_mw}(n,k):=\frac{n!}{\prod_{i=1}^{\text{length}(k)} k_i!}$$

Kombinationen ohne Wiederholung:

```
(%i5)  comb_ow(n,k):=n!/(k!*(n-k)!);
```

$$\text{(\%o5)}\quad \text{comb\_ow}(n,k):=\frac{n!}{k!\,(n-k)!}$$

Kombinationen mit Wiederholung:

```
(%i6)  comb_mw(n,k):=(n+k-1)!/(k!*(n-1)!);
```

$$\text{(\%o6)}\quad \text{comb\_mw}(n,k):=\frac{(n+k-1)!}{k!\,(n-1)!}$$

```
(%i7)  var_ow(n,k):=n!/(n-1)!;
```

$$\text{(\%o7)}\quad \text{var\_ow}(n,k):=\frac{n!}{(n-1)!}$$

Variationen ohne Wiederholung:

```
(%i7)  var_ow(n,k):=n!/(n-1)!;
```

$$(\%o7)\quad \mathrm{var\_ow}(n,k):=\frac{n!}{(n-1)!}$$

Variationen mit Wiederholung

```
(%i8)  var_mw(n,k):=n^k;
```

$$(\%o8)\quad \mathrm{var\_mw}(n,k):=n^k$$

Weitere Möglichkeiten finden man unter dem mitgelieferten **Paket functs**

Beschreibende Statistik

Die meisten Befehle zur beschreibenden Statistik sind im Paket **desciptive** zusammengefasst, bevor mit diesen Fluktuationen gearbeitet werden kann, muss es geladen werden;

load (desciptive)

siehe hierzu auch das Tutorium unter beschreibende Statistik, hier sollen nur die Befehle aufgelistet werden:

`mini(L), maxi(L)`	Minimum bzw. Maximum einer Datenliste
`range(L)`	Max - Min, Spannweite der Elemente der Liste
`discrete_freq(L)`	Absolute Häufigkeiten der Listenelemente, ausgegeben in einer zweiten Subliste.
`median(L)`	Medianwert einer Liste, zentrales / mittleres Element
`quantile(L,p)`	p-Quantile, p aus dem Intervall [0,1], zentraler Wert an der angegebenen Stelle p des Samples - z.B. quantile(L,0.5) entspricht dem Median, quantile(L,0.25) wäre die erste Quartile, usw. (Achtung: zu Quantilen existieren mehrere unterschiedliche Definitionsmöglichkeiten)
`continuous_freq(L,n)`	Einteilung des Samples in *n* Klassen - gibt die Klassengrenzen und die absoluten Häufigkeiten der Vorkommen von Elementen der Liste in den Klassen aus. Standard (wenn keine Angabe getroffen wird) sind 10 Klassen.
`mean(L)`	Mittelwert - Arithmetische Mittel der Elemente der Liste *L*
`geometric_mean(L)`	Geometrisches Mittel - `product(x[i],i,1,n)^(1/n)`
`harmonic_mean(L)`	Harmonisches Mittel - `n/sum(1/x[i],i,1,n)`
`var(L)`	Varianz - `s^2=1/n*sum((x[i]-mean(L))^2,i,1,n)`
`var1(L)`	(n-1)-Varianz - `s^2=1/(n-1)*sum((x[i]-mean(L))^2,i,1,n)`
`std(L)`	Standardabweichung - Wurzel der Funktion *var*
`std1(L)`	(n-1)-Standardabweichung - Wurzel der Funktion *var1*

Hinweis - für den Umgang mit statistischen Daten sind Listenbefehle von zentraler Bedeutung, z.B. **`sort(L)`** (Sortieren einer Liste), **`length(L)`** (Anzahl der Elemente einer Liste).

Binomialverteilung

Wir gehen von einem Zufallsexperiment (Basisexperiment oder auch Bernoulliexperiment) aus. Ein derartiges Experiment hat 2 mögliche Ausgänge:

A das Ereignis tritt ein
¬A das Ereignis tritt nicht ein
P(A) = p :Wahrscheinlichkeit für den Eintritt von A
P(¬A) = 1 :- p Wahrscheinlichkeit, dass A nicht eintritt

Beispiel: A man würfelt einen Sechser, P(A) = 1/6, für das Gegenereignis gilt: P(¬A) = 1 - 1/6 = 5/6

Welche Funktionen bietet *Maxima* für derartige Berechnungen - welche Funktionen kann man sich leicht konstruieren?

`binomial(n,k)` ... eingebaute *Maxima*-Funktion für `n!/(k!·(n-k)!)`

`binv(n,p,k)` ... selbst definierte Funktion für die Anzahl P(k), mit der eine bestimmte Eigenschaft mit Wahrscheinlichkeit *p* k-mal auftritt. Sie wird über die Formel für „Ziehen mit Zurücklegen“ bestimmt:

$$P(k)=\binom{n}{k}\cdot p^k\cdot(1-p)^{n-k}\quad k\in(0,1,\ldots,n)$$

`binvert(n,p,a,b)` ... meist wollen wir Bereiche wissen - „mindestens k-mal“ oder „k zwischen a und b“ - dies lässt sich (selbst definiert) wie folgt ausdrücken:

$$P(k)=\sum_{k=a}^{b}P(k)\quad k,a,b\in(0,1,\ldots,n)$$

Damit lassen sich sämtliche Fälle berechnen!

Da wir uns Buchstaben nicht so leicht merken können, bezeichnen wir zumindest a und b mit mindestens (*mind*(und höchstens (*hoechst*), und wir wollen den Dezimalwert, so dass unsere Formel lautet:

$$\texttt{binvert}(n,p,\mathit{mind},\mathit{hoechst}):=\texttt{float}\left(\sum_{k=\mathit{mind}}^{\mathit{hoechst}}\texttt{binv}(n,p,k)\right)$$

```
(%i10)  binv(n,p,k):= binomial(n,k)*p^k*(1-p)^(n-k);
```

$$(\%o10)\quad \mathrm{binv}(n,p,k):=\binom{n}{k}p^{k}(1-p)^{n-k}$$

```
(%i11)
binvert(n,p,mind,hoechst):=float(sum(binv(n,p,k),k,mind,hoechst));
```

$$(\%o11)\quad \mathrm{binvert}(n,p,mind,hoechst):=\mathrm{float}\left(\sum_{k=mind}^{hoechst}\mathrm{binv}(n,p,k)\right)$$

Beispiel: (Aus Erfolg im Mathe-Abi 2006, Freiburger Verlag) Eine Münze wird 5-mal geworfen. Wie groß ist die Wahrscheinlichkeit folgender Ereignisse.?

A: Es tritt zweimal Zahl auf
B: Es tritt nur Wappen auf
C: Es tritt höchstens einmal Zahl auf
D: Es tritt mindestens einmal Zahl auf;
E: Es tritt mindestens 2x, jedoch höchstens 3x Zahl auf

```
(%i64)  binv(5,0.5,2);
(%o64)  0.3125

(%i65)  binv(5,0.5,0);
(%o65)  0.03125

(%i66)  binvert(5,0.5,0,1);
(%o66)  0.1875

(%i67)  binvert(5,0.5,1,5);
(%o67)  0.96875

(%i77)  binvert(5,0.5,2,3);
(%o77)  0.625
```

Hypergeometrische Verteilung

Die hypergometrische Verteilung beschriebt die Wahrscheinlichkeit, dass bei G gegebenen Elementen (Grundgesamtheit)m von denen M die gewünschte Eigenschaft besitzen, beim Herausgreifen von n Probestücken (Stichprobe) genau k Treffer erzielt werden.

Eine hypergeometrisch verteilte Zufallsvariable hat die Wahrscheinlichkeitsfunktion:

```
(%i13)
hypv(g,m,n,k):=binomial(m,k)*binomial(g-m,n-k)/binomial(g,n);
```

$$(\%o13)\quad hypv(g,m,n,k):=\frac{\binom{m}{k}\binom{g-m}{n-k}}{\binom{g}{n}}$$

Ein Beispiel hierfür ist die Berechnung von Wahrscheinlichkeiten beim Lotto 6 aus 49. Aus g = 49 Möglichkeiten, m = 6 Treffer, n =6 Stichprobe, k = 4 Richtige folgt die Berechnung:

```
(%i52)  hypv(49,6,6,4);
```

$$(\%o52)\quad \frac{645}{665896}$$

```
(%i53)  float(%);
```

$$(\%o53)\quad 9.6861972440140799\ 10^{-4}$$

(aus dem Tutorium um von MAXIMA:)

„

Vektoren (über Listen)

versteht unter einem Vektor einen Datentyp, der in den meisten Programmiersprachen als **Array** bezeichnet wird - eine unter einem Namen zusammengefasste geordnete Liste gleichartiger Elemente, die über einen Index ansprechbar sind (auch als eindimensionale Matrizen zu sehen). Wir betrachten hier vorerst zwei- und dreidimensionale Vektoren

Das Tutorium von Maxima gibt folgende Matrizenbefehle an:

LENGTH	Bestimmt die *Dimension* eines Vektors = Anzahl seiner Elemente
+ , - , * , /	Addition, Subtraktion, Multiplikation bzw. Division mit einem Skalar
. (ein Punkt)	Skalares Produkt zweier Vektoren
~	Vektorielles Produkt (Kreuzprodukt) zweier Vektoren. Zuvor muss mit `load(„vect")` das dafür notwendige Paket geladen werden. Das Produkt wird erst nach der Erweiterung mit **`express`** angezeigt. Beispiel: `express(v~w)` berechnet das Kreuzprodukt der Vektoren *v* und *w*.
uvect() / unitvector()~	Einheitsvektor eines Vektors. Zuvor muss mit `load(„eigen")` das dafür notwendige Paket geladen werden.

```
(%i1)  a:[5,10,15];
(%o1)  [ 5 , 10 , 15 ]

(%i2)  a[1];
(%o2)  5

(%i3)  a[2];
(%o3)  10
```

```
(%i5)  [b:[1,3,5],c:[-1,2,-3]]$

(%i6)  b . c;
(%o6)  - 10

(%i7)  10*b;
(%o7)  [ 10 , 30 , 50 ]

(%i8)  b+c;
(%o8)  [ 0 , 5 , 2 ]

(%i15)  length(a);
(%o15)  3
```

```
(%i16)  absv3(v):=
        sqrt(v[1]^2+v[2]^2+v[3]^2);
(%o16)  absv3(v):=sqrt(v_1^2+v_2^2+v_3^2)

(%i14)  absv3(b);
(%o14)  sqrt(35)

(%i11)  load("vect")$

(%i12)  express(b~c);
(%o12)  [ - 19 , - 2 , 5 ]
```

Der Zugriff auf die Elemente des Vektors erfolgt über den Index, der in eckigen Klammern angegeben wird - z. B. **`v[1]`** für das erste Element des Vektors *v*.

Der Index n ist im Normalfall eine positive ganze Zahl - diese definiert das **n-te Element** vom linken Ende des Vektors“

Matrizen und Vektoren

Ein rechteckigies Tableau von mathematischen Ausdrücken – im einfachen Fall sind das Zahlen – nennt man eine Matrix. Sie wird gekennzeichnet durch Anzahl ihrer Zeilen und Spalten. Eine Matrix kann in MAXIMA in Vektoren von gleicher Dimension aufgefasst werden. Umgekehrt können auch Vektoren als Matrizen aufgefasst werden.

Eingeben werden Matrizen über das Menü ALGEBRA-MARTIX EINGEBEN.

Alternativ können Matrizen auch direkt über die Tastatur über den Befehl **Matrix()** eingegeben werden.. Beispiel:

A:matrix[4,5,6], [4,3,1],[1,-5,0]

Matrizenbefehle:

A[i], A[i,j]	A[i] liefert eine Liste mit den Elementen der i-ten Zeile der Matrix A, A[i,j] liefert das ij-te Element (i-te Zeile, j-te Spalte).
row(A,1), col(A,1)	Zugriff auf eine (hier die erste) Reihe/Spalte von A
+, -, *, /	Matrizenaddition, -subtraktion etc. (Elementweise)
A*k	Multikplikation der Matrix A mit einem *Skalar* k. Ein Skalar ist zum Unterschied von einem Vektor und einer Matrix eine Größe, die nur aus einer einzigen Komponente - im Allgemeinen einer reellen Zahl - besteht
A.B	Multiplikation von Matrizen (Produktmatrix). Zwei Matrizen können nur multipliziert werden, wenn die Spaltenanzahl der linken Matrix mit der Zeilenanzahl der rechten übereinstimmt! A (5 x 2) * B (2 x 3) ergibt eine Matrix C (5 x 3) - allgemein: A(i x j) * B(j x k) = C(i x k). Achtung: *MAXIMA* kennt auch komponentenweise Multiplikation bzw. Division - Eingabe mit `*, /`.
matrix_size(A)	Gibt die Anzahl der Reihen und Spalten der Matrix A zurück.
rank(A)	Rang einer Matrix - Anzahl der linear unabhängigen Zeilen
transpose(A)	Transponierte Matrix von A
invert(A) oder **A^^-1**	Inverse Matrix von A
determinant(A)	Determinante von A
diagmatrix(n,x)	Erzeugt eine nxn Diagonalmatrix (alle Diagonalelemente werden auf „x“ gesetzt, alle anderen Elemente auf 0).
ident(n)	Erzeugt eine nxn Einheitsmatrix (alle Diagonalelemente werden auf „1“ gesetzt, alle anderen Elemente auf 0).
zeromatrix(n,m)	Erzeugt eine nxm Nullmatrix.
triangularize(A)	Erzeugt die ober Dreiecksform der Matrix A (entsteht durch Gauß’sche Elimination) - unter der Diagonalen stehen nur mehr Nullen!.
gramschmidt(A,i)	Gram-Schmidt Orthogonalisierung von A.

	Spalte bzw. Reihe hinzufügen
addcol(), addrow()	Beispiel: addcol(A,[1,5,7]) fügt die Spalte [1,5,7] zur Matrix A hinzu.
submatrix(i1,...,im,A) **submatrix(A,j1,...,jn)**	Entfernt die Zeilen i1, ..., im aus der Matrix A bzw. entfernt die Spalten j1, ..., jn.
coefmatrix([Gleich.],[Variable])	Erzeugt aus den gegebenen Gleichungen die Koeffizientenmatrix mit den angegebenen Variablen.
augcoefmatrix([Gleich.],[Var.])	Ergänzt die Koeffizientenmatrix eines Systems von linearen Gleichungen um die konstanten Glieder als zusätzliche Spalte
charpoly(A,v)	Charakteristisches Polynom der quadratischen Matrix A mit der Variable v
eigenvalues(A)	liefert die Eigenwerte der quadratischen Matrix A - zuerst kommen die Eigenwerte, dann die Häufigkeit des Vorkommens
eigenvectors(A)	liefert die Eigenwerte der quadratischen Matrix A - zuerst kommen die Eigenwerte, dann die dazugehörigen Eigenvektoren

Hinweis - Matrizen (wie alle Listenobjekte) lassen sich nicht durch eine einfache Variablenzuordnung kopieren, man benötigt dafür einen eigenen Befehl! (**copymatrix(A)**).
Die Zuordnung A:m1 erzeugt nur eine Referenz auf die Matrix m1, aber keine neue Matrix A!

Beispiel: Löse das Gleichungsystesm (zur Kontrolle mit *linsolve*, sonst mit Maztrizen)

x+2y-z = 8
-x+y+2z = 0
x+-5y-4z = -12

```
(%i9)
linsolve([x+2*y-z=8, -x+y+2*z=0, +x-5*y-4*z=-12], [x,y,z]);
(%o9)  [ x = 6 , y = 2 , z = 2 ]

(%i10)  A:matrix([1,2,-1],[-1,1,2],[1,-5,-4]);
        [  1   2   - 1 ]
        [              ]
(%o10)  [ - 1  1    2  ]
        [              ]
        [  1   - 5 - 4 ]

(%i11)  b:[8,0,-12];
(%o11)  [ 8 , 0 , - 12 ]

(%i12)  X:invert(A).b;
        [ 6 ]
        [   ]
(%o12)  [ 2 ]
        [   ]
        [ 2 ]
```

Ein weiteres Beispiel aus Mathemaik für die Fachhochschulreife (Biesterfeld, / Schröder / Spiritula):

```
(%i1)  A:matrix([3,-2,4],[5,3,-6],[4,7,-9]);
       [ 3  - 2   4  ]
       [             ]
(%o1)  [ 5   3   - 6 ]
       [             ]
       [ 4   7   - 9 ]

(%i2)  b:matrix([2],[16],[27]);
       [ 2  ]
       [    ]
(%o2)  [ 16 ]
       [    ]
       [ 27 ]
```

```
(%i6)  invert(A).b;
                 [ 2 ]
                 [   ]
(%o6)            [ 4 ]
                 [   ]
                 [ 1 ]
```

Speichern von Arbeitsabläufen in Arbeitsblättern

Jedes mathematische Problem, das einmal gelöst ist, lässt sich in Arbeitsblättern speichern und beim Wiederauftreten, lässt sich das Problem mit veränderten Werten durch Öffnen und überschreiben (Doppelklick auf die blauen Inputzeilen und anschließendem **Bearbeiten, Eingabe neu berechnen** , oder *Bearbeiten, re-evaluate all* oder mittels *STRG-Return*) wiederholen mit anderen Werten wiederholen. Erste Sammlungen haben wir schon für Sie erstellt.

In Maxima kann auch programmiert werden, lesen Sie dazu das Tutorium. Für Schüler sollte diese Anleitung genügen.

Winkelmaß – Bogenmaß, Logarithmen

Für die Umrechnung vom Winkelmaß und Bogenmaß (und umgekehrt), von natürlichen Logarithmen in dekadische Logarithmen und eine Vielzahl von Problemen haben wir bereits Arbeitsblätter erstellt.