Maxima

Theorie und Übungen
Lösungen

1 Erste Schritte

Die Arbeitsumgebung

Starte Maxima durch einen Doppelklick (XMaxima oder wxMaxima) oder oder in Shell (Befehlsinterpreter) mit dem Befehl `maxima`. Die Versionsnummer und der Prompt (Eingabeaufforderung `(%i1)` werden angezeigt.

Jede Eingabe *muss* mit einem Strichpunkt (Semikolon) oder einem Dollarzeichen abgeschlossen werden. Ein einfaches Beispiel:

```
(%i1) 5+3;
(%o1)                          8
```

Dabei steht `(%i1)` für den 1. Input und `(%o1)` steht für den 1. Output. Auf diese Weise kann man sich in den neuen Rechnungen auf bereits vorhandene Eingaben und Ausgaben beziehen:

```
(%i2) 20 + %i1;
(%o2)                         32

(%i3) 50 + %o2;
(%o3)                         82
```

Manchmal möchte man das Resultat eines Ausdrucks nicht ausgeben. Dann schliesst man die Eingabe mit dem Dollarzeichen ab:

```
(%i4) 42 * 35$
(%o4)
```

Hilfe

Kennt man den Namen eines Befehls, so kann man mit einem dem vorangestellten Fragezeichen Hilfe anfordern:

```
(%i5) ? quit;
(%o5) ...
```

Kennt man nur ein Stichwort, kann man mit einem mit einem doppelten vorangestellten Fragezeichen die Hilfeseiten durchsuchen:

```
(%i6) ?? equations;
(%o6) ...
```

Wenn Maxima in den Hilfeseiten nichts zum angegeben Suchwort findet, gibt es als Antwort `false` zurück. Wenn es einen Hilfetext findet, so wird dieser angezeigt. Findet Maxima aber mehrere Hilfeseiten zum angegebenen Stichwort, so weden die Themen mit Nummern aufgelistet und man muss die Nummer der gewünschten Seite oder für einen Abbruch `none` eingeben.

Mit den Pfeiltasten nach oben und nach unten können die letzten Befehle wieder ausgewählt und aufgerufen werden.

Zuweisungen für Variablen

Der Operator : weist einer Variablen einen Wert zu.

```
(%i7) a : 5$
(%i8) 4*a;
(%o8)                                  20
```

Definition von Funktionen

Eine eigene Funktion wird mit := definiert.

```
(%i9) f(x) := 4*x + 3$
(%i10) f(2);
(%o10)                                 11
```

Gleichheit

Ein Gleichheitszeichen = stellt einen logischen Vergleich dar.

```
(%i11) y = 3*x + 5;
(%o11)                                y = 5 + 3 x
```

In den folgenden Kapiteln erfahren wir, wie diese Symbole verwendet werden.

Abbruch

Einen laufenden Befehl bricht man mit `Ctrl-G` ab. (Control-Taste halten und die Taste `G` (manchmal `Control-C`) drücken. Meldet sich der Debugger („das Fehlersuchprogramm“) beendet man diesen mit der Eingabe von `Q`.

Aufgabe 1.1

Gib die Rechnung 3+4+5; ein und führe sie aus. In der nächsten Input-Zeile gibst du %; ein. Was genau geschieht?

% ist ein Platzhalter für den letzten Output.

Aufgabe 1.2

Du befindest dich in der Eingabezeile `(%i56)` und hast in `(%i37)` und in `(%i51)` komplizierte Rechnungen eingegeben. Wie kannst du die Summe dieser Resultate berechnen, ohne die Rechnungen noch einmal einzugeben?

```
(%i56) (%i37) + (%i51);
```

Aufgabe 1.3

Kann man mehrere Rechnungen durch `;` oder `$` getrennt auf eine Zeile schreiben? Probiere es aus!

Ja. Zum Beispiel:

```
(%i3) 27*30; x : 16$ 34*x;
```

Aufgabe 1.4

Probiere mit der Hilfefunktion, ob es einen Befehl mit dem Namen `date` gibt. Wenn nicht, dann suche einen Befehl, der der `date` enthält. Notiere das Resultat deiner Suche und probiere diesen Befehl bzw. diese Befehle aus.

Einen Befehl mit dem Namen `date` gibt es nicht. Dafür einen Befehl `timedate()`, der die Zeit und das Datum ausgibt.

Aufgabe 1.5

Erkläre, warum es in der folgenden Zeile die Symbole `:` und `=` braucht.

```
(%i6) g : y = 3*x - 4;
```

`:` weist der Variablen `g` die Gleichung `y = 3*x - 4` zu. In der Variablen ist also eine Gleichung *gespeichert.*

2 Arithmetik

Genauigkeit

Man beachte, dass Maxima so gut es geht exakt rechnet.

```
(%i1) 5/3 * 4/7;
(%o1)                20/21

(%i2) sqrt(2) * sqrt(3);
(%o2)                sqrt(6)
```

Das heisst, dass $\sqrt{6}$ nicht als gerundete Dezimalzahl `2.449489742783178` sondern als Wurzel dargestellt wird. Möchte man dennoch einen gerundeten dezimalen Wert, so verwendet man die Funktion `float`.

```
(%i3) float(sqrt(6));
(%o3)           2.449489742783178
```

Man bekommt auch dann sofort gerundete Dezimalzahlen, wenn man die Zahlen bereits als Dezimalzahl darstellt:

```
(%i4) sqrt(3.0);
(%o4)           1.732050807568877

(%i5) 5.0/2;
(%o5)           2.5
```

Normalerweise rechnet Maxima Dezimalzahlen auf 16 Ziffern genau, wobei die letzte Ziffer unzuverlässig ist. Da unsere Computer im Binärsystem rechnen, treten manchmal unerwünschte Rundungsphänomene auf:

```
(%i6) 7*9.3;
(%o6)           65.10000000000001
```

Will man die Anzahl der Nachkommastellen verändern, so verwendet man den Befehl `bfloat(<zahl>` gemeinsam mit der Zuweisung `fpprec : <n>`, welche n Stellen der Zahl anzeigt. (floating point precision)

```
(%i7) fpprec : 50$ bfloat(1/7);
(%o7)   1.4285714285714285714285714285\
        14285714285714285714b-1

(%i8) fpprec : 16$ bfloat(1/7);
(%o8)           1.428571428571429b-1
```

Primzahlen und Teilbarkeit

- Um zu prüfen, ob eine natürliche Zahl `n` eine Primzahl ist, gibt man `primep(n)` ein.
- Die grösste Primzahl, die kleiner als die natürliche Zahl `n` ist, kann mit `prev_prime(n)` bestimmt werden.
- Die kleinste Primzahl, die grösser als die natürlich Zahl `n` ist, kann mit `next_prime(n)` bestimmt werden.
- Durch `factor(n)` wird die Primfaktorzerlegung einer natürlichen Zahl `n` ausgegeben.
- `gcd(m, n)` berechet den grössten gemeinsamen Teiler der natürlichen Zahlen `m` und `n`.
- `lcm(m, n)` berechet das kleinste gemeinsame Vielfache (least common multiple) der natürlichen Zahlen `m` und `n`. Achtung: Damit diese Funktion zur Verfügung steht, müssen zuvor mit `'load("functs")'` zusätzliche Funktionen geladen werden.

Aufgabe 2.1

Zerlege die Zahl 44 099 088 000 in Primfaktoren.

```
(%i1) factor(44099088000);

(%o1)           7    3      4
               2  3 5  11 17
```

Aufgabe 2.2

Berechne den ggT von 4725 und 3465.

```
(%i2) gcd(4725, 3465);

(%o2)          315
```

Aufgabe 2.3

Berechne das kgV von 1386 und 924.

```
(%i3) load("functs")$ lcm(1386, 924);

(%o3)      2772;
```

Aufgabe 2.4

Berechne den ggT von 450, 315 und 1323.

```
(%i4) gcd(gcd(450, 315), 1323);

(%o4)          9
```

Aufgabe 2.5

Kürze $\frac{432}{576}$.

```
(%i5) 432/576;

(%o5)          3
               -
               4
```

Aufgabe 2.6

Verwandle $\frac{12}{625}$ in einen Dezimalzahl.

```
(%i6) float(12/625);

(%o6)          0.0192
```

Aufgabe 2.7

Berechne $\sqrt{\dfrac{7.94 \cdot 9.342^3}{17.2345^2 + 645.7288}}$

```
(%i7) sqrt(7.94 * 9.342^3 / (17.2345^2 + 645.7288));

(%o7)          2.620415810278678
```

Aufgabe 2.8

Welcher Rest entsteht, wenn man 123 456 durch 6543 dividiert?

```
(%i8) mod(123456, 6543);

(%o8)5682
```

Aufgabe 2.9

Mersenne-Zahlen sind Zahlen der Form $2^n - 1$ für $n = 1, 2, 3, \ldots$. Bestimme alle Mersenne-Zahlen, die kleiner als 10 000 sind. Welche davon sind Primzahlen (sogenannte Mersenne-Primzahlen)? Halte die Ergebnisse in der folgenden Tabelle fest.

n	Mersenne-Zahl	ist prim?
1	1	nein
2	3	ja
3	7	ja
4	15	nein
5	31	ja
6	63	nein
7	127	ja
8	255	nein
9	511	nein
10	1023	nein
11	2047	nein
12	4095	nein
13	8192	ja

3 Algebra

Im Unterschied zu einem „gewöhnlichen“ Taschenrechner kann Maxima auch *symbolisch*; also mit Symbolen (Buchstaben) rechnen. Programme oder Taschenrechner mit dieser Fähigkeit werden Computer Algebra Systeme (CAS) genannt. Dazu ein einfaches Beispiel:

```
(%i1) 5*a + 3*b - 4*a;
(%o1)                  3 b + a
```

Man beachte, dass in Maxima jede Multiplikation geschrieben werden muss.

```
(%i2) 5a + 3b - 4a;
(%o2) Incorrect syntax: A is not an infix operator
      5aSpace
       ^
```

Einfache Produkte und Quotienten werden sofort vereinfacht:

```
(%i3) 54*a^2*b*c^3 / (9*a*b*c);
                          2
(%o3)                  6 a c
```

Kommen neben den Produkten auch Summen oder Differenzen vor, ist Maxima zunächst nicht sehr kooperativ:

```
(%i4) (5*x^2 - 15*x^3)/(5*x);
                     2       3
(%o4)               5 x  - 15 x
                    ------------
                        5 x
```

Mit dem Befehl `ratsimp` („simplify rational expressions“) versucht Maxima, gebrochen rationale Ausdrücke (Quotienten aus Polynomen) zu vereinfachen.

```
(%i5) ratsimp( (5*x^2 - 15*x^3)/(5*x) );
                          2
(%o5)                 x - 3 x
```

Mit dem Befehl `factor` werden Summen faktorisiert.

```
(%i6) factor(12*a^2*b + 18*a*b^2);

(%o6) 6 a b (3 b + 2 a)
```

Maxima „kennt“ die binomischen Formeln:

```
(%i7) factor( x^2 + 2*x*y + y^2 );
                          2
(%o7)                (y + x)
```

Der Befehl `factor` ist eine Art Ersatz für `ratsimp`. Wendet man `factor` auf einen Quo-

tienten an, so werden sowohl der Zähler als auch der Nenner faktorisiert. Treten dabei die gleichen Faktoren auf, werden sie sofort gekürzt. Daher hat der folgende Befehl eine ähnliche Wirkung wie der in `(%i5)`.

```
(%i8) factor( (5*x^2 - 15*x^3)/(5*x) );
(%o8) - x (3 x - 1)
```

Die Umkehrung des Faktorisierens ist das Ausmultiplizieren. Dafür gibt es in Maxima den Befehl `expand`.

```
(%i9) expand( 3*a*(a+b)*(a-b) );

(%o9)                3         2
                  3 a  - 3 a b
```

Auch grosse Potenzen von Binomen können damit „gerechnet“ werden:

```
(%i10) expand( (x+y)^4 );
(%o10)             4         3       2  2       3       4
                   y  + 4 x y  + 6 x  y  + 4 x  y + x
```

Hinweis: Damit Maxima die Terme in alphabetischer statt in umgekehrt alphabetischer Reihenfolge darstellt, muss man der Variablen `powerdisp` den Wert `true` zuweisen.

Aufgabe 3.1

Vereinfache den Term $(19m - 5n + 1) - (4m - 2n - 3)$.

```
(%i1) (19*m - 5*n + 1) - (4*m - 2*n - 3);
(%o1)   4 + 15 m - 3 n
```

Aufgabe 3.2

Vereinfache den Term $\dfrac{(ab)^2c^3}{a^2bc}$.

```
(%i2) (a*b)^2*c^3/(a^2*b*c);
(%o2)              2
                b c
```

Aufgabe 3.3

Vereinfache den Term $\dfrac{6x^3 + 39x^2 + 39x - 30}{2x^2 + 9x - 5}$.

```
(%i3) ratsimp( (6*x^3+39*x^2+39*x-30)/(2*x^2+9*x-5) );
(%o3)           3 x + 6
```

Aufgabe 3.4

Faktorisieren den Term $64a^2 - 240a + 225 - 40ac + 75c$.

```
(%i4) factor( 64*a^2 - 240*a + 225 - 40*a*c + 75*c );

(%o4)           - (8 a - 15) (5 c - 8 a + 15)
```

Aufgabe 3.5

Berechne das Produkt $(x^2 + 4x + 5)(x^2 - 4x + 5)$.

```
(%i5) expand( (x^2 + 4*x + 5)*(x^2 - 4*x + 5) );

(%o5)             4      2
                 x  - 6 x  + 25
```

Aufgabe 3.6

Wende nacheinander `expand`, `factor` und `ratsimp` auf den Term

$$(a-7)(a+1)(y+z) + (a-7)(a+1)(-z)$$

an. Vergleiche die Resultate.

```
(%i6) expand( (a-7)*(a+1)*(y+z)+(a-7)*(a+1)*(-z) );

(%o6)             2
                 a  y - 6 a y - 7 y
```

`expand` rechnet die Produkte aus und fasst anschliessend gleiche Summanden zusammen. Daher entsteht eine Summe bzw. Differenz.

```
(%i7) factor( (a-7)*(a+1)*(y+z)+(a-7)*(a+1)*(-z) );

(%o7)           (a - 7) (a + 1) y
```

`factor` versucht so viel wie möglich zu faktoriseren. Dies gelingt auch mit den Faktoren $(a-7)$ und $(a+1)$. Der übrige Term vereinfacht sich zu y.

```
(%i8) ratsimp( (a-7)*(a+1)*(y+z)+(a-7)*(a+1)*(-z) );

(%o8)             2
                 (a  - 6 a - 7) y
```

`ratsimp` wird normalerweise auf Quotienten angewendet. Daher ist das Ergebnis etwas „seltsam“.

4 Variablen und Konstante

Manchmal möchte man Zahlen in einer Variablen speichern. In Maxima gibt es dafür den Doppelpunkt-Operator:

```
(%i1) a : 10;
(%o1)                10

(%i2) radius : 2.536;
(%o2)                2.536
```

Danach kann man mit den Namen als Platzhalter für die Zahlen rechnen:

```
(%i3) a * radius;
(%o3)                25.36
```

Es ist aber auch möglich Variablen als Platzhalter für andere Variablen oder Terme zu verwenden:

```
(%i4) a : (x + y + z);
(%o4)                z+y+x
```

und damit zu rechnen:

```
(%i5) 2 * a;

(%o5)              2*(z+y+x)
```

Ordnet man mit dem Doppelpunkt einem Namen einen neuen Wert zu, dann wird ein allfälliger bestehender Wert überschrieben.

Möchte man die Bindung zwischen einem Variablennamen und dem Werte aufheben, verwendet man den Befehl `kill(...)`:

```
(%i6) kill(radius);
(%o6)                done
```

Für das folgende Kapitel ist es gut zu wissen, dass man auch Gleichungen unter einem Namen speichern kann:

```
(%i7) glg : 5*x - 7 = 3*x + 8;
(%o7)                5 x - 7 = 3 x + 8
```

Mathematische Konstanten wie die Kreiszahl π, die Eulersche Zahl e oder die imaginäre Einheit haben in Maxima eigene Namen:

```
(%i8) %pi;
(%o8)                %pi

(%i9) float(%pi);
(%o9)                3.141592653589793
```

```
(%i10) float(%e);
(%o10)           2.718281828459045
```

Die Eulersche Zahl e kann als Grenzwert der Folge

$$\left(1+\frac{1}{n}\right)^n$$

gewonnen werden, wenn die natürliche Zahl n gegen Unendlich strebt.

```
(%i11) n : 1$ float((1 + 1/n)^n);
(%o11)           2.0

(%i12) n : 100$ float((1 + 1/n)^n);
(%o12)           2.704813829421526

(%i13) n : 10000$ float((1 + 1/n)^n);
(%o13)           2.718145926825225
```

Die imaginäre Einheit ist diejenige „Zahl“, deren Quadrat ... ergibt:

```
(%i13) %i^2;
(%o13)           -1
```

Aufgabe 4.1

Berechne für $a = 2.593$, $b = -4.255$ und $c = \frac{1}{4}$ den Wert des Terms $\frac{a^2 - 4b + 5}{(b+c)^2}$.

```
(%i1) a : 2.593$ b : -4.255$ c : 1/4$ (a^2-4*b+5)/(b+c)^2;
(%o1)          1.79199527432158
```

Aufgabe 4.2

Berechne für $r = 5.327$ den Wert von $\frac{4}{3}\pi r^3$. Um welche Formel handelt es sich?

```
(%i2) r : 5.327$ float(4/3 * %pi * r^3);
(%o2)          633.1948669171323
```

Es handelt sich um die Formel zur Berechnung des Kugelvolumens.

Aufgabe 4.3

Berechne $\sqrt{-4}$.

```
(%i3) sqrt(-4);
(%o3)          2 %i
```

Aufgabe 4.4

Mit der Heronschen Flächenformel lässt sich der Flächeninhalt eines Dreiecks direkt aus den drei Seitenlängen berechnen.

- Berechne zuerst $s = \frac{a+b+c}{2}$ (halber Umfang)
- und damit $A = \sqrt{s(s-a)(s-b)(s-c)}$

Berechne damit den Flächenihnalt der folgenden Dreiecke:

(a) $a = 3$, $b = 4$, $c = 5$

(b) $a = 7$, $b = 3$, $c = 8$

(c) $a = 5$, $b = 9$, $c = 2$

(d) $a = x$, $b = x$, $c = x$

(a) $a = 3$, $b = 4$, $c = 5$

```
(%i4) a : 3$ b : 4$ c : 5$
```

```
(%i5) s : (a + b + c)/2;
```

```
(%i6) A : sqrt(s*(s-a)*(s-b)*(s-c));
```

```
(%o6)                6
```

(b) `(%i7) a=7, b=3, c=8$`

10.39

(c) $a = 5$, $b = 9$, $c = 2$

kein Dreieck!

$a = x$, $b = x$, $c = x$

$\frac{\sqrt{3}}{4}x^2$

Aufgabe 4.5

Lösche alle Variablenbindungen, die in diesem Kapitel definiert wurden.

```
(%i1) kill(all)$
```

5 Gleichungen

Einfache Gleichungen kann man mit dem Befehl `solve` lösen. Man muss dazu eine Gleichung und die Variable angeben, nach der man die Gleichung auflösen möchte.

```
(%i1) solve(5*x - 7 = 0, x);
(%o1) [x = 7/5]
```

Maxima löst Gleichungen auch symbolisch:

```
(%i2) solve(a*x + b = c, x);
(%o2) [x = (c-b)/a]

(%i3) solve(a*x + b = c, b);
(%o3) [b = c-a*x]
```

Auch quadratische Gleichungen oder Gleichungen höheren Grades können gelöst werden:

```
(%i4) solve(x^2 + 3*x - 2 = 0, x);
(%o4) [x = -(sqrt(17)+3)/2,x = (sqrt(17)-3)/2]

(%i5) solve(x^5-8*x^4-15*x^3+139*x^2+49*x-563=0, x);
(%o5) [0 = x^5-8*x^4-15*x^3+139*x^2+49*x-563]
```

Die letzte Gleichung ist etwas zu schwierig. Das liegt daran, dass es für Gleichungen vom Grad 5 im Allgemeinen keine Lösungsformel gibt. Dafür kann man Näherungslösungen mit `allroots()` bestimmen:

```
(%i6) allroots(x^5-8*x^4-15*x^3+139*x^2+49*x-563=0);
(%o6) [x = 2.650040027878449, x = -2.468649242391743,
       x = -3.267064232568197, x = 3.449600929215315,
       x = 7.636072517866177]
```

Aufgabe 5.1

$\sqrt{4x+3} = 9$

```
(%i1) solve(sqrt(4*x + 3) = 9, x);

(%o1)                 39
                [x = --]
                      2
```

Aufgabe 5.2

$ax + bx = 9$

```
(%i2) solve(a*x + b*x = 9, x);
```

```
(%o2)                  9
                [x = -----]
                     b + a
```

Aufgabe 5.3

$ax^2 + bx + c = 0$

```
(%i3) solve(a*x^2 + b*x + c = 0, x);

(%o3)             2
             sqrt(b  - 4 a c) + b
      [x = - --------------------,
                    2 a

                     2
                sqrt(b  - 4 a c) - b
            x = --------------------]
                      2 a
```

Aufgabe 5.4

$\frac{x-1}{x+1} + \frac{3}{x} = 9$

```
(%i4) solve((x-1)/(x+1) + 3/x = 9, x);

(%o4)              sqrt(145) + 7      sqrt(145) - 7
          [x = - -------------, x = -------------]
                       16                 16
```

Aufgabe 5.5

$2x^5 - 10x^4 + x^3 + 30x^2 - 10x - 14 = 0$

```
(%i4) allroots(2*x^5-10*x^4+x^3+30*x^2-10*x-14=0);

(%o4)[x = - 0.56752890832055, x = - 1.492448744378336,
      x = 1.044142336517212,  x = 1.943695597380042,
      x = 4.072139718801632]
```

Aufgabe 5.6

$x^2 + 1 = 0$

```
(%i6) solve(x^2 + 1 = 0, x);

(%o6)           [x = - %i, x = %i]
```

6 Gleichungssysteme

Gleichungssysteme werden wie Gleichungen mit `solve` gelöst, nur dass man die einzelnen Gleichungen und auch die Lösungsvariablen als Liste (mit eckigen Klammern) eingibt.

$$2x - 3y = 11$$
$$5x + 2y = -1$$

```
(%i1) solve([2*x - 3*y = 11,  5*x + 2*y = -1], [x,y]);
(%o1)                [[x = 1, y = - 3]]
```

Manchmal ist es sinnvoll, die Gleichungen als Variablen zu speichern und dann zu lösen.

$$5x + 3y - 4z = 7$$
$$3x - y + 2z = 9$$
$$-x + 6y - 7z = 3$$

```
(%i2) g1 : 5*x + 3*y - 4*z = 7$
(%i3) g2 : 3*x - y +  2*z = 9$
(%i4) g3 : x - 6*y + 7*z = -4$
(%i5) solve( [g1, g2, g3], [x, y, z]);
                      3      17      13
(%o5)            [[x = -, y = --, z = --]]
                      2      2       2
```

Ein lineares Gleichungssystem hat immer entweder keine, genau eine oder unendlich viele Lösungen. Den zweiten Fall haben wir oben bereits kennen gelernt. Im nächsten Beispiel werden wir offensichtlich keine Lösungen erhalten. Wie reagiert Maxima darauf? Beispiele:

$$2x + 3y = 8$$
$$2x + 3y = 9$$

```
(%i6) solve([2*x + 3*y = 8, 2*x + 3*y = 9], [x, y]);
(%o6)                []
```

Nun betrachten wir noch den Fall, in dem es mehr Variablen als Lösungen gibt:

$$3x - 5y + 2z = 1$$
$$-x + 2y - 4z = 6$$

```
(%i7) solve([3*x-5*y+2*z=1, -x+2*y-4*z=6], [x,y,z]);
(%o7) [[x = 16 %r1 + 32, y = 10 %r1 + 19, z = %r1]]
```

Hier wählt Maxima einen Parameter `%r1` ($= z$), der einen beliebigen aber festen Wert annehmen kann. Dadurch wird eine von unendlich vielen Lösungen definiert.

Um zum Beispiel die Lösung für den Parameter `%r1=3` zu berechnen, speichern wir zunächst die Lösung(en) unter einer Variable ab und verwenden anschliessend den Befehl `substitute` oder seine Kurzform `subst`:

```
(%i8) lsg : %$
(%i9) subst(%r1=3, lsg);
(%o9)                [[x = 80, y = 49, z = 3]]
```

Auch nichtlineare Gleichungssysteme können mit `solve` gelöst werden:

$$\begin{aligned} x^2 + y^2 &= 5 \\ x + y &= 1 \end{aligned}$$

```
(%i10) solve( [x^2 + y^2 = 5, x + y = 1], [x, y] );
(%o10)      [[x = - 1, y = 2], [x = 2, y = - 1]]
```

Aufgabe 6.1

Löse das Gleichungssystem.

$$\left|\begin{array}{r} a - b + 3c + d = -2 \\ -4a + 2b - c + 5d = -5 \\ 3a - 3b + 2c - d = 4 \\ 2a + b - 7d = -1 \end{array}\right|$$

```
(%i1) g1 : a - b + 3*c + d  = -2$
(%i2) g2 : -4*a + 2*b - c + 5*d  = -5$
(%i3) g3 : 3*a - 3*b + 2*c - d = 4$
(%i4) g4 : 2*a + b - 7*d = -1$
(%i5) solve( [g1,g2,g3,g4], [a,b,c,d] );
(%o5)              [[a = 3, b = 0, c = - 2, d = 1]]
```

Aufgabe 6.2

Bestimme zwei verschiedene Lösungen des Gleichungssystem.

$$\left|\begin{array}{r} 3s - 4t + 2u - v = 3 \\ -s + t + u - 5v = 1 \\ 2s - 7t + 3u - 2v = 3 \end{array}\right|$$

```
(%i6) g1 : 3*s - 4*t + 2*u - v = 3$
(%i7) g2 : -s + t + u - 5*v = 1$
```

```
(%i8) g3 : 2*s - 7*t + 3*u - 2*v = 3$

(%i9) lsg : solve( [g1,g2,g3], [s,t,u,v] );

(%o9)        6 %r1 - 5      4 %r1 + 1      68 %r1 + 25
     [[s = - ---------, t = ---------, u = ------------, v = %r1]]
                10             4               20

(%i10) float(subst([%r1 = 1], lsg));

(%o10)[[s = - 0.1, t = 1.25, u = 4.65, v = 1.0]]

(%i11) float(subst([%r1 = 2], lsg));

(%o11)[[s = - 0.7, t = 2.25, u = 8.05, v = 2.0]]
```

Aufgabe 6.3

Prüfe mit dem `subst`-Befehl, ob das Zahlentripel $a = 28$, $b = -64$, $c = 15$ eine Lösung der Gleichung $4a^2 + 3bc - 5b = 13a + c^2$ ist.

```
(%i11) subst([a=28, b=-64, c=15],
          4*a^2 + 3*b*c - 5*b = 13*a +c^2);

(%o11)           576 = 589
```

Die angegebenen Werte sind keine Lösung der Gleichung.

7 Funktionen

Funktion werden mit dem Operator `:=` definiert.

```
(%i1) f(x) := 3*x + 5;

(%o1)           f(x) := 3 x + 5;
```

Zur Definition einer Funktion gehören somit ihr Name (hier `f`), gefolgt vom sogenannten *formalen Parameter*, der dann rechts von `:=` wieder auftritt und stellvertretend für den Wert steht, mit dem man später die Funktion auswertet. Zum Beispiel so:

```
(%i2) f(2);

(%o2)              11
```

oder so:

```
(%i3) f(a+b);

(%o3)           3 (b + a) + 5
```

Auf diese Weise lassen sich die Nullstellen einer Funktion bestimmen:

```
(%i4) solve( f(x) = 0, x );
```

```
(%o4)                    5
                 [x = - -]
                         3
```

Auch Funktionen in zwei Variablen sind kein Problem:

```
(%i5) g(u, v) := u^2 + v^2$
(%i6) g(3, 4);
(%o6)            25
```

Zusammen mit der Funktion `makelist` können Wertetabellen erzeugt werden.

```
(%i7) h(z) := z^2 + 1$
(%i8) makelist( h(a), a, -5, 5 );
(%o8) [26, 17, 10, 5, 2, 1, 2, 5, 10, 17, 26]
```

Aufgabe 7.1

Definiere die Funktion mit der Funktionsgleichung $f(x) = x^2 + 2x - 3$ und berechne damit

- $f(29)$
- die Nullstellen von f.

```
(%i1) f(x) := x^2 + 2*x - 3$
(%i2) f(29);
(%o2)           896
(%i3) solve(f(x) = 0, x);
(%o3)           [x = - 3, x = 1]
```

Aufgabe 7.2

Erstelle eine Wertetabelle der Funktion mit der Gleichung

$$g(x) = \frac{x-7}{x+2}$$

für $x = 1, 2, \ldots, 10$

```
(%i5) g(x) := (x-7)/(x+2)$
(%i6) makelist(g(x), x, 1, 10);
(%o6)       5    4    1    2    1     1   2   1
     [- 2, - -, - -, - -, - -, - -, 0, --, --, -]
            4    5    2    7    8     10  11  4
```

Aufgabe 7.3

Berechne die Schnittpunkte der Funktionen mit den Gleichungen $f(x) = \frac{1}{4}x^2 - 2x + 3$ und $g(x) = 2x - 4$.

```
(%i7) f(x) := 1/4*x^2 - 2*x + 3$ g(x) := 2*x - 4$

(%i8) solve(f(x) = g(x), x);

(%o8)          [x = 2,x = 14]

(%i9) f(2); f(14);

(%o9)          0
               24
```

Aufgabe 7.4

Definiere eine Funktion mit dem Namen `mw3`, welche aus drei Zahlen a, b und c das arithmetische Mittel (den „Durchschnitt“) bestimmt. Berechne damit das arithmetische Mittel von 2.2, 7.5 und 4.4.

```
(%i10) mw3(a, b, c) := (a+b+c)/3$

(%i11) mw3(2.2, 7.5, 4.4);

(%o11)          4.7
```

8 Graphen

Graphische Darstellungen werden von einem anderen Programm mit dem Namen *Gnuplot* erzeugt. Maxima leitet die Befehle zur graphischen Darstellung an dieses Programm weiter. Gnuplot wird bei der Installation von Maxima automatisch mit installiert und der Benutzer merkt wenig davon, dass er zeitweise mit einem anderen Programm arbeitet.

8.1 Graph einer Funktion

Allgemein gilt für die Darstellung von Funktionen mit einem Argument und einem Wert:

`plot2d(`*`funktion, Definitionsbereich, Optionen`*`)`

Oder konkret:

```
(%i1) plot2d (1/2*x^2 + 1, [x, -5, 5], [y, -5, 5]);
(%o1) <hier müsste ein Grafikfenster erscheinen>
```

Hier wurde zusätzlich ein Wertebereich angegeben. Lässt man diesen weg, versucht Maxima (Gnuplot) selber, einen geeigneten Bereich zu finden.

Beachte: So lange noch ein Gnuplot-Fenster geöffnet ist, reagiert Maxima nicht auf weitere Eingaben. Also: zuerst die offenen Gnuplot-Fenster schliessen.

8.2 Mehrere Graphen in einem Koordinatensystem

Wenn wir Funktionen unter einem Namen definieren, können wir den Funktionsnamen anstelle des Funktionsterms eingeben. Zudem können mit Hilfe der Listenschreibweise mehrere Funktionen gleichzeitig in einem Koordinatensystem dargestellt werden.

```
(%i2) f(x) := 1/4*x^2 - 2*x + 3$ g(x) := 2*x-4$
(%i3) plot2d([f(x), g(x)], [x, -5, 15], [y, -10, 30]);
```

8.3 Darstellung von Punktmengen

Die folgenden Daten stellen die durchschnittliche Körpergrössen und das durchschnittliche Körpergewicht von Amerikanerinnen im Alter von 30–39 Jahren dar. (Quelle: *The World Almanac and Book of Facts, 1975*)

Grösse [in]	58	59	60	61	62	63	64	65
Gewicht [lbs]	115	117	120	123	126	129	132	135

Grösse [in]	66	67	68	69	70	71	72
Gewicht [lbs]	139	142	146	150	154	159	164

Gewicht und Grösse wurden in normaler Hauskleidung und mit Schuhen gemessen.

Darstellung von Punktmengen

```
(%i4) groesse : [58, 59, 60, 61, 62, 63, 64, 65, 66, 67,
                 68, 69, 70, 71, 72]$
(%i5) gewicht : [115, 117, 120, 123, 126, 129, 132, 135, 139, 142,
                 146, 150, 154, 159, 164]$
(%i6) plot2d([discrete, groesse, gewicht], [xlabel,"inch"],
               [ylabel,"lbs"]);
(%i7) plot2d([discrete, groesse, gewicht], [style, [points, 1]]);
```

8.4 Funktionen in Parameterdarstellung

Bei einer Funktion in Parameterdarstellung werden sowohl die x- als auch die y-Koordinate als Funktion eines Parameters (der oft mit t bezeichnet wird) dargestellt.

Beispiel: $x = 2t + 1$, $y = t^2$

t	1.2	1.3	1.4	1.5	1.6	1.7	1.8
x	2.6	2.8	3.0	3.2	3.4	3.6	3.8
y	1.44	1.69	1.96	2.25	2.56	2.89	3.24

Und so wird eine Funktionen in der Parameterdarstellung dargestellt:

```
(%i8) plot2d([parametric, 2*t+1, t^2, [t, 1, 2]]);
```

8.5 Funktionen der Form $z = f(x, y)$

Hängt eine Funktion $z = f(x, y)$ von den Variablen x und y ab, so können die unabhängigen Variablen x und y als Punkt $P(x|y)$ in der Koordinatenebene aufgefasst

werden. Wenn wir uns von diesem Punkt P aus z Einheiten senkrecht von der Koordinatenebene entfernen, erhalten wir einen Punkt $P'(x|y|z)$ im Raum. Führen wir dieses Verfahren für sehr viele nahe beieinander liegende Punkte $P(x|y)$ durch erhalten wir bei einer „gutartigen“ Funktion f auch sehr viele nahe beieinander liegende Punkte $P'(x|y|z)$. Diese Punkte liegen auf einer sogenannten *Hyperfläche*. Dazu ein Beispiel:

```
(%i9) plot3d(1/(1+x^2+y^2), [x,-3,3], [y,-3,3]);
```

Mit der Option `grid` kann die Anzahl der Gitterlinen in x- und y-Richtung verändert werden.

```
(%i10) plot3d(1/(1+x^2+y^2), [x,-3,3], [y,-3,3],
           [grid, 100, 100]);
```

8.6 Art der Darstellung verändern

Leider lässt sich die Darstellung der Grafiken (Farben, Linienstil, Koordinatenachsen, ...) im Gnuplot-Fenster nachträglich nur mühsam (wenn überhaupt) verändern. Daher sollte man die Grafik-Parameter mit dem `plot`-Befehl Gnuplot an übermitteln. Mit

```
(%i11) plot_options;
```

erhält man eine Liste, die wiederum kleinere Listen enthält. Eine der Sublisten (das sind Listen innerhalb von Listen) ist `[axes, true]`. Das bedeutet, dass der betreffende Parameter `axes` heisst und den Wert `true` hat. Mit anderen Worten: *Achsen werden gezeichnet!*

```
(%i12) set_plot_option([axes, false])$
```

können wir beispielsweise dafür sorgen, dass für alle folgenden `plot`-Befehle die Achsen nicht mehr dargestellt werden.

Wenn wir die Achsen aber nur für eine einzelne Grafik ein- oder ausschalten wollen, können wir dies innerhalb der Grafik ändern:

```
(%i13) plot2d(x^2 - 2*x, [x, -5, 5], [y, -5, 5],
           [axes, true])$
```

Hier eine Übersicht über oft verwendete Grafik-Optionen.

`[x,`*`xmin`*`,`*`xmax`*`]` Darstellungsbereich auf der x-Achse

`[y,`*`ymin`*`,`*`ymax`*`]` Darstellungsbereich auf der y-Achse

`[nticks,`*`n`*`]` Anzahl Stützstellen zur Kurvenberechnung

`[xlabel,`*`"text"`*`]` Beschriftung der x-Achse bei 2d-Plots

`[ylabel,`*`"text"`*`]` Beschriftung der y-Achse bei 2d-Plots

`[legend,`*`"text1"`*`,`*`"text2"`*`,...]` Legenden für die einzelnen Kurven in einem 2d-Plot

`[style,`*`style1`*`,`*`style2`*`,...]` Plotstile für die einzelnen Kurven in einem 2d-Plot in der Form `[name,w,c]`

`[png_file,` *`"Dateiname"`*`]` Ausgabeziel im PNG-Format

Aufgabe 8.1

Zeichne die Funktionen $f(x) = 0.5x^2$, $g(x) = x^2$ und $h(x) = 1.5x^2$ in ein Koordinatensystem.

```
(%i1) plot2d([0.5*x^2, x^2, 1.5*x^2], [x,-10,10], [y,-10,10]);
```

Aufgabe 8.2

Erzeuge mit

```
x : makelist(random(10.0), n, 1, 100);
y : makelist(random(10.0), n, 1, 100);
```

Zwei Listen mit je 100 zufälligen Punkten zwischen 0 und 9.999...

Stelle diese Punktmenge (ohne Verbindungslinien!) dar. Experimentiere mit verschiedenen Arten von Punkttypen. (Siehe in der Hilfe nach)

Achtung: Nach dem Plot müssen die Bindungen zwischen den Listen und `x` bzw. y aufgehoben werden. Andernfalls können `x` und `y` nicht mehr als Variable verwendet werden. (Alternative: Andere Variablen als x und y wählen.)

```
(%i2) x : makelist(random(10.0), n, 1, 100)$
      y : makelist(random(10.0), n, 1, 100)$
      plot2d([discrete, x, y], [xlabel, "x"],
         [ylabel, "y"]);
      kill(x, y)$
```

Aufgabe 8.3

Zeichne die folgende Parameterfunktion.

$x = \sin(3t)$

$y = \sin(4t)$

mit $t \in [0, 2\pi]$. Die Graphen von Parameterfunktionen mit trigonometrischen Funktionen in den beiden Komponenten werden Lissajous-Figuren genannt.

Hinweis: Setze die Option `[nticks, 200]`.

```
(%i3) plot2d([parametric, sin(3*t), sin(4*t),
          [t, 0, 2*%pi]], [nticks, 1000]);
```

Aufgabe 8.4

Stelle die folgende Parameterfunktion dar.

$x(t) = 2 \cdot \cos t$

$y(t) = 2 \cdot \left(\sin t + \sqrt{|\cos t|}\right)$ [Absolutbetrag: siehe Seite 4]

mit $t \in [0, 2\pi]$ und `[nticks, 500]`.

```
(%i4) x(t) := 2*cos(t)$
      y(t) := 2*(sin(t) + \sqrt(abs(cos(t))))$
      plot2d([parametric, x(t), y(t), [t,0,2*%pi]],
          [nticks, 500]);
```

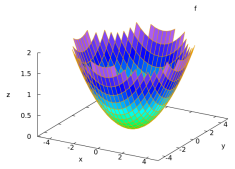
Aufgabe 8.5

Stelle den Graphen der Funktion

$f(x, y) = 0.1(x^2 + y^2)$

mit dem folgenden Kommando als 3D-Plot dar:

```
plot3d(0.1*(x**2 + y**2), [x,-10,10], [y,-10,10])$
```



Aufgabe 8.6

Stelle den Graphen der Funktion

$f(x, y) = \sin(x) + \sin(y)$

in gleicher Weise wie in der letzten Aufgabe dar. Hinweis: um mehr Gitterpunkte darzustellen, ist die Option `[grid, 50, 50]` vor das Ende des `plot3d(...)`-Kommandos einzugeben.

```
plot3d(sin(x)*sin(y), [x,-10,10], [y,-10,10], [grid,100,100])$
```