Einführung in *Mathematica* - Teil II -

zur Vorlesung *Mathematische Methoden der Physik* im WiSe 2013/14

: : Prof. Dr. Olaf Lechtenfeld und PD Dr. Michael Flohr : : 13. 11. 2013 : :

Analysis

Mathematica kann exellent mit allem umgehen, was mit Analysis zu tun hat. Differenzieren haben wir bereits in Teil I kennen gelernt. Natürlich kann *Mathematica* auch integrieren, und zwar so gut, dass, wenn es denn überhaupt eine in elementaren Funktionen ausdrückbare Stammfunktion gibt, sie von *Mathematica* auch gefunden wird. Und *Mathematica* löst auch Differentialgleichungen - die fundamentalen Gleichungen in der Physik.

Differentialgleichungen

Analytische Integration (Man beachte die Integrationskonstante `C[1]`.)

```
DSolve[y'[x] - x y[x] == 0, y[x], x]
```

$$\left\{\left\{y[x] \to e^{\frac{x^2}{2}} C[1]\right\}\right\}$$

Beim Zuweisen zu einer neuen Funktion `z` ist wiederum der Ersetzungsoperator notwendig; `[[1]]` "löst" sozusagen die verschachtelnde Liste auf.

```
z[x_] = y[x] /. DSolve[y'[x] - x y[x] == 0, y[x], x][[1]]
```

$$e^{\frac{x^2}{2}} C[1]$$

Nun kann, bis auf die verbleibende Konstante, mit der neuen Funktion numerisch gerechnet werden ...

```
z[1]
```

$$\sqrt{e}\, C[1]$$

```
N[%]
```

$$1.64872\, C[1]$$

Die Eleminierung der Konstanten erfolgt, wie gehabt, über eine Ersetzung mit einer Transformationsregel (`I` ist die komplexe Zahl, siehe Teil I):

```
% /. C[1] → I
```

$$0. + 1.64872\, i$$

Analytische Integration mit Randbedingung:

```
DSolve[{y''[x] + ω^2 y[x] == 0, y[0] == 1, y'[0] == 1/2}, y[x], x]
```

$$\left\{\left\{y[x] \to \frac{2\,\omega\,\text{Cos}[x\,\omega] + \text{Sin}[x\,\omega]}{2\,\omega}\right\}\right\}$$

Allgemeine Lösung einer Differentialgleichung in Form der sog. “reinen” Funktion (beachte, dass das Argument bei y nicht mit angegeben wird):

```
DSolve[y''[x] + ω^2 y[x] == 0, y, x]
```

```
{{y → Function[{x}, C[1] Cos[x ω] + C[2] Sin[x ω]]}}
```

Lösung für einen bestimmten Fall durch Substitution, vergleiche mit dem ersten Beispiel:

```
(y[x] /. %)[[1]]
```

```
C[1] Cos[x ω] + C[2] Sin[x ω]
```

Hinweis: Die Substitutionsanweisung schreibt man vorteilhafterweise in runde Klammern, bevor die Liste aufgelöst wird, denn *Mathematica* ist sehr logisch: Geschweifte Klammern erzeugten gleich wieder eine neue Liste, für die eine weitere Operation mit `[[1]]` nötig wäre.

```
{y[x] /. %%}[[1]][[1]]
```

```
C[1] Cos[x ω] + C[2] Sin[x ω]
```

Vektoranalysis

Die Vorlesung “Mathematische Methoden der Physik” macht Sie vor allem mit den Techniken der Vektoranalysis vertraut, die *Mathematica* natürlich auch beherrscht.

```
Clear[x, y, z, f, g, h]
```

Früher musste man hier erst ein Zusatzpaket laden, das aber mittlerweile in den Kern der grundlegenden *Mathematica*i Funktionen, die immer zur Verfügung stehen, integriert wurde. Überhaupt gibt es so viele spezielle Funktionen und Algorithmen in *Mathematica,* dass es keinen Sinn macht, alle diese bei jedem Programmstart vollständig zu laden. Statt dessen wir nur der als absolut essentiell erachtete Teil immer geladen, den Rest kann man bei Bedarf in Form von Paketen dazuladen, die sich jeweiligen speziellen Themen widmen.

```
<< VectorAnalysis`
```

```
General::obspkg : VectorAnalysis` is now obsolete. The legacy version being loaded may conflict with current
    Mathematica functionality. See the Compatibility Guide for updating information.
```

Hinweis: Zuerst sollte man das Koordinatensystems und seine Variablen festlegen (*Mathematica* gibt zwar **Cartesian**, d.h. “kartesisch”, aber konsistent zu seinen Groß- und Kleinschreibungsregeln **Xx**,**Yy** und **Zz** vor. Ein anderes mögliches Koordinatensystem ist z. B. **Spherical** mit den Variablen **Rr**, **Ttheta**, **Pphi**.)

```
SetCoordinates[Cartesian[x, y, z]]
```

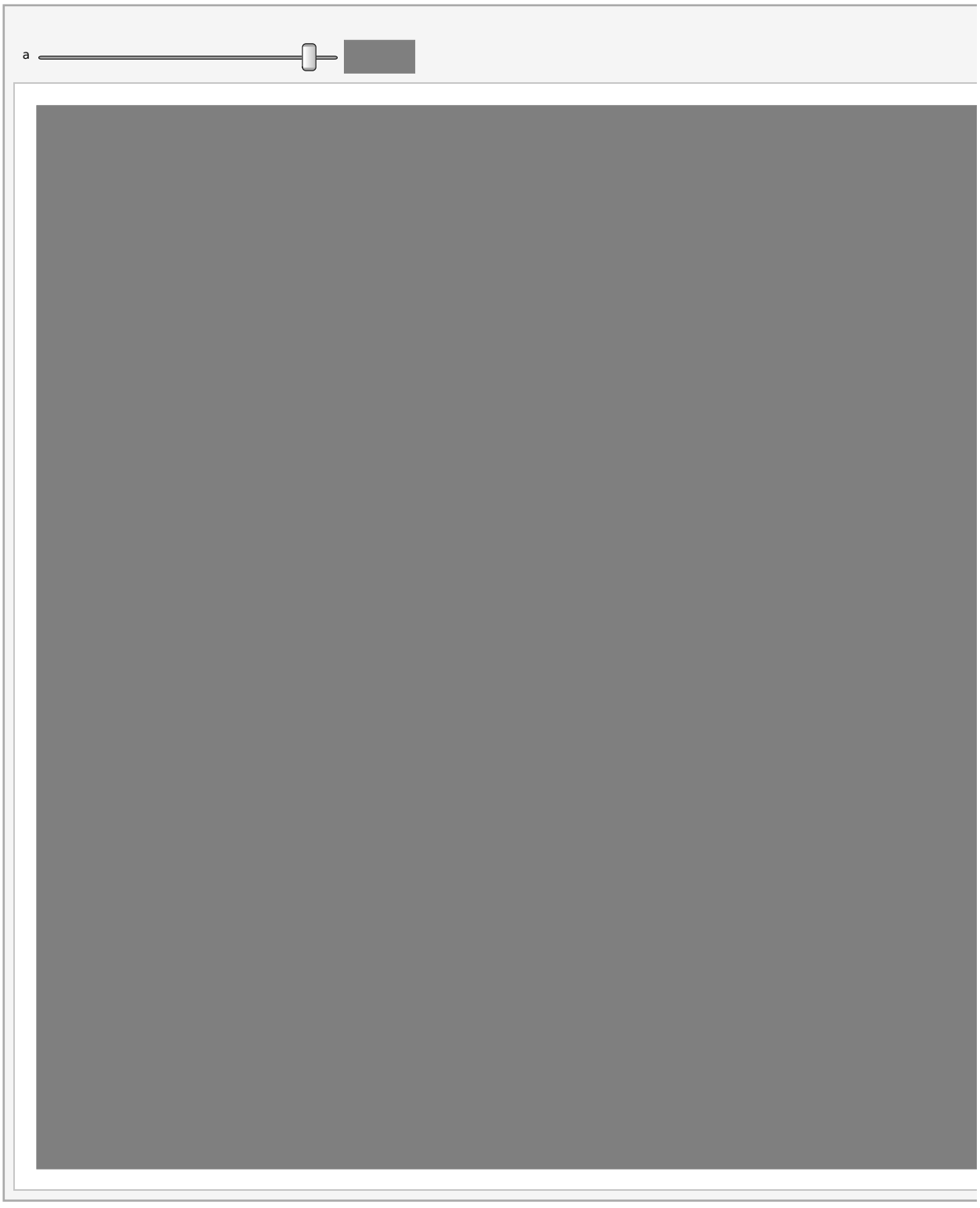
```
Cartesian[x, y, z]
```

Definition einer (skalaren) Funktion:

```
f[x_, y_, z_] := x^2 (3 - 2 y^2) - 2 (x^2 + y^2) z^2 + 3 (y^2 + z^2)
```

SIe sollen Äquipotentiallinien in der derzeitigen Hausübung zeichen. Hier plotten wir eine Äquipotentialfläche:

```
Manipulate[
 ContourPlot3D[f[x, y, z] == a, {x, -2, 2}, {y, -2, 2}, {z, -2, 2}], {{a, 3}, 1, 5}]
```



Berechnung des Gradienten von `f` im oben vorgegebenen Koordinatensystem:

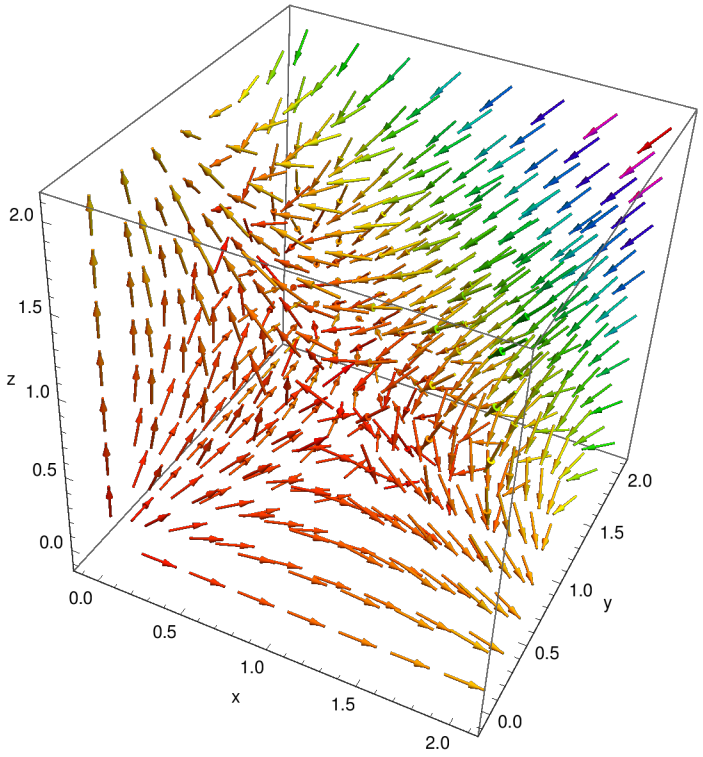
```
g = Grad[f[x, y, z]]
```

$$\left\{2\,x\,\left(3-2\,y^2\right)-4\,x\,z^2,\ 6\,y-4\,x^2\,y-4\,y\,z^2,\ 6\,z-4\,\left(x^2+y^2\right)\,z\right\}$$

Achtung: `g` ist keine Funktion gemäß der für `f` gebrauchten Definition, sondern eine "vektorartige" Liste!

Darstellung eines dreidimensionalen Vektorfeldes: Unter den benutzten Darstellungsoptionen sei die universelle, auch in `Plot` verwendbare `AxesLabel` hervorgehoben.

```
VectorPlot3D[g, {x, 0, 2}, {y, 0, 2}, {z, 0, 2},
 AxesLabel → {"x", "y", "z"}, VectorStyle → "Arrow3D",
 VectorScale → {Small, Small, None}, VectorColorFunction → Hue]
```



Wir berechnen nun die zwei grundlegenden Differentationen für Vektorfelder, die Divergenz und die Rotation. Letztere sollte für ein Gradientenfeld natürlich verschwinden:

```
h = Div[g]
r = Curl[g]
```

$$12-4\,x^2+2\,\left(3-2\,y^2\right)-4\,\left(x^2+y^2\right)-8\,z^2$$

$$\{0,\ 0,\ 0\}$$

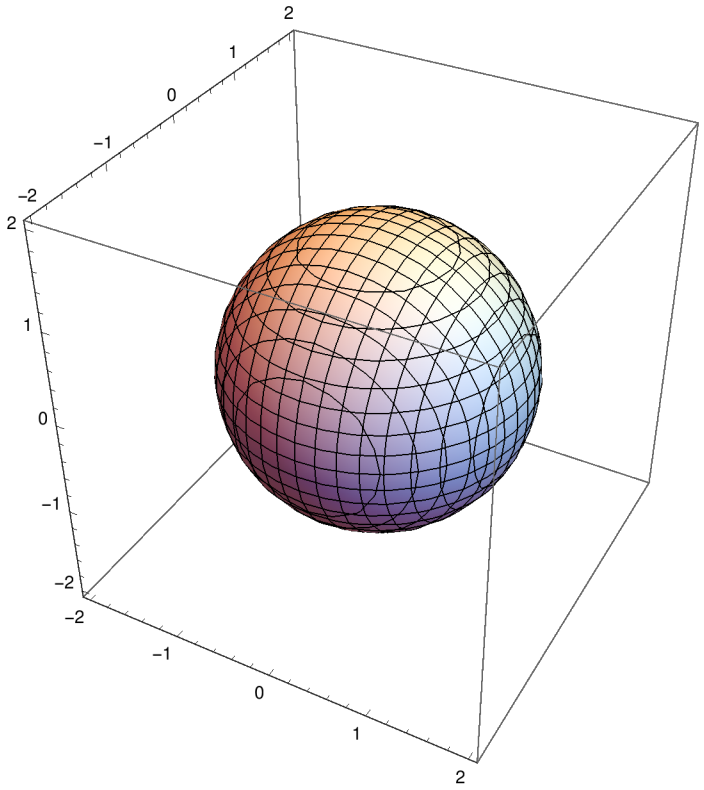
```
FullSimplify[%%]
```

$$-2\,\left(-9+4\,x^2+4\,y^2+4\,z^2\right)$$

Achtung: `h` ist ebenfalls keine Funktion gemäß der für `f` gebrauchten Definition, sondern eine "skalarartige" (also einelementige) Liste!

Wie man nach der Vereinfachung mittels **FullSimplify** sieht, ist diese Funktion rotationssymmetrisch:

```
ContourPlot3D[h == 0, {x, -2, 2}, {y, -2, 2}, {z, -2, 2}]
```



Listen

Mathematica arbeitet intern unglaublich stark mit dem Konzept von Listen. Das kann man daher ausnutzen, um sehr effizienten Code zu schreiben, oder manche mathematischen Probleme sehr einfach in *Mathematica* zu implementieren. Deshalb ist es wichtig, die grundlegenden Befehle zum Hantieren mit Listen zu kennen.

Listenerzeugung

Ein einfaches Beispiel zum Einstieg:

```
Table[n, {n, 0, 16}]
```

```
{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16}
```

```
Table[n, {n, 0, 16, 2}]
```

```
{0, 2, 4, 6, 8, 10, 12, 14, 16}
```

Tabellen sind im wesentlichen geordnete Listen. Und sie können ganz einfach, ineinander verschachtelt werden. Wir kennen das im Grunde schon ähnlich von den **Array**, die wir in Teil I zur Erzeu-

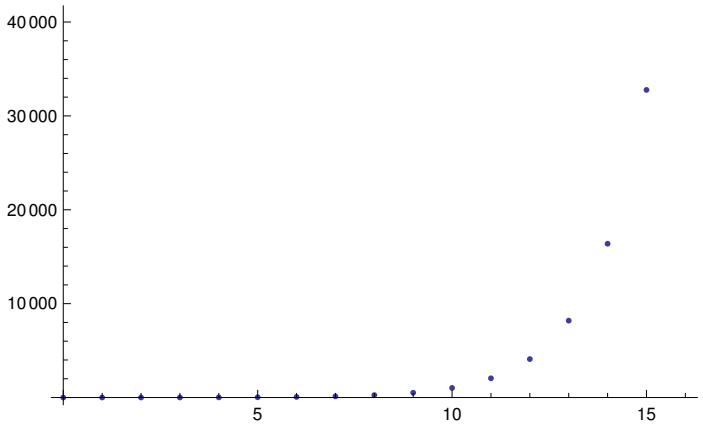
gung von Vektoren und Matrizen genutzt haben. **Table** kann also auch eine Liste von Listen generieren, beispielsweise Wertetabellen.

```
Table[{n, 2^n}, {n, 0, 16}]
```

```
{{0, 1}, {1, 2}, {2, 4}, {3, 8}, {4, 16}, {5, 32}, {6, 64}, {7, 128}, {8, 256}, {9, 512},
 {10, 1024}, {11, 2048}, {12, 4096}, {13, 8192}, {14, 16 384}, {15, 32 768}, {16, 65 536}}
```

Listen dieser Art mit numerischen Daten lassen sich unkompliziert graphisch darstellen.

```
ListPlot[%]
```



Symolisch geht's mit der Listenerzeugung freilich auch:

```
Table[(a + b)^n, {n, 0, 4}]
```

$$\{1, a+b, (a+b)^2, (a+b)^3, (a+b)^4\}$$

```
liste = Expand[%]
```

$$\{1, a+b, a^2+2\,a\,b+b^2, a^3+3\,a^2\,b+3\,a\,b^2+b^3, a^4+4\,a^3\,b+6\,a^2\,b^2+4\,a\,b^3+b^4\}$$

Listenmanipulationen

Zunächst wird beispielhaft das erste Listenelement der vorletzten Programmausgabe entfernt ...

```
Drop[%%, 1]
```

$$\{a+b, (a+b)^2, (a+b)^3, (a+b)^4\}$$

... und dann das erste "extrahiert".

```
First[%]
```

$$a+b$$

Den letzten Listeneintrag erhält man "natürlich" mit **Last**!

```
Last[liste]
```

$$a^4+4\,a^3\,b+6\,a^2\,b^2+4\,a\,b^3+b^4$$

Ausschneiden von Listenteilen erlaubt **Part**.

```
Part[liste, 2 ;; 3]
```

$\{a + b, a^2 + 2\,a\,b + b^2\}$

Oder alternativ auf diesem Weg (vergleiche mit dem Extrahieren von **Array** Elementen in Teil I):

```
liste[[2 ;; 3]]
```

$\{a + b, a^2 + 2\,a\,b + b^2\}$

Der Zugriff auf ein einzelnes Element erfolgt in gewohnter Manier:

```
%[[1]]
```

$a + b$

```
Table[{n, 1/n}, {n, 1, 7}]
```

$\{\{1, 1\}, \{2, \frac{1}{2}\}, \{3, \frac{1}{3}\}, \{4, \frac{1}{4}\}, \{5, \frac{1}{5}\}, \{6, \frac{1}{6}\}, \{7, \frac{1}{7}\}\}$

Ein extrem nützlicher Befehl: **Flatten** reduziert die beiden Ebenen zuvor generierter Liste auf eine.

```
Flatten[%]
```

$\{1, 1, 2, \frac{1}{2}, 3, \frac{1}{3}, 4, \frac{1}{4}, 5, \frac{1}{5}, 6, \frac{1}{6}, 7, \frac{1}{7}\}$

Die Summe aller Listenelemente läßt sich mit **Total** berechnen.

```
Total[%]
```

$\frac{4283}{140}$

Bedingungen

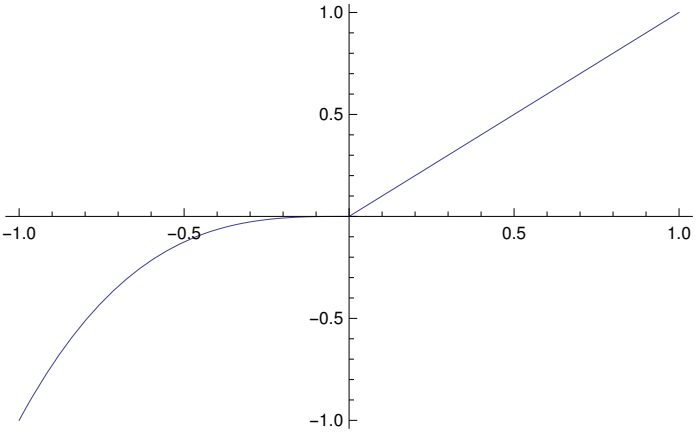
Mathematica ist nicht nur eine Ansammlung zahlloser Befehle und Datenstrukturen zum Umgang mit mathematischen Objekten, sondern eine vollwertige “High-Level” Programmiersprache. Wie jede vernünftige Programmiersprache bietet es die Möglichkeit, Bedingungen zu formulieren, auszuwerten, und dann in Abhängigkeit vom Ergebnis unterschiedlich weiter zu verfahren.

Abschnittsweise definierte Funktionen

Eine reelle Funktion, die auf zwei Intervallen unterschiedlich definiert ist,

```
f[x_] := If[x < 0, x^3, x]
```

```
Plot[f[x], {x, -1, 1}]
```



sowie deren bestimmtes Integral,

```
Integrate[f[x], {x, -1, 1}]
```

$$\frac{1}{4}$$

deren "Stammfunktion"

```
Integrate[f[x], x]
```

$$\begin{cases} \frac{x^4}{4} & x \leq 0 \\ \frac{x^2}{2} & \text{True} \end{cases}$$

und deren erste Ableitung

```
Clear[g]
```

```
g[x_] = D[f[x], x]
```

```
If[x < 0, 3 x^2, 1]
```

Wir wollen uns das in einem Plot ansehen. Wir hätten bei diesem Plot allerdings auch gerne eine schöne Legende, die die verschiedenen Funktionen, die wir plotten, auflistet. Es gibt zahllose solche “Sonderwünsche”, für die es in *Mathematica* bereits Lösungen in Form von Zusatzpaketen gibt. Damit das Programm beim Starten jedoch nicht eine riesige Menge solcher Zusatzfunktionen laden muss, wird nur ein kleiner Teil grundlegender Funktionen geladen, der Rest kann in Form dieser Zusatzpakete, falls benötigt, jederzeit nachgeladen werden.
So musste das Paket **`PlotLegends`** zum Erzeugen einer Legende zuerst geladen werden. Das Laden eines Paketes geht mit folgender Syntax, wobei dieses spezielle Paket inzwischen zum grundlegenden Kern von *Mathematica* zählt und nicht mehr geladen werden müsste.

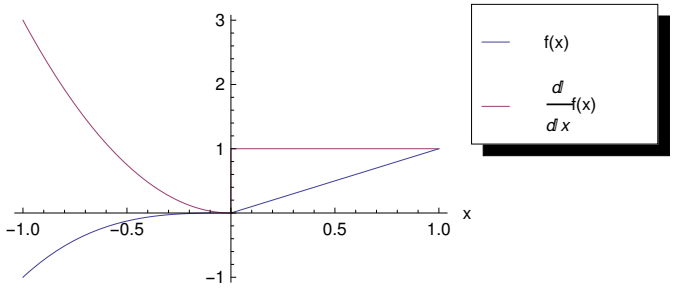
```
<< PlotLegends`
```

LegendPosition::shdw : Symbol LegendPosition appears in multiple contexts {PlotLegends`, Global`}; definitions in context PlotLegends` may shadow or be shadowed by other definitions. »

PlotLegend::shdw : Symbol PlotLegend appears in multiple contexts {PlotLegends`, Global`}; definitions in context PlotLegends` may shadow or be shadowed by other definitions. »

General::obspkg : PlotLegends` is now obsolete. The legacy version being loaded may conflict with current Mathematica functionality. See the Compatibility Guide for updating information.

```
Plot[{f[x], g[x]}, {x, -1, 1}, AxesLabel → {"x"},
 PlotLegend → {"f(x)", "d/dx f(x)"}, LegendPosition → {1, 0}]
```



Weitere Anwendungen

Hier kombinieren wir das bisher gelernte über Listen und Bedingungen in ein paar wenigen Beispielen, die sich um Primzahlen drehen.

```
primzahlen = Table[Prime[i], {i, 20}]
```

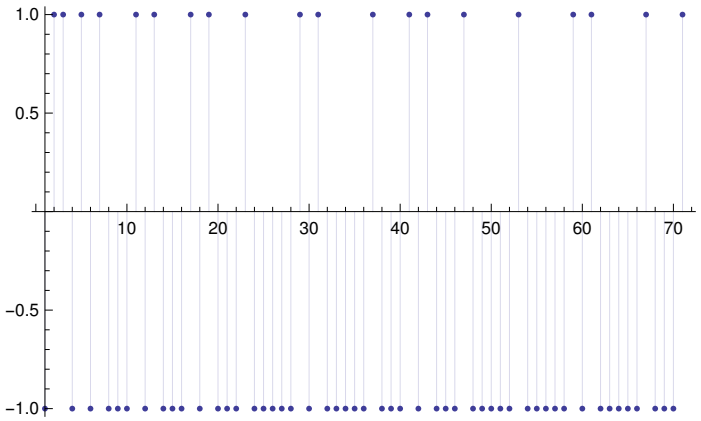
```
{2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71}
```

```
j = Last[primzahlen]
```

```
71
```

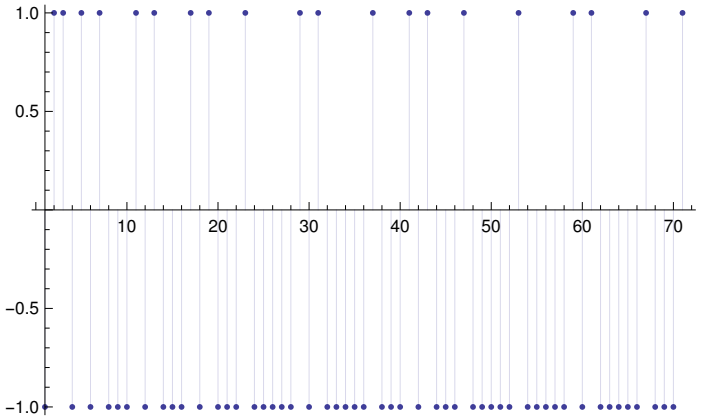
Die Funktion **MemberQ** durchsucht eine Liste darauf hin, ob ein bestimmtes Element enthalten ist.

```
DiscretePlot[If[MemberQ[primzahlen, i], 1, -1], {i, 1, j}]
```



Durchsuchen ist aber eigentlich gar nicht nötig, da die Liste aufsteigend sortiert ist und aufsteigend "abgearbeitet" wird.

```
DiscretePlot[If[i == First[primzahlen], primzahlen = Drop[primzahlen, 1];
  1, -1], {i, 1, j}]
```



Achtung: Nach obiger "Abarbeitung" der Primzahlenliste ist diese leer, denn die Liste wurde tatsächlich während der Auswertung bei jedem Schritt um ein Element verkürzt.

```
primzahlen
```

```
{}
```

Logische Ausdrücke

Mathematica kann in vielfältiger Weise logische Ausdrücke konstruieren und auswerten. Jeder auswertbare logische Ausdruck basiert auf Grundeinheiten, die meist als Vergleich formuliert werden können. Alle Vergleiche können nur zwei Werte annehmen: Wahr oder falsch.

```
2^0 == 1
```

```
True
```

```
2 < 0
```

```
False
```

Verknüpfung zweier Vergleiche mit "und" bzw. "oder"

```
3 > E && 3 < Pi
```

```
True
```

Äquivalente Eingabe:

```
3 > E ∧ 3 < Pi
```

```
True
```

```
3 < E || 3 < Pi
```

```
True
```

Äquivalente Eingabe:

```
3 > E ∨ 3 > Pi
```

```
True
```

Negation

```
! (a > b)
```

```
a ≤ b
```

Verschachtelte `If`-Befehle

```
Integrate[If[x < 0, If[x == -1, 1, 2], 3], x]
```

```
⎧ 2 x   x ≤ -1 || -1 < x ≤ 0
⎩ 3 x   True
```

Schleifenstrukturen und Iteration

Der nächste wichtige Typ von Strukturbefehlen, der *Mathematica* zu einer vollen und extrem leistungsfähigen Programmiersprache macht, sind Schleifen, mit denen die gleiche Sequenz von Befehlen mehrfach ausgeführt werdden können.

```
i = 0;
While[i < 3, Print[i]; i = i + 1]
```

```
0
```

```
1
```

```
2
```

Eine klassische "`for`"-Schleife, wie sie wohl nahezu jede (imperative) Programmiersprache kennt:

```
For[i = 0, i < 3, i = i + 1, Print[i]]
```

```
0
```

```
1
```

```
2
```

In *Mathematica* gibt es auch das einfachere, zu **`For`** äquivalente **`Do`**,

```
Do[Print[i], {i, 0, 2}]
```

```
0
```

```
1
```

```
2
```

welches von der Struktur beispielsweise analog zu **Table** ist.

```
Table[i, {i, 0, 2}]
```

```
{0, 1, 2}
```

Anwendungen

Wozu ist das gut? Hier ein paar wirklich extrem simple Beispiele. Wie alles in *Mathematica* können auch diese Strukturen beliebig ineinander verschachtelt werden. Es hat auch seinen Grund, dass viele Dinge in *Mathematica* auf verschiedene Weisen erreicht werden können. So hat jede der verschiedenen Methoden einer **For** Schleife besondere Vorzüge für bestimmte Anwendungen, was zu schnellerem und effizienterem Code führt, wenn man sich damit auskennt.

```
For[i = 0, i < 3, i = i + 1, k = i]
```

```
k
```

```
2
```

Eine (Summations)schleife in mathematisch etwas vertrauterer Darstellung

```
k = Sum[1, {i, 1, 10}]
```

```
10
```

Dieselbe sieht "programmiert" wie folgt aus:

```
k = 0;
For[i = 1, i ≤ 10, i = i + 1, k = k + 1]
k
```

```
10
```

Ein weiteres Beispiel, die Gauß-Summe

```
k = ∑_{i=1}^{10} i
```

```
55
```

```
k = 0;
For[i = 1, i ≤ 10, i = i + 1, k = k + i]
k
```

```
55
```

Am Rande sei erwähnt, dass *Mathamtica* die Gauß-Summe natürlich auch für eine unbestimmte obere Grenze symbolisch ausrechnen kann.

```
k = ∑_{i=1}^{N} i
```

$$\frac{1}{2} N \left(1 + N\right)$$

Fakultät, iterativ

Und hier noch ein Klassiker, die iterative Definition der Fakultät:

```
fi[n_] := Product[m, {m, 2, n}]
```

```
fi[137]
```

```
5 012 888 748 274 991 661 034 926 292 112 253 883 237 205 694 398 754 483 388 962 668 892 510 972 ↓
  746 226 260 034 675 717 797 072 343 372 830 591 567 227 826 571 884 373 881 355 612 819 314 826 ↓
  377 917 827 129 740 056 802 397 016 509 378 163 883 274 055 583 382 110 208 000 000 000 000 000 ↓
  000 000 000 000 000 000
```

Und zur Kontrolle hier mit dem in *Mathematica* eingebauten Befehl

```
137!
```

```
5 012 888 748 274 991 661 034 926 292 112 253 883 237 205 694 398 754 483 388 962 668 892 510 972 ↓
  746 226 260 034 675 717 797 072 343 372 830 591 567 227 826 571 884 373 881 355 612 819 314 826 ↓
  377 917 827 129 740 056 802 397 016 509 378 163 883 274 055 583 382 110 208 000 000 000 000 000 ↓
  000 000 000 000 000 000
```

Rekursion

Mathematica kann hervorragend mit rekursiv definierten Strukturen umgehen. Das ist zwar nicht immer die effizientere Lösung, aber oft elegant und in vielen Fällen einfacher zu programmieren

Fakultät, rekursiv

```
fr[n_] := n fr[n - 1]
```

```
fr[1] = 1;
```

```
fr[137]
```

```
5 012 888 748 274 991 661 034 926 292 112 253 883 237 205 694 398 754 483 388 962 668 892 510 972 ↓
  746 226 260 034 675 717 797 072 343 372 830 591 567 227 826 571 884 373 881 355 612 819 314 826 ↓
  377 917 827 129 740 056 802 397 016 509 378 163 883 274 055 583 382 110 208 000 000 000 000 000 ↓
  000 000 000 000 000 000
```

Die Funktion **Timing** gibt die Verarbeitungszeit in Sekunden an.

```
Timing[fr[10 000];]
```

```
$RecursionLimit::reclim : Recursion depth of 1024 exceeded. ≫

{0.042090, Null}
```

Um eine Systemvariable wie **$RecursionLimit** nur in einer Eingabezeile zu verändern, muß die beeinflußte Funktion in **Block** verschachtelt werden.

```
Timing[Block[{$RecursionLimit = 10 004}, rf[10 000];]]
```

```
{0.000019, Null}
```

Hier ein Geschwindigkeitsvergleich der drei Möglichkeiten:

```
Timing[Block[{$RecursionLimit = 10 004}, rf[10 000];]]
```

```
{0.000019, Null}
```

```
Timing[fi[10 000];]
```

```
{0.010920, Null}
```

```
Timing[10 000!;]
```

```
{0.001345, Null}
```

Aha! Die iterative Definition ist die schlechteste. Dies liegt daran, dass bei der iterativen Definition sehr viele Zwischenergebnisse erzeugt werden, was wiederum daran liegt, dass die **`Product`** Funktion im Grunde in einer Art **`For`** Schleife implementiert ist. Die rekursive Definition hingegen arbeitet "in place",und erledigt daher die 10.000-fache Multiplikation effizienter. Und die eingebaute Funktion ist ohnehin wesentlich komplizierter programmiert, und hat einen sogenannten "overhead", weil sie erst einmal prüft, ob das Argument eine einfache Berechnung mit natürlichen Zahlen zulässt, eine symbolisch exakte Berechnung, eine numerische, oder gar keine.
Ergebnis des Geschwindigkeitsvergleiches. Die eingebauten Funktionen von *Mathematica* sind hochopti miert und meist unschlagbar schnell. Dass die rekursive Definition hier gewinnt, liegt daran, dass *Mathematica* einmal berechnete Werte cached, und das die eingebaute Funktion für die Fakultät wesentlich mehr kann. Dahinter steckt nämlich die Γ-Funktion, die analytische Fortsetzung der Fakultät auf die ganzen komplexen Zahlen.

Und wieder eine Randbemerkung: Die interne Definition der Fakultät ist in der Tat in Wirklichkeit viel allgemeiner, denn sie kann auch auf beliebige reelle (und komplexe!) Zahlen angewandt werden. In Wirklichkeit ist dies die Gammafunktion:

```
5.5!
```

```
287.885
```

```
11/2!
```

$$\frac{10\,395\,\sqrt{\pi}}{64}$$

Einführung in *Mathematica* - Teil 1 -

zur Vorlesung *Mathematische Methoden der Physik* im WiSe 2013/14

: : Prof. Dr. Olaf Lechtenfeld und PD Dr. Michael Flohr : : 06. 11. 2013 : :

Vorbemerkungen

- *Mathematica* unterscheidet strikt zwischen Groß- und Kleinschreibung. Alle internen Befehle, Symbole, Funktionen, Konstanten usw. beginnen mit einem Großbuchstaben, bspw. `Plot`. Zusammensetzungen enthalten auch große Buchstaben innerhalb der Bezeichnung, z. B. `ParametricPlot`.
- *Mathematica* führt Kommandos nach Eingabe durch die Tastenkombination SHIFT + ENTER aus, während ein einfaches ENTER lediglich einen Zeilenumbruch zur besseren Gliederung der Eingabe erzeugt.
- Wenn Sie dieses Notebook zu Hause selber durcharbeiten, sollten Sie manchmal nicht nur meine Eingaben ausführen lassen, sondern darunter neue Eingabebereiche öffnen und eingene Eingaben ausprobieren. Denn durch dieses Ausprobieren lernt man am allerschnellsten, wie die Dinge funktionieren. Am Ende jedes Ein-/Ausgabebereiches erscheint ganz links ein + Zeichen, wenn man dort mit der Maus klickt oder mit dem Cursor nach unten unter das Ende des Bereiches geht. Dies erlaubt die Auswahl der Art des neuen Eingabebereiches. Ganz rechts sieht man lange Klammerungen, die die Bereiche anzeigen. Diese sind in einem Dokument of verschachtetl, da ein einzelner Eingabebereich zum Beispiel Teil eines Abschmitts sein kann, der Teil eines Kapitels ist usw.

Arithmetik

Elementarer geht es kaum: Selbstverständlich kann *Mathematica* auch, was man von einem Taschenrechner erwartet! Die Eingabe wird jeweils nach Drücken von SHIFT + ENTER am Zeilenende ausgewertet.

```
5 - 4
```

```
1
```

(Grundsätzliche Nebenbemerkung: Programmintern wird zur Befehlsverarbeitung eine sog. Infix-Notation verwendet, die in Baumstrukturen beliebig verschachtelt werden kann.)

```
Plus[5, Times[-1, 4]]
```

```
1
```

Als Multiplikationszeichen kann Stern eingegeben werden.

```
7 * 9
```

```
63
```

Gebräuchlicher ist jedoch schlicht ein Leerzeichen, welches *Mathematica* bei Zahlen automatisch durch ein Kreuz ersetzt.

```
7 × 9
```

```
63
```

Probieren Sie dies nun selbst aus, indem Sie unter diesem Satz einen *Mathematica*Input Eingabebereich erzeugen, zwei durch ein Leerzeichen getrennte beliebige Zahlen eingeben, und dies ausführen lassen.

Potenzen werden durch das Symbol **^** erzeugt.

```
2^64
```

```
18 446 744 073 709 551 616
```

Mathematica bleibt exakt, wo immer möglich.

```
2 / 12
```

$$\frac{1}{6}$$

Eine Gleitkommadarstellung des vorausgegangenen Ergebnisses bekommt man mit dem Befehl **N**. Befehlsargumente werden immer in eckigen Klammer eingegeben. Die hier verwendete **%**-Variable ist stets mit der letzten (**%%** mit der vorletzten) Ausgabe "befüllt", was eine schnelle Methode darstellt, auf die letzten paar Ergebnisse zuzugreifen.

```
N[%]
```

```
0.16666 7
```

Der Dezimalpunkt kennzeichnet reelle Zahlen, die Dezimaldarstellung wird dadurch erzwungen.

```
2. / 12
```

```
0.16666 7
```

Konstanten sind, wie man auf diesen etwas umständlichen Wegen sieht, eingebaut.

```
Exp[1]
```

e

```
2 ArcCos[0]
```

π

Numerische Berechnungen bzw. Ausgaben können mit beliebiger Genauigkeit erfolgen: Zweites (optionales) Argument von **N** ist die Anzahl der Nachkommastellen. (Man beachte, dass *Mathematica* bei der obigen Ausgabe der Konstanten eine typographisch ansprechende Form - π und e - verwendet; die Eingabe kann hingegen (auch) über den lateinischen Zeichensatz der Tastatur im internen Format - **Pi** und **E** - erfolgen.)

```
N[Pi, 100]
```

```
3.14159 26535 89793 23846 26433 83279 50288 41971 69399 37510 58209 74944 59230 78164 ↘
  06286 20899 86280 34825 34211 7068
```

```
N[E, 1000]
```

```
2.71828 18284 59045 23536 02874 71352 66249 77572 47093 69995 95749 66967 62772 40766 ⁞
 30353 54759 45713 82178 52516 64274 27466 39193 20030 59921 81741 35966 29043 57290 03342 ⁞
 95260 59563 07381 32328 62794 34907 63233 82988 07531 95251 01901 15738 34187 93070 21540 ⁞
 89149 93488 41675 09244 76146 06680 82264 80016 84774 11853 74234 54424 37107 53907 77449 ⁞
 92069 55170 27618 38606 26133 13845 83000 75204 49338 26560 29760 67371 13200 70932 87091 ⁞
 27443 74704 72306 96977 20931 01416 92836 81902 55151 08657 46377 21112 52389 78442 50569 ⁞
 53696 77078 54499 69967 94686 44549 05987 93163 68892 30098 79312 77361 78215 42499 92295 ⁞
 76351 48220 82698 95193 66803 31825 28869 39849 64651 05820 93923 98294 88793 32036 25094 ⁞
 43117 30123 81970 68416 14039 70198 37679 32068 32823 76464 80429 53118 02328 78250 98194 ⁞
 55815 30175 67173 61332 06981 12509 96181 88159 30416 90351 59888 85193 45807 27386 67385 ⁞
 89422 87922 84998 92086 80582 57492 79610 48419 84443 63463 24496 84875 60233 62482 70419 ⁞
 78623 20900 21609 90235 30436 99418 49146 31409 34317 38143 64054 62531 52096 18369 08887 ⁞
 07016 76839 64243 78140 59271 45635 49061 30310 72085 10383 75051 01157 47704 17189 86106 ⁞
 87396 96552 12671 54688 95703 5035
```

Mehrere Anweisungen lassen sich in einer Zelle zusammenfassen. Aus Gründen der Übersichtlichkeit sollte mit dieser kompakten Eingabeform jedoch vorsichtig umgegangen werden.

```
137!
N[%]
```

```
5 012 888 748 274 991 661 034 926 292 112 253 883 237 205 694 398 754 483 388 962 668 892 510 972 ⁞
 746 226 260 034 675 717 797 072 343 372 830 591 567 227 826 571 884 373 881 355 612 819 314 826 ⁞
 377 917 827 129 740 056 802 397 016 509 378 163 883 274 055 583 382 110 208 000 000 000 000 000 ⁞
 000 000 000 000 000 000
```

5.01289×10^{234}

Algebra

Definition von Funktionen

Neue Funktionen (Kleinschreibung empfohlen, um Konflikte mit den Namen der wirklich unglaublich zahlreichen in *Mathematica* eingebauten Funktionen zu vermeiden) werden wie folgt definiert, wobei der Doppelpunkt die Auswertung der rechten Seite verhindert, ...

```
f[x_] := x^6 - 2 x^5 - 30 x^4 + 36 x^3 + 190 x^2 - 36 x - 150
```

... und verhalten sich wie interne Befehle (die stets einen großen Anfangsbuchstaben haben). Bei Funktionsdefinitionen mit **:=** erfolgt keine bestätigende Ausgabe. Man beachte ferner, daß bei einem Produkt aus einer Zahl und einem Symbol ein Leerzeichen zwischen beiden Faktoren automatisch eingefügt wird. Da Variablennamen aus mehreren Buchstaben bestehen dürfen, muß/müssen beim Produkt von Symbolen das/die Leerzeichen manuell eingegeben werden.

Wichtig: An zu "befüllende" Variablen eines Befehles (jede Funktion stellt im Grunde einen Befehl dar),

hier **x**, muß das Zeichen _ angehängt werden. Alle nicht deklarierten Symbole werde als Konstanten angesehen, siehe unten.

```
f[2903]
```

```
598 110 410 321 877 793 837
```

Die Zuweisung eines Wertes zu einer Variablen erfolgt mit dem üblichen Gleichheitszeichen. Hier unterdrückt das abschließende Semikolon die Bestätigungsantwort.

```
r = 17;
```

```
f[r]
```

```
19 023 241
```

"Typographische" Eingaben, auch solche von griechischen Buchstaben, Relationssymbolen, Operatoren etc. können aus Paletten (siehe Writing Assistent im Menü Palettes) an der aktuellen Schreibposition eingegeben werden.

In folgendem Beispiel ist μ ein Argument der Funktion λ, ν hingegen nicht.

```
λ[μ_] := μ^ν
```

```
λ[2]
```

$$2^{\nu}$$

```
ν = 4
```

```
4
```

```
λ[2]
```

```
16
```

Selbstdefinierte Befehle und Variablen werden wie folgt gelöscht:

```
Clear[λ, ν]
```

Hier sind nun sowohl μ wie auch ν Funktionsargumente.

```
λ[μ_, ν_] := μ^ν
```

```
λ[2, 4]
```

```
16
```

Symbolische Argumente - also solche, die Variablen ohne zugewiesenen numerischen Wert enthalten - werden eingesetzt und der Gesamtausdruck gegebenenfalls vereinfacht und sortiert.

```
λ[(ξ - 2), 3]
```

$$(-2+\xi)^3$$

```
f[1 / (2 t)]
```

$$-150+\frac{1}{64\,t^6}-\frac{1}{16\,t^5}-\frac{15}{8\,t^4}+\frac{9}{2\,t^3}+\frac{95}{2\,t^2}-\frac{18}{t}$$

Auflösen von Gleichungen

Die Nullstellenfindung ist hier analytisch möglich, und wird dann von *Mathematica* auch analytischdurchgeführt. Das Symbol == bezeichnet die Gleichheit zweier Ausdrücke.

```
Solve[x^2 + p x + q == 0, x]
```

$$\left\{\left\{x \to \frac{1}{2}\left(-p - \sqrt{p^2 - 4\,q}\right)\right\}, \left\{x \to \frac{1}{2}\left(-p + \sqrt{p^2 - 4\,q}\right)\right\}\right\}$$

Mathematica liefert mehrere Ergebnisse stets als Liste, eingefaßt in geschweifte Klammern, zurück; genauergesagt handelt es sich bei voriger Ausgabe um eine (zweielementige) Liste aus (einelementigen) Listen.

Der Zugriff auf ein Listenelement erfolgt über dessen Nummer (beginnend mit 1) behelfs des Zugriffsbefehls aus `[[` und `]]`, was dann so aussiieht ...

```
%[[2]]
```

$$\left\{x \to \frac{1}{2}\left(-p + \sqrt{p^2 - 4\,q}\right)\right\}$$

... und was sich mehrfach aneinandergereiht werwenden läßt.

```
%%[[2]][[1]]
```

$$x \to \frac{1}{2}\left(-p + \sqrt{p^2 - 4\,q}\right)$$

Der Pfeil → symbolisiert eine Transformationsregel; dieses in *Mathematica* wichtige Zeichen wird bei der Eingabe automatisch aus `->` erzeugt.

Der Ersetzungsoperator

Wird nur die rechte Seite obiger Lösung benötigt, muß mit dem Ersetzungsoperator `/.` "extrahiert" werden: `x` wird hier durch das ersetzt, wozu `x` im vorherigen Ergebnis "transformiert" wird - oder schlichter gesagt: Die Transformationsregel wird für `x` explizit ausgeführt. Allgemein erwartet der Ersetzungsoperator links stets einen Ausdruck und rechts eine Transformationsregel.

```
x /. %
```

$$\frac{1}{2}\left(-p + \sqrt{p^2 - 4\,q}\right)$$

```
f[x] /. %%
```

$$-150 - 18\left(-p + \sqrt{p^2 - 4\,q}\right) + \frac{95}{2}\left(-p + \sqrt{p^2 - 4\,q}\right)^2 + \frac{9}{2}\left(-p + \sqrt{p^2 - 4\,q}\right)^3 - \frac{15}{8}\left(-p + \sqrt{p^2 - 4\,q}\right)^4 - \frac{1}{16}\left(-p + \sqrt{p^2 - 4\,q}\right)^5 + \frac{1}{64}\left(-p + \sqrt{p^2 - 4\,q}\right)^6$$

Wie SIe vielleicht wissen, ist die Nullstellenfindung für ein Polynom vom Grade größer als vier im allgemeinen analytisch nicht möglich. *Mathematica* gibt hier keinen Fehler, sondern die Wurzeln als

"allgemeine Lösung" in Form der Vorstufe zur numerischen Lösung (gekennzeichnet mit **Root**) aus. Unser Polynom sechsten Grades hat also sechse Wurzeln:

```
Solve[f[x] == 0, x]
```

$$\{\{x \to \text{Root}[-150 - 36\,\#1 + 190\,\#1^2 + 36\,\#1^3 - 30\,\#1^4 - 2\,\#1^5 + \#1^6\,\&,\, 1]\},$$
$$\{x \to \text{Root}[-150 - 36\,\#1 + 190\,\#1^2 + 36\,\#1^3 - 30\,\#1^4 - 2\,\#1^5 + \#1^6\,\&,\, 2]\},$$
$$\{x \to \text{Root}[-150 - 36\,\#1 + 190\,\#1^2 + 36\,\#1^3 - 30\,\#1^4 - 2\,\#1^5 + \#1^6\,\&,\, 3]\},$$
$$\{x \to \text{Root}[-150 - 36\,\#1 + 190\,\#1^2 + 36\,\#1^3 - 30\,\#1^4 - 2\,\#1^5 + \#1^6\,\&,\, 4]\},$$
$$\{x \to \text{Root}[-150 - 36\,\#1 + 190\,\#1^2 + 36\,\#1^3 - 30\,\#1^4 - 2\,\#1^5 + \#1^6\,\&,\, 5]\},$$
$$\{x \to \text{Root}[-150 - 36\,\#1 + 190\,\#1^2 + 36\,\#1^3 - 30\,\#1^4 - 2\,\#1^5 + \#1^6\,\&,\, 6]\}\}$$

```
N[%]
```

$$\{\{x \to -4.42228\},\ \{x \to -2.14285\},\ \{x \to -0.937347\},$$
$$\{x \to 0.972291\},\ \{x \to 3.35802\},\ \{x \to 5.17217\}\}$$

Numerische Nullstellensuche bekommt man so auch auf direktem Wege mit dem Ergebnis auf zehn Nachkommastellen, "natürlich" in Listenform:

```
NSolve[f[x] == 0, x, 10]
```

$$\{\{x \to -4.422280787\},\ \{x \to -2.142852331\},\ \{x \to -0.937347323 0\},$$
$$\{x \to 0.972291061 0\},\ \{x \to 3.358016638\},\ \{x \to 5.172172742\}\}$$

Einige weitere algebraische Manipulationen

```
x y^6
```

$$x\,y^6$$

Anwendung (**/.**) einer neuen Transformationsregel (**y→v+w**) auf die vorherige Eingabe, also die Substitution von Symbolen durch algebraische Ausdrücke:

```
% /. y → v + w
```

$$(v + w)^6\,x$$

Bei der Manipulation algebraischer Ausdrücke sind folgende grundsätzlicheTechniken extrem hilfreich:

Ausmultiplizieren ...

```
Expand[%]
```

$$v^6\,x + 6\,v^5\,w\,x + 15\,v^4\,w^2\,x + 20\,v^3\,w^3\,x + 15\,v^2\,w^4\,x + 6\,v\,w^5\,x + w^6\,x$$

... und wieder zusammenfassenm, bzw. allgemein nach einer ganzen Reihe von Regeln vereinfachen:

```
Simplify[%]
```

$$(v + w)^6\,x$$

Eine weitere oft nützliche Vereinfachung kann das Faktorisieren sein:

```
Factor[3 628 800 - 10 628 640 x + 12 753 576 x^2 - 8 409 500 x^3 +
   3 416 930 x^4 - 902 055 x^5 + 157 773 x^6 - 18 150 x^7 + 1320 x^8 - 55 x^9 + x^10]
```

$$(-10+x)\,(-9+x)\,(-8+x)\,(-7+x)\,(-6+x)\,(-5+x)\,(-4+x)\,(-3+x)\,(-2+x)\,(-1+x)$$

Reihenentwicklungen

Mathematica kann auf sehr allgemeine Weise Reihenentwicklungen von Funktionen berechnen. Am bekanntesten ist natürlich die Taylor- bzw. Potenzreihenentwicklung. Hier diejenige der Exponentialfunktion um 0 bis einschließlich 7. Ordung:

```
Series[Exp[x], {x, 0, 7}]
```

$$1+x+\frac{x^2}{2}+\frac{x^3}{6}+\frac{x^4}{24}+\frac{x^5}{120}+\frac{x^6}{720}+\frac{x^7}{5040}+O[x]^8$$

Oft benötigt man das "Abschneiden" des symbolischen Restterms:

```
Normal[%]
```

$$1+x+\frac{x^2}{2}+\frac{x^3}{6}+\frac{x^4}{24}+\frac{x^5}{120}+\frac{x^6}{720}+\frac{x^7}{5040}$$

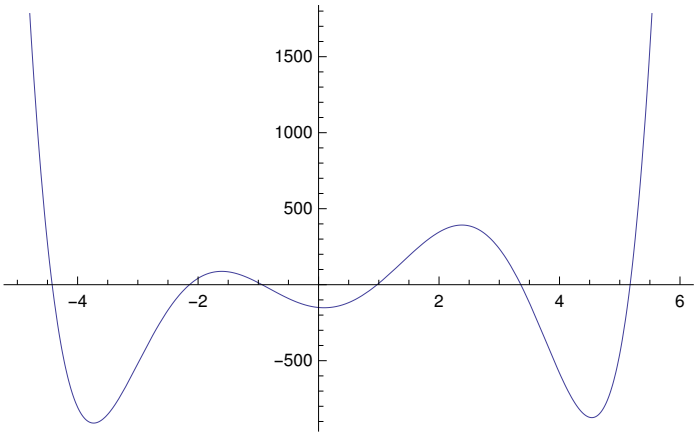
Analysis

Graphische Darstellung von Funktionen

Eine der herausragenden Fähigkeiten von *Mathematica* ist es, Resultate graphisch darstellen zu können. Wir Menschen können oft aus Bildern sehr viel mehr schnell ersehen, als aus komplizierten mathematischen Ausdrücken.

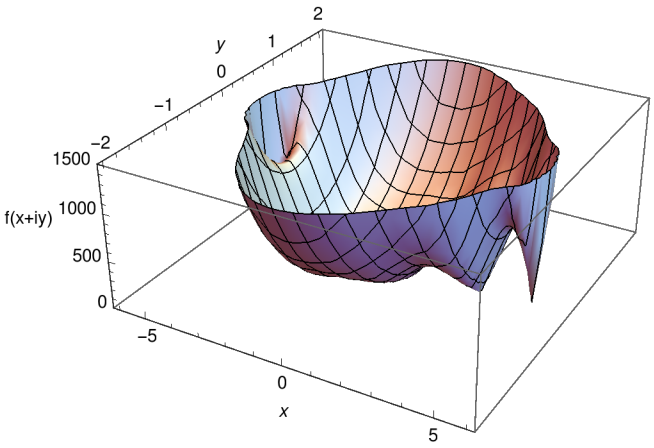
Mathematica wählt viele Darstellungsparameter recht vernünftig.

```
Plot[f[x], {x, -5, 6}]
```



Man kann auch sehr schön drei-dimensionale Plots erstellen, die man dann mit der Maus schön drehen und so von allen Seiten bewundern kann:

```
Plot3D[Abs[f[x + I y]], {x, -6, 6}, {y, -2, 2}, PlotRange → {0, 1500},
 AxesLabel → {x, y, "f(x+iy)"}, ClippingStyle -> None]
```



Ach ja, bei komplexen Zahlen bezeichnet `I` die imaginäre Einheit, und *Mathematica* rechnet mit komplexen Zahlen natürlich ganz natürlich:

```
I^2
```

```
-1
```

Differenzieren

Erst einmal die erste Ableitung: Die Angabe einer 1 im Befehl ist optional. (Achtung: bei einer solchen Funktionsdefinition darf kein Doppelpunkt vor dem Gleichheitszeichen stehen, da ja der Differentiationsbefehl ausgeführt werden soll.)

```
g[x_] = D[f[x], {x, 1}]
```

$$-36 + 380\,x + 108\,x^2 - 120\,x^3 - 10\,x^4 + 6\,x^5$$

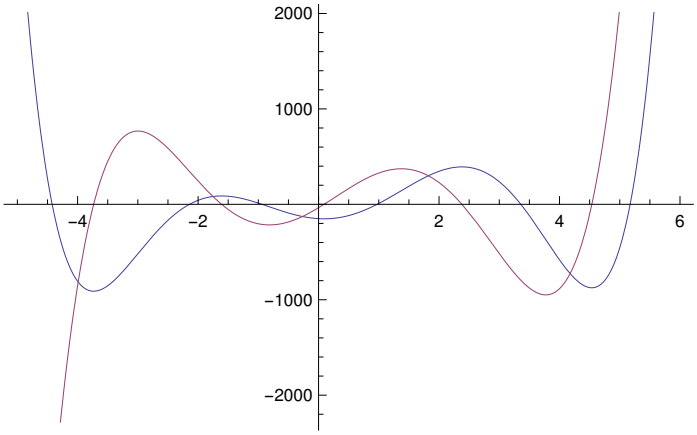
Und hier ein anspruchsvolleres Beispiel, dass wohl die wenigsten von Ihnen im Kopf hinbekommen würden:

```
D[Sqrt[x^3 Exp[4 x] Sin[x]], x]
```

$$\left(e^{4x}\,x^3\,\text{Cos}[x] + 3\,e^{4x}\,x^2\,\text{Sin}[x] + 4\,e^{4x}\,x^3\,\text{Sin}[x]\right) \Big/ \left(2\sqrt{e^{4x}\,x^3\,\text{Sin}[x]}\right)$$

Darstellung beider Funktionen, `f` und deren Ableitung `g`:

```
Plot[{f[x], g[x]}, {x, -5, 6}]
```



Integration

Stammfunktionen findet *Mathematica* strikt analytisch, wenn es eine solche gibt:

```
e[x_] = Integrate[f[x], x]
```

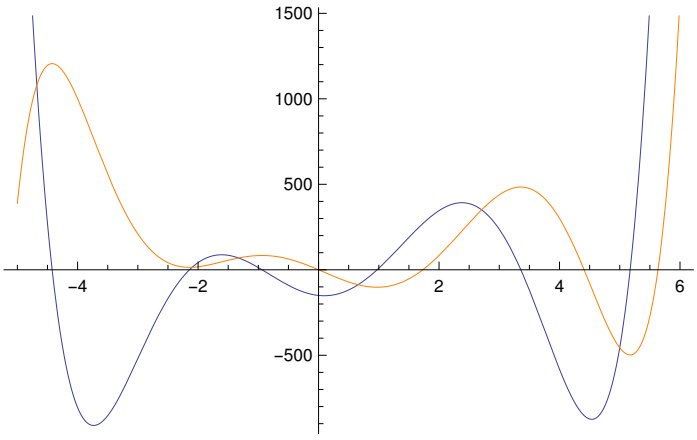
$$-150\,x-18\,x^2+\frac{190\,x^3}{3}+9\,x^4-6\,x^5-\frac{x^6}{3}+\frac{x^7}{7}$$

Und wieder ein anspruchsvolleres Beispiel, das man nicht so einfach aus dem Ärmel schüttelt:

```
Integrate[(2 x + 3) / (x^3 + x^2 - 2 x), x]
```

$$\frac{5}{3}\,\text{Log}[1-x]-\frac{3\,\text{Log}[x]}{2}-\frac{1}{6}\,\text{Log}[2+x]$$

```
Plot[{f[x], e[x]}, {x, -5, 6}, PlotStyle → {Automatic, Orange}]
```



Nochmals die Nullstellen, die jetzt für eine spätere Verwendung in einer Liste gespeichert werden:

```
n = NSolve[f[x] == 0, x]
```

```
{{x → -4.42228}, {x → -2.14285}, {x → -0.937347},
 {x → 0.972291}, {x → 3.35802}, {x → 5.17217}}
```

Zugriff auf das erste Listenelement

```
n[[1]]
```

```
{x → -4.42228}
```

Extraktion und Zuweisung der ersten und zweiten Nullstelle mithilfe des bekannten Ersetzungsoperators. (Die letzte Listenebene, welche als Element bloß die Transformationsregel enthält, wird automatisch entfernt.)

```
a = x /. n[[1]]
```

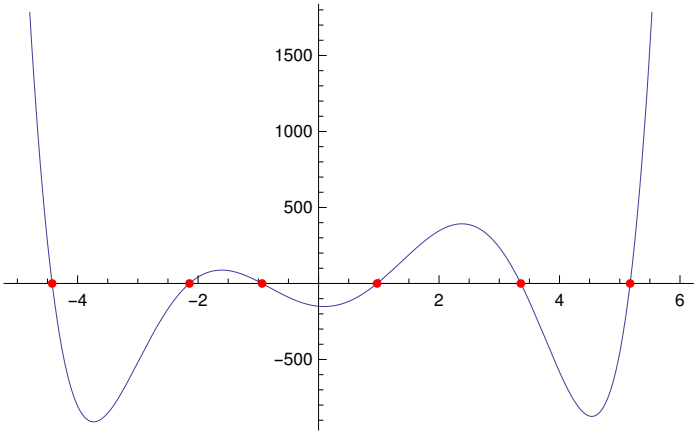
```
-4.42228
```

```
b = x /. n[[2]];
```

WIr sehen bei dem letzten Eingabefeld auch nach Ausführen keine Ausgabe, weil diese mit Hilfe des abschließenden Semikolons **;** unterdrückt wurde. Das ist vor allem praktisch, wenn man etwas definiert, was bei Auswertung einen unübersichtlich langen Ausdruck produziert, an dem man aber gar nicht interessiert ist.

Eine weitere Verwendung der Nullstellenliste: Der Ersetzungsoperator läßt sich "überladen", d.h. praktisch: Er wirkt nicht nur, wie oben genutzt, auf ein ausgewähltes, sondern auf alle Elemente einer Liste und gibt eine entsprechend lange Liste zurück, hier von (zweidimensionalen kartesischen) Koordinaten, die ihrerseits in Listenform **{x,y}** mit **y=0** notiert werden.

```
Plot[f[x], {x, -5, 6}, Epilog → {Red, PointSize[Medium], Point[{x, 0} /. n]}]
```



Epilog "ist" eine Liste von Zeichenbefehlen, die nach dem Plotten der Funktion(en) ausgeführt werden - und damit der Schlüssel zu vielen graphischen "Spielereien". So zeichnet **Point** z. B. einen Punkt, (optional) mit **PointSize** definierter Größe und angegebener Farbe.

Numerische Auswerung eines bestimmten Integrals

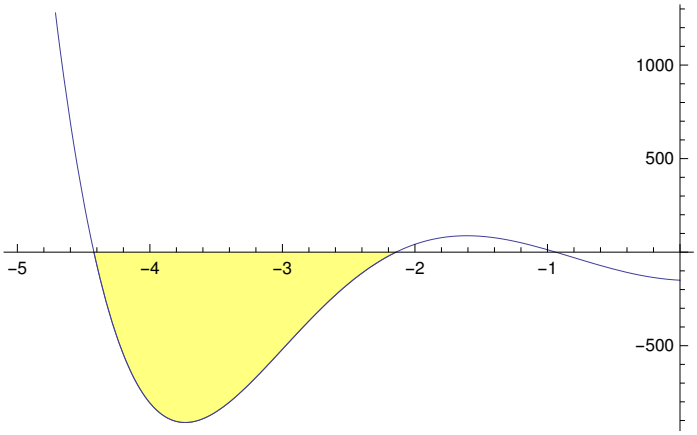
```
NIntegrate[f[x], {x, a, b}]
```

```
-1191.15
```

Alternative Möglichkeit zur Überlagerung von zwei Graphen - notwendig, um die Fläche unter der Abszisse nur im Integrationsintervall einzufärben. (Mit **Directive** werden Stilparameter zusammenge

faßt; `Opacity` bestimmt die Transparenz der Farbe.)

```
Show[Plot[f[x], {x, -5, 0}], Plot[f[ξ], {ξ, a, b},
  Filling → Axis, FillingStyle → Directive[Yellow, Opacity[0.5]]]]
```



Lineare Algebra

Vektoren und Matrizen

Die Behandlung von Vektoren und Matrizen erfolgt in *Mathematica* vollständig über Listen. Im Falle eines Vektors ist jede Komponente einfach ein Listenelement, ...

```
b = {9, -2, 7}
```

```
{9, -2, 7}
```

... im Falle einer Matrix ist jede Zeile, als Vektor notiert, ein Listenelement.

```
p = {{2, 1, 3}, {1, -2, 1}, {3, 2, 2}}
```

```
{{2, 1, 3}, {1, -2, 1}, {3, 2, 2}}
```

Die nützliche Funktion `MatrixForm` führt zu einer ansprechenderen Ausgabe:

```
MatrixForm[%]
```

$$\begin{pmatrix} 2 & 1 & 3 \\ 1 & -2 & 1 \\ 3 & 2 & 2 \end{pmatrix}$$

```
Clear[a, x, y, z]
```

```
c = {x, y, z};
```

Folgende Operationen kann man (beinahe) erraten ...

Konstante mal Vektor

```
a b
```

```
{9 a, -2 a, 7 a}
```

Für das Skalarprodukt gibt es zwei Eingabevarianten, ...

```
b.c
```

$$9\,x - 2\,y + 7\,z$$

```
Dot[b, c]
```

$$9\,x - 2\,y + 7\,z$$

... ebenso eine typographisch schönere für das Vektorprodukt aus dem Abschnitt Typesetting des Writing Assistents.

```
Cross[b, c]
```

$$\{-7\,y - 2\,z,\ 7\,x - 9\,z,\ 2\,x + 9\,y\}$$

```
b × c
```

$$\{-7\,y - 2\,z,\ 7\,x - 9\,z,\ 2\,x + 9\,y\}$$

(Bemerkung: **Cross** und **Dot** sind ebenso, wie eingangs bemerkten **Plus** und **Times**, Infix-Befehle - hier allerdings in sinnvollerer Anwendung.)

```
Det[p]
```

$$13$$

```
MatrixForm[Transpose[p]]
```

$$\begin{pmatrix} 2 & 1 & 3 \\ 1 & -2 & 2 \\ 3 & 1 & 2 \end{pmatrix}$$

Die Funktion **MatrixForm** kann auch als nachgestellte Formatierungsanweisung verwendet werden;

```
Inverse[p] // MatrixForm
```

$$\begin{pmatrix} -\frac{6}{13} & \frac{4}{13} & \frac{7}{13} \\ \frac{1}{13} & -\frac{5}{13} & \frac{1}{13} \\ \frac{8}{13} & -\frac{1}{13} & -\frac{5}{13} \end{pmatrix}$$

Das Produkt zweier Matrizen wird mit dem (überladenen) Punktoperator ausgeführt, der auch beim Skalarprodukt Verwendung findet.

```
MatrixForm[Inverse[p].p]
```

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Zugriffe auf einzelne Elemente

Ein allgemeiner Vektor mit einer beliebigen Dimensionalität läßt sich wie folgt erzeugen:

```
t = Array[x, 3]
```

```
{x[1], x[2], x[3]}
```

Der Zugriff auf eine Komponente erfolgt wie gehabt.

```
t[[2]]
```

```
x[2]
```

```
t[[2]] = 8;
```

```
t
```

```
{x[1], 8, x[3]}
```

An folgendem Beispiel wird der Unterschied zwischen dem Zuweisungsoperator und dem Ersetzungsoperator besonders deutlich:

```
t /. x[3] → 9
```

```
{x[1], 8, 9}
```

```
t
```

```
{x[1], 8, x[3]}
```

Oben für Vektoren gezeigtes gilt auch für Matrizen (und Objekte mit noch mehr Indizes, z.B. das ϵ-Symbol):

```
u = Array[d, {4, 4}]
```

```
{{d[1, 1], d[1, 2], d[1, 3], d[1, 4]}, {d[2, 1], d[2, 2], d[2, 3], d[2, 4]},
 {d[3, 1], d[3, 2], d[3, 3], d[3, 4]}, {d[4, 1], d[4, 2], d[4, 3], d[4, 4]}}
```

```
MatrixForm[u]
```

$$\begin{pmatrix} d[1,1] & d[1,2] & d[1,3] & d[1,4] \\ d[2,1] & d[2,2] & d[2,3] & d[2,4] \\ d[3,1] & d[3,2] & d[3,3] & d[3,4] \\ d[4,1] & d[4,2] & d[4,3] & d[4,4] \end{pmatrix}$$

Alternativ:

```
v = Table[d_i,j, {i, 4}, {j, 4}]
```

```
{{d1,1, d1,2, d1,3, d1,4}, {d2,1, d2,2, d2,3, d2,4},
 {d3,1, d3,2, d3,3, d3,4}, {d4,1, d4,2, d4,3, d4,4}}
```

```
MatrixForm[v]
```

$$\begin{pmatrix} d_{1,1} & d_{1,2} & d_{1,3} & d_{1,4} \\ d_{2,1} & d_{2,2} & d_{2,3} & d_{2,4} \\ d_{3,1} & d_{3,2} & d_{3,3} & d_{3,4} \\ d_{4,1} & d_{4,2} & d_{4,3} & d_{4,4} \end{pmatrix}$$

```
MatrixForm[Array[e, {3, 3, 3}]]
```

$$\begin{pmatrix} \begin{pmatrix} e[1,1,1] \\ e[1,1,2] \\ e[1,1,3] \end{pmatrix} & \begin{pmatrix} e[1,2,1] \\ e[1,2,2] \\ e[1,2,3] \end{pmatrix} & \begin{pmatrix} e[1,3,1] \\ e[1,3,2] \\ e[1,3,3] \end{pmatrix} \\ \begin{pmatrix} e[2,1,1] \\ e[2,1,2] \\ e[2,1,3] \end{pmatrix} & \begin{pmatrix} e[2,2,1] \\ e[2,2,2] \\ e[2,2,3] \end{pmatrix} & \begin{pmatrix} e[2,3,1] \\ e[2,3,2] \\ e[2,3,3] \end{pmatrix} \\ \begin{pmatrix} e[3,1,1] \\ e[3,1,2] \\ e[3,1,3] \end{pmatrix} & \begin{pmatrix} e[3,2,1] \\ e[3,2,2] \\ e[3,2,3] \end{pmatrix} & \begin{pmatrix} e[3,3,1] \\ e[3,3,2] \\ e[3,3,3] \end{pmatrix} \end{pmatrix}$$

Man kann die Matrizen automatisch mit den Werten von Funktionen befüllen lassen:

```
Table[h[i_, j_] = i + j, {i, 4}, {j, 4}] // MatrixForm
```

$$\begin{pmatrix} 2 & 3 & 4 & 5 \\ 3 & 4 & 5 & 6 \\ 4 & 5 & 6 & 7 \\ 5 & 6 & 7 & 8 \end{pmatrix}$$

Entnahme einer Zeile, also eines Elements der äußeren Liste

```
u[[3]]
```

```
{d[3, 1], d[3, 2], d[3, 3], d[3, 4]}
```

Entnahme einer Komponente des Zeilenvektors

```
%[[2]]
```

```
d[3, 2]
```

Praktisch: Der nun hinlänglich bekannte Listenzugriff kann auch direkt mit Zeilen - und Spaltenindex durchgeführt werden.

```
u[[3, 2]]
```

```
d[3, 2]
```

was eine kürzere Schreibweise für die äquivalente Langform ist:

```
u[[3]][[2]]
```

```
d[3, 2]
```

Und wie kommt man an eine Spalte? Indem man für jede Zeile ein Element rauspickt. Das sagt man *Mathematica* damit, dass man für den Zeilenbereich einfach `All` schreibt:

```
u[[All, 2]]
```

```
{d[1, 2], d[2, 2], d[3, 2], d[4, 2]}
```

Lösung von Gleichungssystemen

Automatisch ...

```
LinearSolve[p, b]
```

```
{-1, 2, 3}
```

... oder manuell, in diesem Sonderfall "direkt"

```
Inverse[p].b
```

```
{-1, 2, 3}
```