Universitäts-Rechenzentrum Trier

AWS.MATHE.1

Trier, den 23.6.2004

Bernhard Baltes-Götz

Einführung in Mathematica 5.0

Herausgeber: Universitäts-Rechenzentrum Trier
Universitätsring 15
D-54286 Trier
Tel.: (0651) 201-3417, Fax.: (0651) 3921
Leiter: Prof. Dr.-Ing. Manfred Paul

1 Einleitung

Dieses Manuskript gibt eine kurze Einführung in die Verwendung von Mathematica 5.0 an der Universität Trier. Es basiert im Wesentlichen auf dem mit „A Practical Introduction to Mathematica“ betitelten ersten Teil des Originalhandbuchs (Wolfram 2003), bietet also nur einen ersten und in vielerlei Hinsicht recht oberflächlichen Eindruck von Mathematica 5.0.
Im Manuskript wird primär die MS-Windows-Version 5.0 von Mathematica behandelt. Jedoch werden auch die weitgehend analog zu bedienenden 5.0-Versionen für UNIX und Mac-OS-X berücksichtigt. Da wir uns auf elementare Funktionen beschränken, gelten die meisten Aussagen auch für die Mathematica-Versionen 4.x und 3.x.

1.1 Was ist Mathematica?

Mathematica kann u.a. folgende Aufgaben erledigen:

- Numerische Berechnungen
  Es stehen mehrere tausend eingebaute Funktionen zur Verfügung, wobei der Benutzer jede beliebige Präzision wählen kann.
- Symbolische Berechnungen
  Mathematica beherrscht u.a. die Differential- und Integralrechnung, kann also z.B. Stammfunktionen symbolisch angeben.
- Graphische Darstellungen
  Mathematica stellt in zwei- oder dreidimensionalen Grafiken mathematische Funktionen oder Datenlisten dar.

Alle Anweisungen an Mathematica werden in einer einheitlichen Syntax formuliert. Das folgende Fenster zeigt Beispiele zu den drei Aufgabenarten:

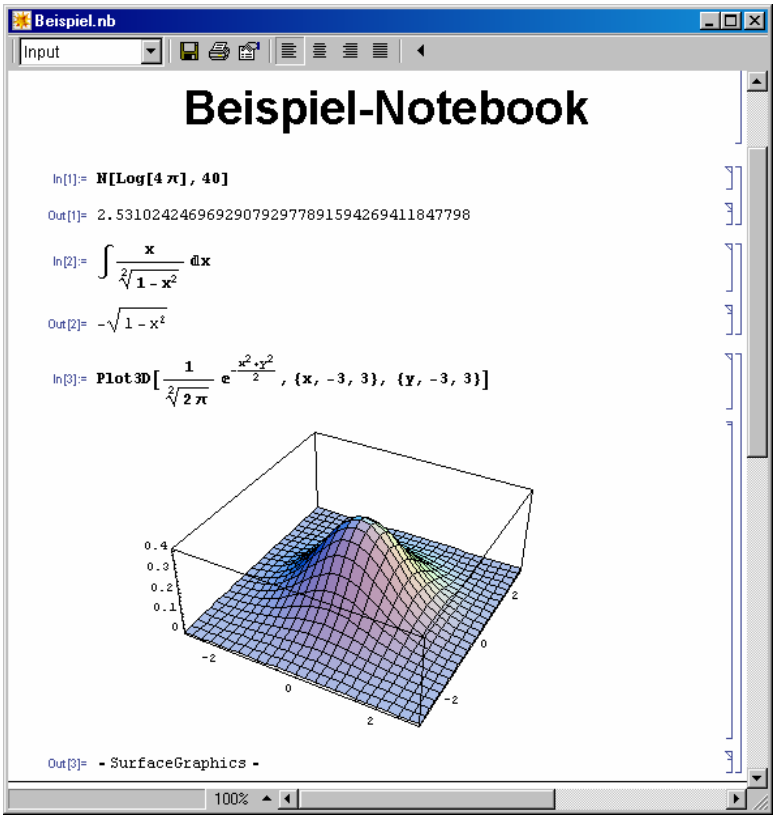


Abbildung 1 Notebook-Fenster unter MS-Windows

Wie dieses Beispiel mit Benutzereingaben *In[n]* und Programmreaktionen *Out[n]* zeigt, kann man mit Mathematica gut interagieren, wenn man seine Sprache beherrscht. In der ersten Eingabe wird z.B. die Berechnung von log(4 π) mit 40-stelliger Genauigkeit angefordert.

1.2 Die Bestandteile von Mathematica

Die beiden wichtigsten Bestandteile von Mathematica sind:

- **Benutzeroberfläche** (engl.: **Frontend**)
  Das Frontend nimmt die Anweisungen des Benutzers entgegen, übergibt die auszuwertenden Mathematica-Ausdrücke (Input) an den Kernel, und bereitet dessen Output für den Benutzer auf. Zwar ist die Benutzerschnittstelle an das jeweilige Betriebssystem (z.B. MS-Windows, UNIX/Linux, Mac-OS) angepasst, doch sind Optik und Bedienungslogik auf allen Plattformen sehr ähnlich.
- **Kern** (engl.: **Kernel**)
  Er ist plattform**un**abhängig und führt im Hintergrund alle mathematischen Operationen.

Benutzeroberfläche und Kern sind separate Prozesse, die sogar auf verschiedenen Computern ablaufen können. Beide kommunizieren über die Programmschnittstelle **Mathlink** miteinander, die auch dazu benutzt werden kann, von Mathematica aus Mathlink-kompatible andere Programme zu steuern, bzw. Mathematica von solchen Programmen aus zu verwenden.
Der Kernprozeß wird normalerweise erst bei Anforderung der ersten Berechnung gestartet.

Eine wichtige Methode zur individuellen Erweiterung des Mathematica-Systems stellen die **Pakete** (engl.: packages) dar. Es handelt sich um Dateien mit zusätzlichen oder modifizierten Funktionsdefinitionen in der Mathematica-Sprache.
Die so genannten **Standard Packages** werden als Bestandteil des Mathematica-Systems gleich mitgeliefert. So findet sich etwa die Cholesky-Zerlegung im Zusatzpaket **LinearAlgebra**.

1.3 Mathematica an der Universität Trier

Öffentlich zugängliche Mathematica-Installationen sind an der Universität Trier für MS-Windows und UNIX vorhanden.

1.3.1 Mathematica unter Windows

Mathematica für Windows steht Benutzern der NT-Domäne URT auf Pool- und Büro-PCs zur Verfügung.

Mathematica auf Pool-PCs unter Windows NT
In den vom Rechenzentrum betreuten PC-Pools unter Windows wird Mathematica folgendermaßen gestartet:

Start > Programme > Wissenschaftliche Programme > Mathematica

Wer Mathematica häufiger benutzt, wird sich vielleicht zum bequemeren Starten ein Symbol auf den individuell konfigurierbaren Desktop legen.

Mathematica auf vernetzten Büro-PCs unter Windows NT/2000/XP
Sie können Mathematica auf Ihrem Büro-PC unter Windows NT/2000/XP verwenden und dabei ohne eigenen Kostenbeitrag von der Mehrfachlizenz des Rechenzentrums profitieren. Nähere Informationen erhalten Sie in der Benutzerberatung.

1.3.2 Mathematica unter UNIX/Linux

Auf jedem Rechner mit X-Server-Funktionalität kann die auf dem UNIX-Server RZSUN01 installierte X-Version von Mathematica genutzt werden. Wer Mathematica lokal auf einem UNIX/Linux – Rechner installieren und den URT-Lizenzserver nutzen möchte, kann eine entsprechende CD in der Benutzerberatung anfordern.

2 Die Mathematica-Benutzeroberfläche: Notizbücher und Zellen

Die mit der Standard-Oberfläche[1] von Mathematica erstellten Dokumente werden als **Notizbücher** (engl.: **Notebooks**) bezeichnet. Wesentliche Bestandteile eines Notizbuchs sind:

- Mathematica-Input und –Output
- Grafiken
- erklärender Text und Überschriften

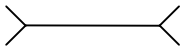
Ein Notizbuch-Beispiel haben Sie schon in Abbildung 1 gesehen. Analog zu den Dokumenten eines Textverarbeitungsprogramms kann man Mathematica-Notizbücher neu erstellen, öffnen, schließen, editieren und drucken. Für Dateien mit Mathematica-Notizbüchern wird in der Regel die Namenserweiterung „.nb“ verwendet.
Die Notizbuchinhalte sind in **Zellen** organisiert, deren Erstreckung am rechten Rand des Notizbuchfensters durch Klammern angezeigt wird (siehe Abbildung 1). Es ist auch eine *hierarchische* Zellenstruktur möglich, die durch Mehrebenen-Anordnung der Zellenklammern angezeigt wird.

Ein Notizbuch kann z.B. die Beschreibung einer mathematischen Theorie mit erläuternden Grafiken und ausführbaren Ausdrücken enthalten, so dass die Leser(innen) neu erworbenes Wissen sofort ausprobieren können. Mathematica-Notizbücher sind folglich gut für Ausbildungszwecke geeignet, und auch ein E-Learning per World Wide Web wird gut unterstützt.

2.1 Elementare Operationen mit Notizbuch-Zellen

2.1.1 Mathematica-Ausdrücke schreiben und auswerten lassen (Input- und Output-Zellen)

Am Anfang eines neuen Notizbuch-Fensters befindet sich eine waagerechte Linie als Zelleinfügemarke. Sobald Sie Text eintragen, wird für selbigen eine **Input**-Zelle eingerichtet, und die Einfügemarke verschwindet.
Eine Zelle darf mehrere Zeilen umfassen, die jeweils mehrere Ausdrücke enthalten dürfen.
In einem bereits bearbeiteten Notizbuch können Sie die Zelleinfügemarke per Maus oder mit den vertikalen Pfeiltasten an eine zulässige Stelle setzen. In Abbildung 1 sehen Sie die Zelleinfügemarke am unteren Fensterrand. Über zulässigen Einfügestellen sieht der Mauszeiger folgendermaßen aus:



Wenn Sie nach dem Setzen der Zelleinfügemarke Text eintippen, verschwindet die Einfügemarke und an ihrer Stelle wird eine neue **Input**-Zelle eingerichtet.
Die Klammer am rechten Rand einer Input-Zelle sieht folgendermaßen aus:

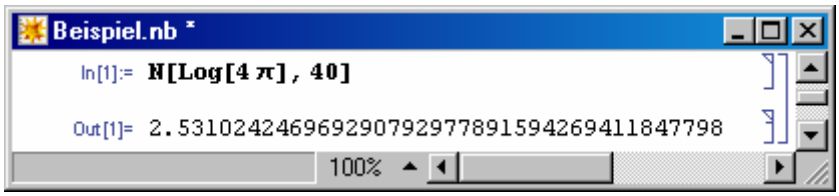
[1] Die auf modernen Computersystemen in der Regel weniger attraktive *textorientierte* Schnittstelle von Mathematica wird in diesem Manuskript *nicht* verwendet.

Um den Ausdruck in einer Zelle als Mathematica-Input auswerten zu lassen, muss diese markiert sein (s.u.), oder die Schreibmarke (|) muss sich darin befinden. Die Auswertung wird dann angefordert mit der Tastenkombination:

<Shift> + <Return>

Bei Tastaturen mit Ziffernblock hat die dortige **Enter**-Taste (*ohne* **Shift**) den selben Effekt.
Die auszuwertenden Input-Zellen erhalten eine Beschriftung mit fortlaufender Nummer (siehe oben).
Gelingt die Auswertung einer Zelle, so produziert Mathematica eine (ebenfalls beschriftete und numerierte) **Output**-Zelle, die unmittelbar hinter der ausgewerteten Input-Zelle erscheint.
Die Klammer einer Output-Zelle unterscheidet sich von der Input-Klammer durch einen zusätzlichen Querstrich:

Per Voreinstellung bilden die zu einer Anforderung gehörenden Zellen eine *Gruppe*, was durch eine gemeinsame Klammer (ohne Verzierungen) angezeigt wird, z.B.:



Mathematica nummeriert alle Zellen nach der Auswertungsreihenfolge, so dass die Nummerierung nicht mit der räumlichen Anordnung korrespondieren muss.
Über den Menübefehl **Kernel > Show In/Out Names** lassen sich die Zellennummern ab- bzw. anschalten.

Während der Kernel-Aktivität können Sie mit dem Frontend weiterarbeiten. Die gerade in Bearbeitung befindlichen Input-Zellen sind an folgender Doppelklammer zu erkennen:

Bei der ersten Auswertungsanforderung in einer Mathematica-Sitzung wird der Kernel geladen, so dass eine längere Wartezeit entsteht.

Auf dem Macintosh erhalten Sie die beim Arbeiten mit Mathematica unverzichtbaren Klammern folgendermaßen:

[	Alt (Wahl) + <5>
]	Alt (Wahl) + <6>
|	Alt (Wahl) + <7>
{	Alt (Wahl) + <8>
}	Alt (Wahl) + <9>

2.1.2 Zellen für die weitere Bearbeitung markieren

Zum Markieren von Zellen für eine anschließende Operation (z.B. Auswertung anfordern, Löschen), bietet Mathematica folgende Möglichkeiten:

- Eine einzelne Zelle oder Zellengruppe wird per Mausklick auf ihre Klammer markiert. Über markierbaren Klammern hat der Mauszeiger folgende Form:

- Zur Markierung einer *Serie* klickt man auf das erste Element und zieht die Maus dann bei gedrückter linker Maustaste bis zum letzten Element.
- Mit dem Menübefehl **Edit > Select All** oder der Tastenkombination **<Strg>+<A>** können Sie *alle* Zellen eines Notizbuchs markieren.

2.1.3 Zellinhalte verschieben, kopieren oder löschen

Markieren Sie alle Zellen bzw. Zellengruppen und benutzen Sie die Befehle des Betriebssystems zum Ausschneiden, Kopieren und Einfügen. Die Zelleinfügemarke (s.o.) können Sie per Mausklick oder mit den vertikalen Pfeiltasten an die gewünschte (und zulässige) Stelle im Notizbuch setzen.

Um die ab Einfügemarke in Aufwärtsrichtung nächstgelegene Input-Zelle unter die Einfügemarke zu kopieren, können Sie den Menübefehl

Input > Copy Input From Above

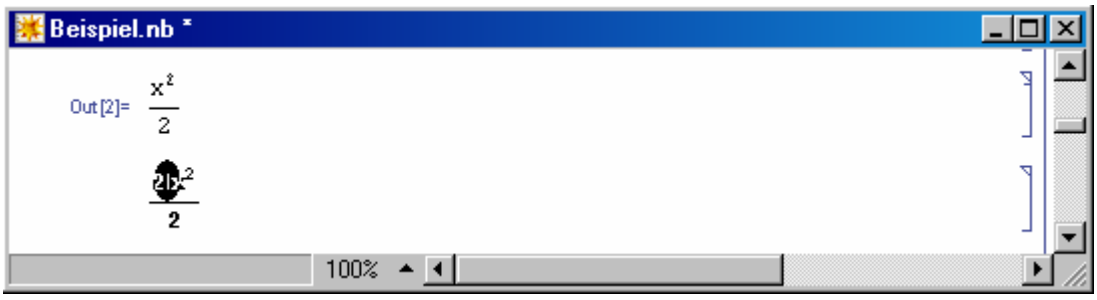
bzw. die folgende Tastenkombination benutzen:

MS-Windows	Macintosh	X-Window
<Strg><L>	**<Befehl><L>**	**<Strg><L>**

Um die ab Einfügemarke in Aufwärtsrichtung nächstgelegene Output-Zelle unter die Einfügemarke zu kopieren und dann zu einer neuen Eingabe zu verarbeiten, können Sie den Menübefehl **Input > Copy Output From Above** bzw. die folgende Tastenkombination benutzen:

MS-Windows	Macintosh	X-Window
<Shift><Strg><L>	**<Shift><Befehl><L>**	**<Shift><Strg><L>**

Alternativ kann man Mathematica durch Editieren einer Ausgabezelle dazu bringen, mit dem veränderten Inhalt eine neue Input-Zelle zu eröffnen und die Output-Zelle in ihren vorherigen Zustand zurück zu versetzen, wobei auf diese Heinzelmännchen-Aktion durch einen kurzen „Kugelblitz“ hingewiesen wird, z.B.:

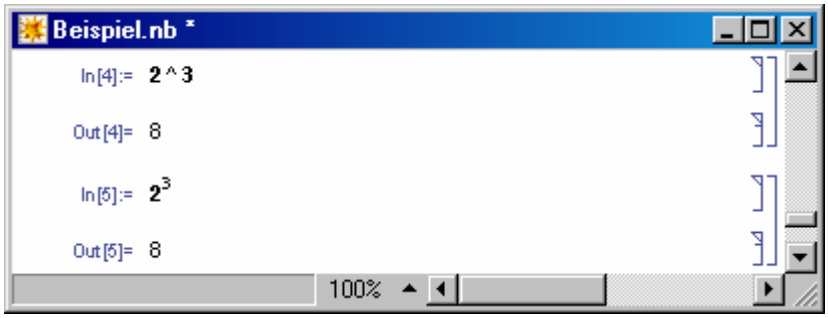


2.2 Eingabe in mathematischer Notation, Arbeiten mit Paletten

Beim Verfassen von Ausdrücken in Input-Zellen haben Sie folgende Alternativen:

- Verwendung der traditionellen Mathematica-Syntax (InputForm), die mit den Zeichen einer Standardtastatur auskommt.
- Verwendung der in mathematischen Publikationen üblichen „zweidimensionale“ Notation mit zahlreichen Sonderzeichen.

Im folgenden Beispiel ist der Ausdruck 2^3 in InputForm (mit dem Zeichen ^ für die Potenz-Operation) sowie in mathematischer Notation zu sehen:



Für die klassische Mathematica-Syntax spricht:

- Universelle Verwendbarkeit in *allen* (auch in den textorientierten) Mathematica-Frontends
- Rationelle Eingabe per Tastatur

Als Nachteile sind zu nennen:

- Die traditionelle Syntax muss erst erlernt werden.
- Sie ist optisch weniger attraktiv als die übliche mathematische Notation.

Mathematica unterstützt seit der Version 3.0 die mathematische Notation nicht nur in Output–, sondern auch in Input–Zellen. Wir werden uns in diesem Abschnitt mit den Eingabehilfen beschäftigen, die zum Erstellen der zwar intuitiven, aber leider auch technisch aufwändigen mathematischen Notation verfügbar sind.
In späteren Abschnitten wird meist die klassische Mathematica-Syntax bevorzugt, die in allen Frontends unterstützt und auch in der Mathematica-Dokumentation (z.B. Wolfram 2003) bevorzugt wird.

2.2.1 Eingabe in mathematischer Notation mit Hilfe von Paletten

Eine wesentliche Rolle spielt die Palette **BasicInput**, die nötigenfalls über den Menübefehl **File > Palettes > 3 BasicInput** aktiviert werden kann:

Analog zur Arbeitsweise bei den Formeleditoren von modernen Textverarbeitungsprogrammen lässt sich mit Hilfe dieser Palette eine Formel über intuitive Aktionen aufbauen. Zur Demonstration soll zu der folgenden, in traditioneller Syntax verfassten, Input-Zelle zur Berechnung des bestimmten Integrals der Funktion e^x x in den Grenzen von 0 bis *a* eine äquivalente Formulierung in zweidimensionaler Notation erstellt werden:

In[1]:= **`Integrate[Exp[x]x,{x,0,a}]`**

Setzen Sie die Zelleinfügemarke an eine zulässige Stelle, und gehen Sie dann folgendermaßen vor:

- Klicken Sie in der Palette auf das Integral mit Platzhaltern für die Integrationsgrenzen  , welches daraufhin in der aktuellen Input-Zelle erscheint.
- Ersetzen Sie den (durch einen kleinen Punkt markierten) Platzhalter für die untere Integrationsgrenze durch eine Null.
- Steuern Sie den Platzhalter für die obere Integrationsgrenze per Mausklick oder einmaliges Betätigen der Tabulator-Taste an, und ersetzen Sie ihn durch das Symbol *a*.
- Steuern Sie den Platzhalter für den Integranden per Mausklick oder Tabulator-Taste an, und ersetzen Sie ihn per Paletten-Schalter  durch den Platzhalter für einen Potenz-Ausdruck.
- Ersetzen Sie die aktuell markierte Basis per Paletten-Schalter  durch das Mathematica-Symbol für die Eulersche Zahl.
- Steuern Sie den Platzhalter für den Exponenten per Mausklick oder Tabulator-Taste an, und ersetzen Sie ihn durch das Variablen-Symbol *x*.
- Erweitern Sie den Integranden noch um den Faktor *x*. Die richtige Position der Einfügemarke lässt sich per Maus oder über die horizontalen Pfeiltasten einstellen.
- Steuern Sie den Platzhalter für die Integrationsvariable per Mausklick oder Tabulator-Taste an, und fügen Sie dort die Variable *x* ein.

Ihre Input-Zelle sollte nun folgendermaßen aussehen und kann ausgeführt werden:

$$\int_0^a e^x \, x \, dx$$

Als Ergebnis erhalten wir:

$$1 + (-1 + a)\, e^a$$

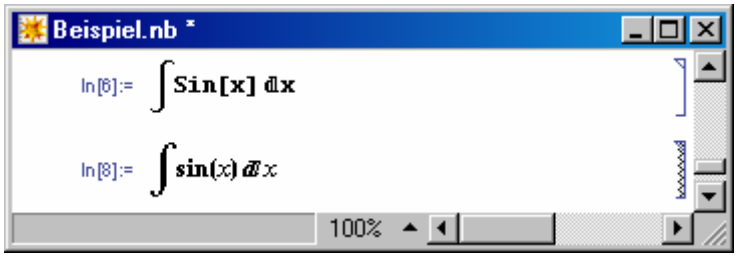
Hinweis: Die Arbeit mit den teilweise recht kleinen Platzhaltersymbolen kann durch eine Vergrößerung der Notebook-Anzeige über **Format > Magnification** erleichtert werden.

Über den Menübefehl **File > Palettes** stehen noch weitere Paletten zur Verfügung, die u.a. über 700 mathematische Sonderzeichen enthalten. Außerdem kann man über den Menübefehl **Input > Create Table/Matrix/Palette** beliebige eigene Paletten zusammenstellen.

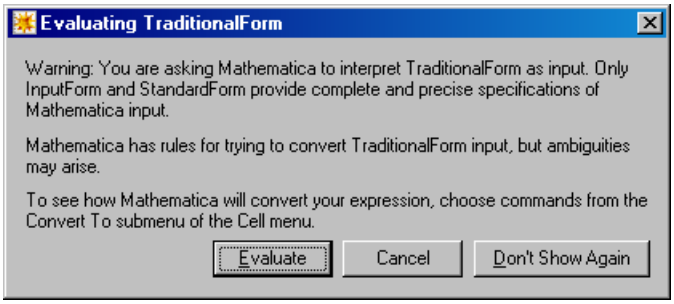
2.2.2 Zwei Varianten der mathematischen Notation: Standardform und traditionelle Form

Wir haben bisher die **Standardform** der mathematischen Notation kennen gelernt, die durch absolut eindeutige Bezeichnungen gekennzeichnet ist. Daneben unterstützt Mathematica noch die so genannte **traditionelle Form**, die zugunsten einer weitestgehenden Angleichung an übliche mathematische Publikationen einige Mehrdeutigkeiten in Kauf nimmt. Hier werden z.B. Funktionsnamen klein geschrieben, während ansonsten in Mathematica die Namen der vordefinierten Funktionen alle mit einem Großbuchstaben beginnen, um Verwechslungen mit Variablennamen auszuschließen.

Eine Input-Zelle in mathematischer Standardnotation (oder auch in der eindimensionalen InputForm) kann über den Menübefehl **Cell > ConvertTo > TraditionalForm** in die traditionelle Form gebracht werden, z.B.:



Beim Versuch, den unteren Ausdruck (in traditioneller Form) auszuwerten, warnt Mathematica:



Im konkreten Beispiel gelingt die korrekte Auswertung trotz Kleinschreibung im Funktionsnamen und runder Argument-Klammern. Generell sollte die traditionelle Form jedoch nur dazu verwendet werden, Output für den Export in andere Dokumente aufzupolieren.

Insgesamt kennt Mathematica vier Formate, die Input- oder Output-Zellen mit dem Menübefehl **Cell > Convert To** zugewiesen werden können:

InputForm	Klassische Mathematica-Syntax, kann mit jeder Standardtastatur eindeutig und bequem eingegeben werden, z.B. (Drittel der Eulerschen Zahl): `E/3`
OutputForm	Dieses Format nutzt einige zweidimensionale Darstellungselemente (z.B. bei Brüchen), beschränkt sich aber auf die Zeichen einer Standardtastatur, z.B.: $\frac{E}{3}$
StandardForm	Diese für Input und Output geeignete Form verwendet zweidimensionale Positionierung sowie Sonderzeichen und zeichnet sich durch perfekt-eindeutige Bezeichnungen aus, z.B.: $\frac{e}{3}$
TraditionalForm	Diese primär für Output-Zellen gedachte Form imitiert die übliche mathematische Typographie, wobei gelegentlich die Bezeichnungseindeutigkeit verloren geht, allerdings nicht im aktuellen Beispiel: $\frac{e}{3}$

Über die Menübefehle

Cell > Default Input FormatType

bzw.

Cell > Default Output FormatType

können die Mathematica-Voreinstellungen für neue Zellen festgelegt werden.

2.2.3 Eingabe in mathematischer Notation mit Hilfe der Tastatur

Bei der Erstellung von Input-Zellen in mathematischer Notation sind Sie nicht auf Paletten und Maus angewiesen, sondern können alle Eingaben auch per Tastatur bewerkstelligen. Ähnlich wie bei der Input-Form (s.o.) ist der Lohn für den Lernaufwand eine höhere Eingabegeschwindigkeit.
Gehen Sie folgendermaßen vor, um unser Integral-Beispiel zu erstellen:

- Drücken Sie die **Esc**-Taste, Tippen Sie *int* und drücken Sie erneut die **Esc**-Taste. Es erscheint ein Integralzeichen.
- Drücken Sie die Tastenkombination **<Strg>+<+>**, und geben Sie die Null als untere Integrationsgrenze ein.
- Drücken Sie die Tastenkombination **<Strg>+<%>**, und geben Sie das Variablensymbol *a* als obere Integrationsgrenze ein.
- Drücken Sie die Tastenkombination **<Strg>+<Leertaste>**, um auf die Eingabeebene für den Integranden zu gelangen.
- Drücken Sie die **Esc**-Taste, Tippen Sie *ee*, und drücken Sie erneut die **Esc**-Taste. Es erscheint das Sonderzeichen für die Eulersche Zahl.
- Drücken Sie die Tastenkombination **<Strg>+<^>**, so dass ein Platzhalter für den Exponenten zur Eulerschen Zahl erscheint. Tragen Sie die Variable *x* als Exponenten ein, bewegen Sie die Schreibmarke per Pfeiltaste nach rechts, und notieren Sie die Variable *x* auch noch als Faktor im Integranden.
- Drücken Sie die **Esc**-Taste, Tippen Sie *dd*, und drücken Sie erneut die **Esc**-Taste. Es erscheint das Sonderzeichen für den Differentialoperator. Ergänzen Sie noch die Integrationsvariable *x*.

Nun sollte folgende Input-Zelle entstanden sein:

$$\int_0^a e^x \, x \, dx$$

Die paletten- und die tastaturgestützte Eingabe können natürlich auch kombiniert werden.

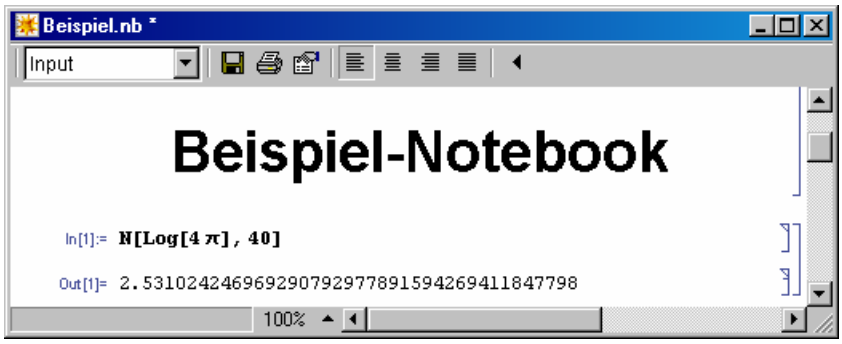
2.3 Notizbücher verwalten und drucken

In Mathematica stehen die üblichen Kommandos zum Verwalten von Dokumenten zur Verfügung:

		Tastenkombination		
Notizbuch ...	Menübefehl	MS-Windows **<Strg> +**	Macintosh **<Befehl> +**	X-Window **<Strg> +**
Erstellen	**File > New**	**<N>**	**<N>**	**<N>**
Sichern	**File > Save**	**<S>**	**<S>**	**<S>**
Sichern als	**File > Save as ....**	**<Shift> <S**	**<Shift> <S>**	**<Shift> <S>**
Öffnen	**File > Open**	**<O>**	**<O>**	**<O>**
Schließen	**File > Close**	**<F4>**	**<W>**	**<W>**

Die voreingestellte Namenserweiterung für Notizbuch-Dateien lautet: "**.nb**".

Wer das Sichern per Mausklick erledigen möchte, muss mit **Format > Show ToolBar** die Symbolleiste zum Notebook-Fenster einschalten, z.B.:



Das Öffnen der zuletzt bearbeiteten Notizbücher geht am schnellsten per **File**-Menü.

Um das gesamte Notizbuch oder aber die momentane Auswahl zu drucken, können Sie den Menübefehl **File > Print ...** oder eine von den folgenden Tastenkombinationen verwenden:

Drucken	MS-Windows	Macintosh	X-Window
Alles	**<Strg><P>**	**<Befehl><P>**	**<Strg><P>**
Auswahl	**<Shift><Strg><P>**	**<Shift><Befehl><P>**	**<Shift><Strg><P>**

Über den Menübefehl **File > Printing Settings** sind zahlreiche Optionen zur Gestaltung der Ausgabe verfügbar (z.B. Ränder, Kopf- und Fußzeilen). Bei der UNIX/Linux-Version sind die Menübefehle **File > Printing Options** sowie **File > Headers and Footers** zuständig.

3 Arbeiten mit Mathematica

3.1 Der Help-Browser

Bei der Arbeit mit Mathematica sollten Sie unbedingt den mit **Help > Help Browser** oder **<F1>** erreichbaren **Help Browser** nutzen, der u.a. über die Option **Buid-in Functions** umfassende Informationen zum Kernel bereithält und über die Option **Getting Started** bzw. **Tour** einen guten Einblick in das Mathematica-System bietet:

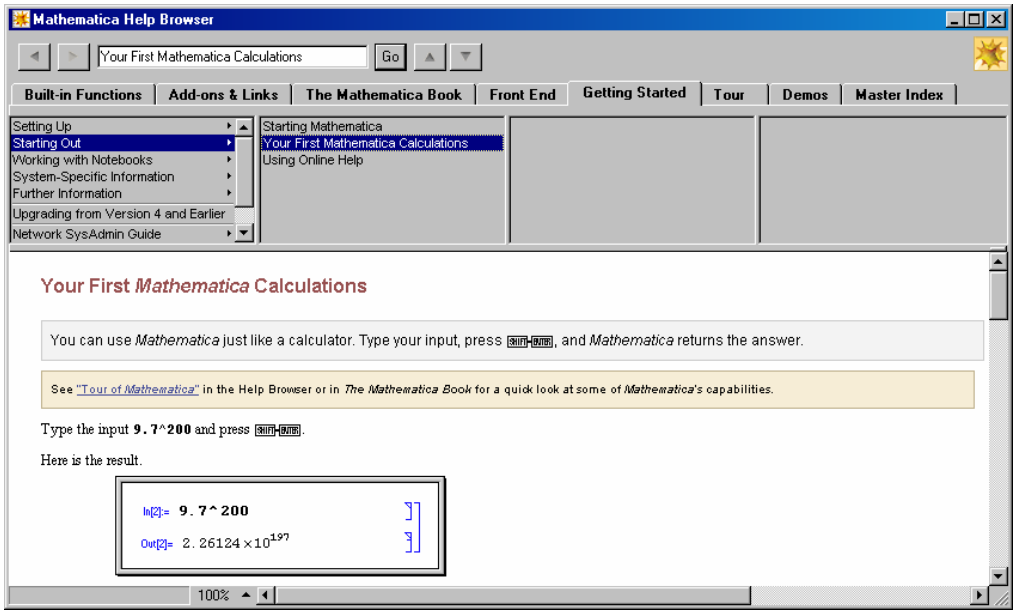


Abbildung 2 Der Help Browser bietet u.a. eine Tour durch die Welt von Mathematica

Der Help Browser bietet außerdem …

- eine elektronische Version des Mathematica-Buchs (Wolfram 2003) über die gleichnamige Registerkarte.
- über **Add-ons & Links > Standard Packages** die Dokumentation der Standard-Erweiterungspakete.
- über **Getting Started** eine Beschreibung der Plattformspezifika unter MS-Windows, Mac-OS und X/UNIX.

3.2 Mathematica als Taschenrechner

In diesem Abschnitt wird beschrieben, wie Sie Mathematica als exzellenten **Taschenrechner** benutzen können. Anspruchsvollere numerische Aufgaben (z.B. Berechnung bestimmter Integrale) werden später behandelt.

3.2.1 Arithmetische Operationen

Es stehen die üblichen arithmetischen Operationen zur Verfügung:

Operation	Eingabe
Addition	`x + y`
Minus	`-x`
Subtraktion	`x - y`
Multiplikation	`x y` oder `x * y`
Division	`x / y`
Potenzieren	`x ^ y`

In der Tabelle wird die InputForm der Operatorzeichen benutzt (vgl. Abschnitt 2.2.2). Dies gilt für die meisten Tabellen und Beispiele in diesem Manuskript.

Beispiel:

In[1]:= **`(3+4)^2-2(3+1)`**

Out[1]= `41`

Die Auswertungsreihenfolge entspricht den üblichen Konventionen und kann nötigenfalls mit **runden** Klammern gesteuert werden.

Bei der **Schreibweise** ist zu beachten:

- Bei der *Multiplikation* darf die übliche mathematische Schreibweise ohne Operatorzeichen verwendet werden. Im Allgemeinen **muss** dabei ein Leerzeichen zwischen den Faktoren stehen. Z.B. wird "hi" von Mathematica *nicht* als Produkt aufgefasst, sondern als neues Symbol. "3i" wird jedoch richtig verstanden. Es darf auch ein Stern zwischen die Faktoren gesetzt werden (z.B. h*i).
- Als Dezimaltrennzeichen ist der **Punkt** zu verwenden.
- Ein Beispiel für die Exponentialschreibweise:

In[1]:= **`2.3 10^-70`**

Out[1]= 2.3×10^{-70}

3.2.2 Exakte und angenäherte Ergebnisse

Während Taschenrechner mit begrenzter Genauigkeit arbeiten, liefert Mathematica nach Möglichkeit exakte Ergebnisse, z.B.:

In[1]:= **`2^150`**

Out[1]= `1427247692705959881058285969449495136382746624`

Wenn Sie ein *Näherungsergebnis* wollen, müssen Sie die Funktion **N** (für: "**Numeric**") auf den auszuwertenden Ausdruck anwenden:

In[1]:= **`N[2^150]`**

Out[1]= 1.42725×10^{45}

Dabei kann optional die **Genauigkeit** festgelegt werden, z.B.:

In[1]:= **`N[452/62,10]`**

Out[1]= `7.290322581`

Für die Funktion **N** gibt es noch eine alternative Postfix-Schreibweise:

In[1]:= **`2^150//N`**

Wenn Sie eine ganze Zahl ohne Dezimaltrennzeichen eingeben, hält Mathematica diese für exakt.
Enthält ein Ausdruck ausschließlich exakte Werte, dann beschränkt sich Mathematica bei der Auswertung auf exakte Operationen, z.B.:

```
In[1]:= 452/62
Out[1]= 226/31
```

Wenn Sie eine Zahl mit Dezimalpunkt eingeben, hält Mathematica diese für gerundet.
Enthält in einem Ausdruck irgendeine Zahl einen Dezimalpunkt, wird das Ergebnis des Ausdrucks numerisch ermittelt, z.B.:

```
In[1]:= 452.0/62
Out[1]= 7.29032
```

3.2.3 Mathematische Funktionen

In Mathematica-Ausdrücken können fest eingebaute und per Paket ergänzte mathematische Funktionen verwendet werden, z.B.:

```
In[1]:= Log[1]
Out[1]= 0
```

Bei der Suche nach einer bestimmten Funktion werden Sie vom **Help Browser** unterstützt, der mit dem Menübefehl **Help > Help Browser** oder mit der Taste **<F1>** gestartet werden kann, z.B.:

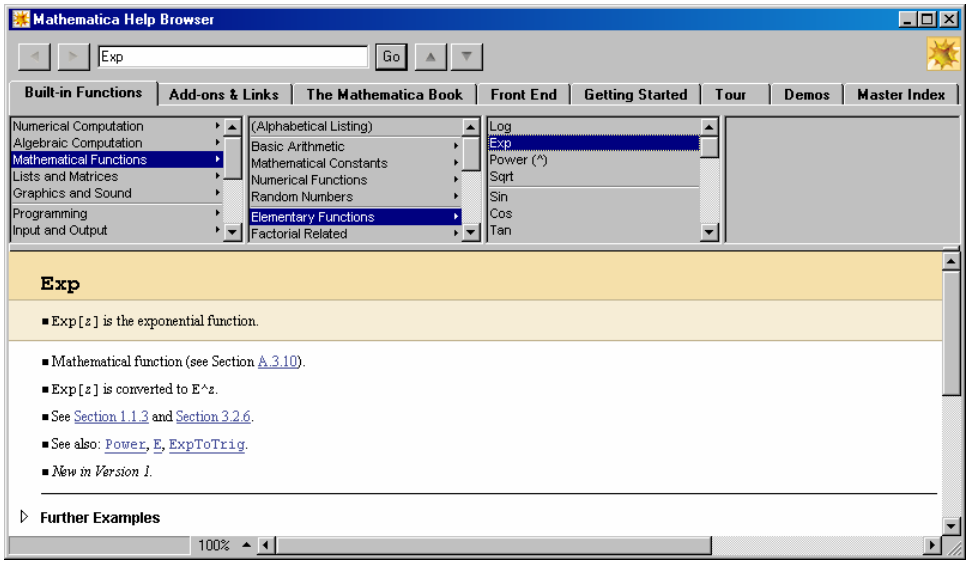


Abbildung 3 Der Help Browser informiert umfasend über mathematische Funktionen

Bei der **Schreibweise von Funktionsaufrufen** ist zu beachten:

- Die Argumente werden in **eckige** Klammern gesetzt.
- Abkürzungen sind verboten, allerdings werden wir eine Möglichkeit kennen lernen, einen Ausdruck automatisch vervollständigen zu lassen.
- **Groß/Klein-Schreibung** ist signifikant!

- Die Namen der **eingebauten Mathematica-Funktionen** beginnen mit einem **Großbuchstaben**. Wenn Sie versehentlich den zugehörigen Kleinbuchstaben verwenden, stößt Mathematica auf einen unbekannten Bezeichner und wendet folglich *nicht* die gemeinte Funktionsvorschrift auf das übergebene Argument an. In diesem Fall macht es aber schlau auf den mutmaßlichen Fehler aufmerksam, z.B.:

```
In[1]:= log[1]
        General::spell1: Possible spelling error:
        new symbol name "log" is similar to existing symbol "Log".
Out[1]= log[1]
```

Wie bei den arithmetischen Operationen (vgl. Abschnitt 3.2.1) beschränkt sich Mathematica auch bei der Auswertung von Funktionen mit exakten Argumenten (z.B. rationalen Ausdrücken) auf exakte Operationen, z.B.:

```
In[1]:= Log[1/3]
Out[1]= -Log[3]
```

Sie können eine Näherungslösung erzwingen, indem Sie die Funktion **N** (für: "**Numeric**") auf den Ausdruck anwenden:

```
In[1]:= N[Log[1/3]]
Out[1]= -1.09861
```

3.2.4 Konstanten

Sie können in Mathematica-Ausdrücken einige **Konstanten** über *Namen* oder Sonderzeichen ansprechen, z.B.:

```
In[1]:= Pi^2//N
Out[1]= 9.8696
```

Einige oft benötigte Konstanten mit ihren Mathematica-Namen und -Symbolen:

Name	Symbol	Wert
Pi	π	3,1459...
E	ℯ	2,7182...
I	ⅈ	
Infinity	∞	∞

Über weitere Konstanten informiert der **Help Browser** unter **Built-in Functions > Mathematical Functions > Mathematical Constants**.

3.2.5 Komplexe Zahlen

Komplexe Zahlen schreibt Mathematica mit Hilfe der imaginären Einheit, die im Standardformat durch das Symbol ⅈ und im Input- bzw. Outputformat durch den Namen I (groß!) dargestellt wird, z.B.:

```
In[1]:= Sqrt[-4]
Out[1]= 2 ⅈ

In[2]:= Exp[2+9I]//N
Out[2]= -6.73239 + 3.04517 ⅈ
```

Mathematica kennt die üblichen Operationen mit komplexen Zahlen:

In[3]:= **Re[1+2I]**

Out[3]= 1

In[4]:= **Im[1+2I]**

Out[4]= 2

In[5]:= **Conjugate[1+2I]**

Out[5]= 1 - 2 i

In[6]:= **Abs[1+I]**

Out[6]= $\sqrt{2}$

In[7]:= **Arg[1+I]**

Out[7]= $\frac{\pi}{4}$

3.3 Wichtige Bestandteile und Regeln der Mathematica-Sprache

Natürlich ist Mathematica unvergleichlich mächtiger als ein Taschenrechner. Sie können seine Leistungen über die Mathematica-Sprache abrufen, über die Sie in diesem Abschnitt einige elementare Informationen erhalten.

3.3.1 Rückgriff auf vorangegangene Ausgaben

Beim Arbeiten mit Mathematica ist es oft bequem, in der aktuellen Input-Zelle eine frühere Ausgabe anzusprechen:

Symbol	Bedeutung	Beispiel
%	letzte Ausgabe	**N[%]**
%%	vorletzte Ausgabe	**Show[%%, ViewPoint -> {0,-2,0}]**
%*n*	Inhalt der Ausgabe Out[*n*]	**%2 + %3**

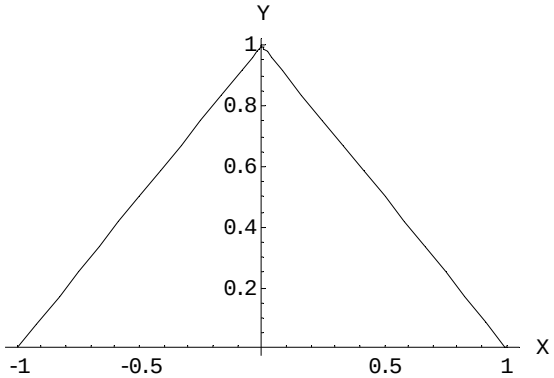
3.3.2 Definition von Variablen

Beim Aufbau komplizierter Ausdrücke ist es sinnvoll, Zwischenergebnisse in benannten Variablen zu speichern. Diese können nicht nur Zahlen aufnehmen, sondern auch Ausdrücke, z.B.:

In[1]:= **rd = 1-Sign[x]x**

Out[1]= 1 - x Sign[x]

In[2]:= **Plot[rd,{x,-1,1}, AxesLabel -> {X, Y}]**



Out[2]= `-Graphics-`

Achtung: Variablendefinitionen bleiben bis zum Widerruf wirksam. Aus der Nichtbeachtung dieser Regel resultieren die meisten Fehler beim Arbeiten mit Mathematica. Nicht mehr benötigte Definitionen sollten daher zur Vermeidung unerwünschter Effekte folgendermaßen aufgehoben werden:

In[3]:=
```
rd = .
```

Regeln für Variablennamen:

- Das erste Zeichen muss ein Buchstabe sein.
- **Groß/Klein-Schreibung** ist signifikant, „**b**“ und „**B**“ bezeichnen also zwei verschiedene Variablen.
- Der erste Buchstabe sollte *klein* geschrieben werden, damit man benutzerdefinierte Variablen gut von eingebauten Mathematica-Objekten unterschieden kann, deren Namen grundsätzlich mit einem Grossbuchstaben beginnen.

3.3.3 Listen von Objekten

Sie können mit Hilfe der geschweiften Klammern mehrere Objekte zu einer Liste zusammenfassen und anschließend gemeinsam behandeln, z.B.:

In[1]:=
```
liste = {3,4,5}
```
Out[1]=
```
{3, 4, 5}
```
In[2]:=
```
liste ^ 2
```
Out[2]=
```
{9, 16, 25}
```
In[3]:=
```
Log[liste]//N
```
Out[3]=
```
{1.09861, 1.38629, 1.60944}
```

Man kann ...

- eine Liste einer Variablen als Wert zuweisen (siehe *In[1]* oben)
- Listen in arithmetischen Operationen verwenden (siehe *In[2]* oben)
- Listen als Argumente in mathematischen Funktionen verwenden (siehe *In[3]* oben)

Listen übernehmen in Mathematica die Rolle der Felder (Arrays) anderer Computersprachen. Wenn Sie Elemente eine Liste ansprechen wollen, sind die Indizes in **doppelte** eckige Klammern einzuschließen, z.B.

In[4]:= **liste[[2]]**

Out[4]= 4

3.4 Mathematica-Pakete

Mathematica-Pakete sind Dateien (mit der Extension "**.m**"), die zahlreiche Mathematica-Definitionen enthalten und dadurch spezielle Anwendungsgebiete erschließen. Zum Mathematica-Lieferumfang gehören zahlreiche *Standardpakete*. Mit dem Help Browser (**Add-ons & Links > Standard Packages**) kann man sich einen Überblick über die Standardpakete und ihre Funktionen verschaffen.
Die Paketfunktionen werden genauso benutzt wie die eingebauten Funktionen. Manche regelmäßig verfügbare Mathematica-Funktionen sind sogar über Pakete realisiert, die beim Start automatisch geladen werden.

Bevor man die Funktionen eines Paketes benutzen kann, muss dieses geladen werden, wobei in der zuständigen Anweisung sein Name zwischen Rückwärts-Hochkommata hinter dem Namen des Ablage-Ordners angegeben wird, z.B.:

In[1]:= **data={1,2,3}**

Out[1]= {1, 2, 3}

In[2]:= **<<Statistics`DescriptiveStatistics`**

Im Beispiel wird das im Ordner

…\Wolfram Research\Mathematica\5.0\AddOns\StandardPackages\Statistics

abgelegte Paket **DescriptiveStatistics** geladen.
Sofern das Laden klappt, produziert Mathematica übrigens *keine* Ausgabezelle mit Erfolgsmeldung.

Im Beispiel kann nun per **GeometricMean** das geometrische Mittel der in `data` abgelegten Listenwerte berechnet werden:

In[3]:= **GeometricMean[data]//N**

Out[3]= 1.81712

Achtung: Wenn Sie *vor* dem Laden eines Paketes den Namen einer darin befindlichen Funktion verwendet haben, müssen Sie die damit eingeführte Definition zunächst wieder aus dem System entfernen, weil sonst die konkurrierende Definition des Paketes *nicht* benutzt wird, z.B.:

In[2]:= **GeometricMean[data]**

Out[2]= GeometricMean[{1, 2, 3}]

In[3]:= **<<Statistics`DescriptiveStatistics`**

```
GeometricMean::shdw: Symbol GeometricMean appears in multiple
   contexts {Statistics`DescriptiveStatistics`,Global`};
   definitions in context Statistics`DescriptiveStatistics` may
   shadow or be shadowed by other definitions.
```

Mit dem Befehl **Remove** entfernt man ein Symbol komplett aus dem System, so dass es beim Importieren von Paketen nicht mehr stört.

In[4]:= **Remove[GeometricMean]**

Mit der Anweisung

In[4]:= **GeometricMean=.**

bzw. dem mit dem äquivalenten Ausdruck

In[4]:= **Clear[GeometricMean]**

verliert ein vorbelastetes Symbol zwar seinen aktuellen Wert, doch es wird *nicht* frei für eine neue Funktionsdefinition.

3.5 Algebraische Manipulationen von Ausdrücken

Bisher hat Mathematica für uns numerische Lösungen (exakt oder angenähert) ermittelt. Nun lernen wir seine Fähigkeiten kennen, algebraische bzw. arithmetische Ausdrücke mit **Symbolen** umzuformen, z.B.:

In[1]:= **Sqrt[z]^2 (3x - y - 2x)(y + x) z^-1**

Out[1]= (x - y) (x + y)

Wie *Out[1]* zeigt, bemüht sich Mathematica automatisch, Ausdrücke zu vereinfachen. Darüber hinaus gibt es zahlreiche Möglichkeiten, die Umformung eines algebraischen Ausdrucks zu steuern, von denen hier nur die wichtigsten vorgeführt werden sollen:

In[2]:= **Expand[%]**

Out[2]= $x^2 - y^2$

In[3]:= **Factor[%]**

Out[3]= (x - y) (x + y)

In[4]:= **Apart[%^-1]**

Out[4]= $-\frac{1}{2x(-x+y)} + \frac{1}{2x(x+y)}$

In[5]:= **Simplify[%]**

Out[5]= $\frac{1}{x^2 - y^2}$

In den Beispielen wurden folgende Funktionen benutzt:

Expand[*Ausdruck*]	Produkte und Potenzen werden ausmultipliziert, bei rationalen Ausdrücken aber nur im Zähler.
ExpandAll[*Ausdruck*]	Im Unterschied zu **Expand** wird bei rationalen Ausdrücken auch der Nenner ausmultipliziert.
Factor[*Ausdruck*]	**Factor** versucht, den Ausdruck in Faktoren zu zerlegen.
Apart[*Ausdruck*]	Zerlegt einen rationalen Ausdruck in eine Summe von Termen mit möglichst einfachen Nennern.
Simplify[*Ausdruck*]	Sucht eine möglichst einfache Form des Ausdrucks.

Die bisher genannten Transformationen sind *immer* korrekt, d.h. gültig für alle möglichen Werte der Variablen. Das gilt z.B. auch für die folgende, von Mathematica automatisch ausgeführte, Vereinfachung der ganzzahligen Potenz eines Produktes:

In[1]:= **(x y)^2**

Out[1]= $x^2 y^2$

Mathematica vereinfacht Ausdrücke der Form $(x\,y)^c$ jedoch *nicht* automatisch, wenn c keine ganze Zahl ist, z.B.:

In[2]:= **(x y)^c**

Out[2]= $(x\,y)^c$

Die Potenzregel

$$(x \cdot y)^c = x^c \cdot y^c$$

gilt nämlich bei nicht-ganzzahligem c und Beschränkung auf reellwertige Ergebnisse nur für $x, y > 0$.

PowerExpand führt demgegenüber auch Transformationen durch, die nur unter bestimmten Voraussetzungen stimmen, im Beispiel für $x, y > 0$:

In[3]:= **PowerExpand[%]**

Out[3]= $x^c y^c$

Im **Help Browser** finden Sie eine komplette Liste der möglichen Transformationen von algebraischen Ausdrücken unter **Built-in Functions > Algebraic Computation**.

3.6 Methoden der Analysis

In diesen Abschnitt beschäftigen wir uns damit, wie analytische Standardmethoden (z.B. Differenzieren, Integrieren) in Mathematica unterstützt werden.

3.6.1 Differenzieren

Mathematica kann alle elementaren mathematischen Funktionen ableiten. Die folgende Darstellung beschränkt sich auf die wichtigsten Möglichkeiten. Eine vollständige Beschreibung finden Sie in der Online-Hilfe zu den Mathematica-Operatoren **D** und **Dt**.

3.6.1.1 Partielle Ableitung (Operator D)

Mit dem Mathematica-Operator **D** können Sie eine Funktion f partiell nach einer Variablen x ableiten lassen, wobei für alle anderen Symbole im Funktionsausdruck angenommen wird, dass sie nicht von x abhängen.

D[*f*, *x*]	$\frac{\partial f}{\partial x}$

Im folgenden Beispiel wird n als Konstante behandelt:

In[1]:= **D[x^n,x]**

Out[1]= $n\,x^{-1+n}$

3.6.1.2 Totale Ableitung (Operator Dt)

Bei der so genannten *totalen* Ableitung einer Funktion f nach einer Variablen x wird im Gegensatz zu der eben beschriebenen partiellen Ableitung angenommen, dass die restlichen Symbole im Ausdruck von x abhängen.

Dt[*f*, *x*]	$\frac{df}{dx}$

Beispiel:

```
In[1]:= Dt[x^n,x]
Out[1]= x^n (n/x + Dt[n, x] Log[x])
```

Hier wird angenommen, dass *n* von *x* abhängen kann. "Dt[n, x]" steht für $\frac{dn}{dx}$. Wer will, kann sich von der Richtigkeit des Ergebnisses überzeugen:

$$\left[x^{n(x)}\right]' = \left[e^{n(x)\ln(x)}\right]' = x^{n(x)}\left[n'(x)\ln(x) + \frac{n(x)}{x}\right] = n(x)x^{n(x)-1} + x^{n(x)}n'(x)\ln(x)$$

3.6.1.3 Ableiten unbekannter Funktionen

Mathematica kann auch Ausdrücke mit Symbolen für unbekannte Funktionen ableiten, z.B.:

```
In[1]:= D[f[x]^2,x]
Out[1]= 2 f[x] f'[x]
```

Im nächsten Beispiel liefert Mathematica die Ableitungsregel für Quotienten von Funktionen:

```
In[2]:= D[f[x]/g[x],x]
Out[2]= f'[x]/g[x] - f[x] g'[x]/g[x]^2
```

3.6.2 Integrieren

3.6.2.1 Unbestimmtes Integral

Mathematica kann zahlreiche Stammfunktionen bestimmen.

Integrate[*f*, *x*]	$\int f(x)dx$

Um seine Leistungen zu testen, wollen wir zunächst eine Stammfunktion *F* zur Funktion

$$f(x) = \frac{x}{\sqrt{1-x^2}}$$

mit der Substitutionsregel selbst bestimmen (vgl. z.B. Heuser 1986, Teil 1, S. 442):

Wir substituieren *x* durch $\varphi(t) := \sqrt{1-t}$, wobei gilt: $\varphi'(t) = -\frac{1}{2}\frac{1}{\sqrt{1-t}}$ und $\varphi^{-1}(x) = 1-x^2$. Die Stammfunktion $\Phi(t)$ von $f(\varphi(t))\varphi'(t)$ ist leicht zu ermitteln:

$$\int f(\varphi(t))\varphi'(t)dt = -\frac{1}{2}\int \frac{\sqrt{1-t}}{\sqrt{1-1+t}}\frac{1}{\sqrt{1-t}}dt = -\frac{1}{2}\int \frac{1}{\sqrt{t}}dt = -\frac{1}{2}\frac{t^{\frac{1}{2}}}{\frac{1}{2}} = -\sqrt{t}$$

Dann ist (wg. der Ketten- und der Umkehrregel der Differentiation) $F(x) := \Phi(\varphi^{-1}(x))$ eine Stammfunktion zu *f*:

$$F(x) = -\sqrt{1-x^2}$$

Nun die Lösung von Mathematica:

In[1]:=
```
Integrate[x/Sqrt[1-x^2],x]
```

Out[1]=
```
-Sqrt[1 - x^2]
```

3.6.2.2 Bestimmtes Integral

Auch bei bestimmten Integralen mit symbolischen Grenzen versucht Mathematica, einen geschlossenen Ausdruck als Lösung zu finden.

Integrate[*f*, {*x*, *min*, *max*}] $\int_{min}^{max} f(x)dx$

Auch hier wollen wir zur Probe ein Beispiel per Hand ausrechnen (mit partieller Integration, siehe z.B. Heuser 1986, Teil 1, S. 463):

$$\int_0^a e^x x dx = e^x x\Big|_0^a - \int_0^a e^x dx = e^a a - e^x\Big|_0^a = e^a a - (e^a - 1) = e^a(a-1) + 1$$

Die Lösung von Mathematica:

In[1]:=
```
Integrate[Exp[x]x,{x,0,a}]
```

Out[1]=
```
1 + (-1 + a)ℯ^a
```

3.6.2.3 Numerische Lösungen

Für manche bestimmte Integrale existiert kein geschlossener Ausdruck, z.B. für das Integral der Normalverteilungsdichte:

$$\frac{1}{\sqrt{2\pi}} \int_{-\infty}^{a} e^{-\frac{x^2}{2}} dx$$

In[1]:=
```
Sqrt[2Pi]^-1 Integrate[Exp[-1/2 * x^2],{x,-Infinity,a}]
```

Out[1]=
```
1/2 (1 + Erf[a/Sqrt[2]])
```

Mathematica kann lediglich eine Reformulierung der Aufgabe unter Verwendung der so genannten Fehlerfunktion (**Erf**) anbieten:

$$\frac{2}{\sqrt{\pi}} \int_0^z e^{-t^2} dt$$

In einer solchen Situation erhält man durch Einsetzen konkreter Integrationsgrenzen jedoch eine *numerische* Lösung, z.B.:

```
In[2]:= Sqrt[2Pi]^-1 Integrate[Exp[-1/2 * x^2],{x,-Infinity, 1.96}]
Out[2]= 0.975002
```

3.6.3 Summen und Produkte

3.6.3.1 Summe $\sum_{i=i\min}^{i\max} f$

In folgendem Beispiel werden die ersten 11 Glieder der geometrischen Reihe zur Basis *a* aufaddiert, wobei die Input-Zelle mit Hilfe der Eingabe-Palette in mathematischer Standardnotation erstellt wurde:

```
In[1]:= Sum[a^i, {i, 0, 10}]
```

$$\sum_{i=0}^{10} a^i$$

```
Out[1]= 1 + a + a^2 + a^3 + a^4 + a^5 + a^6 + a^7 + a^8 + a^9 + a^10
```

Wird ein expliziter Wert als Basis angegeben, dann produziert Mathematica ein numerisches Ergebnis, das nach der in Abschnitt 3.2.2 angegebenen Regel entweder exakt:

```
In[2]:= Sum[(1/2)^i,{i,0,10}]
Out[2]= 2047/1024
```

oder angenähert ausgegeben wird:

```
In[3]:= Sum[0.5^i,{i,0,10}]
Out[3]= 1.99902
```

Die Syntax für das Berechnen einer Summe ist denkbar einfach:

Sum[*f*, {*i*, *imin*, *imax*, *di*}]

Mit dem optionalen letzten Parameter kann die Schrittweite für den Laufindex auf einen alternativen Wert gesetzt werden, z.B.:

```
In[4]:= Sum[a^i,{i,0,10,2}]
Out[4]= 1 + a^2 + a^4 + a^6 + a^8 + a^10
```

Mathematica kann den Wert konvergenter unendlicher Reihen bestimmen, z.B. bei der geometrischen Reihe zu einer Basis *a* mit $|a| < 1$, die gegen $\frac{1}{1-a}$ konvergiert (Endl & Luh 1983, S. 2):

```
In[5]:= Sum[(1/2)^i,{i,0,Infinity}]
Out[5]= 2
```

3.6.3.2 Doppelsumme $\sum_{i=i\min}^{i\max}\sum_{j=j\min}^{j\max} f$

In folgendem Beispiel werden vier geometrische Reihen addiert:

In[1]:= **Sum[(1/i)^j,{i,2,5},{j,0,Infinity}]**

Out[1]= $\frac{73}{12}$

Die Mathematica-Syntax für Doppelsummen:

Sum[*f*, {*i*, *imin*, *imax*}, {*j*, *jmin*, *jmax*}]

3.6.3.3 Produkt $\prod_{i=i\min}^{i\max} f$

Wegen

$$\prod_{i=i}^{\infty} e^{(1/2)^i} = e^{\sum_{i=1}^{\infty}(1/2)^i} = e^1$$

kann man die Eulersche Zahl auch auf die folgende, umständliche, Art ausrechnen:

In[1]:= **Product[Exp[0.5^i],{i,1,Infinity}]**

Out[1]= 2.71828

Die Mathematica-Syntax für Produkte:

Product[*f*, {*i*, *imin*, *imax*, *di*}]

3.6.4 Vergleiche und logische Ausdrücke

Mit dem einfachen Gleichheitszeichen wird in Mathematica einer Variablen ein Ausdruck zugeordnet, z.B.:

In[1]:= **y=2**

Out[1]= 2

In[2]:= **x=2y**

Out[2]= 4

Ein *doppeltes* Gleichheitszeichen steht (wie in den Programmiersprachen C, C++, Java, C#, etc.) für den *Identitätsoperator*, mit dem getestet wird, ob die Ausdrücke auf den beiden Seiten des Operatorzeichens übereinstimmen, z.B.:

In[3]:= **x==4**

Out[3]= True

Mathematica kann auch symbolische Ausdrücke miteinander vergleichen:

In[4]:= **a-a==0**

Out[4]= True

Zu komplexeren Identitäten äußert sich Mathematica aber nicht, z.B.:

In[5]:= **(a+b)(a-b)==a^2-b^2**

Out[5]= (a–b) (a+b) == a^2 – b^2

Als Notbehelf bietet sich die Funktion **Simplify** an (vgl. Abschnitt 3.5):

In[6]:= **Simplify[(a+b)(a-b) – (a^2-b^2)]**

Out[6]= 0

Wir haben eben mit Hilfe des Identitätsoperators **Vergleiche** formuliert, also sprachliche Gebilde, die wahr, falsch oder unbestimmt (z.B. „a == b“ mit undefinierten Variablen *a* und *b*) sein können. Natürlich sind auch alle anderen Vergleichsoperatoren in Mathematica vertreten:

x == *y*	gleich
x != *y*	ungleich
x > *y*	größer als
x >= *y*	größer oder gleich
x < *y*	kleiner als
x <= *y*	kleiner oder gleich

Beispiele:

In[1]:= **6 > 8**

Out[1]= False

In[2]:= **a < b**

Out[2]= a < b

Der letzte Ausdruck ist unbestimmt, weil die Variablen *a* und *b* momentan keine numerische Bedeutung haben.

Ein Vergleich ist ein besonders einfach aufgebauter **logischer Ausdruck**. Mit Hilfe von logischen Operatoren kann man aus vorhandenen logischen Ausdrücken neue, komplexere herstellen. Die drei bekanntesten logischen Operatoren werden in Mathematica (wie in der Programmiersprache C) folgendermaßen notiert:

!*p*	Negation (NOT)
p && *q*	Konjunktion (AND)
p || *q*	Adjunktion (OR)

Beispiel:

In[3]:= **2<4 && 7>6**

Out[3]= True

Weitere logische Operatoren finden Sie im Help Browser unter:

Build-in Functions > Programming > Logical Operations

3.6.5 Gleichungen lösen

Einen Vergleich mit Identitätsoperator bezeichnet man üblicherweise als *Gleichung*, und bei Anwesenheit von Variablen in den beteiligten Ausdrücken interessiert man sich für Werte, welche die Gleichung erfüllen. In diesem Abschnitt beschäftigen wir uns mit der Suche nach den *exakten* Lösungen von Gleichungen. Numerische Lösungsmethoden werden später behandelt.

3.6.5.1 Polynomgleichungen mit einer Variablen

Mit der Mathematica-Funktion **Solve** kann man Polynomgleichungen mit einer höchsten Potenz kleiner oder gleich vier stets lösen, z.B.:

```
In[1]:= Solve[x^2==1,x]
Out[1]= {{x -> -1}, {x -> 1}}
```

Solve liefert (im Gegensatz zur später vorzustellenden Funktion **Reduce**) als Ergebnis eine Liste von **Ersetzungssregeln** für *x*-Werte, welche die Gleichung erfüllen.
Wendet man diese Ersetzungsregeln auf den einfachen Ausdruck *x* an, resultiert eine Liste mit den Lösungswerten:

```
In[2]:= x/.%
Out[2]= {-1, 1}
```

Man kann die Ersetzungen aber auch auf beliebige andere Ausdrücke mit der Variablen *x* anwenden:

```
In[3]:= Exp[x]/.{{x -> -1}, {x -> 1}}
```

$$\text{Out[3]=}\ \left\{\frac{1}{e}, e\right\}$$

Die allgemeine Syntax zur Verwendung von Ersetzungsregeln:

Ausdruck /. *Ersetzungsregeln*	Verwende die Liste der Ersetzungsregeln, um Werte für den Ausdruck zu erhalten.

Es können auch Gleichungen gelöst werden, die symbolische Parameter enthalten. Im folgenden Beispiel erhalten wir als Lösungsmenge die wohlbekannte „p-q-Regel“:

```
In[4]:= Solve[x^2 + p x + q == 0, x]
```

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

Erinnerung: Sollten Sie eine andere Lösungsmenge erhalten haben, waren in Ihrer Mathematica-Sitzung mit großer Wahrscheinlichkeit die Symbole *a* und *b* bereits mit Werten belegt.

Mathematica kann auch einige Polynomgleichungen mit Potenzen *größer* als vier lösen, z.B.:

```
In[5]:= Solve[x^5==5,x]
```

$$\text{Out[5]=}\ \left\{\left\{x \to -(-5)^{1/5}\right\}, \left\{x \to 5^{1/5}\right\}, \left\{x \to (-1)^{2/5}\,5^{1/5}\right\}, \left\{x \to -(-1)^{3/5}\,5^{1/5}\right\}, \left\{x \to (-1)^{4/5}\,5^{1/5}\right\}\right\}$$

Ist eine explizite Bestimmung der Lösungen nicht möglich, wird das Ergebnis in symbolischer Form mit Hilfe der **Root**-Funktion ausgedrückt, z.B.:

```
In[7]:= Solve[x^6-x^5==6,x]
Out[7]= {{x → Root[-6 - #1^5 + #1^6 &, 1]}, {x → Root[-6 - #1^5 + #1^6 &, 2]},
  {x → Root[-6 - #1^5 + #1^6 &, 3]}, {x → Root[-6 - #1^5 + #1^6 &, 4]},
  {x → Root[-6 - #1^5 + #1^6 &, 5]}, {x → Root[-6 - #1^5 + #1^6 &, 6]}}
```

Root[*f*, *k*] repräsentiert die *k*-te Wurzel (Nullstelle) der Gleichung *f*[*x*] == 0, also eine feste, wenngleich nur implizit definierte, Zahl. Dabei wird *f* als so genannte **reine Funktion** (**pure function**) notiert, wobei das &-Zeichen hinter der Funktionsvorschrift besagt, dass die Argumente keine Namen haben, sondern mit #1, #2, usw. durchnumeriert sind.
Man kann **Root**-Objekte in weiteren Ausdrücken verwenden und z.B. mit Hilfe der **N**–Funktion die numerischen Lösungen der untersuchten Polynomgleichung anfordern:

```
In[8]:= N[%]
Out[8]= {{x → -1.22}, {x → 1.59015},
  {x → -0.537119 - 1.13651 i}, {x → -0.537119 + 1.13651 i},
  {x → 0.852045 - 1.10965 i}, {x → 0.852045 + 1.10965 i}}
```

3.6.5.2 Transzendente Gleichungen in einer Variablen

Mathematica kann auch manche transzendente Gleichungen lösen, wobei in der Regel eine Warnung darauf hinweist, dass vermutlich nicht alle Lösungen gefunden wurden, z.B.:

```
In[1]:= Solve[Sin[x]==a,x]
Solve::ifun:
 Inverse functions are being used by Solve, so some solutions may not
  be found; use Reduce for complete solution information.
Out[1]= {{x → ArcSin[a]}}
```

In der Regel können transzendente Gleichungen jedoch *nicht* in symbolischer Form gelöst werden, z.B.:

```
In[2]:= Solve[Cos[x]==x,x]
Solve::tdep: The equations appear to involve the variables
  to be solved for in an essentially non-algebraic way.
Out[2]= Solve[Cos[x]==x,x]
```

Mit der Funktion **FindRoot** und einem geeigneten Startwert lässt sich aber eine numerische Näherungslösung ermitteln, z.B.:

```
In[3]:= FindRoot[Cos[x]==x,{x,1}]
Out[3]= {x → 0.739085}
```

Mit den numerischen Lösungen für transzendente Gleichungen werden wir uns später noch näher beschäftigen.

3.6.5.3 Gleichungssysteme mit mehreren Variablen

Mit der Mathematica-Funktion **Solve** lassen sich alle linearen Gleichungssysteme und zahlreiche Systeme mit Polynomgleichungen explizit lösen.

i) Zwei Gleichungen mit einer Variablen *x*

In[1]:=
```
Solve[{x^2==1,x^3==1},x]
```
Out[1]=
```
{{x→1}}
```

ii) Zwei Gleichungen in zwei Variablen *x* und *y*

Hier wird man in der Regel nach *x und y* auflösen wollen, weil eine Auflösung nach *x* allein der Suche nach denjenigen *x*-Werten entspräche, welche die Gleichungen für beliebige Werte des symbolischen Parameters *y* erfüllen.

In[2]:=
```
Solve[{y==x/2+3,y==x-3},{x,y}]
```
Out[2]=
```
{{x→12,y→9}}
```

iii) Variablen eleminieren

Wenn z.B. zwei Gleichungen in den drei Veränderlichen *x*, *y* und *a* nach *x* und *y* aufgelöst werden sollen, erhalten wir Ergebnisterme mit dem symbolischen Parameter *a*:

In[3]:=
```
Solve[{x==y-3a, y==2x-a},{x,y}]
```
Out[3]=
```
{{x→4 a,y→7 a}}
```

In dieser Situation kann man aber auch eine Veränderliche, z.B. *a*, eliminieren. Dann bleibt noch *eine* Gleichung mit den beiden Veränderlichen *x* und *y* übrig:

$$\begin{array}{rcrcrcl} x & - & y & + & 3a & = & 0 \\ 2x & - & y & - & a & = & 0 \\ \hline 7x & - & 4y & & & = & 0 \end{array}$$

Mit folgender Syntax erhalten Sie sofort die Auflösung nach *x* bei Elimination von *a*:

In[4]:=
```
Solve[{x==y-3a, y==2x-a},x,a]
```
Out[4]=
$$\left\{\left\{x \to \frac{4\,y}{7}\right\}\right\}$$

iv) Die allgemeine Syntax der Funktion Solve

Solve[{ls_1==rs_1, ls_2==rs_2, ...}, {*x*, *y*, ...}, {*a*, *b*, ...}]	Löse das Gleichungssystem nach *x*, *y*, ... durch Eliminieren von *a*, *b*, ...

Die geschweiften Klammern dürfen entfallen, wenn die zugehörige Liste einelementig ist.

3.6.5.4 Vollständige Lösungen

Wir haben in Abschnitt 3.6.5 bislang die **Solve**-Funktion benutzt. Alle dabei erzielten Ergebnisse können wir auch mit der verwandten Funktion **Reduce** ermitteln, wobei aber ein anderes Ausgabeformat verwendet wird. An Stelle von Ersetzungslisten liefert **Reduce** logische Ausdrücke, z.B.:

```
In[1]:= Solve[x^2==1,x]

Out[1]= {{x→-1},{x→1}}

In[2]:= Reduce[x^2==1,x]

Out[2]= x==-1||x==1
```

Dabei stehen die beiden senkrechten Striche für den logischen Operator **OR** (vgl. Abschnitt 3.6.4).

Es gibt jedoch noch einen weiteren wichtigen Unterschied zwischen **Solve** und **Reduce**. Während **Solve** nur *allgemeingültige* Lösungen liefert, sucht **Reduce** zusätzlich auch die Lösungen, die nur für *bestimmte* Werte anderer Variablen gelten:

```
In[3]:= Solve[a x == 0, x]

Out[3]= {{x→0}}
```

Reduce berücksichtigt auch die Möglichkeit, dass *a* gleich Null ist:

```
In[4]:= Reduce[a x == 0, x]

Out[4]= a == 0 || x == 0
```

Die Syntax der Funktion **Reduce** stimmt mit der **Solve**-Syntax überein (s.o.).

3.6.6 Differentialgleichungen

Die Funktion **DSolve** löst lineare und nichtlineare gewöhnliche Differentialgleichungen sowie Differentialgleichungssysteme und liefert wie die Funktion **Solve** Ersetzungsregeln als Ergebnisse. In folgendem Beispiel wird die Differentialgleichung

$$y'(x) = a\,y(x)$$

gelöst, die z.B. zur Modellierung eines Populationswachstums oft Verwendung findet. Man nimmt dabei an, dass zu jedem Zeitpunkt *x* die Veränderung *y*'(*x*) der Population im Sinne einer linearen Funktion vom derzeitigen Bestand *y*(*x*) abhängt.

```
In[1]:= DSolve[y'[x]==a y[x],y[x],x]

Out[1]= {{y[x] → ℯ^(a x) C[1]}}
```

Die unbestimmte Konstante C[1] in der Lösungsfunktion kann über eine **Anfangsbedingung** festgelegt werden, z.B.:

```
In[2]:= DSolve[{y'[x]==a y[x], y[0]==1},y[x],x]

Out[2]= {{y[x] → ℯ^(a x)}}
```

Wir erhalten das Ergebnis in Form einer Ersetzungsregel für y[x].

Leider lässt sich diese Ersetzungsregel nur eingeschränkt weiter verwenden, wie der folgende misslungene Versuch einer „Probe“ für die gefundene Lösung demonstriert:

```
In[3]:= y'[x] == a y[x] /. %

Out[3]= {y'[x] == a ℯ^(a x)}
```

Offenbar wird zwar y[x] ersetzt, doch ist keine Ersetzung für die Ableitung y'[x] verfügbar.

Um besser verwendbare Ersetzungsregeln zu erhalten, muss man im **DSolve**-Aufruf statt der Lösung für $y(x)$ eine Lösung für y im Form einer so genannten **reinen Funktion** (**pure function**) anfordern:

In[4]:=
```
DSolve[{y'[x]==a y[x], y[0]==1},y,x]
```

Out[4]=
```
{{y → Function[{x}, e^(a x)]}}
```

Wir erhalten eine Ersetzungsregel mit **Function**-Objekt, die sich wunschgemäß weiterverarbeiten lässt:

In[5]:=
```
y'[x] == a y[x] /. %
```

Out[5]=
```
{True}
```

Die allgemeine Syntax der Funktion **DSolve**:

DSolve[{gl_1, gl_2, ...}, y[x], x]	Löse die Differentialgleichungen nach y[x] auf, mit x als Argument der Lösungsfunktionen
DSolve[{gl_1, gl_2, ...}, y, x]	Ermittle eine Lösung für y in Form einer reinen Funktion.

Die geschweiften Klammern dürfen entfallen, wenn die zugehörige Liste einelementig ist.

3.6.7 Potenzreihen (Taylor-Reihen)

Viele Funktionen lassen sich durch Potenzreihen endlicher Ordnung gut approximieren. Eine wesentliche Rolle spielt dabei der *Satz von Taylor*:

> Sei f auf dem Intervall I (n+1) mal stetig differenzierbar und x_0 ein innerer Punkt von I. Dann gilt für alle $x \in I$:
>
> $$f(x) = \sum_{\upsilon=0}^{n} \frac{f^{(\upsilon)}(x_0)}{\upsilon!}(x - x_0)^{\upsilon} + R_n(x, x_0)$$
>
> Zum Restglied R_n siehe z.B. Endl & Luh (1983, Band 2, S.202ff).

Mit der Mathematica-Operation **Series** kann man für viele Funktionen eine Potenzreihenentwicklung um einen Punkt x_0 bis zur gewünschten Ordnung n ermitteln, z.B.:

In[1]:=
```
Series[Exp[x], {x,0,4}]
```

Out[1]=
$$1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \frac{x^4}{24} + O[x]^5$$

An der Lösung für eine unbekannte Funktion f wird klar, dass Mathematica den Satz von Taylor verwendet:

In[2]:=
```
Series[f[x], {x,0,4}]
```

Out[2]=
$$f[0] + f'[0]\,x + \frac{1}{2} f''[0]\,x^2 + \frac{1}{6} f^{(3)}[0]\,x^3 + \frac{1}{24} f^{(4)}[0]\,x^4 + O[x]^5$$

Für manche Funktionen liefert Mathematica allerdings Potenzreihenentwicklungen, die *keine* Taylor-Reihen sind.

Die Syntax der Funktion **Series**:

Series[*Ausdruck*, {*x*, x_0, *n*}	Ermittle die Reihenentwicklung für den von x abhängigen Ausdruck um den Entwicklungspunkt x_0 bis zur *n*-ten Ordnung.

3.6.8 Grenzwerte

In Mathematica können Sie z.B. den Grenzwert der Funktion $\frac{\sin(x)}{x}$ für $x \to 0$ folgendermaßen bestimmen:

In[1]:=
```
Limit[Sin[x]/x, x -> 0]
```

Out[1]=
```
1
```

3.7 Graphische Darstellung von Funktionen und Daten

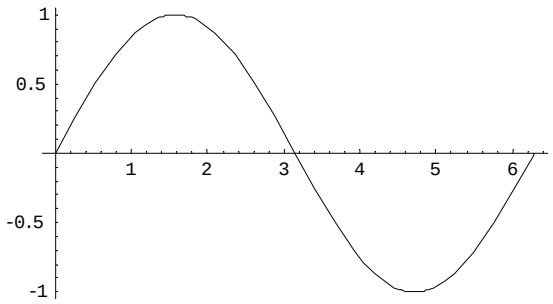
In diesem Abschnitt beschäftigen wir uns mit der graphischen Darstellung von Funktionen und Daten.

Bei der Ausführung von Grafikbefehlen (z.B. **Plot**) erstellt der Kernel zunächst einen (auch als *Grafikobjekt* bezeichneten) Mathematica-Ausdruck, in dem alle graphischen Elemente (z.B. Linien, Punkte, Polygone) durch so genannte **Grafik-Primitive** dargestellt sind. Anschließend übersetzt der Kernel das Grafikobjekt in die Seitenbeschreibungssprache **Postscript** und übergibt es in dieser Form an das Frontend, das die Darstellung auf dem gewünschten Ausgabegerät (z.B. Bildschirm) erledigt.

3.7.1 Funktionsverläufe plotten

Sie können den **Plot**-Befehl verwenden, um den Graphen einer reellwertigen Funktionen von *einer* Variablen zu zeichnen, z.B.:

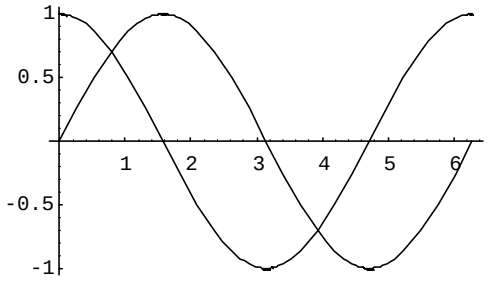
In[1]:=
```
Plot[Sin[x], {x,0,2 Pi}]
```



Out[1]=
```
-Graphics-
```

Es sind auch simultane Darstellungen von *mehreren* Funktionen möglich, z.B.:

In[2]:= **Plot[{Sin[x], Cos[x]}, {x,0,2 Pi}]**



Out[2]= -Graphics-

Die Syntax des **Plot**-Befehls:

Plot[*f*,{*x*, *xmin*, *xmax*}]	*f* wird als Funktion von *x* im Intervall [*xmin*, *xmax*] gezeichnet.
Plot[{f_1, f_2, ...},{x, *xmin*, *xmax*}]	Es werden mehrere Funktionen gezeichnet.

Wer den zu einem Plot äquivalenten Mathematica-Ausdruck sehen möchte, kann diesen per **InputForm** anfordern, z.B.:

In[3]:= **InputForm[%]**

Out[3]//InputForm=

```
Graphics[{{{Line[{{2.617993877991494*^-7, 2.6179938779914644*^-7}, {0.25488992540742256,
  0.25213889196341294},
   {0.5328694051959509, 0.508006999749293}, {0.7939393140028286, 0.7131204212611485},
   {1.04500937601917, 0.864929243756943}, {1.1741328775392965, 0.9223551787683757},
      . . .
      . . .
   {5.756221700231304, -0.502911193409889}, {6.016521477574974, -0.2635146543573849},
   {6.266821408128108, -0.01636316874809926}, {6.283185045380199, -2.6179938774695577*^-7}}]}},
 {{Line[{{2.617993877991494*^-7, 0.9999999999999657}, {0.007756134596682049,
 0.9999699213388479},
   {0.0148126607801162, 0.9998902945462463}, {0.02295800454829159, 0.9997364765884917},
   {0.03065785598007853, 0.9995300847413566}, {0.04435293706021389, 0.9990165697185377},
      . . .
      . . .
   {6.268267781388509, 0.9998887357754854}, {6.276033964522214, 0.9999744292580763},
   {6.283185045380199, 0.9999999999999657}}]}}},
   {PlotRange->Automatic, AspectRatio -> GoldenRatio^(-1), DisplayFunction :> $DisplayFunction,
   ColorOutput -> Automatic, Axes -> Automatic, AxesOrigin -> Automatic,
   PlotLabel -> None, AxesLabel -> None, Ticks -> Automatic, GridLines -> None, Prolog -> {},
   Epilog -> {}, AxesStyle -> Automatic, Background -> Automatic, DefaultColor -> Automatic,
   DefaultFont :> $DefaultFont, RotateLabel -> True, Frame -> False, FrameStyle -> Automatic,
   FrameTicks -> Automatic, FrameLabel -> None, PlotRegion -> Automatic,
   ImageSize -> Automatic, TextStyle :> $TextStyle, FormatType :> $FormatType}]
```

Hier handelt es sich um einen **Graphics**-Aufruf mit zwei **Line**-Elementen und zahlreichen Grafikoptionen (siehe Abschnitt 3.7.2). Aus Platzgründen wurden die **Line**-Definitionen stark gekürzt.

3.7.2 Optionen zum Gestalten von Grafikelementen

Viele Eigenschaften einer Grafik können über **Optionen** beeinflusst werden (z.B. Stützstellen, Beschriftungen). Nach der Mathematica-Syntax werden Optionen generell am Ende eines Funktionsaufrufs als Folge von **Regeln** mit der Syntax

Option -> Wert

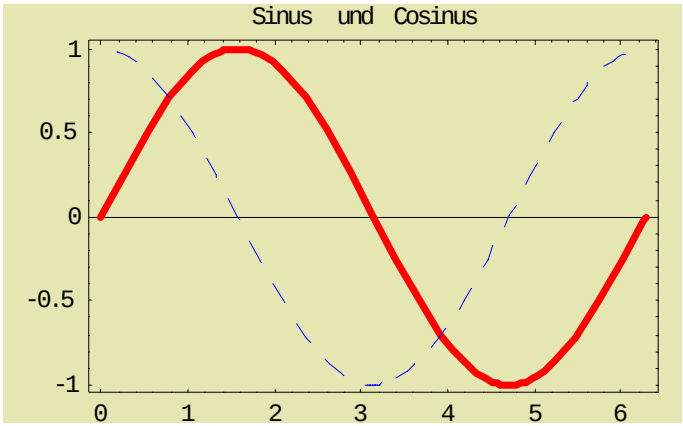
angegeben. Für die **Plot**-Funktion lautet also die erweiterte Syntax:

Plot[{f_1, f_2, ...},{*x*, *xmin*, *xmax*}, *Option1 -> Wert1*, *Option2 -> Wert2*, ...]

Im folgenden Beispiel kommen etliche Grafikoptionen zum Einsatz:

- **Frame** aktiviert einen Rahmen.
- **PlotLabel** ergänzt einen Titel.
- **Background** beeinflusst die Hintergrundfarbe.
- **PlotStyle** definiert eine Liste von Attribut-Listen, die den in einem Befehl angeforderten Plots nacheinander zugewiesen werden. Ist eine Liste einelementig, dürfen die geschweiften Klammern entfallen.

```
In[1]:= Plot[{Sin[x], Cos[x]}, {x, 0, 2 Pi},
         Frame -> True, PlotLabel -> "Sinus und Cosinus",
         Background -> RGBColor[0.9, 0.9, 0.7],
         PlotStyle -> {{RGBColor[1,0,0], AbsoluteThickness[3]},
                       {RGBColor[0,0,1], AbsoluteDashing[{10,10}]}}]
```




```
Out[1]= -Graphics-
```

In der Grafikanweisung **RGBColor** zur Definition eines Farbwertes dürfen die drei Kanäle (Rot, Grün, Blau) jeweils Werte zwischen 0 und 1 annehmen.
Neben **AsoluteThickness** (Liniestärke) und **AbsoluteDashing** (Linienmuster) sind in der **PlotStyle**-Option noch weitere Grafikanweisungen verfügbar, die man im Help Browser folgendermaßen findet:

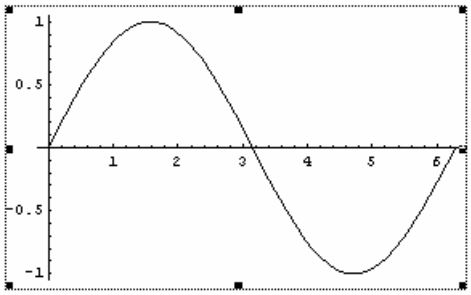
Build-in Functions > Graphics and Sound > Graphics Primitives

In Abschnitt 3.7.11 wird die Option **TextStyle** zum Gestalten von Beschriftungen ausführlich beschrieben. Manche Optionen sind nur bei bestimmten Grafiktypen verfügbar und werden dort behandelt (z.B. **ViewPoint** bei 3D-Plots).
Eine vollständige Liste aller Grafik-Optionen liefert der **Help Browser** unter:

Build-in Functions > Graphics and Sound > Basic Options bzw. **3D Options** etc.

3.7.3 Größe und Position einer Grafik verändern

Nach dem Anklicken einer Grafik wird ein Rahmen mit Anfassern eingeblendet, die eine Größenänderung (bei konstantem Seitenverhältnis) per Maus erlauben:



Während einer Größenänderung erscheinen Breite und Höhe der Grafik in der Statuszeile des Notebook-Fensters.

Eine Grafik kann innerhalb ihrer Zelle, die nötigenfalls nach unten beliebig vergrößert wird, per Maus-Drag-and-Drop verschoben werden. Während der Bewegung erscheinen die aktuellen Koordinaten der linken, oberen Ecke des Grafikrahmens in der Statuszeile des Notebook- Fensters.

3.7.4 Grafiken übertragen

Unter Windows lässt sich eine Mathematica-Grafik folgendermaßen via **Zwischenablage** übertragen (innerhalb eines Notizbuchs, in ein anderes Notizbuch oder in eine andere Anwendung):

- Markieren Sie die Grafik.
- Übertragen Sie die Grafik mit **Edit > Copy** oder **<Strg><C>** in die Zwischenablage.
- Wechseln Sie ggf. zum Zielfenster.
- Nach **Edit > Paste** bzw. **Bearbeiten > Einfügen** oder **<Strg><V>** landet die Grafik dort im voreingestellten Format.
- Nach **Edit > Paste As** bzw. **Bearbeiten > Inhalte einfügen** haben Sie die Wahl zwischen mehreren Datenformaten, die Mathematica in der Zwischenablage erstellt hat.

Unter UNIX oder Mac-OS-X ist ein analoges Vorgehen zu wählen.

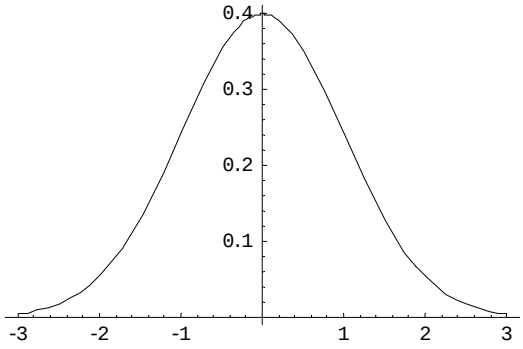
Um die Grafik in eine **Datei** zu schreiben, müssen Sie folgendermaßen vorgehen:

- Markieren Sie die Abbildung.
- Wählen Sie den Menübefehl **Edit > Save Selection As...**
- Wählen Sie ein Grafik-Dateiformat
  Unter Windows sind z.B. folgende Optionen verfügbar:
  - EPS (Encapsulated PostScript File)
  - Bitmap (BMP)
  - Enhanced Metafile (EMF)
  - Windows Metafile (WMF)

3.7.5 Plots in Variablen speichern, abrufen und modifizieren

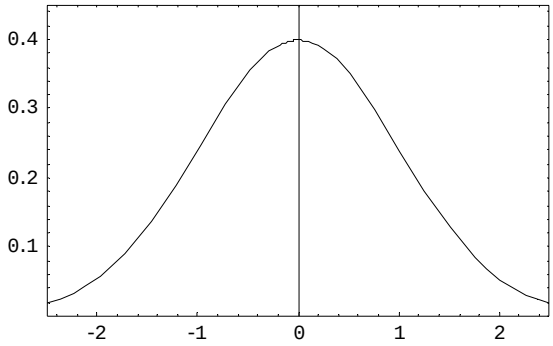
Wie oben erwähnt, stehen hinter den angezeigten Plots letztlich Mathematica-Ausdrücke. Dies ermöglicht es, einen Plot später per **Show** nochmals zu zeichnen, wobei neue Options-Definitionen möglich sind, z.B.:

In[1]:=
```
Plot[Exp[-x^2/2]/Sqrt[2 Pi],{x,-3,3}]
```



Out[1]= -Graphics-

In[2]:=
```
Show[%,Frame->True, PlotRange->{{-2.5,2.5},{0,0.45}}]
```



Out[2]= -Graphics-

Wie *In[2]* demonstriert, kann mit **PlotRange** der anzuzeigende *x*- und *y*-Achsenbereich nachträglich geändert werden. Dabei greift Mathematica allerdings auf *vorhandene* Stützstellen zurück, d.h. bei starker Vergrößerung werden die Geradensegmente sichtbar, aus denen eine Kurve zusammengesetzt ist. Um die Anzahl der Stützstellen mit der Option **PlotPoints** zu erhöhen, muss man die **Plot**-Funktion erneut anwenden. Im **Show**-Befehl stehen also nicht *alle* Grafik-Optionen zur Verfügung.

Um das spätere Abrufen zu erleichtern, können Grafikobjekte in Variablen gespeichert werden. Dies gelingt auch bei den mit **Show** reproduzierten Plots, z.B.:

In[3]:=
```
alt = Plot[Exp[-x^2/2]/Sqrt[2 Pi],{x,-3,3}]
```
```
...
```
In[4]:=
```
neu = Show[alt, Frame->True, FrameLabel->"Dichte"]
```
```
...
```

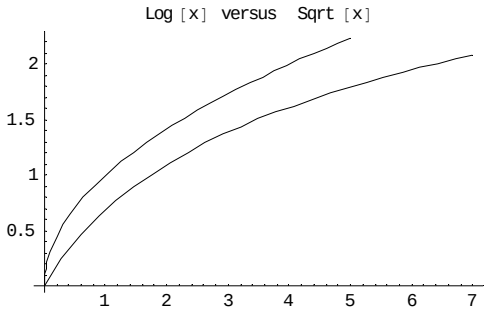
Syntax-Varianten der **Show**-Funktion:

Show[*Plot*]	*Plot* nochmals zeichnen
Show[*Plot*, *Option1* -> *Wert1*, ...]	*Plot* mit neuen Optionen zeichnen

3.7.6 Diagramme kombinieren

Man kann mehrere vorhandene Plots zu einem neuen kombinieren, wobei Mathematica das Koordinatensystem geeignet angepasst, z.B.:

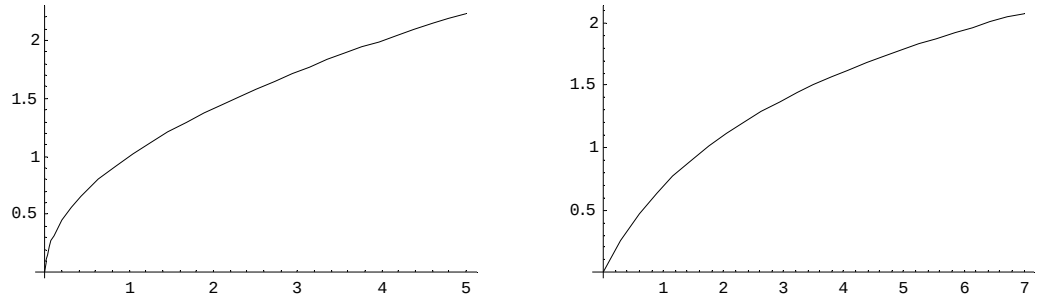
```
In[1]:= p1 = Plot[Sqrt[x],{x,0,5}]
   ...
In[2]:=  p2 = Plot[Log[x+1],{x,0,7}]
   ...
In[3]:=  Show[p1,p2,PlotLabel->"Log[x] versus Sqrt[x]"]
```




```
Out[3]= -Graphics-
```

Man kann aber auch per **GraphicsArray** ein Feld aus *mehreren* Plots aufbauen, z.B.:

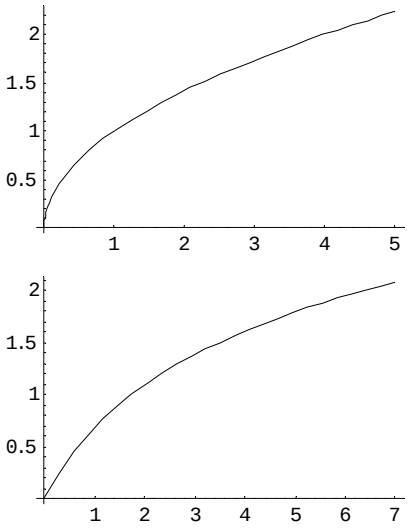
```
In[4]:=  Show[GraphicsArray[{p1, p2}, GraphicsSpacing -> {0.2,0}]
```




```
Out[4]= -Graphics-
```

Über die allgemeine Listen-Syntax (vg. Abschnitt 3.3.3) lassen sich alternative Anordnungen der einzelnen Plots erreichen, z.B.:

In[5]:= **Show[GraphicsArray[{{p1},{p2}}]**



Out[5]= -Graphics-

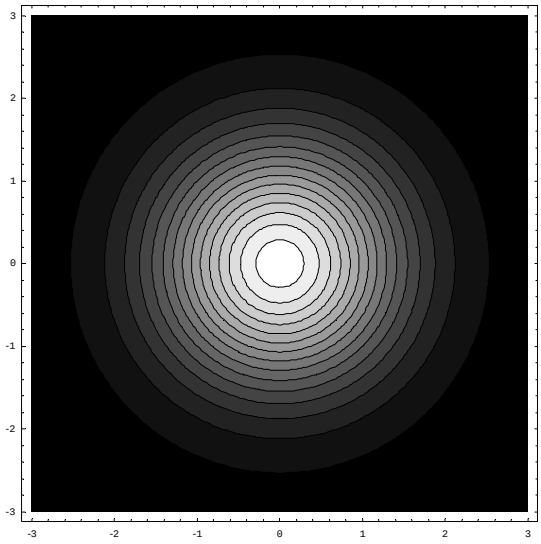
Als neue Syntax-Varianten der **Show**-Funktion traten in diesem Abschnitt auf:

Show[*Plot1*, *Plot2*, ..., *Option1* -> *Wert1*, ...]	Mehrere Plots zu einem neuen Plot kombinieren
Show[GraphicsArray[{{*Plot1*,*Plot2*,...},…},…]]	Ein Feld von *Plots* anzeigen

3.7.7 Konturdiagramme

Konturdiagramme eignen sich für Funktionen vom $\mathbb{R}^2$ in den $\mathbb{R}^1$. Dabei werden Argumentbereiche mit Funktionswerten innerhalb bestimmter Intervalle durch Grenzlinien und Graustufen (je größer der Funktionswert, desto heller) markiert, z.B.:

In[1]:=
```
ContourPlot[Exp[-(x^2+y^2)/2]/Sqrt[2 Pi],{x,-3,3}, {y,-3,3},
 PlotPoints->200,Contours->15]
```



Out[1]= -Graphics-

Mit der **Contours**–Option legt man fest, wie viele Funktionswert-Intervalle gebildet und durch eine eigene Graustufe dargestellt werden sollen. Durch die Option **PlotPoints** wird das Gitternetz mit den

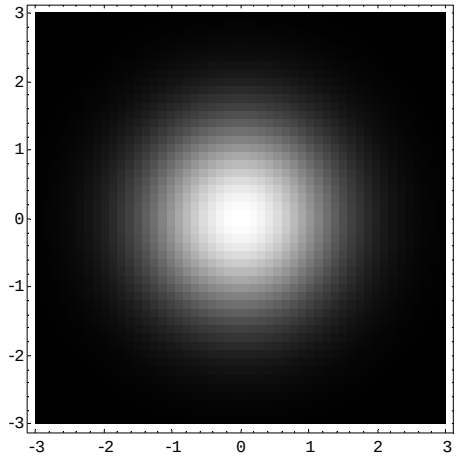
Stützstellen definiert, für die jeweils das zugehörige Funktionswert-Intervall bzw. die Graustufe ermittelt wird.

Im Beispiel wird die bivariate Normalverteilungsdichte dargestellt, die gleich noch aus anderen Perspektiven zu sehen ist.

3.7.8 Dichtediagramme

Dichtediagramme sind ebenfalls für Funktionen vom $\mathbb{R}^2$ in den $\mathbb{R}^1$ geeignet. Die Argumentenebene wird in Quadrate eingeteilt, die mit wachsendem mittleren Funktionswert heller dargestellt werden, z.B.:

In[1]:=
```
DensityPlot[Exp[-(x^2+y^2)/2]/Sqrt[2 Pi],{x,-3,3}, {y,-3,3},
 PlotPoints -> 50, Mesh -> False]
```



Out[1]= `-Graphics-`

Mit **`PlotPoints -> 50`** wird das Gitternetz verfeinert, und mit **`Mesh -> False`** werden die Gitterlinien abgeschaltet.

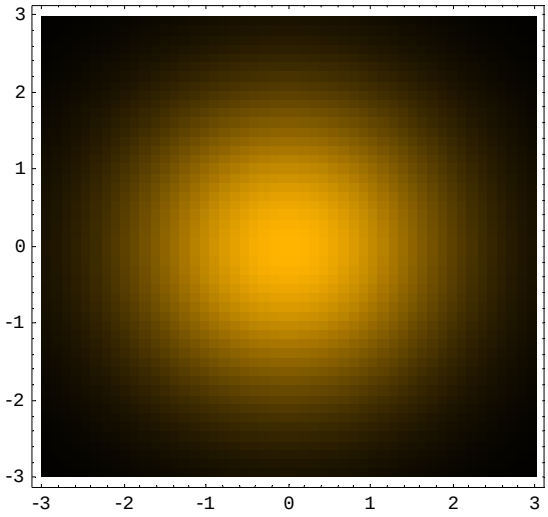
Während ein Konturdiagramm die Urbilder einer (einstellbaren) Anzahl von Funktionswertintervallen durch unterschiedlich Farb- bzw. Helligkeitswerte darstellt, erhält bei einem Dichtediagramm jeder Quadrant im Gitternetz einen individuellen Farb- bzw. Helligkeitswert.

Über die Grafik-Option **ColorFunction** kann man Kontur- oder Dichtediagramme freundlicher und/oder informativer gestalten. Dazu ist eine eigene Funktion zu definieren (siehe unten), die Höhen in Farbwerte umsetzt, z.B.:

In[2]:=
```
cf[f_]:=RGBColor[Sqrt[f],Sqrt[f/2],0]
```

Hier kommt die Grafikanweisung **RGBColor** zum Einsatz, welche die Definition einer Farbe über die Kanäle Rot, Grün und Blau erlaubt, deren Beitrag zwischen 0 und 1 variieren darf.
Im Beispiel erhalten kleine Werte per Wurzelfunktion eine Anhebung, um die große Dunkelzone aufzuhellen. Aus ästhetischen Gründen wird ein gelber Farbton gemischt.
Im **DensityPlot**-Aufruf übernimmt die selbst definierte **ColorFunction** ihr Argument über einen nummerierten Parameter (vgl. Abschnitt 3.6.5.1):

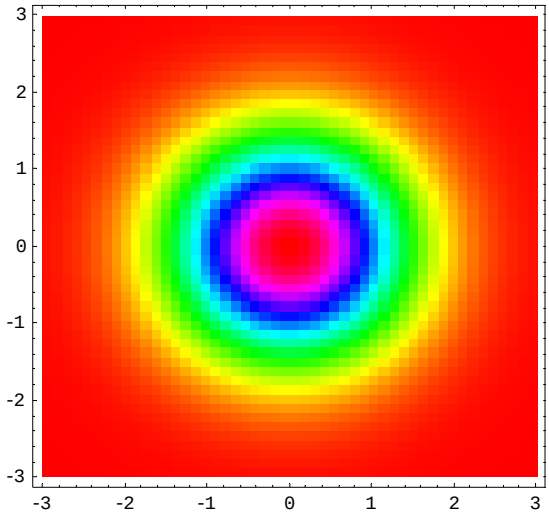
In[3]:=
```
DensityPlot[Exp[-(x^2+y^2)/2]/Sqrt[2 Pi],{x,-3,3}, {y,-3,3},
 PlotPoints -> 50, Mesh -> False, ColorFunction -> (cf[#]&)]
```



Out[3]= -Graphics-

Statt eine eigene **ColorFunction** zu definieren, kann man auch über die Option **ColorFunction -> Hue** Farbe ins Spiel bringen:

In[4]:=
```
DensityPlot[Exp[-(x^2+y^2)/2]/Sqrt[2 Pi],{x,-3,3}, {y,-3,3},
 PlotPoints -> 50, Mesh -> False, ColorFunction -> Hue]
```

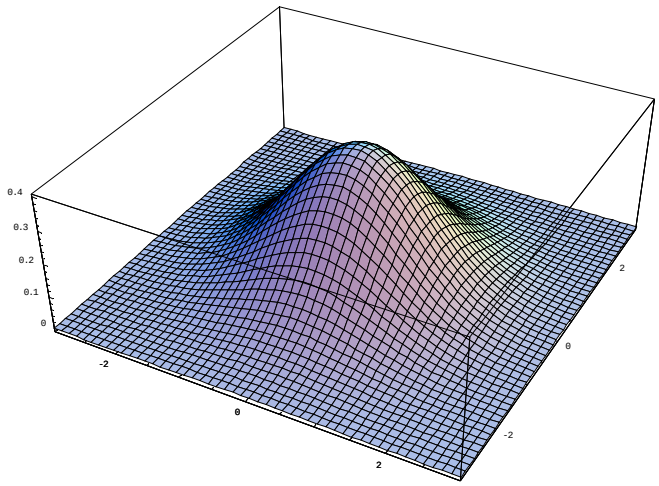


Out[4]= -Graphics-

3.7.9 3D-Oberflächendiagramme

Nun kommen wir zu der bei vielen Abbildungen vom $\mathbb{R}^2$ in den $\mathbb{R}^1$ sehr attraktiven räumlichen Darstellung der Funktionsoberfläche mit **Plot3D**, z.B.:

In[1]:=
```
Plot3D[Exp[-(x^2+y^2)/2]/Sqrt[2 Pi],{x,-3,3}, {y,-3,3},
PlotPoints -> 25]
```



Out[1]= `-Graphics-`

Anders als bei Kontur- bzw. Dichtediagramme sind die Farben hier nicht durch den Funktionswert definiert, sondern resultieren aus den Gelände- und Beleuchtungsverhältnissen.

Zu den Grafik-Optionen, die auch bei *zwei*dimensionalen Grafiken verfügbar sind (z.B. **PlotPoints**, **PlotRange**, **AxesLabel**), gesellen sich spezielle 3D-Optionen. Besonders wichtig ist der bequem einstellbare 3D-Aussichtspunkt:

3.7.10 3D-Aussichtspunkt festlegen

Sie können den Aussichtspunkt festlegen, von dem aus die anschließend zu zeichnenden Grafikobjekte betrachtet werden sollen. Bei jedem speziellen Grafikobjekt beziehen sich die Koordinaten des Aussichtspunktes auf ein Koordinatensystem, das vom dreidimensionalen Rahmen um das Objekt abhängt:

- Die längste Seite des Rahmens hat die Länge 1.
- Der Mittelpunkt des Rahmens hat die Koordinaten {0 ,0, 0}.

Per Voreinstellung wählt Mathematica den Aussichtspunkt mit den kartesischen Koordinaten$\{x, y, z\}$ = {1.3, -2.4, 2}.

Zur Festlegung eines alternativen 3D-Aussichtspunktes dient die Grafik-Option **ViewPoint**, die nach dem Menübefehl

Input > 3D ViewPoint Selector

mit Hilfe folgender Dialogbox produziert werden kann:

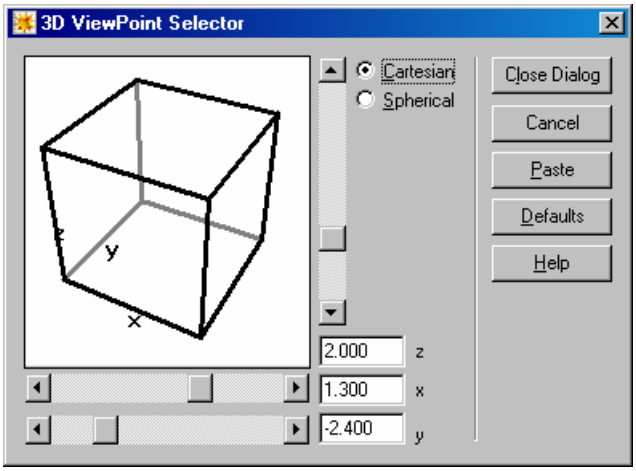


Abbildung 4 Dialogbox zur Festlegung des 3D-Aussichtsspunktes

Zur Spezifikation der Koordinaten sind drei Methoden verfügbar:

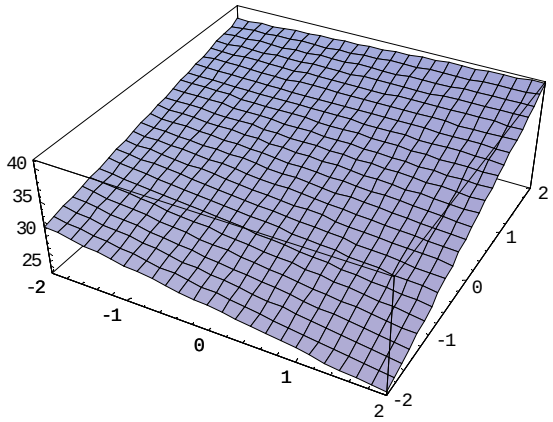
- Würfel per Maus anpacken und drehen
- Rollbalken benutzen
- Numerische Eingabe nach vorheriger Wahl des gewünschten Koordinatensystems (kartesisch oder sphärisch).

Die einem gewählten Aussichtspunkt entsprechende **ViewPoint**-Option überträgt Mathematica nach einem Mausklick auf den **Paste**-Schalter in das Notizbuch ab dem aktuellen Einfügepunkt.

Damit bietet sich z.B. folgende Vorgehensweise an:

- Erstellen Sie das gewünschte 3D-Diagramm, z.B.:

  In[1]:= **`Plot3D[33.92-0.64x + 3.03 y + 0.54 x y,{x,-2,2},{y,-2,2}]`**



  Out[1]= `-Graphics-`

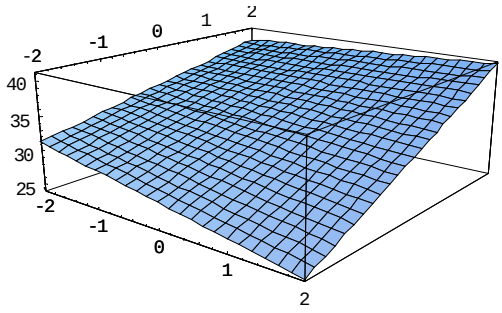
- Bereiten Sie ein **Show**-Kommando zur Aufnahme einer **ViewPoint**-Option vor:

  In[2]:= **`Show[%,|]`**

  (Der senkrechte Strich ist *nicht* einzugeben, sondern stellt die Einfügemarke dar.)

- Öffnen Sie den **3DViewPoint Selector**, wählen Sie den Aussichtspunkt, und klicken Sie auf den **Paste**-Schalter. Daraufhin erscheint die zugehörige **ViewPoint**-Option im **Show**-Aufruf, z.B.:

  In[2]:= **`Show[%, ViewPoint->{1.600,-2.000,0.800}]`**

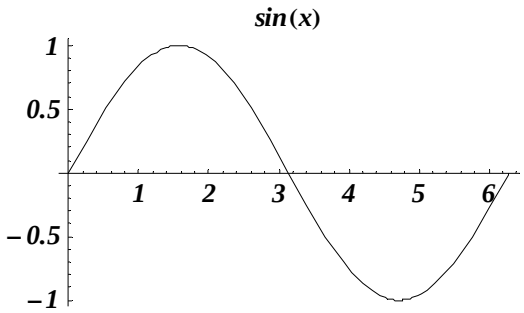


Out[2]= `-Graphics-`

3.7.11 Beschriftungen in Grafiken gestalten

Über die Option **TextStyle** lassen sich diverse Schriftmerkmale für die Texte in einem Grafikobjekt ändern, z.B.:

In[1]:=
```
Plot[Sin[x], {x,0,2 Pi}, PlotLabel -> "sin(x)",
 TextStyle -> {FontFamily -> "Times", FontSize -> 14,
               FontSlant -> "Italic", FontWeight -> "Bold"}]
```



Out[1]= `-Graphics-`

In einer **TextStyle**-Liste sind u.a. folgende Optionen erlaubt:

Option	(Beispiel-)werte
FontFamily	Times, Arial
FontSlant	Plain, Italic
FontWeight	Plain, Bold
FontSize	12
FontColor	RGBColor[0,0,1]

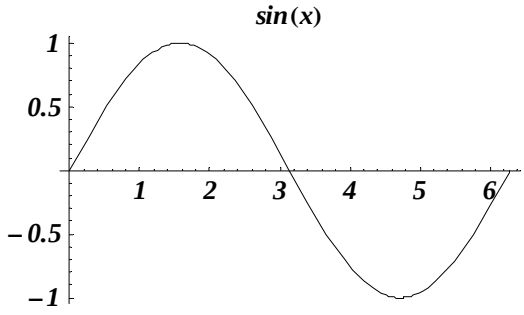
Als **FontFamily**-Wert sind PostScript-Schriften und lokal installierte Schriftarten erlaubt. Fehlt eine angeforderte Schriftart, kommt eine serifenfreie Standardschrift zum Einsatz.

Wird eine bestimmte **TextStyle**-Optionsliste wiederholt benötigt, sollte man sie der globalen Variablen **$TextStyle** zuweisen und damit zur Voreinstellung für spätere Plots machen, z.B.:

In[2]:=
```
$TextStyle={FontFamily -> "Times", FontSize -> 14,
 FontSlant -> "Italic", FontWeight -> "Bold"}
```

Out[2]=
```
{FontFamily → "Times", FontSize → 14,
 FontSlant → "Italic", FontWeight → "Bold"}
```

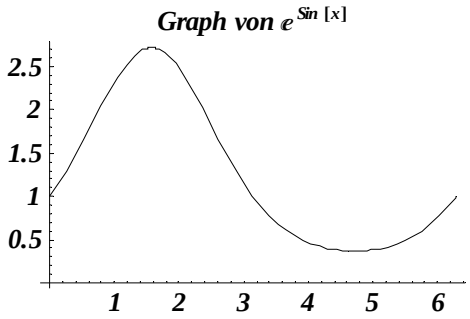
In[3]:=
```
Plot[Sin[x], {x,0,2 Pi}, PlotLabel -> "sin(x)"]
```



Out[3]= -Graphics-

In einer Beschriftung sind nicht nur Zeichenfolgen, sondern auch Mathematica-Ausdrücke erlaubt, die per Voreinstellung in **StandardForm** (vgl. Abschnitt 2.2.2) erscheinen, z.B.:

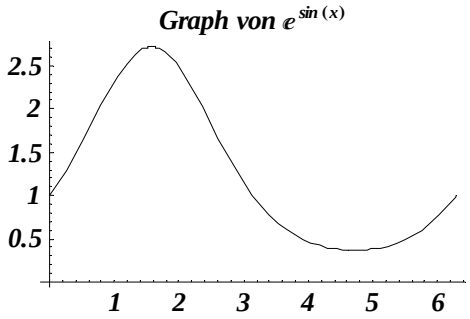
In[4]:=
```
Plot[Exp[Sin[x]],{x,0,2 Pi},PlotLabel->"Graph von" Exp[Sin[x]]]
```



Out[4]= -Graphics-

Mit der Option **FormatType** kann man auf **TraditionalForm** umschalten:

In[5]:=
```
Plot[Exp[Sin[x]],{x,0,2 Pi},PlotLabel->"Graph von" Exp[Sin[x]]
 FormatType -> TraditionalForm]
```



Out[5]= -Graphics-

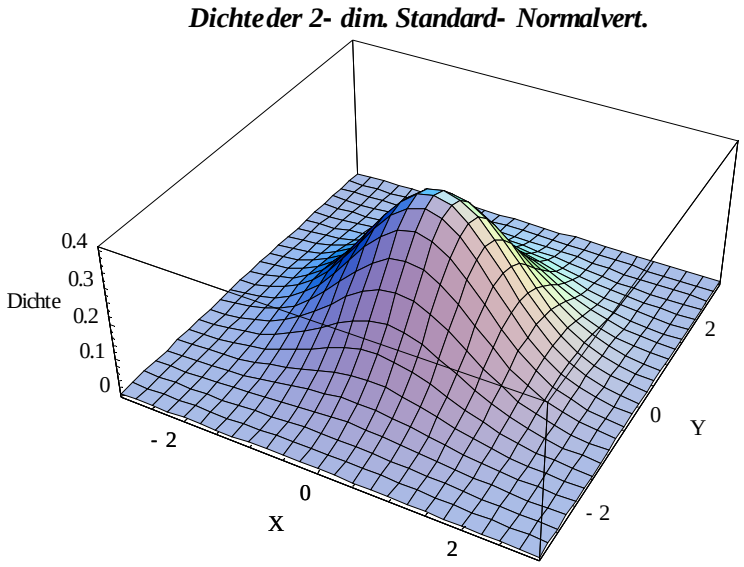
Über die globale Variable **$FormatType** ändert man die Voreinstellung für spätere Plots, z.B.:

In[6]:=
```
$FormatType = TraditionalForm
```

Out[6]= TraditionalForm

Mit Hilfe der Funktion **StyleForm** kann man Schriftattribute für *einzelne* Texte festlegen, z.B.

```
In[7]:= Plot3D[Exp[-(x^2 + y^2)/2]/Sqrt[2Pi], {x, -3, 3}, {y, -3, 3},
         PlotPoints -> 25, AxesLabel -> {"X", "Y", "Dichte"},
         PlotLabel->StyleForm["Dichte der 2-dim. Standard-Normalvert.",
         FontFamily->"Times",FontSlant->"Italic",FontWeight -> "Bold",
         FontSize -> 16]]
```

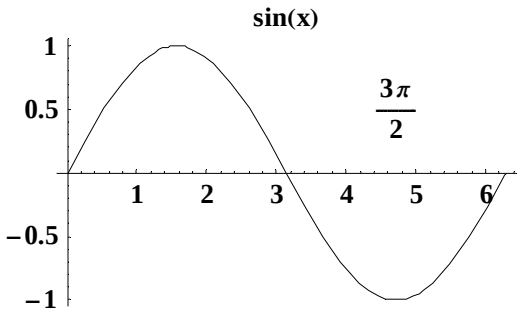



```
Out[7]= -Graphics-
```

Abschließend soll noch demonstriert werden, wie man Plots um freie Texte erweitert:

```
In[8]:= sinus = Show[sinus, Graphics[Text[3/ 2Pi, {4.71, 0.5}]]]

In[9]:= Show[sinus, Graphics[Text[3/2Pi, {4.71,0.5}]]]
```

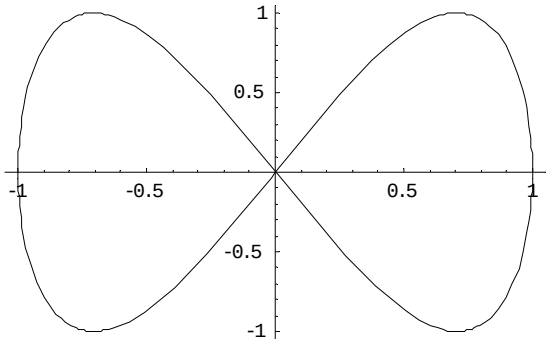



```
Out[9]= -Graphics-
```

3.7.12 Parametrische Diagramme

Bei zweidimensionalen parametrischen Diagrammen wird eine Kurve in der Ebene über zwei vom der selben Variablen abhängige Funktionen für die *x*- und die *y*-Koordinate beschrieben, z.B.:

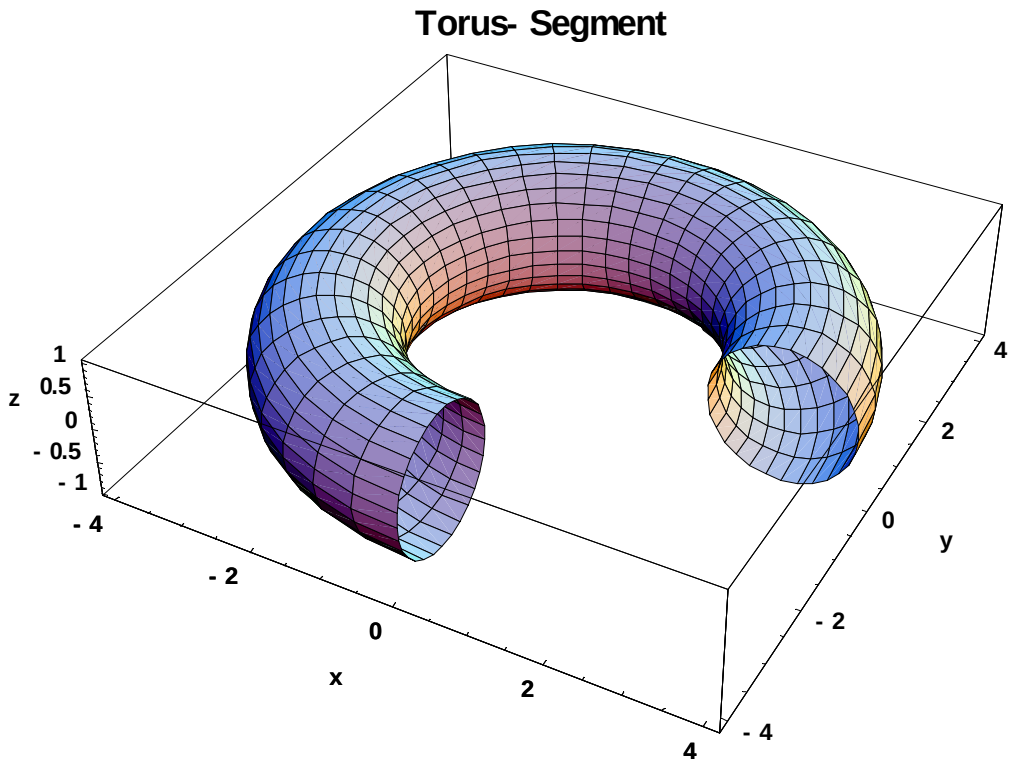
In[1]:=
```
ParametricPlot[{Sin[t], Sin[2t]}, {t,0,2Pi}]
```



Out[1]= `-Graphics-`

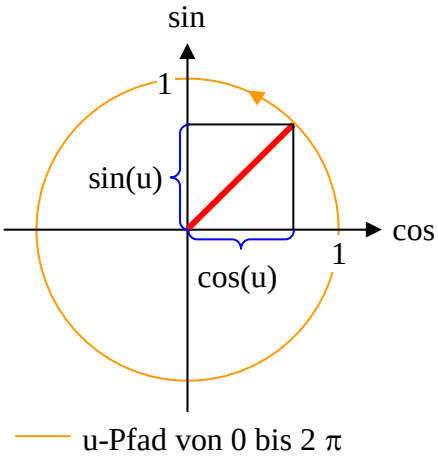
Besonders eindrucksvoll sind parametrische *3D*-Plots, z.B.:

In[2]:=
```
ParametricPlot3D[{Cos[t](3 + Cos[u]), Sin[t](3 + Cos[u]), Sin[u]},
 {t, 0, 3/2Pi}, {u, 0, 2Pi}, PlotPoints -> 30, AxesLabel -> {x, y, z},
 TextStyle->{FontFamily->"Helvetica",FontSize->14,FontWeight->"Bold"},
 PlotLabel -> StyleForm["Torus-Segment", FontSize -> 20]]
```



Out[2]= `-Graphics-`

Hier wird über *drei* Funktionen, die von *zwei* Variablen (im Beispiel: *u* und *t)* abhängen, eine Fläche im Raum beschrieben.
Im Beispiel durchläuft der Variable *u* das Intervall von 0 bis 2 π:



Für jeden festen Wert von *u* beschreiben die dann nur noch von *t* abhängigen Funktionen

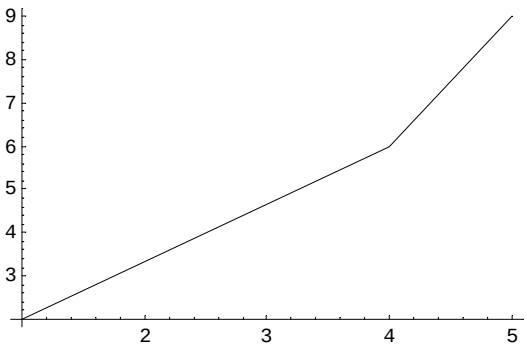
- cos(*t*) (3 + cos(*u*))
- sin(*t*) (3 + cos(*u*))

einen $\frac{3}{4}$-Kreis in der (*x*,*y*)-Ebene mit dem Radius 3 + cos(*u*), wenn *t* von 0 bis $\frac{3\pi}{2}$ verläuft. Als Höhenkoordinate hat diese Kurve den Wert sin(*u*).

3.7.13 Datenlisten zeichnen

Mathematica kann nicht nur Diagramme von Funktionen erstellen, sondern auch Datenlisten 2- oder 3-dimensional grafisch darstellen. Wir beschränken uns auf einfache Liniendiagramme, die mit dem Befehl **ListPlot** gezeichnet werden:

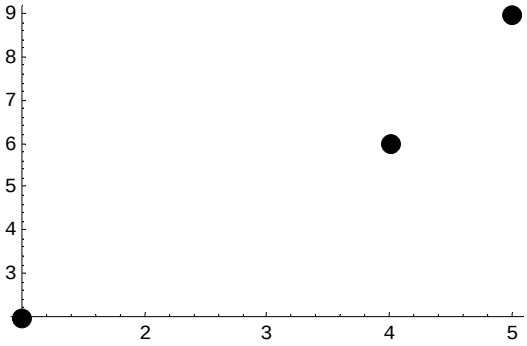
In[1]:=
```
ListPlot[{{1,2}, {4,6}, {5,9}}, PlotJoined -> True]
```



Out[1]=
```
-Graphics-
```

Um die vorgegebenen Punkte zu markieren, kann z.B. per **PlotStyle**-Option das Attribut **AbsolutePoint-Size** modifiziert werden:

```
In[2]:= ListPlot[{{1,2}, {4,6}, {5,9}},
         PlotStyle -> AbsolutePointSize[10]]
```



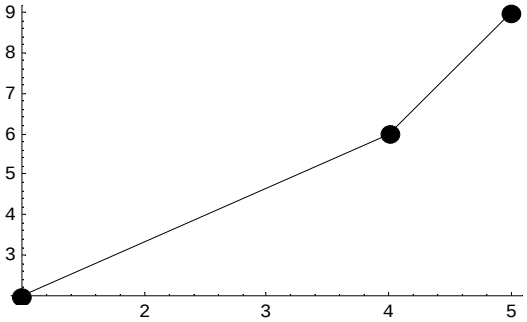

```
Out[2]= -Graphics-
```

Weil **`PlotJoined -> True`** die Ausgabe von Markierungen verhindert, muss man für eine gemeinsame Darstellung von Linien *und* Markierungen zwei Plots kombinieren, z.B.

```
In[3]:= p1 = ListPlot[{{1,2}, {4,6}, {5,9}}, PlotJoined -> True]
```

```
In[4]:= p2 = ListPlot[{{1,2},{4,6},{5,9}},
         PlotStyle -> {AbsolutePointSize[10]}]
```

```
In[5]:= Show[p1, p2]
```




```
Out[5]= -Graphics-
```

3.8 Numerische Mathematik

Die Behandlung der numerischen Mathematik ist in diesem Manuskript leider etwas zu kurz geraten, z.B. fehlt das Thema Interpolation komplett.

3.8.1 Exakte Ergebnisse und numerische Approximationen

Die im Abschnitt über analytische Methoden vorgestellte Funktion **Integrate** sucht nach einem *exakten* Ergebnis bestehend aus mathematischen Symbolen und rundungsfehlerfreien Zahlen. Kann eine solche Lösung nicht ermittelt werden, liefert sie eine mehr oder weniger hilfreiche Reformulierung des Problems. Um in diesem Fall eine *numerische* Lösung zu erzwingen, kann man im Argument-Ausdruck eine Zahl per Dezimaltrennzeichen als gerundet kennzeichnen, oder aber den **N**-Operator anwenden, z.B.:

```
In[1]:= Integrate[Exp[x^2/-2],{x,-Infinity,2}]
```

$$\text{Out[1]=}\quad \sqrt{\frac{\pi}{2}}\,\left(1+\text{Erf}\left[\sqrt{2}\right]\right)$$

```
In[2]:= N[%]
Out[2]= 2.4496
```

Die Funktion **NIntegrate** beschränkt sich demgegenüber von vornherein auf numerische Ergebnisse, z.B.:

```
In[3]:= NIntegrate[Exp[x^2/-2],{x,-Infinity,2}]
Out[3]= 2.4496
```

Analog gibt es zu den Funktionen **Solve**, **Sum**, **Product** usw. die numerischen Varianten **NSolve**, **NSum**, **NProduct** usw.

3.8.2 Numerische Lösung von Gleichungen

3.8.2.1 Polynom-Gleichungen

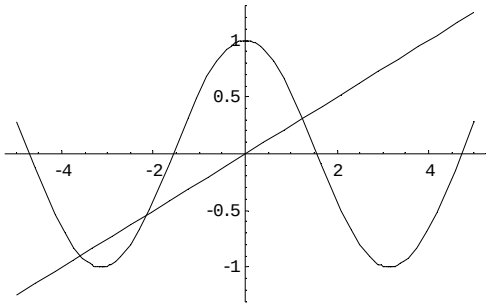
Der Funktion **Solve** zum Lösen von Polynom-Gleichungen entspricht die numerische Variante **NSolve**, die für jede Polynom-Gleichung alle numerischen Lösungen liefert, z.B. (vgl. Abschnitt 3.6.5.1):

```
In[1]:= Solve[x^6-x^5==6,x]
Out[1]= {{x→Root[-6-#1^5+#1^6&,1]},
         {x→Root[-6-#1^5+#1^6&,2]},
         {x→Root[-6-#1^5+#1^6&,3]},
         {x→Root[-6-#1^5+#1^6&,4]},
         {x→Root[-6-#1^5+#1^6&,5]},
         {x→Root[-6-#1^5+#1^6&,6]}}
In[2]:= N[%]
Out[2]= {{x→-1.22},{x→1.59015},
         {x→-0.537119-1.13651 i},{x→-0.537119+1.13651 i},
         {x→0.852045-1.10965 i},{x→0.852045+1.10965 i}}
In[3]:= NSolve[x^6-x^5==6,x]
Out[3]= {{x→-1.22},{x→-0.537119-1.13651 i},
         {x→-0.537119+1.13651 i},{x→0.852045-1.10965 i},
         {x→0.852045+1.10965 i},{x→1.59015}}
```

3.8.2.2 Transzendente Gleichungen

Da es bei transzendenten Funktionen keine systematische Methode zum Auffinden *aller* numerischen Lösungen gibt, führt der Operator **N** hier nicht viel weiter. Wir wollen in einem Beispiel die Schnittpunkte der beiden Funktionen cos(*x*) und *x*/4 suchen lassen:

In[1]:= **Plot[{Cos[x], x/4}, {x, -5, 5}]**



Out[1]= -Graphics-

In[2]:= **Solve[Cos[x]==x/4,x]**

```
Solve::tdep: The equations appear to involve the
   variables to be solved for in an essentially non-algebraic way.
```

Out[2]= $\text{Solve}\left[\text{Cos}[x] == \frac{x}{4}, x\right]$

In[3]:= **N[%]**

```
Solve::tdep: The equations appear to involve the
   variables to be solved for in an essentially non-algebraic way.
```

Out[3]= Solve[Cos[x] == 0.25 x, x]

Zum selben „Ergebnis“ kommt auch die Funktion **NSolve**, die bei Polynomgleichungen noch mit numerischen Lösungen dienen konnte (siehe Abschnitt 3.8.2.1):

In[4]:= **NSolve[Cos[x]==x/4,x]**

Out[4]= $\text{NSolve}\left[\text{Cos}[x] == \frac{x}{4}, x\right]$

In dieser Situation steht die Funktion **FindRoot** zur Verfügung, die in Abhängigkeit von einem *Startwert* nach einer *numerischen* Lösung sucht:

In[4]:= **FindRoot[Cos[x]==x/4, {x,1}]**

Out[4]= {x -> 1.25235}

Während sich **Solve** und **NSolve** bemühen, *alle* Lösungen einer Gleichung zu finden, liefert **FindRoot** nur *eine* Lösung, die vom gewählten Startwert des Verfahrens abhängt, z.B.:

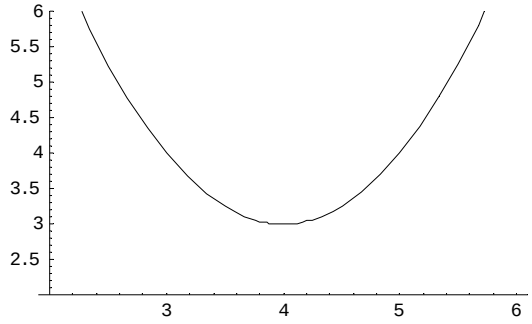
In[5]:= **FindRoot[Cos[x]==x/4, {x,-2}]**

Out[5]= {x -> -2.13333}

3.8.3 Numerische Optimierung

Zur numerischen Optimierung **ohne Nebenbedingungen** stehen die Funktionen **FindMinimum** und **FindMaximum** zur Verfügung, z.B.:

```
In[1]:= Plot[19 - 8*x + x^2, {x,2,6}, PlotRange->{2,6}]
```




```
Out[1]= -Graphics-
```

```
In[2]:= FindMinimum[19 - 8*x + x^2, {x,2}]
Out[2]= {3., {x -> 4.}}
```

Die untersuchte Funktion erreicht ihr Minimum 3 bei $x = 4$, das in diesem Fall offenbar ein globales Minimum ist. Im Allgemeinen finden **FindMinimum** und **FindMaximum** ein vom vorgegebenen Startwert abhängiges *lokales* Extremum, falls ein solches existiert.

Die Syntax:

FindMinimum[*Funktion*, {*x*, *Startwert*}]	Für die von *x* abhängige *Funktion* wird ab *Startwert* ein lokales Minimum gesucht.

Mit den Methoden der **linearen Programmierung** kann für eine lineare Zielfunktion auf einem durch lineare Nebenbedingungs-Ungleichungen begrenzten Gebiet das globale Minimum bzw. Maximum bestimmt werden. In Mathematica werden diese Methoden durch die Funktionen **NMinimize** bzw. **NMaximize**[2] implementiert, was anhand eines landwirtschaftlichen Beispiels demonstriert werden soll (vgl. Hettich 1989).
Auf einem Bauernhof mit Kühen und Schafen sind folgende Ressourcen vorhanden:

Kuhstall	für max. 50 Kühe
Schafstall	für max. 200 Schafe
Weide	72 Morgen
Arbeitszeit	10000 Stunden

Pro Tier werden folgende Ressourcen verbraucht:

	Kuh	Schaf
Platz im Kuhstall	1	0
Platz im Schafstall	0	1
Weide	1	0,2
Arbeitszeit	150	25

Pro Tier werden folgende Gewinne gemacht:

Kuh	250

[2] Die Funktionen **NMaximize** und **NMinimize** sind neu in Mathematica 5. In den älteren Versionen wird die lineare Programmierung von den Funktionen **ConstrainedMin** und **ConstrainedMax** unterstützt.

Schaf	45

Wenn x für die Anzahl gehaltener Kühe und y für die Anzahl gehaltener Schafe steht, dann ist die folgende Zielfunktion zu maximieren:

$$250\,x + 45\,y$$

Als Nebenbedingungen sind einzuhalten:

$$x \le 50$$
$$y \le 200$$
$$x + 0{,}2\,y \le 72$$
$$150\,x + 25\,y \le 10000$$

Mit **NMaximize** stellt sich heraus, dass der maximale Gewinn von 17200 mit 40 Kühen und 160 Schafen erzielt wird:

```
In[3]:= NMaximize[{250x+45y,{x<=50,y<=200,x+0.2y<=72,150x+25y<=10000}},
          {x, y}]
Out[3]= {17200., {x -> 40., y -> 160.}}
```

3.9 Funktionen und Programme

Einige Hinweise vorab:

- In späteren Abschnitten folgen noch Themen von hoher Relevanz für das Programmieren mit Mathematica (z.B. Lesen und Schreiben von externen Dateien).
- Wesentliche Möglichkeiten (z.B. das objektorientierte Programmieren mit Mathematica), können in diesem Manuskript nicht behandelt werden.

3.9.1 Funktionen definieren und verwenden

In folgendem Beispiel wird die Funktion *f* definiert, welche zu jedem Argument das Quadrat liefert:

```
In[1]:= f[x_]:=x^2
```

Mathematica erzeugt *keine* Ausgabezelle zur Bestätigung der Definition.
Bei der Syntax ist zu beachten:

- Auf der **linken** Seite der Definitionsgleichung wird an das Argumentensymbol ein Unterstrich angehängt.
- Es ist das in der Mathematik übliche *Definitionszeichen* (":=") zu verwenden.
- Die benutzerdefinierte Funktionsbezeichnung sollte mit einem Kleinbuchstaben beginnen, um Verwechslungen mit eingebauten Mathematica-Funktionen zu vermeiden.

Selbst definierte Funktionen verarbeiten sowohl numerische als auch symbolische Argumente, z.B.:

```
In[2]:= f[2]
Out[2]= 4

In[3]:= f[3x+x^2]
Out[3]= (3 x + x^2)^2
```

Sie können sich jederzeit über die aktuelle Definition eines Symbols informieren:

```
In[4]:= ?f
        Global`f
        f[x_] := x^2
```

Mit der folgenden Anweisung wird die Funktionsdefinition für *f* wieder gelöscht:

```
In[5]:= Clear[f]
```

Wie bei der Definition einer Funktion gibt Mathematica auch beim Löschen *keine* Bestätigung.

Das **Clear**-Kommando löscht auch die zugewiesenen Werte von Variablen. Ebenso wie überflüssig gewordene Variablen-Definitionen sollten auch obsolete Funktions-Definitionen sofort gelöscht werden, damit sie nicht später für Verwirrung sorgen.
Clear arbeitet weniger radikal als der verwandte Befehl **Remove**, den wir im Zusammenhang mit dem Laden von Paketen kennen gelernt haben:

- **Clear** löscht den Wert bzw. die Definition eines Symbols. Jedoch bleibt das Symbol samt Attributen erhalten.
- **Remove** beseitigt ein Symbol vollständig.

Selbstverständlich sind Funktionsdefinitionen mit *mehreren Argumenten* erlaubt. Ferner darf man in einer Definition auch *mehrere Ausdrücke* verwenden, wobei folgende Regeln zu beachten sind:

- Die Ausdrücke werden durch Strichpunkte getrennt und insgesamt mit runden Klammern umgeben.
- Der letzte Ausdruck liefert den Funktionswert.

Damit lassen sich unübersichtliche Verschachtelungen von Funktionsaufrufen vermeiden, z.B.:

```
In[6]:= kk[n_,k_]:=(t=Product[x+i,{i, 1, n}];Coefficient[t,x^k])
```

Diese Funktion ermittelt zum folgenden Polynom in *x*

$$\prod_{i=1}^{n}(x+i)$$

mit Hilfe der Mathematica-Funktion **Coefficient** den Koeffizienten von x^k, z.B.:

```
In[7]:= kk[3,2]
Out[7]= 6
```

3.9.2 Module

Wie das Beispiel `kk` aus dem letzten Abschnitt zeigt, können benutzerdefinierte Funktionen als Funktions-*Unterprogramme* aufgefasst werden. Es ist nun wünschenswert, intern verwendete Variablen (im Beispiel: `t`, `x`, `i`) so zu kapseln, dass unerwünschte Beziehungen zur Außenwelt ausgeschlossen sind. Dazu bietet Mathematica die Möglichkeit, ein **Modul** zu erstellen, z.B.:

```
In[1]:= mkk[n_,k_]:=Module[{u,x,i},u=Product[x+i,{i,1,n}];Coefficient[u,x^k]]
```

`kk` und `mkk` verhalten sich identisch:

```
In[2]:= mkk[3,2]
Out[2]= 6
```

Die in der `mkk`-Definition verwendeten Variablen (`u`, `x`, `i`) sind aber im Unterschied zu den Variablen in der `kk`-Definition, nicht *global*, d.h. außerhalb der Funktion nicht bekannt, z.B.:

```
In[3]:= t
Out[3]= (1 + x) (2 + x) (3 + x)
In[4]:= u
Out[4]= u
```

Die **Module**-Syntax:

Module[{*lokaleVar1*, *lokaleVar2*, ...}, *Ausdruck1*; *Ausdruck2*; ...]

3.9.3 Bedingte Anweisungen

Mit dem Konditionaloperator **If** lässt sich die bedingungsabhängige Entscheidung zwischen zwei Ausdrücken formulieren, z.B.:

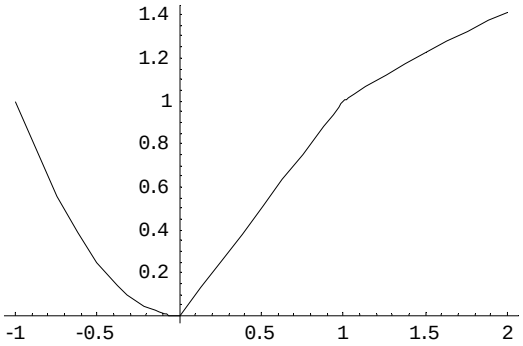
```
In[1]:= x=-1; y=3
Out[1]= 3
In[1]:= If[x>=0, x, y]
Out[1]= 3
```

Die allgemeine **If**-Syntax:

If[*Ausdruck*, *Dann*, *Sonst*]	Ist der *Ausdruck* wahr, wird *Dann* ausgewertet, anderenfalls *Sonst*.

Bei Fallunterscheidungen mit mehr als zwei Alternativen ist die Funktion **Which** zu bevorzugen, z.B.:

```
In[2]:= f[x_] := Which[x < 0, x^2, x < 1, x, x >= 1, Sqrt[x]]
In[3]:= Plot[f[x],{x,-1,2}]
```




```
Out[3]= -Graphics-
```

Die allgemeine Syntax der **Which**-Funktion:

Which[*Test1*, *Ausdruck1*, *Test2*, *Ausdruck1*, ...]	Mathematica ermittelt den ersten wahren *Test* und wertet den zugehörigen *Ausdruck* aus.

3.9.4 Schleifen

In folgendem Beispiel wird mit Hilfe der **Do**-Schleife eine Funktion definiert, welche die Summe der quadrierten natürlichen Zahlen von 1 bis *n* ausgibt:

```
In[1]:= sumq1n[n_]:=Module[{s,i}, s=0; Do[s=s+i^2,{i,1,n}]; s]
In[2]:= sumq1n[10]
Out[2]= 385
```

Die Syntax der **Do**-Schleife:

Do[*Ausdruck*, {*indexVar*, *Min*, *Max*, *Dist*}]	Der *Ausdruck* wird für *indexVar* = *Min* bis *Max* bei Schrittweite *Dist* ausgewertet.

Eine Schrittweite von Eins muss nicht angegeben werden

3.9.5 Werte ausgeben

Die **Print**-Funktion dient dazu, die Werte von mehreren Ausdrücken (optional durch Leerzeichen getrennt) in einer Zeile auszugeben. In folgendem Beispiel geschieht dies innerhalb einer **Do**-Schleife:

```
In[1]:= Do[Print[n, "  ", Log[n]/5.0], {n,1,5}]
        1  0
        2  0.138629
        3  0.219722
        4  0.277259
        5  0.321888
```

Auf jede **Print**-Ausgabe folgt automatisch ein Zeilenwechsel.

Die **Print**-Syntax:

Print[*Ausdruck1*; *Ausdruck2*; ...]

3.10 Listen und Matrizen

Mathematica-Listen stellen Verallgemeinerungen zu mehreren Standardbegriffen der Mathematik und der Informatik dar. In diesem Abschnitt lernen Sie einige wichtige Operationen mit Listen bzw. Matrizen kennen. Weiterführende Informationen finden sich im Help Browser unter

Build-in Functions > Lists and Matrices.

3.10.1 Methoden der linearen Algebra

3.10.1.1 Vektoren und Matrizen als (formatierte) Listen

Vektoren und Matrizen werden in Mathematica durch Listen repräsentiert, z.B.:

```
In[1]:= v1={x, y}
Out[1]= {x, y}

In[2]:= m1={{a, b}, {c, d}}
Out[2]= {{a, b}, {c, d}}
```

Per **MatrixForm** erhält man die vertrauten Darstellungen:

```
In[3]:= MatrixForm[v1]
Out[3] ]//MatrixForm=
```

$$\begin{pmatrix} x \\ y \end{pmatrix}$$

```
In[4]:= MatrixForm[m1]
Out[4]//MatrixForm=
```

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

Achtung: Wenden Sie **MatrixForm** *nicht* zusammen mit Wertzuweisungen an, weil mit den entsprechend formatierte Variablen keine Operationen ausgeführt werden, z.B.:

```
In[5]:= mf = MatrixForm[{{a, b}, {c, d}}]
Out[5]//MatrixForm=
```

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

```
In[6]:= vf = MatrixForm[{x, y}]
Out[6] ]//MatrixForm=
```

$$\begin{pmatrix} x \\ y \end{pmatrix}$$

Nach diesen Definitionen bewirkt der Matrixmultiplikations-Operator (siehe Abschnitt 3.10.1.3) zwar eine nette Ausgabe, aber keine Rechnung:

```
In[7]:= mf . vf
Out[7]=
```

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} . \begin{pmatrix} x \\ y \end{pmatrix}$$

Die *i*-te Unterliste einer zweidimensionalen Liste ist natürlich eine eindimensionale Liste, z.B.:

```
In[8]:= m1[[1]]
Out[8]= {a, b}
```

Um einzelne Elemente einer zweidimensionalen Liste (bzw. Matrix) anzusprechen, muss man doppelt indizieren:

```
In[9]:= m1[[2,1]]
Out[9]= c
```

3.10.1.2 Operationen mit Skalaren

Die Operationen

Matrix · Skalar

bzw.

Matrix + Skalar

werden elementweise ausgeführt, z.B.:

In[1]:=
```
MatrixForm[3 m1 + 1]
```

Out[1] //MatrixForm=

$$\begin{pmatrix} 1+3a & 1+3b \\ 1+3c & 1+3d \end{pmatrix}$$

Der Stern als Operatorzeichen für die skalare Multiplikation kann wie üblich weggelassen werden.

3.10.1.3 Matrixprodukt

Mathematica berechnet das Produkt zweier Matrizen nach den üblichen Regeln, d.h. das Element (*i*, *j*) der Produktmatrix ist gerade das innere Produkt aus der *i*-ten Zeile der ersten Matrix mit der *j*-ten Spalte der zweiten Matrix, z.B.:

In[1]:=
```
m2= {{1, 2}, {3, 4}}
```

Out[1]=
```
{{1,2},{3,4}}
```

In[2]:=
```
MatrixForm[m1.m2]
```

Out[2]//MatrixForm=

$$\begin{pmatrix} a+3b & 2a+4b \\ c+3d & 2c+4d \end{pmatrix}$$

Als Operationszeichen für das Matrixprodukt ist ein **Punkt** zu verwenden. Setzt man den „Multiplikations-Stern“ oder gar kein Operatorzeichen, führt Mathematica eine skalare (elementweise) Multiplikation aus.

In folgendem Beispiel wird die Matrix `m1` mit dem Vektor `v1` nachmultipliziert:

In[3]:=
```
MatrixForm[m1.v1]
```

Out[3]//MatrixForm=

$$\begin{pmatrix} a\,x+b\,y \\ c\,x+d\,y \end{pmatrix}$$

Beim Produkt eines Vektors mit einer Matrix wird der Vektor situationsadäquat als Zeile oder Spalte behandelt, z.B.:

In[4]:=
```
MatrixForm[v1.m1]
```

Out[4]//MatrixForm=

$$\begin{pmatrix} a\,x+b\,y \\ c\,x+d\,y \end{pmatrix}$$

Ein Vektor ist in Mathematica weder Zeile noch Spalte, sondern *Liste*. Angewandt auf zwei Vektoren liefert daher die Punkt-Operation unabhängig von der Reihenfolge der Argumente stets das *innere* Produkt, z.B.:

In[5]:= **v2={1,2}**

Out[5]= {1, 2}

In[6]:= **v1 . v2**

Out[6]= x + 2y

In[7]:= **v2 . v1**

Out[7]= x + 2y

Natürlich gilt diese „Kommutativität“ keinesfalls generell bei der Matrix-Multiplikation.

Wenn Sie für zwei Vektoren das *äußere* Produkt benötigen (Kronecker-Produkt), z.B.:

$$\begin{pmatrix} x \\ y \end{pmatrix} \begin{pmatrix} 1 & 2 \end{pmatrix} = \begin{pmatrix} x & 2x \\ y & 2y \end{pmatrix}$$

müssen Sie die **Outer**-Funktion verwenden:

In[6]:= **MatrixForm[Outer[Times, v1, v2]]**

Out[6]//MatrixForm=

$$\begin{pmatrix} x & 2\,x \\ y & 2\,y \end{pmatrix}$$

Natürlich kann Mathematica auch das Kronecker-Produkt zweier *Matrizen* ausrechnen:

In[9]:= **MatrixForm[Outer[Times, m1, m2]]**

Out[9]//MatrixForm=

$$\begin{pmatrix} \begin{pmatrix} a & 2\,a \\ 3\,a & 4\,a \end{pmatrix} & \begin{pmatrix} b & 2\,b \\ 3\,b & 4\,b \end{pmatrix} \\ \begin{pmatrix} c & 2\,c \\ 3\,c & 4\,c \end{pmatrix} & \begin{pmatrix} d & 2\,d \\ 3\,d & 4\,d \end{pmatrix} \end{pmatrix}$$

3.10.1.4 Transponierte Matrix

Die Transponierte zur Matrix `m1` erhält man mit:

In[1]:= **MatrixForm[Transpose[m1]]**

Out[1]//MatrixForm=

$$\begin{pmatrix} a & c \\ b & d \end{pmatrix}$$

3.10.1.5 Determinante einer Matrix

Die Funktion **Det** liefert die Determinante einer quadratischen Matrix, z.B.:

In[1]:= **Det[m1]**

Out[1]= -bc + ad

3.10.1.6 Inverse Matrix

Zum Invertieren von quadratischen Matrizen bietet Mathematica die Funktion **Inverse**, z.B.:

In[1]:=
```
MatrixForm[Inverse[m1]]
```

Out[1]//MatrixForm=

$$\begin{pmatrix} \frac{d}{-bc+ad} & -\frac{b}{-bc+ad} \\ -\frac{c}{-bc+ad} & \frac{a}{-bc+ad} \end{pmatrix}$$

3.10.1.7 Eigenwerte einer Matrix

Die Funktion **Eigenvalues** liefert die Eigenwerte einer quadratischen Matrix, z.B.:

In[1]:=
```
Eigenvalues[m1]
```

Out[1]=

$$\left\{\frac{1}{2}\left(a+d-\sqrt{a^2+4bc-2ad+d^2}\right),\ \frac{1}{2}\left(a+d+\sqrt{a^2+4bc-2ad+d^2}\right)\right\}$$

Wir wollen das Ergebnis verifizieren, um unseres Grundwissens in linearer Algebra aufzufrischen:

$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \neq \begin{pmatrix} 0 \\ 0 \end{pmatrix}$ ist ein *Eigenvektor* von $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$ zum *Eigenwert* λ falls gilt:

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \lambda \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \Leftrightarrow \begin{pmatrix} a & b \\ c & d \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} - \lambda \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \Leftrightarrow \begin{pmatrix} a-\lambda & b \\ c & d-\lambda \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

Wenn eine Matrix von $\begin{pmatrix} 0 \\ 0 \end{pmatrix}$ verschiedene Vektoren auf den Nullvektor abbildet, muss ihre Determinante gleich Null ist:

$$\det\begin{pmatrix} a-\lambda & b \\ c & d-\lambda \end{pmatrix} = 0 \Leftrightarrow (a-\lambda)(d-\lambda) - bc = 0 \Leftrightarrow ad - bc - (a+d)\lambda + \lambda^2 = 0$$

Durch Lösen der quadratischen Gleichung erhält man die Eigenwerte. Diese Aufgabe überlassen wir Mathematica:

In[2]:=
```
Solve[a d - b c - (a + d) lambda + lambda^2 == 0, lambda]
```

Out[2]=

$$\left\{\left\{\text{lambda} \to \frac{1}{2}\left(a+d-\sqrt{a^2+4bc-2ad+d^2}\right)\right\},\ \left\{\text{lambda} \to \frac{1}{2}\left(a+d+\sqrt{a^2+4bc-2ad+d^2}\right)\right\}\right\}$$

3.10.1.8 Eigenvektoren einer Matrix

Mit **Eigenvectors** können wir schließlich auch noch Eigenvektoren zu unserer Matrix `m1` berechnen lassen:

In[1]:=
```
Eigenvectors[m1]
```

Out[1]=
$$\left\{\left\{-\frac{-a+d+\sqrt{a^2+4bc-2ad+d^2}}{2c},\ 1\right\},\ \left\{-\frac{-a+d-\sqrt{a^2+4bc-2ad+d^2}}{2c},\ 1\right\}\right\}$$

Wir werden diese Lösung in Abschnitt 3.10.1.9 verifizieren.

3.10.1.9 Lösung linearer Gleichungssysteme in Matrixschreibweise

Mit Hilfe der Mathematica-Funktion **LinearSolve** lässt sich bequem der Lösungsvektor *x* zu einer Matrix-Gleichung

$$m \,.\, x == b$$

ermitteln. Zunächst soll ein **inhomogenes** Gleichungssystem mit numerischen Koeffizienten gelöst werden:

In[1]:=
```
km = {{1, -3, 1}, {1, 0, 0}, {0, 0, 1}}
```

Out[1]=
```
{{1, -3, 1}, {1, 0, 0}, {0, 0, 1}}
```

In[2]:=
```
MatrixForm[km]
```

Out[2]//MatrixForm=
$$\begin{pmatrix} 1 & -3 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

In[3]:=
```
b = {1,5,-1}
```

Out[3]=
```
{1, 5, -1}
```

In[4]:=
```
LinearSolve[km, b]
```

Out[4]=
```
{5, 1, -1}
```

Nun soll noch ein **homogenes** Gleichungssystem gelöst werden. Als Beispiel greifen wir das Eigenvektor-Problem aus Abschnitt 3.10.1.8 auf und prüfen nach, ob die Funktion **Eigenvectors** zum ersten Eigenwert der dortigen Matrix `m1` einen korrekten Eigenvektor geliefert hat.

```
In[5]:= a=.; b=.; m1={{a, b}, {c, d}}
Out[5]= {{a, b}, {c, d}}

In[6]:= lambda = Eigenvalues[m1]
```

$$\text{Out[6]}= \left\{\frac{1}{2}\left(a+d-\sqrt{a^2+4bc-2ad+d^2}\right), \frac{1}{2}\left(a+d+\sqrt{a^2+4bc-2ad+d^2}\right)\right\}$$

```
In[7]:= km = m1- lambda[[1]] IdentityMatrix[2]
```

$$\text{Out[7]}= \left\{\left\{a+\frac{1}{2}\left(-a-d+\sqrt{a^2+4bc-2ad+d^2}\right), b\right\}, \left\{c, d+\frac{1}{2}\left(-a-d+\sqrt{a^2+4bc-2ad+d^2}\right)\right\}\right\}$$

```
In[8]:= MatrixForm[km]
```

Out[8]//MatrixForm=

$$\begin{pmatrix} a+\frac{1}{2}\left(-a-d+\sqrt{a^2+4bc-2ad+d^2}\right) & b \\ c & d+\frac{1}{2}\left(-a-d+\sqrt{a^2+4bc-2ad+d^2}\right) \end{pmatrix}$$

```
In[9]:= NullSpace[km]
```

$$\text{Out[9]}= \left\{\left\{-\frac{-a+d+\sqrt{a^2+4bc-2ad+d^2}}{2c}, 1\right\}\right\}$$

Zur Lösung des homogenen Gleichungssystems wird hier die Funktion **NullSpace** benutzt, welche eine Liste mit Basisvektoren zum Kern der Matrix `km` liefert.

3.10.2 Listen generieren

Bei einer zur Funktion **Do** (siehe Abschnitt 3.9.4) analogen Syntax ermöglicht es die Funktion **Table**, eine Liste mit Ausdrücken zu erzeugen, z.B.:

```
In[1]:= xpot = Table[x^i, {i,0,5}]
Out[1]= {1, x, x^2, x^3, x^4, x^5}

In[2]:= xpot[[3]]
Out[2]= x^2
```

Hinweis: Sollten Sie statt *Out[1]* eine andere Ausgabe erhalten haben, liegt dies vermutlich an einer noch bestehenden Definition des Symbols *x*, die Sie mit „`x = .`“ aufheben können.

Unter Verwendung der üblich Mathematica-Laufindex-Syntax können mit **Table** auch **mehrdimensionale Tabellen** erstellt werden, z.B. solche Listen von Listen:

```
In[3]:= xpot2=Table[x^(i+j), {i,1,4}, {j,1,4}]
Out[3]= {{x^2, x^3, x^4, x^5}, {x^3, x^4, x^5, x^6}, {x^4, x^5, x^6, x^7}, {x^5, x^6, x^7, x^8}}
```

Mit Hilfe der Funktion **TableForm** bringt man Listen in Tabellenform:

In[4]:= **TableForm[xpot2]**

Out[4]//TableForm=

x^2	x^3	x^4	x^5
x^3	x^4	x^5	x^6
x^4	x^5	x^6	x^7
x^5	x^6	x^7	x^8

Über die Funktion **Array** erhält man eine Liste mit *indizierten* Variablen, z.B.:

In[5]:= **Array[b,4]**

Out[5]= {b[1], b[2], b[3], b[4]}

In[6]:= **MatrixForm[Array[m23,{2,3}]]**

Out[6]//MatrixForm=

$$\begin{pmatrix} m23[1,1] & m23[1,2] & m23[1,3] \\ m23[2,1] & m23[2,2] & m23[2,3] \end{pmatrix}$$

Die *n*-te Einheitsmatrix erhält man mit der Funktion **IdentityMatrix**, z.B.:

In[7]:= **MatrixForm[IdentityMatrix[3]]**

Out[7]//MatrixForm=

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

Über die Funktion **DiagonalMatrix** erzeugt man aus einer Liste mit Diagonalelementen die zugehörige Diagonalmatrix, z.B.:

In[8]:= **MatrixForm[DiagonalMatrix[{1,2,3}]]**

Out[8] //MatrixForm=

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{pmatrix}$$

3.10.3 Weitere Operationen mit Listen

Mathematica erlaubt noch zahlreiche weitere Operationen mit Listen, z.B.:

- Zugriff auf bestimmte Elemente einer Liste, z.B. auf das letzte Element, auf die ersten *n* Elemente
- Suche nach der Position eines Ausdrucks in einer Liste
- Einfügen, Anhängen oder Löschen von Listenelementen
- Elemente umordnen oder permutieren

3.11 Verwendung externer Dateien

3.11.1 Ergebnisse und Definitionen in externe Dateien schreiben

Selbstverständlich sind Notebook-Dateien (Namenserweiterung **.nb**) sehr gut zum Speichern von Mathematica–Ausdrücken und –Ergebnissen geeignet. In diesem Abschnitt geht es um Optionen zum Abspeichern *einzelner* Ausgaben oder Definitionen in Textdateien, z.B. im Rahmen von Mathematica-Programmen.
Per Voreinstellung werden dabei Mathematica-Ausdrücke geschrieben, die nach einem Reimport wieder also solche zu verwenden sind. Es stehen aber auch alternative Formate zur Verfügung.

3.11.1.1 Ausgabe im Mathematica-Format

3.11.1.1.1 Umleitung von Ergebnissen

In folgendem Beispiel wird eine mit der Funktion **Table** erzeugte Liste mit Hilfe des Umleitungsoperators ">>" in eine externe Datei geschrieben:

```
In[1]:= Table[x y,{x,1,4},{y,1,4}]>>"U:\\Eigene Dateien\\Mathe\\tab1"
```

Die Datei **tab1** hat anschließend folgenden Inhalt:

```
{{1, 2, 3, 4}, {2, 4, 6, 8}, {3, 6, 9, 12}, {4, 8, 12, 16}}
```

Hinsichtlich der Dateispezifikation ist bei der **Windows**-Version von Mathematica zu beachten:

- Wenn die Dateispezifikation Leerzeichen, eine Laufwerksangabe oder eine Pfadangabe enthält, muss sie mit doppelten Anführungszeichen (") begrenzt werden. Bei einfachen Dateinamen ohne Leerzeichen ist dies nicht nötig, z.B.:

  ```
  In[1]:= Table[x y,{x,1,4},{y,1,4}]>>tab1
  ```

  Dann wird das aktuelle Arbeitsverzeichnis verwendet.
  Mit dem folgenden Mathematica-Befehl lässt sich das aktuelle Arbeitsverzeichnis einstellen:

  ```
  In[2]:= SetDirectory["U:\\Eigene Dateien\\Mathe"]
  ```

- Die umgekehrten Schrägstriche in den Windows-Pfadangaben müssen verdoppelt werden.

Durch den oben erklärten Umleitungsoperator ">>" wird eine bereits vorhandene Ausgabedatei vor dem Schreiben zurückgesetzt, so dass der vorherige Inhalt verloren geht. Soll die Ausgabedatei *erweitert* werden, ist der Umleitungsoperator ">>>" zu verwenden, z.B.:

```
In[3]:= Table[x y,{x,1,4},{y,1,4}]>>>tab1
```

Natürlich ist die Ausgabeumleitung nicht auf die Funktion **Table** beschränkt, sondern klappt mit beliebigen Mathematica-Ausdrücken. Die allgemeine Syntax lautet:

Ausdruck >> *Dateispezifikation*	Das Ergebnis des Ausdrucks wird in die Datei geschrieben, wobei ein vorhandener Inhalt ggf. ersetzt wird.
Ausdruck >>> *Dateispezifikation*	Das Ergebnis des Ausdrucks wird am Ende der Datei angehängt, so dass vorhandene Zeilen erhalten bleiben.

3.11.1.1.2 Sichern von Definitionen

Mit den eben beschriebenen Umleitungsoperatoren ">>" und ">>>" kann man Mathematica-*Ergebnisse* bequem abspeichern. Variablen- oder Funktions*definitionen* sollten hingegen mit der Mathematica-Funktion **Save** gesichert werden, weil man die Definitionen dann später besonders leicht reaktivieren kann. In folgendem Beispiel

```
In[1]:= t = Table[x y,{x,1,4},{y,1,4}]
Out[1]= {{1, 2, 3, 4}, {2, 4, 6, 8}, {3, 6, 9, 12}, {4, 8, 12, 16}}

In[2]:= Save["t_def",t]
```

wird die Definition der Variablen `t` in die externe Datei **t_def** geschrieben, die anschließend folgenden Inhalt hat:

```
t = {{1, 2, 3, 4}, {2, 4, 6, 8}, {3, 6, 9, 12}, {4, 8, 12, 16}}
```

Bezüglich der Dateispezifikation gelten im Wesentlichen die Ausführungen aus dem vorherigen Abschnitt, wobei der Dateiname in der **Save**-Funktion allerdings stets in doppelte Hochkommata eingeschlossen werden muss. Weil im Beispiel auf Laufwerks- und Pfadangaben verzichtet wird, landet die Datei **t_def** also im momentan eingestellten Arbeitsverzeichnis.

Mit Hilfe des Umleitungsoperators "<<" kann eine so gespeicherte Definition später leicht reaktiviert werden, z.B.:

```
In[3]:= << t_def
Out[3]= {{1, 2, 3, 4}, {2, 4, 6, 8}, {3, 6, 9, 12}, {4, 8, 12, 16}}
```

Die allgemeine Syntax der Funktion **Save** lautet:

Save["*Dateispezifikation*", *f*, *g*, ...]	Die Definitionen der Variablen und Funktionen landen am Ende der Ausgabedatei, vorhandener Inhalt bleibt also erhalten.

3.11.1.2 Ausgabe im Standardformat

Bei allen bisher vorgestellten Ausgabemethoden werden Mathematica-Ausdrücke gesichert, die nach einem Reimport wieder als solche zu verwenden sind. Gelegentlich möchte man aber z.B. eine zweidimensionale Liste in einem einfachen Textformat ausgeben (ohne Mathematica-Syntaxelemente), z.B. zur Weiterverarbeitung mit einem anderen Programm. Dies ermöglicht die Funktion **OutputForm**, z.B.:

```
In[1]:= t=Table[x y,{x,1,4},{y,1,4}]
Out[1]:= {{1, 2, 3, 4}, {2, 4, 6, 8}, {3, 6, 9, 12}, {4, 8, 12, 16}}

In[2]:= OutputForm[MatrixForm[t]]>>tab2
```

Anschließend hat die Datei **tab2** folgenden Inhalt:

1	2	3	4
2	4	6	8
3	6	9	12
4	8	12	16

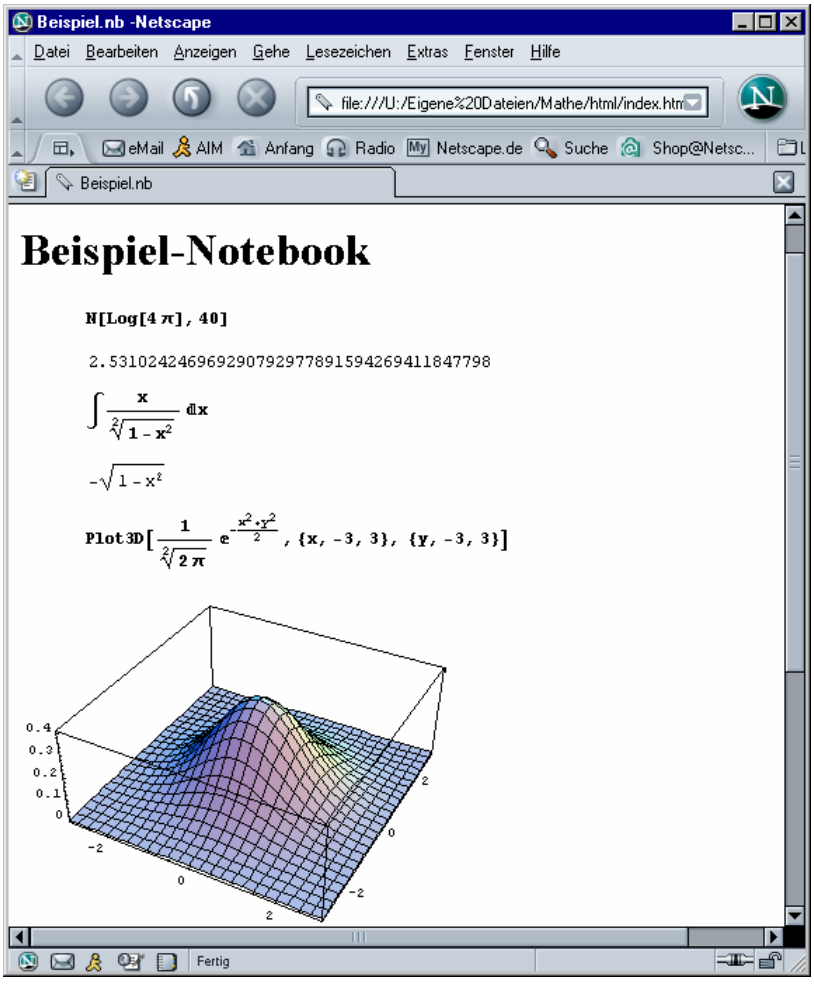
Im Hinblick auf spätere Abschnitte löschen wir die Definition der Variablen `t` wieder:

In[3]= `t=.`

3.11.1.3 Weitere Export- bzw. Publikationsoptionen

Mathematica beherrscht noch zahlreiche weitere Exportoptionen, z.B.:

- Der Export von Grafiken wurde schon in Abschnitt 3.7.4 behandelt.
- Mit dem Menübefehl **File > Save As Special > TeX** (bzw. mit der Funktion **TeXSave**) kann der Inhalt des aktiven Notizbuchs im TeX-Format gesichert werden.
- Mit dem Menübefehl **File > Save As Special > HTML** (bzw. mit der Funktion **HTMLSave**) kann Inhalt des aktiven Notizbuchs im HTML-Format gesichert werden.
  Die HTML-Version zum Demo-Notebook aus der Einleitung sieht folgendermaßen aus:



3.11.2 Aus externen Dateien lesen

3.11.2.1 Mathematica-Ausdrücke einlesen

Die eben in eine Datei namens **t_def** gesicherte Definition der Variablen `t` kann mit Hilfe des Umleitungsoperators "<<" bequem reaktiviert werden:

```
In[1]:= << t_def
Out[1]= {{1, 2, 3, 4}, {2, 4, 6, 8}, {3, 6, 9, 12}, {4, 8, 12, 16}}
```

Der Umleitungsoperator "<<", den wir auch beim Einlesen von Mathematica-Paketen verwenden (siehe Abschnitt 3.4) bewirkt, dass die Mathematica-Ausdrücke in der Eingabedatei *ausgewertet* werden.

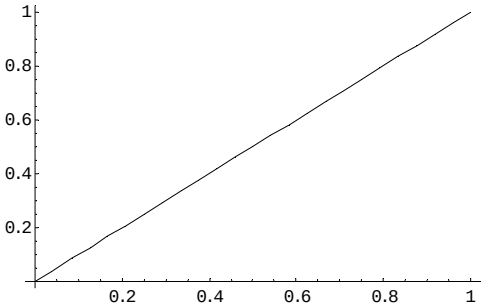
Gehen Sie wie im folgenden Beispiel vor, wenn Sie Dateiinhalte nur *anzeigen* lassen wollen:

```
In[2]:= !!t_def
t = {{1, 2, 3, 4}, {2, 4, 6, 8}, {3, 6, 9, 12}, {4, 8, 12, 16}}
```

Eine per Ausgabeumleitung im Mathematica-Format gesicherte Grafik kann folgendermaßen neu eingelesen und per **Show**-Funktion angezeigt werden:

```
In[3]:= Plot[x,{x,0,1}] >> gerade

In[4]:= Show[<<gerade]
```




```
Out[4]= -Graphics-
```

3.11.2.2 Aus Textdateien lesen

Die Funktion **ReadList** ermöglicht es, aus einer Textdatei Daten (z.B. Zahlen) in eine Mathematica-Liste einzulesen. Die Beispieldatei **test.dat** enthält zwei Zeilen mit tabulator-getrennten Zahlen:

```
In[1]:= !!test.dat
16  9  4  1
4   3  2  1
In[2]:= ReadList["test.dat", Number]
Out[2]= {16, 9, 4, 1, 4, 3, 2, 1}
```

Mit **Number** wird der Datentyp angegeben. Alternative Datentypen sind z.B. **Character** oder **String**.

Im vorliegenden Beispiel wird man es normalerweise bevorzugen, die Daten in eine Liste von Listen zu importieren, die in Mathematica zur Darstellung von Matrizen verwendet wird. Dazu ist im **ReadList**-Aufruf eine Option zu ergänzen:

```
In[3]:=  ma=ReadList["test.dat", Number, RecordLists->True]
Out[3]= {{16, 9, 4, 1}, {4, 3, 2, 1}}
```

4 Zusätzliche Hinweise zur Notizbuch-Benutzeroberfläche

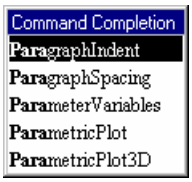
4.1 Hilfen beim Erstellen und Ausführen von Mathematica-Ausdrücken

4.1.1 Funktionsbezeichnungen automatisch vervollständigen lassen

Die Bezeichnungen von Funktionen (intern oder benutzerdefiniert) müssen nicht unbedingt komplett eingetippt werden, sondern können von Mathematica vervollständigt werden. Dazu verwendet man nach dem Eintippen des Wortanfangs den Menübefehl **Input > Complete Selection** oder besser die folgende Tastenkombination:

MS-Windows	Macintosh	X-Window
<Strg><K>	**<Befehl><K>**	**<Ctrl><K>**

Ist keine eindeutige Expansion möglich, werden alle Optionen in einem Fenster angeboten, das z.B. bei der Windows-Version nach Eingabe des Wortanfangs **Para** so aussieht:



4.1.2 Mathematica unterbrechen

Sie können eine laufende Berechnung mit dem Menübefehl **Kernel > Abort Evaluation** bzw. mit der Tastenkombination **<Alt><.>** (Macintosh: **<Befehl><.>**) unterbrechen, z.B.:

```
In[1]:= Do[i^33, {i, 3, 9999999}]
Out[2]= $Aborted
```

4.2 Zellenattribute und -stile

4.2.1 Zellenattribute

Über **Cell** > **Cell Properties** lassen sich für die **markierten** Zellen verschiedene Attribute ein- bzw. ausgeschalten. Das Markieren einer Zelle oder Zellengruppe erledigt man per Mausklick auf die zugehörige Klammer.

4.2.1.1 Editable

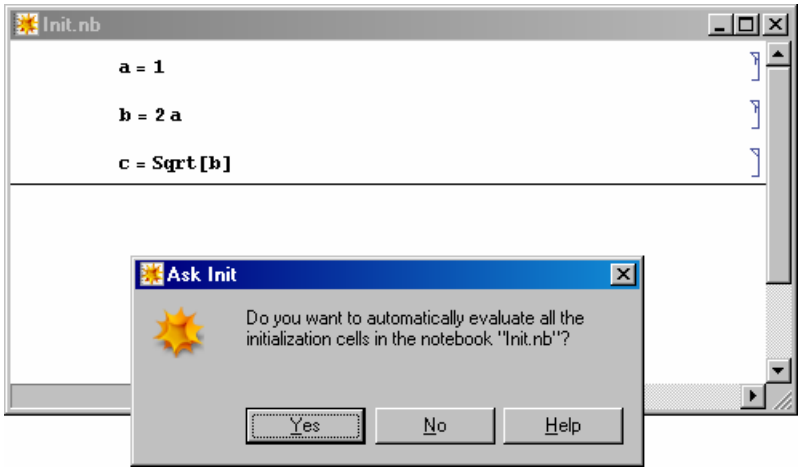
Man kann z.B. eine Input-Zelle vor unbeabsichtigten Veränderungen schützen, indem man ihr das Attribut **Editable** entzieht. Dieser Zustand wird durch ein "X" am oberen Rand der Zellenklammer gekennzeichnet:

4.2.1.2 Initialization

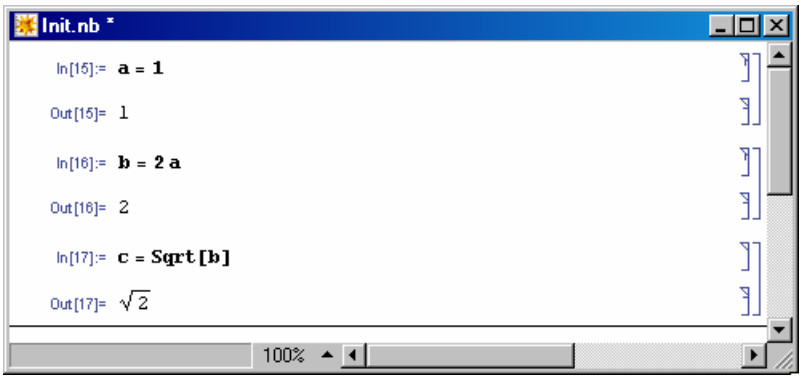
Die Initialisierungszellen eines Notebooks werden von Mathematica automatisch ausgeführt (nach Anfrage), sobald für irgendeine Zelle des Notebooks die Auswertung angefordert oder der Menübefehl

Kernel > Evaluation > Evaluate Initialization

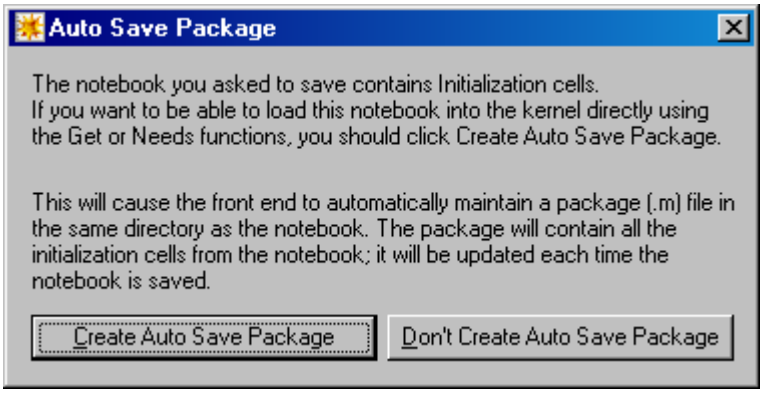
ausgeführt wird. So ist sichergestellt, dass gewisse Definitionen bei der Arbeit mit einem Notebook grundsätzlich zur Verfügung stehen. Das folgende Notebook enthält zwei einleitende Initialisierungszellen, zu erkennen am "I" in der Zellenklammer:



Lässt man die *dritte* Zelle ausführen und bestätigt die Auswertung der Initialisierungszellen, wird der Wert von c unter Berücksichtigung der Definitionen von a und b berechnet:



Beim ersten Speichern eines Notebooks mit Initialisierungszellen bietet Mathematica das Erstellen einer zugehörigen Package-Datei mit dem Inhalt aller Initialisierungszellen an:



Für die oben beschriebene Arbeitsweise mit Initialisierungszellen ist diese Datei *nicht* erforderlich, jedoch bietet sich hier eine gute Gelegenheit, ein eigenes Paket mit häufig benötigten Definitionen anzulegen und später komfortabel pflegen.

Das in der Dialogbox angesprochene **Get**-Kommando entspricht dem in Abschnitt 3.4 über Pakete vorgestellten Operator „<<“. Ein per Notizbuch mit Initialisierungszellen gepflegtes eigenes Paket lässt sich also bei Bedarf bequem importieren, z.B.:

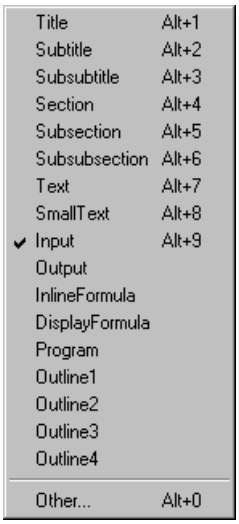
In[1]:=
```
<<"init.m"
```

4.2.2 Zellenstile

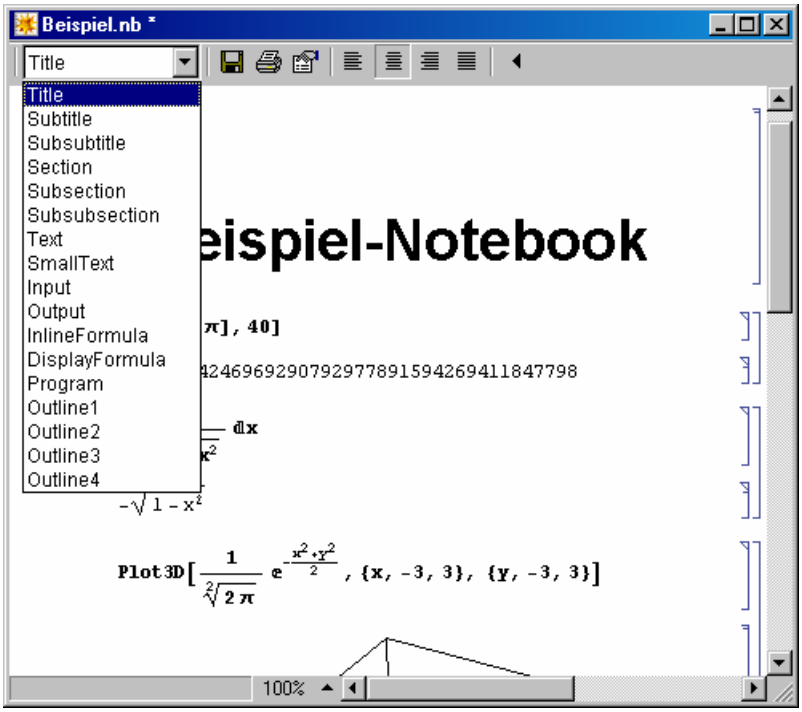
Jede Zelle hat einen bestimmten Stil, gekennzeichnet durch eine Kombination von Zellenattributen und Textformatierungen. Sie können für die markierten Zellen nach dem Menübefehl

Format > Style

aus dem folgenden Menü einen Zellenstil auswählen:



Wer mit **Format > Show ToolBar** die Symbolleiste zum Notebook-Fenster eingeschaltet hat, kann auch über die dortige versteckte Liste einen Zellenstil auswählen, z.B.:



Wie in obigem Beispiel mit einer einleitenden **Title**-Zelle zu sehen ist, wirken sich manche Stile auch auf die Gestalt der Zellenklammer aus, wobei wir uns die Einzelheiten aber sparen wollen.

4.2.3 Style Sheets

Die Ausgestaltung der einzelnen Zellenstile hängt vom aktiven **Style Sheet** ab. Mathematica bringt etliche allgemein verfügbare Style Sheets mit, die Sie einem Notebook über

Format > Style Sheet

zuweisen können. Das am Anfang des Manuskriptes gezeigte Beispiel-Notebook sieht nach Zuweisung des Style Sheets **PastelColor** (und Abschaltung der Zellennumerierung mit **Kernel > Show In/Out Names**) folgendermaßen aus:

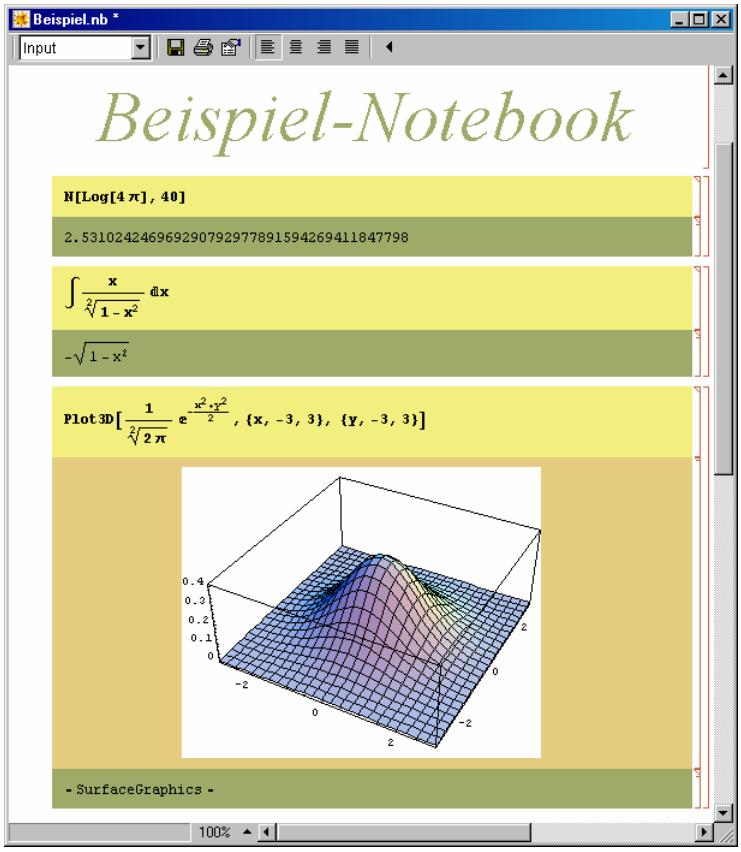


Abbildung 5 Beispiel-Notebook mit Style Sheet *PastelColor* und abgeschalteter Zellennumerierung

Wer ein eigenes Style Sheet gestalten möchte, kann dies nach **Format > Edit Style Sheet** tun, wobei folgende Möglichkeiten angeboten werden:

- Ein allgemein verfügbares Style Sheet ändern.
- Ein privates Style Sheet für das aktuelle Notebook erstellen.

4.3 Verfahren für Gruppen von Zellen

4.3.1 Zellen zu einer Gruppe zusammenfassen

Markieren Sie die gewünschten Zellen bzw. Zellengruppen, und wählen Sie den Menübefehl

Cell > Cell Grouping > Group Cells

Am rechten Fensterrand erscheint eine zusätzliche Gruppenklammer, die alle einbezogenen Zellen bzw. Zellengruppen umfasst.

4.3.2 Eine Gruppierung aufheben

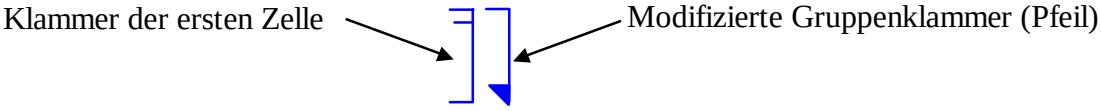
Markieren Sie die Gruppenklammer, und wählen Sie den Menübefehl

Cell > Cell Grouping > Ungroup Cells.

4.3.3 Zellengruppen aus- und einblenden

Eine Gruppe von Zellen kann per Doppelklick auf die Gruppenklammer (bzw. über den Menübefehl **Cell > Cell Grouping > Open/Close Group**) aus- bzw. wieder eingeblendet werden.

Eine ausgeblendete Gruppe wird im Notizbuch durch ihre *erste* Zelle vertreten, die sinnvollerweise einen Titel enthalten sollte, und in der Klammernzone folgendermaßen gekennzeichnet:



Wenn in *jeder* Gruppe die erste Zelle eine Überschrift enthält, ergibt sich durch Schließen aller Gruppen eine Gliederungsansicht des Notizbuchs.

4.3.4 Zellen aufteilen

Mit dem Menübefehl **Cell > Devide Cell** kann eine markierte Zelle u.a. folgendermaßen aufgeteilt werden:

i) Einfügemarke (|) befindet sich irgendwo innerhalb der Zelle

Zellinhalt vor der Einfügemarke	=	erste neue Zelle
Zellinhalt nach der Einfügemarke	=	zweite neue Zelle

ii) Eine Zeichenfolge ist markiert

Zellinhalt vor der Markierung	=	erste neue Zelle
Markierung	=	zweite neue Zelle
Zellinhalt nach der Markierung	=	dritte neue Zelle

4.3.5 Zellen zusammenlegen

Mehrere markierte Zellen können mit

Cell > Merge Cells

zusammengelegt werden, wobei die Attribute und Stile (siehe oben) der neuen Zelle von der ersten markierten Zelle übernommen werden.

5 Literatur und sonstige Informationsquellen

Im Manuskript werden folgende Publikationen zitiert:

Endl, K. & Luh, W. (1983). *Analysis II* (6. Aufl.). Wiesbaden: AULA-Verlag.

Hettich, R. (1989). *Numerische Lineare Algebra*. Trier: Vorlesungsmitschrift

Heuser, H. (1986). *Lehrbuch der Analysis, Teil 1*. Stuttgart: Teubner.

Wolfram Research Inc (1999). *Mathematica 4. Standard Add-On Packages*. Champaign, IL: Wolfram Media.

Wolfram, S. (2003). *The Mathematica Book* (5^{th} ed.). Champaign, IL: Wolfram Media.

Die Originalhandbücher zu Mathematica sind in elektronischer Form im Help Browser verfügbar (siehe Abschnitt 3.1). Wer gedruckte Informationen bevorzugt, kann die Handbücher aber auch in der Universitäts-Bibliothek ausleihen.

Im Internet finden sich diverse Ressourcen zu Mathematica, z.B. auf der WWW-Seite des Herstellers:

http://www.wolfram.com

6 Stichwortverzeichnis

H

I

K

L

M

N

O

P

R

S

T

U

V

W

Z