View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Juelich Shared Electronic Resources

FORSCHUNGSZENTRUM JÜLICH GmbH
Zentralinstitut für Angewandte Mathematik
D-52425 Jülich, Tel. (02461) 61-6402

Interner Bericht

Mathematik mit Maple
Eine Einführung mit Beispielen aus der Analysis, Linearen Algebra und Numerik

Johannes Grotendorst

FZJ-ZAM-IB-2000-19

November 2000

(letzte Änderung: 18.04.2001)

Inhaltsverzeichnis

Vorwort

Das vorliegende Manuskript ist aus Kursen im Forschungszentrum Jülich und Lehrveranstaltungen an der Fachhochschule Aachen/Abteilung Jülich zum Thema "Einführung in Maple" bzw. "Mathematik mit Maple" entstanden. Nach einer Einführung in die Grundlagen des Computeralgebra-Systems Maple werden die vielfältigen mathematischen Funktionen des Systems an Beispielen aus der Analysis, Linearen Algebra und Numerik demonstriert. Auf die Programmiermöglichkeiten in Maple wird ebenfalls eingegangen.

Die Maple-Kurse werden schon seit vielen Jahren in einem Ausbildungsraum, ausgestattet mit einem Pool von Rechnern (Workstations bzw. PCs) und einem Projektor für den Referenten, "interaktiv" durchgeführt. Dieses Skript und zusätzliches Übungsmaterial steht den Kursteilnehmern in Form von interaktiven mathematischen Dokumenten (Maple-Worksheets) zur Verfügung, die auf den Rechnern direkt ausgeführt und manipuliert werden können. Die Themengebiete sind so aufbereitet, daß sie auch leicht im Selbststudium bearbeitet und am Rechner nachvollzogen werden können. Die im Kurs behandelten Befehle und Funktionen sind in den zugehörigen Maple-Worksheets über Hyperlinks mit den entsprechenden Hilfeseiten des Maple-Systems verknüpft. Alle in diesem Skript verwendeten Beispiele wurden mit der Version Maple 6 erstellt.

In vielen Bereichen der industriellen Entwicklung und der wissenschaftlichen Forschung ist die Anwendung von Computeralgebra-Systemen auf dem Vormarsch. Im Bereich der Ausbildung an Schulen und Hochschulen erleben wir derzeit die Einführung bzw. Erprobung dieser Mathematikwerkzeuge. Nach den Logarithmentafeln, Tabellenwerken für mathematische Funktionen, Integral- und Integraltransformationstafeln, Rechenschiebern und Taschenrechnern werden nun Computeralgebra-Systeme zum Standardwerkzeug von Naturwissenschaftlern, Ingenieuren und Technikern.

Die nächste Generation mathematischer Werkzeuge steht schon vor der Tür. Mit der Einführung standardisierter Protokolle (OpenMath, MathML) für den Austausch mathematischer Datenstrukturen wird eine effiziente Integration bzw. Kopplung von mathematischer Software, inbesondere von Computeralgebra- und Numerik-Software, angestrebt. So werden in Maple 6 im neuen Paket LinearAlgebra erstmals NAG-Algorithmen zur linearen Algebra bereitgestellt.

Bei aller Begeisterung über das enorme Potential dieser Mathematikwerkzeuge darf der kritische Verstand bei der Anwendung dieser Systeme nicht abgeschaltet werden. Es sollte immer überlegt werden, ob die erhaltene Lösung sinnvoll ist (mathematisch oder physikalisch), ob die Größenordnung paßt oder was die Lösung in Spezialfällen besagt. Auch Mathematikprogramme machen Fehler!

Dieses Dokument wurde nahezu vollautomatisch aus dem zugrunde liegenden Maple-Dokument mittels der Funktion "Export As LaTeX" erzeugt. Für die Hilfe bei der Formatierung mit LaTeX möchte ich Frau Ilona Lütje und Frau Judith Mextorf danken.

Johannes Grotendorst, November 2000

Kapitel 1

Grundlagen

1.1 Maple als Taschenrechner

```
> restart;
```

Maple kann wie ein konventioneller Taschenrechner für einfache arithmetische Operationen benutzt werden. Jede Eingabe muß durch ein Semikolon (;) oder einen Doppelpunkt (:) abgeschlossen werden. Bei Abschluß der Eingabe mit einem Semikolon wird die Ausgabe angezeigt, andernfalls unterbleibt die Anzeige des Ergebnisses. Einfache Operatoren sind **+, -, *, /,** ^ bzw. ****** und **!**

```
> 1 + 2;
```

$$3$$

```
> 75 - 3;
```

$$72$$

```
> 5*3;
```

$$15$$

```
> 120/2;
```

$$60$$

Ganze und rationale Zahlen werden in Maple exakt verarbeitet, d.h. die Rechnungen erfolgen nach den Regeln der rationalen Arithmetik.

```
> 1/2 + 1/6;
```

$$\frac{2}{3}$$

Rationale Zahlen werden automatisch in der einfachsten Form ausgegeben, d.h. Zähler und Nenner haben keine gemeinsamen Teiler mehr. Maple kann mit beliebig großen Zahlen arbeiten:

```
> 100!;
```

$$933262154439441526816992388562667004907159682643816214685929638952175999932299156089414639761565182862536979208272237582511852109168640000000000000000000000000$$

Diese Zahl hat

```
> length(%);
```

$$158$$

Ziffern und die folgende Primzahlzerlegung:

```
> ifactor(%%);
```

$$(2)^{97}\ (3)^{48}\ (5)^{24}\ (7)^{16}\ (11)^{9}\ (13)^{7}\ (17)^{5}\ (19)^{5}\ (23)^{4}\ (29)^{3}\ (31)^{3}\ (37)^{2}\ (41)^{2}\ (43)^{2}\ (47)^{2}\ (53)\ (59)\ (61)\ (67)\ (71)\ (73)\ (79)\ (83)\ (89)\ (97)$$

Auf die letzten drei Ergebnisse einer Rechnung kann mittels des Prozentzeichens % **(ditto)** zurückgegriffen werden. Dabei übergibt % das letzte, %% das vorletzte und %%% das vorvorletzte Ergebnis an die Kommandozeile.

Der größte gemeinsame Teiler zweier natürlicher Zahlen:

```
> igcd(123,45);
```

$$3$$

Klammern werden so verwendet, wie es in der Mathematik üblich ist.

```
> 2*(3+1/3)/(5/3-4/5);
```

$$\frac{100}{13}$$

Einige mathematische Konstanten sind in Maple bereits vordefiniert, z.B. bezeichnet

`-Pi` die Kreisteilungszahl und

`-gamma` die Eulersche Konstante.

Eine Übersicht über die vordefinierten symbolischen Konstanten erhält man mit:

```
> constants;
```

$$false,\ \gamma,\ \infty,\ true,\ Catalan,\ FAIL,\ \pi$$

Maple kann Gleitkommazahlen mit beliebig vielen Dezimalstellen verarbeiten:

```
> Pi;
```

$$\pi$$

```
> evalf(Pi,100);
```

$$3.14159265358979323846264338327950288419716939937510582097494459230781640628620899862803482534211706\backslash 8$$

evalf berechnet einen numerischen Wert (f = floating-point) mit der angegebenen Stellenzahl, hier 100. Bei der Eingabe ist immer zu unterscheiden, ob eine Zahl mit oder ohne Dezimalpunkt eingegeben wird.

```
> 1/3;
```

$$\frac{1}{3}$$

```
> 1/3.;
```

$$.3333333333$$

```
> 3*%% -3*%;
```

$$.1\,10^{-9}$$

Wird in einem Ausdruck eine Zahl mit Dezimalpunkt angegben, dann werden alle vorkommenden arithmetischen Operationen numerisch mit einer vereinbarten Genauigkeit, hier mit 10 Dezimalstellen, durchgeführt.

```
> 2*(3.+1/3)/(5/3-4/5);
```

$$7.692307691$$

Maple kennt alle elementaren mathematischen Funktionen

```
> sin(Pi/4);
```

$$\frac{1}{2}\sqrt{2}$$

```
> sqrt(2);
```

$$\sqrt{2}$$

```
> sqrt(2.);
```

$$1.414213562$$

```
> exp(1);
```

$$e$$

```
> exp(1.);
```

$$2.718281828$$

```
> ln(1);
```

$$0$$

Eine Übersicht über die in Maple implementierten mathematischen Funktionen erhält man mit **inifcns**.

1.2 Wertzuweisung

```
> restart:
```

Wertzuweisungen (**assignment**) erfolgen in Maple nach der allgemeinen Vorschrift

<lhs> := <rhs>;

Die rechte Seite der Zuweisung <rhs> wird von Maple ausgewertet und der linken Seite <lhs> zugewiesen. Die linke Seite kann hierbei ein Name (**name**) eine Folge von Namen, ein indizierter Name oder ein Funktionsaufruf sein. Auf der rechten Seite können nicht nur mathematische Ausdrücke (**expression**), sondern auch Gleichungen oder Funktionen stehen. Für die Namen (Bezeichner) gelten die folgenden Regeln:

- Der Bezeichner muß mit einem Buchstaben beginnen.
- Danach können Zahlen, Unterstriche (_) oder weitere Buchstaben folgen.
- Sonderzeichen, wie z.B. %, #, @ u.a. sind nicht zugelassen.
- Namen von Maple-Befehlen dürfen nicht verwendet werden.
- Maple unterscheidet bei Namen zwischen Groß- und Kleinschreibung, so ist z.B. Var1 von var1 verschieden.

```
> Kreis_Umfang:=2 * Pi * r;
```

$$Kreis\_Umfang := 2\,\pi\,r$$

```
> Kreis_Flaeche:=F = Pi * r^2;
```

$$Kreis\_Flaeche := F = \pi\,r^2$$

Die Eingabe eines Namens liefert den aktuellen Wert der entsprechenden Variablen:

```
> Kreis_Umfang;
```

$$2\,\pi\,r$$

```
> Kreis_Flaeche;
```

$$F = \pi\,r^2$$

Mehrfachzuweisungen sind ebenfalls möglich:

```
> U,KF:=2*Pi*r,Pi*r^2;
```

$$U,\ KF := 2\,\pi\,r,\ \pi\,r^2$$

```
> U;KF;
```

$$2\,\pi\,r$$

$$\pi\,r^2$$

Solange einem Namen noch kein Wert zugewiesen wurde, steht dieser Name für sich selbst.

```
> r;
```

$$r$$

Eine Übersicht über alle in Maple vordefinierten Namen erhält man mit **ininame** .

Vorbesetzte Namen können durch Wertzuweisung nicht überschrieben werden:

```
> Pi:=3.14;
Error, attempting to assign to ‘Pi‘ which is protected
```

Der Namensschutz kann mit der Funktion **unprotect** aufgehoben werden:

```
> unprotect(Pi);
> Pi:=3.14;
```

$$\pi := 3.14$$

```
> evalf(2*Pi);
```

$$6.28$$

```
> r:=10;
```

$$r := 10$$

```
> Kreis_Flaeche;
```

$$F = 314.00$$

Das Löschen einzelner Wertzuweisungen erfolgt durch
<name> := '<name>';

```
> r:='r';
```

$$r := r$$

```
> Kreis_Flaeche;
```

$$F = 3.14\,r^2$$

```
> Pi:='Pi';
```

$$\pi := \pi$$

```
> Kreis_Flaeche;
```

$$F = \pi\,r^2$$

```
> evalf(Pi);
```

$$3.141592654$$

```
> protect(Pi);
> Pi :=3.14;
Error, attempting to assign to `Pi` which is protected
```

Soll für alle nachfolgenden Rechnungen mit einer fest eingestellten Anzahl von Dezimalstellen gerechnet werden, so muß die globale Variable **Digits** neu definiert werden:

```
> Digits:=100;
```

$$Digits := 100$$

```
> a:=1/3.;
```

$$a := .33333333333333333333333333333333333333333333333333333333333333333333333333333333\backslash$$
$$33333333333333333333333333333333333333$$

```
> '3*a - 1' = 3*a - 1;
```

$$3\,a - 1 = -.1\,10^{-99}$$

```
> sin(a);
```

$$.32719469679615224417334408526762060606430140689375979159005627707\backslash$$
$$05763744817615233969107939075693610$$

Die Eingabe eines Maple-Kommandos mit vorangestelltem Fragezeichen aktiviert die entsprechende Hilfeseite des Kommandos:

```
> ?solve
```

Es folgen einige typische Eingabefehler:

```
> b = sin(x)*cos(y)/(x*y);
```

$$b = \frac{\sin(x)\cos(y)}{x\,y}$$

```
> b;
```

$$b$$

Fehlt bei einer Zuweisung der Doppelpunkt, so betrachtet Maple die Zuweisung als Gleichung.
Hier fehlt eine Klammer um -1:

```
> v:=b^-1;
Error, `-` unexpected
> v := b^(-1);
```

$$v := \frac{1}{b}$$

Es muß grundsätzlich das Multiplikationszeichen (*) mit angegeben werden:

```
> 2x;
Error, missing operator or `;`
```

```
> 2*x;
```

$$2\,x$$

Hier fehlt bei **convert** das zweite Argument:

```
> convert(tan(x));
```

```
Error, wrong number (or type) of parameters in function convert
```

```
> convert(tan(x), sincos);
```

$$\frac{\sin(x)}{\cos(x)}$$

Siehe auch **convert,sincos**

1.3 Funktionen

```
> restart:
```

Für die Definition von Funktionen gibt es in Maple die Pfeil-Notation $->$:

```
> f:=x -> (x^4-2*x^3+3*x)/(2*x^2-4*x);
```

$$f := x \rightarrow \frac{x^4 - 2\,x^3 + 3\,x}{2\,x^2 - 4\,x}$$

```
> f(5.5);
```

$$15.55357143$$

```
> f(y);
```

$$\frac{y^4 - 2\,y^3 + 3\,y}{2\,y^2 - 4\,y}$$

Die Berechnung der Ableitungsfunktion f' erfolgt mit Hilfe des Operators **D** (nicht mit **diff**).

```
> f_strich:=D(f);
```

$$f\_strich := x \rightarrow \frac{4\,x^3 - 6\,x^2 + 3}{2\,x^2 - 4\,x} - \frac{(x^4 - 2\,x^3 + 3\,x)\,(4\,x - 4)}{(2\,x^2 - 4\,x)^2}$$

```
> f_strich(5.5);
```

$$5.377551021$$

diff erwartet als Argument einen Ausdruck, also z.B. den Funktionsterm f(x):

```
> fstx:=diff(f(x),x);
```

$$fstx := \frac{4\,x^3 - 6\,x^2 + 3}{2\,x^2 - 4\,x} - \frac{(x^4 - 2\,x^3 + 3\,x)\,(4\,x - 4)}{(2\,x^2 - 4\,x)^2}$$

Namen mit Sonderzeichen müssen in Maple in **backquotes** eingeschlossen werden:

```
> `f'`:=D(f);
```

$$f' := x \rightarrow \frac{4\,x^3 - 6\,x^2 + 3}{2\,x^2 - 4\,x} - \frac{(x^4 - 2\,x^3 + 3\,x)\,(4\,x - 4)}{(2\,x^2 - 4\,x)^2}$$

```
> `f'(5)`=`f'`(5);
```

$$f'(5) = \frac{29}{6}$$

Durch **unapply** kann aus einem Ausdruck eine Funktion erzeugt werden:

```
> fst:=unapply(fstx,x);
```

$$fst := x \rightarrow \frac{4\,x^3 - 6\,x^2 + 3}{2\,x^2 - 4\,x} - \frac{(x^4 - 2\,x^3 + 3\,x)\,(4\,x - 4)}{(2\,x^2 - 4\,x)^2}$$

```
> h:=f_strich - fst;
```

$$h := f\_strich - fst$$

```
> h(x);
```

$$0$$

Komposition von Funktionen mit **@**

```
> f:=x -> sin(x);
```

$$f := \sin$$

```
> g:=x -> x^2;
```

$$g := x \to x^2$$

```
> h:=g@f;
```

$$h := g@\sin$$

```
> h(x);
```

$$\sin(x)^2$$

```
> g(f(x));
```

$$\sin(x)^2$$

Man beachte den Unterschied:
h:= g@f liefert als Ergebnis eine Funktion, aber g(f(x)) liefert einen Ausdruck.
Mehrfach geschachtelte Funktionen:

```
> (f@@3)(x);
```

$$(\sin^{(3)})(x)$$

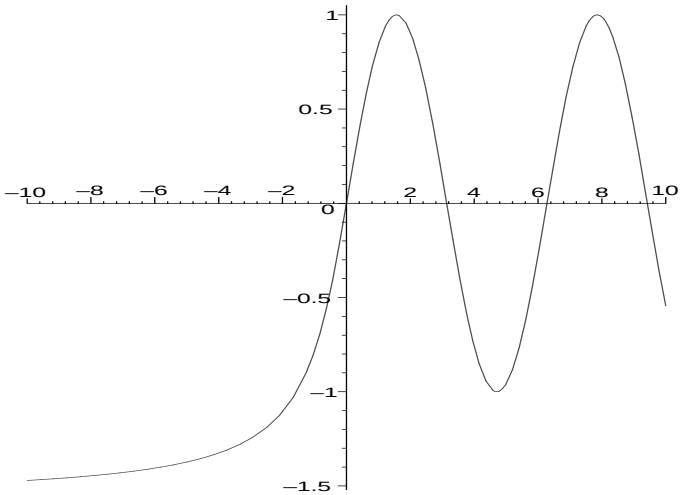
Stückweise definierte Funktionen:

```
> s:=x -> if x < 0 then arctan(x) else sin(x) end if;
```

$s :=$ **proc**(x) **option** $operator,\ arrow$; **if** $x < 0$ **then** $\arctan(x)$ **else** $\sin(x)$ **end if end proc**

Funktionen lassen sich in einfacher Weise graphisch darstellen:

```
> plot(s);
```



Berechnung einer Umkehrfunktion:

```
> y=exp(2*x);
```

$$y = e^{(2\,x)}$$

Gleichung nach x auflösen:

```
> solve(%,x);
```

$$\frac{1}{2}\ln(y)$$

```
> unapply(%,y);
```

$$y \rightarrow \frac{1}{2}\ln(y)$$

Das Volumen V einer Kugel ist eine Funktion des Radius r:

```
> V:=r -> 4/3* Pi*r^3;
```

$$V := r \rightarrow \frac{4}{3}\pi r^3$$

Der Radius hänge wiederum von der Zeit t ab (wie bei einem kugelförmigen Ballon, der mit Gas gefüllt wird). Um das Volumen V als Funktion von t darzustellen, nutzen wir wieder das Maple-Kommando unapply und erhalten so aus dem Ausdruck V(r(t)) eine Funktion der Zeit.

```
> V(r(t));
```

$$\frac{4}{3}\pi\,\mathrm{r}(t)^3$$

```
> V1:=unapply(V(r(t)),t);
```

$$V1 := t \rightarrow \frac{4}{3}\pi\,\mathrm{r}(t)^3$$

D(V1) liefert uns eine Funktion, die die Änderung des Volumens als Funktion der Zeit beschreibt.

```
> DV1:=D(V1);
```

$$DV1 := t \rightarrow 4\pi\,\mathrm{r}(t)^2\,\mathrm{D}(r)(t)$$

Definieren wir nun r als konkrete Zeitfunktion, dann erhalten wir bei Aufruf der Funktionen V1 und DV1 die zeitabhängigen Ausdrücke:

```
> r:=t -> 1 - exp(-t);
```

$$r := t \rightarrow 1 - e^{(-t)}$$

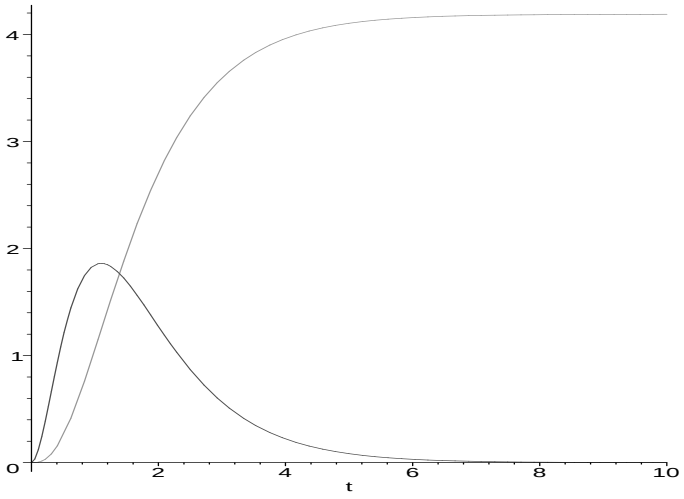
```
> V1(t);
```

$$\frac{4}{3}\pi\,(1 - e^{(-t)})^3$$

```
> DV1(t);
```

$$4\pi\,(1 - e^{(-t)})^2\,e^{(-t)}$$

```
> plot({V1(t),DV1(t)},t=0..10);
```



1.4 Library-Struktur

```
> restart:
```

Ein Design-Kriterium bei der Entwicklung von Maple war, daß Maple auch auf Rechnern mit kleinem Hauptspeicher laufen sollte. Es stehen deshalb nicht immer alle Funktionen direkt zur Verfügung (Maple hat über 3000 Funktionen).

Die Funktionen der Standard Library lassen sich unmittelbar aufrufen. Eine Übersicht erhält man mit **index,function**. Beispiel: Anwenden der Funktion **extrema**

```
> extrema(x*y,x^2+y^2=1,{x,y});
```

$$\{\frac{1}{2}, \frac{-1}{2}\}$$

Maple bietet für viele Anwendungsgebiete eigene Funktionspakete. Eine Liste der verfügbaren Pakete erhält man mit **index,package**. Das Paket **combinat** enthält z. B. die Funktion fibonacci zur Berechnung der Fibonacci-Zahlen **combinat,fibonacci**.

Die Funktion fibonacci kann nicht direkt genutzt werden:

```
> fibonacci(10);
```

$$\mathrm{fibonacci}(10)$$

Beim Aufruf der Funktion ist zusätzlich noch der Name des Paketes anzugeben und zwar in der Form

<package_name>[<function >](<parameter>);

```
> combinat[fibonacci](10);
```

$$55$$

Mit dem Kommando **with** können die Funktionen eines Paketes geladen werden:
with(<package_name>, <function>);
lädt nur die Funktion <function>, mit
with(<package_name>);
werden alle Funktionen eines Paketes verfügbar gemacht.

```
> with(combinat,fibonacci);
```

$$[fibonacci]$$

```
> fibonacci(10);
```

$$55$$

Die anderen Funktionen des Paketes combinat stehen noch nicht zur Verfügung:

```
> permute([a,b,c],2);
```

$$\mathrm{permute}([a, b, c], 2)$$

```
> with(combinat);
```

```
Warning, the protected name Chi has been redefined and unprotected
```

$$[\mathrm{Chi}, bell, \mathrm{binomial}, cartprod, character, choose, composition, conjpart, decodepart, encodepart, fibonacci, firstpart, graycode, inttovec, lastpart, multinomial, nextpart, numbcomb, numbcomp, numbpart, numbperm, partition, permute, powerset, prevpart, randcomb, randpart, randperm, stirling1, stirling2, subsets, vectoint]$$

Jetzt können alle Funktionen des Paketes combinat direkt aufgerufen werden:

```
> permute([a,b,c],2);
```

$$[[a, b], [a, c], [b, a], [b, c], [c, a], [c, b]]$$

```
> fibonacci(10);
```

$$55$$

1.5 Einfache Datenstrukturen

```
> restart:
```

Es soll eine einfache Folge von Termen (expression sequence, **exprseq**) eingegeben werden:

```
> 1,2,3,4,5;
```

$$1,\ 2,\ 3,\ 4,\ 5$$

```
> expseq1:=0,Pi/3,Pi/2,Pi;
```

$$expseq1 := 0,\ \frac{1}{3}\,\pi,\ \frac{1}{2}\,\pi,\ \pi$$

Eine Termfolge kann auch mit der Funktion **seq** erzeugt werden:

```
> expseq2:=seq(i^2,i=1..5);
```

$$expseq2 := 1,\ 4,\ 9,\ 16,\ 25$$

```
> expseq3:=seq (x^i,i=1..6);
```

$$expseq3 := x,\ x^2,\ x^3,\ x^4,\ x^5,\ x^6$$

Listen (**list**):

```
> data_list:=[expseq2];
```

$$data\_list := [1,\ 4,\ 9,\ 16,\ 25]$$

```
> monomials:=[expseq3];
```

$$monomials := [x,\ x^2,\ x^3,\ x^4,\ x^5,\ x^6]$$

Zugriff auf einzelne Elemente der Liste:

```
> data_list[3];
```

$$9$$

Anzahl der Listenelemente (**nops**):

```
> nops(monomials);
```

$$6$$

Liste in Termfolge verwandeln (**op**):

```
> op(monomials);
```

$$x,\ x^2,\ x^3,\ x^4,\ x^5,\ x^6$$

Mengen (**set**):

```
> set1:={1,2,3,a,b,c};
```

$$set1 := \{1,\ 2,\ 3,\ a,\ b,\ c\}$$

```
> set2:={0,1,y,a};
```

$$set2 := \{0,\ 1,\ a,\ y\}$$

Mengenoperationen:

```
> set1 union set2;
```

$$\{0,\ 1,\ 2,\ 3,\ a,\ b,\ c,\ y\}$$

```
> set1 intersect set2;
```

$$\{1,\ a\}$$

```
> set1 minus set2;
```

$$\{2,\ 3,\ b,\ c\}$$

Anzahl der Elemente einer Menge:

```
> nops(set1);
```

$$6$$

Menge in Folge von Termen verwandeln:

```
> op(set1);
```

$$1,\ 2,\ 3,\ a,\ b,\ c$$

Eine Funktion auf alle Elemente einer Menge (Liste) anwenden (**map**):

```
> map(sin,{expseq1});
```

$$\{0,\ 1,\ \frac{1}{2}\sqrt{3}\}$$

```
> map(sqrt,data_list);
```

$$[1,\ 2,\ 3,\ 4,\ 5]$$

Felder sind mehrdimensionale Datenstrukturen (**array**). Die Feldelemente können direkt bei der Feldvereinbarung durch eine geschachtelte Liste spezifiziert werden

```
> A:=array(-1..0,0..1,1..2,[[[1,x],[x,x^2]],[[1,y],[y,y^2]]]);
```

$$\begin{aligned}
&A := \mathrm{array}(-1..0,\ 0..1,\ 1..2,\ [\\
&(-1,\ 0,\ 1) = 1\\
&(-1,\ 0,\ 2) = x\\
&(-1,\ 1,\ 1) = x\\
&(-1,\ 1,\ 2) = x^2\\
&(0,\ 0,\ 1) = 1\\
&(0,\ 0,\ 2) = y\\
&(0,\ 1,\ 1) = y\\
&(0,\ 1,\ 2) = y^2\\
&])
\end{aligned}$$

oder mittels Wertzuweisung auf die einzelnen Feldelemente definiert werden (als Indizes sind ganze Zahlen erlaubt):

```
> A[-1,1,2]:=z^2;
```

$$A_{-1,\,1,\,2} := z^2$$

Matrizen (**matrices**) sind spezielle zweidimensionale Felder mit positiven Indizes:

```
> B:=matrix(2,2,(i,j) -> x^(i+j-1));
```

$$B := \begin{bmatrix} x & x^2 \\ x^2 & x^3 \end{bmatrix}$$

Die Matrixelemente lassen sich direkt mit Hilfe einer anonymen Funktion generieren (dritter Parameter). Mit der Funktion map können alle Elemente einer Matrix oder eines Feldes manipuliert werden:

```
> map(diff,B,x);
```

$$\begin{bmatrix} 1 & 2\,x \\ 2\,x & 3\,x^2 \end{bmatrix}$$

Für die effiziente Nutzung der Hardware-Gleitkommaarithmetik, insbesondere im Bereich der Linearen Algebra, gibt es in Maple die Datenstrukturen **Array** und **Matrix**.

```
> H:=Array(1..2,1..2, datatype=float[8]);
```

$$H := \begin{bmatrix} 0. & 0. \\ 0. & 0. \end{bmatrix}$$

```
> H[1,1]:=Pi; H[2,1]:=3.14;
```

```
Error, unable to store Pi when datatype=float[8]
```

$$H_{2,\,1} := 3.14$$

```
> H;
```

$$\begin{bmatrix} 0. & 0. \\ 3.140000000000000013 & 0. \end{bmatrix}$$

Die Gleitkommazahlen sind doppeltgenau (16 Stellen) nach Standard IEEE-754.

Symbolische Ausdrücke lassen sich erst speichern, wenn der angegebene Datentyp dies zulässt.

```
> H:=Array(1..2,1..2, datatype=anything);
```

$$H := \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

```
> H[1,1]:=Pi; H[2,1]:=3.14;
```

$$H_{1,1} := \pi$$

$$H_{2,1} := 3.14$$

```
> H;
```

$$\begin{bmatrix} \pi & 0 \\ 3.14 & 0 \end{bmatrix}$$

1.6 Strings, Namen

```
> restart:
```

In Maple werden drei Typen von Anführungszeichen (**quotes**) unterschieden:

- das Zeichen double-quote (“) wird als Begrenzungszeichen für Zeichenketten (**strings**) verwendet;

- das Zeichen back-quote (‘) wird als Begrenzungszeichen für Namen (**symbols**) verwendet;

- das Zeichen single-quote (’) schließlich verhindert die Auswertung von Ausdrücken (**uneval**).

Aneinanderhängen von Strings mit dem || Operator

```
> str:="Dies ist" ||" ein String";
```

$$str := \text{“Dies ist ein String”}$$

oder mittels **cat**:

```
> cat("Dies ist","," ein String");
```

“Dies ist ein String”

Länge des Strings (**length**):

```
> length(str);
```

19

Extrahieren und Suchen von Teilstrings (**substring**):

```
> substring(str,3..3);
```

“e”

```
> substring(str,3..8);
```

“es ist”

```
> searchtext("ist",str);
```

6

Maple-Ausdruck in String verwandeln (**convert,string**):

```
> convert(2/3,string);
```

“2/3”

```
> evalf(%);
```

“2/3”

String in Maple-Ausdruck zurückverwandeln (**parse**):

```
> parse(%);
```

$$\frac{2}{3}$$

```
> evalf(%);
```

.6666666667

Die bisher verwendeten Kommandos gelten auch für Namen (Symbole), z. B.

```
> 'Dies ist'||' ein Name';
```

$$Dies\ ist\ ein\ Name$$

Namen kann man Werte zuweisen:

```
> 'Dies ist'||' ein Name':=length(%);
```

$$Dies\ ist\ ein\ Name := 17$$

Strings können nicht auf der linken Seite einer Zuweisung stehen:

```
> "string" := 19;
```

```
Error, invalid left hand side of assignment
```

Es folgen noch einige Beispiele, wie Namen (Symbole) in Maple verwendet werden können.
Erzeugen von numerierten Namen:

```
> expseq:=1,2,3,4,5;
```

$$expseq := 1,\ 2,\ 3,\ 4,\ 5$$

```
> x||expseq;
```

$$x1,\ x2,\ x3,\ x4,\ x5$$

oder direkt

```
> x||(1,2,3,4,5);
```

$$x1,\ x2,\ x3,\ x4,\ x5$$

oder

```
> seq(x||k,k=1..5);
```

$$x1,\ x2,\ x3,\ x4,\ x5$$

Polynom mit unbestimmten Koeffizienten generieren mittels **add**:

```
> add(a||k*x^k,k=0..6);
```

$$a0 + a1\,x + a2\,x^2 + a3\,x^3 + a4\,x^4 + a5\,x^5 + a6\,x^6$$

Schöner wird dies mit indizierten Koeffizienten:

```
> add(a[k]*x^k,k=0..6);
```

$$a_0 + a_1\,x + a_2\,x^2 + a_3\,x^3 + a_4\,x^4 + a_5\,x^5 + a_6\,x^6$$

1.7 Ein- und Ausgabe über Dateien

```
> restart:
```

Mit dem **save** Kommando können die in Maple definierten Variablen in eine Datei geschrieben werden. Die Werte der Variablen werden dabei in linearer Ausgabeform (**lprint** Format) gespeichert. Diese Form hat den Vorteil, daß sie sich wieder als Eingabe für Maple-Befehle eignet. Es ist empfehlenswert, Dateinamen in Anführungszeichen (**doublequotes**) einzuschließen, so daß Maple sie als Strings interpretiert und nicht versehentlich als Ausdrücke behandelt. Dies ist insbesondere notwendig, wenn der Dateiname Sonderzeichen, wie z. B. einen Schrägstrich / (slash), einen Punkt . oder einen Bindestrich - enthält.

```
> a:=sin(x)/(x^2+1);
```

$$a := \frac{\sin(x)}{x^2+1}$$

```
> b:=diff(a,x);
```

$$b := \frac{\cos(x)}{x^2+1} - \frac{2\sin(x)\,x}{(x^2+1)^2}$$

```
> save a,b,"datei.txt";
```

Werden die Variablen in eine Textdatei geschrieben, so müssen die Namen der Variablen beim save Kommando explizit als Parameter angegeben werden. Endet der Dateiname mit .m, so werden die Daten in einem internen binären Maple-Format gespeichert. Mit

```
> save "datei.m";
```

werden alle definierten Variablen in die Datei “datei.m” geschrieben. Dateien mit Binärformat können von Maple schneller eingelesen werden als normale Textdateien. Durch die Kommandos **writeto** und **appendto** werden alle folgenden Eingaben und die zugehörigen Ausgaben (in 2-dimensionalem Textformat) in eine Datei geschrieben:

```
> writeto("file.txt");
> c:=a+b;
> writeto(terminal):
> c;
```

$$\frac{\sin(x)}{x^2+1}+\frac{\cos(x)}{x^2+1}-\frac{2\sin(x)\,x}{(x^2+1)^2}$$

```
> appendto("file.txt");
> d:=a*b;
> writeto(terminal):
```

Mit dem **read** Kommando lassen sich in Maple Text- und Binärdateien wieder einlesen:

```
> read "datei.txt";
```

$$a := \frac{\sin(x)}{x^2+1}$$

$$b := \frac{\cos(x)}{x^2+1}-\frac{2\sin(x)\,x}{(x^2+1)^2}$$

```
> read "datei.m";
```

Beim Einlesen von Binärdateien werden die Wertzuweisungen nicht am Bildschirm angezeigt. Das **writedata** Kommando schreibt numerische Daten von einem Vektor, einer Matrix oder einer Liste von Listen in eine Textdatei.

```
> A:=array([[1.5,2.2,3.4],[2.7,3.4,5.6],[1.8,3.1,6.7]]);
```

$$A := \begin{bmatrix} 1.5 & 2.2 & 3.4 \\ 2.7 & 3.4 & 5.6 \\ 1.8 & 3.1 & 6.7 \end{bmatrix}$$

```
> writedata(terminal,A,float);

1.5         2.2         3.4

2.7         3.4         5.6

1.8         3.1         6.7
> writedata("matrix.txt",A,float);
```

Das Kommando **readdata** liest zeilenweise Zahlen aus einer Datei und liefert eine Liste von Listen.

```
> readdata("matrix.txt",float,3);
```

$$[[1.5,\ 2.2,\ 3.4],\ [2.7,\ 3.4,\ 5.6],\ [1.8,\ 3.1,\ 6.7]]$$

```
> readdata("matrix.txt",[float,float,integer]);
```

$$[[1.5,\ 2.2,\ 3],\ [2.7,\ 3.4,\ 5],\ [1.8,\ 3.1,\ 6]]$$

```
> convert(%,matrix);
```

$$\begin{bmatrix} 1.5 & 2.2 & 3 \\ 2.7 & 3.4 & 5 \\ 1.8 & 3.1 & 6 \end{bmatrix}$$

Für das formatierte Lesen und Schreiben von Daten stehen die von der Programmiersprache C her bekannten Kommandos **scanf** und **printf** zur Verfügung.

Die Funktionen **latex**, **C** und **fortran** ermöglichen die Transformation von Maple-Ausdrücken nach Latex, C und Fortran:

```
> latex(b);
```

```
{\frac {\cos(x)}{{x}^{2}+1}}-2\,{\frac {\sin(x)x}{\left ({x}^{2}+1
\right )^{2}}}
```

```
> codegen[C](b);
```

```
   t0 = cos(x)/(x*x+1.0)-2.0*sin(x)/pow(x*x+1.0,2.0)*x;
```

```
> codegen[fortran](b);
```

```
   t0 = cos(x)/(x**2+1)-2*sin(x)/(x**2+1)**2*x
```

Die Ausgabe in eine Datei erfolgt gemäß

```
> latex(b,"datei.tex");
> codegen[C](b,filename="datei.c");
> codegen[fortran](b,filename="datei.f");
```

Weitere Ausgabe- und Transformationsmöglichkeiten (z.B. nach PostScript, LaTeX, RTF oder HTML) bietet die graphische Arbeitsoberfläche von Maple (**worksheet interface**).

1.8 Einfache Plotfunktionen

```
> restart:
```

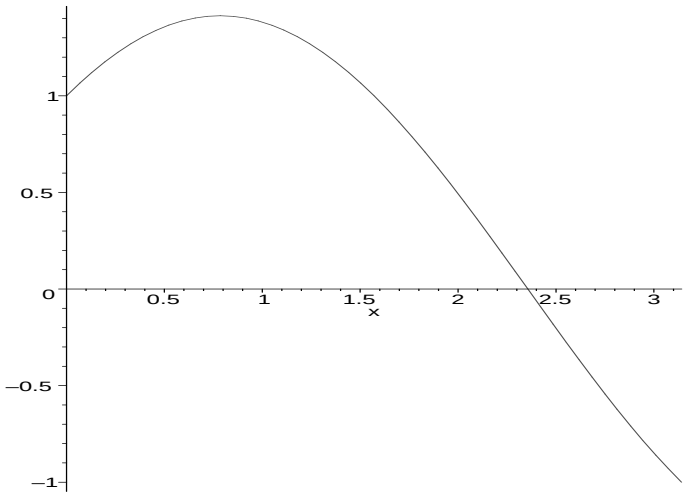
Zur graphischen Darstellung von mathematischen Formeln und Funktionen gibt es in Maple die Kommandos **plot** und **plot3d**. Weitergehende Plotfunktionen findet man in den Paketen **plots** und **plottools** bzw. in **DEtools,DEplot** und **stats,statplots**.

Es folgt ein einfaches Beispiel zur graphischen Darstellung einer Funktion:

```
> f:=x -> cos(x) + sin(x);
```

$$f := x \rightarrow \cos(x) + \sin(x)$$

```
> plot(f(x),x=0..Pi);
```



Die Graphik wird in das aktuelle Arbeitsblatt oder in ein separates Plotfenster gezeichnet, je nach Einstellung der Plot Display Option. Anschließend kann sie in der jeweiligen Plotumgebung über menügesteuerte Funktionen weiter verändert werden.

Das Plotten stückweise definierter Funktionen erfordert besondere Beachtung:

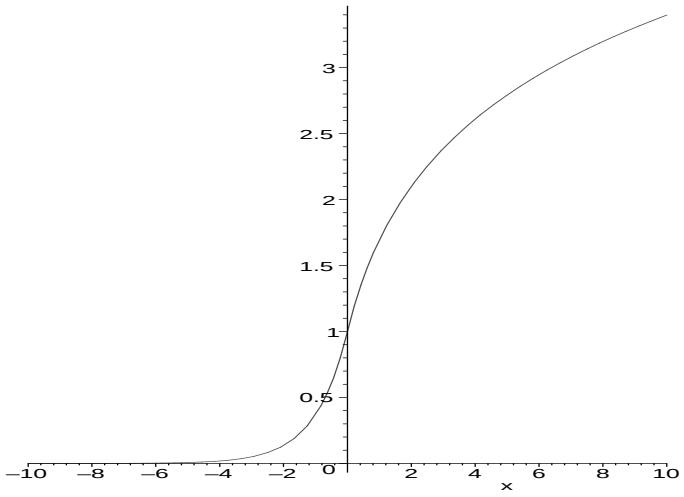
```
> s:=x -> if x<0 then exp(x) else log(x+1)+1 end if;
```

$s := \mathbf{proc}(x)$
option $operator,\ arrow$;
  if $x < 0$ **then** $\exp(x)$ **else** $\log(x+1)+1$ **end if**
end proc

```
> plot(s(x),x=-10..10);

Error, (in s) cannot evaluate boolean: x < 0
```

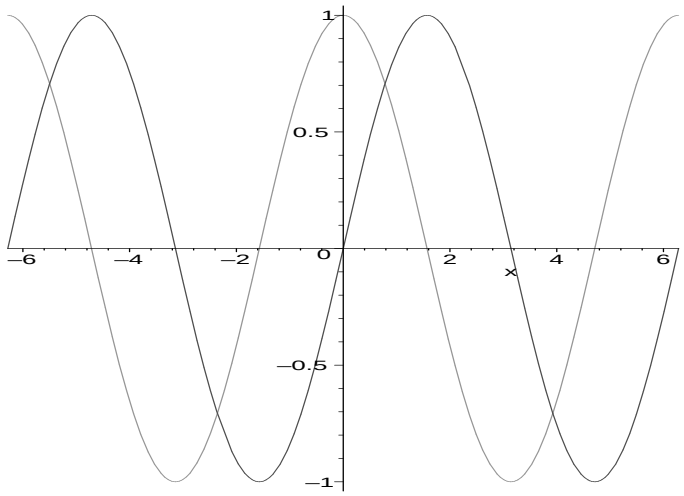
Beim Aufruf der Funktion plot wird zunächst der Funktionsausdruck s(x) symbolisch ausgewertet, d.h. Maple versucht den logischen Ausdruck $x<0$ auszuwerten, was zu der obigen Fehlermeldung führt. Die symbolische Auswertung wird unterdrückt, wenn man die Eingabe s(x) in Anführungszeichen setzt:

```
> plot('s(x)',x=-10..10);
```



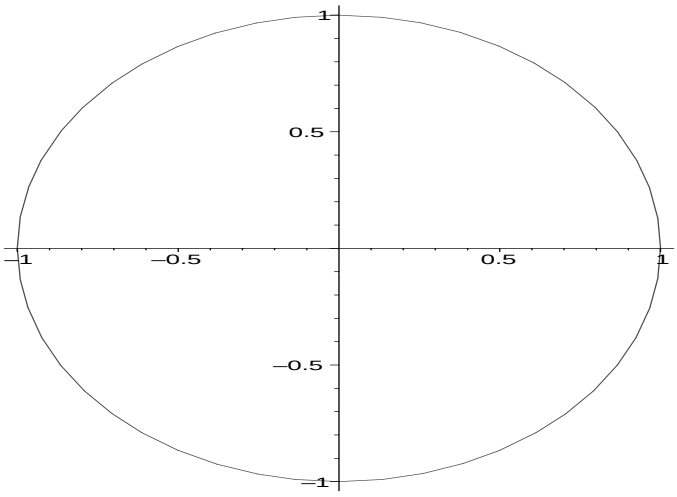
Darstellung von mehreren Funktionen in einem Schaubild:

```
> plot([sin(x),cos(x)],x=-2*Pi..2*Pi);
```



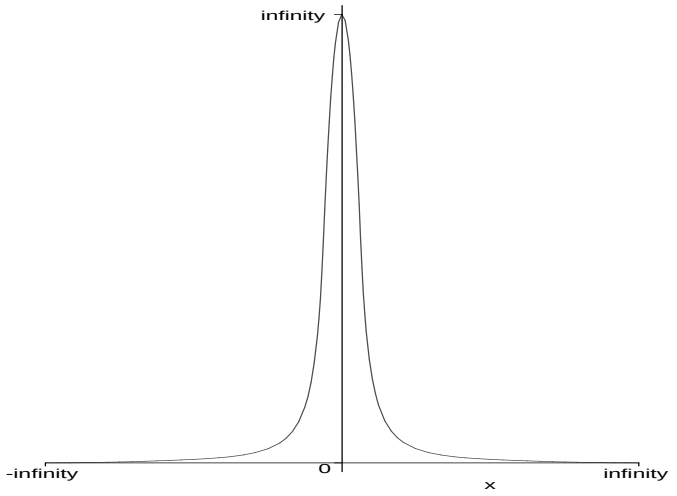
Darstellung einer Funktion, die in Parameterform gegeben ist:

```
> plot([sin(t),cos(t),t=-Pi..Pi]);
```



Soll der Funktionsverlauf für $x \to -\infty$ bzw. $x \to \infty$ dargestellt werden, so muß für die Plotbereichsgrenzen -infinity bzw. **infinity** eingesetzt werden.

```
> plot(1/x^2,x=-infinity..infinity);
```



Auch Punktmengen können graphisch dargestellt werden. Im nachfolgenden Beispiel werden die Koordinaten der Punkte

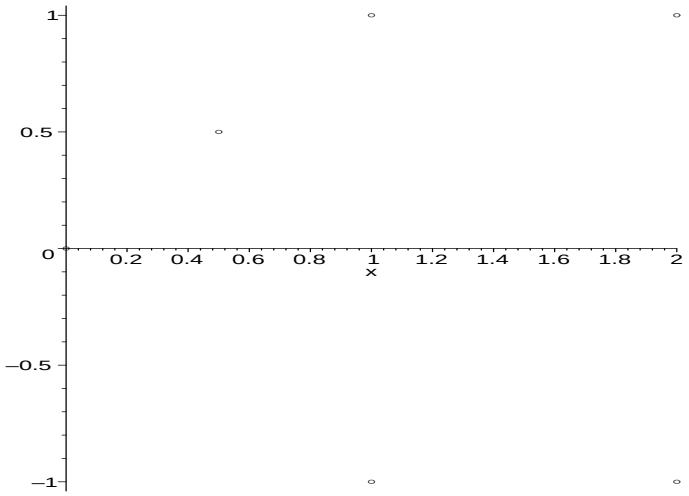
(0.5 , 0.5), (1 , 1), (2 , 1), (2 , -1), (1 , -1), (0 , 0)

als Liste an das plot Kommando übergeben:

```
> liste1:=[[0.5,0.5],[1,1],[2,1],[2,-1],[1,-1],[0,0]];
```

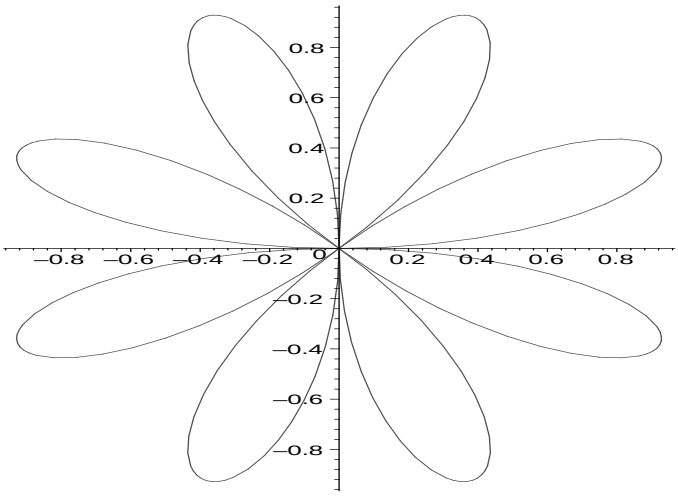
$$liste1 := [[.5,\ .5],\ [1,\ 1],\ [2,\ 1],\ [2,\ -1],\ [1,\ -1],\ [0,\ 0]]$$

```
> plot(liste1,x=0..2,style=point,symbol=circle);
```



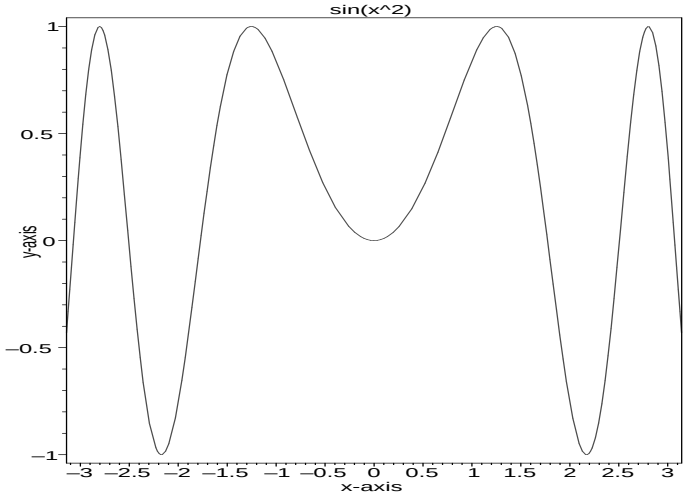
Darstellung mittels Polarkoordinaten:

```
> plot([sin(4*x),x,x=0..2*Pi],coords=polar);
```



Verschiedene Optionen können beim Aufruf des plot Kommandos gesetzt werden. Eine Liste der Optionen erhält man mit **plot,options** .

```
> plot(sin(x^2),x=-Pi..Pi,title="sin(x^2)",axes=boxed,
> xtickmarks=10,ytickmarks=5,labels=["x-axis","y-axis"],
> labeldirections=[horizontal,vertical]);
```



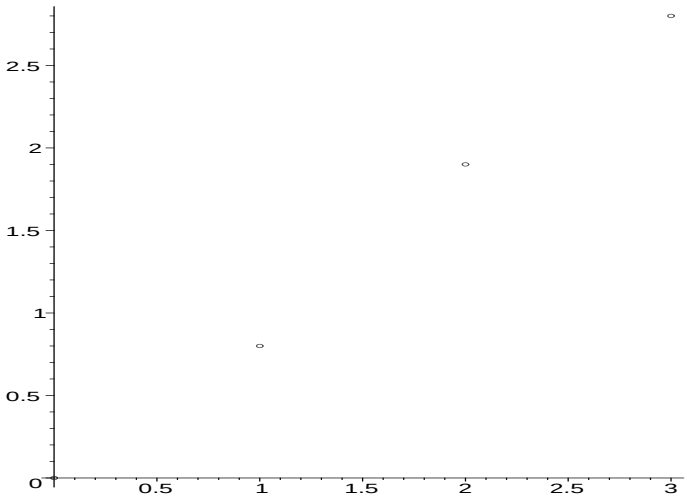
Die Datenstrukuren für 2D und 3D Plots lassen sich, wie andere Maple-Objekte auch, als benamte Objekte abspeichern. Mit der Funktion display können die gespeicherten Graphiken dann visualisiert werden. So lassen sich z.B. Plots mit unterschiedlichen Attributen zu einem Bild zusammenfassen.

Beispiel:

```
> liste2:=[[0,0],[1,0.8],[2,1.9],[3,2.8]];
```

$$liste2 := [[0,\ 0],\ [1,\ .8],\ [2,\ 1.9],\ [3,\ 2.8]]$$

```
> plt1:=plot(liste2,style=point,color=red,symbol=circle):

> plt1;
```



Wir wollen jetzt diese Daten sowie eine Ausgleichsgerade in einem Schaubild darstellen. Dazu werden zwei Graphiken mit unterschiedlichen Attributen erstellt. Die erste Graphik wird mit style=point erstellt und enthält nur die Datenpunkte. Danach wird mit der Funktion **leastsquare** aus dem Statistikpaket **stats,fit** die Ausgleichsgerade berechnet und diese Gerade mit dem Attribut style=line dargestellt.

```
> with(stats):

> with(fit):

> xl:=[seq(liste2[i][1],i=1..nops(liste2))];
```

$$xl := [0,\ 1,\ 2,\ 3]$$

```
> yl:=[seq(liste2[i][2],i=1..nops(liste2))];
```

$$yl := [0,\ .8,\ 1.9,\ 2.8]$$

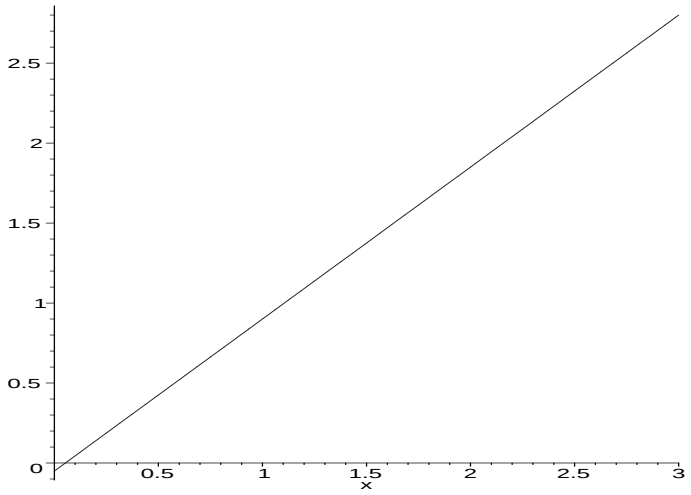
```
> leastsquare[[x,y],y=a*x+b,{a,b}]([xl,yl]);
```

$$y = .9500000000\,x - .05000000000$$

```
> g:=rhs(%);
```

$$g := .9500000000\,x - .05000000000$$

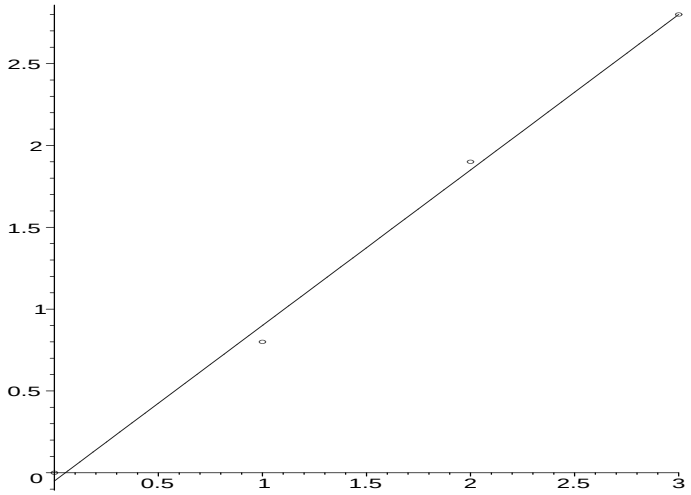
```
> plt2:=plot(g,x=0..3,style=line,color=blue):

> plt2;
```



Beide Graphiken lassen sich nun mit der Funktion display aus dem Paket plots in ein Bild zeichnen (**plots,display**):

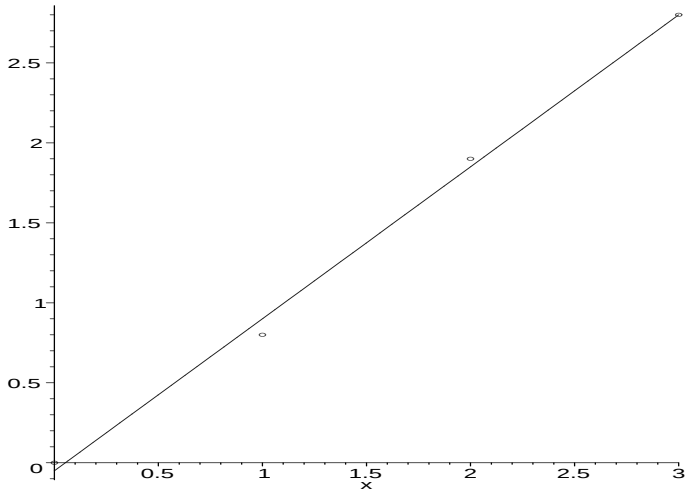
```
> with(plots):
> display({plt1,plt2});
```

```
Warning, the name changecoords has been redefined
```



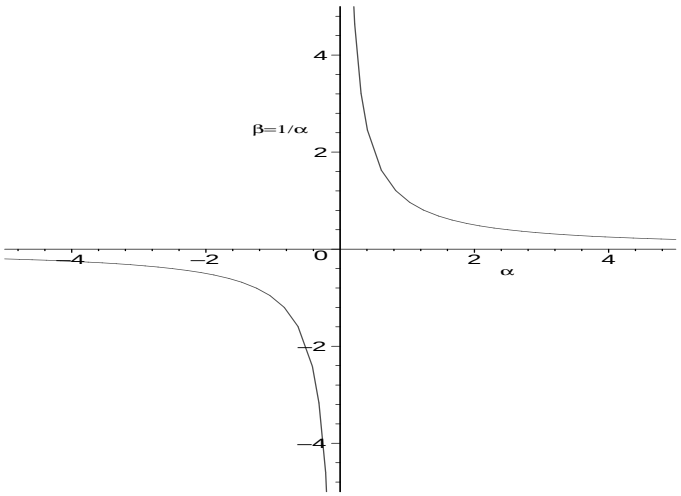
Mehrere 2D Graphiken mit unterschiedlichen Attributen können auch direkt mit dem Kommando plot veranschaulicht werden. Der folgende Befehl mit Optionslisten erzeugt die gleiche Graphik:

```
> plot([liste2,g],x=0..3,color=[red,blue],
> style=[point,line],symbol=[circle]);
```



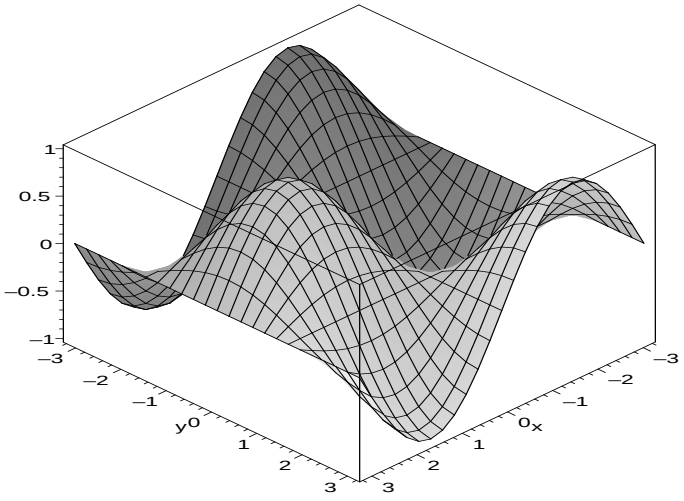
Eine Beschriftung der Koordinatenachsen mit griechischen Buchstaben erreicht man z. B. mit den folgenden Optionen:

```
> plot(1/a,a=-5..5,b=-5..5,labels=["a","b=1/a"],labelfont=[SYMBOL]);
```



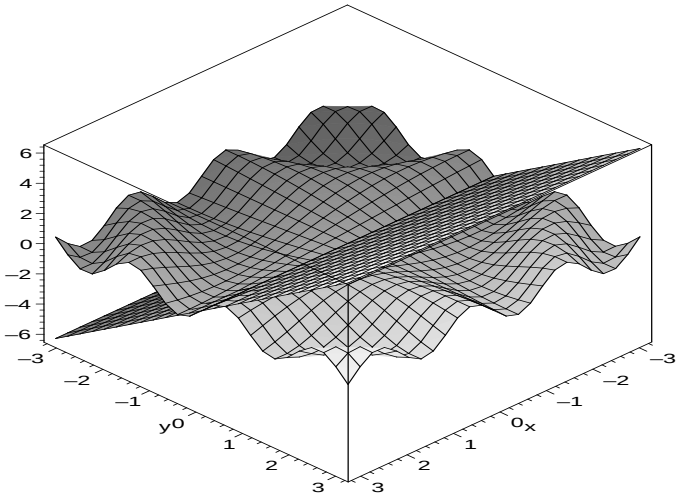
3D Graphiken werden mit dem Kommando **plot3d** erstellt. Es gibt hier ähnliche Optionen wie im 2D Fall.

```
> plot3d(sin(x)*cos(y),x=-Pi..Pi,y=-Pi..Pi,axes=boxed);
```



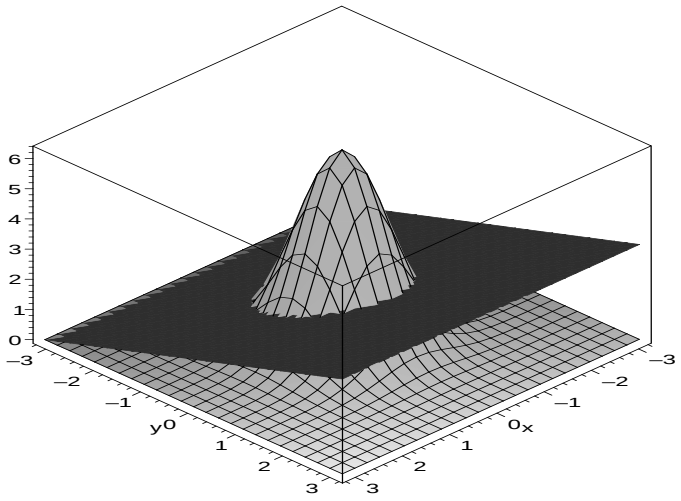
Es können auch mehrere Funktionen mit einem plot3d Aufruf veranschaulicht werden:

```
> plot3d({sin(x*y),2*y},x=-Pi..Pi,y=-Pi..Pi,style=patch,
> axes=boxed);
```



Falls 3D Graphiken mit unterschiedlichen Attributen in ein Bild sollen, kann man hierzu wieder die Funktion display verwenden:

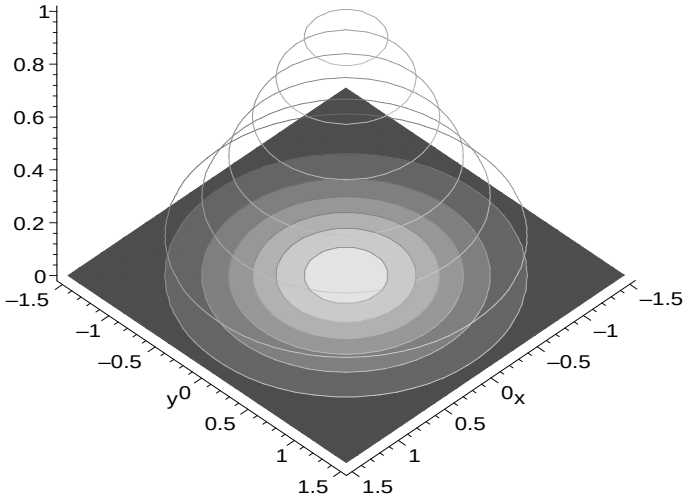
```
> plt1:=plot3d(2*Pi*exp(-x^2-y^2),x=-Pi..Pi,y=-Pi..Pi,style=patch):
> plt2:=plot3d(y/2+Pi/2,x=-Pi..Pi,y=-Pi..Pi,style=patchnogrid,
> color=navy):
> display({plt1,plt2},axes=boxed);
```



Die Funktion transform (**plottools,transform**) aus dem Plotpaket **plottools** ermöglicht die Transformation von Plotstrukturen. Eine Anwendung dieser Funktion ist z.B. die Einbettung eines 2D Plots in eine 3D Darstellung.

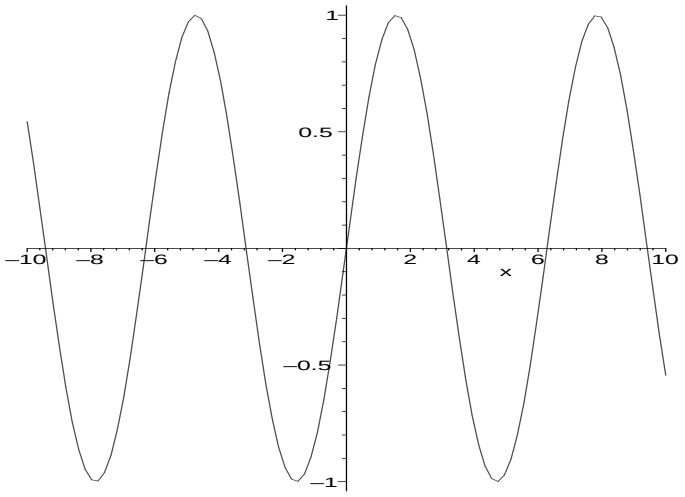
```
> p:=plot3d(exp(-x^2-y^2),x=-1.5..1.5,y=-1.5..1.5,style=contour,
> contours=[0.15,0.3,0.45,0.6,0.75,0.9]):
> q:=contourplot(exp(-x^2-y^2),x=-1.5..1.5,y=-1.5..1.5,
> contours=[0.15,0.3,0.45,0.6,0.75,0.9],filled=true):
> with(plottools):
> f:=transform((x,y) -> [x,y,0]):

Warning, the name transform has been redefined
```

```
> display({p,f(q)},axes=frame);
```



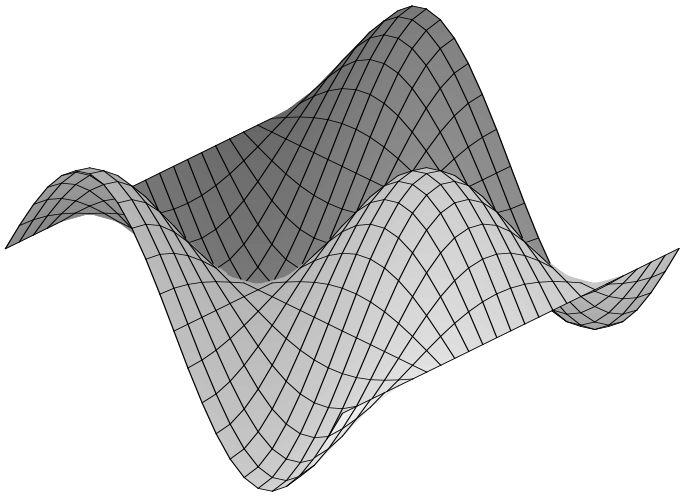
Animationen lassen sich mit den Funktionen **animate** und **animate3d** aus dem Plotpaket plots erstellen.

```
> animate(sin(x*t),x=-10..10,t=1..2,frames=50,numpoints=100);
```



Kontrolle und Steuerung der Animation erfolgen über eine kontextsensitive Menüleiste (**worksheet,plotinterface,animate**).

```
> animate3d(cos(t*x)*sin(t*y),x=-Pi..Pi,y=-Pi..Pi,t=1..2,
> style=patch);
```



Graphiken können in verschiedenen Formaten (**plot,device**) abgespeichert werden;

mit der Funktion **plotsetup** wird die Graphikausgabe gesteuert.

Die nachfolgende Graphik soll nun im PostScript-Format, ohne Bildrahmen und in Porträtdarstellung, in die Datei plotfile.ps geschrieben werden:

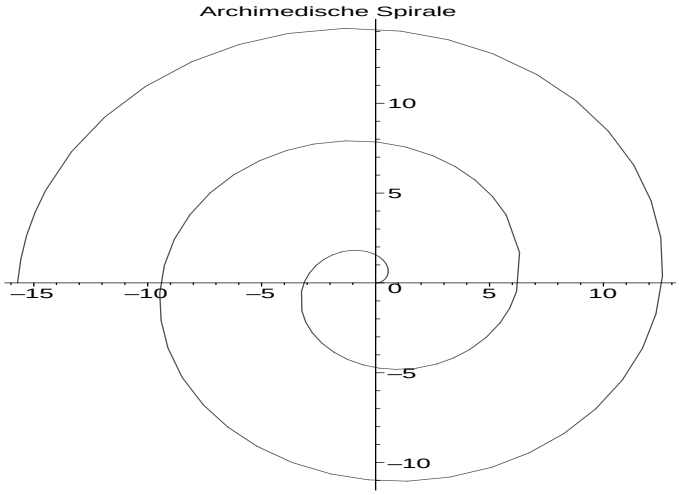
```
> plotsetup(ps,plotoutput="plotfile.ps",
> plotoptions="portrait,noborder");
> plot([t*cos(t),t*sin(t),t=0..5*Pi],title="Archimedische Spirale");
```

Anschließend werden wieder die Standardwerte für die Graphikausgabe in der aktuellen Benutzeroberfläche gesetzt:

```
> plotsetup(default);
> plotsetup();
```

$preplot = []$, $postplot = []$, $plotdevice = inline$, $plotoutput = terminal$, $plotoptions =$

```
> plot([t*cos(t),t*sin(t),t=0..5*Pi],title="Archimedische Spirale");
```



Kapitel 2

Manipulation von Ausdrücken und Datenstrukturen

2.1 simplify

```
> restart:
```

simplify - Anwendung von Vereinfachungsregeln auf Ausdrücke

```
> expr1:=4^(1/2) + 3;
```

$$expr1 := \sqrt{4} + 3$$

```
> simplify(expr1);
```

$$5$$

```
> trig_expr:=cos(x)^5+sin(x)^4+2*cos(x)^2-2*sin(x)^2-cos(2*x);
```

$$trig\_expr := \cos(x)^5 + \sin(x)^4 + 2\cos(x)^2 - 2\sin(x)^2 - \cos(2\,x)$$

```
> simplify(trig_expr);
```

$$\cos(x)^5 + \cos(x)^4$$

simplify kennt für viele Ausdrücke Vereinfachungsregeln. Wird eine bestimmte Regel als Argument spezifiziert, dann verwendet simplify nur diese Regel:

```
> expr2:= ln(x*y) + sin(x)^2 + cos(x)^2;
```

$$expr2 := \ln(x\,y) + \sin(x)^2 + \cos(x)^2$$

```
> simplify(expr2,trig);
```

$$\ln(x\,y) + 1$$

```
> simplify(expr2,ln);
```

$$\ln(x\,y) + \sin(x)^2 + \cos(x)^2$$

Die Produktregel des Logarithmus wird erst angewendet, wenn die Argumente bestimmte Voraussetzungen erfüllen. Soll eine Vereinfachung ohne weitere Annahmen symbolisch

durchgeführt werden, so kann man dies mit Hilfe des optionalen Parameters symbolic erreichen:

```
> simplify(expr2,ln,symbolic);
```

$$\ln(x) + \ln(y) + \sin(x)^2 + \cos(x)^2$$

simplify mit Annahmen:

```
> expr3:=sqrt(x^2);
```

$$expr3 := \sqrt{x^2}$$

```
> simplify(expr3);
```

$$\mathrm{csgn}(x)\,x$$

```
> ?csgn
```

```
> simplify(expr3,assume=real);
```
$$|x|$$
```
> simplify(expr3,assume=positive);
```
$$x$$
```
> simplify(expr2,ln,assume=positive);
```
$$\ln(x) + \ln(y) + \sin(x)^2 + \cos(x)^2$$

simplify kann unter Nebenbedingungen vereinfachen; es werden allerdings nur polynomiale Ausdrücke als Nebenbedingung berücksichtigt:

```
> expr4:=x*y*z+x*y+x*z+y*z;
```
$$expr4 := x\,y\,z + x\,y + x\,z + y\,z$$
```
> simplify(expr4,{x*z=1});
```
$$x\,y + y\,z + y + 1$$

Die Form der Vereinfachung kann man durch die Angabe einer Liste mit Variablen steuern:

```
> expr5:=x^3+y^3;
```
$$expr5 := x^3 + y^3$$
```
> siderel:=x^2+y^2=1;
```
$$siderel := x^2 + y^2 = 1$$
```
> simplify(expr5,{siderel},[x, y]);
```
$$y^3 - x\,y^2 + x$$
```
> simplify(expr5,{siderel},[y, x]);
```
$$x^3 - y\,x^2 + y$$

2.2 expand, factor

```
> restart:
```

expand - distributive Darstellung von Ausdrücken

```
> expand((x+1)*(x+2));
```
$$x^2 + 3\,x + 2$$
```
> expand(sin(2*x));
```
$$2\sin(x)\cos(x)$$
```
> expand(exp(a+ln(b)));
```
$$e^a\,b$$

Für Teilausdrücke kann die distributive Auswertung unterbunden werden:

```
> expand((x+1)*(x+y));
```
$$x^2 + x\,y + x + y$$
```
> expand((x+1)*(x+y),x+1);
```
$$(x+1)\,x + (x+1)\,y$$
```
> restart:
```

factor - Faktorisierung von Polynomen

```
> factor(x^2-1);
```
$$(x-1)\,(x+1)$$
```
> factor(x^3+y^3);
```
$$(x+y)\,(x^2 - x\,y + y^2)$$

Als Eingabe sind auch rationale Ausdrücke erlaubt; es werden dann Zähler und Nenner separat faktorisiert und anschließend gemeinsame Faktoren eliminiert.

```
> rat_expr:=(x^16-y^16)/(x^7-y^7);
```

$$rat\_expr := \frac{x^{16} - y^{16}}{x^7 - y^7}$$

```
> factor(rat_expr);
```

$$\frac{(x+y)\,(x^2+y^2)\,(x^4+y^4)\,(x^8+y^8)}{x^6 + y\,x^5 + y^2\,x^4 + y^3\,x^3 + y^4\,x^2 + y^5\,x + y^6}$$

```
> expand(numer(%)*(x-y))/expand(denom(%)*(x-y));
```

$$\frac{x^{16} - y^{16}}{x^7 - y^7}$$

Ohne zweites Argument erfolgt die Faktorisierung im Ring der Koeffizienten (ganze Zahlen, rationale Zahlen, ...). Faktorisierung über bestimmte algebraische Zahlkörper ist ebenfalls möglich:

```
> factor(x^2+1);
```

$$x^2 + 1$$

```
> factor(x^2+1,I);
```

$$-(x+I)\,(-x+I)$$

```
> factor(x^4-2,sqrt(2));
```

$$-(-x^2+\sqrt{2})\,(x^2+\sqrt{2})$$

2.3 normal, rationalize

```
> restart:
```

normal - in rationalen Ausdrücken mit ganzen Koeffizienten werden gemeinsame Teiler in Zähler- und Nennerpolynom eliminiert.

```
> expr1:=(x^2 - y^2)/(x-y)^3;
```

$$expr1 := \frac{x^2 - y^2}{(x-y)^3}$$

```
> normal(expr1);
```

$$\frac{y+x}{(-x+y)^2}$$

normal wirkt rekursiv auf die Komponenten eines Ausdrucks:

```
> expr2:=sin((x*(x+1)-x)/(x+2))^2+cos((x^3+x^2)/(-(x^2+3*x+2)))^2;
```

$$expr2 := \sin(\frac{x\,(x+1)-x}{x+2})^2 + \cos(\frac{x^3+x^2}{-x^2-3\,x-2})^2$$

```
> normal(expr2);
```

$$\sin(\frac{x^2}{x+2})^2 + \cos(\frac{x^2}{x+2})^2$$

normal kennt keine Vereinfachungsregeln:

```
> simplify(%);
```

$$1$$

Die Option expanded liefert die distributive Darstellung von Zähler und Nenner:

```
> expr3:=1/x+x/(x+1);
```

$$expr3 := \frac{1}{x} + \frac{x}{x+1}$$

```
> normal(expr3);
```

$$\frac{x+1+x^2}{x\,(x+1)}$$

```
> normal(1/x+x/(x+1),expanded);
```

$$\frac{x+1+x^2}{x^2+x}$$

Da normal den Zähler immer in distributiver Form darstellt, ist die Anwendung von normal nicht immer sinnvoll!

```
> expr4:= (x^20-1)/(x-1);
```

$$expr4 := \frac{x^{20}-1}{x-1}$$

```
> normal(expr4);
```

$$x^{19}+x^{18}+x^{17}+x^{16}+x^{15}+x^{14}+x^{13}+x^{12}+x^{11}+x^{10}+x^9+x^8+x^7+x^6+x^5+x^4+x^3+x^2+x+1$$

Hier ist factor sinnvoller.

```
> factor(expr4);
```

$$(x+1)\,(x^2+1)\,(x^8-x^6+x^4-x^2+1)\,(x^4-x^3+x^2-x+1)\,(x^4+x^3+x^2+x+1)$$

rationalize - Rationalisieren von Ausdrücken (Quadratwurzeln werden im Nenner beseitigt)

```
> (x+y)/(x+sqrt(3));
```

$$\frac{y+x}{x+\sqrt{3}}$$

```
> rationalize(%);
```

$$\frac{(y+x)\,(x-\sqrt{3})}{x^2-3}$$

2.4 convert

```
> restart:
```

convert - Umformung von Ausdrücken in äquivalente Darstellungen

```
> convert(cot(x),sincos);
```

$$\frac{\cos(x)}{\sin(x)}$$

Partialbruchzerlegung:

```
> convert((x^5+1)/(x^4-x^2),parfrac,x);
```

$$x+\frac{1}{x-1}-\frac{1}{x^2}$$

Gleitkommazahl in eine rationale Zahl umwandeln:

```
> pi:=evalf(Pi);
```

$$\pi := 3.141592654$$

```
> convert(pi,rational,exact);
```

$$\frac{1570796327}{500000000}$$

In Maple wird eine Gleitkommazahl intern als Sequenz von zwei ganzen Zahlen (Mantisse und Exponent) dargestellt. Mit dem Kommando **op** läßt sich dieses Zahlenpaar anzeigen:

```
> piseq:=op(pi);
```

$$piseq := 3141592654,\ -9$$

```
> piseq[1]*10^piseq[2];
```

$$\frac{1570796327}{500000000}$$

Gleitkommazahl durch eine rationale Zahl annähern:

```
> pi10:=convert(pi,rational,10);
```

$$\pi 10 := \frac{104348}{33215}$$

```
> evalf(%);
```

$$3.141592654$$

pi10 und Pi stimmen in 10 Dezimalstellen überein:

```
> evalf(Pi - pi10);
```

$$0.$$

```
> pi3:=convert(pi,rational,3);
```

$$\pi 3 := \frac{22}{7}$$

pi3 und Pi stimmen in 3 Dezimalstellen überein:

```
> evalf(Pi - pi3,3) ;
```

$$0.$$

Typänderung von Datenstrukturen oder Ausdrücken:

```
> l:=convert(a+b+c,list);
```

$$l := [a, b, c]$$

```
> convert(l,set);
```

$$\{a, b, c\}$$

```
> taylor_entw:=series(sin(x),x);
```

$$taylor\_entw := x - \frac{1}{6}x^3 + \frac{1}{120}x^5 + \mathrm{O}(x^6)$$

```
> subs(x=2,taylor_entw);

Error, invalid substitution in series
> taylor_poly:=convert(taylor_entw,polynom);
```

$$taylor\_poly := x - \frac{1}{6}x^3 + \frac{1}{120}x^5$$

```
> subs(x=2,taylor_poly);
```

$$\frac{14}{15}$$

2.5 collect, combine

```
> restart:
```

collect - Zusammenfassen von Koeffizienten mit gleichen Faktoren

```
> poly1:=x^2+2*y*x-3*y+y^2*x^2;
```

$$poly1 := x^2 + 2\,y\,x - 3\,y + y^2\,x^2$$

```
> collect(poly1,x);
```

$$(1 + y^2)\,x^2 + 2\,y\,x - 3\,y$$

```
> collect(poly1,y);
```

$$y^2\,x^2 + (2\,x - 3)\,y + x^2$$

```
> trig_expr:=sin(x)*cos(x)+sin(x)+y*sin(x);
```

$$trig\_expr := \sin(x)\cos(x) + \sin(x) + y\sin(x)$$

```
> collect(trig_expr,sin(x));
```

$$(\cos(x) + 1 + y)\sin(x)$$

```
> poly2:=z*x*y+2*x*y+z;
```

$$poly2 := z\,x\,y + 2\,y\,x + z$$

```
> collect(poly2,x*y);

Error, (in collect) cannot collect y*x
```

Koeffizienten von Produkten werden nicht zusammengefaßt; die Einführung einer Zwischengröße hilft hier aber oft weiter:

```
> subs(x=xyprod/y,poly2);
```

$$z\,xyprod + 2\,xyprod + z$$

```
> collect(%,xyprod);
```

$$(z+2)\,xyprod + z$$

```
> subs(xyprod=x*y,%);
```

$$(z+2)\,y\,x + z$$

Sollen die Koeffizienten mehrerer Variablen zusammengefaßt werden, so kann man die rekursive Methode (default) oder die distributive Methode (distributed) anwenden:

```
> poly3:=x*y+z*x*y+y*x^2-z*y*x^2+x+z*x;
```

$$poly3 := y\,x + z\,x\,y + y\,x^2 - z\,y\,x^2 + x + z\,x$$

```
> collect(poly3,[x, y]);
```

$$(1-z)\,y\,x^2 + ((1+z)\,y + 1 + z)\,x$$

```
> collect(poly3,[y, x]);
```

$$((1-z)\,x^2 + (1+z)\,x)\,y + (1+z)\,x$$

```
> collect(poly3,[x, y], distributed);
```

$$(1+z)\,y\,x + (1+z)\,x + (1-z)\,y\,x^2$$

combine - Zusammenfassen von Termen in Summen, Produkten und Potenzen

```
> exp(x)^2*exp(y);
```

$$(e^x)^2\,e^y$$

```
> combine(%,exp);
```

$$e^{(2\,x+y)}$$

combine wirkt in einigen Fällen wie die Umkehrung von expand.

```
> expand(%);
```

$$(e^x)^2\,e^y$$

```
> (x^a)^2;
```

$$(x^a)^2$$

```
> combine(%,power);
```

$$x^{(2\,a)}$$

```
> expand(%);
```

$$(x^a)^2$$

```
> assume(x>0);
> expr1:=(x+1)^(1/2)*(x+2)^(1/2);
```

$$expr1 := \sqrt{x\tilde{} + 1}\,\sqrt{x\tilde{} + 2}$$

```
> combine(expr1);
```

$$\sqrt{(x\tilde{} + 1)\,(x\tilde{} + 2)}$$

```
> factor(%);
```

$$\sqrt{(x\tilde{} + 1)\,(x\tilde{} + 2)}$$

```
> expr2:=exp(sin(a)*cos(b))*exp(cos(a)*sin(b));
```

$$expr2 := e^{(\sin(a)\cos(b))}\,e^{(\cos(a)\sin(b))}$$

```
> combine(expr2);
```

$$e^{\sin(a+b)}$$

```
> expand(%);
```

$$e^{(\sin(a)\cos(b))}\,e^{(\cos(a)\sin(b))}$$

Zusammenfassen von Termen mit Integralen und Grenzwerten:

```
> x := 'x':
> Int(x,x=a..b) - Int(x^2,x=a..b);
```

$$\int_a^b x\,dx - \int_a^b x^2\,dx$$

```
> combine(%);
```

$$\int_a^b x - x^2\,dx$$

```
> Limit(x, x=a) * Limit(x^2,x=a) + c;
```

$$(\lim_{x\to a} x)\,(\lim_{x\to a} x^2) + c$$

```
> combine(%);
```

$$\lim_{x\to a} x^3 + c$$

2.6 sort

```
> restart:
```

sort - sortiert eine Liste mit Zahlen in aufsteigender Reihenfolge und die Terme eine Polynoms in absteigender Ordnung

```
> sort([1,3,2,5,3,6,3,6]);
```

$$[1,\ 2,\ 3,\ 3,\ 3,\ 5,\ 6,\ 6]$$

Eine Liste mit Strings wird lexikographisch sortiert:

```
> sort([a,z,x,e,f,r,b]);
```

$$[a,\ b,\ e,\ f,\ r,\ x,\ z]$$

Sortierung mit gegebener Sortierprozedur:

```
> f:=(x,y) -> if length(x) < length(y) then true else false end if;
```

$f := \mathbf{proc}(x,\ y)$
option $operator,\ arrow$;
  if $\mathrm{length}(x) < \mathrm{length}(y)$ **then** $true$ **else** $false$ **end if**
end proc

```
> sort([maple,blue,car,a],f);
```

$$[a,\ car,\ blue,\ maple]$$

Gemischte Listen werden nicht sortiert:

```
> big_list:=[1,d,2,5,a,c,b,9];
```

$$big\_list := [1,\ d,\ 2,\ 5,\ a,\ c,\ b,\ 9]$$

```
> sort(big_list);
```

$$[1,\ 2,\ 5,\ 9,\ b,\ d,\ c,\ a]$$

Selektiv könnte etwa so sortiert werden:

```
> list1:=[d,a,c,b];
```

$$list1 := [d,\ a,\ c,\ b]$$

```
> list2:=[1,2,5,9];
```

$$list2 := [1,\ 2,\ 5,\ 9]$$

```
> list1:=sort(list1); list2:=sort(list2);
```

$$list1 := [a,\ b,\ c,\ d]$$

$$list2 := [1,\ 2,\ 5,\ 9]$$

```
> sorted_list:=[op(list1),op(list2)];
```

$$sorted\_list := [a,\ b,\ c,\ d,\ 1,\ 2,\ 5,\ 9]$$

Polynome werden in absteigender Ordnung sortiert:

```
> poly:=x^2+3*x+x^3+1;
```

$$poly := x^2 + 3\,x + x^3 + 1$$

```
> sort(poly);
```

$$x^3 + x^2 + 3\,x + 1$$

sort ersetzt Polynom poly durch das sortierte Polynom:

```
> poly;
```

$$x^3 + x^2 + 3\,x + 1$$

Polynome mit mehreren Variablen werden nach der Potenzsumme sortiert, die Variablenordnung kann angegeben werden:

```
> sort(x^2*y^3+y^2*x^2+x^3,[x,y]);
```

$$x^2\,y^3 + x^2\,y^2 + x^3$$

```
> sort(x^2*y^3+y^2*x^2+x^3,[y,x]);
```

$$y^3\,x^2 + y^2\,x^2 + x^3$$

2.7 map, zip

```
> restart:
```

map - Anwendung einer Maple-Funktion auf alle Operanden eines Ausdrucks oder einer Datenstruktur

```
> data_list:=[0,x,y,exp(x),3,-4];
```

$$data\_list := [0,\ x,\ y,\ e^x,\ 3,\ -4]$$

```
> map(t->t^2,data_list);
```

$$[0,\ x^2,\ y^2,\ (e^x)^2,\ 9,\ 16]$$

```
> map(sin,y+cos(x)+Pi/2);
```

$$\sin(y) + \sin(\cos(x)) + 1$$

```
> eqnset:={x^2+x=2,x^2=4};
```

$$eqnset := \{x^2 + x = 2,\ x^2 = 4\}$$

```
> map(lhs-rhs,eqnset);
```

$$\{x^2 - 4,\ x^2 + x - 2\}$$

```
> map(factor,%);
```

$$\{(x-2)\,(x+2),\ (x+2)\,(x-1)\}$$

```
> fcn_list:=[sin(x),ln(x),x^2];
```

$$fcn\_list := [\sin(x),\ \ln(x),\ x^2]$$

```
> map(diff,fcn_list,x);
```

$$[\cos(x),\ \frac{1}{x},\ 2\,x]$$

zip - Verknüpfen zweier Listen

```
> list1:=[1,15,8];
```

$$list1 := [1,\ 15,\ 8]$$

```
> list2:=[2,6,12];
```

$$list2 := [2,\ 6,\ 12]$$

```
> zip(gcd,list1,list2);
```

$$[1,\ 3,\ 4]$$

```
> zip((x,y) -> x+y,list1,list2);
```

$$[3,\ 21,\ 20]$$

2.8 Strukturelle Manipulationen

```
> restart:
```

Linke Seite (**lhs**) und rechte Seite (**rhs**) einer Gleichung:

```
> eqn1:=x^2=3;
```

$$eqn1 := x^2 = 3$$

```
> lhs(eqn1);
```

$$x^2$$

```
> rhs(eqn1);
```

$$3$$

Extrahieren von Teilausdrücken mit **op**:

```
> poly:=1+x+x^2;
```

$$poly := 1 + x + x^2$$

```
> nops(1+x+x^2);
```

$$3$$

```
> op(poly);
```

$$1,\ x,\ x^2$$

```
> op(1,poly);
```

$$1$$

```
> op(3,poly);
```

$$x^2$$

```
> op(2,op(3,poly));
```

$$2$$

```
> op(2 .. 3,poly);
```

$$x,\ x^2$$

Substitution:

```
> subs(x=2,x^2+3*x-3);
```

$$7$$

subs führt syntaktische, keine algebraischen Substitutionen durch:

```
> subs(a+b=x,a+b+c);
```

$$a + b + c$$

Es werden nur die Operanden eines Ausdrucks ersetzt; a+b ist kein Operand des Ausdrucks a+b+c:

```
> op(a+b+c);
```

$$a,\ b,\ c$$

Für algebraische Substitutionen steht das allgemeinere Kommando **algsubs** zur Verfügung:

```
> algsubs(a+b=x,a+b+c);
```

$$x + c$$

subs führt keine Vereinfachungen oder Auswertungen durch, algsubs hingegen wohl:

```
> subs(y=ln(x),exp(y));
```

$$e^{\ln(x)}$$

```
> eval(%);
```

$$x$$

```
> algsubs(y=ln(x),exp(y));
```

$$x$$

subsop ermöglicht die Substitution von Operanden in Ausdrücken:

```
> expr:=x^2;
```

$$expr := x^2$$

```
> subsop(2=3,expr);
```

$$x^3$$

2.9 Häufig gestellte Fragen

```
> restart:
```

Wie kann ich einen Wert für das Produkt von zwei Variablen substituieren? Durch Anwendung von simplify mit einer Nebenbedingung. Beispiel:

```
> expr1:=a^3*b^2;
```

$$expr1 := a^3\,b^2$$

```
> subs(a*b=5,expr1);
```

$$a^3\,b^2$$

```
> simplify(expr1,{a*b=5});
```

$$25\,a$$

Liefert simplify nicht die einfachste Form für einen Ausdruck, so erzielt simplify mit Nebenbedingung(en) oft das gewünschte Resultat.

```
> simplify(1-sin(x)^2);
```

$$\cos(x)^2$$

```
> simplify(1-cos(x)^2);
```

$$1-\cos(x)^2$$

```
> simplify(1-cos(x)^2,{1-cos(x)^2=sin(x)^2});
```

$$\sin(x)^2$$

Wie kann der konstante Faktor in den Ausdrücken 2x+2y oder -x-y ausgeklammert werden? Dies ist derzeit in Maple nicht möglich! Warum? Nun, weil für Maple Summen einfacher als Produkte sind (Design-Entscheidung in Maple: Konstante Zahlen als Faktoren werden automatisch auf alle Terme eines Ausdrucks verteilt). Für viele Fälle trifft dies auch zu, z. B.

```
> x^19 - x;
```

$$x^{19}-x$$

ist einfacher als

```
> factor(x^19-x);
```

$$x\,(x-1)\,(x^2+x+1)\,(x^6+x^3+1)\,(x+1)\,(1-x+x^2)\,(1-x^3+x^6)$$

-x-y ist für Maple somit einfacher als (-1)(x+y).

```
> -1*(x+y);
```

$$-x-y$$

Erzwungene Ausklammerung:

```
> expr2:=-x-y;
```

$$expr2 := -x-y$$

```
> subs(-1=vorz,expr2);
```

$$x\;vorz + y\;vorz$$

```
> factor(%);
```

$$vorz\,(x+y)$$

2.10 Fallstudie: Vollständige Induktion

```
> restart:
```

Wir setzen nun Maple zur Lösung eines mathematischen Problems ein, und zwar zum Beweis von Summenformeln (**sum**) mit vollständiger Induktion. Ist z.B. zu beweisen, daß

```
> Sum(f(k),k=1..m) = g(m);
```

$$\sum_{k=1}^{m} \mathrm{f}(k) = \mathrm{g}(m)$$

gilt, so muß man nur f und g definieren, der Induktionsbeweis wird dann weitgehend automatisch von Maple erledigt. Beispiel:

```
> f:=k->k/2^k;
```

$$f := k \to \frac{k}{2^k}$$

```
> g:=m->2-(m+2)/2^m;
```

$$g := m \to 2 - \frac{m+2}{2^m}$$

```
> Behauptung:=Sum(f(k),k=1 .. m) = g(m);
```

$$Behauptung := \sum_{k=1}^{m} \frac{k}{2^k} = 2 - \frac{m+2}{2^m}$$

Induktionsanfang: m = 1

```
> subs(m=1,Behauptung);
```

$$\sum_{k=1}^{1} \frac{k}{2^k} = \frac{1}{2}$$

```
> value (%);
```

$$\frac{1}{2} = \frac{1}{2}$$

Induktionsannahme: m = n

```
> Annahme:=subs(m=n,Behauptung);
```

$$Annahme := \sum_{k=1}^{n} \frac{k}{2^k} = 2 - \frac{n+2}{2^n}$$

Induktionsschluß: m = n+1

```
> Schluss:=Sum(f(k),k=1..n+1) = lhs(Annahme) + f(n+1);
```

$$Schluss := \sum_{k=1}^{n+1} \frac{k}{2^k} = \left(\sum_{k=1}^{n} \frac{k}{2^k}\right) + \frac{n+1}{2^{(n+1)}}$$

Also folgt aus der Annahme

```
> lhs(Schluss) = rhs(Annahme) + f(n+1);
```

$$\sum_{k=1}^{n+1} \frac{k}{2^k} = 2 - \frac{n+2}{2^n} + \frac{n+1}{2^{(n+1)}}$$

```
> lhs(Schluss) = op(1,rhs(%)) +
> simplify(normal(op(2,rhs(%))+op(3,rhs(%))));
```

$$\sum_{k=1}^{n+1} \frac{k}{2^k} = 2 - 2^{(-n-1)}\,(n+3)$$

qed.

Für Identitäten mit endlichen Produkten (**product**) läuft der Induktionsbeweis analog.

Kapitel 3

Grundlagen der Analysis

3.1 Folgen und Grenzwerte

```
> restart:
```

Eine Funktion f := IN -> M nennt man Folge mit Werten in M. Ist M=IR, so spricht man von einer reellen Zahlenfolge. Gegeben seien die Folgen

```
> a:=n -> n^2/(2*n^2 +4*n - 1);
```

$$a := n \to \frac{n^2}{2\,n^2 + 4\,n - 1}$$

und

```
> b:=n -> n*sin(1/n);
```

$$b := n \to n \sin(\frac{1}{n})$$

Endlich viele Folgenglieder können mit der Funktion **seq** berechnet werden.

```
> seq(a(n),n=1..10);
```

$$\frac{1}{5}, \frac{4}{15}, \frac{9}{29}, \frac{16}{47}, \frac{25}{69}, \frac{36}{95}, \frac{49}{125}, \frac{64}{159}, \frac{81}{197}, \frac{100}{239}$$

```
> seq(evalf(b(n)),n=5 ..9);
```

$$.9933466540,\ .9953767962,\ .9966021086,\ .9973978672,\ .9979436565$$

Die Grenzwerte dieser unendlichen Folgen lassen sich in Maple mit **limit** berechnen:

```
> Limit(a(n),n=infinity) = limit(a(n),n=infinity);
```

$$\lim_{n\to\infty} \frac{n^2}{2\,n^2 + 4\,n - 1} = \frac{1}{2}$$

```
> Limit(b(n),n=infinity) = limit(b(n),n=infinity);
```

$$\lim_{n\to\infty} n \sin(\frac{1}{n}) = 1$$

Die ersten zwanzig Glieder der Folge b werden zusammen mit ihrem jeweiligen Index als Liste mit 2-elementigen Listen abgespeichert, diese dient dann zur graphischen Anzeige der Folge:

```
> points:=[seq([i,b(i)],i=1..20)]:
> p1:=plot(points,n=0..20,style=point):
```

In einer vorgegebenen ε-Umgebung

```
> epsilon:=10^(-3);
```

$$\varepsilon := \frac{1}{1000}$$

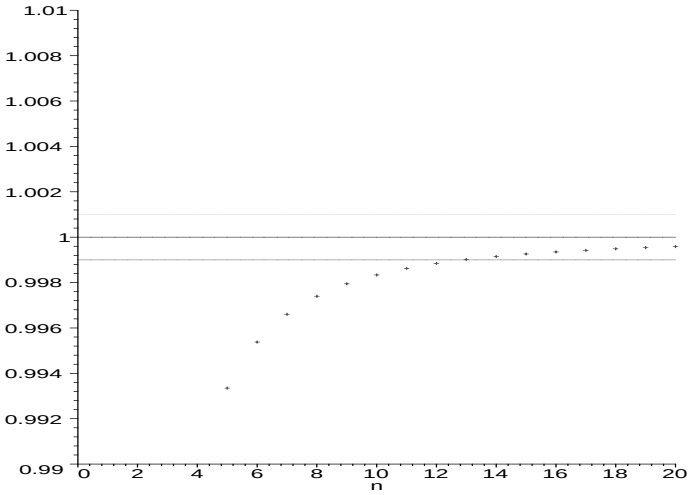
um den Grenzwert von b

```
> q:=limit(b(n),n=infinity);
```

$$q := 1$$

befinden sich dann alle bis auf endlich viele Folgenglieder (Definition des Grenzwertes einer unendlichen Folge).

```
> p2:=plot({q-epsilon,q,q+epsilon
> },n=0..20,q-10*epsilon..q+10*epsilon):
> plots[display]({p1,p2});
```



Die nächste Folge hat keinen Grenzwert, dafür aber zwei Häufungspunkte.

```
> c:=n -> (-1)^n*(n+2)/(n+cos(n));
```

$$c := n \to \frac{(-1)^n (n+2)}{n + \cos(n)}$$

Mehrere Häufungspunkte einer Folge werden von Maple durch den Bereichsoperator a..b (**range**) angezeigt.

```
> Limit(c(n),n=infinity) = limit(c(n),n=infinity);
```

$$\lim_{n\to\infty} \frac{(-1)^n (n+2)}{n + \cos(n)} = -1..1$$

Bei divergenten Folgen meldet Maple ∞ bzw. $-\infty$:

```
> d[1]:=n -> 2^n;
```

$$d_1 := n \to 2^n$$

```
> Limit(d[1](n),n=infinity) = limit(d[1](n),n=infinity);
```

$$\lim_{n\to\infty} 2^n = \infty$$

```
> d[2]:=n -> -2^n;
```

$$d_2 := n \to -2^n$$

```
> Limit(d[2](n),n=infinity) = limit(d[2](n),n=infinity);
```

$$\lim_{n\to\infty} -2^n = -\infty$$

3.2 Grenzwerte bei Funktionen, Stetigkeit

```
> restart:
```

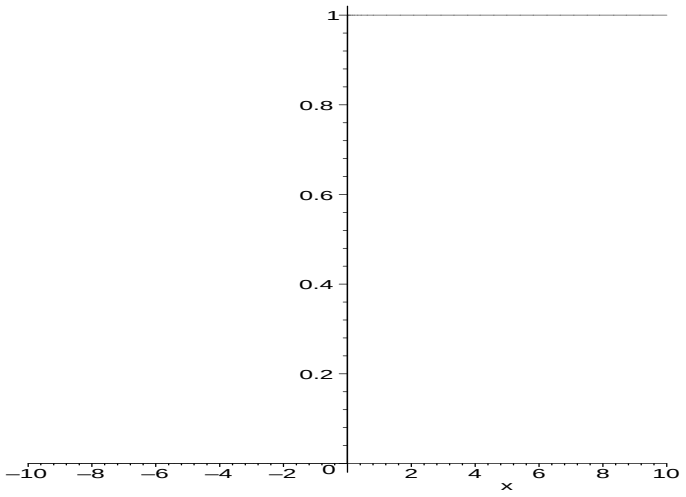
Wir betrachten die Stufenfunktion **Heaviside** und führen die Abkürzung

```
> alias(h=Heaviside);
```

$$h$$

ein. Mit dem **alias** Kommando kann der Benutzer die Maple-Notation seinen Bedürfnissen ensprechend anpassen. Die Heaviside-Funktion ist eine Stufenfunktion

```
> plot(h(x),x=-10..10);
```



die wir an einigen Stellen untersuchen wollen, d.h wir interessieren uns für Grenzwerte der Form

```
> Limit(h(x),x=a);
```

$$\lim_{x \to a} \mathrm{h}(x)$$

Für $x \to -\infty$ bzw. $x \to \infty$ erhalten wir

```
> Limit(h(x),x=-infinity): % = value(%);
```

$$\lim_{x \to (-\infty)} \mathrm{h}(x) = 0$$

```
> Limit(h(x),x=+infinity): % = value(%);
```

$$\lim_{x \to \infty} \mathrm{h}(x) = 1$$

An der Stelle x=0 befindet sich eine Unstetigkeitsstelle:

```
> Limit(h(x),x=0): % = value(%);
```

$$\lim_{x \to 0} \mathrm{h}(x) = \mathit{undefined}$$

Der linksseite Grenzwert

```
> Limit(h(x),x=0,left): % = value(%);
```

$$\lim_{x \to 0-} \mathrm{h}(x) = 0$$

und der rechtsseitige Grenzwert

```
> Limit(h(x),x=0,right): % = value(%);
```

$$\lim_{x \to 0+} \mathrm{h}(x) = 1$$

stimmen nicht überein. Die Stetigkeit einer Funktion kann auch mit der Prozedur **iscont** überprüft werden:

```
> iscont(h(x),x=-1..1);
```

$$\mathit{false}$$

```
> iscont(h(x),x=0..1);
```

$$\mathit{true}$$

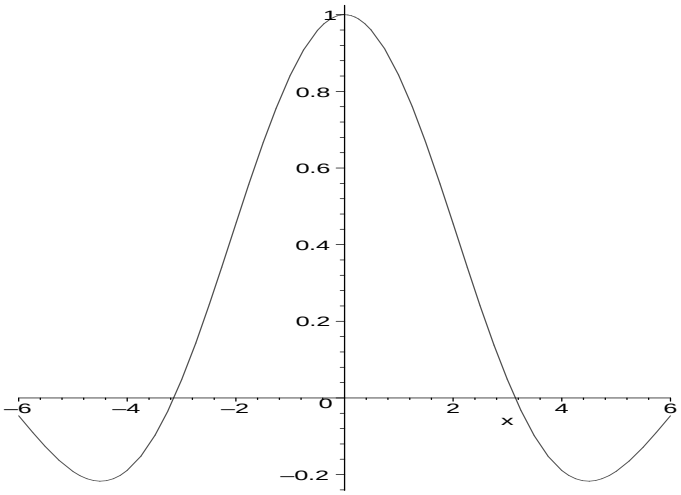
iscont erwartet als zweites Argument ein Intervall.

Wir betrachten nun die Funktion

```
> g:=x -> sin(x)/x;
```

$$g := x \to \frac{\sin(x)}{x}$$

```
> plot(g(x),x=-6..6);
```




```
> g(0);
```

```
Error, (in g) division by zero
```

```
> iscont(g(x),x=-1..1);
```

$$false$$

Die Funktion g kann an der Stelle x=0 nicht ausgewertet werden (Ausdruck 0/0). Der Graph von g ist aber in der Umgebung von 0 offenbar stetig:

```
> Limit(g(x),x=0,left): % = value(%);
```

$$\lim_{x \to 0-} \frac{\sin(x)}{x} = 1$$

```
> Limit(g(x),x=0,right): % = value(%);
```

$$\lim_{x \to 0+} \frac{\sin(x)}{x} = 1$$

Der links- und rechtsseitige Grenzwert für x -> 0 sind gleich, d.h. die Funktion g ist an der Stelle x=0 stetig ergänzbar.

```
> g(0):=1;
```

$$g(0) := 1$$

Durch diese Zuordnung wird ein Eintrag in die **remember table** von g vorgenommen. Wird nun die Funktion (Prozedur) g mit dem Argument 0 aufgerufen, dann wird g nicht an der Stelle 0 ausgewertet, sondern es wird der gespeicherte Funktionswert aus der remember table zurückgegeben. Den Inhalt der remember table erhält man mit

```
> op(4,op(g));
```

$$\mathrm{table}([0 = 1])$$

```
> g(0);
```

$$1$$

Die remember table wird später bei der Einführung von Maple-Prozeduren noch genauer besprochen. Die Rechenregeln für Grenzwerte lassen sich mit der (trägen) Funktion Limit (**student,Limit**) aus dem Maple-Paket **student** und der Funktion expand formal herleiten.

```
> restart:
> with(student):
```

Regel für multiplikative Konstanten:

```
> Limit(c*f(x),x=a): % = expand(%);
```

$$\lim_{x \to a} c\,\mathrm{f}(x) = c\,(\lim_{x \to a} \mathrm{f}(x))$$

Beispiel:

```
> subs(f(x)=x^2,c=2,a=1,%);
```

$$\lim_{x \to 1} 2\,x^2 = 2\,(\lim_{x \to 1} x^2)$$

```
> value(%);
```

$$2 = 2$$

Summenregel:

```
> Limit(f(x)+g(x),x=a): % = expand(%);
```

$$\lim_{x \to a} \mathrm{f}(x) + \mathrm{g}(x) = (\lim_{x \to a} \mathrm{f}(x)) + (\lim_{x \to a} \mathrm{g}(x))$$

Beispiel:

```
> subs(f(x)=x^2,g(x)=x,a=2,%);
```

$$\lim_{x \to 2} x^2 + x = (\lim_{x \to 2} x^2) + (\lim_{x \to 2} x)$$

```
> value(%);
```

$$6 = 6$$

Produktregel:

```
> Limit(f(x)*g(x),x=a): % = expand(%);
```

$$\lim_{x \to a} \mathrm{f}(x)\,\mathrm{g}(x) = (\lim_{x \to a} \mathrm{f}(x))\,(\lim_{x \to a} \mathrm{g}(x))$$

Beispiel:

```
> subs(f(x)=x-2,g(x)=x^2,a=1,%);
```

$$\lim_{x \to 1} (x-2)\,x^2 = (\lim_{x \to 1} x - 2)\,(\lim_{x \to 1} x^2)$$

```
> value(%);
```

$$-1 = -1$$

Quotientenregel (limit(g(x), x=a) $<>$ 0):

```
> Limit(f(x)/g(x),x=a): % = expand(%);
```

$$\lim_{x \to a} \frac{\mathrm{f}(x)}{\mathrm{g}(x)} = \frac{\lim\limits_{x \to a} \mathrm{f}(x)}{\lim\limits_{x \to a} \mathrm{g}(x)}$$

Beispiel:

```
> subs(f(x)=x,g(x)=x^2+1,a=2,%);
```

$$\lim_{x \to 2} \frac{x}{x^2+1} = \frac{\lim\limits_{x \to 2} x}{\lim\limits_{x \to 2} x^2+1}$$

```
> value(%);
```

$$\frac{2}{5} = \frac{2}{5}$$

Regel für Potenzen:

```
> Limit(f(x)^g(x),x=a): % = expand(%);
```

$$\lim_{x \to a} \mathrm{f}(x)^{\mathrm{g}(x)} = (\lim_{x \to a} \mathrm{f}(x))^{(\lim_{x \to a}\ \mathrm{g}(x))}$$

Beispiel:

```
> subs(f(x)=x+2,g(x)=x,a=2,%);
```

$$\lim_{x \to 2} (x+2)^x = (\lim_{x \to 2} x + 2)^{(\lim_{x \to 2}\ x)}$$

```
> value(%);
```

$$16 = 16$$

Regel für Ungleichungen: f(x) < g(x) => lim(f(x), x=a) < lim(g(x), x=a), falls die Grenzwerte existieren. Beispiel:

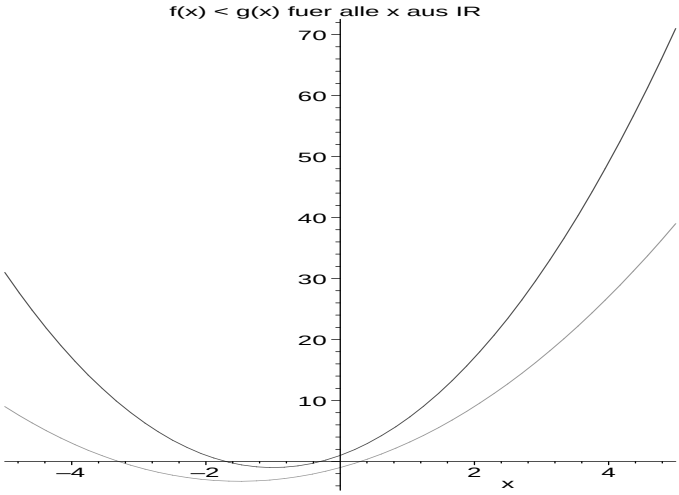
```
> f:=x->x^2+3*x-1;
```

$$f := x \to x^2 + 3\,x - 1$$

```
> g:=x->2*x^2+4*x+1;
```

$$g := x \to 2\,x^2 + 4\,x + 1$$

```
> plot({f(x),g(x)},x=-5..5,title='f(x) < g(x) fuer alle x aus
> IR');
```



Für alle reellen a gilt

```
> f(a) < g(a);
```

$$a^2 + 3\,a < 2\,a^2 + 4\,a + 2$$

Also haben wir für die Grenzwerte

```
> Limit(f(x),x=a) < Limit(g(x),x=a);
```

$$\lim_{x \to a} x^2 + 3\,x - 1 < \lim_{x \to a} 2\,x^2 + 4\,x + 1$$

```
> value(%);
```

$$a^2 + 3\,a < 2\,a^2 + 4\,a + 2$$

```
> subs(a=4,%);
```

$$28 < 50$$

3.3 Differentiation

```
> restart:
```

Zunächst wollen wir die graphische Bedeutung der Ableitung einer Funktion f an der Stelle x_0 illustrieren. Dazu definieren wir die Funktion

```
> f:=x -> exp(sin(x));
```

$$f := x \to e^{\sin(x)}$$

und bestimmen die Steigung der Tangente an die Kurve von f im Punkt [x_0, f(x_0)] mit x_0=1.

```
> x[0]:=1;
```

$$x_0 := 1$$

Wir betrachten die Punkte

```
> p[0]:=[x[0],f(x[0])];
```

$$p_0 := [1,\ e^{\sin(1)}]$$

und

```
> p[1]:=[x[0] + h,f(x[0] + h)];
```

$$p_1 := [1 + h,\ e^{\sin(1+h)}]$$

auf dem Graphen von f. Die Steigung der Sekante durch p_0 und p_1 wird mit dem Kommando slope (**student,slope**) aus dem Maple-Paket student bestimmt

```
> with(student):
> m:=slope(p[0],p[1]);
```

$$m := -\frac{e^{\sin(1)} - e^{\sin(1+h)}}{h}$$

Z.B. ist die Steigung für h=1

```
> subs(h=1,m);
```

$$-e^{\sin(1)} + e^{\sin(2)}$$

```
> evalf(%);
```

$$.162800903$$

Die Sekante wird durch folgende lineare Gleichung beschrieben:

```
> y - p[0][2] = m*(x - p[0][1]);
```

$$y - e^{\sin(1)} = -\frac{(e^{\sin(1)} - e^{\sin(1+h)})\,(x-1)}{h}$$

Mit dem **isolate** Kommando isolieren wir y auf der linken Seite der Gleichung und erhalten

```
> isolate(%,y);
```

$$y = -\frac{(e^{\sin(1)} - e^{\sin(1+h)})\,(x-1)}{h} + e^{\sin(1)}$$

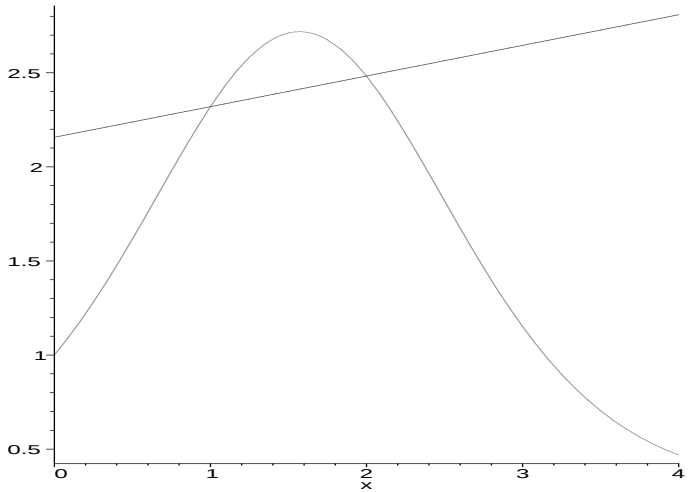
Die Sekante als Funktion von x und h läßt sich nun wie folgt definieren:

```
> s:=unapply (rhs(%),x,h);
```

$$s := (x,\ h) \to -\frac{(e^{\sin(1)} - e^{\sin(1+h)})\,(x-1)}{h} + e^{\sin(1)}$$

Wir zeichnen jetzt die Funktionen f und s (mit h=1) in ein gemeinsames Schaubild.

```
> plot({f(x),s(x,1)},x=0..4);
```



Die Schnittpunkte von f und s sind hier gegeben durch

```
> 'p[0]' = evalf(p[0]);
```

$$p_0 = [1.,\ 2.319776825]$$

und

```
> 'p[1]' = evalf(subs(h=1,p[1]));
```

$$p_1 = [2., 2.482577728]$$

Offenbar konvergiert die Steigung m für $h \to 0$

```
> h_values:=[seq(1/i,i=1..15)];
```

$$h\_values := [1, \frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \frac{1}{5}, \frac{1}{6}, \frac{1}{7}, \frac{1}{8}, \frac{1}{9}, \frac{1}{10}, \frac{1}{11}, \frac{1}{12}, \frac{1}{13}, \frac{1}{14}, \frac{1}{15}]$$

```
> seq(evalf(m),h=h_values);
```

$$.162800903,\ .783408386,\ .969854001,\ 1.053234750,\ 1.09952854,\ 1.12872746,$$
$$1.14874064,\ 1.16327997,\ 1.17430579,\ 1.18294680,\ 1.18989715,\ 1.19560651,$$
$$1.20037863,\ 1.20442589,\ 1.20790129$$

Die Konvergenz der Sekantensteigung m wollen wir als Folge von 2D-Plots graphisch darstellen.

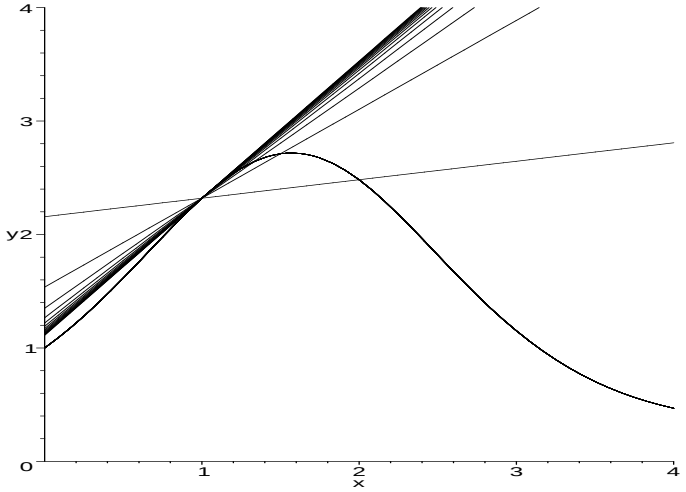
```
> S:=seq(plot({f(x),s(x,h)},x=0..4,y=0..4,color=black),
> h=h_values):
```

Mit dem display Kommando aus dem Maple-Paket plots (**plots,display**) läßt sich diese Plotfolge in ein Schaubild zeichnen.

```
> with(plots):
```

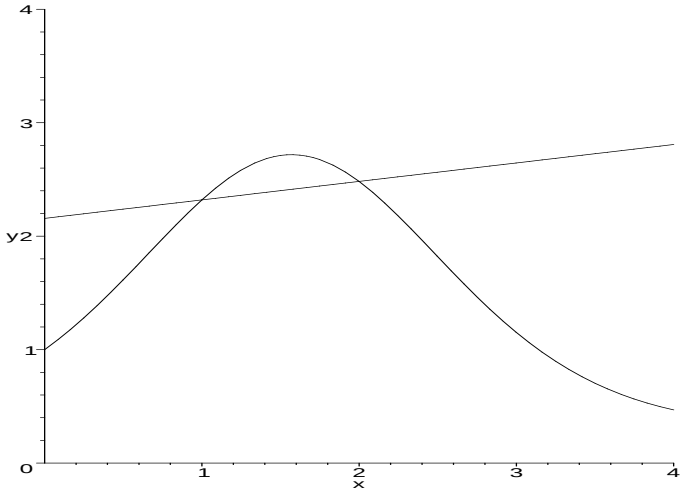
```
Warning, the name changecoords has been redefined
```

```
> display([S]);
```



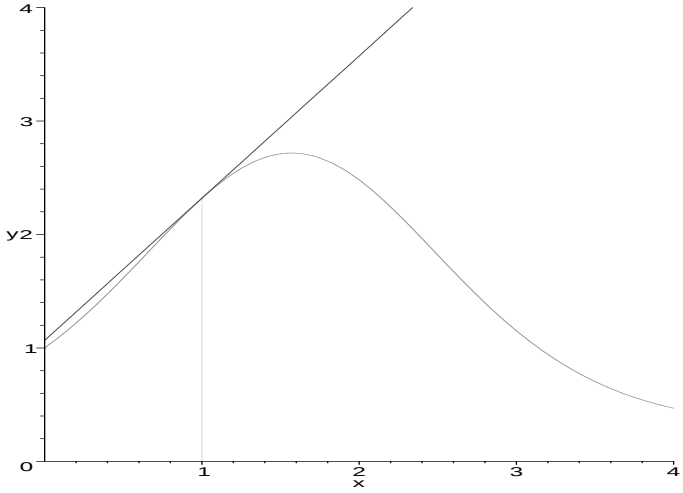
Die Option insequence=true ermöglicht die Animation dieser Plotfolge.

```
> display([S],insequence=true);
```



Mit dem Kommando **showtangent** aus dem Paket student läßt sich zu einer gegebenen Stelle x die Tangente an die Kurve f zeichnen.

```
> showtangent(f(x),x=1,x=0..4,y=0..4);
```



Im Grenzwert $h \to 0$ ist die Steigung

```
> Limit(m,h=0);
```

$$\lim_{h \to 0} -\frac{e^{\sin(1)} - e^{\sin(1+h)}}{h}$$

```
> value(%);
```

$$e^{\sin(1)} \cos(1)$$

Dies ist natürlich die Ableitung von f an der Stelle x_0=1.

```
> D(f)(x[0]);
```

$$e^{\sin(1)} \cos(1)$$

Allgemein ist die Ableitung von f an der Stelle x_0 definiert durch den folgenden Grenzwert:

```
> f:='f': x[0]:='x[0]':
> Limit((f(x[0]+h)-f(x[0]))/h,h=0);
```

$$\lim_{h \to 0} \frac{\mathrm{f}(x_0 + h) - \mathrm{f}(x_0)}{h}$$

```
> value(%);
```

$$\mathrm{D}(f)(x_0)$$

Die Ableitung als Funktion von x erhält man mit dem **D**-Operator.

```
> f:=x -> exp(sin(x));
```

$$f := x \to e^{\sin(x)}$$

```
> 'f'':=D(f);
```

$$f' := x \to \cos(x)\, e^{\sin(x)}$$

Die Differentiation eines Maple-Ausdrucks (einer Formel) erfolgt mit dem Kommando **diff**

```
> Diff (f(x),x) = diff (f(x),x);
```

$$\frac{\partial}{\partial x} e^{\sin(x)} = \cos(x)\, e^{\sin(x)}$$

Diff ist hier wieder die träge Form des diff-Kommandos. Die Ableitungsfunktion erhält man hieraus mittels

```
> f1:=unapply(rhs(%),x);
```

$$f1 := x \to \cos(x)\, e^{\sin(x)}$$

Die zweite Ableitung von f erhalten wir durch Differenzieren von f1

```
> diff(f1(x),x);
```

$$-\sin(x)\, e^{\sin(x)} + \cos(x)^2\, e^{\sin(x)}$$

oder direkt mit

```
> diff(f(x),x,x);
```

$$-\sin(x)\,e^{\sin(x)} + \cos(x)^2\,e^{\sin(x)}$$

oder durch

```
> diff(f(x),x$2);
```

$$-\sin(x)\,e^{\sin(x)} + \cos(x)^2\,e^{\sin(x)}$$

Hierbei ist

```
> x$2;
```

$$x,\ x$$

Mit Hilfe des D-Operators läßt sich die zweite Ableitung von f durch

```
> D(D(f))(x);
```

$$-\sin(x)\,e^{\sin(x)} + \cos(x)^2\,e^{\sin(x)}$$

oder unter Verwendung des Operators **@** für die Komposition (Hintereinanderschaltung) von Funktionen

```
> (D@@2)(f)(x);
```

berechnen.

$$-\sin(x)\,e^{\sin(x)} + \cos(x)^2\,e^{\sin(x)}$$

3.4 Differentiationsregeln

```
> restart:
```

Wir wollen nun die bekannten Differentiationsregeln mit Hilfe der Maple-Funktion **limit** ableiten. Zunächst betrachten wir die einfache Potenzfunktion

```
> f:=x -> x^n;
```

$$f := x \to x^n$$

und berechnen den Grenzwert des Differenzenquotienten für $x \to x_0$ bzw. $h \to 0$.

```
> Limit((f(x[0]+h)-f(x[0]))/h,h=0): % = simplify(value(%));
```

$$\lim_{h\to 0} \frac{(x_0+h)^n - {x_0}^n}{h} = {x_0}^{(n-1)}\,n$$

Für die Potenzfunktion kennt die limit Funktion also den Grenzwert des Differenzenquotienten. Allgemein gilt für x aus IR

```
> Diff(f(x),x) = simplify(diff(f(x),x));
```

$$\frac{\partial}{\partial x}\,x^n = x^{(n-1)}\,n$$

Die Ableitungsfunktion ergibt sich daraus wie folgt:

```
> 'f'':=unapply(rhs(%),x);
```

$$f' := x \to x^{(n-1)}\,n$$

Nun betrachten wir unbestimmte Funktionen auf IR. Seien die Funktionen f und g überall differenzierbar. Dann gilt für die Funktion f+g an der Stelle x_0

```
> f:='f':
> Limit((f(x[0]+h)+g(x[0]+h)-f(x[0])-g(x[0]))/h,h=0): % =
> value(%);
```

$$\lim_{h\to 0} \frac{\mathrm{f}(x_0+h) + \mathrm{g}(x_0+h) - \mathrm{f}(x_0) - \mathrm{g}(x_0)}{h} = \mathrm{D}(f)(x_0) + \mathrm{D}(g)(x_0)$$

Limit kennt also auch den Grenzwert des Differenzenquotienten für die Summenfunktion f+g. Allgemein haben wir die folgende Summenregel

```
> 'D(f+g)(x)' = D(f+g)(x);
```

$$\mathrm{D}(f+g)(x) = \mathrm{D}(f)(x) + \mathrm{D}(g)(x)$$

oder in der Leibniz-Notation

```
> Diff((f+g)(x),x): % = value(%);
```

$$\frac{\partial}{\partial x}\left(\mathrm{f}(x)+\mathrm{g}(x)\right)=\left(\frac{\partial}{\partial x}\,\mathrm{f}(x)\right)+\left(\frac{\partial}{\partial x}\,\mathrm{g}(x)\right)$$

Für die Ableitung der Funktion f*g an der Stelle x_0 gilt

```
> Limit((f(x[0]+h)*g(x[0]+h)-f(x[0])*g(x[0]))/h,h=0): % =
> value(%);
```

$$\lim_{h\to 0}\frac{\mathrm{f}(x_0+h)\,\mathrm{g}(x_0+h)-\mathrm{f}(x_0)\,\mathrm{g}(x_0)}{h}=\mathrm{f}(x_0)\,\mathrm{D}(g)(x_0)+\mathrm{D}(f)(x_0)\,\mathrm{g}(x_0)$$

Allgemein gilt die Produktregel

```
> 'D(f*g)(x)' = D(f*g)(x);
```

$$\mathrm{D}(f\,g)(x)=\mathrm{D}(f)(x)\,\mathrm{g}(x)+\mathrm{f}(x)\,\mathrm{D}(g)(x)$$

Die Ableitung der Funktion f/g an der Stelle x_0 ($\mathrm{g}(x_0)\neq 0$) läßt sich mit der Formel

```
> Limit((f(x[0]+h)/g(x[0]+h)-f(x[0])/g(x[0]))/h,h=0): % =
> value(%);
```

$$\lim_{h\to 0}\frac{\dfrac{\mathrm{f}(x_0+h)}{\mathrm{g}(x_0+h)}-\dfrac{\mathrm{f}(x_0)}{\mathrm{g}(x_0)}}{h}=\frac{\mathrm{D}(f)(x_0)\,\mathrm{g}(x_0)-\mathrm{f}(x_0)\,\mathrm{D}(g)(x_0)}{\mathrm{g}(x_0)^2}$$

berechnen. Allgemein gilt die Quotientenregel

```
> 'D(f/g)(x)' = normal(D(f/g)(x));
```

$$\mathrm{D}(\frac{f}{g})(x)=-\frac{-\mathrm{D}(f)(x)\,\mathrm{g}(x)+\mathrm{f}(x)\,\mathrm{D}(g)(x)}{\mathrm{g}(x)^2}$$

Für die Ableitung der geschachtelten Funktion f@g an der Stelle x_0 gilt

```
> Limit((f(g(x+h))-f(g(x)))/h,h=0): % = value(%);
```

$$\lim_{h\to 0}\frac{\mathrm{f}(\mathrm{g}(x+h))-\mathrm{f}(\mathrm{g}(x))}{h}=\mathrm{D}(f)(\mathrm{g}(x))\,\mathrm{D}(g)(x)$$

und somit allgemein die Kettenregel

```
> 'D(f@g)(x)' = D(f@g)(x);
```

$$\mathrm{D}(f@g)(x)=\mathrm{D}(f)(\mathrm{g}(x))\,\mathrm{D}(g)(x)$$

3.5 Kurvendiskussion

```
> restart:
```

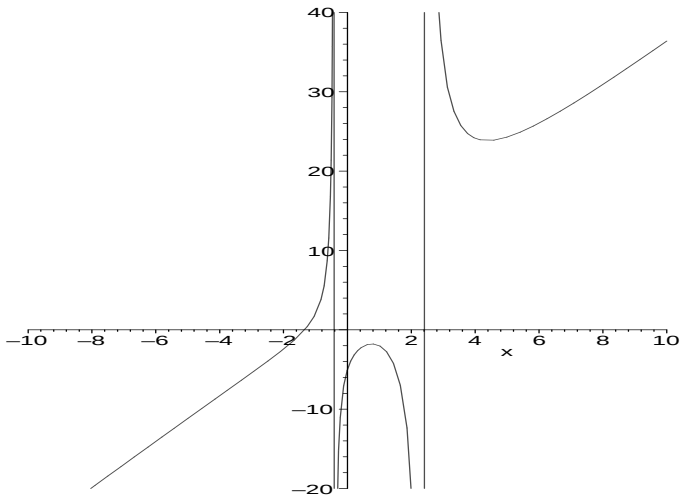
Mit den Hilfsmitteln der Differentialrechnung lassen sich die Eigenschaften und Kennzeichen nichtlinearer Kurven bestimmen. Als Beispiel wollen wir den Graph der rationalen Funktion

```
> f:=x -> (3*x^3-x^2-3*x+5) / (x^2-2*x-1);
```

$$f:=x\to\frac{3\,x^3-x^2-3\,x+5}{x^2-2\,x-1}$$

diskutieren. Zunächst plotten wir die Funktion f.

```
> plot(f(x),x=-10..10,-20 .. 40);
```



Der Funktionsgraph besteht offenbar aus drei Teilstücken, die durch Asymptoten begrenzt werden. Es gibt zwei Polstellen (vertikale Asymptoten). Durch Berechnung der Nullstellen des Nennerpolynoms (**denom**) bestimmen wir diese Polstellen.

```
> Pole:=solve(denom(f(x)) = 0,x);
```

$$Pole := 1 + \sqrt{2},\ 1 - \sqrt{2}$$

```
> evalf(Pole);
```

$$2.414213562,\ -.414213562$$

Neben den beiden vertikalen Asymptoten (Polstellen) gibt es noch eine weitere nicht-vertikale Asymptote, die sich durch Berechnung des Quotienten (**quo**) aus Zähler- und Nennerpolynom ermitteln läßt.

```
> q:=quo(numer(f(x)),denom(f(x)),x,'r');
```

$$q := 3\,x + 5$$

Der Rest der Polynomdivision steht in der Variablen

```
> r;
```

$$10 + 10\,x$$

zur Verfügung, spielt aber für die weiteren Betrachtungen keine Rolle.

```
> q:=unapply(q,x);
```

$$q := x \to 3\,x + 5$$

Die rationale Funktion f verhält sich also für große x wie die lineare Funktion q. Diese Tatsache wird auch durch die asymptotische Entwicklung von f(x) bestätigt. Die Anwendung des Maple-Kommandos **asympt** liefert nämlich

```
> asympt(f(x),x);
```

$$3\,x + 5 + \frac{10}{x} + \frac{30}{x^2} + \frac{70}{x^3} + \frac{170}{x^4} + \frac{410}{x^5} + \mathrm{O}(\frac{1}{x^6})$$

Mit dem limit Kommando läßt sich das asymptotische Verhalten von f(x) für große x verifizieren.

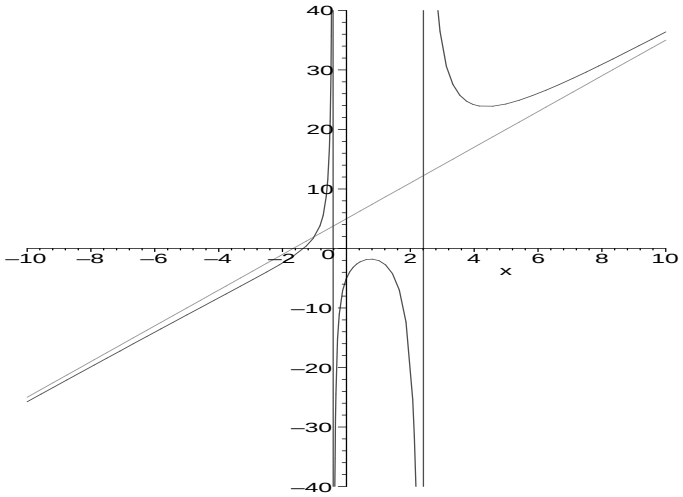
```
> Limit('f(x)/q(x)',x=infinity) = limit(f(x)/q(x),x=infinity);
```

$$\lim_{x\to\infty} \frac{\mathrm{f}(x)}{\mathrm{q}(x)} = 1$$

```
> Limit('f(x)/q(x)',x=-infinity) = limit(f(x)/q(x), x=-infinity);
```

$$\lim_{x\to(-\infty)} \frac{\mathrm{f}(x)}{\mathrm{q}(x)} = 1$$

```
> plot({f(x),q(x)},x=-10..10,-40 .. 40);
```



Nun bestimmen wir die Schnittstelle des Graphen mit der x-Achse. Dazu müssen die Nullstellen der Funktion f berechnet werden.

```
> f_null:=simplify([solve(f(x) = 0,x)]);
```

$$f\_null := \left[-\frac{1}{9}\,\frac{(566+18\sqrt{921})^{(2/3)}+28-(566+18\sqrt{921})^{(1/3)}}{(566+18\sqrt{921})^{(1/3)}},\right.$$
$$-\frac{1}{18}\,\frac{-\%1^{(2/3)}-28-2\,\%1^{(1/3)}+I\sqrt{3}\,\%1^{(2/3)}-28\,I\sqrt{3}}{\%1^{(1/3)}},$$
$$\left.\frac{1}{18}\,\frac{\%1^{(2/3)}+28+2\,\%1^{(1/3)}+I\sqrt{3}\,\%1^{(2/3)}-28\,I\sqrt{3}}{\%1^{(1/3)}}\right]$$
$$\%1 := 566+18\sqrt{3}\sqrt{307}$$

Zwei dieser Lösungen sind komplex:

```
> evalf(f_null);
```

$$[-1.340384212,\ .8368587728-.7369477378\,I,\ .8368587728+.7369477378\,I]$$

Der Schnittpunkt des Graphen mit der x-Achse wird durch die reelle Nullstelle bestimmt.

```
> pts:=x -> [x, f(x)]:
> simplify(pts(f_null[1]));
```

$$\left[-\frac{1}{9}\,\frac{(566+18\sqrt{921})^{(2/3)}+28-(566+18\sqrt{921})^{(1/3)}}{(566+18\sqrt{921})^{(1/3)}},\ 0\right]$$

```
> evalf(%);
```

$$[-1.340384212,\ 0.]$$

Der Schnittpunkt des Funktionsgraphen mit der y-Achse ist durch

```
> pts(0);
```

$$[0,\ -5]$$

gegeben. Die Asymptote q schneidet die Funktion f im Punkt

```
> pts(solve(f(x) = q(x),x));
```

$$[-1,\ 2]$$

Nun bestimmen wir die relativen Extrema und Wendepunkte des Funktionsgraphen. Hierzu benötigen wir die ersten drei Ableitungen der Funktion f.

```
> diff(f(x),x);
```

$$\frac{9\,x^2-2\,x-3}{x^2-2\,x-1}-\frac{(3\,x^3-x^2-3\,x+5)\,(2\,x-2)}{(x^2-2\,x-1)^2}$$

```
> f1:=unapply(normal(%),x);
```

$$f1 := x \rightarrow \frac{3\,x^4 - 12\,x^3 - 4\,x^2 - 8\,x + 13}{(x^2 - 2\,x - 1)^2}$$

```
> diff(f(x),x$2);
```

$$\frac{18\,x - 2}{x^2 - 2\,x - 1} - \frac{2\,(9\,x^2 - 2\,x - 3)\,(2\,x - 2)}{(x^2 - 2\,x - 1)^2} + \frac{2\,(3\,x^3 - x^2 - 3\,x + 5)\,(2\,x - 2)^2}{(x^2 - 2\,x - 1)^3} - \frac{2\,(3\,x^3 - x^2 - 3\,x + 5)}{(x^2 - 2\,x - 1)^2}$$

```
> f2:=unapply(normal(%),x);
```

$$f2 := x \rightarrow 20\,\frac{x^3 + 3\,x^2 - 3\,x + 3}{(x^2 - 2\,x - 1)^3}$$

```
> diff(f(x),x$3);
```

$$18\,\frac{1}{x^2 - 2\,x - 1} - \frac{3\,(18\,x - 2)\,(2\,x - 2)}{(x^2 - 2\,x - 1)^2} + \frac{6\,(9\,x^2 - 2\,x - 3)\,(2\,x - 2)^2}{(x^2 - 2\,x - 1)^3} - \frac{6\,(9\,x^2 - 2\,x - 3)}{(x^2 - 2\,x - 1)^2} - \frac{6\,(3\,x^3 - x^2 - 3\,x + 5)\,(2\,x - 2)^3}{(x^2 - 2\,x - 1)^4} + \frac{12\,(3\,x^3 - x^2 - 3\,x + 5)\,(2\,x - 2)}{(x^2 - 2\,x - 1)^3}$$

```
> f3:=unapply(normal(%),x);
```

$$f3 := x \rightarrow -60\,\frac{-7 + 12\,x + 4\,x^3 - 6\,x^2 + x^4}{(x^2 - 2\,x - 1)^4}$$

Notwendig für das Vorhandensein eines Extremums ist das Verschwinden der ersten Ableitung. Der Zähler von f1 ist ein Polynom vierten Grades. Da die expliziten Lösungen einer polynomialen Gleichung vierten Grades im allgemeinen sehr komplex sind, werden sie in Maple nur dann ausgegeben, wenn man die globale Variable _EnvExplicit vorher auf true setzt (**solve**).

```
> _EnvExplicit:=true;
```

$$\_EnvExplicit := true$$

```
> solve(f1(x) = 0,x);
```

$$1 + \frac{1}{2}\sqrt{10} + \frac{1}{6}\sqrt{42 + 24\sqrt{10}},\ 1 + \frac{1}{2}\sqrt{10} - \frac{1}{6}\sqrt{42 + 24\sqrt{10}},\ 1 - \frac{1}{2}\sqrt{10} + \frac{1}{6}\sqrt{42 - 24\sqrt{10}},\ 1 - \frac{1}{2}\sqrt{10} - \frac{1}{6}\sqrt{42 - 24\sqrt{10}}$$

```
> evalf(%);
```

$$4.390793984,\ .771483676,\ -.581138830 + .9703187314\,I,\ -.581138830 - .9703187314\,I$$

Die erste Ableitung hat also zwei reelle Nullstellen, die wir mit dem **select** Kommando aus der Lösungsmenge selektieren (select erwartet als zweites Argument eine Liste oder eine Menge).

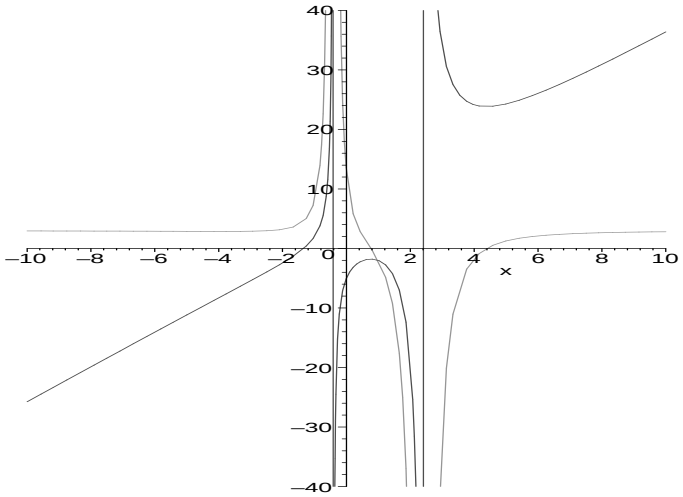
```
> f1_null:=select(type,[%],realcons);
```

$$f1\_null := [4.390793984,\ .771483676]$$

```
> f1_null:=sort(%);
```

$$f1\_null := [.771483676,\ 4.390793984]$$

```
> plot({f1(x),f(x)},x=-10..10,-40 .. 40);
```



Die zweite Ableitung f2 hat an diesen Stellen die Werte

```
> f2(f1_null[1]);
```

$$-7.930860162$$

```
> f2(f1_null[2]);
```

$$3.088974052$$

Also haben wir im Punkt

```
> pts(f1_null[1]);
```

$$[.771483676,\ -1.780433469]$$

ein relatives Maximum und im Punkt

```
> pts(f1_null[2]);
```

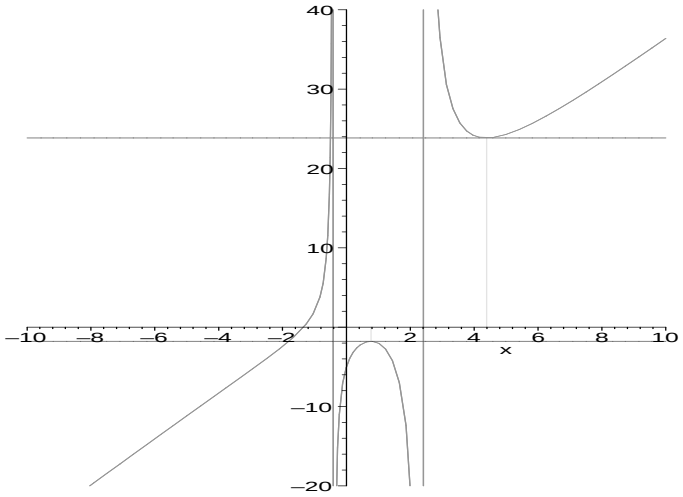
$$[4.390793984,\ 23.84840528]$$

ein relatives Minimum.

```
> with(student): with(plots):
```

```
Warning, the name changecoords has been redefined
```

```
> p1:=showtangent(f(x),x=f1_null[1],x=-10..10,-20 .. 40):
> p2:=showtangent(f(x),x=f1_null[2],x=-10..10,-20 .. 40):
> display({p1,p2});
```



Notwendig für das Vorhandensein eines Wendepunktes ist das Verschwinden der zweiten Ableitung. Die Nullstellen der zweiten Ableitung sind

```
> simplify([solve(f2(x) = 0,x)]);
```

$$\left[-\frac{\%1+2+(4+2\sqrt{2})^{(1/3)}}{(4+2\sqrt{2})^{(1/3)}},\ -\frac{1}{2}\,\frac{-\%1-2+2\,(4+2\sqrt{2})^{(1/3)}+I\sqrt{3}\,\%1-2\,I\sqrt{3}}{(4+2\sqrt{2})^{(1/3)}},\right.$$
$$\left.\frac{1}{2}\,\frac{\%1+2-2\,(4+2\sqrt{2})^{(1/3)}+I\sqrt{3}\,\%1-2\,I\sqrt{3}}{(4+2\sqrt{2})^{(1/3)}}\right]$$
$$\%1 := (4+2\sqrt{2})^{(2/3)}$$

```
> evalf(%);
```

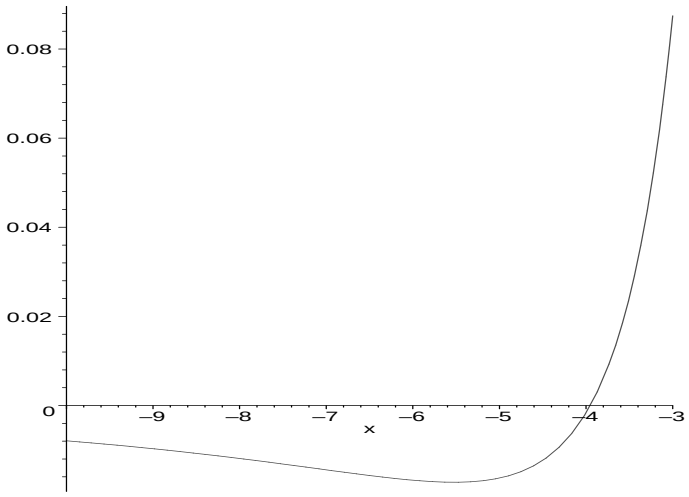
$$[-3.951373036,\ .4756865178-.7300356820\,I,\ .4756865178+.7300356820\,I]$$

Es gibt also nur eine reelle Nullstelle, nämlich

```
> f2_null:=op(select(type,%,realcons));
```

$$f2\_null := -3.951373036$$

```
> plot(f2(x),x=-10..-3);
```



Die dritte Ableitung hat an dieser Stelle den Wert

```
> f3(f2_null);
```

$$.03527228299$$

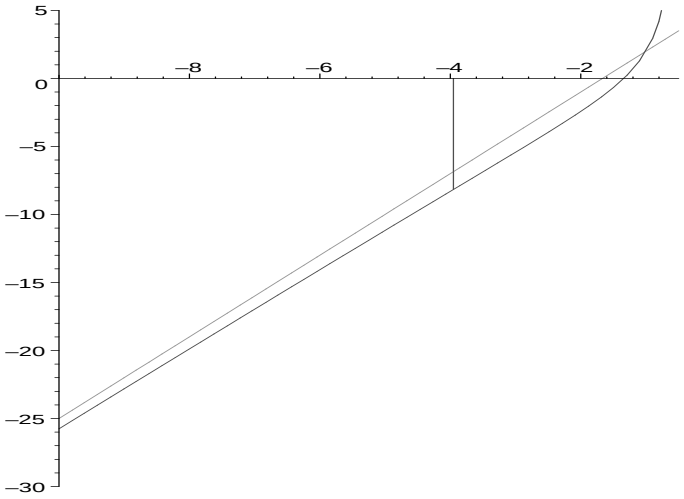
Also haben wir im Punkt

```
> pts(f2_null);
```

$$[-3.951373036,\ -8.164902817]$$

einen Wendepunkt.

```
> p3:=plot([[f2_null,0],pts(f2_null)]):

> p4:=plot({f(x),q(x)},x=-10..-0.5,-30..5):

> display({p3,p4});
```



3.6 Extremwertprobleme

```
> restart:
```

Die Lösung von Extremwertproblemen ist ein weiteres Anwendungsfeld der Differentialrechnung. Es folgt ein einfaches Beispiel: Es soll eine zylindrische Dose aus Blech mit vorgegebenem Inhalt (z.B. 1 Liter) hergestellt werden. Um den Blechverbrauch zu minimieren, wird die Dose mit der kleinsten Oberfläche gesucht.

Oberfläche und Volumen eines Zylinders sind durch die Gleichungen

```
> eqn1:=A = 2*Pi*r^2 + 2*Pi*r*h: %;
```

$$A = 2\,\pi\,r^2 + 2\,\pi\,r\,h$$

und

```
> eqn2:=V = Pi*r^2*h: %;
```

$$V = \pi\,r^2\,h$$

gegeben. Hierbei ist r der Radius und h die Höhe des Zylinders. Wir eliminieren nun die Höhe h in der Gleichung für die Oberfläche mit Hilfe der zweiten Beziehung, der Volumengleichung:

```
> solve(eqn2,{h});
```

$$\{h = \frac{V}{\pi\,r^2}\}$$

```
> assign(%);
```

Mit dem **assign** Kommando wird die Lösung der Variablen h zugewiesen.

```
> h;
```

$$\frac{V}{\pi\,r^2}$$

```
> eqn1;
```

$$A = 2\,\pi\,r^2 + \frac{2\,V}{r}$$

Die Zylinderoberfläche als Funktion von r ist damit gegeben durch

```
> A:=unapply(rhs(%),r);
```

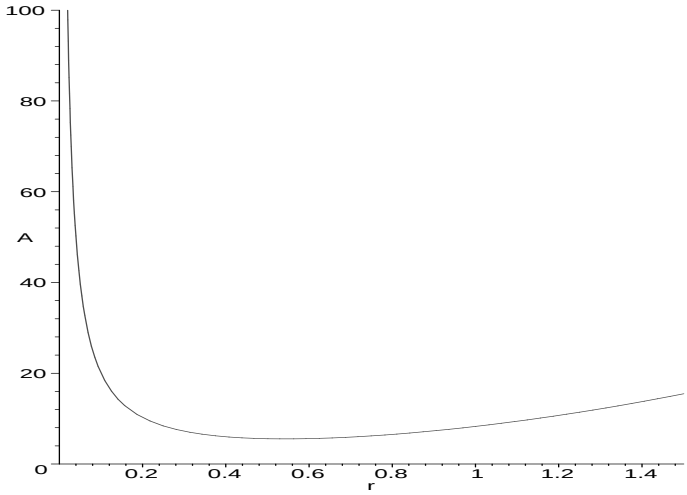
$$A := r \rightarrow 2\,\pi\,r^2 + \frac{2\,V}{r}$$

Bei gegebenem Volumen muß nun die Oberfläche minimiert werden.

```
> V:=1;
```

$$V := 1$$

```
> plot(A(r),r=0..1.5,A=0..100);
```



In der Umgebung von 0.5 befindet sich offenbar ein Minimum. Wir bestimmen nun die Nullstellen der ersten Ableitung.

```
> A1:=D(A);
```

$$A1 := r \to 4\,\pi\,r - \frac{2\,V}{r^2}$$

```
> A1_null:=simplify([solve(A1(r) = 0,r)]);
```

$$A1\_null := \left[\frac{1}{2}\,\frac{2^{(2/3)}}{\pi^{(1/3)}},\ \frac{1}{4}\,\frac{2^{(2/3)}\,(-1+I\,\sqrt{3})}{\pi^{(1/3)}},\ -\frac{1}{4}\,\frac{2^{(2/3)}\,(1+I\,\sqrt{3})}{\pi^{(1/3)}}\right]$$

```
> evalf(%);
```

$$[.5419260700,\ -.2709630350 + .4693217438\,I,\ -.2709630350 - .4693217438\,I]$$

Es existiert eine reelle Nullstelle. Die zweite Ableitung ist an dieser Stelle

```
> D(A1)(A1_null[1]);
```

$$12\,\pi$$

positiv, d.h. der Funktionsgraph hat im Punkt

```
> simplify([A1_null[1],A(A1_null[1])]);
```

$$\left[\frac{1}{2}\,\frac{2^{(2/3)}}{\pi^{(1/3)}},\ 3\,\pi^{(1/3)}\,2^{(1/3)}\right]$$

```
> evalf(%);
```

$$[.5419260700,\ 5.535810447]$$

ein Minimum. Die zylindrische Dose hat also bei

```
> r:=A1_null[1]: 'r' = evalf(%);
```

$$r = .5419260700$$

und

```
> 'h' = evalf(h);
```

$$h = 1.083852140$$

die minimale Oberfläche

```
> A = evalf(A(r));
```

$$A = 5.535810447$$

Kapitel 4

Integralrechnung

4.1 Riemann-Integral

```
> restart:
```

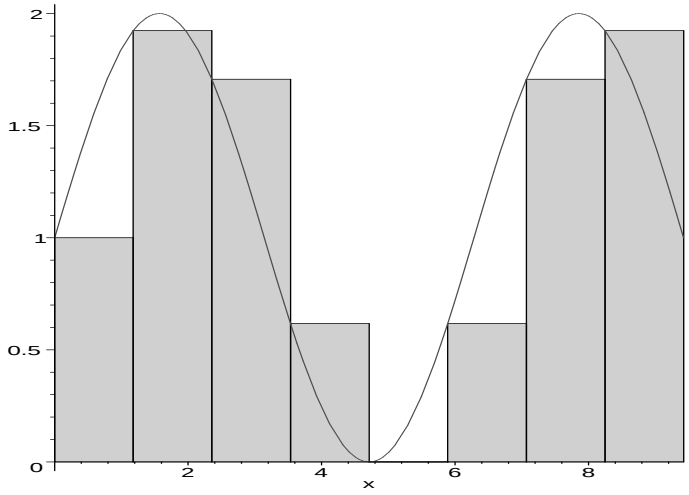
Das Riemann-Integral einer Funktion f mißt die Fläche zwischen der x-Achse und dem Funktionsgraphen. Diese geometrische Bedeutung des Integralbegriffs wollen wir nun mathematisch genauer interpretieren. Dazu definieren wir die Funktion

```
> f:=x -> 1 + sin(x);
```

$$f := x \to 1 + \sin(x)$$

und zeichnen mit dem **leftbox** Kommando aus dem Paket student acht "linksseitige" Rechtecke zwischen dem Graphen von f und der x-Achse.

```
> with(student):
> leftbox(f(x),x=0..3*Pi,8);
```



Mit dem Kommando **leftsum** wird der Flächeninhalt der Rechtecke aufaddiert.

```
> leftsum(f(x),x=0..3*Pi,8);
```

$$\frac{3}{8}\,\pi\,\left(\sum_{i=0}^{7}(1+\sin(\frac{3}{8}\,i\,\pi))\right)$$

Für die Fläche zwischen Kurve und x-Achse gilt also näherungsweise

```
> evalf(%);
```

$$11.18792509$$

Je mehr Rechtecke für die Approximation verwendet werden, desto besser wird die Näherung für die Fläche.

```
> boxes:=[seq(n^2,n=1..12)];
```

$$boxes := [1,\ 4,\ 9,\ 16,\ 25,\ 36,\ 49,\ 64,\ 81,\ 100,\ 121,\ 144]$$

```
> seq(evalf(leftsum(f(x),x=0..3*Pi,n)),n=boxes);
```

$$9.424777962,\ 10.40074568,\ 11.23857733,\ 11.36661104,\ 11.40103461,\ 11.41334174,$$
$$11.41860822,\ 11.42116230,\ 11.42252103,\ 11.42329730,\ 11.42376670,$$
$$11.42406396$$

Die Konvergenz der Rechteckapproximationen wollen wir als Folge von 2D-Plots graphisch darstellen.

```
> S:=seq(leftbox(f(x),x=0..3*Pi,n),n=boxes):
> with(plots):
```

```
Warning, the name changecoords has been redefined
```

```
> display([S],insequence=true);
```

Den Flächeninhalt erhalten wir als Grenzwert der Rechteckapproximationen. Die Approximation mit n Rechtecken liefert die Fläche

```
> A[n]:=leftsum(f(x),x=0..3*Pi,n);
```

$$A_n := 3\,\frac{\pi\,\left(\sum_{i=0}^{n-1}(1+\sin(3\,\frac{i\,\pi}{n}))\right)}{n}$$

```
> combine(simplify(value(%)),trig);
```

$$\frac{3\,\pi\,n\cos(2\,\frac{\pi}{n})-3\,\pi\,n\cos(\frac{\pi}{n})-3\,\pi\sin(2\,\frac{\pi}{n})+3\,\pi\sin(\frac{\pi}{n})}{n\cos(2\,\frac{\pi}{n})-n\cos(\frac{\pi}{n})}$$

Für $n \to \infty$ ergibt sich

```
> Limit(A[n],n=infinity) = limit(%,n=infinity);
```

$$\lim_{n\to\infty} 3\,\frac{\pi\,\left(\sum_{i=0}^{n-1}(1+\sin(3\,\frac{i\,\pi}{n}))\right)}{n} = 3\,\pi+2$$

Nun berechnen wir den Flächeninhalt mittels “rechtsseitiger” Rechteckapproximationen.

```
> B[n]:=rightsum(f(x),x=0..3*Pi,n);
```

$$B_n := 3\,\frac{\pi\,\left(\sum_{i=1}^{n}(1+\sin(3\,\frac{i\,\pi}{n}))\right)}{n}$$

```
> combine(simplify(value(%)),trig);
```

$$(3\,\pi\sin(\frac{\pi\,(2+3\,n)}{n})+3\,\pi\sin(\frac{\pi}{n})+6\,\pi\,n\cos(2\,\frac{\pi}{n})-3\,\pi\sin(\frac{\pi\,(1+3\,n)}{n})-6\,\pi\,n\cos(\frac{\pi}{n})$$
$$-3\,\pi\sin(2\,\frac{\pi}{n}))\Big/(2\,n\cos(2\,\frac{\pi}{n})-2\,n\cos(\frac{\pi}{n}))$$

```
> Limit(B[n],n=infinity) = limit(%,n=infinity);
```

$$\lim_{n\to\infty} 3\,\frac{\pi\,\left(\sum_{i=1}^{n}(1+\sin(3\,\frac{i\,\pi}{n}))\right)}{n} = 3\,\pi+2$$

In entsprechender Weise läßt sich der Flächeninhalt mit middelsum durch “Mittelpunktsrechtecke” approximieren. Der eindeutige Grenzwert (Flächeninhalt) heißt bestimmtes Integral von f über dem Intervall [0, 3*Pi]. Bestimmte Integrale werden in Maple mit dem Kommando **int** berechnet:

```
> Int(f(x),x=0..3*Pi) = int(f(x),x=0..3*Pi);
```

$$\int_0^{3\,\pi} 1+\sin(x)\,dx = 3\,\pi+2$$

Int ist hier die träge Form des int Kommandos. Wir lassen nun die obere Grenze des Integrationsintervalls unbestimmt und erhalten

```
> Int(f(t),t=0..x) = int(f(t),t=0..x);
```

$$\int_0^x 1+\sin(t)\,dt = x-\cos(x)+1$$

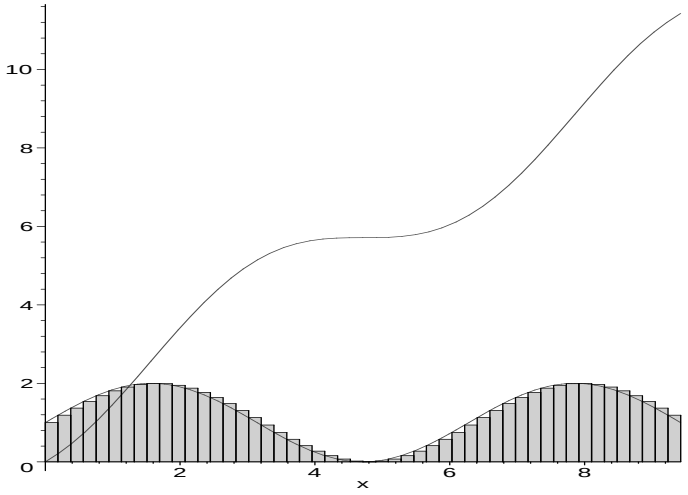
Die Funktion

```
> F:=unapply(rhs(%),x);
```

$$F := x \to x-\cos(x)+1$$

heißt Stammfunktion von f. Wir plotten nun die Funktion F zusammen mit einer Rechtecksappproximation in ein Schaubild.

```
> p1:=leftbox(f(x),x=0..3*Pi,50):
> p2:=plot(F(x),x=0..3*Pi):
> display({p1,p2});
```



Diese Graphik zeigt, daß die Ableitung der Stammfunktion F (Änderungsrate der Flächenzunahme) durch die Funktion f beschrieben wird.

```
> D(F);
```

$$x \to 1+\sin(x)$$

Stammfunktionen sind nicht eindeutig, da additive Konstanten die Ableitung 0 haben.

```
> const:=x -> c;
```

$$const := x \to c$$

```
> D(F+const);
```

$$x \to 1+\sin(x)$$

Die unbestimmte Integration mit der Maple-Prozedur int berücksichtigt keine Integrationskonstanten:

```
> Int(f(x),x) = int(f(x),x);
```

$$\int 1+\sin(x)\,dx = x-\cos(x)$$

Für die Berechnung eines bestimmten Integrals mittels Stammfunktion spielt die Integrationskonstante keine Rolle.

```
> F:=unapply(rhs(%),x);
```

$$F := x \to x-\cos(x)$$

```
> Int(f(x),x=0..3*Pi) = F(3*Pi) - F(0);
```

$$\int_0^{3\,\pi} 1+\sin(x)\,dx = 3\,\pi+2$$

Allgemein gilt für stetige Funktionen f (Fundamentalsatz)

```
> f:='f':
> F:=x -> int(f(t),t=a..x);
```

$$F := x \to \int_a^x \mathrm{f}(t)\,dt$$

die Beziehung

```
> Diff(F(x),x) = diff(F(x),x);
```

$$\frac{\partial}{\partial x}\int_a^x \mathrm{f}(t)\,dt = \mathrm{f}(x)$$

oder in funktionaler Schreibweise

```
> 'D(F)' = D(F);
```

$$\mathrm{D}(F) = f$$

Die Ableitung der Stammfunktion von f ist also wieder f, d.h. mit der Definition des Differentialquotienten gilt

```
> Limit((F(x+h) - F(x))/h,h=0) = limit((F(x+h) - F(x))/h, h=0);
```

$$\lim_{h\to 0}\frac{\int_a^{x+h}\mathrm{f}(t)\,dt - \int_a^x \mathrm{f}(t)\,dt}{h} = \mathrm{f}(x)$$

Für die Berechnung bestimmter Integrale haben wir

```
> Int(f(t),t=c..d) = 'F'(d) - 'F'(c);
```

$$\int_c^d \mathrm{f}(t)\,dt = \mathrm{F}(d) - \mathrm{F}(c)$$

4.2 Elementare Integrationstechniken

```
> restart:
```

In diesem Abschnitt behandeln wir einige elementare Integrationstechniken. Die Technik der Variablensubstitution läßt sich auf die Ableitungsregel für zusammengesetzte Funktionen zurückführen (Kettenregel). Es gilt allgemein

```
> with(student):
> h:=x -> f(g(x));
```

$$h := x \to \mathrm{f}(\mathrm{g}(x))$$

```
> h1:=D(h);
```

$$h1 := x \to \mathrm{D}(f)(\mathrm{g}(x))\,\mathrm{D}(g)(x)$$

```
> i1:=Int(h1(x),x);
```

$$i1 := \int \mathrm{D}(f)(\mathrm{g}(x))\,\mathrm{D}(g)(x)\,dx$$

```
> i2:=Int(h1(x),x=a..b);
```

$$i2 := \int_a^b \mathrm{D}(f)(\mathrm{g}(x))\,\mathrm{D}(g)(x)\,dx$$

Setzen wir nun g(x)=u, dann ist D(g)(x)dx = du und wir erhalten die transformierten Beziehungen mit **changevar**

```
> changevar(g(x)=u,i1,u): % = value(%);
```

$$\int \mathrm{D}(f)(u)\,du = \mathrm{f}(u)$$

bzw.

```
> changevar(g(x)=u,i2,u): % = value(%);
```

$$\int_{\mathrm{g}(a)}^{\mathrm{g}(b)} \mathrm{D}(f)(u)\,du = \mathrm{f}(\mathrm{g}(b)) - \mathrm{f}(\mathrm{g}(a))$$

Beispiele:

```
> Int(2*x*sqrt(1+x^2),x);
```

$$\int 2\,x\,\sqrt{1+x^2}\,dx$$

```
> changevar(1+x^2=u,%,u);
```

$$\int \sqrt{u}\,du$$

```
> value(%);
```

$$\frac{2}{3}\,u^{(3/2)}$$

Als Stammfunktion ergibt sich damit

```
> subs(u=1+x^2,%);
```

$$\frac{2}{3}\,(1+x^2)^{(3/2)}$$

```
> Int(2*(1+x^2)*x,x=2..3);
```

$$\int_2^3 2\,(1+x^2)\,x\,dx$$

```
> changevar(u=x^2+1,%,u);
```

$$\int_5^{10} u\,du$$

```
> value(%);
```

$$\frac{75}{2}$$

Die partielle Integration (**intparts**) basiert auf der Produktregel. Für integrierbare Funktionen f und g gilt

```
> h:='h':
> eq:=h = f*g;
```

$$eq := h = f\,g$$

```
> D(eq)(x);
```

$$\mathrm{D}(h)(x) = \mathrm{D}(f)(x)\,\mathrm{g}(x) + \mathrm{f}(x)\,\mathrm{D}(g)(x)$$

```
> expand(map(Int,%,x));
```

$$\int \mathrm{D}(h)(x)\,dx = \int \mathrm{D}(f)(x)\,\mathrm{g}(x)\,dx + \int \mathrm{f}(x)\,\mathrm{D}(g)(x)\,dx$$

```
> isolate(%,Int(f(x)*D(g)(x),x));
```

$$\int \mathrm{f}(x)\,\mathrm{D}(g)(x)\,dx = \int \mathrm{D}(h)(x)\,dx - \int \mathrm{D}(f)(x)\,\mathrm{g}(x)\,dx$$

```
> value(%);
```

$$\int \mathrm{f}(x)\,\mathrm{D}(g)(x)\,dx = \mathrm{h}(x) - \int \mathrm{D}(f)(x)\,\mathrm{g}(x)\,dx$$

```
> subs(eq,%);
```

$$\int \mathrm{f}(x)\,\mathrm{D}(g)(x)\,dx = (f\,g)(x) - \int \mathrm{D}(f)(x)\,\mathrm{g}(x)\,dx$$

Beispiel:

```
> Int(x^2*exp(x),x);
```

$$\int x^2\,e^x\,dx$$

```
> intparts(%,x^2);
```

$$x^2\,e^x - \int 2\,x\,e^x\,dx$$

```
> intparts(%,x);
```

$$x^2\,e^x - 2\,x\,e^x + \int 2\,e^x\,dx$$

```
> value(%);
```

$$x^2\,e^x - 2\,x\,e^x + 2\,e^x$$

```
> factor(%);
```

$$e^x\,(x^2 - 2\,x + 2)$$

Durch Partialbruchzerlegung (**convert,parfrac**) kann man die Integration rationaler Funktionen oft auf die Integration einfacherer Funktionen zurückführen. Es folgt ein einfaches Beispiel:

```
> f:=x -> (x^3-3*x+2*x-5)/((x^2-x-1)*(x-3)*(x-4));
```

$$f := x \to \frac{x^3 - x - 5}{(x^2 - x - 1)\,(x - 3)\,(x - 4)}$$

```
> convert(f(x),parfrac,x);
```

$$-\frac{19}{5}\,\frac{1}{x-3} + \frac{5}{x-4} - \frac{1}{5}\,\frac{2+x}{x^2-x-1}$$

```
> Int(%,x);
```

$$\int -\frac{19}{5}\,\frac{1}{x-3} + \frac{5}{x-4} - \frac{1}{5}\,\frac{2+x}{x^2-x-1}\,dx$$

```
> expand(%);
```

$$-\frac{19}{5}\int \frac{1}{x-3}\,dx + 5\int \frac{1}{x-4}\,dx - \frac{2}{5}\int \frac{1}{x^2-x-1}\,dx - \frac{1}{5}\int \frac{x}{x^2-x-1}\,dx$$

```
> value(%);
```

$$-\frac{19}{5}\ln(x-3) + 5\ln(x-4) + \frac{1}{5}\sqrt{5}\,\mathrm{arctanh}(\frac{1}{5}\,(2\,x-1)\,\sqrt{5}) - \frac{1}{10}\ln(x^2-x-1)$$

Die Probe ergibt

```
> simplify(diff(%,x) - f(x));
```

$$0$$

4.3 Algorithmen für die unbestimmte Integration

```
> restart:
```

Die unbestimmte Integration ist eines der “highlights” der Computeralgebra. Neben den klassischen heuristischen Methoden zur Bestimmung von Stammfunktionen, die in vielen Lehrbüchern der Analysis beschrieben sind, benutzt Maple nicht-klassische Integrationsalgorithmen, wie z.B. den Risch-Algorithmus. Im folgenden wird der Algorithmus der Maple-Prozedur int grob beschrieben:

- Zunächst versucht Maple heuristische Methoden zu nutzen: Lookup-tables, Variablensubstitution, Partialbruchzerlegung, partielle Integration, usw..

- Wenn heuristische Methoden nicht zum Erfolg führen, wird der Risch-Algorithmus angewendet, d.h. für elementare (transzendente) Funktionen wird überprüft, ob das unbestimmte Integral als Formel mit endlich vielen Termen, bestehend aus elementaren Funktionen, dargestellt werden kann, und wenn ja, dann wird diese Formel berechnet. Die Klasse der elementaren transzendenten Funktionen umfaßt Polynome, rationale und trigonometrische Funktionen, Exponential- und Logarithmusfunktion sowie alle Funktionen, die durch Verknüpfung und/oder Schachtelung dieser Funktionen entstehen.

- Im letzten Schritt wird die Integration algebraischer Funktionen in RootOf-Notation behandelt (Risch-Trager-Algorithmus). Algebraische Funktionen sind Lösungen von Polynomgleichungen, deren Koeffizienten elementare transzendente Funktionen sein können.

Wenn man wissen möchte, wie Maple zu seinen Ergebnissen kommt, kann man beim Integrieren zusehen. Dazu muß man der Systemvariable **infolevel** für int einen Wert zwischen 1 und 5 zuweisen (je höher der Wert, desto ausführlicher werden die Informationen). Beispiele:

```
> infolevel[int]:=1;
```

$$infolevel_{int} := 1$$

```
> Int(ln(x-1)^2,x): % = value(%);
```

```
int/indef1:   first-stage indefinite integration

int/indef2:   second-stage indefinite integration

int/indef2:   applying change of variables

int/indef1:   first-stage indefinite integration

int/indef2:   second-stage indefinite integration

int/ln:   case of integrand containing ln
```

$$\int \ln(x-1)^2\,dx = \ln(x-1)^2\,(x-1) - 2\,(x-1)\ln(x-1) + 2\,x - 2$$

```
> Int(ln(x-1)^2/x^2, x): % = value(%);

int/indef1:   first-stage indefinite integration

int/indef2:   second-stage indefinite integration

int/indef2:   applying change of variables

int/indef1:   first-stage indefinite integration

int/indef2:   second-stage indefinite integration

int/ln:   case of integrand containing ln

int/rischnorm:   enter Risch-Norman integrator

int/rischnorm:   exit Risch-Norman integrator

int/risch:   enter Risch integration

int/risch/algebraic1:   RootOfs should be algebraic numbers and
functions

int/risch:   exit Risch integration
```

$$\int \frac{\ln(x-1)^2}{x^2}\,dx = \int \frac{\ln(x-1)^2}{x^2}\,dx$$

Maple gibt die Eingabe zurück, d.h. das unbestimmte Integral kann nicht als geschlossene Formel mit endlich vielen elementaren Funktionen ausgedrückt werden. In einigen Fällen wird die Klasse der elementaren Funktionen um neudefinierte (spezielle) Funktionen erweitert. Beispiele (Fehlerfunktion und Exponentialintegral):

```
> infolevel[int]:=0;
```

$$infolevel_{int} := 0$$

```
> Int(exp(-x^2),x) = int(exp(-x^2),x);
```

$$\int e^{(-x^2)}\,dx = \frac{1}{2}\,\sqrt{\pi}\,\mathrm{erf}(x)$$

```
> Int(exp(-x)/x,x) =int(exp(-x)/x,x);
```

$$\int \frac{e^{(-x)}}{x}\,dx = -\mathrm{Ei}(1,\,x)$$

Integration einer einfachen algebraischen Funktion:

```
> f:=x -> 1/(1+x^3)^(1/3);
```

$$f := x \to \frac{1}{(1+x^3)^{(1/3)}}$$

```
> infolevel[int]:=1;
```

$$infolevel_{int} := 1$$

```
> ia:=Int(f(x),x);
```

$$ia := \int \frac{1}{(1+x^3)^{(1/3)}}\,dx$$

```
> value(ia);

int/indef1:   first-stage indefinite integration

int/algebraic2/algebraic:   algebraic integration

int/rischnorm:   enter Risch-Norman integrator

int/rischnorm:   exit Risch-Norman integrator

int/risch:   enter Risch integration

int/indef1:   first-stage indefinite integration

int/indef2:   second-stage indefinite integration

int/indef2:   trying integration by parts

int/risch:   exit Risch integration
```

$$x\,\mathrm{hypergeom}([\frac{1}{3},\,\frac{1}{3}],\,[\frac{4}{3}],\,-x^3)$$

Das unbestimmte Integral wird hier als hypergeometrische Funktion 2F1 zurückgegeben, d.h. als unendliche Reihe. Maple kann dieses Integral jedoch auch als Ausdruck mit endlich vielen elementaren Funktionen darstellen. Die Integration erfolgt dann über den Risch-Trager-Algorithmus. Dieses Verfahren wird aber nur angewendet, wenn der Integrand in der RootOf-Notation erscheint. Die Funktion **RootOf** dient als Platzhalter für die Nullstellen von Gleichungen mit einer Variablen. In Maple ist die RootOf-Notation für algebraische Zahlen und Funktionen voreingestellt. Die Konvertierung in die RootOf-Darstellung erfolgt mit **convert,RootOf**

```
> ia1:=convert(ia,RootOf);
```

$$ia1 := \int \frac{1}{\mathrm{RootOf}(\_Z^3 - 1 - x^3,\ index = 1)}\,dx$$

Die folgende Integration erfordet viel Rechenzeit und sollte daher nur auf Computern mit schneller CPU durchgeführt werden.

```
> settime:=time();
```

$$settime := 4.006$$

```
> ia1:=value(ia1);

int/indef1:   first-stage indefinite integration

int/indef2:   second-stage indefinite integration

int/rischnorm:   enter Risch-Norman integrator

int/rischnorm:   exit Risch-Norman integrator

int/algrisch/int:   Risch/Trager's algorithm for algebraic function

int/algrisch/int:   computation of the algebraic part: start time
5.217

int/algrisch/int:   computation of the algebraic part: end time
5.262

int/algrisch/int:   computation of the transcendental part: start time
  5.271

int/algrisch/int:   computation of the transcendental part: end time
31.747
```

$$ia1 := \frac{1}{3}\,\%1\ln(1+2\,x^3+\%1\,x^3+\%1-x^3\,\%1^2+3\,\%2\,x^2+3\,\%2^2\,x)$$
$$+\frac{1}{3}\ln(2+2\,x^3+\%1\,x^3-\%1-x^3\,\%1^2+3\,\%2\,x^2+3\,\%2^2\,x)$$
$$-\frac{1}{3}\ln(2+2\,x^3+\%1\,x^3-\%1-x^3\,\%1^2+3\,\%2\,x^2+3\,\%2^2\,x)\,\%1$$
$$\%1 := \mathrm{RootOf}(\_Z^2-\_Z+1)$$
$$\%2 := \mathrm{RootOf}(\_Z^3-1-x^3,\ index=1)$$

```
> time_consumed:=time() - settime;
```

$$time\_consumed := 27.766$$

convert,radical liefert schließlich die Wurzeldarstellung:

```
> convert(ia1,radical);
```

$$\frac{1}{3}(\frac{1}{2}-\frac{1}{2}\,I\sqrt{3})\ln(\frac{3}{2}+2\,x^3+(\frac{1}{2}-\frac{1}{2}\,I\sqrt{3})\,x^3-\frac{1}{2}\,I\sqrt{3}-x^3\,(\frac{1}{2}-\frac{1}{2}\,I\sqrt{3})^2+3\,(1+x^3)^{(1/3)}\,x^2$$
$$+3\,(1+x^3)^{(2/3)}\,x)+\frac{1}{3}\ln(\frac{3}{2}+2\,x^3+(\frac{1}{2}-\frac{1}{2}\,I\sqrt{3})\,x^3+\frac{1}{2}\,I\sqrt{3}-x^3\,(\frac{1}{2}-\frac{1}{2}\,I\sqrt{3})^2$$
$$+3\,(1+x^3)^{(1/3)}\,x^2+3\,(1+x^3)^{(2/3)}\,x)-\frac{1}{3}\ln(\frac{3}{2}+2\,x^3+(\frac{1}{2}-\frac{1}{2}\,I\sqrt{3})\,x^3+\frac{1}{2}\,I\sqrt{3}$$
$$-x^3\,(\frac{1}{2}-\frac{1}{2}\,I\sqrt{3})^2+3\,(1+x^3)^{(1/3)}\,x^2+3\,(1+x^3)^{(2/3)}\,x)(\frac{1}{2}-\frac{1}{2}\,I\sqrt{3})$$

Probe:

```
> simplify(diff(%,x) - f(x));
```

$$0$$

4.4 Bestimmte Integration

```
> restart:
```

Für die bestimmte Integration wird in der Prozedur int nicht einfach das entsprechende unbestimmte Integral berechnet und dann an den Integrationsgrenzen ausgewertet. Dies würde oft zu Fehlern führen, wie im folgenden Beispiel:

```
> Int(1/x^2,x) = int(1/x^2,x);
```

$$\int \frac{1}{x^2}\,dx = -\frac{1}{x}$$

```
> Int(1/x^2,x=-1..1) = subs(x=1,rhs(%)) - subs(x=-1,rhs(%));
```

$$\int_{-1}^{1} \frac{1}{x^2}\,dx = -2$$

Der Integrand hat im Intervall [-1, 1] eine nicht behebbare Singularität an der Stelle x=0. Maple überprüft die Stetigkeit des Integranden im vorgegebenen Integrationsintervall.

```
> Int(1/x^2,x=-1..1) = int(1/x^2,x=-1..1);
```

$$\int_{-1}^{1} \frac{1}{x^2}\,dx = \infty$$

Für die bestimmte Integration werden in der Prozedur int unterschiedliche Techniken genutzt, wie z.B. lookup-tables, pattern matching, Differentiation von speziellen Funktionen nach Parametern, u.a..

```
> infolevel[int]:=5;
```

$$infolevel_{int} := 5$$

```
> i1:=Int(exp(-u*x^2),x=0..infinity): % = value(%);

int/cook/nogo1:

Given Integral

Int(exp(-u*x^2),x = 0 .. infinity)

Fits into this pattern:

Int(exp(-Ucplex*x^S1-U2*x^S2)*x^N*ln(B*x^DL)^M*cos(C1*x^R)/((A0+A1*x^D
)^P),x = t1 .. t2)

Definite integration: Can't determine if the integral is convergent.

Need to know the sign of --> u

Will now try indefinite integration and then take limits.

int/elliptic:   trying elliptic integration

int/ellalg/elltype:   Checking for an elliptic integral   exp(-u*x^2)
 freeof(x)   x

int/indef1:   first-stage indefinite integration

int/indef2:   second-stage indefinite integration

int/exp:   case of integrand containing exp

int/indef1:   first-stage indefinite integration

int/indef2:   second-stage indefinite integration

int/exp:   case of integrand containing exp

int/prpexp:   case ratpoly*exp(arg)
```

$$\int_0^\infty e^{(-u\,x^2)}\,dx = \lim_{x\to\infty}\frac{1}{2}\,\frac{\sqrt{\pi}\,\mathrm{erf}(\sqrt{u}\,x)}{\sqrt{u}}$$

In diesem Beispiel hängt das bestimmte Integral von einem Parameter ab. Die Existenz des Integrals wird bestimmt durch das Vorzeichen von u. Nach der Definition der Parametereigenschaft mit **assume**

```
> assume(u>0);
```

wird das Integral ausgewertet.

```
> value(i1);
```

$$\frac{1}{2}\,\frac{\sqrt{\pi}}{\sqrt{u\tilde{}}}$$

```
> about(u);

Originally u, renamed u~:

  is assumed to be: RealRange(Open(0),infinity)
```

Variablen mit definierten Eigenschaften erhalten in Maple als Kennzeichen eine Tilde. Die Parametereigenschaft wird durch die Zuweisung

```
> u:='u':
```

wieder gelöscht.

```
> about(u);

u:
```

```
  nothing known about this object
```

```
> i2:=Int(exp(-u*x^2)*erf(x),x=0..infinity);
```

$$i2 := \int_0^{\infty} e^{(-u\,x^2)}\,\mathrm{erf}(x)\,dx$$

```
> value(i2);

int/cook/nogo1:

Given Integral

Int(exp(-u*x^2)*erf(x),x = 0 .. infinity)

Fits into this pattern:

Int(exp(-U*x^S1)*x^N*ln(B*x^DL)^M*erf(F*x^V+G),x = t1 .. t2)

int/elliptic:   trying elliptic integration

int/ellalg/elltype:   Checking for an elliptic integral
exp(-u*x^2)*erf(x)   freeof(x)   x
```

$$\frac{1}{2}\,\frac{\dfrac{\pi}{\sqrt{u}}-\dfrac{2\arctan(\sqrt{u})}{\sqrt{u}}}{\sqrt{\pi}}$$

4.5 Numerische Integration

```
> restart:
```

Für die numerische Berechnung bestimmter Integrale stehen im Paket student mehrere Verfahren zur Verfügung.

```
> with(student):
```

Die Rechteckregeln haben wir bereits bei der Einführung des Riemann-Integrals kennengelernt. Für die numerische Approximation eines Integrals mittels “linksseitiger” Rechtecke gilt

```
> Int(f(x),x=a..b) = limit(leftsum(f(x),x=a..b,n),n=infinity);
```

$$\int_a^b \mathrm{f}(x)\,dx = \lim_{n\to\infty} \frac{(b-a)\,\left(\displaystyle\sum_{i=0}^{n-1} \mathrm{f}(a+\frac{i\,(b-a)}{n})\right)}{n}$$

Werden die Rechtecke durch Trapeze ersetzt, so erhalten wir die Trapezregel (**trapezoid**).

```
> Int(f(x),x=a..b) = limit(trapezoid(f(x),x=a..b,n),n=infinity);
```

$$\int_a^b \mathrm{f}(x)\,dx = \lim_{n\to\infty} \frac{1}{2}\,\frac{(b-a)\,\left(\mathrm{f}(a)+2\,\left(\displaystyle\sum_{i=1}^{n-1} \mathrm{f}(a+\frac{i\,(b-a)}{n})\right)+\mathrm{f}(b)\right)}{n}$$

Theoretisch läßt sich zeigen, daß der Approximationsfehler kleiner als

```
> M*(b-a)^3/(12*n^2);
```

$$\frac{1}{12}\,\frac{M\,(b-a)^3}{n^2}$$

ist. Hierbei ist M der maximale Wert der zweiten Ableitung von f auf [a, b]. Wird die Kurve der Funktion f stückweise durch quadratische Polynome q ersetzt mit der Eigenschaft, daß die Funktionswerte von f und q jeweils an den Intervallgrenzen und in der Intervallmitte übereinstimmen, so folgt die Simpson-Regel (**simpson**).

```
> Int(f(x),x=a..b) = limit(simpson(f(x),x=a..b,n),n=infinity);
```

$$\int_a^b \mathrm{f}(x)\,dx = \lim_{n\to\infty} \frac{1}{3}$$

$$\frac{(b-a)\left(\mathrm{f}(a)+\mathrm{f}(b)+4\left(\sum_{i=1}^{1/2\,n}\mathrm{f}(a+\frac{(2\,i-1)\,(b-a)}{n})\right)+2\left(\sum_{i=1}^{1/2\,n-1}\mathrm{f}(a+\frac{2\,i\,(b-a)}{n})\right)\right)}{n}$$

Der Approximationsfehler der Simpson-Regel wird durch

```
> M*(b-a)^5/(180*n^4);
```

$$\frac{1}{180}\,\frac{M\,(b-a)^5}{n^4}$$

begrenzt. M ist hier der maximale Wert der vierten Ableitung von f auf [a,b]. Beispiel:

```
> f:=x -> sin(x);
```

$$f := \sin$$

```
> Int(f(x),x=0..Pi/2) = int(f(x),x=0..Pi/2);
```

$$\int_0^{1/2\,\pi} \sin(x)\,dx = 1$$

```
> simplify(value(leftsum(f(x),x=0..Pi/2,n)));
```

$$-\frac{1}{4}\,\frac{\pi\,(\cos(\frac{1}{2}\,\frac{\pi}{n})-1+\sin(\frac{1}{2}\,\frac{\pi}{n}))}{n\,(\cos(\frac{1}{2}\,\frac{\pi}{n})-1)}$$

```
> seq(evalf(%),n=1..8);
```

$$0.,\ .5553603666,\ .7152492302,\ .7907662606,\ .8346821365,\ .8633821951,\ .8836004770,\ .8986104019$$

```
> simplify(value(trapezoid(f(x),x=0..Pi/2,n)));
```

$$-\frac{1}{4}\,\frac{\pi\sin(\frac{1}{2}\,\frac{\pi}{n})}{n\,(\cos(\frac{1}{2}\,\frac{\pi}{n})-1)}$$

```
> seq(evalf(%),n=1..8);
```

$$.7853981635,\ .9480594481,\ .9770486184,\ .9871158012,\ .9917617690,\ .9942818888,\ .9958002145,\ .9967851722$$

```
> combine(expand(value(simpson(f(x),x=0..Pi/2, n))),trig);
```

$$\frac{-4\,\pi\sin(\frac{1}{2}\,\frac{\pi}{n})-\pi\sin(\frac{\pi}{n})}{6\,n\cos(\frac{\pi}{n})-6\,n}$$

```
> seq(evalf(%),n=1..8);
```

$$1.047197551,\ 1.002279878,\ 1.000431595,\ 1.000134584,\ 1.000054759,\ 1.000026313,\ 1.000014172,\ 1.000008296$$

Die Maple-Prozedur int bietet ebenfalls numerische Integrationsverfahren (**evalfint**).

```
> int(exp(arcsin(x)),x=0..1);
```

$$\int_0^1 e^{\arcsin(x)}\,dx$$

```
> evalf(%);
```

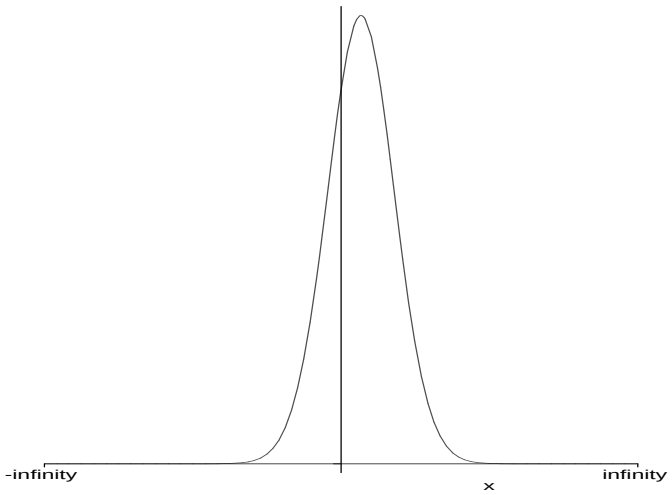
$$1.905238690$$

Die Clenshaw-Curtis-Quadratur wird als Standardverfahren angewandt. Zur Behandlung von Singularitäten werden symbolische Methoden wie Variablentransformation und verallgemeinerte Reihenentwicklung angewendet.

```
> g:=x -> exp(x-x^2/2)/(1+1/2*exp(x));
```

$$g := x \to \frac{e^{(x-1/2\,x^2)}}{1+\frac{1}{2}\,e^{x}}$$

```
> plot(g(x),x=-infinity..infinity);
```




```
> int(g(x),x=-infinity..infinity);
```

$$\int_{-\infty}^{\infty} \frac{e^{(x-1/2\,x^2)}}{1+\frac{1}{2}\,e^{x}}\,dx$$

Die Integration erfolgt in Maple numerisch, wenn evalf auf das unausgewertete Integral angewandt wird:

```
> evalf(Int(g(x),x=-infinity..infinity),15);
```

$$1.80557715381926$$

Kapitel 5

Programmieren in Maple

5.1 Verzweigungen, Schleifen

```
> restart:
```

Maple besitzt eine prozedurale Programmiersprache, deren Aufbau den Programmiersprachen wie Pascal, C oder Fortran sehr ähnlich ist. Maple kennt acht verschiedene Anweisungen (**statements**):

- assignment statement
- selection statement
- repetition statement
- read statement
- save statement
- empty statement
- quit statement
- expressions

Die Wertzuweisung (**assignment** statement) hatten wir bereits kennengelernt. Wir kommen jetzt zur Syntax von **if** Abfragen (selection statement). Verzweigungen, Fallunterscheidungen und Abfragen werden in Maple mit den Schlüsselwörtern if, then, elif, else und endif gebildet. Es gibt vier verschiedene Abfrageformen. Es folgt die Syntax der ersten zwei Formen:

if <expr> then <statseq> end if;

if <expr> then <statseq1> else <statseq2> end if;

Hierbei ist <expr> ein Boolscher Ausdruck oder eine Boolsche Funktion und <statseq> eine Folge von Befehlen (statement sequence). Beispiele:

```
> x:=-2:
> if x > 0 then
> x:=x - 1
> end if;
> x;
```

$$-2$$

```
> if x < 0 then
> 0
> else
> 1
> end if;
```

$$0$$

Geschachtelte Abfragen sind ebenfalls möglich.

```
> printlevel:=2;
```

$$printlevel := 2$$

Um den Wert einer geschachtelten if Abfrage anzuzeigen, erhöhen wir die globale Variable **printlevel** auf den Wert 2.

```
> if x>1 then
> 1
> else
> if x=0 then
> 0
> else
> -1
> end if
> end if;
```

$$-1$$

Für viele Fallunterscheidungen bietet Maple die beiden folgenden Abfrageformen:

if <expr1> then <statseq1> elif <expr2> then <statseq2> end if;

bzw.

if <expr1> then <statseq1> elif <expr2> then <statseq2>
else <statseq3> end if;

Das Konstrukt 'elif ... then ...' kann mehrmals erscheinen. Die Vorzeichenfunktion läßt sich in dieser Abfrageform wie folgt realisieren:

```
> if x < 0 then
> -1
> elif x = 0 then
> 0
> else
> 1
> end if;
```

$$-1$$

Diese Form der if Abfrage kann man auch als case statement ansehen. Im folgenden Beispiel soll der Parameter n nur die vier Werte 0, 1, 2, 3 annehmen:

```
> n:=5:
> if n = 0 then
> 0
> elif n = 1 then
> 1/2
> elif n = 2 then
> sqrt(2)/2
> elif n = 3 then
> sqrt(3)/2
> else
> ERROR(‘bad argument‘,n)
> end if;

Error, bad argument, 5
```

Wir kommen nun zum **repetition** (**for**/**while**/**do**) statement. Als allgemeinste Form kennt Maple die for Schleife. Es gibt zwei Formen: for-from und for-in. Die Syntax der Schleifenform for-from ist wie folgt:

for <name> from <start> by <step> to <finish> while <expr>

do <statseq> end do;

Hierbei ist <start>, <step> und <finish> vom Datentyp numeric, d.h. integer, fraction oder float, und <expr> ein Boolscher Ausdruck. Die Schlüsselwörter for, from, by, to und while können fehlen, <statseq> kann ebenfalls fehlen. Für <name>, <start>, <step>, <finish> und <expr> sind die folgenden Werte vordefiniert:

```
name      dummy variable
start     1
step      1
finish    infinity
expr      true
```

Beispiele:

```
> for i from 2 to 5
> do
> i^2
> end do;
```

$$4$$
$$9$$
$$16$$
$$25$$

```
> i;
```

$$6$$

Nach Beendigung der Schleife hat der Laufindex i den Wert 6 (i + 1). Voll ausgeschrieben hat dieses Beispiel somit die Kontrollstruktur

```
> for i from 2 by 1 to 5 while true
> do
> i^2
> end do;
```

$$4$$
$$9$$
$$16$$
$$25$$

Ist der Wert der Schrittweite <step> negativ, so wird im Schleifenindex rückwärts gezählt.

```
> for i from 5 by -1 to 2
> do
> i^2
> end do;
```

$$25$$
$$16$$
$$9$$
$$4$$

Die erste Primzahl größer als 10^7 erhält man z.B. mit der Schleife

```
> for i from 10^7 while not isprime(i)
> do
> end do;
```

```
> i;
```

$$10000019$$

Der Schleifenkörper ist in diesem Beispiel leer! Werden alle Schleifenbedingungen weggelassen, dann entsteht eine Endlosschleife.

do <statseq> end do;

Die **while** Schleife hat die folgende Syntax:

while <expr> do <statseq> end do;

```
> x :=256:
> while x > 1
> do
> x:=x/4
> end do;
```

$$x := 64$$
$$x := 16$$
$$x := 4$$
$$x := 1$$

Jetzt kommen wir zur Schleifenform for-in.

for <name> in <expr1> while <expr2> do <statseq> end do;

Die Scheife durchläuft die Liste der Operanden op(<expr1>) des Ausdrucks <expr1>; <expr2> ist ein Boolscher Ausdruck. Beispiel:

```
> s:=sin(y) + ln(y) + y^2;
```

$$s := \sin(y) + \ln(y) + y^2$$

```
> op(s);
```

$$\sin(y),\ \ln(y),\ y^2$$

```
> printlevel := 1:
> for term in s while type(term,function)
> do
> diff(term, y)
> end do;
```

$$\cos(y)$$
$$\frac{1}{y}$$

```
> type(y^2,function);
```

$$false$$

```
> term;
```

$$y^2$$

In Maple gibt es zwei weitere Kontrollstrukturen für Schleifen: **break** und **next**. Mit break kann man eine Schleife vorzeitig verlassen. Mit next kann man die Anweisungen bis zum Beginn der nächsten Iteration überspringen.

Beispiel:

```
> for i from 3 by 2 do
> if isprime(2^i-1) then
> print(2^i-1,‘ist Primzahl‘)
> else
> break
> end if
> end do;
```

$$7,\ ist\ Primzahl$$
$$31,\ ist\ Primzahl$$
$$127,\ ist\ Primzahl$$

Bemerkung: In vielen Fällen lassen sich for Schleifen durch effizientere Kommandos ersetzen, z.B. seq, map, zip.

5.2 Prozeduren

```
> restart:
```

In Maple werden Prozeduren mit der proc ... end proc Syntax geschrieben. Die allgemeine Form einer Prozedur (**procedure**) hat die folgende Gestalt:

proc(<parameter_sequence>)

local <variable_sequence>;

global <variable_sequence>;

options <option_sequence>;

description <string>;

<statement_sequence>

end proc;

Bis auf den Prozedurrahmen proc() end proc können alle anderen Teile fehlen. Es folgt ein einfaches Beispiel:

```
> proc(x,y) x^2+y^2 end proc;
```

$$\textbf{proc}(x,\ y)\ x^2+y^2\ \textbf{end proc}$$

Wie jedem anderen Objekt in Maple kann man der Prozedur auch einen Namen zuweisen.

```
> F:=proc(x,y) x^2+y^2 end proc;
```

$$F := \textbf{proc}(x,\ y)\ x^2+y^2\ \textbf{end proc}$$

Die Prozedur wird durch den Aufruf

```
> F(diff(x^2,x),3);
```

$$4\,x^2+9$$

ausgeführt. Alle aktuellen Parameter werden zunächst ausgewertet (call by value), anschließend wird jeder formale Parameter durch den entsprechenden aktuellen Parameter ersetzt (call by name). Das Resultat einer Prozedur ist normalerweise der Wert des zuletzt ausgewerteten Befehls. Die Anzahl der aktuellen Parameter muß nicht mit der Anzahl der formalen Parameter übereinstimmen. Überflüssige aktuelle Parameter werden bei der Prozedurausführung ignoriert. Werden beim Aufruf der Prozedur zuwenig aktuelle Parameter angegeben, dann wird nur dann ein Fehler gemeldet, wenn ein Parameter bei der Auswertung des Prozedurkörpers fehlt.

```
> f:=proc(x,y,z)
> if x>y then
> x
> else
> z
> end if
> end proc:
> f(1,2,3,4);
```

$$3$$

```
> f(1,2);

Error, (in f) f uses a 3rd argument, z, which is missing
> f(2,1);
```

$$2$$

Einfache Prozeduren, bestehend aus einem Ausdruck oder einer if Abfrage, lassen sich alternativ mit Hilfe der Pfeil-Notation als Funktion (**operators,functional**) definieren.

```
> G:=(x,y) -> x^2+y^2;
```

$$G := (x,\ y) \rightarrow x^2+y^2$$

Bei Funktionen mit einem Argument können die runden Klammern für die Parameterliste fehlen.

```
> H:=n -> if n<0 then 0 else 1 end if;
```

$$H := \mathbf{proc}(n)\ \mathbf{option}\ operator,\ arrow;\ \mathbf{if}\ n < 0\ \mathbf{then}\ 0\ \mathbf{else}\ 1\ \mathbf{end\ if}\ \mathbf{end\ proc}$$

Prozeduren sind gültige Maple-Ausdrücke. Sie können auch ohne Bezeichner als anonyme Prozeduren (Funktionen) verwendet werden.

```
> (x->x^2)(t);
```

$$t^2$$

```
> proc(x,y) x^2+y^2 end proc(u,v);
```

$$u^2 + v^2$$

```
> map(x->x^2,[1,2,3,4]);
```

$$[1,\ 4,\ 9,\ 16]$$

```
> D(x->x^2);
```

$$x \to 2\,x$$

```
> D(exp + 2*ln);
```

$$\exp + 2\,(a \to \frac{1}{a})$$

Die Deklarierung formaler Parameter ermöglicht die Überprüfung von Datentypen. Die Syntax im Prozedurkopf ist

<parameter>::<type>

Beispiele:

```
> MAX:=proc(x::numeric,y::numeric)
> if x>y then
> x
> else
> y
> end if;
> end proc:

> MAX(Pi,3);

Error, MAX expects its 1st argument, x, to be of type numeric, but
received Pi

> G:=proc(n::even)
> n!*(n/2)!
> end proc:

> G(6);
```

$$4320$$

```
> G(5);

Error, G expects its 1st argument, n, to be of type even, but received
5
```

Innerhalb einer Prozedur lassen sich mit **nargs** die Anzahl der aktuellen Parameter, mit **args** die Folge der aktuellen Parameter und mit args[i] der i-te aktuelle Parameter ermitteln. Eine allgemeine Prozedur MAX, die das Maximum einer endlichen Zahlenfolge errechnet, läßt sich damit wie folgt realisieren:

```
> MAX:=proc()
> local i,m;
> if nargs = 0 then
> return(FAIL)
> end if;
> m:=args[1];
> for i from 2 to nargs
> do
> if args[i] > m then
> m:=args[i]
> end if
> end do;
> m
> end proc:
> MAX(1/2,4/7,2/3);
```

$$\frac{2}{3}$$

Variablen innerhalb einer Prozedur sind entweder lokal oder global zu dieser Prozedur. i und m sind in der Prozedur MAX als lokale Variablen vereinbart. Werden Variablen in einer Prozedur nicht als lokal oder global vereinbart, so entscheidet Maple darüber, wie sie behandelt werden. Eine nicht-vereinbarte Variable wird als lokale Variable verwendet, wenn

- sie auf der linken Seite einer Zuweisung steht oder

- als Index in einer for Schleife oder einem seq Kommando steht.

Andernfalls wird die Variable als globale Variable behandelt.

```
> MAX:=proc()
> if nargs = 0 then
> return(FAIL)
> end if;
> m:=args[1];
> for i from 2 to nargs
> do
> if args[i] > m then
> m:=args[i]
> end if
> end do;
> m
> end proc:
```

```
Warning, ‘m‘ is implicitly declared local to procedure ‘MAX‘

Warning, ‘i‘ is implicitly declared local to procedure ‘MAX‘
```

In einer Prozedur werden die lokalen Variablen nur einmal ausgewertet, während globale Variablen beliebig oft ausgewertet werden. Beispiel:

```
> f:=a + b;
```

$$f := a + b$$

```
> a:=c^2/b;
```

$$a := \frac{c^2}{b}$$

```
> c:=b^3 +3;
```

$$c := b^3 + 3$$

Die volle rekursive Auswertung liefert nun

```
> f;
```

$$\frac{(b^3+3)^2}{b}+b$$

Die Auswertungsstufen kann man mit eval kontrollieren:

```
> eval(f,1);
```

$$a+b$$

```
> eval(f,2);
```

$$\frac{c^2}{b}+b$$

```
> eval(f,3);
```

$$\frac{(b^3+3)^2}{b}+b$$

Mit dem eval Kommando läßt sich die volle Auswertung von lokalen Variablen erzwingen.

```
> F:=proc()
> local x, y, z;
> x := y^2;
> y := z;
> z := 3;
> eval(x);
> end proc:
> F();
```

$$9$$

Mit **return**(<sequence>); kann man eine Prozedur an einer vorgewählten Stelle verlassen. Der Wert der Prozedur ist dann <sequence>.

```
> GCD:=proc(a::integer,b::integer)
> local g;
> if a=0 and b=0 then
> return(0,0,0)
> end if;
> g:=igcd(a,b);
> g,iquo(a,g),iquo(b,g);
> end proc:
> GCD(0,0);
```

$$0,\ 0,\ 0$$

```
> GCD(12,8);
```

$$4,\ 3,\ 2$$

Mit **error**(<sequence>); wird die Abarbeitung der Prozedur unterbrochen und es wird eine Fehlermeldung der From

Error, (in <procname>) <sequence>

ausgegeben.

```
> inv:=proc(x::numeric)
> if x = 0 then
> error("Die Eingabe",x,"liefert Division durch 0");
> else
> 1/x
> end if;
> end proc:
> inv(0);

Error, (in inv) Die Eingabe, 0, liefert Division durch 0
```

In bestimmten Situationen ist es sinnvoll, den unausgewerteten Prozeduraufruf zurückzugeben. Beispiel:

```
> MAX:=proc(x,y)
> if x > y then
> x
> else
> y
> end if;
> end proc:
> MAX(3.2,2);
```

$$3.2$$

```
> MAX(x,1/x);

Error, (in MAX) cannot evaluate boolean: 1/x-x < 0
```

Die Prozedur MAX kann natürlich keine symbolischen Parameter verarbeiten. Beim Plotten der Funktion MAX tritt die gleiche Fehlersituation auf.

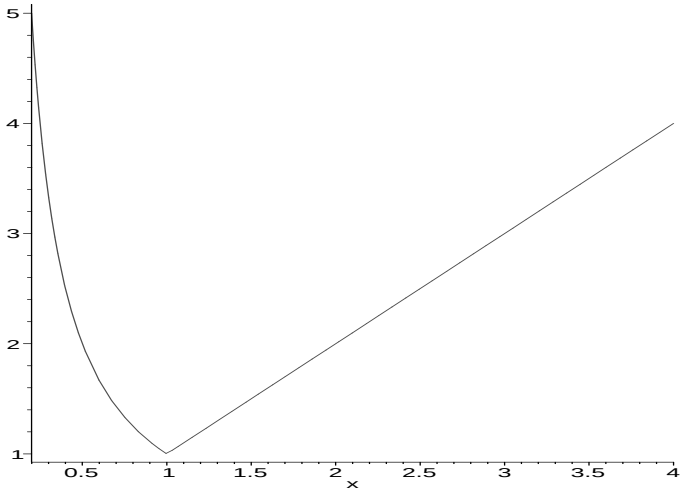
```
> plot(MAX(x,1/x),x=1/2..2);

Error, (in MAX) cannot evaluate boolean: 1/x-x < 0
```

Maple wertet hier zunächst den Ausdruck MAX(x, 1/x) symbolisch aus, bevor das plot Kommando aufgerufen wird. Eine Lösung des Problems ist hier, die Prozedur zu erweitern und bei symbolischen Parametern den Aufruf unausgewertet zurückzugeben.

```
> MAX:=proc(x, y)
> if type(x,numeric) and type(y,numeric) then
> if x > y then
> x
> else
> y
> end if
> else return('procname'(args))
> end if;
> end proc:
> MAX(x,1/2);
```

$$\mathrm{MAX}(x, \frac{1}{2})$$

```
> plot(MAX(x,1/x),x=0.2..4);
```



Manchmal möchte man, daß Werte über Prozedurparameter zurückgegeben werden. Die folgende Prozedur MEMBER bestimmt, ob eine gegebene Liste L einen Ausdruck x enthält. Darüberhinaus soll die Position von x in L über den dritten Parameter p zurückgegeben werden.

```
> MEMBER:=proc(x::anything,L::list,p::name)
> local i;
> for i to nops(L)
> do
> if x = L[i] then
> if nargs > 2 then
> p:=i
> end if;
> return(true)
> end if;
> end do;
> false
> end proc:
```

```
> MEMBER(b,[a,b,c,d],'q');
```

$$true$$

```
> q;
```

$$2$$

5.3 Optionen und interne Prozedurverwaltung

```
> restart:
```

Eine Prozedur kann eine oder mehrere Optionen (**options**) haben. Mögliche Optionen sind: **remember**, builtin, system, arrow, angle, trace und Copyright. Die Option remember ist für Anwendungen von besonderer Bedeutung. Wird eine Prozedur mit der Option remember aufgerufen, so erfolgt ein Eintrag in die remember **table** dieser Prozedur. Dabei werden die aktuellen Eingabeparameter (**indices**) eindeutig mit dem Resultat der Prozedurauswertung (**entries**) verknüpft . Maple überprüft bei jedem Prozeduraufruf, ob ein Eintrag in der remember table mit den aktuellen Parametern übereinstimmt. Ist dies der Fall, dann wird die Prozedur nicht durchlaufen, sondern der Eintrag aus der remember table ausgegeben. Im folgenden Beispiel werden die Tschebyscheff-Polynome mit Hilfe der Rekursionsbeziehung T(n, x) = 2xT(n-1, x) - T(n-2, x), n $>$ 1, berechnet.

```
> T:=proc(n,x)
> option remember;
> 2*x*T(n-1,x) - T(n-2,x)
> end proc;
```

$T := \mathbf{proc}(n,\, x)\ \mathbf{option}\ remember;\ 2 * x * \mathrm{T}(n-1,\, x) - \mathrm{T}(n-2,\, x)\ \mathbf{end\ proc}$

Wir definieren die beiden Startwerte der Rekursion explizit. Die Zuweisungen werden dann automatisch in die remember table eingetragen. Diese Methode haben wird schon bei stetig ergänzbaren Funktionen angewandt; sie funktioniert auch ohne remember option.

```
> T(0,x):=1:
```

```
> T(1,x):=x:
```

Die remember table ist der vierte Operand einer Maple-Prozedur.

```
> op(4,eval(T));
```

$$\mathrm{table}([(1,\, x) = x,\, (0,\, x) = 1])$$

Bei jedem Prozeduraufruf mit neuen Eingabeparametern werden weitere Einträge in die remember table aufgenommen.

```
> T(4,x);
```

$$2\,x\,(2\,x\,(2\,x^2 - 1) - x) - 2\,x^2 + 1$$

```
> op(4,eval(T));
```

$$\text{table}([(4,\, x) = 2\,x\,(2\,x\,(2\,x^2-1)-x)-2\,x^2+1,\ (1,\, x) = x,\ (3,\, x) = 2\,x\,(2\,x^2-1)-x,$$
$$(2,\, x) = 2\,x^2-1,$$
$$(0,\, x) = 1$$
$$])$$

Mit der Funktion **forget** kann man einzelne Einträge oder auch die gesamte remember table löschen.

```
> forget(T,4,x);
```

```
> op(4,eval(T));
```

$$\text{table}([(1,\, x) = x,\ (3,\, x) = 2\,x\,(2\,x^2-1)-x,\ (2,\, x) = 2\,x^2-1,\ (0,\, x) = 1])$$

```
> forget(T);
```

```
> op(4,eval(T));
```

Prozeduren werden in Pfeil-Notation ausgegeben, wenn die Optionen operator und arrow eingeschaltet werden.

```
> f:=proc(x)
> option operator,arrow;
> x^2
> end proc;
```

$$f := x \to x^2$$

Jede Option, die mit dem Wort Copyright anfängt, wird als Copyright Option interpretiert. Bei Copyright geschützten Prozeduren wird der Prozedurkörper nur dann angezeigt, wenn die interface Variable **verboseproc** den Wert 2 hat.

```
> d:=proc(expr::anything,x::name)
> option 'Copyright (c) 1684 by G.W. Leibniz';
> Diff(expr,x);
> end proc;
```

$$d := \textbf{proc}(expr{::}anything,\ x{::}name)$$
$$\textbf{option}\ \text{`}Copyright\ (c)\ 1684\ by\ G.W.\ Leibniz\text{`};$$
$$\quad \text{Diff}(expr,\ x)$$
$$\textbf{end proc}$$

```
> d(sin(x),x);
```

$$\frac{\partial}{\partial x}\sin(x)$$

```
> value(%);
```

$$\cos(x)$$

Wir setzen jetzt die Variable verboseproc auf 2. Der Quelltext einer Prozedur wird dann mit print ausgegeben.

```
> interface(verboseproc=2);
```

```
> print(d);
```

$$\textbf{proc}(expr{::}anything,\ x{::}name)$$
$$\textbf{option}\ \text{`}Copyright\ (c)\ 1684\ by\ G.W.\ Leibniz\text{`};$$
$$\quad \text{Diff}(expr,\ x)$$
$$\textbf{end proc}$$

Bei Prozeduren aus der Maple-Library läßt sich der Quelltext in analoger Weise anzeigen.

```
> print(ln);
```

```
proc(x::algebraic)
option `Copyright (c) 1992 by the University of Waterloo. All rights reserved.`;
  if nargs ≠ 1 then error "expecting 1 argument, got %1", nargs
  elif type(x, 'complex(float)') then evalf('ln'(x))
  elif x = 0 thenNumericEvent('real_to_complex',
    NumericEvent('division_by_zero', -∞ + undefined * I))
  elif type(x, '∞') then
    if type(x, 'neg_infinity') then NumericEvent('real_to_complex', ∞ + I * π)
    else ∞ + `if`(type(x, 'pos_infinity'), 0, I * argument(x))
    end if
  elif type(x, 'undefined') then
    if type(x, 'extended_numeric') then NumericEvent('real_to_complex', (1 + I) * x)
    else (1 + I) * x
    end if
  elif type(x, 'negative') then NumericEvent('real_to_complex', ln(-x) + I * π)
  elif type(x, 'function') and op(0, x) = 'exp' and ℑ(op(1, x)) = 0 then op(1, x)
  elif type(x, 'rational') and numer(x) = 1 and 0 < x then - ln(denom(x))
  elif type(x, 'SymbolicInfinity') and type(signum(0, x, undefined), 'numeric') then
    `if`(0 < signum(0, x, undefined), ∞, NumericEvent('real_to_complex', ∞ + I * π))
  elif type(x, 'constant^numeric') and signum(op(1, x)) = 1 then
    op(2, x) * ln(op(1, x))
  elifype(x, '`^`') and
  (type(op(2, x), 'integer') or traperror(is(op(2, x), 'integer'))) = true and
  (signum(0, ℜ(op(1, x)), 1) = 1 or member(signum(0, ℑ(op(1, x)), 0), {-1, 1}))
  then op(2, x) * ln(op(1, x))
  else ln(x) := 'ln'(x)
  end if
end proc
```

Eine Maple-Prozedur kann bis zu 6 Operanden haben:

- formale Parameter

- lokale Variablen

- Optionen

- remember table

- description

- globale Variablen

Beispiel:

```
> g:=proc(x::list,n::posint)
> local i;
> global y;
> option remember,Copyright;
> description `Summenbildung`;
> sum(x[i]+y[i],i=1..n);
> end proc:
> y:=[1,2,3];
```

$$y := [1, 2, 3]$$

```
> g([1,2,3],3);
```

$$12$$

```
> for k to 6
> do
> op(k,eval(g));
> end do;
```

$$x{::}list,\ n{::}posint$$

$$i$$

$$remember,\ Copyright$$

$$\mathrm{table}([([1,\ 2,\ 3],\ 3) = 12])$$

$$Summenbildung$$

$$y$$

Maple erlaubt die Definition geschachtelter Prozeduren, d.h. Prozeduren können innerhalb anderer Prozeduren definiert werden oder von Prozeduren übergeben werden. Lexical scoping ermöglicht es, daß in geschachtelten Prozeduren die Variablen der umgebenen Prozedur genutzt werden können. Mit lexical scoping wird eine bessere Einkapselung von Maple-Programmen erreicht. Mit der folgenden einfachen Maple-Funktion werden spezielle Matrizen generiert. Die Matrixstruktur wird über eine innere anonyme Prozedur definiert, die Dimension der Matrix und ein Parameter der Matrix werden über zwei Eingabeparameter einer äußeren Prozedur gesteuert. Die innere Prozedur soll dabei auf die Eingabeparameter zurückgreifen.

```
> A:=(n,a) -> matrix(n,n,(i,k) -> n+1+a - max(i,k));
```

$$A := (n,\ a) \to \mathrm{matrix}(n,\ n,\ (i,\ k) \to n + 1 + a - \max(i,\ k))$$

```
> A(4,5);
```

$$\begin{bmatrix} 9 & 8 & 7 & 6 \\ 8 & 8 & 7 & 6 \\ 7 & 7 & 7 & 6 \\ 6 & 6 & 6 & 6 \end{bmatrix}$$

Kapitel 6

Differentialgleichungen

6.1 Symbolische Lösung

```
> restart:
```

Die symbolische Lösung von Differentialgleichungen ist ein weiterer Schwerpunkt der Computeralgebra. Wir beschränken uns hier auf die Beschreibung der Maple-Prozedur **dsolve**, die Algorithmen für die analytische Lösung von gewöhnlichen Differentialgleichungen bereitstellt. Eine (im allgemeinen nichtlineare) Gleichung der Form

F(y(t),y'(t),y"(t), ..., y^(n)(t),t) = 0

heißt gewöhnliche Differentialgleichung n-ter Ordnung. In dsolve sind Lösungsalgorithmen für zahlreiche lineare und nichtlineare Differentialgleichungstypen implementiert. Außerdem können lineare Differentialgleichungssysteme gelöst werden. Für die Behandlung nicht lösbarer Differentialgleichungen werden einige numerische Lösungsverfahren angeboten. Die Syntax von dsolve ist

dsolve(<deqns>,<vars>, <opt>)

Hierbei bedeuten <deqns> eine Menge mit Differentialgleichungen und zugehörigen Anfangswerten, <vars> die Menge der gesuchten Lösungsfunktionen und <opt> optionale Argumente. Es folgt eine einfache Differentialgleichung mit Anfangsbedingung:

```
> de1:=diff(v(t),t)+2*t=0;
```

$$de1 := (\frac{\partial}{\partial t}\,\mathrm{v}(t)) + 2\,t = 0$$

```
> ini1:=v(1)=5;
```

$$ini1 := \mathrm{v}(1) = 5$$

```
> dsolve({de1,ini1},v(t));
```

$$\mathrm{v}(t) = -t^2 + 6$$

Werden die Anfangsbedingungen weggelassen, so gibt dsolve Lösungen mit numerierten Integrationskonstanten _C<n> zurück.

```
> de2:=diff(y(x),x$2) - y(x) = 1;
```

$$de2 := (\frac{\partial^2}{\partial x^2}\,\mathrm{y}(x)) - \mathrm{y}(x) = 1$$

```
> dsolve({de2},y(x));
```

$$\mathrm{y}(x) = -1 + \_C1\ e^{x} + \_C2\ e^{(-x)}$$

Die Anfangs- oder Randbedingungen für Ableitungen von Funktionen müssen in der folgenden Notation angegeben werden:

D(<function>)(<var_value >)=<value>

und für höhereAbleitungen

(D@@n)(<function>)(<var_value >)=<value>

Um Informationen zur Lösungssuche zu bekommen, erhöhen wir zunächst den Wert der Variablen infolevel.

```
> infolevel[dsolve]:=5;
```

$$infolevel_{dsolve} := 5$$

Beispiel:

```
> de3:=diff(y(x),x$2)-b*diff(y(x),x)-b*k*tau*y(x)=-b*k*tau;
```

$$de3 := (\tfrac{\partial^2}{\partial x^2}\,\mathrm{y}(x)) - b\,(\tfrac{\partial}{\partial x}\,\mathrm{y}(x)) - b\,k\,\tau\,\mathrm{y}(x) = -b\,k\,\tau$$

Mit dem Kommando **odeadvisor** kann der Differentialgleichungstyp ermittelt werden; wird die Option help mit angegeben, so werden weitere Informationen zum Lösungsverfahren in einer Hilfeseite angezeigt.

```
> DEtools[odeadvisor](de3,help);
```

$$[[\_2nd\_order, \_missing\_x]]$$

```
> ini3:=D(y)(0)-b*y0=0,D(y)(1)=0;
```

$$ini3 := \mathrm{D}(y)(0) - b\,y0 = 0,\ \mathrm{D}(y)(1) = 0$$

```
> dsolve({de3,ini3},y(x));
```

```
Methods for second order ODEs:

Trying to isolate the derivative d^2y/dx^2...

Successful isolation of d^2y/dx^2

-> Trying classification methods

trying a quadrature

trying high order exact linear fully integrable

trying differential order: 2; linear nonhomogeneous with symmetry
[0,1]

trying 2nd order linear exact nonhomogeneous

trying a double symmetry of the form [xi=0, eta=F(x)]

trying linear constant coefficient

linear constant coefficient successful
```

$$\mathrm{y}(x) = 1 + \frac{2\,b\,y0\,(e^{(1/2\,b-1/2\,\%1)})^2\,e^{(1/2\,(b+\sqrt{b\,(b+4\,k\,\tau)})\,x)}}{((e^{(1/2\,b-1/2\,\%1)})^2 - e^b)\,(b+\%1)}$$

$$- \frac{2\,b\,y0\,e^b\,e^{(1/2\,(b-\sqrt{b\,(b+4\,k\,\tau)})\,x)}}{(b-\%1)\,((e^{(1/2\,b-1/2\,\%1)})^2 - e^b)}$$

$$\%1 := \sqrt{b^2 + 4\,b\,k\,\tau}$$

Manchmal wird die Lösung einer Differentialgleichung in impliziter Form zurückgegeben.

```
> infolevel[dsolve]:=0;
```

$$infolevel_{dsolve} := 0$$

```
> de4:=diff(y(x),x) = exp(y(x))*y(x);
```

$$de4 := \tfrac{\partial}{\partial x}\,\mathrm{y}(x) = e^{\mathrm{y}(x)}\,\mathrm{y}(x)$$

```
> dsolve(de4,y(x));
```

$$x + \mathrm{Ei}(1,\,\mathrm{y}(x)) + \_C1 = 0$$

Die explizite Lösung kann man (wenn möglich) über die Option explicit anfordern:

```
> dsolve(de4,y(x),explicit);
```

$$\mathrm{y}(x) = \mathrm{RootOf}(x + \mathrm{Ei}(1,\,\_Z) + \_C1)$$

Mit der Option implicit (default) wird die Lösung in impliziter Form angegeben:

```
> dsolve(de4,y(x),implicit);
```

$$x + \mathrm{Ei}(1,\, \mathrm{y}(x)) + \_C1 = 0$$

6.2 Laplace-Transformation

```
> restart:
```

Die **Laplace**-Transformation von Differentialgleichungen ermöglicht oft die Reduktion der Komplexität des Problems. Die vorgegebenen Differentialgleichungen werden zunächst mittels Laplace-Transformation in algebraische Gleichungen überführt, die dann mit algebraischen Methoden gelöst werden können. Durch Rücktransformation der algebraischen Lösungen erhält man anschließend die gesuchten Lösungen. Das folgende Beispiel kommt aus der klassischen Mechanik:
Zwei Massen, m und $\alpha\, m$ (α sei ein Faktor), verbunden durch eine Feder, befinden sich auf einer reibungsfreien Fläche. Die Federkonstante sei k. Gesucht sind die Bewegungsbahnen, wenn die erste Masse zur Zeit t=1 angestoßen wird. Die Differentialgleichungen folgen aus dem zweiten Newtonschen Gesetz und dem Hookschen Gesetz für elastische Federn:
Kraft = Masse * Beschleunigung = k * Auslenkung

```
> eq1:=alpha*m*diff(x[1](t),t$2) = k*(x[2](t) - x[1](t)) + u(t);
```

$$eq1 := \alpha\, m\, (\tfrac{\partial^2}{\partial t^2}\, x_1(t)) = k\, (x_2(t) - x_1(t)) + \mathrm{u}(t)$$

```
> eq2:=m*diff(x[2](t),t$2) = k*(x[1](t) - x[2](t));
```

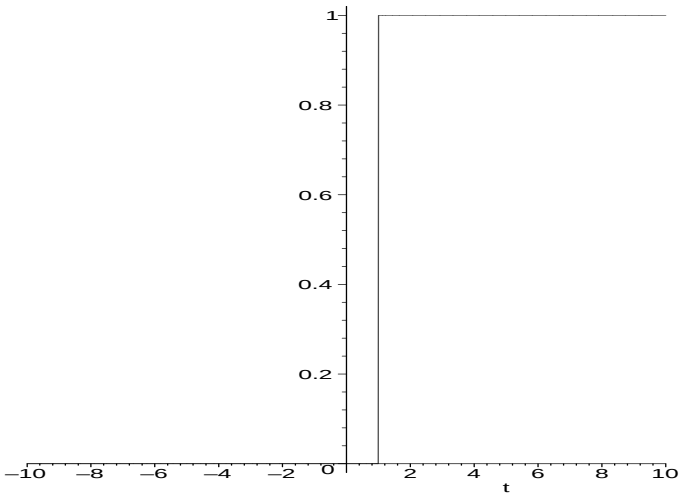
$$eq2 := m\, (\tfrac{\partial^2}{\partial t^2}\, x_2(t)) = k\, (x_1(t) - x_2(t))$$

Der Stoß zur Zeit t=1 an der ersten Masse werde durch eine Kraft in Form der Einheitssprungfunktion Heaviside beschrieben. Um den Schreibaufwand zu begrenzen und die Ergebnisse etwas übersichtlicher darstellen zu können, wird die Heaviside-Funktion durch σ abgekürzt.

```
> alias(sigma=Heaviside):
> u:=t -> sigma(t-1);
```

$$u := t \to \sigma(t-1)$$

```
> plot(u(t),t=-10..10);
```




```
> ini:=x[1](0) = 2,D(x[1])(0) = 0,
> x[2](0) = 0,D(x[2])(0) = 0;
```

$$ini := x_1(0) = 2,\, \mathrm{D}(x_1)(0) = 0,\, x_2(0) = 0,\, \mathrm{D}(x_2)(0) = 0$$

```
> #dsolve({eq1,eq2,ini},{x[1](t),x[2](t)});
```

Die Lösung ohne Laplace-Transformation erfordert viel CPU-Zeit!

```
> dsolve({eq1,eq2,ini},{x[1](t),x[2](t)},method=laplace);
```

$$\left\{x_1(t) = \frac{1}{2}((\alpha\,k - 2\,t\,k\,\alpha + t^2\,k\,\alpha + t^2\,k + 2\,m + k - m\,e^{(-\frac{\%1\,(t-1)}{m\,\alpha})} - m\,e^{(\frac{\%1\,(t-1)}{m\,\alpha})} - 2\,t\,k)\sigma(t-1))\Big/(k\,(\alpha+1)^2\,m) + \frac{e^{(-\frac{\%1\,t}{m\,\alpha})} + e^{(\frac{\%1\,t}{m\,\alpha})} + 2\,\alpha}{\alpha+1}, x_2(t) = \frac{1}{2}((-2\,m\,\alpha - 2\,t\,k\,\alpha + m\,\alpha\,e^{(-\frac{\%1\,(t-1)}{m\,\alpha})} + \alpha\,k + m\,\alpha\,e^{(\frac{\%1\,(t-1)}{m\,\alpha})} + t^2\,k\,\alpha + k - 2\,t\,k + t^2\,k)\sigma(t-1))\Big/(k\,(\alpha+1)^2\,m) - \frac{\alpha\,(-2 + e^{(-\frac{\%1\,t}{m\,\alpha})} + e^{(\frac{\%1\,t}{m\,\alpha})})}{\alpha+1}\right\}$$

$$\%1 := \sqrt{-m\,\alpha\,k\,(\alpha+1)}$$

Wir erhalten spezielle Lösungsfunktionen, wenn wir die allgemeine Lösung an speziellen Werten für die Konstanten α, m und k auswerten. Die resultierenden Ausdrücke werden anschließend in eine möglichst einfache Form transformiert:

```
> sol1:=collect(combine(simplify(eval(%,{alpha=1/10,m=1,k=1})),
> trig),sigma(t-1));
```

$$sol1 := \{x_1(t) = (\frac{155}{121} - \frac{10}{11}\,t + \frac{5}{11}\,t^2 - \frac{100}{121}\cos(\sqrt{11}\,t - \sqrt{11}))\,\sigma(t-1) + \frac{20}{11}\cos(\sqrt{11}\,t) + \frac{2}{11}, x_2(t) = (\frac{45}{121} - \frac{10}{11}\,t + \frac{5}{11}\,t^2 + \frac{10}{121}\cos(\sqrt{11}\,t - \sqrt{11}))\,\sigma(t-1) - \frac{2}{11}\cos(\sqrt{11}\,t) + \frac{2}{11}\}$$

Für die graphische Veranschaulichung definieren wir diese speziellen Lösungen als Maple-Funktionen y_1 und y_2.

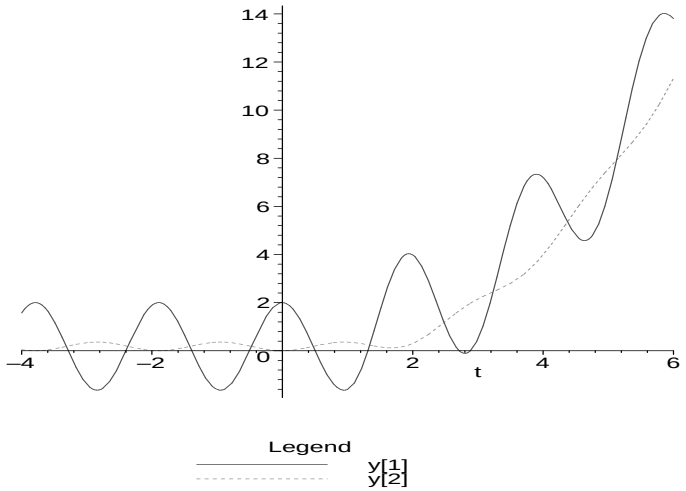
```
> y[1]:=unapply(eval(x[1](t),sol1),t);
```

$$y_1 := t \to (\frac{155}{121} - \frac{10}{11}\,t + \frac{5}{11}\,t^2 - \frac{100}{121}\cos(\sqrt{11}\,t - \sqrt{11}))\,\sigma(t-1) + \frac{20}{11}\cos(\sqrt{11}\,t) + \frac{2}{11}$$

```
> y[2]:=unapply(eval(x[2](t),sol1),t);
```

$$y_2 := t \to (\frac{45}{121} - \frac{10}{11}\,t + \frac{5}{11}\,t^2 + \frac{10}{121}\cos(\sqrt{11}\,t - \sqrt{11}))\,\sigma(t-1) - \frac{2}{11}\cos(\sqrt{11}\,t) + \frac{2}{11}$$

```
> plot([y[1](t),y[2](t)],t=-4..6, legend=["y[1]", "y[2]"]);
```



Die Lösung des obigen Differentialgleichungssystems kann man natürlich auch direkt mit Hilfe der Kommandos **laplace** für die Laplace-Transformation und **invlaplace** für die inverse Laplace-Transformation ermitteln.

Die Laplace-Transformation ist eine Integraltransformation, die für (Zeit-)Funktionen f(t) wie folgt erklärt ist:

```
> Int(f(t)*exp(-s*t),t=0..infinity);
```

$$\int_0^\infty \mathrm{f}(t)\, e^{(-s\,t)}\, dt$$

Wir führen nun die folgenden Abkürzungen ein, um die Gleichungen möglichst einfach zu halten:

```
> alias(F1(s)=laplace(x[1](t),t,s)):
> alias(F2(s)=laplace(x[2](t),t,s)):
> with(inttrans);
```

$$[addtable,\ fourier,\ fouriercos,\ fouriersin,\ hankel,\ hilbert,\ invfourier,\ invhilbert,\ invlaplace,\ invmellin,\ laplace,\ mellin,\ savetable]$$

```
> laplace(eq1,t,s);
```

$$m\,\alpha\,(s\,(s\,\mathrm{F1}(s)-x_1(0))-\mathrm{D}(x_1)(0))=k\,(\mathrm{F2}(s)-\mathrm{F1}(s))+\frac{e^{(-s)}}{s}$$

```
> laplace(eq2,t,s);
```

$$m\,(s\,(s\,\mathrm{F2}(s)-x_2(0))-\mathrm{D}(x_2)(0))=k\,(\mathrm{F1}(s)-\mathrm{F2}(s))$$

```
> eval({%,%%},{ini});
```

$$\{m\,\alpha\,s\,(s\,\mathrm{F1}(s)-2)=k\,(\mathrm{F2}(s)-\mathrm{F1}(s))+\frac{e^{(-s)}}{s},$$
$$m\,s^2\,\mathrm{F2}(s)=k\,(\mathrm{F1}(s)-\mathrm{F2}(s))\}$$

Dieses Gleichungssystem in den Laplace-Transformierten F1(s) und F2(s) ist linear und kann leicht mit solve gelöst werden.

```
> solve(%,{F1(s),F2(s)});
```

$$\{\mathrm{F1}(s)=\frac{(m\,s^2+k)\,(2\,m\,\alpha\,s^2\,e^s+1)}{e^s\,s^3\,m\,(k+m\,\alpha\,s^2+\alpha\,k)},\ \mathrm{F2}(s)=\frac{k\,(2\,m\,\alpha\,s^2\,e^s+1)}{e^s\,s^3\,m\,(k+m\,\alpha\,s^2+\alpha\,k)}\}$$

Die inverse Laplace-Transformation liefert schließlich die Lösungsfunktionen des ursprünglichen Differentialgleichungssystems.

```
> invlaplace(%,s,t);
```

$$\left\{ x_1(t) = \frac{1}{2}((\alpha\,k - 2\,t\,k\,\alpha + t^2\,k\,\alpha + t^2\,k + 2\,m + k - m\,e^{(-\frac{\%1\,(t-1)}{m\,\alpha})} - m\,e^{(\frac{\%1\,(t-1)}{m\,\alpha})} - 2\,t\,k)\sigma(t-1))\Big/((\alpha^2 + 2\,\alpha + 1)\,k\,m) + \frac{e^{(-\frac{\%1\,t}{m\,\alpha})} + e^{(\frac{\%1\,t}{m\,\alpha})} + 2\,\alpha}{\alpha + 1}, x_2(t) = \frac{1}{2}((-2\,m\,\alpha - 2\,t\,k\,\alpha + m\,\alpha\,e^{(-\frac{\%1\,(t-1)}{m\,\alpha})} + \alpha\,k + m\,\alpha\,e^{(\frac{\%1\,(t-1)}{m\,\alpha})} + t^2\,k\,\alpha + k - 2\,t\,k + t^2\,k)\sigma(t-1))\Big/(k\,(\alpha+1)^2\,m) - \frac{\alpha\,(-2 + e^{(-\frac{\%1\,t}{m\,\alpha})} + e^{(\frac{\%1\,t}{m\,\alpha})})}{\alpha + 1} \right\}$$

$$\%1 := \sqrt{-m\,\alpha\,k\,(\alpha+1)}$$

Für die Konstanten setzen wir wieder die obigen Werte ein und vereinfachen:

```
> sol2:=collect(combine(simplify(eval(%,{alpha=1/10,m=1,k=1})),
> trig),sigma(t-1));
```

$$sol2 := \{x_1(t) = (\frac{155}{121} - \frac{10}{11}\,t + \frac{5}{11}\,t^2 - \frac{100}{121}\cos(\sqrt{11}\,t - \sqrt{11}))\,\sigma(t-1) + \frac{20}{11}\cos(\sqrt{11}\,t) + \frac{2}{11}, x_2(t) = (\frac{45}{121} - \frac{10}{11}\,t + \frac{5}{11}\,t^2 + \frac{10}{121}\cos(\sqrt{11}\,t - \sqrt{11}))\,\sigma(t-1) - \frac{2}{11}\cos(\sqrt{11}\,t) + \frac{2}{11}\}$$

Diese speziellen Lösungen definieren wir nun als Maple-Funktionen $z_1$und z_2

```
> z[1]:=unapply(eval(x[1](t),sol2),t);
```

$$z_1 := t \to (\frac{155}{121} - \frac{10}{11}\,t + \frac{5}{11}\,t^2 - \frac{100}{121}\cos(\sqrt{11}\,t - \sqrt{11}))\,\sigma(t-1) + \frac{20}{11}\cos(\sqrt{11}\,t) + \frac{2}{11}$$

```
> z[2]:=unapply(eval(x[2](t),sol2),t);
```

$$z_2 := t \to (\frac{45}{121} - \frac{10}{11}\,t + \frac{5}{11}\,t^2 + \frac{10}{121}\cos(\sqrt{11}\,t - \sqrt{11}))\,\sigma(t-1) - \frac{2}{11}\cos(\sqrt{11}\,t) + \frac{2}{11}$$

und vergleichen sie dann mit den dsolve Lösungsfunktionen y_1 und y_2.

```
> simplify(z[1](t)-y[1](t));
```

$$0$$

```
> simplify(z[2](t)-y[2](t));
```

$$0$$

6.3 Näherungslösung durch Reihenentwicklung

```
> restart:
```

Eine weitere mögliche Option von dsolve lautet series (**dsolve,series**). Maple führt dann für die betreffenden Funktionen eine Reihenentwicklung durch und liefert die Lösung der Differentialgleichung als Reihe. Diese Vorgehensweise hat den Vorteil, daß unter Umständen auch dann eine symbolische Lösung berechnet werden kann, wenn dsolve eine exakte Lösung nicht mehr findet. Der Nachteil der series Option besteht darin, daß die resultierende Lösung nur eine Näherungslösung ist, die die exakte Lösung nur in einer (zumeist kleinen) Umgebung des Entwicklungspunktes gut approximiert. Beispiel:

```
> de1:=diff(y(t),t$2) +  diff(y(t),t) + y(t) = t + sin(t);
```

$$de1 := (\frac{\partial^2}{\partial t^2}\,\mathrm{y}(t)) + (\frac{\partial}{\partial t}\,\mathrm{y}(t)) + \mathrm{y}(t) = t + \sin(t)$$

```
> ini1:=y(0) = 0,D(y)(0) = 0;
```

$$ini1 := \mathrm{y}(0) = 0,\ \mathrm{D}(y)(0) = 0$$

Die Anzahl der Terme der Reihenentwicklung wird durch die globale Variable **Order** bestimmt, die mit dem Wert 6 voreingestellt ist.

```
> Order:=10:
> dsolve({de1,ini1},y(t),series);
```

$$\mathrm{y}(t) = \frac{1}{3}\,t^3 - \frac{1}{12}\,t^4 - \frac{1}{120}\,t^5 + \frac{1}{240}\,t^6 - \frac{1}{5040}\,t^7 - \frac{1}{20160}\,t^8 + \frac{1}{181440}\,t^9 + \mathrm{O}(t^{10})$$

```
> f[1]:=unapply(convert(rhs(%),polynom),t);
```

$$f_1 := t \to \frac{1}{3}\,t^3 - \frac{1}{12}\,t^4 - \frac{1}{120}\,t^5 + \frac{1}{240}\,t^6 - \frac{1}{5040}\,t^7 - \frac{1}{20160}\,t^8 + \frac{1}{181440}\,t^9$$

Diese Differentialgleichung kann Maple auch exakt lösen.

```
> dsolve({de1,ini1},y(t)):
> combine(%,trig);
```

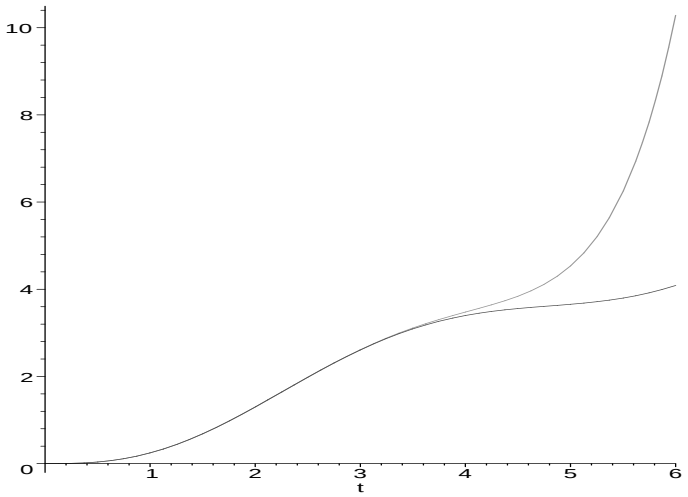
$$\mathrm{y}(t) = -1 - \cos(t) + t + 2\,e^{(-1/2\,t)}\cos(\frac{1}{2}\,\sqrt{3}\,t)$$

```
> f[2]:=unapply(rhs(%),t);
```

$$f_2 := t \to -1 - \cos(t) + t + 2\,e^{(-1/2\,t)}\cos(\frac{1}{2}\,\sqrt{3}\,t)$$

Die folgende Abbildung zeigt die Kurven der tatsächlichen Lösung und der Näherungslösung.

```
> plot({f[1](t),f[2](t)},t=0..6);
```



Wenn die Differentialgleichung symbolisch nicht gelöst werden kann, gibt Maple manchmal eine Darstellung in Form von **DESol** Termen zurück. Die Datenstruktur DESol ist mit RootOf vergleichbar und dient als Platzhalter für die Lösung der Differentialgleichung. Beispiel:

```
> de2:=diff(y(t),t$3) + (2*t+2)*diff(y(t),t$2) +
> (4*t+4-1/t)*diff(y(t),t) + y(t) + 1;
```

$$de2 := (\tfrac{\partial^3}{\partial t^3}\,\mathrm{y}(t)) + (2\,t+2)\,(\tfrac{\partial^2}{\partial t^2}\,\mathrm{y}(t)) + (4\,t+4-\frac{1}{t})\,(\tfrac{\partial}{\partial t}\,\mathrm{y}(t)) + \mathrm{y}(t) + 1$$

```
> sol:=dsolve(de2,y(t));
```

$$sol := \mathrm{y}(t) = -1 + \mathrm{DESol}(\{t\,(\tfrac{\partial^3}{\partial t^3}\,\_\mathrm{Y}(t)) + (2\,t^2+2\,t)\,(\tfrac{\partial^2}{\partial t^2}\,\_\mathrm{Y}(t)) + (4\,t^2+4\,t-1)\,(\tfrac{\partial}{\partial t}\,\_\mathrm{Y}(t)) + t\,\_\mathrm{Y}(t)\}, \{\_\mathrm{Y}(t)\})$$

Die mit dsolve bestimmte explizite oder implizite Lösung einer gewöhnlichen Differentialgleichung läßt sich mit **odetest** überprüfen:

```
> odetest(sol,de2);
```

$$0$$

Diese Lösung kann man nun mit dem Kommando **series** in eine Reihe entwickeln:

```
> y(t)=series(rhs(sol),t=0,4);
```

$$\mathrm{y}(t) = (-1-4\,\_C3) + (12\,\_C3 + 12\,\_C2)\,t + (\_C3\,(-6+6\ln(t)) + \_C1 + \_C2\,(6\ln(t)+5))\,t^2 + (\_C3\,(-\frac{20}{3} - 2\ln(t)) - \frac{1}{3}\,\_C1 + \_C2\,(-2\ln(t)-11))\,t^3 + \mathrm{O}(t^4)$$

6.4 Numerische Lösung

```
> restart:
```

Wenn dsolve keine symbolische Lösung der Differentialgleichung findet, besteht noch die Hoffnung, daß mit der Option numeric (**dsolve,numeric**) eine numerische Behandlung der Gleichung möglich ist. Voraussetzung für den Einsatz der Option numeric ist allerdings, daß die Differentialgleichung keine symbolischen Konstanten enthält und ausreichend Rand- und Nebenbedingungen formuliert werden können. dsolve bietet mit der Option numeric verschiedene Ausgabemöglichkeiten. Standardmäßig (output=procedurelist) gibt dsolve eine numerische Lösungsprozedur zurück. Beispiel:

```
> de1:=diff(x(t),t) = y(t), diff(y(t),t) = x(t)+y(t);
```

$$de1 := \tfrac{\partial}{\partial t}\,\mathrm{x}(t) = \mathrm{y}(t),\ \tfrac{\partial}{\partial t}\,\mathrm{y}(t) = \mathrm{x}(t) + \mathrm{y}(t)$$

```
> ini1:=x(0) = 2, y(0) = 1;
```

$$ini1 := \mathrm{x}(0) = 2,\ \mathrm{y}(0) = 1$$

```
> sol1:=dsolve({de1, ini1},{x(t),y(t)},type=numeric);
```

$$sol1 := \mathbf{proc}(rkf45\_x)\ \ldots\ \mathbf{end\ proc}$$

Für einzelne Zeitpunkte berechnet diese Lösungsprozedur eine Ergebnisliste.

```
> sol1(0);
```

$$[t = 0,\ \mathrm{x}(t) = 2.,\ \mathrm{y}(t) = 1.]$$

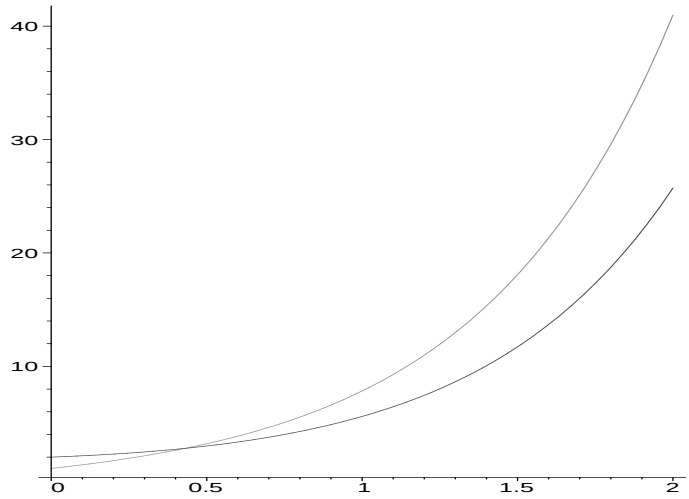
```
> sol1(1);
```

$$[t = 1,\ \mathrm{x}(t) = 5.58216868924484544,\ \mathrm{y}(t) = 7.82689113711079365]$$

Mit der Funktion odeplot aus dem Graphikpaket plots lassen sich die numerischen Lösungen direkt veranschaulichen.

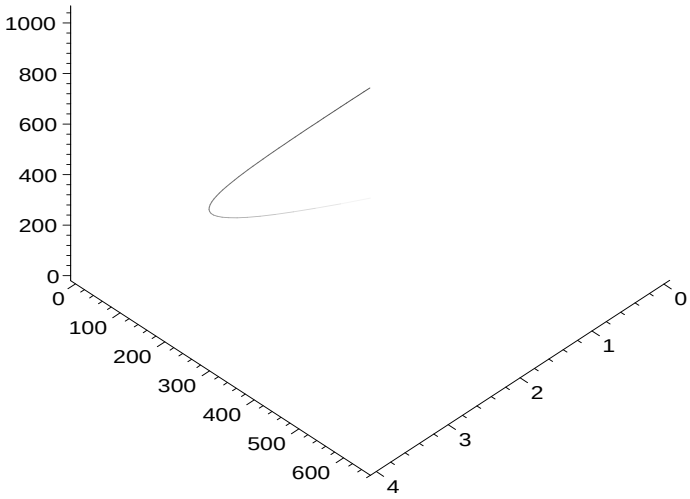
```
> with(plots):

Warning, the name changecoords has been redefined
```

```
> odeplot(sol1,[[t,x(t)],[t,y(t)]],0..2);
```



Die gleiche Lösung als Kurve im Raum:

```
> odeplot(sol1,[t,x(t),y(t)],0..4,axes=frame);
```



Mit output=listprocedure wird eine Liste von Gleichungen erzeugt. Unabhängige und abhängige Variablen stehen auf der linken Seite, Prozeduren stehen auf der rechten Seite.

```
> sol2:=dsolve({de1,ini1},{x(t),y(t)},type=numeric,
> output=listprocedure);
```

$$sol2 := [t = (\textbf{proc}(t) \ldots \textbf{end proc}), \mathrm{x}(t) = (\textbf{proc}(t) \ldots \textbf{end proc}),$$
$$\mathrm{y}(t) = (\textbf{proc}(t) \ldots \textbf{end proc})]$$

Durch Substitution lassen sich die Lösungsprozeduren aus der Liste extrahieren und dann plotten.

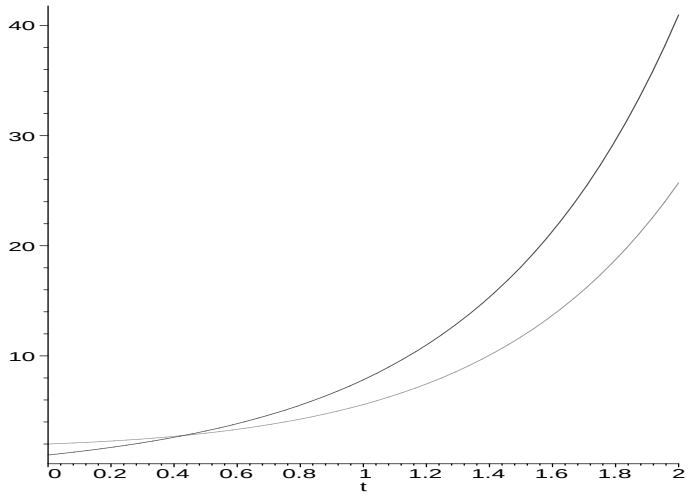
```
> fx:=eval(x(t),sol2);
```

$$fx := \textbf{proc}(t) \ldots \textbf{end proc}$$

```
> fy:=eval(y(t),sol2);
```

$$fy := \textbf{proc}(t) \ldots \textbf{end proc}$$

```
> plot({'fx(t)','fy(t)'},t=0..2);
```



Hier wird die symbolische Auswertung der numerischen Prozeduren fx und fy durch die Aposthrophierung verhindert. Mit der Option value=array([...]) werden die Lösungsfunktionen an vorgegebenen Punkten berechnet und die numerischen Werte in Form einer Tabelle (Matrix) ausgegeben.

```
> F:=dsolve({de1,ini1}, {x(t),y(t)},type=numeric,
> value=array([0,.6,1.1,1.5,2.3,2.5]));
```

$$F := \left[\begin{array}{c} [t,\ \mathrm{x}(t),\ \mathrm{y}(t)] \\ \left[\begin{array}{ccc} 0 & 2. & 1. \\ .6 & 3.330277376 & 3.845239673 \\ 1.1 & 6.435591567 & 9.279991634 \\ 1.5 & 11.72115275 & 18.08036901 \\ 2.3 & 41.56679155 & 66.71679091 \\ 2.5 & 57.32933204 & 92.28386849 \end{array}\right] \end{array}\right]$$

Neben diesen numerischen Lösungsmöglichkeiten hat Maple noch ein eigenes Paket **DEtools** für die graphische Analyse von Differentialgleichungen.

Kapitel 7

Lineare Algebra

7.1 Vektoroperationen

```
> restart:
```

Alle Funktionen im Paket **LinearAlgebra** werden zunächst aktiviert:

```
> with(LinearAlgebra);
```

$$[Add,\ Adjoint,\ BackwardSubstitute,\ BandMatrix,\ Basis,\ BezoutMatrix,\ BidiagonalForm,\ BilinearForm,\ CharacteristicMatrix,\ CharacteristicPolynomial,\ Column,\ ColumnDimension,\ ColumnOperation,\ ColumnSpace,\ CompanionMatrix,\ ConditionNumber,\ ConstantMatrix,\ ConstantVector,\ CreatePermutation,\ CrossProduct,\ DeleteColumn,\ DeleteRow,\ Determinant,\ DiagonalMatrix,\ Dimension,\ Dimensions,\ DotProduct,\ Eigenvalues,\ Eigenvectors,\ Equal,\ ForwardSubstitute,\ FrobeniusForm,\ GenerateEquations,\ GenerateMatrix,\ GetResultDataType,\ GetResultShape,\ GivensRotationMatrix,\ GramSchmidt,\ HankelMatrix,\ HermiteForm,\ HermitianTranspose,\ HessenbergForm,\ HilbertMatrix,\ HouseholderMatrix,\ IdentityMatrix,\ IntersectionBasis,\ IsDefinite,\ IsOrthogonal,\ IsSimilar,\ IsUnitary,\ JordanBlockMatrix,\ JordanForm,\ LA\_Main,\ LUDecomposition,\ LeastSquares,\ LinearSolve,\ Map,\ Map2,\ MatrixAdd,\ MatrixInverse,\ MatrixMatrixMultiply,\ MatrixNorm,\ MatrixScalarMultiply,\ MatrixVectorMultiply,\ MinimalPolynomial,\ Minor,\ Multiply,\ NoUserValue,\ Norm,\ Normalize,\ NullSpace,\ OuterProductMatrix,\ Permanent,\ Pivot,\ QRDecomposition,\ RandomMatrix,\ RandomVector,\ Rank,\ Row,\ RowDimension,\ RowOperation,\ RowSpace,\ ScalarMatrix,\ ScalarMultiply,\ ScalarVector,\ SchurForm,\ SingularValues,\ SmithForm,\ SubMatrix,\ SubVector,\ SumBasis,\ SylvesterMatrix,\ ToeplitzMatrix,\ Trace,\ Transpose,\ TridiagonalForm,\ UnitVector,\ VandermondeMatrix,\ VectorAdd,\ VectorAngle,\ VectorMatrixMultiply,\ VectorNorm,\ VectorScalarMultiply,\ ZeroMatrix,\ ZeroVector,\ Zip]$$

Im LinearAlgebra-Paket werden Vektoren mit der Funktion **Vector** konstruiert.

```
> V:=Vector(1..3);
```

$$V := \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Die zugrundeliegende Datenstruktur wird über die Routine **rtable** festgelegt. Konkrete Elemente eines Vektors lassen sich direkt über eine Liste eingeben:

```
> V1:=Vector([1/2,5/2,9]);
```

$$V1 := \begin{bmatrix} \frac{1}{2} \\ \frac{5}{2} \\ 9 \end{bmatrix}$$

oder mit Hilfe der folgenden **shortcut** Notation:

```
> V2:=<a|b|c>;
```

$$V2 := [a,\ b,\ c]$$

```
> V2:=<a,b,c>;
```

$$V2 := \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

Erfüllen die Elemente eines Vektors eine vorgegebene Gesetzmäßigkeit, so kann man den Vektor mit Hilfe einer anonymen Prozedur erzeugen:

```
> V3:=Vector(3,n->sin(n*Pi/4));
```

$$V3 := \begin{bmatrix} \frac{1}{2}\sqrt{2} \\ 1 \\ \frac{1}{2}\sqrt{2} \end{bmatrix}$$

Die Länge des Vektors wird hier durch den ersten Parameter des Vector Kommandos festgelegt. Einzelne Elemente eines Vektors können durch die Angabe des Index in eckigen Klammern angesprochen werden; Teile eines Vektors lassen sich mit Hilfe der **range** Notation extrahieren:

```
> V[2]; V3[2..3];
```

$$0$$

$$\begin{bmatrix} 1 \\ \frac{1}{2}\sqrt{2} \end{bmatrix}$$

Teile eines vorhandenen Vektors können wiederum für die Definition eines neuen Vektors verwendet werden (**SubVector**):

```
> SubVector(V3,[2..3,1]);
```

$$\begin{bmatrix} 1 \\ \frac{1}{2}\sqrt{2} \\ \frac{1}{2}\sqrt{2} \end{bmatrix}$$

Die verfügbaren Optionen zur Manipulation der Datenstruktur eines Vektors erhält man mit **VectorOptions**

```
> VectorOptions(V2);
```

$$shape = [],\ datatype = anything,\ orientation = column,\ storage = rectangular,$$
$$order = Fortran\_order$$

Die Länge bzw. der Betrag eines Vektors wird durch das **Norm** Kommando berechnet:

```
> Norm(V2,2);
```

$$\sqrt{|a|^2 + |b|^2 + |c|^2}$$

Die Ausführung der Addition zweier Vektoren und die Multiplikation eines Vektors mit einem Skalar erfolgt nach den Regeln der **Matrixalgebra**:

```
> V1 + V2 ;
```

$$\begin{bmatrix} \frac{1}{2}+a \\ \frac{5}{2}+b \\ 9+c \end{bmatrix}$$

```
> 2 * V2;
```

$$\begin{bmatrix} 2\,a \\ 2\,b \\ 2\,c \end{bmatrix}$$

Das Skalarprodukt ist durch das Kommando **DotProduct** (Punktprodukt) realisiert:

```
> DotProduct(V2,V3);
```

$$\frac{1}{2}\,\overline{(a)}\,\sqrt{2}+\overline{(b)}+\frac{1}{2}\,\overline{(c)}\,\sqrt{2}$$

Wenn der zweite Vektor komplexe Werte enthält, bildet Dotproduct vor der Multiplikation automatisch die konjugiert komplexen Werte. Durch die Option conjugate=false kann die Bildung konjugiert komplexer Werte vermieden werden:

```
> DotProduct(V2,V3,conjugate=false);
```

$$\frac{1}{2}\,a\,\sqrt{2}+b+\frac{1}{2}\,c\,\sqrt{2}$$

Der Winkel zwischen zwei Vektoren läßt sich mit der **VectorAngle** Funktion berechnen:

```
> VectorAngle(V1,V3);
```

$$\arccos(\frac{1}{70}\,(\frac{5}{2}+\frac{19}{4}\,\sqrt{2})\,\sqrt{14}\,\sqrt{2})$$

```
> evalf(%);
```

$$.7998997844$$

Das Ergebnis wird im Bogenmaß angegeben; mit **convert,degrees** erhält man die Angabe in Grad:

```
> convert(%,degrees);
```

$$143.9819612\,\frac{degrees}{\pi}$$

```
> evalf(%);
```

$$45.83088167\,degrees$$

Für das Kreuzprodukt (Vektorprodukt) steht der **CrossProduct** Befehl zur Verfügung:

```
> CrossProduct(V1,V2);
```

$$\begin{bmatrix} \frac{5}{2}\,c-9\,b \\ 9\,a-\frac{1}{2}\,c \\ \frac{1}{2}\,b-\frac{5}{2}\,a \end{bmatrix}$$

Mit Hilfe des Kreuzproduktes berechnet sich der Flächeninhalt des von zwei Vektoren aufgespannten Parallelogramms wie folgt:

```
> CP:=CrossProduct(V1,V3);
```

$$CP := \begin{bmatrix} \frac{5}{4}\,\sqrt{2}-9 \\ \frac{17}{4}\,\sqrt{2} \\ \frac{1}{2}-\frac{5}{4}\,\sqrt{2} \end{bmatrix}$$

```
> Flaeche:=evalf(Norm(CP,2));
```

$$Flaeche := 9.488805400$$

Mit **Normalize** kann man die Länge eines Vektors (gemessen in der 2-Norm) auf 1 normieren.

```
> Normalize(V1,2);
```

$$\begin{bmatrix} \frac{1}{70}\sqrt{14} \\ \frac{1}{14}\sqrt{14} \\ \frac{9}{35}\sqrt{14} \end{bmatrix}$$

```
> Norm(%,2);
```

$$1$$

Das Spatprodukt läßt sich als Kombination von Skalarprodukt und Kreuzprodukt darstellen. Wir definieren zunächst drei Zeilenvektoren mit unbestimmten Koeffizienten:

```
> A:=Vector[row](3,i->a[i]);
> B:=Vector[row](3,i->b[i]);
> C:=Vector[row](3,i->c[i]);
```

$$A := [a_1,\ a_2,\ a_3]$$
$$B := [b_1,\ b_2,\ b_3]$$
$$C := [c_1,\ c_2,\ c_3]$$

Für das Volumen eines Spates fogt dann:

```
> Vol:=abs(DotProduct(A,CrossProduct(B,C),conjugate=false));
```

$$Vol := |(b_2\,c_3 - b_3\,c_2)\,a_1 + (b_3\,c_1 - b_1\,c_3)\,a_2 + (b_1\,c_2 - b_2\,c_1)\,a_3|$$

7.2 Matrizenrechnung

```
> restart:
> with(LinearAlgebra):
```

Eine Matrix wird in LinearAlgebra durch ein zweidimensionales Feld repräsentiert, dessen Zeilen- und Spaltenindizes mit 1 beginnen. Das Standardkommando zur Definition einer Matrix lautet **Matrix** .

```
> Matrix(3,2);
```

$$\begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

Die Matrixelemente sind mit 0 vorbesetzt. Mit der shortcut Notation lassen sich einzelne Matrixelemente eingeben.

```
> A:=«a|b>,<c|d>,<e|f»;
```

$$A := \begin{bmatrix} a & b \\ c & d \\ e & f \end{bmatrix}$$

Die Elemente einer Matrix können auch mittels einer geschachtelten Liste übergeben werden:

```
> Matrix([[1,1],[2,2],[3,3]]);
```

$$\begin{bmatrix} 1 & 1 \\ 2 & 2 \\ 3 & 3 \end{bmatrix}$$

Die Listen werden hierbei als Zeilen der Matrix interpretiert. Die Initialisierung der Matrixelemente kann auch über eine zweiparametrige anonyme Funktion erfolgen. Der erste Parameter dieser Funktion enthält den Zeilenindex, der zweite Parameter den Spaltenindex.

```
> Matrix(3,3,(n,m) -> n^m);
```

$$\begin{bmatrix} 1 & 1 & 1 \\ 2 & 4 & 8 \\ 3 & 9 & 27 \end{bmatrix}$$

Für die Erzeugung von Spezialmatrizen stehen zahlreiche Kommandos (**IdentityMatrix**, **DiagonalMatrix**, **BandMatrix**, **ToeplitzMatrix**, **HilbertMatrix**, **RandomMatrix**, ...) zur Verfügung, z. B.

```
> IdentityMatrix(3);
```

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

```
> DiagonalMatrix([1,2,3]);
```

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

```
> BandMatrix([-1,1,-1],1,4);
```

$$\begin{bmatrix} 1 & -1 & 0 & 0 \\ -1 & 1 & -1 & 0 \\ 0 & -1 & 1 & -1 \\ 0 & 0 & -1 & 1 \end{bmatrix}$$

```
> ToeplitzMatrix([1,2,3,4,5]);
```

$$\begin{bmatrix} 3 & 2 & 1 \\ 4 & 3 & 2 \\ 5 & 4 & 3 \end{bmatrix}$$

```
> HilbertMatrix(3);
```

$$\begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} \end{bmatrix}$$

```
> RandomMatrix(3,3,generator=0..1.);
```

$$\begin{bmatrix} .040523676814643750 0 & .740557213285480742 & .677656650071760680 \\ .146020650134548902 & .0692927748006439970 & .734176186536910081 \\ .0346728035936152424 & .275536701743494206 & .703487905570597150 \end{bmatrix}$$

Durch die Option generator=0..1. wird das Intervall für die Zufallszahlen festgelegt. Der Zugriff auf ein einzelnes Matrixelement erfolgt durch die Angabe der Indizes (zuerst Zeile, dann Spalte):

```
> A[1,1];
```

$$a$$

Für den Zugriff auf Teilmatrizen gibt es das **SubMatrix** Kommando:

```
> SubMatrix(A,2..3,1..2);
```

$$\begin{bmatrix} c & d \\ e & f \end{bmatrix}$$

Das Lesen ganzer Zeilen und Spalten erfolgt mit den Befehlen **Column** und **Row**:

```
> Column(A,1);
```

$$\begin{bmatrix} a \\ c \\ e \end{bmatrix}$$

```
> Row(A,2..3);
```

$$[c,\ d],\ [e,\ f]$$

Addition und Subtraktion von Matrizen und die Multiplikation einer Matrix mit einem Skalar werden mit +, -, * gekennzeichnet; für die nicht-kommutative Matrixmultiplikation wird der **dot** Operator verwendet, da "*" für die kommutative Multiplikation reserviert ist.

```
> B:=Matrix([[r,s,t],[u,v,w]]);
```

$$B := \begin{bmatrix} r & s & t \\ u & v & w \end{bmatrix}$$

```
> A.B <> B.A;
```

$$\begin{bmatrix} a\,r+b\,u & a\,s+b\,v & a\,t+b\,w \\ c\,r+d\,u & c\,s+d\,v & c\,t+d\,w \\ e\,r+f\,u & e\,s+f\,v & e\,t+f\,w \end{bmatrix} \neq \begin{bmatrix} a\,r+c\,s+e\,t & r\,b+s\,d+t\,f \\ u\,a+v\,c+w\,e & b\,u+d\,v+f\,w \end{bmatrix}$$

Transpose transponiert eine Matrix, d.h. die Elemente werden entlag der Hauptdiagonalen vertauscht:

```
> Transpose(B);
```

$$\begin{bmatrix} r & u \\ s & v \\ t & w \end{bmatrix}$$

```
> 2*A - 3*Transpose(B);
```

$$\begin{bmatrix} 2\,a-3\,r & 2\,b-3\,u \\ 2\,c-3\,s & 2\,d-3\,v \\ 2\,e-3\,t & 2\,f-3\,w \end{bmatrix}$$

Die Addition eines Skalars führt zur Addition der Einheitsmatrix multipliziert mit dem Skalar:

```
> B + 2;
```

$$\begin{bmatrix} r+2 & s & t \\ u & v+2 & w \end{bmatrix}$$

Die Potenz einer Matrix entspricht der wiederholten Matrixmultiplikation:

```
> M:=Matrix([[1,2],[3,4]]);
```

$$M := \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

```
> M^3 = M.M.M;
```

$$\begin{bmatrix} 37 & 54 \\ 81 & 118 \end{bmatrix} = \begin{bmatrix} 37 & 54 \\ 81 & 118 \end{bmatrix}$$

Die Determinante einer Matrix wird mit **Determinant** und die Inverse einer Matrix mit **MatrixInverse** berechnet.

```
> Determinant(M);
```

$$-2$$

```
> Minv:=MatrixInverse(M);
```

$$Minv := \begin{bmatrix} -2 & 1 \\ \frac{3}{2} & \frac{-1}{2} \end{bmatrix}$$

```
> 'M.Minv' = M.Minv;
```

$$M\ .\ Minv = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

7.3 Lineare Gleichungssysteme

```
> restart:
> with(LinearAlgebra):
```

Die Konditionszahl (**ConditionNumber**) einer Matrix

```
> A:=Matrix([[1,0,3],[-4,2,0],[0,3,-2]]);
```

$$A := \begin{bmatrix} 1 & 0 & 3 \\ -4 & 2 & 0 \\ 0 & 3 & -2 \end{bmatrix}$$

wird über die Matrixnorm nach der Formel

```
> 'ConditionNumber(A)'=Norm(A)*Norm(MatrixInverse(A));
```

$$\mathrm{ConditionNumber}(A) = \frac{33}{10}$$

berechnet. Sie spielt eine große Rolle bei linearen Gleichungssystemen $Ax = b$ mit fehlerbehaftetem b. Eine Änderung $\Delta\, b$ hat eine Änderung der Lösung $\Delta\, x$ zur Folge. Es läßt sich zeigen, daß der relative Fehler von x kleiner oder gleich dem relativen Fehler von b multipliziert mit der Konditionzahl $\mathrm{cond}(A)$ ist:

$\frac{|\Delta\, x|}{|x|} \leq \frac{|\Delta\, b|\ \mathrm{ConditionNumber}(A)}{|b|}$

Ein Gleichungssystem ist daher schlecht konditioniert, wenn die Konditionszahl viel größer als 1 ist. Beispiel:

Betrachten wir die Polynominterpolation durch vorgegebene Punkte (x_1, y_1), (x_2, y_2), ..., (x_{n+1}, y_{n+1}) mit einem Polynom $P_n(x) = a_0$ + ...+ $a_n\, x^n$ *n*-ten Grades, so erhalten wir die Bestimmungsgleichungen $P_n(x_i) = y_i$ für die gesuchten Polynomkoeffizienten a_0, a_1,..., a_n in Matrixform $Aa = y$ aus**:**

```
> n:=5:
> X:=Vector[row](n,i->x[i]);
```

$$X := [x_1,\ x_2,\ x_3,\ x_4,\ x_5]$$

```
> A:=VandermondeMatrix(X);
```

$$A := \begin{bmatrix} 1 & x_1 & {x_1}^2 & {x_1}^3 & {x_1}^4 \\ 1 & x_2 & {x_2}^2 & {x_2}^3 & {x_2}^4 \\ 1 & x_3 & {x_3}^2 & {x_3}^3 & {x_3}^4 \\ 1 & x_4 & {x_4}^2 & {x_4}^3 & {x_4}^4 \\ 1 & x_5 & {x_5}^2 & {x_5}^3 & {x_5}^4 \end{bmatrix}$$

Die Koeffizientenmatrix hat die Form einer Vandermonde-Matrix **(VandermondeMatrix).**

```
> x:=Vector[row](n,i->1+i*0.1);
```

$$x := [1.1,\ 1.2,\ 1.3,\ 1.4,\ 1.5]$$

```
> A:=Matrix(5,5,(i,j)->A[i,j]);
```

$$A := \begin{bmatrix} 1 & 1.1 & 1.21 & 1.331 & 1.4641 \\ 1 & 1.2 & 1.44 & 1.728 & 2.0736 \\ 1 & 1.3 & 1.69 & 2.197 & 2.8561 \\ 1 & 1.4 & 1.96 & 2.744 & 3.8416 \\ 1 & 1.5 & 2.25 & 3.375 & 5.0625 \end{bmatrix}$$

A

ist eine schlecht konditionierte Matrix, d.h. kleine Meßunsicherheiten im Datenvektor y rufen große Änderungen $\Delta\, a$ in den Polynomkoeffizienten hervor.

```
> ConditionNumber(A,Frobenius);
```

$$539487.2910$$

$\mathrm{ConditionNumber}(A,\ \mathit{Frobenius})$ ist die Euklidische Matrixnorm (Wurzel aus der Betragsquadratsumme aller Matrixelemente).

```
> y1:=Vector[row](n,i->sin(1+i*0.1));
```

$$y1 := [.8912073601,\ .9320390860,\ .9635581854,\ .9854497300,\ .9974949866]$$

Wir lösen nun das lineare Gleichungssystem $Ax = y1$ mit dem Kommando **LinearSolve**:

```
> a1:=LinearSolve(A,y1);
```

$$a1 := [.0137858150991867490,\ .942415800585900243,\ .0980857320803062011,$$
$$-.252894508331752477,\ .0400812916663582930]$$

Der Vektor $y1$ wird nun ein wenig verändert:

```
> y2:=Vector[row](n,[y1[1]*1.01, y1[2]*0.99, y1[3]*0.995, y1[4]*1.003,
> y1[5]*1.002]);
```

$$y2 := [.9001194337,\ .9227186951,\ .9587403945,\ .9884060792,\ .9994899766]$$

```
> a2:=LinearSolve(A,y2);
```

$$a2 := [14.7542787186289922,\ -39.9139724351729016,\ 41.9621764096866414,\ -19.0311589917193871,\ 3.14695004167671000]$$

```
> Norm(y1-y2,Frobenius)/Norm(y1,Frobenius);
```

$$.006661239060$$

```
> % * ConditionNumber(A,Frobenius);
```

$$3593.653815$$

```
> Norm(a1-a2,Frobenius)/Norm(a1,Frobenius);
```

$$64.44299860$$

Die Ungleichung $\frac{|\Delta\, a|}{|a|} \leq \frac{|\Delta\, y|\ \mathrm{ConditionNumber}(A)}{|y|}$

ist zwar immer noch erfüllt, aber die Schwankungen $\Delta\, a$ der Poynomkoeffizienten sind inakzeptabel groß. Nun wollen wir die Gaußelimination zur Lösung linearer Gleichungssysteme verwenden. Dabei formt man die Koeffizientenmatrix durch Zeilenoperationen in eine obere Dreiecksmatrix um und wendet auf b dieselben Zeilenoperationen an. Mittels Rückwärtseinsetzen erhält man dann die gesuchte Lösung. Beispiel:

```
> A:=Matrix([[3,-13,9,3],[-6,4,1,-18],[6,-2,2,4],[12,-8,6,10]]);
```

$$A := \begin{bmatrix} 3 & -13 & 9 & 3 \\ -6 & 4 & 1 & -18 \\ 6 & -2 & 2 & 4 \\ 12 & -8 & 6 & 10 \end{bmatrix}$$

```
> b:=Vector([-19,-34,16,26]);
```

$$b := \begin{bmatrix} -19 \\ -34 \\ 16 \\ 26 \end{bmatrix}$$

Die Matrix A und der Vektor b werden nun zur erweiterten Matrix B zusammengefügt.

```
> B:=Matrix([A,b]);
```

$$B := \begin{bmatrix} 3 & -13 & 9 & 3 & -19 \\ -6 & 4 & 1 & -18 & -34 \\ 6 & -2 & 2 & 4 & 16 \\ 12 & -8 & 6 & 10 & 26 \end{bmatrix}$$

Durch LU-Zerlegung (**LUDecomposition**) soll nun die obere Dreiecksmatrix ermittelt werden.

```
> L,det,r:=LUDecomposition(B,method=GaussianElimination,
> output=[U,determinant,rank]);
```

$$L,\ det,\ r := \begin{bmatrix} 3 & -13 & 9 & 3 & -19 \\ 0 & -22 & 19 & -12 & -72 \\ 0 & 0 & \frac{52}{11} & \frac{-166}{11} & \frac{-270}{11} \\ 0 & 0 & 0 & \frac{-6}{13} & \frac{-6}{13} \end{bmatrix},\ 144,\ 4$$

Mit method=GaussianElimination wird die Faktorisierungsmethode angegeben, output=[U,determinant,rank] legt die Ausgabe fest (obere Dreiecksmatrix, Determinante und Rang der Matrix).

```
> LUDecomposition(B,method=FractionFree,output=[U]);
```

$$\begin{bmatrix} 3 & -13 & 9 & 3 & -19 \\ 0 & -66 & 57 & -36 & -216 \\ 0 & 0 & -312 & 996 & 1620 \\ 0 & 0 & 0 & 144 & 144 \end{bmatrix}$$

Mit der Option method=FractionFree wird die obere Dreiecksmatrix bruchfrei. Die Funktion **BackwardSubstitute** führt das Rückwärtseinsetzen durch und liefert den gesuchten Lösungsvektor:

```
> x:=BackwardSubstitute(L);
```

$$x := \begin{bmatrix} 3 \\ 1 \\ -2 \\ 1 \end{bmatrix}$$

Eine der Gauß-Elimination verwandte Methode ist das Gauß-Jordan-Verfahren (RREF, Reduced Row Echelon Form). Bei dieser Methode wird die Matrix auf Diagonalform gebracht; damit erspart man sich das Rückwärtseinsetzen, der Lösungsvektor steht in der letzten Spalte der Matrix:

```
> LUDecomposition(B,method=RREF,output=[R]);
```

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 3 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & -2 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

Wenn die Determinante verschwindet, ist das Gleichungssystem entweder nicht lösbar, oder die Lösung enthält noch freie Parameter.

```
> A:=Matrix([[1,2,3],[4,5,6],[7,8,9]]);
```

$$A := \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

```
> Determinant(A);
```

$$0$$

```
> b:=Vector([1,2,3]);
```

$$b := \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

```
> LUDecomposition(Matrix([A,b]),method=GaussianElimination,
> output=[U]);
```

$$\begin{bmatrix} 1 & 2 & 3 & 1 \\ 0 & -3 & -6 & -2 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

```
> BackwardSubstitute(%);
```

$$\begin{bmatrix} -\frac{1}{3} + \_t18_1 \\ \frac{2}{3} - 2\,\_t18_1 \\ \_t18_1 \end{bmatrix}$$

Die Lösung besitzt einen freien Parameter.

7.4 Diagonalisierung

```
> restart:
> with(LinearAlgebra):
```

Funktionen zur Lösung des Eigenwertproblems $A\,x = \lambda\,x$.

```
> A:=Matrix([[1,2],[3,4]]);
```

$$A := \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

Konstruktion der charakteristischen Matrix $A - \lambda I$ (**CharacteristicMatrix**); :

```
> CharacteristicMatrix(A,lambda);
```

$$\begin{bmatrix} -\lambda + 1 & 2 \\ 3 & -\lambda + 4 \end{bmatrix}$$

Das charakteristische Polynom ist somit

```
> Determinant(%);
```

$$\lambda^2 - 5\,\lambda - 2$$

Direkt erhält man das charakteristische Polynom mit der Funktion **CharacteristicPolynomial**

```
> P:=unapply(CharacteristicPolynomial(A,lambda),lambda);
```

$$P := \lambda \rightarrow \lambda^2 - 5\,\lambda - 2$$

Die Eigenwerte werden dann durch Berechnung der Nullstellen des charakteristischen Polynoms bestimmt:

```
> solve(P(lambda)=0,lambda);
```

$$\frac{5}{2} + \frac{1}{2}\sqrt{33},\ \frac{5}{2} - \frac{1}{2}\sqrt{33}$$

Direkt erhält man die Eigenwerte mit der Funktion **Eigenvalues**:

```
> Eigenvalues(A);
```

$$\begin{bmatrix} \frac{5}{2} + \frac{1}{2}\sqrt{33} \\ \frac{5}{2} - \frac{1}{2}\sqrt{33} \end{bmatrix}$$

```
> evalf(%);
```

$$\begin{bmatrix} 5.372281323 \\ -.372281323 \end{bmatrix}$$

Satz von Hamilton-Cayley:

```
> P(A);
```

$$\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

Bestimmung der Eigenvektoren mit der Funktion **Eigenvectors**:

```
> ev:=Eigenvectors(A,output=[list]);
```

$$ev := \left[\left[\frac{5}{2} + \frac{1}{2}\sqrt{33},\ 1,\ \left\{\begin{bmatrix} \frac{3}{4} + \frac{1}{4}\sqrt{33} \\ 1 \end{bmatrix}\right\}\right],\ \left[\frac{5}{2} - \frac{1}{2}\sqrt{33},\ 1,\ \left\{\begin{bmatrix} \frac{3}{4} - \frac{1}{4}\sqrt{33} \\ 1 \end{bmatrix}\right\}\right]\right]$$

Das Kommando Eigenvectors gibt als Ergebnis eine Liste von Listen zurück. Jede Teilliste besteht aus drei Elementen, wobei das erste Element ein Eigenwert und das zweite Elemente die zugehörige Vielfachheit ist. Das dritte Element stellt eine Menge unabhängiger Eigenvektoren zum Eigenwert dar.

```
> for i to 2 do
> lambda[i]:=ev[i][1];
> u[i]:=map(simplify,Normalize(op(ev[i][3]),2));
> od;
```

$$\lambda_1 := \frac{5}{2} + \frac{1}{2}\sqrt{33}$$

$$u_1 := \begin{bmatrix} 4\,\dfrac{1}{\sqrt{58 + 6\sqrt{33}}} \\ \dfrac{3 + \sqrt{33}}{\sqrt{58 + 6\sqrt{33}}} \end{bmatrix}$$

$$\lambda_2 := \frac{5}{2} - \frac{1}{2}\sqrt{33}$$

$$u_2 := \begin{bmatrix} 4\dfrac{1}{\sqrt{58-6\sqrt{33}}} \\ -\dfrac{-3+\sqrt{33}}{\sqrt{58-6\sqrt{33}}} \end{bmatrix}$$

Durch die Funktion **Normalize** werden die Eigenvektoren auf die Länge 1 normiert.
Diagonalisieren der Matrix A: Wir bilden eine Matrix U, deren Spalten die normierten Eigenvektoren von A sind:

```
> U:=Matrix([u[1],u[2]]);
```

$$U := \begin{bmatrix} 4\dfrac{1}{\sqrt{58+6\sqrt{33}}} & 4\dfrac{1}{\sqrt{58-6\sqrt{33}}} \\ \dfrac{3+\sqrt{33}}{\sqrt{58+6\sqrt{33}}} & -\dfrac{-3+\sqrt{33}}{\sqrt{58-6\sqrt{33}}} \end{bmatrix}$$

Überführung in Diagonalform, die Diagonalelemente sind die Eigenwerte von A:

```
> Map(expand,MatrixInverse(U).A.U);
```

$$\begin{bmatrix} \dfrac{5}{2}+\dfrac{1}{2}\sqrt{33} & 0 \\ 0 & \dfrac{5}{2}-\dfrac{1}{2}\sqrt{33} \end{bmatrix}$$

Lösung des verallgemeinerten Eigenwertproblems $A\,x = \lambda\,B\,x$:

```
> A:=Matrix([[1,2],[3,4]]);
```

$$A := \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

```
> B:=Matrix([[1,1],[2,4]]);
```

$$B := \begin{bmatrix} 1 & 1 \\ 2 & 4 \end{bmatrix}$$

Bestimmung der Eigenwerte und Eigenvektoren mit der Funktion Eigenvectors:

```
> ev:=Eigenvectors(A,B,output=[list]);
```

$$ev := \left[\left[\frac{1}{4}+\frac{1}{4}\sqrt{17},\ 1,\ \left\{\begin{bmatrix} \dfrac{1}{2}+\dfrac{1}{2}\sqrt{17} \\ 1 \end{bmatrix}\right\}\right],\ \left[\frac{1}{4}-\frac{1}{4}\sqrt{17},\ 1,\ \left\{\begin{bmatrix} \dfrac{1}{2}-\dfrac{1}{2}\sqrt{17} \\ 1 \end{bmatrix}\right\}\right]\right]$$

Berechnung von normierten Eigenvektoren:

```
> lambda:='lambda':
> for i to 2 do
> lambda[i]:=ev[i][1];
> u[i]:=map(simplify,Normalize(op(ev[i][3]),2));
> od;
```

$$\lambda_1 := \frac{1}{4}+\frac{1}{4}\sqrt{17}$$

$$u_1 := \begin{bmatrix} \dfrac{1+\sqrt{17}}{\sqrt{22+2\sqrt{17}}} \\ 2\dfrac{1}{\sqrt{22+2\sqrt{17}}} \end{bmatrix}$$

$$\lambda_2 := \frac{1}{4}-\frac{1}{4}\sqrt{17}$$

$$u_2 := \begin{bmatrix} -\dfrac{-1+\sqrt{17}}{\sqrt{22-2\sqrt{17}}} \\ 2\dfrac{1}{\sqrt{22-2\sqrt{17}}} \end{bmatrix}$$

```
> evalf(u[1]);
```

$$\begin{bmatrix} .9315320884 \\ .3636591382 \end{bmatrix}$$

```
> evalf(u[2]);
```

$$\begin{bmatrix} -.8421229398 \\ .5392855962 \end{bmatrix}$$

7.5 Matrixzerlegung

```
> restart:
> with(LinearAlgebra):
```

In der numerischen Mathematik spielen Matrixzerlegungen wie die QR-Zerlegung, die Cholesky-Zerlegung und die Singulärwertzerlegung eine bedeutende Rolle.

1. QR-Zerlegung

Jede quadratische Matrix A kann in eine orthonormale Matrix Q mit $Q^T\,Q = I$ und eine obere Dreiecksmatrix R der Form $A = Q\,R$ zerlegt werden.

```
> A:=Matrix([[1,1,1],[1,4,5],[2,3,1]]);
```

$$A := \begin{bmatrix} 1 & 1 & 1 \\ 1 & 4 & 5 \\ 2 & 3 & 1 \end{bmatrix}$$

Mit der Funktion **QRDecomposition** läßt sich die obere Dreiecksmatrix berechnen:

```
> Q,R:=QRDecomposition(A);
```

$$Q,\,R := \begin{bmatrix} \frac{1}{6}\sqrt{6} & -\frac{1}{42}\sqrt{210} & \frac{1}{7}\sqrt{35} \\ \frac{1}{6}\sqrt{6} & \frac{13}{210}\sqrt{210} & \frac{1}{35}\sqrt{35} \\ \frac{1}{3}\sqrt{6} & -\frac{2}{105}\sqrt{210} & -\frac{3}{35}\sqrt{35} \end{bmatrix}, \begin{bmatrix} \sqrt{6} & \frac{11}{6}\sqrt{6} & \frac{4}{3}\sqrt{6} \\ 0 & \frac{1}{6}\sqrt{210} & \frac{4}{15}\sqrt{210} \\ 0 & 0 & \frac{1}{5}\sqrt{35} \end{bmatrix}$$

Überprüfung der Zerlegung:

```
> Q.R;
```

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 4 & 5 \\ 2 & 3 & 1 \end{bmatrix}$$

2. Cholesky-Zerlegung

Jede positiv definite Matrix A kann in ein Produkt $A = L\,L^T$ zerlegt werden, wobei L eine untere Dreiecksmatrix mit positiven Diagonalelementen ist.

```
> A:=Matrix([[7,4,3],[4,5,2],[3,2,10]]);
```

$$A := \begin{bmatrix} 7 & 4 & 3 \\ 4 & 5 & 2 \\ 3 & 2 & 10 \end{bmatrix}$$

Mit der Funktion **IsDefinite** läßt sich überprüfen, ob die Matrix A positiv definit ist:

```
> IsDefinite(A,query='positive_def');
```

$$true$$

Zur Berechnung der unteren Dreiecksmatirx L verwenden wir die Funktion **LUDecomposition:**

```
> L:=LUDecomposition(A,method=Cholesky);
```

$$L := \begin{bmatrix} \sqrt{7} & 0 & 0 \\ \frac{4}{7}\sqrt{7} & \frac{1}{7}\sqrt{133} & 0 \\ \frac{3}{7}\sqrt{7} & \frac{2}{133}\sqrt{133} & \frac{1}{19}\sqrt{3135} \end{bmatrix}$$

Überprüfung, ob $A = LL^T$ erfüllt ist:

```
> L.Transpose(L);
```

$$\begin{bmatrix} 7 & 4 & 3 \\ 4 & 5 & 2 \\ 3 & 2 & 10 \end{bmatrix}$$

3. Singulärwertzerlegung

Die Singulärwertzerlegung einer $m\,x\,n$ Matrix A mit $n \leq m$ hat die Form $A = USV^T$ und kann mit der Funktion **SingularValues** berechnet werden. U ist eine $m\,x\,m$ Orthogonalmatrix, V eine $n\,x\,n$ Orthogonalmatrix, S ist eine $n\,x\,n$ Diagonalmatrix mit den Singulärwerten $0 < s_i$ als Elemente. Die Singulärwerte werden von der Funktion SingularValues zurückgeliefert. Diese Zerlegung ist in der numerischen Mathematik zur Lösung von Fehlergleichungen von Bedeutung.

```
> A:=Matrix([[1,4],[2,5],[3,6]]);
```

$$A := \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}$$

```
> SingularValues(A);
```

$$\begin{bmatrix} 0 \\ \frac{1}{2}\sqrt{182+2\sqrt{8065}} \\ \frac{1}{2}\sqrt{182-2\sqrt{8065}} \end{bmatrix}$$

Anmerkung: Die Singulärwerte der Matrix A lassen sich direkt als Quadratwurzel aus den Eigenwerten der Matrix $A^T\,A$ bestimmen:

```
> Map(sqrt,Eigenvalues(Transpose(A).A));
```

$$\begin{bmatrix} \frac{1}{2}\sqrt{182+2\sqrt{8065}} \\ \frac{1}{2}\sqrt{182-2\sqrt{8065}} \end{bmatrix}$$

```
> evalf(%);
```

$$\begin{bmatrix} 9.508032000 \\ .7728696365 \end{bmatrix}$$

Die Orthogonalmatrizen U und V werden von der Funktion SingularValues nur dann berechnet, wenn die Matrixelemente von A als Gleitkommazahlen eingelesen werden:

```
> A:=evalf(A);
```

$$A := \begin{bmatrix} 1. & 4. \\ 2. & 5. \\ 3. & 6. \end{bmatrix}$$

```
> U,S,V:=SingularValues(A,output=['U','S','Vt']);
```

$$U,\ S,\ V := \begin{bmatrix} -.428667133548626066 & .805963908589297450 & .408248290463862850 \\ -.566306918848035078 & .112382414096593650 & -.816496580927726034 \\ -.703946704147444091 & -.581199080396110013 & .408248290463863128 \end{bmatrix},$$

$$\begin{bmatrix} 9.50803200069572618 \\ .772869635673484767 \\ 0. \end{bmatrix}, \begin{bmatrix} -.386317703118611356 & -.922365780077058273 \\ -.922365780077058273 & .386317703118611356 \end{bmatrix}$$

Überprüfung der Orthogonalität von U und V :

```
> Transpose(U).U;
```

$$\begin{bmatrix} .99999999999999978, & .111022302462515654\,10^{-15}, & -.555111512312578272\,10^{-16} \\ .111022302462515654\,10^{-15}, & .99999999999999966, & -.138777878078144568\,10^{-15} \\ -.555111512312578272\,10^{-16}, & -.138777878078144568\,10^{-15}, & .99999999999999988 \end{bmatrix}$$

```
> Transpose(V).V;
```

$$\begin{bmatrix} .99999999999999978 & 0. \\ 0. & .99999999999999978 \end{bmatrix}$$

$U^T\,A\,V$

liefert eine Diagonalmatrix mit den Singulärwerten s_i als Diagonalelemente. Durch Rundungsfehler sind aber die Nebendiagonalen nicht exakt Null.

```
> S:=Transpose(U).A.V;
```

$$S := \begin{bmatrix} 9.50803200069572085 & .177635683940025046\,10^{-14} \\ .444089209850062616\,10^{-15} & .772869635673484212 \\ -.857797617645175938\,10^{-16} & -.204806345233578718\,10^{-15} \end{bmatrix}$$

$U\,S\,V^T$

müßte bis auf Rundungsfehler wieder die Matrix A ergeben:

```
> U.S.Transpose(V);
```

$$\begin{bmatrix} .99999999999999966 & 3.99999999999999778 \\ 1.99999999999999912 & 4.99999999999999734 \\ 2.99999999999999866 & 5.99999999999999644 \end{bmatrix}$$

7.6 Spreadsheets

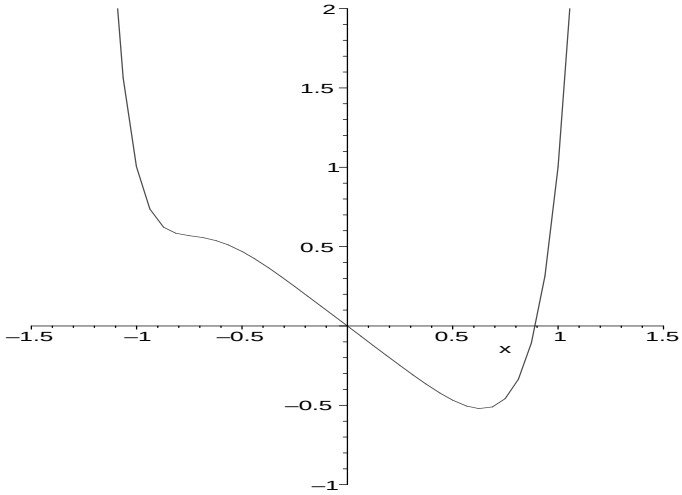
```
> restart:
```

Spreadsheets ermöglichen die übersichtliche Darstellung mathematischer Sachverhalte in Tabellenform. Im Unterschied zu 'normalen Tabellen' (z.B. in Excel) kann in den Zellen eines Maple-Spreadsheets auch symbolisch gerechnet werden. Ein Spreadsheet wird mit dem Menüpunkt *Insert/Spreadsheet* eingefügt.

Beispiel 1: Kurvendiskussion von ganzrationalen Funktionen auf engstem Raum und 'ohne Programmierung'. Man beachte aber die Befehle in den Zellen mit Bezügen zu anderen Zellen!

	Funktionen	*Nullstellen*	*Funktionswerte*	*nächste Ableitungen*
$\mathrm{f}(x)$	$x^5 - x + x^{10}$	$0,\ .89$	$0,\ -.02$	$-1.,\ 5.7$
$\frac{\partial}{\partial x}\,\mathrm{f}(x)$	$5\,x^4 - 1 + 10\,x^9$	$.64$	$-.52$	7.7
$\frac{\partial^2}{\partial x^2}\,\mathrm{f}(x)$	$20\,x^3 + 90\,x^8$	$-.74,\ 0,\ 0,\ 0$	$.57,\ 0,\ 0,\ 0$	$-53.,\ 0,\ 0,\ 0$
$\frac{\partial^3}{\partial x^3}\,\mathrm{f}(x)$	$60\,x^2 + 720\,x^7$			

Im Kontextmenü können mit *Properties...* die Zelleneigenschaften gesetzt werden (z.B. Berechnung mit 10 Dezimalstellen und Darstellung mit 2 Dezimalstellen). Durch Eingabe eines Funktionsterms in Zelle B2 und Auswertung des Spreadsheets (*Evaluate Spreadsheet* im Kontextmenü) erhält man die Funktionseigenschaften in tabellarischer Form. Eine Zeichnung darf bei Kurvendiskussionen nicht fehlen. Das geht nun ohne Befehle: Zelleninhalte können durch Cut&Paste in **smartplots** übernommen werden (hier der Inhalt von Zelle B2).



Beispiel 2: Integraltransformationen

Funktion	$expr \rightarrow \int \sin(x)\ expr\ dx$	*transformierte Funktion*
1	$\int \sin(x)\,dx$	$-\cos(x)$
x	$\int x\sin(x)\,dx$	$\sin(x) - \cos(x)\,x$
x^2	$\int x^2\sin(x)\,dx$	$-\cos(x)\,x^2 + 2\cos(x) + 2\,x\sin(x)$
$\sin(x)$	$\int \sin(x)^2\,dx$	$-\frac{1}{2}\cos(x)\sin(x) + \frac{1}{2}\,x$
$\cos(x)$	$\int \cos(x)\sin(x)\,dx$	$-\frac{1}{2}\cos(x)^2$
e^x	$\int e^x\sin(x)\,dx$	$-\frac{1}{2}\,e^x\cos(x) + \frac{1}{2}\,e^x\sin(x)$

Durch Ändern der Integraltransformation und Neuberechnung des Spreadsheets wird eine neue Integraltransformationstabelle erzeugt.

Tabellenteile eines Spreadsheets können außerhalb des Spreadsheets als Maple-Matrizen weiterverarbeitet werden; Markierung des Tabellenteils und Cut&Paste übertragen die entsprechende Matrix in die Maple-Eingabezeile.

Beispiel:

```
> MATRIX([[Int(sin(x)^2,x), -1/2*cos(x)*sin(x)+1/2*x],
[Int(cos(x)*sin(x),x), -1/2*cos(x)^2]]);
```

$$\begin{bmatrix} \int \sin(x)^2\,dx & -\frac{1}{2}\cos(x)\sin(x)+\frac{1}{2}x \\ \int \cos(x)\sin(x)\,dx & -\frac{1}{2}\cos(x)^2 \end{bmatrix}$$

Die Ableitung dieser Matrix ergibt sich nun in gewohnter Weise:

```
> map(diff,%,x);
```

$$\begin{bmatrix} \sin(x)^2 & \frac{1}{2}\sin(x)^2-\frac{1}{2}\cos(x)^2+\frac{1}{2} \\ \cos(x)\sin(x) & \cos(x)\sin(x) \end{bmatrix}$$

Kapitel 8

Numerische Verfahren

8.1 Newton-Iteration

```
> restart:
```

Die Kenntnis - und damit die Bestimmung - der Konvergenz- bzw. Fehlerordnung von Approximationsverfahren hat in der Numerik eine große Bedeutung. Dies gilt insbesondere, wenn ein Verfahren praktisch eingesetzt werden soll und der Rechenaufwand bei geforderter Rechengenauigkeit abzuschätzen ist.

Definition: Die Iterationsfolge $\{x_k\}$ konvergiert mit der Ordnung p gegen den Grenzwert ξ, wenn für die Folge der Fehler $\{e_k\}$, $e_k = |x_k - \xi|$, die Beziehung

```
> Limit(e[k+1]/e[k]^p, k=infinity) = M;
```

$$\lim_{k\to\infty} \frac{e_{k+1}}{{e_k}^p} = M$$

gilt; $0 < M$ und $1 \leq p$ sind positive Konstanten. $x_{k+1} = \phi(x_k)$ heißt Iterationsverfahren der Ordnung p. Ist $p = 1$, dann spricht man von linearer Konvergenz, bei $p = 2$ von quadratischer Konvergenz.

Als eines der ersten iterativen Verfahren $x_{k+1} = \mathrm{F}(x_k)$ lernen Studenten in der Numerik die Newton-Iteration zur Lösung nichtlinearer Gleichungen $\mathrm{f}(x) = 0$ kennen. Sei x_k eine Näherung der Nullstelle ξ, dann erhält man durch Anwendung der Iterationsfunktion

```
> F:=x -> x - f(x)/D(f)(x);
```

$$F := x \to x - \frac{\mathrm{f}(x)}{\mathrm{D}(f)(x)}$$

eine bessere Näherung x_{k+1}. Diese Iteration konvergiert quadratisch gegen die Nullstelle ξ von $\mathrm{f}(x)$, wenn die erste Ableitung von $\mathrm{F}(x)$ an der Stelle $x = \xi$ verschwindet und die zweite Ableitung ungleich Null ist.

```
> dF:=D(F)(x);
```

$$dF := \frac{\mathrm{f}(x)\,\mathrm{D}^2(f)(x)}{\mathrm{D}(f)(x)^2}$$

```
> subs(f(xi)=0,D(F)(xi));
```

$$0$$

```
> subs(f(xi)=0,(D@@2)(F)(xi));
```

$$\frac{\mathrm{D}^2(f)(\xi)}{\mathrm{D}(f)(\xi)}$$

Unter der Annahme $\mathrm{D}(f)(\xi) \neq 0$ und $\mathrm{D}^2(f)(\xi) \neq 0$ ist die obige Aussage also richtig.

Diese Konvergenzeigenschaft kann nur für einfache Nullstellen gelten,

denn hat $\mathrm{f}(x)$ an der Stelle $x = \xi$ eine zweifache Nullstelle, so gilt:

```
> f:=x -> (x-xi)^2*g(x);
```

$$f := x \to (x - \xi)^2\, \mathrm{g}(x)$$

mit $g(\xi) = 0$ und

```
> D(f)(xi);
```

$$0$$

Entsprechendes gilt für n größer als 2.

Die quadratische Konvergenz läßt sich eindrucksvoll mit Hilfe der beliebig genauen Gleitkommaarithmetik in Maple demonstrieren. Als Beispiel berechnen wir die Quadratwurzel von 9, d.h. wird lösen die Gleichung $x^2 - 9 = 0$ mit der Newton-Iteration. Wählen wir als Startwert $x_0 = 1$, so erhalten wir mit jeder weiteren Iteration eine Verdoppelung der korrekten Dezimalstellen:

```
> f:=x -> x^2 - 9:
> Digits := 70:
> xk := 1.0: printlevel:=0:
> to 8 do
> xk := F(xk):
> print(xk);
> end do;
```

$$5.0000000000000000000000000000000000000000000000000000000000000000000000$$
$$3.4000000000000000000000000000000000000000000000000000000000000000000000$$
$$3.0235294117647058823529411764705882352941176470588235294117647058823529$$
$$3.0000915541313801785305561913481345845731288624399176012817578393224 99$$
$$3.0000000013969838622484784252588819771210853888469481349100703687663 35$$
$$3.0000000000000000032526065174565133021986825552331682604822127820501 3$$
$$3.0000000000000000000000000000000000017632415262334312619531048058334$$
$$3.0000000000000000000000000000000000000000000000000000000000000000000000$$

Wir diskutieren nun das Verhalten der Newton-Iteration für Gleichungen mit mehrfachen Nullstellen. Hat $f(x)$ an der Stelle $x = \xi$ eine Nullstelle der Vielfachheit n, dann läßt sich f schreiben als

```
> f:=x -> (x-xi)^n*g(x);
```

$$f := x \to (x-\xi)^n \, g(x)$$

mit $g(\xi) \neq 0$. Für die erste Ableitung der Iterationsfunktion $F(x)$

```
> dF;
```

$$\frac{(x-\xi)^n \, g(x) \, (\frac{(x-\xi)^n \, n^2 \, g(x)}{(x-\xi)^2} - \frac{(x-\xi)^n \, n \, g(x)}{(x-\xi)^2} + \frac{2 \, (x-\xi)^n \, n \, D(g)(x)}{x-\xi} + (x-\xi)^n \, (D^{(2)})(g)(x))}{(\frac{(x-\xi)^n \, n \, g(x)}{x-\xi} + (x-\xi)^n \, D(g)(x))^2}$$

erhalten wir an der Stelle $x = \xi$ den Wert

```
> limit(%, x=xi);
```

$$\frac{n-1}{n}$$

Die erste Ableitung von $F(x)$ ist von Null verschieden, d. h. die Newton-Iteration konvergiert linear mit dem Konvergenzfaktor $\frac{n-1}{n}$.

Die Newton-Iteration hat eine einfache geometrische Interpretation: Ausgehend von einer Näherung x_k, ergibt sich die nächste Näherung durch den Schnittpunkt $[x_{k+1}, 0]$ der Tangente an f im Punkte $[x_k, f(x_k)]$ mit der x-Achse. Diese Eigenschaft werden wir nun mit Maple beweisen. Sei $\rho(x) = a\,x + b$ die Gleichung für die Tangente. $\rho(x)$ erfüllt als Tangente die Bedingung $\rho(x_k) = f(x_k)$ und hat an der Stelle $x = x_k$ die gleiche Ableitung wie $f(x)$. Mit diesen beiden Bedingungen sind die Parameter a und b der Tangente $\rho(x)$ festgelegt:

```
> f:='f':
> rho:=x -> a*x+b;
```

$$\rho := x \to a\,x + b$$

```
> solve({rho(x[k])=f(x[k]), D(rho)(x[k])=D(f)(x[k])},{a,b});
```

$$\{a = \mathrm{D}(f)(x_k),\, b = -\mathrm{D}(f)(x_k)\, x_k + \mathrm{f}(x_k)\}$$

Den Wert der Näherung x_{k+1} erhalten wir nun als Lösung der Tangentengleichung $\rho(x) = 0$:

```
> assign(%);
> x[k+1] = expand(solve(rho(t)=0,t));
```

$$x_{k+1} = x_k - \frac{\mathrm{f}(x_k)}{\mathrm{D}(f)(x_k)}$$

Die Approximation x_{k+1} berechnet sich also mit der Formel der Newton-Iteration, d.h. die geometrische Interpretation ist korrekt.

8.2 Sekanten-Methode

```
> restart:
```

Die Sekanten-Methode ist ein zweistufiges Iterationsverfahren. Verglichen mit der Newton-Iteration hat sie den Vorteil, daß sie keine Ableitungen benötigt; die Ableitung in der Newton-Formel wird durch eine endliche Differenz approximiert. Wir werden nun mit Hilfe von Maple die Konvergenzordnung der Sekanten-Methode ermitteln. Die Iterationsfunktion ist gegeben durch:

```
> F:=(u,v) -> u-f(u)*(u-v)/(f(u)-f(v));
```

$$F := (u,\, v) \to u - \frac{\mathrm{f}(u)\,(u - v)}{\mathrm{f}(u) - \mathrm{f}(v)}$$

```
> x[k+1] = F(x[k],x[k-1]);
```

$$x_{k+1} = x_k - \frac{\mathrm{f}(x_k)\,(x_k - x_{k-1})}{\mathrm{f}(x_k) - \mathrm{f}(x_{k-1})}$$

Für den Fehler $e_k = x_k - \xi$ haben wir die Rekursionbeziehung

```
> e[k+1] = 'F(xi+e[k],xi+e[k-1])-s';
```

$$e_{k+1} = \mathrm{F}(\xi + e_k,\, \xi + e_{k-1}) - s$$

Die rechte Seite dieser Gleichung läßt sich in eine zweidimensionale Taylorreihe (**mtaylor**) um die Punkte $e_k = 0$ und $e_{k-1} = 0$ entwickeln. Sei ξ eine einfache Nullstelle mit $\mathrm{D}(f)(\xi) \neq 0$ und $\mathrm{D}(\mathrm{D}(f))(\xi) \neq 0$. Wir setzen $\mathrm{f}(\xi) = 0$ und berechnen den ersten von Null verschiedenen Term in der Taylorreihenentwicklung:

```
> f(xi):=0:
> e[k+1] = normal(mtaylor(F(xi+e[k],xi+e[k-1])-xi,[e[k],e[k-1]],2));
```

$$e_{k+1} = 0$$

```
> e[k+1] = normal(mtaylor(F(xi+e[k],xi+e[k-1])-xi,[e[k],e[k-1]],3));
```

$$e_{k+1} = 0$$

```
> e[k+1] = normal(mtaylor(F(xi+e[k],xi+e[k-1])-xi,[e[k],e[k-1]],4));
```

$$e_{k+1} = \frac{1}{2}\,\frac{e_{k-1}\,(\mathrm{D}^{(2)})(f)(\xi)\,e_k}{\mathrm{D}(f)(\xi)}$$

Dividieren wir diesen führenden Koeffizienten in der Taylorreihe durch e_{k-1} und e_k, so bleibt auf der rechten Seite eine von Null verschiedene Konstante stehen

```
> %/e[k]/e[k-1];
```

$$\frac{e_{k+1}}{e_k\, e_{k-1}} = \frac{1}{2}\,\frac{(\mathrm{D}^{(2)})(f)(\xi)}{\mathrm{D}(f)(\xi)}$$

Für die Fehler gilt nach der Definition der Konvergenzordnung näherungsweise $e_{k+1} = M\, {e_k}^p$ und $e_k = M\, {e_{k-1}}^p$, wenn p die Ordnung und M der Faktor der Konvergenz ist. Substituieren wir diese Beziehungen in die letzte Gleichung und dividieren noch durch M^p, so erhalten wir

```
> simplify(subs(e[k+1]=M*e[k]^p,e[k]=M*e[k-1]^p, %/M^p),symbolic);
```

$${e_{k-1}}^{(-p-1+p^2)} = \frac{1}{2}\,\frac{M^{(-p)}\,(\mathrm{D}^{(2)})(f)(\xi)}{\mathrm{D}(f)(\xi)}$$

Diese Gleichung ist gültig für alle e_{k-1}. Da die rechte Seite konstant ist, muß die linke Seite von e_{k-1} unabhängig sein. Dies ist nur der Fall, wenn der Exponent von e_{k-1} gleich Null ist. Die Lösung der erhaltenen Bestimmungsgleichung für p

```
> op(2,lhs(%))=0;
```

$$-p-1+p^2=0$$

liefert nun die bekannte Konvergenzordnung der Sekanten-Methode:

```
> solve(%,p);
```

$$\frac{1}{2}\sqrt{5}+\frac{1}{2},\ \frac{1}{2}-\frac{1}{2}\sqrt{5}$$

```
> p = %[1];
```

$$p=\frac{1}{2}\sqrt{5}+\frac{1}{2}$$

```
> evalf(%);
```

$$p=1.618033989$$

8.3 Numerische Differentiation

```
> restart:
```

Um die Differentiation einer Funktion f an der Stelle x_0 auf einem Rechner numerisch berechnen zu können, geht man auf die Definition der Ableitung über den Differentialquotient zurück:

```
> D(f)(x[0])=Limit((f(x[0]+h)-f(x[0]))/h,h=0);
```

$$\mathrm{D}(f)(x_0)=\lim_{h\to 0}\frac{\mathrm{f}(x_0+h)-\mathrm{f}(x_0)}{h}$$

Die Ableitung der Funktion f an der Stelle x_0 wird für $0<h$ durch die Sekantensteigung

```
> Dfr:=(f(x[0]+h)-f(x[0]))/h;
```

$$Dfr:=\frac{\mathrm{f}(x_0+h)-\mathrm{f}(x_0)}{h}$$

numerisch angenähert. Dies ist die sogenannte einseitige (rechtsseitige) Differenzenformel. Diese einseitige Differenzenformel hat die Eigenschaft, daß Polynome vom Grade $n=1$ exakt differenziert werden, denn es gilt:

```
> f:=x -> m*x+b;
```

$$f:=x\to m\,x+b$$

```
> D(f)(x[0])-Dfr;
```

$$m-\frac{m\,(x_0+h)-m\,x_0}{h}$$

```
> simplify(%);
```

$$0$$

Eine genauere Differenzenformel erhält man, wenn man den Mittelwert der rechtsseitigen und linksseitigen Differenzenformel nimmt:

```
> f:='f':
> Dfz:=(f(x[0]+h)-f(x[0]-h))/(2*h);
```

$$Dfz:=\frac{1}{2}\,\frac{\mathrm{f}(x_0+h)-\mathrm{f}(x_0-h)}{h}$$

Mit dieser zentralen Differenzenformel werden Polynome bis zum Grad 2 exakt differenziert, denn es gilt:

```
> f:=x -> a+b*x+c*x^2;
```

$$f:=x\to a+b\,x+c\,x^2$$

```
> simplify(D(f)(x[0])-Dfz);
```

$$0$$

Die obigen Differenzenformeln lassen sich allgemein nach folgendem Prinzip gewinnen: Wir bestimmen zunächst ein Interpolationspolynom durch die Punkte $[x_0,\, \mathrm{f}(x_0)],\, [x_0+h,\, \mathrm{f}(x_0+h)]$ bzw. $[x_0-h,\, \mathrm{f}(x_0-h)],\, [x_0+h,\, \mathrm{f}(x_0+h)]$, leiten anschließend dieses Polynom ab und werten an der gesuchten Zwischenstelle aus. Für die Polynominterpolation verwenden wir die Maple-Funktion **interp**.

```
> f:='f':
> interp([x[0],x[0]+h],[f(x[0]),f(x[0]+h)],z);
```

$$\frac{(\mathrm{f}(x_0+h)-\mathrm{f}(x_0))\,z}{h}+\frac{\mathrm{f}(x_0)\,h-\mathrm{f}(x_0+h)\,x_0+\mathrm{f}(x_0)\,x_0}{h}$$

```
> subs(z=x[0],diff(%,z));
```

$$\frac{\mathrm{f}(x_0+h)-\mathrm{f}(x_0)}{h}$$

```
> subs(z=x[0],diff(interp([x[0]-h,x[0]+h],[f(x[0]-h),f(x[0]+h)],z),z));
```

$$\frac{1}{2}\,\frac{\mathrm{f}(x_0+h)-\mathrm{f}(x_0-h)}{h}$$

Die Fehlerordnung dieser Differenzenformeln erhalten wir, indem wir die entsprechenden Fehler in eine Reihe entwickeln:

```
> series(D(f)(x[0])-Dfr,h,3);
```

$$-\frac{1}{2}\,(\mathrm{D}^{(2)})(f)(x_0)\,h+\mathrm{O}(h^2)$$

```
> series(D(f)(x[0])-Dfz,h,4);
```

$$-\frac{1}{6}\,(\mathrm{D}^{(3)})(f)(x_0)\,h^2+\mathrm{O}(h^3)$$

Der Fehler der einseitigen Differenzenformel ist also proportional zu h, während der Fehler der zentralen Differenzenformel proportional zu h^2 ist. Differenzenformeln höherer Ordnung für die erste Ableitung erhält man, wenn im obigen Verfahren weitere Interpolationspunkte hinzugenommen werden. Beispiel:

```
> Df3:=normal(subs(z=x[0],diff(interp([seq(x[0]+i*h,i=0..3)],
> [seq(f(x[0]+i*h),i=0..3)],z),z)));
```

$$Df3 := -\frac{1}{6}\,\frac{-2\,\mathrm{f}(x_0+3\,h)+9\,\mathrm{f}(x_0+2\,h)-18\,\mathrm{f}(x_0+h)+11\,\mathrm{f}(x_0)}{h}$$

```
> series(D(f)(x[0])-%,h,5);
```

$$-\frac{1}{4}\,(\mathrm{D}^{(4)})(f)(x_0)\,h^3+\mathrm{O}(h^4)$$

Diese Differenzenformel hat die Fehlerordnung 3, d.h. Polynome bis zum Grad 3 werden exakt differenziert:

```
> f:=x -> a+b*x+c*x^2+d*x^3;
```

$$f := x \to a+b\,x+c\,x^2+d\,x^3$$

```
> simplify(Df3-D(f)(x[0]));
```

$$0$$

Differenzenformeln (Diskretisierungsformeln) für die zweite und höhere (n-te) Ableitungen sowie bei nichtäquidistanter Unterteilung erhält man, indem durch vorgegebene Punkte $[s_0,\, \mathrm{f}(s_0)],\, [s_1,\, \mathrm{f}(s_1)]$, ..., $[s_k,\, \mathrm{f}(s_k)]$ das Interpolationspolynom gelegt, dieses n-mal differenziert und anschließend die auszuwertende Stelle eingesetzt wird. Beispiel:

```
> f:='f':
> normal(subs(z=s[0],diff(interp([seq(s[i],i=0..2)],
> [seq(f(s[i]),i=0..2)],z),z$2)));
```

$$-2\,\frac{\mathrm{f}(s_2)\,s_1-\mathrm{f}(s_2)\,s_0-\mathrm{f}(s_1)\,s_2+\mathrm{f}(s_1)\,s_0+\mathrm{f}(s_0)\,s_2-\mathrm{f}(s_0)\,s_1}{(-s_1+s_0)\,(-s_2+s_0)\,(-s_2+s_1)}$$

Interpolieren wir nun an den äquidistanten Stellen $[x_0-h,\, \mathrm{f}(x_0-h)],\, [x_0,\, \mathrm{f}(x_0)]$ und $[x_0+h,\, \mathrm{f}(x_0+h)]$, so erhalten wir die zentrale Differenzenformel für die zweite Ableitung:

```
> simplify(subs(seq(s[i]=x[0]+i*h-h,i=0..2),%));
```

$$\frac{\mathrm{f}(x_0+h)-2\,\mathrm{f}(x_0)+\mathrm{f}(x_0-h)}{h^2}$$

```
> series(D(D(f))(x[0])-%,h,5);
```

$$-\frac{1}{12}\,(\mathrm{D}^{(4)})(f)(x_0)\,h^2+\mathrm{O}(h^3)$$

Der Fehler für diese Differenzenformel ist wieder proportional zu h^2. Wir fassen die besprochenen Maple-Anweisungen zur Erzeugung einer Differenzenformel für die n-te Ableitung zu einer Prozedur zusammen.

```
> DiffFormeln:=proc(t::list,s::list,n::integer,k::integer)
> # t: Liste der x-Werte
> # s: Liste der y-Werte (Funktionswerte)
> # n: Ordnung der Ableitung
> # k: Stelle, an der die Differenzenformel erstellt werden soll.
> local x;
> interp(t,s,x);
> diff(%,x$n);
> normal(subs(x=t[k],%));
> end proc;
```

$$\begin{aligned}&\mathit{DiffFormeln} := \mathbf{proc}(t{::}\mathit{list},\ s{::}\mathit{list},\ n{::}\mathit{integer},\ k{::}\mathit{integer})\\ &\mathbf{local}\ x;\\ &\quad \mathrm{interp}(t,\ s,\ x)\,;\ \mathrm{diff}(\%,\ x\,\$\,n)\,;\ \mathrm{normal}(\mathrm{subs}(x=t_k,\ \%))\\ &\mathbf{end\ proc}\end{aligned}$$

Wir generieren einige Differenzenformeln mit dieser Prozedur, zunächst *Df3* :

```
> DiffFormeln([seq(x[0]+i*h,i=0..3)],
> [seq(f(x[0]+i*h),i=0..3)],1,1);
```

$$\frac{1}{6}\,\frac{-9\,\mathrm{f}(x_0+2\,h)-11\,\mathrm{f}(x_0)+18\,\mathrm{f}(x_0+h)+2\,\mathrm{f}(x_0+3\,h)}{h}$$

Die entsprechende Formel für die zweite Ableitung hat die Gestalt

```
> DiffFormeln([seq(x[0]+i*h,i=0..3)],
> [seq(f(x[0]+i*h),i=0..3)],2,1);
```

$$-\frac{-4\,\mathrm{f}(x_0+2\,h)+5\,\mathrm{f}(x_0+h)-2\,\mathrm{f}(x_0)+\mathrm{f}(x_0+3\,h)}{h^2}$$

8.4 Newton-Cotes-Quadratur

```
> restart:
```

Die Quadraturformeln von Newton-Cotes für die numerische Berechnung bestimmter Integrale $\int_a^b \mathrm{f}(x)\,dx$ lassen sich mit Maple leicht erzeugen. Wir leiten zunächst die bekannte Simpson-Regel her. Die Formel erhält man durch Interpolation der Funktionswerte an den Stellen a, $\frac{a+b}{2}$ und b und Integration des erhaltenen Interpolationspolynoms zweiten Grades. Die Koeffizienten des Interpolationspolynoms

```
> p:=x -> c[0]+c[1]*x+c[2]*x^2;
```

$$p := x \to c_0 + c_1\,x + c_2\,x^2$$

bestimmen wir mit Hilfe der linearen Gleichungen:

```
> eqns:={p(a)=f(a),p((a+b)/2)=f((a+b)/2),p(b)=f(b)};
```

$$\begin{aligned}\mathit{eqns} := \{&c_0+c_1\,a+c_2\,a^2=\mathrm{f}(a),\ c_0+c_1\,(\frac{1}{2}\,a+\frac{1}{2}\,b)+c_2\,(\frac{1}{2}\,a+\frac{1}{2}\,b)^2=\mathrm{f}(\frac{1}{2}\,a+\frac{1}{2}\,b),\\ &c_0+c_1\,b+c_2\,b^2=\mathrm{f}(b)\}\end{aligned}$$

```
> solve(eqns,{c[0],c[1],c[2]}):
> collect(factor(%),[f(a),f(b),f((a+b)/2)]);
```

$$\left\{ c_0 = -\frac{(-a\,b-b^2)\,\mathrm{f}(a)}{(a-b)^2} - \frac{(-a^2-a\,b)\,\mathrm{f}(b)}{(a-b)^2} - \frac{4\,a\,b\,\mathrm{f}(\frac{1}{2}\,a+\frac{1}{2}\,b)}{(a-b)^2},\right.$$

$$c_2 = 2\,\frac{\mathrm{f}(a)}{(a-b)^2} + \frac{2\,\mathrm{f}(b)}{(a-b)^2} - \frac{4\,\mathrm{f}(\frac{1}{2}\,a+\frac{1}{2}\,b)}{(a-b)^2},$$

$$\left. c_1 = \frac{(-a-3\,b)\,\mathrm{f}(a)}{(a-b)^2} + \frac{(-3\,a-b)\,\mathrm{f}(b)}{(a-b)^2} + \frac{(4\,a+4\,b)\,\mathrm{f}(\frac{1}{2}\,a+\frac{1}{2}\,b)}{(a-b)^2} \right\}$$

```
> assign(%);
> factor(int(p(x),x=a..b));
```

$$-\frac{1}{6}\,(a-b)\,(\mathrm{f}(b)+4\,\mathrm{f}(\frac{1}{2}\,a+\frac{1}{2}\,b)+\mathrm{f}(a))$$

Der Quadraturfehler der Simpson-Regel läßt sich durch eine Reihenentwicklung ermitteln. Sei $h = \frac{b-a}{2}$, dann gilt

```
> QF2:=subs(b=a+2*h,int(f(x),x=a..b) - %);
```

$$QF2 := \int_a^{a+2\,h} \mathrm{f}(x)\,dx - \frac{1}{3}\,h\,(\mathrm{f}(a+2\,h)+4\,\mathrm{f}(a+h)+\mathrm{f}(a))$$

```
> series(QF2,h);
```

$$-\frac{1}{90}\,(\mathrm{D}^{(4)})(f)(a)\,h^5 + \mathrm{O}(h^6)$$

Der Quadraturfehler der Simpson-Regel ist also proportional zu h^5.

Wir verkürzen nun die Herleitung der Simpson-Regel, indem wir die Maple-Funktion **interp** zur Polynominterpolation verwenden. Wählen wir als Interpolationspunkte $[a,\,\mathrm{f}(a)],\,[a+h,\,\mathrm{f}(a+h)]$ und $[a+2\,h,\,\mathrm{f}(a+2\,h)]$, so liefert die analytische Integration wieder die obige Darstellung der Simpson-Regel:

```
> factor((b-a)/(2*h)*int(interp([a,a+h,a+2*h],
> [f(a),f(a+h),f(a+2*h)],z),z=a..a+2*h));
```

$$-\frac{1}{6}\,(a-b)\,(\mathrm{f}(a+2\,h)+4\,\mathrm{f}(a+h)+\mathrm{f}(a))$$

Allgemeiner lassen sich für das bestimmte Integral $\int_a^b \mathrm{f}(x)\,dx$ Newton-Cotes-Formeln herleiten, indem wir $n+1$ Funktionswerte mit äquidistanten Stützstellen interpolieren und das erhaltene Interpolationspolynom n-ten Grades integrieren. Die folgende Funktion NC erzeugt eine normalisierte ($n+1$)-Punkt Newton-Cotes-Formel:

```
> NC:=n -> factor((b-a)/(n*h)*int(interp([seq(a+i*h,i=0..n)],
> [seq(f(a+i*h),i=0..n)],z),z=a..a+n*h));
```

$$NC := n \to \mathrm{factor}\left(\frac{(b-a)}{n\,h}\int_a^{a+n\,h} \mathrm{interp}([\mathrm{seq}(a+i\,h,\,i=0..n)],\,[\mathrm{seq}(\mathrm{f}(a+i\,h),\,i=0..n)],\,z)\,dz\right)$$

Wir generieren nun die ersten acht Newton-Cotes-Formeln und die zugehörigen Quadraturfehler, ausgedrückt in einer Potenz der Schrittweite h und einer Ableitung an der Stelle $x=a$. Zu beachten ist, daß die Potenz von h und der Grad der Ableitung bei den geraden Ordnungen jeweils um 2 springen. Die Funktion **modp** wird für die Bestimmung einer geeigneten Ordnung bei der durchzuführenden Reihenentwicklung benötigt.

```
> for i from 1 to 8 do
> NC[i]=NC(i);
> QF[i]=series(subs(b=a+i*h,int(f(x),x=a..b)-rhs(%)),
> h,i+3+modp(i+3,2));
> end do;
```

$$NC_1 = -\frac{1}{2}\,(a-b)\,(\mathrm{f}(a+h)+\mathrm{f}(a))$$

$$QF_1 = -\frac{1}{12}\,(\mathrm{D}^{(2)})(f)(a)\,h^3+\mathrm{O}(h^4)$$

$$NC_2 = -\frac{1}{6}\,(a-b)\,(\mathrm{f}(a+2\,h)+4\,\mathrm{f}(a+h)+\mathrm{f}(a))$$

$$QF_2 = -\frac{1}{90}\,(\mathrm{D}^{(4)})(f)(a)\,h^5+\mathrm{O}(h^6)$$

$$NC_3 = -\frac{1}{8}\,(a-b)\,(3\,\mathrm{f}(a+2\,h)+3\,\mathrm{f}(a+h)+\mathrm{f}(a+3\,h)+\mathrm{f}(a))$$

$$QF_3 = -\frac{3}{80}\,(\mathrm{D}^{(4)})(f)(a)\,h^5+\mathrm{O}(h^6)$$

$$NC_4 = -\frac{1}{90}\,(a-b)\,(7\,\mathrm{f}(a)+7\,\mathrm{f}(a+4\,h)+32\,\mathrm{f}(a+3\,h)+12\,\mathrm{f}(a+2\,h)+32\,\mathrm{f}(a+h))$$

$$QF_4 = -\frac{8}{945}\,(\mathrm{D}^{(6)})(f)(a)\,h^7+\mathrm{O}(h^8)$$

$$NC_5 = -\frac{1}{288}(a-b)(\\ 19\,\mathrm{f}(a)+50\,\mathrm{f}(a+2\,h)+19\,\mathrm{f}(a+5\,h)+75\,\mathrm{f}(a+4\,h)+50\,\mathrm{f}(a+3\,h)+75\,\mathrm{f}(a+h))$$

$$QF_5 = -\frac{275}{12096}\,(\mathrm{D}^{(6)})(f)(a)\,h^7+\mathrm{O}(h^8)$$

$$NC_6 = -\frac{1}{840}(a-b)(216\,\mathrm{f}(a+5\,h)+41\,\mathrm{f}(a)+41\,\mathrm{f}(a+6\,h)+27\,\mathrm{f}(a+4\,h)+27\,\mathrm{f}(a+2\,h)\\ +272\,\mathrm{f}(a+3\,h)+216\,\mathrm{f}(a+h))$$

$$QF_6 = -\frac{9}{1400}\,(\mathrm{D}^{(8)})(f)(a)\,h^9+\mathrm{O}(h^{10})$$

$$NC_7 = -\frac{1}{17280}(a-b)(751\,\mathrm{f}(a)+751\,\mathrm{f}(a+7\,h)+3577\,\mathrm{f}(a+6\,h)+1323\,\mathrm{f}(a+5\,h)\\ +2989\,\mathrm{f}(a+3\,h)+3577\,\mathrm{f}(a+h)+2989\,\mathrm{f}(a+4\,h)+1323\,\mathrm{f}(a+2\,h))$$

$$QF_7 = -\frac{8183}{518400}\,(\mathrm{D}^{(8)})(f)(a)\,h^9+\mathrm{O}(h^{10})$$

$$NC_8 = -\frac{1}{28350}(a-b)(5888\,\mathrm{f}(a+h)-928\,\mathrm{f}(a+2\,h)+10496\,\mathrm{f}(a+5\,h)\\ +10496\,\mathrm{f}(a+3\,h)+989\,\mathrm{f}(a+8\,h)+5888\,\mathrm{f}(a+7\,h)-928\,\mathrm{f}(a+6\,h)\\ -4540\,\mathrm{f}(a+4\,h)+989\,\mathrm{f}(a))$$

$$QF_8 = -\frac{2368}{467775}\,(\mathrm{D}^{(10)})(f)(a)\,h^{11}+\mathrm{O}(h^{12})$$

Einige dieser Newton-Cotes-Formeln haben spezielle Bezeichnungen oder tragen bekannte Namen: Trapezregel ($n=1$), Simpson-Regel, Keplersche Faßregel ($n=2$), Newtonsche 3/8-Regel ($n=3$), Milne-Regel ($n=4$), Weddle-Regel ($n=6$). Für die Ordnungen $n=1$,..., 7 sind die Integrationsgewichte w_i und damit auch die Quadraturformel stets positiv, ab $n=8$ treten erstmals negative Gewichte auf. Die Summe der Integrationsgewichte in den Newton-Cotes-Formeln ist 1, d. h. es gilt $\sum_{i=1}^{n} w_i = 1$. Für die 9-Punkt-Formel ($n=8$) , die in der MATLAB-Funktion quad8 benutzt wird, überprüfen wir diese Beziehung:

```
> normal(NC(8)/(b-a));
```

$$\frac{989}{28350}\,\mathrm{f}(a)+\frac{5248}{14175}\,\mathrm{f}(a+5\,h)+\frac{5248}{14175}\,\mathrm{f}(a+3\,h)+\frac{2944}{14175}\,\mathrm{f}(a+h)-\frac{464}{14175}\,\mathrm{f}(a+2\,h)\\ +\frac{989}{28350}\,\mathrm{f}(a+8\,h)+\frac{2944}{14175}\,\mathrm{f}(a+7\,h)-\frac{464}{14175}\,\mathrm{f}(a+6\,h)-\frac{454}{2835}\,\mathrm{f}(a+4\,h)$$

```
> map(x->op(1,x),%);
```

$$1$$

Die ($n+1$)-Punkt Newton-Cotes-Formeln haben die Eigenschaft, daß sie Polynome bis zum Grad n exakt integrieren. Wir wollen diese Aussage wieder an der 9-Punkt-Formel überprüfen. Ist f z. B. ein Polynom achten Grades

```
> f:=x -> add(d[i]*x^i,i=0..8):
> f(x);
```

$$d_0 + d_1\,x + d_2\,x^2 + d_3\,x^3 + d_4\,x^4 + d_5\,x^5 + d_6\,x^6 + d_7\,x^7 + d_8\,x^8$$

dann gilt

```
> simplify(subs(h=(b-a)/8,NC(8))-int(f(x),x=a..b));
```

$$0$$

8.5 Gauß-Legendre-Quadratur

```
> restart:
```

Bei der Konstruktion der Newton-Cotes-Formeln haben wir ausgehend von n vorgegebenen Integrationsknoten x_i die Gewichte w_i so bestimmt, daß die Quadraturformel Polynome bis zum Grad $n-1$ exakt integriert. Könnten wir vielleicht mehr erreichen, wenn wir auch die Knoten zur Disposition stellen? Genauer: Wir suchen zu einem vorgegebenen n die Knoten x_0**,...,** x_n und die Gewichte w_0**,...,** w_n , so daß Polynome bis zu einem möglichst hohen Grad durch die Quadraturformel $\sum_{i=1}^{n} w_i\,\mathrm{f}(x_i)$ exakt integriert werden. Für $n=3$ haben wir die sechs Unbekannten $x_1,\ x_2,\ x_3,\ w_1,\ w_2$ und w_3 zu bestimmen. Wenn wir für die Monome x^j**,** $j=0..5$ exakte Integration auf dem Intervall [-1..1] fordern, d. h. $w_1\,{x_1}^j + w_2\,{x_2}^j + w_3\,{x_3}^j = \int_{-1}^{1} x^j\,dx$, dann erhalten wir sechs nichtlineare Bestimmungsgleichungen:

```
> eqns:={seq(add(w[i]*x[i]^k,i=1..3)=int(x^k,x=-1..1),k=0..5)};
```

$$eqns := \{w_1 + w_2 + w_3 = 2,\ w_1\,x_1 + w_2\,x_2 + w_3\,x_3 = 0,\ w_1\,{x_1}^2 + w_2\,{x_2}^2 + w_3\,{x_3}^2 = \frac{2}{3},$$

$$w_1\,{x_1}^3 + w_2\,{x_2}^3 + w_3\,{x_3}^3 = 0,\ w_1\,{x_1}^4 + w_2\,{x_2}^4 + w_3\,{x_3}^4 = \frac{2}{5},\ w_1\,{x_1}^5 + w_2\,{x_2}^5 + w_3\,{x_3}^5 = 0\}$$

Wir können dieses System mit Maple lösen. Wir ermitteln zunächst die Menge der Unbekannten mit **indets**:

```
> id:=indets(eqns,name);
```

$$id := \{w_1,\ x_1,\ w_2,\ x_2,\ w_3,\ x_3\}$$

Die exakte Lösung bestimmen wir nun mit solve:

```
> _MaxSols:=1:
> sol:=solve(eqns,id);
```

$$sol := \{x_3 = \mathrm{RootOf}(-3 + 5\_Z^2),\ w_1 = \frac{8}{9},\ x_2 = -\mathrm{RootOf}(-3 + 5\_Z^2),\ w_2 = \frac{5}{9},\ x_1 = 0,$$

$$w_3 = \frac{5}{9}\}$$

```
> sol_sym:=combine(convert(sol,radical));
```

$$sol\_sym := \{w_1 = \frac{8}{9},\ w_2 = \frac{5}{9},\ x_2 = -\frac{1}{5}\sqrt{15},\ x_1 = 0,\ x_3 = \frac{1}{5}\sqrt{15},\ w_3 = \frac{5}{9}\}$$

Damit ergibt sich die spezielle Quadraturformel:

```
> GL3:=add(w[i]*f(x[i]),i=1..3);
```

$$GL3 := w_1\,\mathrm{f}(x_1) + w_2\,\mathrm{f}(x_2) + w_3\,\mathrm{f}(x_3)$$

```
> GL3_sym:=subs(sol_sym,GL3);
```

$$GL3\_sym := \frac{8}{9}\,\mathrm{f}(0) + \frac{5}{9}\,\mathrm{f}(-\frac{1}{5}\sqrt{15}) + \frac{5}{9}\,\mathrm{f}(\frac{1}{5}\sqrt{15})$$

Wir wollen überprüfen, bis zu welchem Grad Polynome durch diese Quadraturformel exakt integriert werden:

```
> printlevel:=0:
> for k from 0 to 6 do
> f:=x -> add(b[i]*x^i,i=0..k);
> print(int(f(x),x=-1..1) - GL3_sym);
> end do;
```

$$0$$
$$0$$
$$0$$
$$0$$
$$0$$
$$0$$
$$\frac{8}{175}\,b_6$$

Die Quadraturformel $GL3\_sym$ ist also exakt für Polynome bis zum Grad 5. Ziel ist nun, allgemein n Knoten und n Gewichte zu finden, so daß die Quadraturformel Polynome P bis zum Grad $2\,n-1$ exakt integriert, d.h. $\int_{-1}^{1} \mathrm{P}(x)\,dx = \sum_{i=1}^{n} w_i\,\mathrm{P}(x_i)$. Die obige Methode führt für größere n nicht zum Ziel, da das System nichtlinearer Gleichungen zu komplex wird und von Maple symbolisch nicht mehr gelöst werden kann. Wir argumentieren nun folgendermaßen: Sei $P_{2\,n-1}(x)$ ein Polynom vom Grade $2\,n-1$. Division duch ein Polynom n-ten Grades $Q_n(x)$ liefert dann die folgende Zerlegung (Euklidischer Algorithmus):

```
> P[2*n-1](x)=H[n-1](x)*Q[n](x)+R[n-1](x);
```

$$P_{2\,n-1}(x) = H_{n-1}(x)\,Q_n(x) + R_{n-1}(x)$$

$H_{n-1}(x)$ und $R_{n-1}(x)$ sind Polynome $(n-1)$-ten Grades.

```
> Int(lhs(%),x=-1..1)=map(Int,rhs(%),x=-1..1);
```

$$\int_{-1}^{1} P_{2\,n-1}(x)\,dx = \int_{-1}^{1} H_{n-1}(x)\,Q_n(x)\,dx + \int_{-1}^{1} R_{n-1}(x)\,dx$$

Wenden wir die obige Quadraturformel auf beide Seiten dieser Gleichung an, so erhalten wir für den Fehler:

```
> Error = Int(H[n-1](x)*Q[n](x),x=-1..1) -
> Sum(w[i]*H[n-1](x[i])*Q[n](x[i]),i=1..n) +
> Int(R[n-1](x),x=-1..1) - Sum(w[i]*R[n-1](x[i]),i=1..n);
```

$$Error = \int_{-1}^{1} H_{n-1}(x)\,Q_n(x)\,dx - (\sum_{i=1}^{n} w_i\,H_{n-1}(x_i)\,Q_n(x_i)) + \int_{-1}^{1} R_{n-1}(x)\,dx$$
$$- (\sum_{i=1}^{n} w_i\,R_{n-1}(x_i))$$

Damit der Fehler verschwindet, machen wir dir folgenden Annahmen: $Q_n(x)$ seien Legendre-Polynome. Diese Polynome sind Orthogonalpolynome zum Skalarprodukt $(f,\,g) = \int_{-1}^{1} \mathrm{f}(x)\,\mathrm{g}(x)\,dx$, d.h. der erste Term verschwindet. Die Knoten x_i wählen wir als Nullstellen der Legendre-Polynome $Q_n(x)$. Dann verschwindet auch der zweite Term. Schließlich bestimmen wir die Gewichte w_i wie bei den Newton-Cotes-Formeln durch Integration des Interpolationspolynoms für $R_{n-1}(x)$, was natürlich wieder $R_{n-1}(x)$ ist wegen der Eindeutigkeit des Interpolationspolynoms. Also gilt $\int_{-1}^{1} R_{n-1}(x)\,dx = \sum_{i=1}^{n} w_i\,R_{n-1}(x_i)$ und damit verschwinden die letzten beiden Terme. Knoten und Gewichte einer Gauß-Quadraturformel, z. B. für $n=3$, berechnen sich also wie folgt:

```
> X:=[fsolve(orthopoly[P](3,x)=0,x)];
```

$$X := [-.7745966692,\ 0.,\ .7745966692]$$

Das Maple-Paket **orthopoly** enthält Funktionen zur Erzeugung von Orthogonalpolynomen.

```
> f:='f':
> interp(X,[seq(f(x[i]),i=1..3)],z);
```

$$(.8333333333\,\mathrm{f}(x_3) - 1.666666667\,\mathrm{f}(x_2) + .8333333333\,\mathrm{f}(x_1))\,z^2 + (-.6454972247\,\mathrm{f}(x_1) + .6454972243\,\mathrm{f}(x_3))\,z + 1.000000000\,\mathrm{f}(x_2)$$

```
> int(%,z=-1..1);
```

$$.5555555555\,\mathrm{f}(x_3) + .8888888887\,\mathrm{f}(x_2) + .5555555555\,\mathrm{f}(x_1)$$

Für $n = 3$ haben wir damit die numerische Quadraturformel:

```
> GL3_num:=subs(seq(x[i]=X[i],i=1..3), %);
```

$$GL3\_num := .5555555555\,\mathrm{f}(.7745966692) + .8888888887\,\mathrm{f}(0.) + .5555555555\,\mathrm{f}(-.7745966692)$$

Wir wollen noch überprüfen, ob die oben errechnete analytische Quadraturformel $GL3\_sym$ für $n = 3$ mit der numerischen Formel $GL3\_num$ übereinstimmt:

```
> sol_num:=evalf(sol_sym);
```

$$sol\_num := \{w_3 = .5555555556,\ x_1 = 0.,\ w_1 = .8888888889,\ w_2 = .5555555556,\ x_2 = -.7745966692,\ x_3 = .7745966692\}$$

```
> subs(sol_num,GL3);
```

$$.8888888889\,\mathrm{f}(0.) + .5555555556\,\mathrm{f}(-.7745966692) + .5555555556\,\mathrm{f}(.7745966692)$$

$GL3\_num$ und $GL3\_sym$ liefern also innerhalb der numerischen Genauigkeit die gleichen Resultate. Wir fassen die Maple-Anweisungen zur Erzeugung von Gauß-Quadraturformeln in eine Prozedur zusammen:

```
> GL:=proc(n)
> local X;
> X:=[fsolve(orthopoly[P](n,x)=0,x)];
> interp(X,[seq(f(x[i]),i=1..n)],z):
> int(%,z=-1..1):
> subs(seq(x[i]=X[i],i=1..n), %);
> end proc;
```

$GL := \mathbf{proc}(n)$
$\mathbf{local}\ X;$
$X := [\mathrm{fsolve}(orthopoly_P(n,\ x) = 0,\ x)]\,;$
$\mathrm{interp}(X,\ [\mathrm{seq}(\mathrm{f}(x_i),\ i = 1..n)],\ z)\,;$
$\mathrm{int}(\%,\ z = -1..1)\,;$
$\mathrm{subs}(\mathrm{seq}(x_i = X_i,\ i = 1..n),\ \%)$
$\mathbf{end\ proc}$

```
> GL(10);
```

$$.06667134809\,\mathrm{f}(.9739065285) + .1494513586\,\mathrm{f}(.8650633667) + .2692669079\,\mathrm{f}(-.4333953941) + .2955242314\,\mathrm{f}(-.1488743390) + .2955243152\,\mathrm{f}(.1488743390) + .2692667068\,\mathrm{f}(.4333953941) + .2190864152\,\mathrm{f}(.6794095683) + .06667136186\,\mathrm{f}(-.9739065285) + .1494513044\,\mathrm{f}(-.8650633667) + .2190865113\,\mathrm{f}(-.6794095683)$$

Das beschriebene numerische Quadraturverfahren heißt Gauß-Legendre-Quadratur.

Kapitel 9

Optimierung und Transformation von Programmen

9.1 Generierung von C- und Fortran-Code

```
> restart:
```

```
> with(codegen);
```

$[C,\ GRAD,\ GRADIENT,\ HESSIAN,\ JACOBIAN,\ cost,\ declare,\ dontreturn,\ eqn,\ fortran,$
$horner,\ intrep2maple,\ joinprocs,\ makeglobal,\ makeparam,\ makeproc,\ makevoid,$
$maple2intrep,\ optimize,\ packargs,\ packlocals,\ packparams,\ prep2trans,$
$renamevar,\ split,\ swapargs]$

Das Maple-Paket **codegen** stellt eine Reihe von Funktionen zur Erzeugung, Manipulation und Übersetzung (in andere Programmiersprachen) von Maple-Prozeduren zur Verfügung. Die folgende Doppelsumme soll in C bzw. Fortran berechnet werden:

```
> S:=Sum(y^j/j!*Sum(x^i/i!,i=0..n),j=0..m);
```

$$S := \sum_{j=0}^{m} \frac{y^j \left(\sum_{i=0}^{n} \frac{x^i}{i!} \right)}{j!}$$

Wir generieren zunächst eine Maple-Prozedur für diese Aufgabe (**codegen,makeproc**):

```
> F:=makeproc(S,parameters=[n,m,x,y]);
```

$F := \mathbf{proc}(n,\ m,\ x,\ y)\ \mathrm{Sum}(y^j * \mathrm{Sum}(x^i/i!,\ i = 0..n)/j!,\ j = 0..m)\ \mathbf{end\ proc}$

Die symbolische Doppelsumme muß nun in eine entsprechende Doppelschleife transformiert werden (**codegen,prep2trans**), damit eine Übersetzung nach C bzw. Fortran erfolgen kann.

```
> F:=prep2trans(F);
```

$F :=$ **proc**$(n,\ m,\ x,\ y)$
local $i1,\ i2,\ s1,\ s2,\ t1,\ t2$;
if $m < 0$ **then** $s1 := 0$
else
$t1 := 1$;
$s1 := 1$;
for $i1$ **to** m **do** $t1 := y * t1/i1$; $s1 := s1 + t1$ **end do**;
if $n < 0$ **then** $s2 := 0$
else $t2 := 1$; $s2 := 1$; **for** $i2$ **to** n **do** $t2 := x * t2/i2$; $s2 := s2 + t2$ **end do**
end if;
$s1 := s2 * s1$
end if;
$s1$
end proc

Die Datentypen der Parameter können mit **declare** festgelegt werden.

```
> F:=declare([n::integer,m::integer,x::numeric,y::numeric],F);
```

$F :=$ **proc**$(n$::$integer,\ m$::$integer,\ x$::$numeric,\ y$::$numeric$)$
local $i1,\ i2,\ s1,\ s2,\ t1,\ t2$;
if $m < 0$ **then** $s1 := 0$
else
$t1 := 1$;
$s1 := 1$;
for $i1$ **to** m **do** $t1 := y * t1/i1$; $s1 := s1 + t1$ **end do**;
if $n < 0$ **then** $s2 := 0$
else $t2 := 1$; $s2 := 1$; **for** $i2$ **to** n **do** $t2 := x * t2/i2$; $s2 := s2 + t2$ **end do**
end if;
$s1 := s2 * s1$
end if;
$s1$
end proc

Die eigentliche Übersetzung in die Zielsprachen C bzw. Fortran erfolgt schließlich mit den Funktionen **C** bzw. **fortran**:

```
> C(F);
```

```
double F(n,m,x,y)
int n;
int m;
double x;
double y;
{
  int i1;
  int i2;
  double s1;
  double s2;
  double t1;
  double t2;
  {
    if( m < 0.0 )
      s1 = 0.0;
    else
      {
        t1 = 1.0;
        s1 = 1.0;
        for(i1 = 1;i1 <= m;i1++)
        {
```

```
          t1 = y/i1*t1;
          s1 += t1;
        }
        if( n < 0.0 )
          s2 = 0.0;
        else
          {
            t2 = 1.0;
            s2 = 1.0;
            for(i2 = 1;i2 <= n;i2++)
            {
              t2 = x/i2*t2;
              s2 += t2;
            }
          }
        s1 = s2*s1;
      }
    return(s1);
  }
}

> fortran(F);

      real function F(n,m,x,y)
      integer n
      integer m
      real x
      real y

      integer i1
      integer i2
      real s1
      real s2
      real t1
      real t2

        if (m .lt. 0) then
          s1 = 0
        else
          t1 = 1
          s1 = 1
          do 1000 i1 = 1,m,1
            t1 = y/i1*t1
            s1 = s1+t1
1000      continue
          if (n .lt. 0) then
            s2 = 0
          else
            t2 = 1
            s2 = 1
            do 1100 i2 = 1,n,1
              t2 = x/i2*t2
              s2 = s2+t2
1100        continue
          endif
          s1 = s2*s1
        endif
        F = s1
        return
      end
```

In Fortran wird standardmäßig eine Funktion in einfacher Genauigkeit generiert. Eine doppeltgenaue Funktion erhält man mit der Option $precision = double$

```
> fortran(F,precision=double);

      doubleprecision function F(n,m,x,y)
      integer n
      integer m
      doubleprecision x
      doubleprecision y

      integer i1
      integer i2
      doubleprecision s1
      doubleprecision s2
      doubleprecision t1
      doubleprecision t2

        if (m .lt. 0.D0) then
          s1 = 0.D0
        else
          t1 = 1.D0
          s1 = 1.D0
          do 1000 i1 = 1.D0,m,1.D0
            t1 = y/i1*t1
            s1 = s1+t1
1000      continue
          if (n .lt. 0.D0) then
            s2 = 0.D0
          else
            t2 = 1.D0
            s2 = 1.D0
            do 1100 i2 = 1.D0,n,1.D0
              t2 = x/i2*t2
              s2 = s2+t2
1100        continue
          endif
          s1 = s2*s1
        endif
        F = s1
        return
      end
```

9.2 Optimierung von Berechnungssequenzen

```
> restart:
```

```
> with(codegen):
```

In einer Anwendung muß das Integral $\int x^3 \sin(3\,x)\, e^x + x^2 \sin(x)\, e^{(3\,x)}\, dx$ sehr oft numerisch ausgewertet werden. Die Stammfunktion ist gegeben durch

```
> F:=unapply(int(x^3*sin(3*x)*exp(x)+x^2*sin(x)*exp(3*x),x),x);
```

$$F := x \rightarrow (-\frac{3}{10}\,x^3 + \frac{9}{50}\,x^2 + \frac{27}{250}\,x - \frac{36}{625})\, e^x \cos(3\,x)$$
$$+ (\frac{1}{10}\,x^3 + \frac{6}{25}\,x^2 - \frac{39}{250}\,x - \frac{21}{1250})\, e^x \sin(3\,x) + (-\frac{1}{10}\,x^2 + \frac{3}{25}\,x - \frac{13}{250})\, e^{(3\,x)} \cos(x)$$
$$+ (\frac{3}{10}\,x^2 - \frac{4}{25}\,x + \frac{9}{250})\, e^{(3\,x)} \sin(x)$$

Wir suchen nun für $\mathrm{F}(x)$ einen Ausdruck mit möglichst wenig Rechenoperationen. Dazu transformieren wir F in eine Prozedur mit optimierter Berechnungssequenz (**codegen,optimize**).

```
> F1:=optimize(F);
```

$F1 := \mathbf{proc}(x)$
local $t9,\ t10,\ t17,\ t7,\ t22,\ t24,\ t30,\ t1,\ t2$;
option $operator,\ arrow$;
$t1 := x^2$;
$t2 := t1 * x$;
$t7 := \exp(x)$;
$t9 := 3 * x$;
$t10 := \cos(t9)$;
$t17 := \sin(t9)$;
$t22 := \exp(t9)$;
$t24 := \cos(x)$;
$t30 := \sin(x)$;
$(-3/10 * t2 + 9/50 * t1 + 27/250 * x - 36/625) * t7 * t10$
$+ (1/10 * t2 + 6/25 * t1 - 39/250 * x - 21/1250) * t7 * t17$
$+ (-1/10 * t1 + 3/25 * x - 13/250) * t22 * t24$
$+ (3/10 * t1 - 4/25 * x + 9/250) * t22 * t30$
end proc

Die Funktion *optimize* nutzt Maple's **remember** facility und identifiziert gemeinsame Teilausdrücke (linear in Zeit und Speicherplatz), die dann in der optimierten Anweisungssequenz nur einmal berechnet werden. Die Funktionen F und $F1$ berechnen die gleichen Funktionswerte,

```
> simplify((F-F1)(x));
```

$$0$$

die Berechnungskosten aber sind sehr unterschiedlich:

```
> cost(F); cost(F1);
```

$$13\ additions + 30\ multiplications + 8\ functions$$
$$9\ storage + 9\ assignments + 21\ multiplications + 6\ functions + 13\ additions$$

Die Kosten können weiter minimiert werden, indem wir einen aufwendigeren Optimierungsalgorithmus verwenden:

```
> F2:=optimize(F,tryhard);
```

$F2 := \mathbf{proc}(x)$
local $t3,\ t5,\ t4,\ result$;
$t5 := 3 * x$;
$t3 := x^2$;
$t4 := x * t3$;
$result := ((-3/10 * t4 + 9/50 * t3 + 27/250 * x - 36/625) * \cos(t5)$
$+ (1/10 * t4 + 6/25 * t3 - 39/250 * x - 21/1250) * \sin(t5)) * \exp(x) +$
$((-1/10 * t3 + 3/25 * x - 13/250) * \cos(x) + (3/10 * t3 - 4/25 * x + 9/250) * \sin(x)) *$
$\exp(t5)$
end proc

```
> simplify((F-F2)(x));
```

$$0$$

```
> cost(F2);
```

$$4\ storage + 4\ assignments + 19\ multiplications + 13\ additions + 6\ functions$$

Es konnten also noch 2 Multiplikationen eingespart werden. Die Prozeduren $F1$ und $F2$ enthalten noch jeweils zwei Funktionsauswertungen von $\exp$, $\sin$ und $\cos$. Wir wollen diese Funktionsauswertungen nun auf jeweils eine reduzieren, indem wir $e^{(3\,x)}$ in $(e^x)^3$ umwandeln und für $\sin(3\,x)$ und $\cos(3\,x)$ die

Additionstheoreme für Sinus und Kosinus anwenden. Durch Substitution der entsprechenden Regeln erhalten wir

```
> F3:=subs(exp(3*x)=expand(exp(3*x)),sin(3*x)=expand(sin(3*x)),cos(3*x)
> =expand(cos(3*x)),F(x));
```

$$F3 := (-\frac{3}{10}\,x^3 + \frac{9}{50}\,x^2 + \frac{27}{250}\,x - \frac{36}{625})\,e^x\,(4\cos(x)^3 - 3\cos(x))$$
$$+ (\frac{1}{10}\,x^3 + \frac{6}{25}\,x^2 - \frac{39}{250}\,x - \frac{21}{1250})\,e^x\,(4\sin(x)\cos(x)^2 - \sin(x))$$
$$+ (-\frac{1}{10}\,x^2 + \frac{3}{25}\,x - \frac{13}{250})\,(e^x)^3\cos(x) + (\frac{3}{10}\,x^2 - \frac{4}{25}\,x + \frac{9}{250})\,(e^x)^3\sin(x)$$

```
> cost(optimize(F3));
```

$$9\,assignments + 28\,multiplications + 3\,functions + 15\,additions$$

Wir haben drei Funktionsauswertungen eingespart, aber deutlich mehr Multiplikationen bekommen. $F3$ ist ein Polynom in x, e^x, $\sin(x)$ und $\cos(x)$. Für Polynome wird die effizienteste Auswertung mit dem Horner-Schema erreicht. Wenden wir das Horner-Schema rekursiv auf die Variablen e^x, $\sin(x)$, $\cos(x)$ und x an, dann ist die Anzahl der Rechenoperationen des resultierenden Ausdrucks minimal.

```
> F4:=convert(F3,horner,[exp(x),sin(x),cos(x),x]);
```

$$F4 := ((-\frac{36}{625} + (\frac{27}{250} + (\frac{9}{50} - \frac{3}{10}\,x)\,x)\,x)\,(-3 + 4\cos(x)^2)\cos(x)$$
$$+ (-\frac{21}{1250} + (-\frac{39}{250} + (\frac{6}{25} + \frac{1}{10}\,x)\,x)\,x)\,(4\cos(x)^2 - 1)\sin(x)$$
$$+ ((-\frac{13}{250} + (\frac{3}{25} - \frac{1}{10}\,x)\,x)\cos(x) + (\frac{9}{250} + (-\frac{4}{25} + \frac{3}{10}\,x)\,x)\sin(x))\,(e^x)^2)e^x$$

```
> F5:=unapply(F4,x):
> cost(optimize(F5));
```

$$8\,storage + 8\,assignments + 19\,multiplications + 3\,functions + 15\,additions$$

$F5$ hat gegenüber $F2$ drei Funktionsauswertungen weniger, dafür 2 Additionen und jeweils vier Zuweisungen und Speicheroperationen mehr, was die eingesparten Funktionsauswertungen aber sicher nicht aufwiegt. Die optimierte Prozedur $F5$ übersetzen wir jetzt nach C:

```
> C(F5,optimized);

/* The options were    : operatorarrow */

#include <math.h>

double F5(x)
double x;

{
  double t1;
  double t13;
  double t21;
  double t32;
  double t33;
  double t7;
  double t8;
  double t9;
  {
    t1 = 3.0/10.0*x;
    t7 = cos(x);
    t8 = t7*t7;
    t9 = 4.0*t8;
    t13 = x/10.0;
    t21 = sin(x);
    t32 = exp(x);
    t33 = t32*t32;

return(((-36.0/625.0+(27.0/250.0+(9.0/50.0-t1)*x)*x)*(-3.0+t9)*t7+
```

```
      (-21.0/1250.0+(-39.0/250.0+(6.0/25.0+t13)*x)*x)*(t9-1.0)*t21+(
      (-13.0/250.0+(3.0/25.0-t13)*x)*t7+(9.0/250.0+(-4.0/25.0+t1)*x)
      *t21)*t33)*t32);
    }
  }
```

9.3 Automatische Differentiation

```
> restart:
```

Der Operator **D** ermöglicht die Differentiation von Maple-Prozeduren (automatische Differentiation). Beispiel:

```
> f:=proc(x,y)
> local t1, t2, t3, t4;
> t1:=x^2;
> t2:=sin(t1);
> t3:=y*t2 - 4*cos(t1);
> t4:=(t1-y)*t3;
> end proc;
```

$f := \mathbf{proc}(x,\ y)$
local $t1,\ t2,\ t3,\ t4$;
$t1 := x^2\,;\ t2 := \sin(t1)\,;\ t3 := y * t2 - 4 * \cos(t1)\,;\ t4 := (t1 - y) * t3$
end proc

Die Prozedur f repräsentiert die Formel

```
> f(x,y);
```

$$(x^2 - y)\,(y\sin(x^2) - 4\cos(x^2))$$

Die Differentiation von f nach x ergibt

```
> fx:=D[1](f);
```

$fx := \mathbf{proc}(x,\ y)$
local $t1,\ t2,\ t3,\ t1x,\ t2x,\ t3x$;
$t1x := 2 * x\,;$
$t1 := x^2\,;$
$t2x := \cos(t1) * t1x\,;$
$t2 := \sin(t1)\,;$
$t3x := y * t2x + 4 * \sin(t1) * t1x\,;$
$t3 := y * t2 - 4 * \cos(t1)\,;$
$t1x * t3 + (t1 - y) * t3x$
end proc

Die erhaltene Prozedur *fx* berechnet die partielle Ableitung von f nach x. Wir überprüfen dies, indem wir mit **diff** den Ausdruck $\mathrm{f}(x,\ y)$ nach x symbolisch differenzieren und dann $\mathrm{fx}(x,\ y)$ subtrahieren (siehe auch **operators,D**).

```
> diff(f(x,y),x)- fx(x,y);
```

$$0$$

Wie wurde die Prozedur *fx* konstruiert? Die Methode des automatischen Differenzierens wird durch Vergleich der Prozedur-Körper von f und *fx* deutlich:

Jeder Anweisung $u := g(\ v_1,...,\ v_n)$ in der Prozedur f wurde eine Anweisung $u_x := gp(\ v_1,...,\ v_n)$ vorangestellt, $gp(\ v_1,...,\ v_n)$ ist hierbei die formale Ableitung von $g(\ v_1,...,\ v_n)$, $v_1,...,\ v_n$ sind lokale Variable oder formale Parameter. Die letzte Anweisung des Programms wurde durch ihre Ableitung ersetzt.

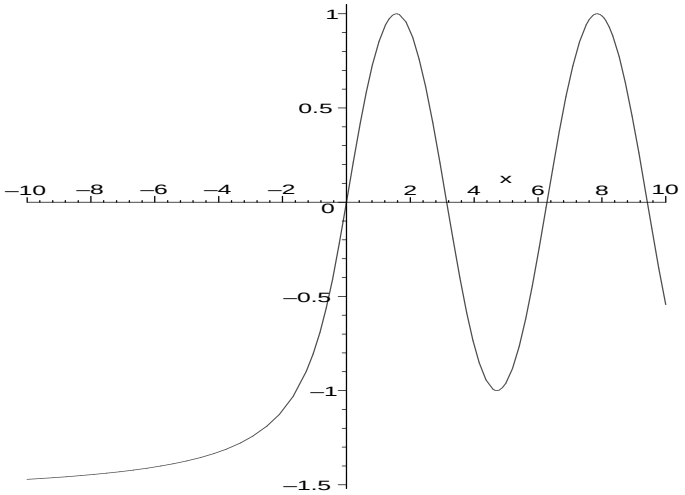
Dieses Verfahren wird als *forward mode* des automatischen Differenzierens bezeichnet. Bei der automatischen Differentiation werden also die Ableitungsregeln (siehe Kapitel 3.4) auf die einzelnen Rechenschritte (Anweisungen) eines Programms angewendet.

Mit dem Operator D lassen sich auch Maple-Funktionen, die stückweise definiert sind, also nicht mehr als Formel darstellbar sind, differenzieren; dies ist ein Grund, warum automatisches Differenzieren eingeführt wurde. Beispiel:

```
> f:=x -> if x > 0 then sin(x) else arctan(x) end if;
```

$f := \mathbf{proc}(x)\ \mathbf{option}\ operator,\ arrow;\ \mathbf{if}\ 0 < x\ \mathbf{then}\ \sin(x)\ \mathbf{else}\ \arctan(x)\ \mathbf{end\ if}\ \mathbf{end\ proc}$

```
> plot('f(x)',x=-10..10);
```



Die Funktion f ist an der Stelle $x = 0$ differenzierbar.

```
> Limit((sin(h) - sin(0))/h,h=0,right) =
> Limit((arctan(h) - arctan(0))/h,h=0,left);
```

$$\lim_{h\to 0+} \frac{\sin(h)}{h} = \lim_{h\to 0-} \frac{\arctan(h)}{h}$$

```
> value(%);
```

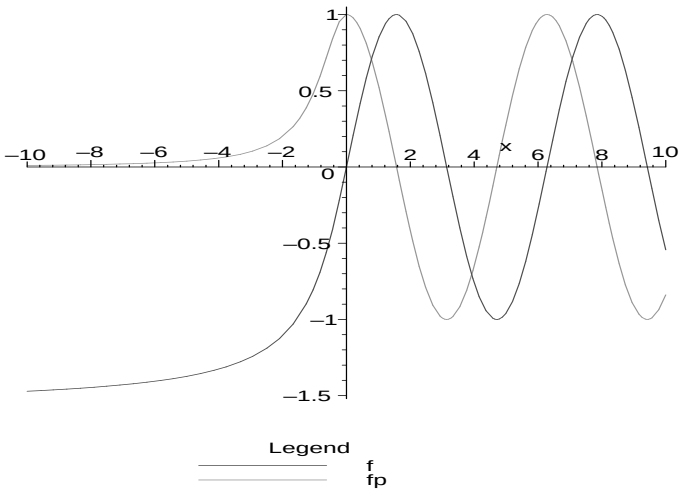
$$1 = 1$$

Als Ableitungsfunktion erhalten wir die Prozedur

```
> fp:=D(f);
```

$fp := \mathbf{proc}(x)\ \mathbf{option}\ operator,\ arrow;\ \mathbf{if}\ 0 < x\ \mathbf{then}\ \cos(x)\ \mathbf{else}\ 1/(x^2+1)\ \mathbf{end\ if}\ \ \mathbf{end\ proc}$

```
> plot(['f(x)','fp(x)'],x=-10..10,legend=[f,fp]);
```



Ein weiterer Grund, der für das automatische Differenzieren spricht, ist die Effizienz bei den Rechenkosten. Wir illustrieren dies am Beispiel der wiederholten Potenzfunktion, die rekursiv durch $g_1 = x$, $g_n = x^{g_{n-1}}$ für $1 < n$ definiert ist.

Die zugehörige Maple-Prozedur lautet:

```
> g:=proc(x,n)
> local i,t;
> t:=1;
> for i to n do
> t:=x^t
> end do;
> t
> end proc;
```

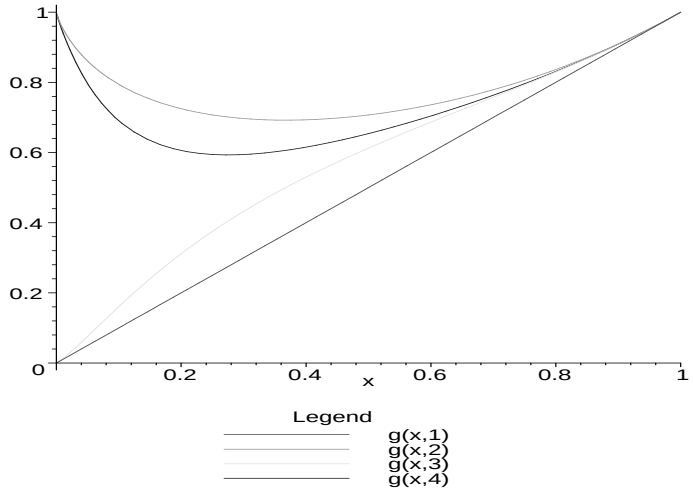
$$g := \mathbf{proc}(x,\, n)\, \mathbf{local}\, i,\, t;\, t := 1\,;\, \mathbf{for}\, i\, \mathbf{to}\, n\, \mathbf{do}\, t := x^t\, \mathbf{end\ do}\,;\, t\, \mathbf{end\ proc}$$

Es folgen die ersten vier Potenzfunktionen mit den zugehörigen Graphen:

```
> seq(g(x,n),n=1..4);
```

$$x,\, x^x,\, x^{(x^x)},\, x^{(x^{(x^x)})}$$

```
> plot([seq(g(x,n),n=1..4)],x=0..1,
> legend=[seq("g(x,"||i||")",i=1..4)]);
```



Einige Grenzwerte bei 0 und 1 sind:

```
> limit(g(x,2),x=0); limit(g(x,2),x=1);
```

$$1$$
$$1$$

```
> limit(g(x,3),x=0); limit(g(x,2),x=1);
```

$$0$$
$$1$$

Mit automatischer Differentiation erhalten wir für die dritte Ableitung von g:

```
> D[1$3](g);
```

proc$(x,\ n)$
local $tx1x,\ t1xx,\ txxx,\ tx,\ i,\ t,\ tx,\ txx,\ t1x$;
$txxx := 0$;
$txx := 0$;
$tx1x := 0$;
$tx := 0$;
$t1xx := 0$;
$t1x := 0$;
$tx := 0$;
$t := 1$;
for i **to** n **do**

$$\begin{aligned} txxx := {} & x^t * (tx * \ln(x) + t/x) * (t1x * \ln(x) + t/x) * (tx * \ln(x) + t/x) \\ & + x^t * (t1xx * \ln(x) + t1x/x + tx/x - t/x^2) * (tx * \ln(x) + t/x) \\ & + x^t * (t1x * \ln(x) + t/x) * (tx1x * \ln(x) + tx/x + tx/x - t/x^2) \\ & + x^t * (tx * \ln(x) + t/x) * (txx * \ln(x) + tx/x + t1x/x - t/x^2) \\ & + x^t * (txxx * \ln(x) + txx/x + tx1x/x - tx/x^2 + t1xx/x - t1x/x^2 - tx/x^2 + 2 * t/x^3); \end{aligned}$$

$$txx := x^t * (t1x * \ln(x) + t/x) * (tx * \ln(x) + t/x) + x^t * (txx * \ln(x) + tx/x + t1x/x - t/x^2);$$

$$tx1x := x^t * (tx * \ln(x) + t/x) * (tx * \ln(x) + t/x) + x^t * (tx1x * \ln(x) + tx/x + tx/x - t/x^2);$$

$$tx := x^t * (tx * \ln(x) + t/x);$$

$$t1xx := x^t * (tx * \ln(x) + t/x) * (t1x * \ln(x) + t/x) + x^t * (t1xx * \ln(x) + t1x/x + tx/x - t/x^2);$$

$$t1x := x^t * (t1x * \ln(x) + t/x);$$

$$tx := x^t * (tx * \ln(x) + t/x);$$

$t := x^t$
end do;
$txxx$
end proc

Wir wollen nun mittels automatischer und symbolischer Differentiation die dritte Ableitung von g_{15} an der Stelle $x = .9$ bestimmen. Die benötigten Rechenkosten (CPU-Zeit und Hauptspeicherplatz) halten wir jeweils fest (**kernelopts**):

```
> setbytes:=kernelopts(bytesused);
```

$$setbytes := 23845228$$

```
> settime:=time();
```

$$settime := 7.014$$

```
> D[1$3](g)(0.9,15);
```

$$5.160434458$$

```
> cpu_time1:=(time() - settime)*seconds;
```

$$cpu\_time1 := .160\ seconds$$

```
> memory_used1:=evalf((kernelopts(bytesused) -
> setbytes)/1000* kbytes,4);
```

$$memory\_used1 := 803.6\ kbytes$$

Mit symbolischer Differentiation ergibt sich für die dritte Ableitung von g_{15}

```
> g(x,15);
```

$$x^{(x^{(x^{(x^{(x^{(x^{(x^{(x^{(x^{(x^{(x^{(x^{(x^{(x^{x})})})})})})})})})})})})}$$

an der Stelle $x = .9$:

```
> setbytes:=kernelopts(bytesused);
```

$$setbytes := 24650128$$

```
> settime:=time();
```

$$settime := 7.174$$

```
> eval(diff(g(x,15),x$3),x=0.9);
```

$$5.160434458$$

```
> cpu_time2:=(time() - settime)*seconds;
```

$$cpu\_time2 := 3.909\ seconds$$

```
> memory_used2:=evalf((kernelopts(bytesused) -
> setbytes)/1000* kbytes,4);
```

$$memory\_used2 := 12400.\ kbytes$$

Die symbolische Differentiation benötigt somit

```
> cpu_time2/cpu_time1;
```

$$24.43125000$$

mal soviel Zeit und

```
> memory_used2/memory_used1;
```

$$15.43056247$$

mal soviel Speicherplatz.

9.4 Gradienten und Hesse-Matrix

```
> restart:
```

```
> with(codegen);
```

$[C,\ GRAD,\ GRADIENT,\ HESSIAN,\ JACOBIAN,\ cost,\ declare,\ dontreturn,\ eqn,\ fortran,$
$horner,\ intrep2maple,\ joinprocs,\ makeglobal,\ makeparam,\ makeproc,\ makevoid,$
$maple2intrep,\ optimize,\ packargs,\ packlocals,\ packparams,\ prep2trans,$
$renamevar,\ split,\ swapargs]$

Gegeben sei die Maple-Prozedur

```
> f:=proc(x,y)
> local u,v,w;
> u:=x^2;
> v:=y^2;
> w:=x*y;
> 3*u*w+5*v*w
> end proc;
```

$f := \mathbf{proc}(x,\ y)\ \mathbf{local}\ u,\ v,\ w;\ u := x^2\ ;\ v := y^2\ ;\ w := x * y\ ;\ 3 * u * w + 5 * v * w\ \mathbf{end\ proc}$

zur Berechnung des Polynoms

```
> f(x,y);
```

$$3\,x^3\,y + 5\,y^3\,x$$

Die Prozedur **GRADIENT** berechnet den Gradienten von f, d. h. $\frac{\partial}{\partial x}\,\mathrm{f}(x,\ y),\ \frac{\partial}{\partial y}\,\mathrm{f}(x,\ y)$, mit Hilfe automatischer Differentiation:

```
> fr:=GRADIENT(f);
```

$$
\begin{array}{l}
fr := \textbf{proc}(x,\ y) \\
\textbf{local}\ dfr0,\ u,\ v,\ w; \\
\quad u := x^2\,; \\
\quad v := y^2\,; \\
\quad w := x * y\,; \\
\quad dfr0 := \mathrm{array}(1..3)\,; \\
\quad dfr0_3 := 3 * u + 5 * v\,; \\
\quad dfr0_2 := 5 * w\,; \\
\quad dfr0_1 := 3 * w\,; \\
\quad \mathrm{RETURN}(dfr0_3 * y + 2 * dfr0_1 * x,\ dfr0_3 * x + 2 * dfr0_2 * y) \\
\textbf{end proc}
\end{array}
$$

Der Vergleich mit symbolischer Differentiation liefert:

```
> expand(fr(x,y)[1]-diff(f(x,y),x));
```

$$0$$

```
> expand(fr(x,y)[2]-diff(f(x,y),y));
```

$$0$$

Es gibt zwei Algorithmen für die automatische Differentiation, den *forward mode* und den *reverse mode*. Beim *reverse mode* wird die Berechnungssequenz der Prozedur einmal in Vorwärtsrichtung zur Bestimmung des Funktionswertes und einmal in Rückwärtsrichtung zur Bestimmung der Ableitung durchlaufen. Im allgmeinen werden mit dem *reverse mode* effizientere Programme erzeugt, weshalb der *reverse mode* bei GRADIENT das Standardverfahren ist.

```
> ff:=GRADIENT(f,mode=forward);
```

$$
\begin{array}{l}
ff := \textbf{proc}(x,\ y) \\
\textbf{local}\ du,\ dv,\ dw,\ u,\ v,\ w; \\
\quad du := \mathrm{array}(1..2)\,; \\
\quad dv := \mathrm{array}(1..2)\,; \\
\quad dw := \mathrm{array}(1..2)\,; \\
\quad du_1 := 2 * x\,; \\
\quad du_2 := 0\,; \\
\quad u := x^2\,; \\
\quad dv_1 := 0\,; \\
\quad dv_2 := 2 * y\,; \\
\quad v := y^2\,; \\
\quad dw_1 := y\,; \\
\quad dw_2 := x\,; \\
\quad w := x * y\,; \\
\quad \mathrm{RETURN}(3 * w * du_1 + 5 * w * dv_1 + (3 * u + 5 * v) * dw_1, \\
\qquad 3 * w * du_2 + 5 * w * dv_2 + (3 * u + 5 * v) * dw_2) \\
\textbf{end proc}
\end{array}
$$

```
> cost(ff); cost(fr);
```

$$9\ storage + 12\ assignments + 12\ subscripts + 19\ multiplications + 6\ additions$$

$$6\ storage + 7\ assignments + 13\ multiplications + 7\ subscripts + 3\ additions$$

GRADIENT kann Prozeduren mit Kontrollstrukturen differenzieren:

```
> g:=proc(x,y,n)
> local i, t;
> t:=x*y;
> for i to n do
> t:=t*x*y
> end do;
> if x < y then
> t*x
> else
> t*y
> end if;
> end proc:
> grad_g:=GRADIENT(g,[x,y]);


Can't apply the reverse mode.

Trying with the forward mode...
```

$$
\begin{array}{l}
grad\_g := \mathbf{proc}(x,\, y,\, n) \\
\mathbf{local}\ dt,\, i,\, t; \\
\quad dt := \mathrm{array}(1..2)\,; \\
\quad dt_1 := y\,; \\
\quad dt_2 := x\,; \\
\quad t := x * y\,; \\
\quad \mathbf{for}\ i\ \mathbf{to}\ n\ \mathbf{do}\ dt_1 := t * y + x * y * dt_1\,;\ dt_2 := t * x + x * y * dt_2\,;\ t := t * x * y\ \mathbf{end\ do}\,; \\
\quad \mathbf{if}\ x < y\ \mathbf{then}\ \mathrm{RETURN}(t + dt_1 * x,\, x * dt_2) \\
\qquad\quad \mathbf{else}\ \mathrm{RETURN}(y * dt_1,\, t + dt_2 * y) \\
\quad \mathbf{end\ if} \\
\mathbf{end\ proc}
\end{array}
$$

Der *reverse mode* kann nicht angewendet werden, wenn die Anzahl der Schleifendurchläufe in der Prozedur unbestimmt ist. GRADIENT verarbeitet auch Prozeduren mit Unterprozeduren:

```
> h:=proc(x)
> local s, d;
> d:=proc(u)
> local t;
> t:=u^2+u;
> t*u
> end proc;
> s:=d(x);
> s*x
> end proc:
> h(x);
```

$$(x^2 + x)\, x^2$$

```
> grad_h:=GRADIENT(h);
```

$grad\_h := \mathbf{proc}(x)$
local $d,\ dd,\ df,\ lf1,\ s$;
$dd := \mathbf{proc}(u)$
local $df,\ t$;
$t := u^2 + u\,;\ df := \mathrm{array}(1..1)\,;\ df_1 := u\,;\ \mathrm{RETURN}([t + df_1 * (2 * u + 1)])$
end proc;
$d := \mathbf{proc}(u)\ \mathbf{local}\ t;\ t := u^2 + u\,;\ t * u\ \mathbf{end\ proc}\,;$
$s := d(x)\,;$
$df := \mathrm{array}(1..1)\,;$
$lf1 := dd(x)\,;$
$df_1 := x\,;$
$\mathrm{RETURN}(s + df_1 * lf1_1)$
end proc

```
> expand(diff(h(x),x) - grad_h(x));
```

$$0$$

Bemerkung: Der Operator D ist im Vergleich zu GRADIENT weniger leistungsfähig: es steht nur der *forward mode* zur Verfügung, es können keine Prozeduren mit (lokalen) Unterprozeduren differenziert werden, u.a..

Wir berechnen nun die Hesse-Matrix der Funktion f mittels automatischer Differentiation (**HESSIAN**).

```
> hess_f:=HESSIAN(f);
```

$hess\_f := \mathbf{proc}(x,\ y)$
local $df,\ dfr0,\ dfr01,\ dfr02,\ dfr03,\ grd,\ grd1,\ grd2,\ t3,\ u,\ v,\ w$;
$u := x^2\,;$
$v := y^2\,;$
$w := x * y\,;$
$dfr03 := 3 * u + 5 * v\,;$
$dfr02 := 5 * w\,;$
$dfr01 := 3 * w\,;$
$t3 := dfr03\,;$
$grd1 := t3 * y + 2 * dfr01 * x\,;$
$grd2 := t3 * x + 2 * dfr02 * y\,;$
$df := \mathrm{array}(1..9)\,;$
$dfr0 := \mathrm{array}(1..9)\,;$
$df_8 := 1\,;$
$df_7 := df_8 * y\,;$
$df_6 := 2 * df_8 * x\,;$
$df_4 := df_7\,;$
$df_3 := 3 * df_6\,;$
$df_2 := 5 * df_4\,;$
$df_1 := 3 * df_4\,;$
$dfr0_9 := 1\,;$
$dfr0_7 := dfr0_9 * x\,;$
$dfr0_5 := 2 * dfr0_9 * y\,;$
$dfr0_4 := dfr0_7\,;$

$dfr0_3 := 5 * dfr0_5$;
$dfr0_2 := 5 * dfr0_4$;
$dfr0_1 := 3 * dfr0_4$;
$grd := \mathrm{array}(1..2,\ 1..2)$;
$grd_{1,1} := 2 * df_8 * dfr01 + df_3 * y + 2 * df_1 * x$;
$grd_{1,2} := df_8 * t3 + df_3 * x + 2 * df_2 * y$;
$grd_{2,1} := dfr0_9 * t3 + dfr0_3 * y + 2 * dfr0_1 * x$;
$grd_{2,2} := 2 * dfr0_9 * dfr02 + dfr0_3 * x + 2 * dfr0_2 * y$;
RETURN(grd)

end proc

```
> eval(hess_f(x,y));
```

$$\begin{bmatrix} 18\,x\,y & 9\,x^2 + 15\,y^2 \\ 9\,x^2 + 15\,y^2 & 30\,x\,y \end{bmatrix}$$

Die zugehörige C-Routine ist dann

```
> C(hess_f,optimized);
```

```
void hess_f(x,y,grd)
double x;
double y;
double grd[2][2];
{
  double df[9];
  double dfr0[9];
  double dfr01;
  double dfr02;
  double dfr03;
  double t13;
  double t22;
  double u;
  double v;
  double w;
  {
    u = x*x;
    v = y*y;
    w = x*y;
    dfr03 = 3.0*u+5.0*v;
    dfr02 = 5.0*w;
    dfr01 = 3.0*w;
    df[5] = 2.0*x;
    df[2] = 3.0*df[5];
    df[1] = 5.0*y;
    df[0] = 3.0*y;
    dfr0[4] = 2.0*y;
    dfr0[2] = 5.0*dfr0[4];
    dfr0[1] = 5.0*x;
    dfr0[0] = 3.0*x;
    t13 = df[2];
    grd[0][0] = 2.0*dfr01+t13*y+2.0*df[0]*x;
    grd[0][1] = dfr03+t13*x+2.0*df[1]*y;
    t22 = dfr0[2];
    grd[1][0] = dfr03+t22*y+2.0*dfr0[0]*x;
    grd[1][1] = 2.0*dfr02+t22*x+2.0*dfr0[1]*y;
    return;
  }
}
```

Literaturverzeichnis

[1] K. M. Heal, M. L. Hansen, K. M. Rickard, *Maple 6 Learning Guide*, waterloo Maple Inc., 2000, ISBN 1-894511-00-X

[2] M. B. Monagan, K. O. Geddes, K. M. Heal, G. Labahn, S. M. Vorkoetter, J. McCarron *Maple 6 Programming Guide*, Waterloo Maple Inc., 2000, ISBN 1-894511-01-8

[3] A. Heck, *Introduction to Maple*, 2nd ed., 1996, Springer-Verlag, ISBN 0-387-94535-0

[4] M. Eikelberg, *Einführung in die Arbeit mit Maple V*, 1998, Fachbuchverlag Leipzig im Carl Hanser Verlag, ISBN 3-446-19458-4

[5] M. Hörhager, *Maple in Technik und Wissenschaft*, 1996, Addison-Wesley (Deutschland) GmbH, ISBN 3-89319-929-2

[6] T. Westermann, *Mathematik für Ingenieure mit Maple, Band 1: Differential- und Integralrechnung für Funktionen einer Variablen, Vektor- und Matrizenrechnung, Komplexe Zahlen, Funktionenreihen*, 1996, Springer-Verlag, ISBN 3-540-61249-1

[7] W. Werner, *Mathematik lernen mit Maple V: Ein Lehr- und Arbeitsbuch für das Grundstudium*, 1993, ELBI-Verlag, ISBN 3-929694-03-4

[8] W. Gander and D. Gruntz, *Derivation of Numerical Methods Using Computer Algebra*, SIAM Review, Vol. **41**, No. 3, 577-593, 1999

Index

T

U

V

W

Z