19 Maple

Britta Nestler

19.1 Grundlagen

Maple ist ein Computeralgebrasystem, dessen Entwicklung im Jahr 1980 von Professoren und Wissenschaftlern an der Universität Waterloo in Canada begonnen wurde.

19.1.1 Programmbedienung

Je nach System (Windows, DOS, UNIX etc.) wird Maple mit `xmaple`, `maple` oder durch Anklicken des Ahornsymbols aufgerufen.

Nach dem Einladen des Maple-Kernels wird der Benutzer durch die Anzeige eines Prompts '>' zur Eingabe von Befehlen aufgefordert.

Durch `Enter` am Ende des Arbeitsblatts (Worksheet) wird eine neue Prompt-Aufforderung für weitere Eingaben erzeugt. Um in einer laufenden Berechnung einen Prompt einzufügen, kann das entsprechende Symbol in der Menüleiste angewendet werden. Beim Starten einer Sitzung steht ein Teil der Befehle und Funktionen sofort zur Verfügung. Zahlreiche andere müssen vom Benutzer durch Hinzuladen von Paketen eingefÃijgt werden.

Ein eingegebener Befehl endet mit einem Semikolon '`;`:, wenn das Ergebnis angezeigt werden soll, oder mit einem Doppelpunkt '`:`', wenn der Rechenbefehl zwar ausgeführt, die Ausgabe des Ergebnisses auf dem aber Bildschirm unterdrückt werden soll. Mehrere Befehle können nacheinander in einer Zeile aufgereiht werden. Für einen längeren Befehl sind auch mehrere Zeilen nutzbar. Die Ausführung eines Maple-Befehls wird durch Drücken der `Enter/Return`-Taste gestartet. Alternativ kann auch das einfache Ausrufezeichen '!' in der Menüleiste verwendet werden.

Zur besseren grafischen Aufbereitung lässt sich mit `Shift-Enter` bei der Eingabe einer Formel ein Zeilenumbruch erwirken. Maple unterscheidet zwischen Groß- und Kleinschreibung. Bei Befehlen mit einem großen Anfangsbuchstaben wird die Rechenoperation angezeigt. Wird eine Eingabe mit einem kleinen Anfangsbuchstaben gesetzt, so wird die Rechenoperation ausgeführt.

Die meisten der Funktionsnamen beginnen jedoch mit einem Kleinbuchstaben wie z.B. die grundlegenden Befehle `op`,`type` und `evalf`. Leerzeichen innerhalb von Befehlen werden ignoriert, so dass deren Einfügen sich oft sehr gut zur Erhöhung der Lesbarkeit eignet. Beachten sollte man neben der Klein- und Großschreibung die Arten der benutzten Klammern, ob eckig '`[ ]`', rund '`()`' oder Mengenklammern '`{ }`', und die Reihenfolge der verwendeten Zeichen.

Durch Einfügen des Prozentzeichens '`%`' kann auf das letzte Ergebnis zurückgegriffen und dieses direkt für den nachfolgenden Rechenbefehl weiterverwendet werden. Das vorletzte Ergebnis kann mit zwei hintereinander gesetzten Prozentzeichen '`%%`' und das vorvorletzte mit drei '`%%%`' als Eingabe direkt wieder verwendet werden.

Durch Betätigen des Maple-eigenen Stop-Icons kann eine Berechnung unterbrochen werden.

Über die Menüsteuerung „File → Beenden/Exit" kann eine Maple-Sitzung und mit „File → Schließen/Close" ein aktuelles Arbeitsdokument (Worksheet) beendet werden. Das vorherige Speichern eines Arbeitsdokumentes wird ebenfalls durch die Menüsteuerung „File → Speicher unter" unterstützt. Maple-Dateien erhalten typischerweise die Endung `.mws`. Zum Weiterbearbeiten und Zurückgreifen auf bereits editierte Ausdrücke müssen nach dem Öffnen des gespeicherten Arbeitsdokuments alle Befehle erneut ausgeführt werden, da sie ansonsten nicht mehr im Arbeitsspeicher vorliegen. Durch Betätigen des Menüknopfes mit dem Symbol der drei Ausrufezeichen '`!!!`' lässt sich das gesamte Worksheet ausführen.

19.1.2 Hilfefunktion

Durch die Eingabe eines Fragezeichens gefolgt von einem Befehlsnamen (mit oder ohne Semikolon-Endung)

```
> ?name
```

wird zu diesem Befehl eine Hilfsinformation aus dem Maple-Menü aufgerufen.

Diese Hilfsinformation enthält eine ausführliche Beschreibung des erfragten Befehls, der zu verwendenden Syntax, Querverweise zu verwandten Befehlen, Beispielanwendungen und andere Hilfsinformationen des Schlagwortregisters. Zusätzlich können Hilfen mit den Befehlen

```
?index, ?library, ?package, ?expressions
```

aufgerufen werden.

19.1.3 Modularisierung durch Pakete

Ein Paket (Package) ist eine Sammlung verwandter Funktionen, die mit dem Befehl

```
> with(package);
```

in eine aktuelle Sitzung eingelesen werden.

Die standardisierten Packages enthalten zahlreiche Operationen und Algorithmen in linearer Algebra, Kombinatorik, Zahlentheorie, Statistik, Grafik usw. Eine Übersicht über die verfügbaren Pakete kann mit `?package` erhalten werden.

Die durch Laden des Paketes zusätzlich bereitgestellten Funktionen werden angezeigt. Sehr umfangreiche Pakete sind zur besseren Handhabung in Unterpakete (Subpackages) aufgeteilt und enthalten dann kleinere Gruppen sachlich zueinander gehörender Funktionsgruppen.

19.2 Einfache Berechnungen

Maple verwendet die in Programmiersprachen übliche Schreibweise für die mathematischen Symbole und Operatoren.

Standardoperatoren sind: + Addition, − Subtraktion, * Multiplikation, / Division, ! Fakultät, ∧ oder ** Potenzbildung, < kleiner als, <= kleiner gleich, > größer als, >= größer gleich, = gleich, <> ungleich.

Durch `evalf` wird das Ergebnis als Dezimalzahl (Float-Zahl) angezeigt. Die Default-Einstellung von Maple ist eine Genauigkeit von 10 Stellen.

Beispiel: Der dezimale Wert von π ergibt sich durch

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

Die Rechengenauigkeit kann mit dem Befehl `> Digits := n;` auf n Stellen festgelegt werden.

Die Funktion `convert` kann zum Konvertieren eines Ausdrucks von einer in eine andere Form verwendet werden. Hierbei sind Datentyp- oder Funktionsumwandlungen möglich.

19.2.1 Mathematische Formeln

Mathematische Formeln können mit `simplify` vereinfacht, mit `expand` expandiert und mit `factor` faktorisiert werden.

Mit `subs` erfolgt das Einsetzen eines Zahlenwertes in eine Formel bzw. das Auswerten eines Ausdrucks an einer Stelle x_0.

In der folgenden Tabelle werden einige Beispiele für Rechenoperationen und Funktionsausdrücke, die in der Maple-Bibliothek vorhanden sind, vorgestellt.

Tabelle 19.1: Maple-Befehle für ausgewählte mathematische Funktionen und Operationen

Ausdruck	Befehl	Bedeutung
$\|x\|$	`abs(x);`	Absolutbetrag
$\sqrt{x}$	`sqrt(x);`	Quadratwurzel
$\sqrt[n]{x}$	`root(x,n);`	n-te Wurzel aus x
$\exp(x)$	`exp(x);`	Exponentialfunktion
$\log(x)$, $\ln(x)$	`log(x); ln(x);`	Natürlicher Logarithmus

Ausdruck	Befehl	Bedeutung
$\sin(x)$, $\cos(x)$, $\tan(x)$	`sin(x); cos(x); tan(x);`	Trigonometrische Funktionen
$\sinh(x)$, $\cosh(x)$, $\tanh(x)$	`sinh(x); cosh(x); tanh(x);`	Hyperbolische Funktionen
$\sum_{k=1}^{n} a_k$	`sum(a(k), k=1..n);`	Summe von 1 bis `n`
$\prod_{k=1}^{n} a_k$	`product(a(k), k=1..n);`	Produkt von 1 bis `n`
$\lim_{k\to n} a_k$	`limit(a(k),k=n);`	Grenzwert einer Zahlenfolge für `k` gegen `n`
$\frac{d}{dx} f(x)$	`diff(f,x);`	Ableitung einer Funktion nach `x`
$\int_a^b f(x)\,dx$	`int(f,x=a..b);`	Integral über eine Funktion von `a` nach `b`
$J(z,x)$	`BesselJ(z, x);`	Besselfunktion
$\mathrm{Erf}(x)$	`erf(x);`	Fehlerfunktion $\frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt$

Maple folgt den Standardregeln für Priorität und Assoziativität von Operatoren, d.h. Potenzen werden zuerst, dann Multiplikationen und Divisionen und zuletzt Additionen und Subtraktionen ausgeführt. Negationen und negative Exponenten stehen in runden Klammern.

Datentypen. Die gebräuchlichsten Zahlentypen sind ganze Zahlen (`integer`), rationale Zahlen (`rational`), Fließkommazahlen (`float`) sowie ergänzend komplexe Zahlen (`complex`). Durch die Angabe eines Dezimalpunktes wird der Typ festgelegt. Die Zahl 5 ist vom Typ `integer`, während die Zahl 5.0 vom Typ `float` ist. Durch `type` lässt sich der Zahlentyp abfragen.

Konstanten. Maple kennt einige mathematische Konstanten, die in der Folge namens `constants` gespeichert sind. Konstanten sind ebenfalls die ogische Werte `true`, `false` und Unendlich ∞ `infinity`.

Tabelle 19.2: Mathematische Konstanten in Maple

Mathematische Konstante	Maple-Aufruf	Wert
Kreiszahl π	`Pi`	3.141592654
Euler'sche Zahl e	`exp(1)`	2.718281828
Catalans-Zahl C	`Catalan`	0.9159655942
Euler-Mascheroni-Konstante	`gamma`	.5772156649

Variablen. Gültige Variablennamen sind einfache Zeichenketten, die mit einem Buchstaben beginnen.

Mit dem Zuweisungsoperator ':=' wird einer Variablen ein Ausdruck zugewiesen:

```
> name :=  ausdruck;
```

Durch den Aufruf des gesetzten Namens kann auf den Ausdruck an einer späteren Stelle im Maple-Programm zurückgegriffen werden. Durch Einbetten eines Ausdrucks in Hochkommata `'... '` wird seine Auswertung verhindert und nur eine triviale Vereinfachung zugelassen.

Eine Variable kann durch erneute Zuweisung ihres eigenen Variablennamens und durch das Setzen von Anführungszeichen auf der rechten Seite der Gleichung `> a:='a';` wieder freigegeben werden. Durch das Einladen von `unassign` aus der Programmbibliothek lassen sich mehrere in Hochkomma gestellte Variablen auf einmal von ihren zugewiesenen Werten freisetzen.

19.3 Funktionen

Beim Starten einer Maple-Sitzung werden einige grundlegende Funktionen wie z.B. `op, type, evalf` und mathematische Befehle wie z.B. `max` und `diff` mit dem Hauptprogramm direkt in den Kernel übersetzt und stehen deshalb dem Benutzer sofort zur Verfügung. Die meisten der gebräuchlichen mathematischen Befehle in Maple wie `int, solve, plot, exp` werden nach der ersten Benutzung als Standard-Bibliotheksfunktionen automatisch geladen. Ihr Quellcode ist für den Anwender anschaubar durch die Eingabe:

```
> interface(verboseproc = 2); print (int);
```

19.3.1 Eigene Funktionen

Maple bietet zwei Ansätze zum Schreiben einer eigenen Funktion und zu deren Berechnung an.

Eine mathematische Funktion kann durch Benutzung des Zuweisungs- oder Pfeiloperators -> oder durch den `unapply`-Befehl definiert werden:

```
> f := x -> f(x);
> f := unapply(f(x),x);
```

Auch die Definition von Funktionen mit zwei oder mehr Variablen ist mit Hilfe des Pfeiloperators -> möglich.

Mathematische Ausdrücke können durch Verkettung der Zuweisungsoperation und der Funktionsdefinition in eine Funktion umgewandelt werden.

Stückweise definierte Funktionen können über `piecewise` oder über eine selbst geschriebene Prozedur mit der `proc`-Konstruktion festgelegt werden.

19.3.2 Grafische Darstellung von Funktionen

Maple ist sehr leistungsfähig bei der Modellierung von Kurven, Flächen, 3-D-Körpern und bei der Erstellung von Diagrammen. Es können Funktionen einer oder mehrerer Variablen grafisch dargestellt werden. Zur Abbildung von Grafen in zwei und drei Dimensionen sind die Anweisungen `plot`, `plot3D`, `smartplot`, `smartplot3D` und zahlreiche andere Befehle aus dem `Plot`-Paket von Maple wie z.B. `logplot` oder `semilogplot` vorgesehen.

Das `Plot`-Paket wird mit `with(plots);` aktiviert. Dieser Aufruf listet alle Plot-Befehle des Pakets auf. Bei den erzeugten grafischen Darstellungen können Achsenbeschriftungen, Linienbreite, das Styling, Drehungen, Orientierung und Projektion der Fläche, Beleuchtung und Schattierung verändert werden.

Die Darstellung von Funktionen in einer Variablen erfolgt mit dem Befehl

```
> plot(f(x), x= a..b, opt);
```

mit den Argumenten `f(x)` als Funktionsausdruck, `x=a..b` für den Bereich der Variablen und `opt` als optionalem Parameter.

Die vollständige Liste der Optionen für diesen Befehlkann mit `?plot[options]` aufgerufen werden. Die Optionen werden durch `option = value` nach dem Funktionsausdruck und dem zu zeichnenden Intervall angegeben. Durch Anklicken des Schaubildes können an der Grafik nachträglich Einstellungen vorgenommen werden. Die Optionen erscheinen entweder durch einen Rechtsklick mit der Maus oder in der Menüleiste oberhalb des Arbeitsdokuments. Die Darstellung kann auf diese Weise in verschiedene Formate wie JPEG, EPS, GIF, BMP exportiert werden.

Beispiel:

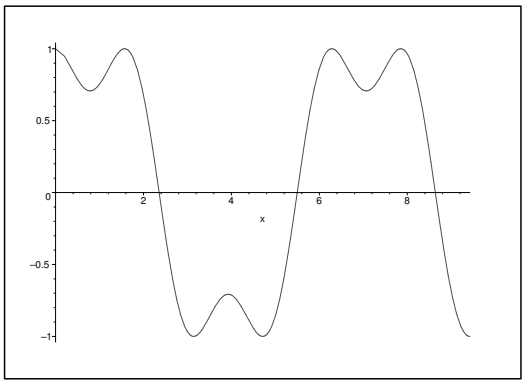
```
> plot(sin(x)^3 + cos(x)^3, x = 0 ..3*Pi);
```

Der Maple-Befehl zur Darstellung von Funktionen $f(x,y)$, die von zwei Variablen x und y abhängen, lautet

```
> plot3d( f(x,y), x=a..b, y=c..d, opt);
```

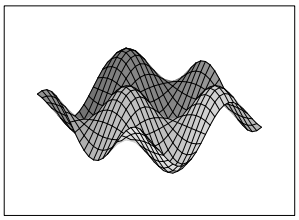
mit den Argumenten `f(x,y)` als Funktionsausdruck, `x=a..b`, `y=c..d` für den Bereich der x- und y-Variablen und `opt` als optionalem Parameter.

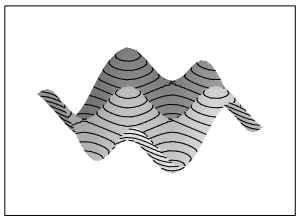
Auch hierzu gibt es eine Vielzahl von Optionen.



Beispiel:

```
>f(x,y) := sin(x)*cos(y);
>plot3d(f(x,y), x=-4..4, y=-4..4, style=patch);
 plot3d( f(x,y), x=-4..4,y=-4..4, style=patchcontour);
```





Neben der Visualisierung von 2-D- und 3-D-Funktionen enthält das Plot-Paket auch Befehle zur Animation von Funktionsplots.

19.3.3 Gleichungen

Lineare und nicht lineare Gleichungen und Gleichungssysteme sowie Ungleichungen können mit dem Maple-Befehl

```
> solve({eq_1,eq_2,...,eq_m}, {var_1,var_2,...,var_n});
```

gelöst werden.

Falls die Gleichung oder das Gleichungssystem eine mehrfache Nullstelle besitzt, wird diese entsprechend ihrer Vielfachheit mehrfach aufgelistet.

Falls eine exakte Lösung in einer algebraischen Form nicht ausgegeben werden kann, so kann entweder mit dem Befehl `evalf(%)` oder mit `fsolve(eq, var)` eine numerische Näherungslösung bestimmt werden.

Der Befehl `assign` weist das Lösungsergebnis den Variablen zu. Die Eingabe des Befehls `isolve` liefert als Ergebnis ganzzahlige Lösungen des Gleichungssystems. Mehrdeutige Lösungen, die von einem freien Parameter abhängen, werden durch `xi=xi` gekennzeichnet.

Um auf die benötigte Rechenzeit Zugriff zu haben, kann das Programm `showtime` eingelesen und mit `on` aktiviert werden. Neben dem Ergebnis werden die benötigte Zeit und der Speicherplatz angezeigt.

Enthält eine Gleichung mehrere Parameter, so ist es erforderlich, die Variable(n) anzugeben, nach der (bzw. denen) aufgelöst werden soll.

Beispiel:

```
> solve(x^2 + p*x + q = 0, x);
```

Nicht lineare Gleichungssysteme. Auch nicht lineare Gleichungssysteme können mit Maple gelöst werden.

Beispiel:

```
> solve({3*x^2 + 2*y + 1 = 0, x^2 + y^2 - 1 = 0},{x,y});
```

Dies liefert als Ergebnis

```
{x = RootOf(10_Z^2+9_Z^4-3),
 y = -1/2-3/2RootOf(10_Z^2+9_Z^4-3)^2}
```

Eine andere Darstellung kann mit der Anweisung

und damit die Lösungen in parametrisierter Form.

```
> allvalues(%);
```

eingeschaltet werden, sie löst die parametrisierte Darstellung auf und gibt die (im Beispiel) insgesamt vier möglichen Lösungen an.

Ungleichungen. Bei Ungleichungen gibt der `solve`-Befehl mit `RealRange` ein Lösungsintervall an.

Mit `Open(n)` wird ein offenes Intervall gekennzeichnet. Keine Angabe bedeutet, dass das Intervall geschlossen ist.

19.4 Programmieren in Maple

19.4.1 Maple und Programmiersprachen

Maple erlaubt das Konvertieren von Ausdrücken in die Syntax anderer Programmiersprachen. Eingebaut ist die Konversion nach LaTeX, durch Verwendung der Funktionen des `CodeGeneration`-Pakets kann Maple-Code auch in die Programmiersprachen C, FORTRAN, Java, Matlab, Visu-

alBasic übersetzt werden. Im Folgenden wird die Übersetzung am Beispiel von ANSI C vorgestellt. Eine Übertragung in die genannten anderen Programmiersprachen erfolgt analog.

Konvertierung nach LaTeX. Durch die Eingabe von `latex` kann ein Ausdruck automatisch in Latexformat umgewandelt und danach z.B. über die Zwischenablage in ein Latexdokument kopiert werden.

Beispiel:

```
> a:=Int(x^2,x=0..2);
> latex(a);
```

liefert das Ergebnis `\int _{0}^{2}\!{x}^{2}{dx}`

Konvertierung nach C. Durch die folgende Funktion lässt sich der Maple-Code in ANSI-C-Code umwandeln:

```
> C(x, opt);
```

Hierbei kann das erste Argument `x` ein Ausdruck, eine Liste, ein Feld, eine Prozedur etc. sein. Der zweite Parameter ist optional. Es können auch mehrere Optionen zur Codegenerierung angegeben werden. Eine Zusammenstellung der möglichen optionalen Parameter kann mit der Hilfe von `?CodeGeneration[Options]` aufgelistet werden.

19.4.2 Programmstrukturen in Maple

Bei der Benutzung von Maple als Programmiersprache ist es nicht erforderlich, jede benutzte Variable oder Funktion zu definieren. Maple bestimmt den Typ eines jeden eingegebenen Ausdrucks, der anschließend noch durch die Abfrage `type` überprüft und mit verschiedenen anderen Typen durch Eingabe des `convert`-Befehls konvertiert werden kann.

Bei der Entwicklung von Programmstrukturen in Maple wird eine Schleifenbildung mit `for` oder `while`, Verzweigungen werden mit `if` und Unterprogrammstrukturen mit `proc` erzielt.

Die Wiederholungsanweisung `for` oder `while` ermöglicht das wiederholte Ausführen einer Folge von Anweisungen, entweder für eine festgelegte Anzahl von Durchläufen (`for`-Schleife) oder bis eine bestimmte Bedingung erfüllt wird (`while`-Schleife).

for-Schleife. Eine `for`−Schleife hat die Syntax:

```
> for index from start by schritt to ende do:
    anweisungen
  end do;
```

Falls die Größen `start` und `schritt` nicht festgelegt sind, werden sie auf den Default-Wert 1 gesetzt. Der Abschluss `end do` ist auch durch `od` austauschbar. Beim Fehlen der Angabe `end do` ist die Schleife unendlich. Durch Betätigen des Stopp-Icons kann die Schleife unterbrochen werden.

while-Schleife. Eine `while`–Schleife hat die Syntax:

```
> while bedingung do anweisungen:
  end do;
```

seq-Funktion. Eine bestimmte Folgenvorschrift wird mit der Funktion `seq` erzeugt und ähnelt einer `for`-Schleife.

Beispiel: Durch Einsetzen der `seq`*-Anweisung wird für die Variable* `x` *eine Folge mit Exponenten von der 20. bis zur 27. Primzahl erzeugt:*

```
> seq (x^ithprime(n), n = 20..27);
```

if-Anweisung. Durch die `if-else`-Konstruktion wird eine Folge von Anweisungen in einem ausgewählten Bereich ausgeführt.

Eine `if-else`-Anweisung hat die Syntax:

```
> if bedingung_1 then anweisungen_1:
  elif bedingung_2 then anweisungen_2:
  ...
  elif bedingung_n then anweisungen_n:
  else anweisungen_(n+1):
  end if;
```

Die Ausdrücke `bedingung_i` sind Boolean-Werte und können durch Verwendung der Relationsoperatoren (`<, <=, >, >=, =, <>`), der logischen Operatoren (`And, Or, Not`) und der logischen Namen (`true, false, FAIL`) gebildet werden. Die Anweisungsfolge im Anschluss an den `else`-Befehl wird ausgeführt, falls alle anderen Bedingungen als falsch ausgewertet wurden.

proc-Konstruktion. Eine `proc`–Konstruktion hat die Syntax:

```
> proc()
  local var; <anweisungen>
  end;
```

Eine lokale Variable ist nur innerhalb des Prozedurkörpers, der durch die Statement-Sequence abgeschlossen ist, bekannt. Eine einmal deklarierte globale Variable ist auch weiterhin außerhalb des Prozedurkörpers für die Dauer der gesamten Maple-Sitzung bekannt. Wie das folgende Beispiel zeigt, kann auch eine Funktion mit Hilfe einer Prozedur definiert werden.

Beispiel: Definition von `f` *als Funktion der Variablen* `x`*.*

```
> f := proc (x)
> x^3 + x + 1
> end:
```

19.4.3 Programmieren eines Maple-Pakets

Im Folgenden soll ein Maple-Paket um zusätzliche Funktionen erweitern kann. Zunächst wird hierzu eine neue Funktion geschrieben. Um die De-

finition auch bei zukünftigen Anwendungen benutzen zu können, gibt es mehrere Möglichkeiten. Sie kann als normale Maple-Datei gespeichert werden und zu Beginn jeder Maple-Sitzung geöffnet werden. Dieses Vorgehen ist vor allem bei größeren Anwendungen unpraktisch, da es beim Speichern von Änderungen des Arbeitsdokuments versehentlich auch zu Änderungen in der Funktion kommen kann.

Eine neue Prozedur kann mit Hilfe des Befehls `save` zu einem so genannten `readlib`– Befehl definiert werden:

```
> save ‘‘.libname.‘/‘.Funktionsname.‘.m‘;
```

Bei dieser Zeichenkette gibt der `Funktionsname` den Namen der neu geschriebenen Prozedur an. `libname` ist ein globaler Name, der den Pfad zu den Maple-Systemdateien beinhaltet. In diesem wird eine neue Datei `Funktionsname.m` erzeugt, auf die man mit dem `readlib`-Befehl zurückgreifen kann. Zur Erläuterung der Zusammensetzung der Zeichenkette siehe auch Abschnitt 19.4.4.

19.4.4 Zeichenketten

In Maple besteht eine einfache Zeichenkette (simple string) aus einer Kombination von Buchstaben und Ziffern. Eine in einfache **rückwärtsgerichtete** Anführungszeichen (backquotes) gesetzte Zeichenkette `‘....‘` wird als quoted string bezeichnet.

Eine Zeichenkette kann Leerstellen, Sonderzeichen wie `+, /` und Steuerbefehle wie einen Zeilenumbruch enthalten. Die Kennzeichnung der leeren Zeichenkette ist `‘‘`. Zeichenketten werden durch den Operator '`.`' aneinandergehängt.

19.4.5 Bereiche, Listen und Mengen

Eine Bereichsangabe in Maple ist ein Ausdruck der Form `a..b`.

Die Bedeutung der Zahlen `a` und `b` hängt vom speziellen Kontext ab. In `sum`-Anweisungen steht `a..b` beispielsweise für den Bereich des Summationsindexes.

Listen. Eine durch eckige Klammern '`[...]`' eingeschlossene Folge von Ausdrücken ist eine Liste (list), die eine beliebige Kombination von Ausdrücken einschließlich anderer Listen enthalten kann, deren Reihenfolge von Maple nur durch explizite Aufforderung geändert wird. Eine leere Liste ist durch leere Klammern '`[]`' gekennzeichnet.

Der Befehl `op(ausdruck);` liefert die Folge aller in einer Liste vorkommenden Elemente, Operanden und Komponenten zurück. Diese `op`-Funktion, versehen mit zwei Argumenten, eignet sich aber auch, um bei

Bedarf auf ein ausgewähltes Element oder auf einen besonderen Bereich von Elementen aus einem Ausdruck oder aus einer Liste zuzugreifen:

```
> op(i,L);
> op(bereich,ausdruck);
```

Die Abfrage der Anzahl in einem Ausdruck enthaltener Operanden gelingt durch Eingabe des verwandten Befehls `nops(ausdruck);`.

Im Zusammenhang mit Listen ist die Benutzung einer `do`-Schleife in Kombination mit einer `for`-Anweisung erlaubt.

Eine einheitliche Verarbeitung aller Listenelemente gestattet die `convert`-Funktion, gefolgt von einem Unterbefehl wie z.B. der Addition oder Multiplikation aller Listenelemente mit `convert(list,'+');` bzw. `convert(list,'*');`.

Mengen. Eine Folge von in geschweifte Klammern '`{...}`' eingeschlossenen, voneinander verschiedenen Ausdrücken ist eine Menge. Im Unterschied zu einer Liste sind die Elemente einer Menge völlig ungeordnet, können aber mit Befehlen geordnet werden, wobei alle doppelt erscheinenden Elemente gestrichen werden. Das Symbol für die leere Menge ist '`{}`'.

Die in den vorangegangenen Abschnitten vorgestellten Befehle `op; nops; select; member; convert/'+'; convert/'*';` die `do`– und `for`–Schleife können völlig analog auch für Mengen eingesetzt werden. Spezielle Mengenoperatoren dagegen sind `intersect; union;` und `minus;`. Die Befehle `[op(menge)]` und `convert(menge,list)` wandeln eine Menge in eine Liste um, `{op(list)}` oder `convert(list,set)` formen eine Liste in eine Menge um.

19.4.6 Felder und Matrizen

Eine mehrdimensionale Datenstruktur ist ein Feld, das aus mehreren Reihen und Spalten mit Einträgen bestehen kann. Der Befehl `A := array(1..m, 1..n)` mit positiven ganzen Zahlen m und n legt ein zweidimensionales Feld `a[i,j]` mit m Zeilen und n Spalten an. Die Definition einer Matrix erfolgt zeilenweise, indem jede Zeile in Form einer Liste angegeben wird.

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

wird erzeugt durch die Eingabe von

```
> A := array(1..2,1..2):
> A[1,1] := x: A[1,2] := x^2: A[2,1] := 1/x:
> A[2,2] := 1-x^3:
```

Tabelle 19.3: Spezielle Arten von Matrizen durch Indexfunktionen

Funktion	Beschränkung
`symmetric`	$A[i,j] = A[j,i]$
`antisymmetric`	$A[i,j] = -A[j,i]$
`diagonal`	Elemente außerhalb der Diagonalen sind 0
`identity`	Diagonalelemente sind 1, alle anderen 0
`sparse`	Nicht explizit initialisierte Einträge werden als 0 vorausgesetzt

Beispiel: Die Erzeugung einer 3×3*-Einheitsmatrix in Maple erfolgt mit den nachstehenden Anweisungen:*

```
> Einheitsmatrix := array(identity, 1..3, 1..3):
> print (Einheitsmatrix);
```

Die `map`-Funktion kann zur Anwendung einer Operation auf jeden Feldeintrag genutzt werden. Die `op`-Funktion nimmt einen Operanden aus der Feldstruktur heraus.

Der Befehl `op(1, eval(A))` liefert die Indexfunktion, `op(2, eval(A))` die Einschränkung der Werte eines Feldes und der Befehl `op(3, eval(A))` die Einträge im `array`. Die Einträge im `array` bilden eine Liste von Gleichungen, die den expliziten Ausdrücken in der Feldfunktion entsprechen. Eine einfache Art, Felder und Matrizen zu definieren, liefern die eingebaute `matrix`– und ebenso die `vector`–Funktion zur linearen Algebra in dem `linalg`–Paket, das mit dem Befehl `with(linalg);` zu öffnen ist. Die Definition einiger Vektoren und der zugehörigen Matrix erfolgt zeilenweise in Form einer Liste und ist in dem nachstehenden Beispiel zu sehen.

$$M := \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 4 \\ 1 & 3 & 9 \end{bmatrix}$$

wird erzeugt durch die Eingabe von

```
> with(linalg):
> v1 := vector([1, 1, 1]):
> v2 := vector([1, 2, 4]):
> v3 := vector([1, 3, 9]):
>  M := matrix ([v1,v2,v3]);
```

Zur zweier Matrizen `M` und `N` wird die Syntax `multiply(M,N)` verwendet.

Weitere Operationen zum Rechnen mit Matrizen und Vektoren sind mit `?matrix` bzw. `?vector` dem Hilfepaket zu entnehmen.

19.4.7 Tabellen

Eine Tabelle (Table) ist in Maple eine Datenstruktur, die mit verschiedenen Einträgen beliebiger Ausdrücke erstellt ist und die keine spezielle Initialisierung benötigt. Die Tabelle entspricht damit einem assoziativen Array.

Beispiel: Die folgenden Anweisungen liefern das Ergbnis 90

```
> days[Jan] := 31:
> days[Feb] := 28:
> days[Mar] := 31:
> days[FirstQuarter]:=days[Jan]+days[Feb]+days[Mar];
```

Die komplette Tabelle kann mit dem `print`–Befehl ausgedruckt werden, die Reihenfolge des Ausdrucks ist dabei nicht mit drejenigen der Definition identisch und auch nicht alphabetisch etc. sortiert.

```
> print(days);
```

19.5 Ausgabe und Speicherung

Ausdrücke lassen sich mit dem `save`–Befehl in Dateien abspeichern, vgl. Abschnitt 19.4.3. Der `read`–Befehl ermöglicht das Einlesen von Definitionen und Anweisungen aus einem abgespeicherten Programm oder aus einer Datei.

Abspeichern in Dateien und Einlesen von Daten aus Dateien kann auch über die Maple-Icons erfolgen. Grafiken können durch Betätigen der rechten Maustaste als separate Bilder abgespeichert werden. Es gibt drei Arten von Maple-Ausgaben, die durch die `interface`-Variable `prettyprint` gesteuert werden. Ist `prettyprint` gleich 0, so wird die Ausgabe in einem linearen Format ausgegeben. Ergebnisse, die in diesem Format abgespeichert sind, können später als Eingabe weiterverwendet werden.

Beispiel:

```
> interface (prettyprint = 0):
> y := Int((sqrt(1+x^2),x));
```

Die Standardausgabe für diese Größe lautet

```
y := Int((1+x^2)^(1/2),x)
```

Hat die Variable von `prettyprint` den Wert 1, so erfolgt die Ausgabe über ein Mehrzeilenformat.

Der Ausdruck aus dem obigen Beispiel erscheint damit in folgender Darstellung:

```
                    /
                    |              2
```

```
y :=  |  sqrt(1 + x )  dx
      |
      /
```

Mit dem Befehl `prettyprint = 2` erfolgt der Ausdruck in der üblichen mathematischen Schreibweise mit Summen-, Integral- und Quadratwurzelzeichen, mit Matrixklammern und griechischen Buchstaben.

$$y := \int \sqrt{1+x^2}\,dx$$

Allgemeine Literatur

Burkhardt, W.: Erste Schritte mit Maple (Springer, 1996)

Devitt, J.S.: Calculus with Maple (Brooks/Cole, 1994)

Dodson, C., Gonzalez, E.: Experiments in Mathematics Using Maple (Springer, 1995)

Ellis, W. et al: Maple V Flight Manual (Brooks/Cole, 1996)

Heck, A.: Introduction to Maple (Springer, 1996)

Heinrich, E., Janetzko, H.D.: Das Maple Arbeitsbuch (Vieweg, Braunschweig 1995)

Kofler, M., Bitsch, G., Komma, M.: Maple (Einführung, Anwendung, Referenz) (Addison-Wesley, 2001)

Werner, W.: Mathematik lernen mit Maple (Band 1 + 2) (dpunkt.verlag, 1996 + 1998).

Westermann, T.: Mathematik für Ingenieure mit Maple (Band 1 + 2). (Springer, 2001 + 2002)

Westermann, T., Buhmann, W., Diemer, L., Endres, E., Laule, M., Wilke, G.: Mathematische Begriffe visualisiert mit Maple (Springer, 2001)

[MapleSoft] MapleSoft Website `http://www.maplesoft.com`

[MAS] Maple Application Center `http://www.mapleapps.com`