MAPLE , eine kleine Einführung für alle, die von DERIVE kommen

Maple und Mathematica sind die beiden stärksten Computer-Algebra-Systeme, die gegenwärtig existieren. Beide Programme werden laufend erweitert und verbessert. Sie sind, anders als DERIVE, "modular" aufgebaut. D.h., dass die Programme eigentlich Programmpakete aus Tausenden von Unterprogramme sind. Alles, was mathemtisch gelöst ist, ist in Unterprogrammen vorhanden. Die Handbücher "für den Anfang" umfassen 1500 Seiten. Usergroups im Internet erweitern die Möglichkeiten beständig. Um die Dimensionen zu illustrieren: Derive kann 1000 mal mehr als ein Schüler, aber Mathematica und Maple können 1.000.000 mal mehr.
Für die Schule ist ohne Belang, dass Maple viel mehr Mathematik kann als DERIVE, aber für die Hochschulen sind die mathematischen Möglichkeiten von Maple, die graphischen Leckerbissen und die Programmierbarkeit natürlich wesentliche Argumente.

Derive ist **termorientiert**: Term eingeben, z.B. eine Gleichung, dann Kommando anklicken:
Beispiel: **x^2+3*x+2=0** eintippen und auf "*Lösen*" klicken, fertig.
Maple, Mathematica und Mupad sind **kommandoorientiert:**
1. Kommando tippen oder anklicken, 2. Objekt angeben (die Funktion), 3. Optionen angeben.
Beispiel: **solve(x^2+3*x+2=0, x);** und dann Return.

Maple erwartet als Eingabe also stets ein Kommando für einen Term.
Die Kommandos müssen fast immer klein geschrieben werden, gewisse Konstanten wie I oder Pi aber groß. Andere gibt es doppelt, „diff“ und „Diff“ bedeuten z.B. Verschiedenes.
Ein Kommando muss mit einem Semikolon abgeschlossen werden. Nach einem einfachen RETURN wird das Kommando ausgeführt.
Bei einem Doppelpunkt anstelle des Semikolons wird der Befehl ausgeführt, das Ergebnis aber nicht angezeigt. (RETURN führt aus, SHIFT-RETURN erzeugt eine neue Zeile. Man löscht eine Zeile durch STRG-ENTF.)
Die Kommandos kann man nicht alle aus Menues heraus aufrufen. Man muss sie ggf. eintippen. Welche Befehle es gibt, erfährt man aus Handbüchern oder über die Hilfe. Im Kern gibt es 500 Befehle, von denen man die 20 wichtigsten schnell erlernt: *expand, factor, solve, diff, int* usw. In den Erweiterungsdateien, „packages“ genannt, findet man mindestens 5000 weitere Befehle.

1. Graphik

Es gibt im Prinzip nur zwei Befehle: **plot** und **plot 3D**, aber diese haben ungezählte Optionen und im Package „plots“ findet man 50 weitere Plot-Kommandos. Man kann mit Maple alles zeichnen, was irgendwie durch Funktionen bestimmt ist. Maple zeichnet normale Funktionen in 2D und 3D, parametrische, implizite, statistische, reine Punktmengen, Linienzüge, Ungleichungen usw. Vierzig verschiedene Koordinatensysteme stehen zur Verfügung. Alles läßt sich farblich verschieden gestalten, selbst die Beleuchtung und der Schattenwurf sind einstellbar und natürlich ist auch die Animation möglich. Beispiel:

- **with(plots): animate3d(sin(sqrt(x^2+y^2)+t),x=-6..6, y=-6..6, t=0..2*Pi-2*Pi/15, frames=15);**

Grundsätzlich kann man in Maple einen einfachen Plotbefehl geben wie:
- **plot(x^2, x=-2..2);** und kann danach alle besonderen Einstellungen aus Menues wählen.

Man kann die 3D-Graphiken auch leicht mit der Maus nach allen Seiten drehen und wenden.)
Anstelle der nachträglichen Experimente kann man dem Plot-Kommando auch alle Einstellungen bzgl. der Achsen, der Farben, des Blickwinkels usw. als Option mitgeben. Bei Mathematica ist das immer Pflicht, bei Maple hat man die Wahl zwischen vorher und nachher, was Schülern sehr entgegenkommt. Beispiel für die Mitgabe von Optionen:

- **with(plots): tubeplot([-t^1.5/5, 3*cos(t), 3*sin(t)], t=0..8*Pi, radius=t/8, scaling=constrained, orientation=[107,56], grid=[60,15]);**

2. Programmierung

Die Autoren verstehen Maple selbst eigentlich als Programmiersprache. D.h., dass es im Kern wenige Befehle gibt, mit denen alles andere aufgebaut wird. Sie können deshalb z.B. sehen, wie z.B. die Sinusfunktion in Maple definiert ist:

- **print(sin)** zeigt nur an, dass sin eine Prozedur ist. Man kann die „Anzeigetiefe“ erhöhen:
- **interface(verboseproc=2): print(sin);** zeigt die Definition von sin bis Tiefe 2.

Sie sehen damit gleichzeitig, wie ein Programm in Maple aufgebaut ist. Die Sprachelemente sind PASCAL oder C++ sehr ähnlich. Es gibt Funktionen, Prozeduren, Verzweigungen, Schleifen, 90 verschiedene Datentypen, Zugriff auf Dateien usw. Das kann z.B. für einen Physiker Sinn machen, der seine Meßdaten in besonderer Weise auswerten und aufbereiten lassen will. Für die Schule sehe ich nur wenige Programmieraufgaben, weil Maple in seinen Packages eigentlich schon alles liefert, wenn man es findet. Dass Maples Sprache recht verständlich ist zeigt ein Beispiel zur Berechnung von Fibonacci-Zahlen:

- **g:=proc(x::nonnegint) option remember;**
- **if x=0 then 0 elif x=1 then 1 else g(x-1)+g(x-2) fi end:**
- **seq(g(n), n=1..20);**
- **g(100);**

Besonders trickreich ist die Option REMEMBER. Die Rekursionstiefe wird für g(1000) auf den meisten Rechnern zu groß sein. Dann rechnet man erst g(500) und danach g(1000). Wegen REMEMBER hat sich Maple den Wert für 500 gemerkt und rechnet gleich von dort aus weiter.

3. Rechnen

Bitte geben Sie ein : „ 1/3 + 2*Pi; “ und danach RETURN.

Sie müssen den **Stern für die Multiplikation** stets eingeben, denn Maple erkennt z.B. „3x“ nicht als „3-mal-x“. Wie erwartet, gibt Maple von sich aus keine gerundeten Werte aus, sondern bleibt exakt. Geben Sie bitte ein:

evalf(“, 30);

Das bedeutet: Werte die letzte Antwort als Fließkommazahl mit 30 Nachkommastellen aus (evaluate-floating-point). Ohne Angabe der Stellenzahl werden 10 Stellen angezeigt.

Die **Anführungszeichen (“), (““) und (““““)** stehen für die letzte, vorletzte und vorvorletzte Antwort. Da die Zeilen nicht numeriert sind, kann man auf weiter zurückliegende Ergebnisse nur zugreifen, wenn man diese durch Definition in einer Variablen gespeichert hat.

4. Polynome umwandeln

Für das Rechnen mit Polynomen gibt es die Befehle factor, expand, simplify, combine usw. Bitte probieren Sie die folgenden Eingaben. (Semikolon am Ende und Abschicken mit RETURN.)

- **expand((a+b)^3);**
- **factor(“);**
- **factor(x^4 – 4*x^3 + 6*x^2 – 4*x + 1);**
- **combine(sqrt(a)*sqrt(b));** (zweite Umkehrung von expand, führt zusammen)
- **simplify(cos(3*x)+3*cos(x), trig);**

Da für SIMPLIFY nicht immer klar ist, was vereinfachen bedeuten soll, gibt es für simplify und combine viele Parameter, mit denen man das Verhalten steuern kann: Geben Sie bitte ein:

- **?simplify;** (Das vorangestellte Fragezeichen ruft die Erklärung zum Kommando auf.)

5. Gleichungen lösen

Das Kommando SOLVE arbeitet mit und ohne Angabe der Lösungsvariablen. Gibt man nicht an, wonach aufgelöst werden soll, sucht sich Maple etwas aus. Bitte probieren Sie:

- **solve(a*x^2 + b*x + c);**
- **solve(a*x^2 + b*x + c, x);**
- **solve(x^2+3*x+2=0,x);** (Gibt man „=0“ nicht an, setzt Maple selbst null.)
- **solve(x^3=8);** (Maple gibt immer auch die komplexen Lösungen an.)
- **solve(x^3=8.0);** (Erkennt Maple Kommazahlen im Term, werden auch solche ausgegeben.)
- **solve(sin(x)=0. 5);** (Näherungslösung)
- **solve(sin(x)=1/2);** (Exakte Lösung)
- **solve(sin(x)=x-1, x);** (Solve kann nicht alles lösen.)
- **fsolve(sin(x)=x-1,x);** (floating-solve gibt erste Näherungslösung an.)

6. Gleichungssysteme lösen

Der Zuweiser für Definitionen ist „**:=**“. Ein **Doppelpunkt** nach einer Anweisung bedeutet: Ausführen, aber nicht anzeigen. Für Listen müssen geschweifte Klammern verwendet werden. Bitte geben Sie ein:

- **Gls:={2*x+3*y+4*z=16, x+y+z=6, x-y-z=0};** (Doppelpunkt am Ende, danach weiterschreiben.)
- **solve(Gls);**
- **fsolve({x^2+y^2=13, x^y=7},{x,y});** (fsolve findet hier noch eine Näherungslösung.)
- **fsolve({x^2+y^2=13, x^y=8},{x,y});** (fsolve findet hier keine Lösung, obwohl {x=2, y=3} eine ist.)

7. Rechnen mit Matrizen und Vektoren: spezielle Operatoren!

Die Befehle für diesen Bereich befinden sich im Package „linalg“. Man aktiviert dieses durch den with-Befehl. Zum Rechnen muss man außer einem speziellen Kommando (evalm) zusätzlich noch spezielle Operatoren einsetzen:

- **with(linalg);**
- **M:=matrix([[1,2],[3,4]]), N:= matrix([[4,3],[2,1]]);**
- **inverse(M), det(M);** (Zwei Befehle in einer Zeile sind durch Komma zu trennen.)
- **evalm(M + N);** (Für die Matrizenoperationen muss Evaluate Matrices aufgerufen werden.)
- **evalm(M &* N);** (Für die M-Multiplikation ist zusätzlich ein eigener Operator definiert.)
- **spaltvek:=matrix([[a],[b]]);** (Ein Spaltenvektor ist eine Matrix.)
- **zeilvek:=vector([a,b]);** (Maple rechnet jedoch lieber mit Zeilenvektoren!)
- **dotprod(zeilvek,zeilvek);** (DotProd= Skalarprodukt)
- **crossprod(zeilvek,zeilvek);** (CrossProd= Kreuzprodukt)
- **evalm(M&*zeilvek);** **(**Matrix mal Zeilenvektor ergibt Zeilenvektor.)
- **evalm(M&*spaltvek);** (Ergebnis: Spaltenvektor)

8. Grenzwerte

- **limit((sqrt(1+x)-1)/x , x=0);**
- **sum(½^n, n=1..infinity);**
- **prod(1/n^2, n=1..infinity);**

9. Funktionen

Probleme können auftreten, wenn man in Maple eigene Funktionen definieren will, denn:

- **f(x):=x^2; wird angenommen, aber f(2) ergibt nur f(2) ohne Auswertung und nicht 4!**
Nur für f(x) wird x^2 ausgegeben.

- **f:= x^2; ist eine Alternative**. Dies ist jedoch eine sogenannte „Scheinfunktion“, d.h. x ist starr vorgeben. Immerhin gelingt die Auswertung mit **subs(x=3, f); .**

Ein „echte“ Funktion erhalten wir erst mit:
- **f:= x -> x^2;** Jetzt wird f(2) als 4 ausgegeben und f(a) als a^2.

Es geht aber noch weiter:
- **f:=x->x^2+sin(x);** ist eine korrekte Definition einer „echten“ Funktion.
- **g:=x->diff(f(x),x);** weist g die Ableitung von f zu**. Aber g(2); scheitert mit Fehlermeldung!**
-

Der Grund: Maple nimmt diff(f(x),x) nur symbolisch, wertet das nicht aus, um 2 einzusetzen. Lösung:
- **g:=unapply(diff(f(x),x),x);** Jetzt funktioniert g(2) usw.
-

Außerdem gibt es noch „anonyme“ Funktionen, durch Prozeduren definierte, rekursive, stückweise usw. Warum das? Maple will bei Rekursionen, Diffgleichungen und bei „höheren“ Programmen schnell sein. Darum wird zwischen Symbolen, Operatoren, echten, unechten Funktionen usw. unterschieden.

10. Analysis

Die Unterscheidungen bei den Funktionsdefinitionen hat natürlich Folgen für die Analysis. So gibt es vier verschiedene Befehle zum Ableiten einer Funktion: „diff“, „Diff“, „D“ und „implicitdiff“.

- **f:=x->x^3+x^2;**
- **diff(f,x);** Dies ergibt 0, weil das Symbol f nach x differenziert wird.
- **diff(f(x),x);** Dies ergibt die normale Ableitung, obwohl f und nicht f(x) definiert wurde.
- **Diff(f(x),x);** Mit großem D ergibt Diff nur die Schreibweise mit Differentialen.
- **D(f(x));** Das Ergebnis dürfte nicht befriedigen.
- **D(f);** Aber dieses bringt die Ableitung als Zuweisung! D leitet funktionale Operatoren ab.

Integrieren:

- **int(x^2+x^3,x);** Einfach **int** gibt die Stammfunktion, wenn sie existiert.
- **int(x^2+x^3,x=2..3);** Mit Angabe der Grenzen erzeugt man bestimmte Integrale.

Reihen:

- **series(sin(x), x=0, 10);**

Differentialgleichungen:

- **dsolve(diff(f(x), x) = f(x), f(x));**

Sonstiges:

Zuweisungen wie „x:=3“ bleiben erhalten, auch wenn sie vom Schirm gelöscht werden.
Aufheben mit: x:=‘x‘. (Gerade Apostrophe)
Maple unterscheidet linksgerichtete, rechtsgerichtete und gerade Apostrophe.

Mit Release 4 hat das Konzept gewechselt. Die Oberfläche ist seitdem ein „worksheet“, d.h. Formeln, Rechnung, Text und Graphik sind auf einem Blatt, ähnlich wie bei MathCad. Deshalb gibt es viele Optionen zum Einstellen des Aussehens (style).

Außerdem kann alles HTML-artig verlinkt werden, als LATEX-Datei abgespeichert werden oder in eine Mathematica-Datei überführt werden.

Selbstverständlich gibt es Newsgroups zu Maple.

Zusammenfassung

Maple ist außerordentlich mächtig und ist zusammen mit Mathematica das führende Mathe-Programm an den Hochschulen der ganzen Welt. Wer ein naturwissenschaftlich-technisches Fach studieren will, der ist gut beraten, wenn er sich in Maple schon vor dem Studium einarbeitet. Auch dann, wenn der Fachbereich, in dem man studiert, lieber MathLab oder MuPad einsetzt, wird sich der Maple-Kenner damit schnell zurechtfinden. DERIVE kann nicht soviel, wie die Hochschulprogramme, aber die Logik von Computer-Algebra-Systemen erlernt man auch damit.