Einige Bemerkungen zum Buch Westermann, Mathematische Probleme lösen mit MAPLE

Während die anderen Bücher des Autors, besonders seine beiden Bände „Mathematik für Ingenieure mit Maple“, eine klare Zielgruppe adressieren und in ihnen MAPLE ein Hilfsmittel bleibt, um Mathematik der Zielgruppe angemessen zu vermitteln, geht es im vorliegenden Buch vor allem um die Vermittlung von Fertigkeiten im Umgang mit MAPLE beim Lösen mathematischer Problemstellungen. Allerdings stellte sich mir beim Lesen immer wieder die Frage, ob man dieselbe Information in möglicherweise viel aktuellerer Form nicht auch direkt aus dem (inzwischen gut ausgebauten) Hilfesystem von MAPLE beziehen und sich somit die Kosten für das Buch sparen kann. So etwa lautet generell mein erster Rat, wenn mich Studenten nach „einführender Literatur“ zu einem speziellen Computeralgebrasystem (CAS) fragen. Für Westermanns Buch kommt hinzu, dass die Beispiele zum großen Teil auch nur Systembefehle erläutern und (sicher aus gutem Grund) genau wie im Hilfesystem aufgebaut sind. Das Buch hat natürlich zwei Besonderheiten im Vergleich zur Online-Hilfe aufzuweisen: es liegt (auch) gedruckt vor und es ist in deutscher Sprache geschrieben. Ob dies aber schon den Anschaffungspreis rechtfertigt?

Zielgruppe des Buches sind also wohl MAPLE-Einsteiger, welche sich von der Komplexität der MAPLE-Dokumentation (welche die Komplexität des Werkzeugs MAPLE widerspiegelt) erschlagen fühlen. Sie finden in Westermanns Buch in der Tat viele wichtige, darunter die meisten für Anfänger unentbehrlichen Kommandos, nach Sachgebieten geordnet und mit je einem Beispiel untersetzt. In der Themenwahl hält sich der Autor an seine anderen Bücher, womit – wenigstens für fortgeschrittenere Themen wie Fourierreihen, Extremwerte oder Differentialgleichungen – auch fortgeschrittenere Mathematik-Kenntnisse erforderlich sind, was den Adressatenkreis des Buches deutlich einschränkt. Ob es weiter gehende Überlegungen gibt, warum sich der Autor gerade für diese Themen (und damit gegen andere) entscheidet, wird nicht erläutert.

Überhaupt hält sich Westermann mit Begründungen über diesen oder jenen Zugang, mit Vergleichen zwischen verschiedenen Alternativen oder Ausführungen konzeptionellen Charakters – alles, was einem Anfänger **auch** wichtig zu erfahren wäre – sehr zurück. So wie angegeben geht es eben und fertig. Hier gibt es deutlich bessere Einführungen in Maple, sowohl in deutscher (etwa Walz: Maple 7. Rechnen und Programmieren; Kofler, Blitsch, Komma: Maple – Einführung, Anwendung, Referenz) als auch in englischer Sprache. In Englisch gibt es eine solche Menge von einführender Literatur, dass ich aufgehört habe, diese überhaupt vollständig zu registrieren; ich nenne nur zwei gute, mehrfach aufgelegte Bände aus dem Hause Springer: A. Hecks „Introduction to Maple“ und R. Corless' „Essential Maple 7“. Aber wahrscheinlich ist der deutschsprachige Markt zu klein für eine Übersetzung eines dieser Bücher.

Wenn schon solcherart „philosophische“ Bemerkungen im vorliegenden Buch fehlen, sind dann wenigstens die Beispiele selbst „lehrbuchhaft“, setzen sie die verfügbaren Mittel nicht „irgendwie“, sondern auf angemessene Weise ein und leben „guten Stil“ vor, dem ein Neueinsteiger nacheifern sollte? Schlechter Programmierstil ist nicht nur im symbolischen Rechnen unter Studenten weit verbreitet und schwer auszumerzen, wenn er sich einmal eingeschlichen hat.

Leider ist auch diese Frage aus meiner Sicht zu verneinen. Ich möchte dies exemplarisch an einigen Beispielen erläutern.

An einer ganzen Reihe von Textstellen kommt der Begriff „inerte Form“ (das erste Mal bereits auf Seite 2) vor, ohne dass der Leser je genauer erfährt, warum eine solche „Dopplung“ von Funktionen manchmal erforderlich ist und was sie genau leistet. Die Möglichkeit, Auswertungen und Zuweisungen früh oder spät vornehmen zu können, gehört aber zu den zentralen Charakteristika des symbolischen Rechnens, in welchem es sich wesentlich von klassischen Programmiersprachen unterscheidet und wofür Neueinsteiger rechtzeitig das richtige Gefühl entwickeln sollten. MAPLE braucht inerte Funktionen, weil es weder hold-Operator (MuPAD) noch späte Auswertung von Funktionen (Mathematica) kennt – also hier eigentlich einen Designfehler aufweist.

Aus demselben Grund gibt übrigens `solve` in jedem vernünftigen CAS sein Ergebnis als Substituti-

onsliste, d.h. als *potenzielle* Liste von Zuweisungen (in MAPLE leider nur für Gleichungs*systeme*), zurück und führt die Zuweisungen nicht bereits (global) aus. Westermann (S. 10) beschneidet die eigenen Ausdrucksmöglichkeiten, wenn er dies mit `assign` sofort nachholt. Abgesehen davon, dass `assign` bei mehrelementigen Lösungsmengen gar nicht eingesetzt werden kann oder in komplizierterem Kontext mysteriöse Ergebnisse liefert (Funktionen als Ergebnis, S. 79 – leider steht da weniger auf der CD als im Buch; warum das so wie beschrieben nicht geht, kann man bei [Corless] im ersten Kapitel nachlesen), braucht man solche globalen Zuweisungen gar nicht, weil mit dem Substitutionsoperator *lokal* Werte zugewiesen werden können. Statt

```
sol:=solve(...,{x}); assign(%); x;
```

also angemessener `x0:=subs(sol,x)` oder bei mehrelementiger Lösungsmenge `sol`

```
map(u->subs(u,<expression>),sol);
```

Überhaupt erscheint die Auswahl der Konzepte an vielen Stellen willkürlich. Warum muss der Leser (S. 3) mehr über `normal` wissen, wo doch `simplify` dasselbe (?) und mehr macht (und was ist neben `expand`, S. 4, mit `collect`, `combine`, `convert` etc.)? Wie bekommt man denn nun *gezielte* Umformungen hin? Antwort: für spezielle Funktionenklassen mit Erweiterungen von `simplify`, etwa `simplify(<expression>, trig)` usw., für allgemeine Ausdrücke nur über extreme Umwege oder gar nicht, denn MAPLE kennt keine Regelsysteme. Wäre ersteres aber nicht ein wichtiges Thema für ein Einsteigerbuch?

Auch hält es Westermann offensichtlich für erforderlich, den Anfänger mit Programmablaufstrukturen weitgehend zu verschonen. Im Kap. 24 fristen sie ein (konzeptionell) randständiges Dasein und im Text wird mehrfach von „Maple-Befehlsfolgen“ gesprochen, wo sich die vorgestellte Algorithmik sinnvoll in eine Prozedur gießen ließe. An einigen Stellen kommen verschämt (etwa S. 40) durch den Autor geschriebene „externe Prozeduren“ ins Spiel, die generell in den Arbeitsblättern auf der CD erst *nach* ihrem Aufruf definiert werden (Natürlich *definiert* und nicht „vor dem erstmaligen Aufruf ausgeführt“). Der Leser wird wenigstens jeweils mit einem fetten Hinweis auf diesen unsinnigen Umstand aufmerksam gemacht.

An den wenigen Stellen, wo Programmablaufstrukturen und Datenaggregationen gar nicht umgangen werden können, vermittelt der Autor den Eindruck, dass er darüber selbst keine klaren Vorstellungen hat. Betrachten wir exemplarisch Abschnitt 13.1: Es geht darum, eine **Folge** $a_n = \ldots$ zu definieren. Mathematisch ist eine Folge bekanntlich eine (besonders geschriebene) **Funktion** $a : \mathbf{N} \to \mathbf{R}$. Entsprechend wäre es logisch, a auch als *Funktion*

```
a:=n -> 1/T*int(f*cos(2*n*Pi/T*t),t=0..T);
```

zu vereinbaren, wobei unter der vorher zu treffenden Annahme `assume(T>0)` der Ausdruck f auch als

```
f:=piecewise(0<=t and t<=T/4, t, T/4<=t and t<=T, T/3-t/3);
```

vereinbart werden kann. Die einzelnen Folgenglieder lassen sich dann über `seq(a(i),i=0..10)` und die ganze Summe als

```
a(0) + 2*add(a(n)*cos(2*n*Pi/T*t),n=1..N);
```

oder auch

```
add(a(n)*cos(2*n*Pi/T*t),n=-N..N);
```

berechnen. Die von Westermann vorgeschlagene Lösung `a[n]:=...` orientiert sich nicht an inhaltlichen Aspekten, sondern an einem äußerlichen Phänomen – der Darstellung von indizierten Variablen in MAPLE. Allerdings kennt MAPLE mit der Variablen a_n die Variablen $a_2, a_3, ...$

noch nicht (und kann sie aus $a[n]$ auch nur durch Substitution erzeugen): a wird als Datenstruktur `table` gespeichert mit einem einzigen (!) Tabelleneintrag, für welchen zwar `sum(a[n] ... , n=1..N)` die Summe zufällig richtig berechnet, wo aber das für festes N angemessenere `add(a[n] ... , n=1..N)` bereits das beschriebene Dilemma offenbart. Wohltuend anders auch hier die entsprechenden Ausführungen zum selben Thema bei [Corless].

Schließlich sind die Beispiele an einigen Stellen bei den syntaktischen Möglichkeiten früher Maple-Versionen stehen geblieben. So wird für Textausgaben konsequent der „Symbolzugang“, etwa

```
print(‘Die Iterationslösung nach ‘,i,‘ Iterationen ist ‘,x[i]);
```

verwendet mit unbefriedigendem Ausgabeformat, das einmal dem Fehlen eines Datentyps `String` geschuldet war. Die gibt es aber seit Maple 5 und deshalb sollte Westermann angemessener

```
printf("Die Iterationslösung nach %1d Iterationen ist %a.\n",i,x[i]);
```

verwenden.

Ähnliches gilt für die Vektoren und Matrizen im Kapitel 20, für die Maple ebenfalls schon seit einiger Zeit mit `rtables` ein neues und konsistentes Bezeichnungskonzept eingeführt hat, das ohne das lästige `evalm` auskommt. Das neue Package `LinearAlgebra` wird wohl in absehbarer Zeit das alte `linalg`-Package ablösen, so dass diese Änderungen in den entsprechenden Abschnitten des Buches eingearbeitet werden sollten. Needless to say: Auch hier ist [Corless] vorbildlich.

Eine letzte Bemerkung: MAPLE gibt es nicht nur unter Windows, sondern für viele verschiedene Plattformen. Überdies ist die Bedienung weitgehend einheitlich, so dass ich die Beschränkung sowohl in der Einführung als auch auf der CD auf die Windowsplattform nicht nachvollziehen kann. Natürlich wird sich die Mehrzahl unserer zahlreichen studentischen Linux-Nutzer von solchen Bemerkungen nicht abschrecken lassen, auch wenn sie auf der CD mit einem separaten Verzeichnis abgespeist werden.

Weiterhin wäre es wünschenswert (und in gewissem Sinne auch ehrlich), nicht nur und erst auf S. 138, sondern auch in der Einleitung und im Klappentext auf der dritten Umschlagseite darauf hinzuweisen, dass MAPLE **nicht** im Lieferumfang des Buches (auch nicht als Reader oder in sonstwie abgespeckter Form) enthalten ist.

Dr. Hans-Gert Gräbe Leipzig, 24. März 2003