

Inhaltsverzeichnis

Vorwort	v
1. Einleitung	1
1.1 Numerische Berechnungen	1
1.2 Computeralgebra	1
1.3 Eigenschaften von Computeralgebra-Systemen.....	3
1.4 Existierende Systeme	3
1.5 MuPAD	4
2. Erste Schritte mit MuPAD	7
2.1 Erklärungen und Hilfe	9
2.2 Das Rechnen mit Zahlen	11
2.2.1 Exakte Berechnungen	13
2.2.2 Numerische Näherungen	14
2.2.3 Komplexe Zahlen	16
2.3 Symbolisches Rechnen	17
2.3.1 Einfache Beispiele	17
2.3.2 Eine Kurvendiskussion	27
2.3.3 Elementare Zahlentheorie	30
3. Die MuPAD-Bibliotheken	37
3.1 Informationen über eine Bibliothek	38
3.2 Das Exportieren von Bibliotheken	39
3.3 Die Standard-Bibliothek	41
4. MuPAD-Objekte	43
4.1 Operanden: Die Funktionen <code>op</code> und <code>nops</code>	44
4.2 Zahlen	47
4.3 Bezeichner	49
4.4 Symbolische Ausdrücke	53
4.4.1 Operatoren	53
4.4.2 Darstellungsbäume	59
4.4.3 Operanden	61

4.5	Folgen	63
4.6	Listen	67
4.7	Mengen	74
4.8	Tabellen	78
4.9	Felder	81
4.10	Logische Ausdrücke	84
4.11	Zeichenketten	85
4.12	Funktionen	87
4.13	Reihenentwicklungen	91
4.14	Algebraische Strukturen: Körper, Ringe usw.	94
4.15	Vektoren und Matrizen	98
	4.15.1 Definition von Matrizen und Vektoren	99
	4.15.2 Rechnen mit Matrizen	105
	4.15.3 Methoden für Matrizen	107
	4.15.4 Die Bibliotheken <code>linalg</code> und <code>numeric</code>	110
	4.15.5 Dünnbesetzte Matrizen	112
	4.15.6 Eine Anwendung	114
4.16	Polynome	116
	4.16.1 Definition von Polynomen	116
	4.16.2 Rechnen mit Polynomen	120
4.17	Intervallarithmetik	125
4.18	Null-Objekte: <code>null()</code> , <code>NIL</code> , <code>FAIL</code> und <code>undefined</code>	130
5.	Auswertung und Vereinfachung	133
	5.1 Bezeichner und ihre Werte	133
	5.2 Vollständige, unvollständige, erzwungene Auswertung	135
	5.3 Automatische Vereinfachungen	141
6.	Substitution: <code>subs</code>, <code>subsex</code> und <code>subsop</code>	143
7.	Differenzieren und Integrieren	149
	7.1 Differenzieren	149
	7.2 Integrieren	151
8.	Das Lösen von Gleichungen: <code>solve</code>	155
	8.1 Polynomgleichungen	155
	8.2 Allgemeine Gleichungen	161
	8.3 Differential- und Rekurrenzgleichungen	163
9.	Manipulation von Ausdrücken	167
	9.1 Umformung von Ausdrücken	168
	9.2 Vereinfachung von Ausdrücken	176
	9.3 Annahmen über symbolische Bezeichner	180

10. Zufall und Wahrscheinlichkeit 189

11. Graphik 197

11.1 Einleitung 197

11.2 Elementare graphische Darstellung:

 Graphen von Funktionen 198

 11.2.1 2D-Funktionsgraphen: `plotfunc2d` 198

 11.2.2 3D-Funktionsgraphen: `plotfunc3d` 207

11.3 Graphik für Fortgeschrittene:

 Grundlagen und erste Beispiele 215

 11.3.1 Allgemeine Grundlagen 216

 11.3.2 Einige Beispiele 221

11.4 Das gesamte Bild: Graphische Bäume 225

11.5 Viewer, Browser und Inspektor: Interaktive Manipulation
von Graphiken 227

11.6 Graphische Primitive 230

11.7 Attribute 233

 11.7.1 Voreinstellungen 234

 11.7.2 Vererbung von Attributen 234

 11.7.3 Primitive, die spezielle Szene-Attribute verlangen:
 „Hints“ 238

 11.7.4 Die Hilfeseiten der Attribute 240

11.8 Farben 242

 11.8.1 RGB-Farben 243

 11.8.2 HSV-Farben 244

11.9 Animationen 245

 11.9.1 Erzeugen einfacher Animationen 245

 11.9.2 Abspielen von Animationen 250

 11.9.3 Anzahl von Einzelbildern und die Echtzeitspanne
 einer Animation 251

 11.9.4 Welche Attribute lassen sich animieren? 253

 11.9.5 Animationen für Fortgeschrittene:
 Das Synchronisationsmodell 254

 11.9.6 „Bild für Bild“-Animationen 257

 11.9.7 Beispiele 262

11.10 Gruppen von Primitiven 265

11.11 Transformationen 267

11.12 Legenden 269

11.13 Fonts (Schriftarten) 272

11.14 Speichern und Exportieren von Graphiken 273

 11.14.1 Interaktives Speichern und Exportieren 273

 11.14.2 Der Batch-Betrieb 274

11.15 Importieren von Graphiken 276

11.16	Kamera-Objekte in 3D	277
11.17	Seltsame Effekte bei 3D-Graphiken? Beschleunigtes OpenGL?	284
12.	Der „History“-Mechanismus	285
13.	Ein- und Ausgabe	289
13.1	Ausdrücke ausgeben	289
13.1.1	Ausdrücke auf dem Bildschirm ausgeben.....	289
13.1.2	Die Form der Ausgabe ändern.....	290
13.2	Dateien einlesen und beschreiben	292
13.2.1	Die Funktionen <code>write</code> und <code>read</code>	292
13.2.2	Eine MuPAD-Sitzung sichern	294
13.2.3	Daten aus einer Textdatei einlesen	295
14.	Nützliches	297
14.1	Eigene Voreinstellungen definieren	297
14.2	Informationen zu MuPAD-Algorithmen	300
14.3	Neuinitialisierung einer MuPAD-Sitzung	302
14.4	Kommandos auf Betriebssystemebene ausführen	302
15.	Typenbezeichner	303
15.1	Die Funktionen <code>type</code> und <code>testtype</code>	303
15.2	Bequeme Typentests: Die <code>Type</code> -Bibliothek	306
16.	Schleifen.....	309
17.	Verzweigungen: if-then-else und case	315
18.	MuPAD-Prozeduren	321
18.1	Prozeduren definieren	322
18.2	Der Rückgabewert einer Prozedur	323
18.3	Rückgabe symbolischer Prozeduraufrufe	325
18.4	Lokale und globale Variablen.....	326
18.5	Unterprozeduren	331
18.6	Gültigkeitsbereiche von Variablen.....	333
18.7	Typdeklaration.....	335
18.8	Prozeduren mit beliebig vielen Argumenten	336
18.9	Optionen: Die Remember-Tabelle	338
18.10	Die Eingabeparameter	342
18.11	Die Auswertung innerhalb von Prozeduren	343
18.12	Funktionsumgebungen	345
18.13	Ein Programmierbeispiel: Differentiation.....	351
18.14	Programmieraufgaben	354

A. Lösungen zu den Übungsaufgaben	357
B. Dokumentation und Literatur	401
C. Graphikgalerie.....	403
D. Hinweise zur Graphikgalerie	419
Index	421

1. Einleitung

Um den Begriff Computeralgebra zu erklären, möchten wir die Berechnungen in der Computeralgebra mit numerischen Rechnungen vergleichen. Beide werden durch einen Computer unterstützt, doch gibt es grundlegende Unterschiede, die wir im folgenden erläutern wollen.

1.1 Numerische Berechnungen

In numerischen Rechnungen wird ein mathematisches Problem näherungsweise gelöst, die Rechenschritte finden mit *Zahlen* statt. Diese Zahlen sind intern in *Gleitpunktdarstellung* gespeichert, wodurch arithmetische Operationen schnell ausgeführt werden können. Diese Darstellung hat allerdings den Nachteil, dass sowohl die Berechnungen als auch die Lösungen nicht exakt sind, da es u. A. durch Rundungen zu Fehlern kommt. Numerische Algorithmen sind in der Regel so konstruiert, dass sie möglichst schnell eine Näherungslösung liefern. Näherungen sind oftmals die einzige Möglichkeit, eine mathematische Aufgabenstellung zu bearbeiten, wenn nämlich eine exakte Lösung in geschlossener Form nicht existiert. Außerdem sind Näherungslösungen dort nützlich, wo exakte Resultate gar nicht benötigt werden (z. B. bei der Visualisierung).

1.2 Computeralgebra

Im Gegensatz zu numerischen Berechnungen mit Zahlen werden in der Computeralgebra *symbolische* Berechnungen durchgeführt, es handelt sich gemäß [Hec 93] um „*Berechnungen mit mathematischen Objekten*“. Ein *Objekt* kann z. B. eine Zahl, aber auch ein Polynom, eine Gleichung, ein Ausdruck oder eine Formel, eine Funktion, eine Gruppe, ein Ring oder ein beliebiges anderes mathematisches Objekt sein. Symbolische Berechnungen mit Zahlen werden

im Gegensatz zu numerischen Berechnungen immer *exakt* durchgeführt, da intern eine genaue Darstellung von beliebig langen ganzen und rationalen Zahlen verwendet wird. Man nennt solche exakten Berechnungen in der Computeralgebra *symbolische* und *algebraische* Berechnungen. In [Hec 93] wird dafür die folgende Definition gegeben:

1. „Symbolisch“ bedeutet, dass es das Ziel ist, eine möglichst geschlossene Form einer Lösung in einer guten (d. h. einfachen) symbolischen Darstellung zu finden.
2. „Algebraisch“ steht für *exakte* Berechnungen im Gegensatz zu den Näherungslösungen, die auf Gleitpunktarithmetik beruhen.

Manchmal wird Computeralgebra auch mit „symbolischer Manipulation“ oder „Formelmanipulation“ erklärt, da mit Formeln und Symbolen gerechnet wird. Beispiele dafür sind die symbolische Integration oder die Differentiation wie

$$\int x \, dx = \frac{x^2}{2}, \quad \int_1^4 x \, dx = \frac{15}{2}, \quad \frac{d}{dx} \ln \ln x = \frac{1}{x \ln x}$$

oder die Berechnung symbolischer Lösungen von Gleichungen. Als Beispiel sei hier die Gleichung $x^4 + px^2 + 1 = 0$ in x mit einem Parameter p betrachtet, die die Lösungsmenge

$$\left\{ \pm \frac{\sqrt{2} \cdot \sqrt{-p - \sqrt{p^2 - 4}}}{2}, \pm \frac{\sqrt{2} \cdot \sqrt{-p + \sqrt{p^2 - 4}}}{2} \right\}$$

besitzt. Für die symbolische Berechnung einer exakten Lösung wird fast immer mehr Rechenzeit und mehr Hauptspeicher benötigt als für die Berechnung einer numerischen Lösung. Aber eine symbolische Lösung ist exakt, allgemeiner und liefert meist weitere Informationen zu dem Problem und seiner Lösung. Betrachten wir z. B. die obige Lösungsformel, die eine Lösung der Gleichung für beliebige Werte des Parameters p liefert: Sie zeigt die funktionale Abhängigkeit von p . Damit kann beispielsweise ermittelt werden, wie empfindlich die Lösungen gegen Änderungen des Parameters sind.

Für spezielle Anwendungen sind Kombinationen von symbolischen und numerischen Methoden sinnvoll. Es gibt z. B. Algorithmen in der Computeralgebra, die von der effizienten Gleitpunktarithmetik der Hardware profitieren. Auf der anderen Seite kann es sinnvoll sein, ein Problem aus der Numerik zunächst symbolisch zu vereinfachen, bevor der eigentliche approximative Algorithmus angewendet wird.

1.3 Eigenschaften von Computeralgebra-Systemen

Die meisten der bekannten Computeralgebra-Systeme sind interaktiv zu benutzende Programmpakete: Der Benutzer gibt dem System eine Reihe von Formeln und Befehlen, die dann vom System bearbeitet (man sagt auch, *ausgewertet*) werden. Das System gibt anschließend eine Antwort zurück, die weiter manipuliert werden kann.

Zusätzlich zu exakten symbolischen Berechnungen können die meisten Computeralgebra-Systeme Lösungen auch numerisch approximieren. Dabei kann die Genauigkeit vom Benutzer auf eine beliebige Anzahl von Stellen vorgegeben werden. In MuPAD geschieht dies durch die globale Variable `DIGITS`. Beispielsweise wird mit dem einfachen Befehl `DIGITS:=100` erreicht, dass MuPAD Gleitpunktberechnungen mit einer Genauigkeit von 100 Dezimalstellen ausführt. Natürlich benötigen solche Berechnungen mehr Rechenzeit und mehr Hauptspeicher als das Benutzen der Gleitpunktarithmetik der Hardware.

Moderne Computeralgebra-Systeme stellen zusätzlich noch eine mächtige Programmiersprache¹ zur Verfügung und bieten Werkzeuge zur Visualisierung und Animation mathematischer Daten. Auch bieten viele Systeme die Möglichkeit zur Vorbereitung druckfertiger Dokumente (so genannte *Notebooks* oder *Worksheets*). Auch in MuPAD existiert ein Notebook-Konzept, welches in dieser Einführung allerdings nicht behandelt werden soll. Das Ziel dieses Buches ist es, eine Einführung in die Benutzung der *mathematischen* Fähigkeiten MuPADs zu geben.

1.4 Existierende Systeme

Es gibt viele verschiedene Computeralgebra-Systeme, von denen einige kommerziell vertrieben werden, während andere frei erhältlich sind.

So genannte *special purpose* Systeme dienen zur Behandlung von speziellen mathematischen Problemen. So gibt es das System *Schoonship* für Probleme in der Hochenergiephysik, *DELiA* zur Behandlung von Differentialgleichungen, *PARI* für Anwendungen in der Zahlentheorie² und *GAP* für Probleme aus der Gruppentheorie.

¹ Die Programmiersprache von MuPAD besitzt eine ähnliche Syntax wie Pascal, wobei Konstrukte für objektorientierte Programmierung zur Verfügung gestellt werden.

² Teile dieses Systems werden intern von MuPAD verwendet.

Daneben gibt es so genannte *general purpose* Computeralgebra-Systeme. Dazu gehören das seit 1980 entwickelte und speziell für Kleincomputer ausgelegte *Derive* sowie *MathView* (ehemals *Theorist*), das seit 1990 entwickelt wird und eine ausgefeilte Benutzungsoberfläche, aber nur eingeschränkte mathematische Fähigkeiten besitzt. Außerdem gibt es die Systeme *Macsyma* und *Reduce*, beide seit 1965 entwickelt und in LISP programmiert. *Maxima* ist eine Abspaltung aus dem originalen *Macsyma*, seit 1998 unter eigenem Namen vertrieben. Modernere Systeme wie *Mathematica* und *Maple* befinden sich seit etwa 1980 in Entwicklung und sind in C programmiert. *Mathematica* war das erste System mit einer benutzerfreundlichen Oberfläche. Weiterhin ist *Axiom* zu erwähnen, das ebenfalls seit etwa 1980 entwickelt wird. Im Gegensatz zu den bereits genannten Systemen verfügt *Axiom* über eine komplett typisierte Sprache und lässt Berechnungen nur in speziellen mathematischen Kontexten zu. Unter allen diesen Systemen ist MuPAD das jüngste: Es wird seit 1990 an der Universität Paderborn entwickelt und versucht, die Stärken verschiedener Vorläufer mit modernen, eigenen Konzepten zu verbinden.

1.5 MuPAD

Zusätzlich zu den bereits genannten Eigenschaften von Computeralgebra-Systemen hat MuPAD die folgenden Fähigkeiten (die in diesem Buch allerdings nicht näher behandelt werden):

- MuPAD bietet Sprachkonstrukte zum objektorientierten Programmieren. Man kann eigene Datentypen definieren und fast alle Operatoren und Funktionen zu deren Behandlung überladen.
- MuPAD stellt einen interaktiven Quellcode-Debugger zur Verfügung.
- Mit dem Modulkonzept kann man in C oder C++ geschriebene Programme zum MuPAD-Kern hinzufügen.

Das Herzstück von MuPAD ist der so genannte *Kern*, der aus Effizienzgründen im Wesentlichen in C und teilweise in C++ implementiert ist. Dieser Kern wiederum besteht aus den folgenden grundlegenden Teilen:

- Der so genannte *Parser* liest die Eingaben an das System und überprüft sie auf richtige Syntax. Eine fehlerfreie Eingabe wird vom Parser in einen MuPAD-Datentyp umgewandelt.

- Der so genannte *Auswerter* (*Evaluiierer*, englisch: *evaluator*) wertet die Eingaben aus und vereinfacht die Ergebnisse. Dieser Vorgang ist in MuPAD genau definiert und wird später näher erläutert.
- Die *Speicherverwaltung* ist für die interne Verwaltung der MuPAD-Objekte zuständig.
- Einige oft benötigte Algorithmen wie z. B. die arithmetischen Funktionen sind aus Effizienzgründen als *Kernfunktionen* auf C-Ebene implementiert.

Parser und Evaluierer definieren die MuPAD-Programmiersprache. Mit Hilfe dieser Sprache sind die zu MuPAD gehörenden Programmbibliotheken implementiert, die das mathematische Wissen des Systems enthalten.

Daneben besitzt MuPAD komfortable Benutzungsoberflächen zur Erzeugung so genannter *Notebooks* oder von Graphiken oder zum Debuggen in der MuPAD-Sprache geschriebener Programme. Das MuPAD-Hilfesystem hat Hypertextfunktionalität. Man kann in Dokumenten navigieren, aber auch Beispiele per Mausklick vom System berechnen lassen. Abbildung 1.1 zeigt die Hauptkomponenten des MuPAD-Systems.