Vorlesung MuPAD
Tag 1
Aufbau von MuPAD
Streifzug durch MuPAD

Stefan Wiedmann

Mathematisches Institut Universität Göttingen

16.01.2009

Organisatorisches

- Anmeldungen zu der Veranstaltung über StudIP
  `https://www.studip.uni-goettingen.de/`
  Einführung in MuPAD (Mathematische Anwendersysteme) (WS 2008/09)
- Alle Unterlagen können von der StudIP-Seite (Reiter Dateien) heruntergeladen werden
- Dozent: Stefan Wiedmann `wiedmann@uni-math.gwdg.de`
- Büro: Zi. 006 (Erdgeschoß), MI
- Telefon: 39-7754

Accounts

- MuPAD ist in Version 4.0 auf allen Rechnern der Mathematischen Fakultät installiert
- Kiosk-System im Übungssaal und im Computerraum – kein Login aber auch keine Speicherung von eigenen Dateien
- Accounts im CIP-Pools der Mathematischen Fakultät:
  - Mit Stud.it-Account: Formular unter `https://xldap.uni-math.gwdg.de/register.php` ausfüllen
  - Ohne Stud.it-Account: An Herrn Jochen Schulz wenden
- Mit Account: Möglichkeit via ssh in der Shell mupad zu starten:
  `ssh -X username@l1.num.math.uni-goettingen.de`
  `ssh -X username@s1.math.uni-goettingen.de`
- Alternative Rechner: `l2 - l14`, bzw. `s2 - s6`

Unterlagen

- Aufgabenblätter: StudIP
- Folien zur Vorlesung: StudIP
- Folien der Vorlesung zum Ausdrucken: StudIP
- Notebooks zur Vorlesung: StudIP
- MuPAD Buch: Preis: 19.95 Euro,
  Rapin, Wassong, Wiedmann, Koospal:
  MuPAD - Eine Einführung,
  Springer 2007, ISBN 978-3-540-73475-8

Ablauf der Veranstaltung

- Blockveranstaltung vom 16.2.-27.2.2008
- Vorlesung: 9.15 Uhr bis 11.30 Uhr
- Nachmittags: 4 Übungsgruppen à je 1h 15min
  - 13:00-14:15 (Tutor: Schulz)
  - 14:15-15:30 (Tutor: Wiedmann)
  - 15:30-16:45 (Tutor: Schulz)
  - 16:45-18:00 (Tutor: Wiedmann)
- Scheinerwerb
  - Regelmäßige Teilnahme an den Übungen
  - Vorführen von Aufgaben
  - Klausur am 27.2.2009; 9:15 – 10:45; Anmeldung über FlexNow!

Inhalt der Vorlesung

1. Tag Organisatorisches, Aufbau von MuPAD, Streifzug durch MuPAD
2. Tag Grundlagen MuPAD (grundlegende Datentypen, Ausdrücke), Lösen von Gleichungen, Symbolisches Rechnen
3. Tag Mengen, natürliche, rationale, reelle und komplexe Zahlen, Gleitkommazahlen, Ungleichungen
4. Tag Vektoren und Matrizen, Lineare Algebra in MuPAD, Programmieren I
5. Tag Datencontainer in MuPAD, Lineare Abbildungen und Matrizen
6. Tag Folgen und Reihen
7. Tag Reelle Funktionen, Grafiken
8. Tag Differenzial- und Integralrechnung
9. Tag Grundlagen der Programmierung, Zeichenketten (Strings)
10. Tag Klausur (9.15 - 10.45 Uhr)

Gliederung des heutigen Tages

- Organisatorisches
- Was ist MuPAD?
- Ein Streifzug durch MuPAD
  - Eine Kurvendiskussion
  - Symbolisches Rechnen
  - Etwas AGLA
  - Etwas Zahlentheorie
  - Nützliches und Hilfe

MuPAD

- MuPAD ≡ Multi Processing Algebra Data tool
- MuPAD ist ein Computeralgebra-System
  - Entwicklung von MuPAD seit 1990 an der Universität Paderborn
  - Seit 1997 Teilausgliederung in die SciFace Sofware GmbH
  - MuPAD wurde mitte des Jahres 2008 an Mathworks verkauft
  - MuPAD ist seither eine Toolbox des Programms Matlab (Symbolic Toolbox)
- MuPAD ist in C/C++ geschrieben.
- MuPAD ist objektorientiert.

Computeralgebra-Systeme

Computeralgebra

beschäftigt sich mit exakten Berechnungen von mathematischen Objekten

Mathematische Objekte

Natürliche Zahlen, reelle Zahlen, Polynome, Funktionen, Gruppen, Ringe, ...

Numerischen Berechnungen

Bei numerischen Rechnungen (z.B. Taschenrechner) benutzt man Zahlen in Gleitpunktdarstellung, also i.A. nur Näherungen an die gesuchte Lösung

Computeralgebra <-> Numerische Berechnung

Beispiel

Mathematische Objekte	π, $\sqrt{2}$
Gleitpunktdarstellung (8 Stellen)	3.1415927, 1.4142136

Andere Computeralgebra-Systeme

General purpose:	Derive (Eingestellt 2006, TI)
	LiveMath (Maple)
	Maxima (Free, GPL)
	Reduce (sehr alt, in Lisp programmiert, free)
	Mathematica (Platzhirsch)
	Maple (Platzhirsch)
	Matlab/Octave (Für große Rechnungen)
	Magma (Spezielle mathematische Rechungen)
Special purpose:	Cadabra (Körpertheorie)
	PARI/GP (Zahlentheorie)
	GAP (Gruppentheorie)
	Macaulay (Algebraische Geometrie)
	Singular (Algebraische Geometrie)

Neuere Entwicklungen

Neue Entwicklungen/Bibliotheken:

Sage Sehr ehrgeiziges Projekt, komplett in Phython geschrieben

SymPy Phython-Bibliotheken als CAS-Verwendbar

SymbolicC++ Bibliotheken zur CA in C++

Überblick:

`http://en.wikipedia.org/wiki/Comparison_of_computer_algebra_systems`

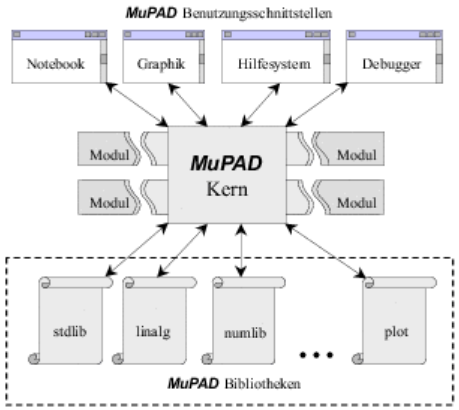
MuPAD - Stärken

- Objektorientiertes Konzept (definieren eigener Datentypen, überladen von Operatoren möglich)
- interaktiver Quellcode-Debugger
- umfangreiches Hilfesystem
- Einfaches Einbinden von C/C++ Routinen (dynamische Module)
- Teil einer großen Software (Weiterenwicklung gesichert)
- Viele freie (Unterrichts-)materialien im Netz

MuPAD - Schwächen

- Befehlsumfang nicht so mächtig wie bei Maple oder Mathematica
- Geringerer Verbreitungsgrad
- Benutzung ist nicht so intuitiv wie bei anderen Systemen
- Teil einer großen Software (Matlab muß installiert/erworben werden)
- Programmierumgebung wenig komfortabel (Es gibt in der Matlab-Version einen Editor)
- Gültigkeitsberereich von Variablen in Funktionen ist global
- Zum Teil nicht konsistent erscheinende Auswertungen von Variablen

Struktur von MuPAD



Kern von MuPAD

- Parser: Liest die Eingaben und überprüft die Syntax; Umwandlung in MuPAD-Datentyp
- Auswerter: Auswertung und Vereinfachung der Ergebnisse
- Speicherverwaltung: (MAMMUT ≡ Memory Allocation Managment Unit) interne Verwaltung der MuPAD-Objekte
- Kernfunktionen: Oft benötigte Funktionen werden aus Effizienzgründen im Kern auf C-Ebene implementiert.

Literatur

- K. Gehrs, F. Postel. MuPAD – Eine praktische Einführung. SciFace. 2001.
- Ch. Creutzig, J. Gerhard, W. Oevel, St. Wehmeier. Das MuPAD Tutorium. Springer. 2. Auflage. 2002.
- M. Majewski. MuPAD Pro Computing Essentials. Springer. 2002.
- Rolf Monnerjahn. Mathematische Anwendungen in Biologie, Chemie, Physik. MuPad im Mathematikunterricht: 5.-10. Schuljahr
- Gerd Rapin, Thomas Wassong, Stefan Wiedmann und Stefan Koospal. MuPAD: Eine Einführung

MuPAD als Taschenrechner

Hier einige Beispiele:

```
>> 3+4*10+12
```

```
  55
```

```
>> sin(PI)
```

```
  0
```

```
>> float(PI)
```

```
  3.141592654
```

```
>> float(sqrt(2))
```

```
  1.414213562
```


Kurvendiskussion I

Betrachte die durch die reelle Zahl a parametrisierte Funktionenschar:

$$f : x \quad \mapsto \quad \frac{2x^2 - 20x + 42}{x - 1} + a, \quad a \in \mathbb{R}$$

- Eingabe der Funktion

```
>> f:= x -> (2*x^2-20*x +42)/(x-1)+a
```

```
  x -> (2*x^2 - 20*x + 42)/(x - 1) + a
```

- Definitionslücken

```
>> Problemstellen:=discont(f(x),x)
```

```
  {1}
```


Kurvendiskussion II

- Pol ?

```
>> limit(f(x),x=1,Left)
```

```
  a - infinity
```

```
>> limit(f(x), x=1, Right)
```

```
  a + infinity
```

- Umformen

```
>> normal(f(x))
```

```
     2
  2 x  - 20 x - a + a x + 42
  --------------------------
             x - 1
```

Kurvendiskussion III

- Nullstellen

```
>> Nullstellen:=solve(f(x)=0,x)
```

```
{       2              1/2         2              1/2          }
{     (a  - 32 a + 64)       a  (a  - 32 a + 64)       a      }
{ 5 - ------------------ - -, ------------------ - - + 5 }
{              4             4           4             4      }
```

- Berechnen der Ableitung

```
>> f'(x)
```

```
                    2
     4 x - 20    2 x  - 20 x + 42
     -------- - -----------------
      x - 1                  2
                      (x - 1)
```


Kurvendiskussion IV

- Extremwerte

```
>> Extremstellen:=solve(f'(x)=0,x)
```

```
                    1/2      1/2
          {1 - 2 3    , 2 3    + 1}
```

- Lokale Minima und Maxima

```
>> float(f''(1-2*sqrt(3)))
```

```
  -1.154700538
```

```
>> float(f''(2*sqrt(3)+1))
```

```
  1.154700538
```


Kurvendiskussion V

- Verhalten von f für große x

```
>> limit(f(x),x=infinity); limit(f(x),x=-
   infinity)
```

```
  a+infinity
  a-infinity
```

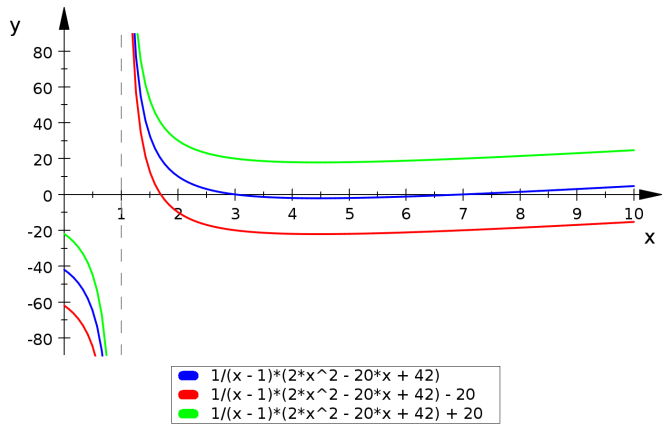
- Definiere f_0, f_1, f_2

```
>>  f0:=subs(f(x),a=0):
>>  f1:=subs(f(x),a=-20):
>>  f2:=subs(f(x),a=20): f0,f1,f2
```

```
    2                2                    2
  2 x  - 20 x + 42  2 x  - 20 x + 42     2 x  - 20 x + 42
  ----------------, ---------------- -20, ---------------- +20
       x - 1             x - 1                x - 1
```


Plot

```
>> plotfunc2d(f0,f1,f2,x=0..10)
```



Zusammenfassung I

- Definieren von Variablen mit ':=', z.B. `a:=3`
- Löschen von Objekten mit `delete`, z.B. `delete a`
- Definieren von Funktionen mit'->', z.B. `f:= x -> x^2- 6*x`
- Symbolisches Rechnen
  - Unstetigkeitsstellen: `discont(f(x),x)`
  - Grenzwertbestimmung: `limit(f(x), x= 2, Left)`
  - Vereinfachen: `normal(f(x))`
  - Bilden von Ableitungen `f'(x)`

Zusammenfassung II

- Lösen von Gleichungen: `solve( f(x)=0, x)`
- Berechnen numerischer Approximationen: `float(f(sqrt(3)+ 4))`
- Plotten einer Funktion: `plotfunc2d(sin(x),x=0..4)`

Symbolisches Rechnen I

- Integrieren von $\int_0^\infty x^4 e^{-x} dx$

```
>> int(x^(4)*exp(-x),x=0..infinity)
```

```
  24
```

- Stammfunktion von $\frac{1+\sin(x)}{1+\cos(x)}$

```
>> f := x -> (1+sin(x))/(1+cos(x)):
>> int(f(x),x)
```

```
    ln(2 cos(x) + 2) - sin(x) + cos(x) ln(2 cos(x) + 2)
  - ---------------------------------------------------
                        cos(x) + 1
```


Symbolisches Rechnen II

- Faktorisieren und Ausmultiplizieren

```
>> expand((x-1)*(x-2)*(x-3)*(x-4))
```

```
   4       3       2
  x  - 10 x  + 35 x  - 50 x + 24
```

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

```
  (x - 1) (x - 2) (x - 3) (x - 4)
```

- Sortieren eines Ausdrucks bezüglich einer Unbekannten

```
collect(x^2+2*x+b*x^2+sin(x)+a*x,x)
```

```
              2
  (b + 1) x  + (a + 2) x + sin(x)
```

Symbolisches Rechnen III

- Partialbruchzerlegung

```
>> partfrac( x^ 2/( x^ 2- 1))
```

```
      1            1
  --------- - --------- + 1
  2 (x - 1)    2 (x + 1)
```

- Vereinfachen von Ausdrücken ($\frac{e^x-1}{e^{(1/2)x}+1}$)

```
>> simplify((exp(x)-1)/(exp(x/2)+1))
```

```
     / x \
  exp| - | - 1
     \ 2 /
```


MuPAD unterscheidet strikt zwischen Funktionen und Ausdrücken I

Beispiele:

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

```
 x -> sin(x)
```

```
>> g:=sin(x)
```

```
  sin(x)
```

```
>> f(1),g(1)
```

```
 sin(1), sin(x)(1)
```


MuPAD unterscheidet strikt zwischen Funktionen und Ausdrücken II

```
>> int(f,x)
```

```
  Error: Illegal integrand [int]
```

```
>> int(f(x),x)
```

```
  -cos(x)
```

```
>> f(x)-g
```

```
  0
```

```
>> h:=fp::unapply(g)
```

```
  x -> sin(x)
```


Analytische Geometrie und Lineare Algebra

Berechnen des Schnittpunkts der Ebene

$$E : \vec{x} = \begin{pmatrix} 2 \\ 1 \\ -1 \end{pmatrix} + l \begin{pmatrix} 1 \\ -1 \\ -1 \end{pmatrix} + m \begin{pmatrix} -3 \\ 1 \\ 4 \end{pmatrix}, \quad l, m \in \mathbb{R}$$

mit der Geraden

$$g : \vec{x} = \begin{pmatrix} 3 \\ 0 \\ 1 \end{pmatrix} + k \begin{pmatrix} 4 \\ -1 \\ 2 \end{pmatrix}, \quad k \in \mathbb{R}$$

Grafische Darstellung

```
>> E1 := 2+l-3*m: E2:=1-l+m: E3:=-1-l+4*m:
```

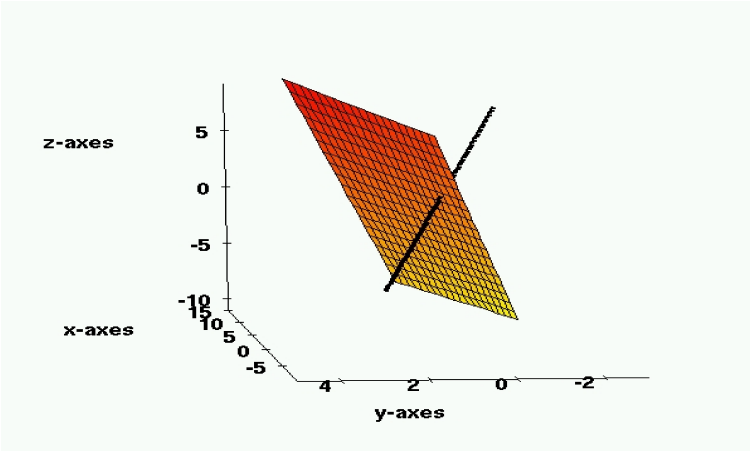
```
>> Ebene1 := plot::Surface([E1,E2,E3],
 l=-2..2,m=-2..2, Mesh=[20,20]):
```

```
>> g1 := 3+4*k: g2 := -k: g3 := 1+2*k:
```

```
>> Gerade1 := plot::Curve3d([g1, g2,
 g3], k=-3..3):
```

```
>> plot(Ebene1,Gerade1)
```


Grafische Darstellung



Analytische Lösung

Gleichsetzen ergibt:

$$\begin{pmatrix} 2 \\ 1 \\ -1 \end{pmatrix} + l \begin{pmatrix} 1 \\ -1 \\ -1 \end{pmatrix} + m \begin{pmatrix} -3 \\ 1 \\ 4 \end{pmatrix} = \begin{pmatrix} 3 \\ 0 \\ 1 \end{pmatrix} + k \begin{pmatrix} 4 \\ -1 \\ 2 \end{pmatrix}$$

oder

$$\underbrace{\begin{pmatrix} 1 & -3 & -4 \\ -1 & 1 & 1 \\ -1 & 4 & -2 \end{pmatrix}}_{=: A} \underbrace{\begin{pmatrix} l \\ m \\ k \end{pmatrix}}_{=: L} = \underbrace{\begin{pmatrix} 1 \\ -1 \\ 2 \end{pmatrix}}_{=: b}$$

oder $AL = b$.

Definieren und Lösen des LGS

- Definieren der Matrix A

```
>> A:= matrix([[ 1, -3, -4],
[- 1,1,1], [- 1,4,- 2]])
```

```
  +-              -+
  |   1, -3, -4   |
  |               |
  |  -1,  1,  1   |
  |               |
  |  -1,  4, -2   |
  +-              -+
```

- Definieren des Vektors b

```
>> b:=matrix([1,-1,2]):
```

- Lösen von $A\ L = b$

```
>> L:=linalg::matlinsolve(A,b)
```

```
  +-      -+
  |   6/5  |
  |        |
  |   3/5  |
  |        |
  |  -2/5  |
  +-      -+
```

- Einsetzen in die Geradengleichung

```
>> k:=L[3]: x_s:=matrix([g1,g2,g3])
```

```
  +-     -+
  |  7/5  |
  |       |
  |  2/5  |
  |       |
  |  1/5  |
  +-     -+
```


- Matrizenoperationen

```
>> B:=matrix([[1,0,0],[0,1,1],[1,1,1]]):
>> A*B, A-B, A+B
```

```
  +-            -+  +-            -+  +-            -+
  |  -3, -7, -7  |  |   0, -3, -4  |  |   2, -3, -4  |
  |              |  |              |  |              |
  |   0,  2,  2  |, |  -1,  0,  0  |, |  -1,  2,  2  |
  |              |  |              |  |              |
  |  -3,  2,  2  |  |  -2,  3, -3  |  |   0,  5, -1  |
  +-            -+  +-            -+  +-            -+
```

- Berechnen der Inversen (mit Probe)

```
>> A^(-1), A*A^(-1)
```

```
  +-                       -+  +-          -+
  |  -2/5, -22/15,  1/15    |  |  1, 0, 0  |
  |                         |  |           |
  |  -1/5,  -2/5,   1/5     |, |  0, 1, 0  |
  |                         |  |           |
  |  -1/5,  -1/15, -2/15    |  |  0, 0, 1  |
  +-                       -+  +-          -+
```


Etwas Zahlentheorie I

Fermatsche Primzahlen: $F_n = 2^{2^n} + 1$. Finden Sie die kleinste Zahl F_n, die keine Primzahl ist!

```
>> F:=2^(2^n)+1:
>> n:=1: F, isprime(F)
  5, TRUE
>> n:=2: F, isprime(F)
  17, TRUE
>> n:=3: F, isprime(F)
  257, TRUE
>> n:=4: F, isprime(F)
  65537, TRUE
>> n:=5: F, isprime(F)
  4294967297, FALSE
>> numlib::divisors(F)
  [1, 641, 6700417, 4294967297]
```


Etwas Zahlentheorie II

- Eine Liste der ersten Primzahlen bis 100

```
>> Menge:= [ i $ i=1..100 ]:
>> select(Menge,isprime)
```

```
  [2, 3, 5, 7, 11, 13, 17, 19, 23, 29,
   31, 37, 41, 43, 47, 53, 59, 61, 67,
   71, 73, 79, 83, 89, 97]
```

- Mersenne-Primzahlen $2^p - 1$, p Primzahl. Bestimmen der ersten Mersenne Primzahlen im Bereich ≤ 200.

```
>> Primes := select([i $ i=1..200],isprime):
>> select(Primes,p->isprime(2^p-1))
```

```
  [2,3,5,7,13,17,19,31,61,89,107,127]
```

```
>> numlib::mersenne()
```

Etwas Zahlentheorie III

Wir geben für die natürlichen Zahlen ≤ 1000 an, wieviele Zahlen $1, 2, 3, \ldots$ Teiler haben.

```
>> Liste:=[i $ i=1..1000]:
>> anz_teiler:= n -> nops(numlib::divisors(n)):
>> Liste1:=map(Liste,anz_teiler):
>> for i from 1 to 50 do
     print(i,nops(select(Liste1, x -> (x = i))))
   end_for:
```

```
  1, 0
  2, 168
   ...
```

Teiler der Zahl 840:

```
>> numlib::divisors(840)
```


Überlebensregeln

- Kommas zwischen Eingaben erzeugen den Datentyp einer Folge!
- Mehrere Befehle in einer Zeile besser durch ';' oder ':' trennen. Ausgabe wird mit ':' am Ende unterdrückt
- Die Eingabe % ergibt die Ausgabe des letzten Befehls
- Die Eingabe %n ergibt die Ausgabe des *n*-letzten Befehls
- Bei Eingaben, die über mehrere Zeilen gehen, kann ein Zeilenumbruch durch <SHIFT>+<ENTER> erreicht werden

Nützliches

- Löschen aller eigenen Variablen und Zurücksetzen auf den Anfangsstatus: reset()
- Anzeigen aller definierten Variablen: anames(All)
- Anzeigen aller selbst definierten Variablen: anames(All,User)
- Alle Ausgaben entfernen: Bearbeiten -> Alle Ausgaben entfernen
- Matheklammer erzeugen: π
- Lücke für Text erzeugen: ¶

Starten von MuPAD

- Kiosk: Lernprogramme -> Mathematisches -> MuPAD 4.0.0
- Benutzung der Terminals:
  - Einloggen auf einem Linuxrechner (l1,...,l14) mit `ssh -X l1.num.math.uni-goettingen.de`
  - Starten von MuPAD: `mupad &`
- Hilfefunktionen in MuPAD
  - ? (startet menügesteuertes Hilfefenster)
  - info(*name*) gibt eine Kurzhilfe zu *name*
  - ? *name* oder help("*name*") gibt ausführliche Hilfe.