MuPAD — Einführung

J. Billing

20.10.1999

Inhaltsverzeichnis

1 MuPAD — Erste Schritte

```
>> /* Linux, UNIX */
•  /* Windows    */
```

Nach dem Start von MuPAD erscheint der Eingabe–„Prompt“. Die Anzeige ist Betriebssystem–abhängig.

Jede Zeile muß mit `;` oder `:` abgeschlossen werden. Zeilen abgeschlossen mit `:` erzeugen keine Ausgaben.

```
>> 1 + 2;
```

Es können mehrere Zeile eingegeben werden. Dazu muß <UMSCHALT–RETURN> (<SHIFT–RETURN>) gedrückt werden.

```
>> for i from 1 to 3 do
     print(i, i*2)
   end_for;
```

Es können auch mehrere Anweisungen (Befehle) in einer Zeile stehen.

```
>> 1 + 2; 2 - 1: 3 + 2: 2*2;
```

Die letzten Ergebnisse können mit der Funktion `last` oder dem „Last“–Operator `%` weiterverarbeitet werden.

```
>> last(2), last(1), %2, %;
```

Kommentare innerhalb von Eingaben beginnen mit `/*` und enden mit `*/`.

```
>> abs(x) /* absoluter Betrag */ - x; /* noch ein Kommentar */
```

Die MuPAD–Hilfe wird durch `?` oder das Menü **Hilfe** aktiviert. Alternativ kann `help(befehl);` oder `?befehl` eingegeben werden, um Informationen zu `befehl` zu bekommen. Wenn das Fragezeichen benutzt wird, muß die Zeile nicht durch `;` abgeschlossen werden.

```
>> ?
>> ?plot
>> plotfunc(abs(x), x = -5..5);
```

Viele Funktionen können durch ihren Namen gefunden werden.

Z. B. Summen, Integrieren, Differenzieren, Faktorisieren (Zerlegung in Faktoren)

```
>> ?sum
>> ?int
>> ?diff
>> ?factor
>> ?fact       /* Fakultaet */
>> ?binomial   /* Binomialkoeffizienten */
```

```
>> sum(binomial(n, k), k = 1..n);
>> fact(5), 5!;
>> binomial(49, 6);
```

Variablen (oder Bezeichner) für symbolische Berechnungen

```
>> x, y, z, n, k, x_variable;
```

Rechnen mit Zahlen und Variablen

```
>> 2 + 2*x - x/y + 1/y - x - 1/2;
```

Zuweisung von Werten an Variablen

```
>> x:= 0: y:= 1:
>> 2 + 2*x - x/y + 1/y - x - 1/2;
```

Bestimmte und unbestimmte Summen

```
>> sum(k, k=1..n);
```

Faktorisieren (Zerlegung in Faktoren)

```
>> ?factor
>> Factor(%2);
```

Erweitern (Ausmultiplizieren)

```
>> expand(%);
```

Definition von Funktionen

```
>> F:= n -> Factor(sum(k, k = 1..n));
>> sum(k, k = 1..777);
>> F(777);
>> F(888);
```

Einige Namen sind reserviert, d. h. sie haben bereits einen Wert.

```
>> DIGITS;
>> D, PI, abs;
```

2 MuPAD — kleine Einführung

Computer**a**lgebra–**S**ystem CAS —
Rechner–unterstütztes Lösen vielfältiger mathematischer Aufgabenstellungen

Symbolisches Rechnen

```
>> sin(2*PI);
>> sqrt(4);
>> sqrt(8);
>> sqrt(4*x);
>> 2*x + 4 - x*y +2*y - 2 - x - y;
```

Unterschiede zum numerischen Rechnen (z. B. Taschenrechner)

nahezu beliebig große Zahlen

```
>> 1332233999736442316^15;
```

beliebig einstellbare Genauigkeit, dadurch Reduzierung von Rundungsfehlern

```
>> sqrt(8);
>> float(sqrt(8));
>> %2^2 - %^2;
>> DIGITS := 250:
>> sqrt(8);
>> float(sqrt(8));
>> %2^2 - %^2;
>> float(PI);
```

Funktionen

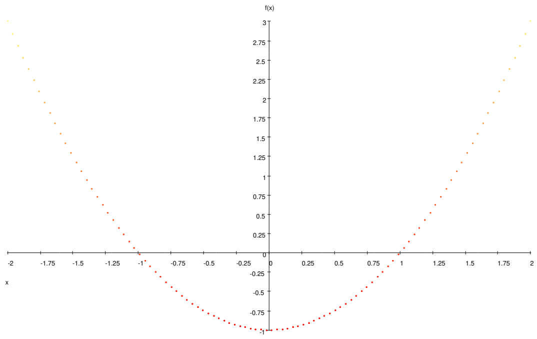
```
>> f := x -> (x^2 + 1); /* Klammern nicht vergessen! */
>> f(1);
>> f(2);
>> f(x + 2);
>> expand(%);
```

Bestimmte und unbestimmte Summen

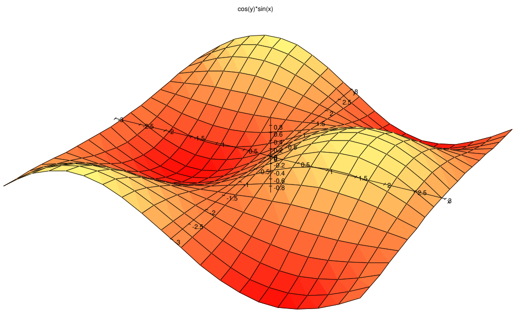
```
>> sum(k, k = 1..n);
>> sum(x^2, x = 0..1000);
>> sum(z^k, k = 1..n);
```

Funktionsgraphen

```
>> f:= x -> (x^2 - 1);
>> plotfunc(f(x), x = -2..2);
```




```
>> ?plotfunc
>> plotfunc(sin(x)*cos(y), x = -3..3, y = -3..3);
```



3 Übungsaufgabe — Lösung mit MuPAD

Gegeben sei die Funktion $x \mapsto x^3 - 3 \cdot x - 1$.

1. Erstellen Sie eine Wertetabelle der Funktion für alle ganzen Zahlen im Intervall -5 bis 5.
2. Erstellen Sie eine Wertetabelle der Funktion für alle Zahlen im Intervall -2 bis 2 mit Schrittweite $\frac{1}{3}$.
3. Berechnen Sie alle Nullstellen der Funktion.
4. Zeichnen Sie den Funktionsgraphen im Intervall von -3 bis 3.
5. Zeichnen Sie den Funktionsgraphen im Intervall von -2 bis -1.

Lösung mit Hilfe von MuPAD

I. Definition der Funktion

```
>> f:= x -> (x^3 - 3*x - 1);
>> f(x);
```

II a. Erstellen einer Wertetabelle -5 bis 5.

```
>> for x from -5 to 5 do
     print(x, f(x))
   end_for;
```

II b. Erstellen einer Wertetabelle -2 bis 2 mit Schrittweite $\frac{1}{3}$.

```
>> for x from -2 to 2 step 1/3 do
     print(x, f(x))
   end_for;
```

III. Berechnen aller Nullstellen.

```
>> unassign(x):
>> L:= solve(f(x) = 0, x);
```

IV. Anzeigen der Näherungswerte der Nullstellen.

```
>> map(L, float);
```

V. Zeichnen des Funktionsgraphen.

```
>> plotfunc(f(x), x = -3..3);
```

VI. Zeichnen des Ausschnitts -2 bis -1.

```
>> plotfunc(f(x), x = -2..-1);
```