Einige Beispiele zur numerischen Berechnung von Arithmetik-Termen im Computeralgebra System Maxima

1 Vorbemerkungen

Das CAS Maxima verfügt über die Zahlenformate integer, float und bfloat ("bigfloat").
Die Ergebnisse von Rechnungen, die strikt im Ganzzahlbereich stattfinden, sind stets korrekt.

Maxima kann mit Brüchen rechnen. Auch die Ergebnisse von Bruchrechnungen auf der Basis von ganzen Zahlen sind stets korrekt.

```
1/3+1/6;
```

$$\frac{1}{2}$$

Enthält eine Zahl einen (Dezimal-) Punkt, so ist sie automatisch eine Gleitkommazahl.

Die Ergebnisse von Rechnungen mit Gleitkommazahlen sind in der Regel nur näherungsweise
(also nicht strikt) korrekt. Jede Gleitkommazahl (float oder bfloat) wird im Folgenden als verseucht bezeichnet.

Ein Term kann in gemischter Form ganze Zahlen und Gleitkommazahlen enthalten; m.a.W.. er kann unverseuchte und verseuchte Zahlen enthalten.

Die Qualität des Ergebnisses ist nur so gut wie die des "schlechtesten" Teilterms;
m.a.W., ist ein Teilterm verseucht, so ist der ganze Term verseucht.

Mit Hilfe des Gleitkommaformats bfloat kann die Qualität von Berechnungen erheblich (genauer: fast beliebig) gesteigert werden.

Einige Beispiele:

```
3/7 + 5/13;
```

$$\frac{74}{91}$$

```
3.0/7 + 5/13;
```

0.8131868131868132

```
bfloat(3.0/7) + bfloat(5/13);
```

8.131868131868132b−1

```
(3/7 + 5/13) - (3.0/7 + 5/13) ;
```

0.0

```
is((3/7 + 5/13) = (3.0/7 + 5/13))  ;
```

false

```
is((3/7 + 5/13) = (bfloat(3/7) + bfloat(5/13))) ;
```

false

```
is((3/7 + 5/13) = 74/91);
```

true

```
1/2 + 1/3+1/6;
```

1

```
is(1/2 + 1/3 + 1/6 = 1) ;
```

true

```
is(1.0/2 + 1/3 + 1/6 = 1) ;
```

false

2 Beispiel: Term-Auswertung

Wir betrachten die Terme

```
z1 :   x*x*x*x - 4*y*y*y*y - 4*y*y $
z21 : (x^2)^2 - 4*(y^2)^2 - 4*y^2 $
z22 : x^4 - 4*y^4 - 4*y^2 $
z3 :   exp(4*log(x)) - 4* exp(4*log(y)) - 4* exp(2*log(y)) $
```

Rein mathematisch betrachtet, sollten alle vier Terme für konkrete (reelle) Werte von x und y dieselben Werte liefern: z1 = z21 = z22 = z3

Bemerkung: Die Werte von z21 und z22 wurden separat behandelt, da in manchen Programmiersprachen das Quadrieren anders gehandhabt wird als das allgemeine Potenzieren.

Wir betrachten nun die Werte

```
x : 665857.0 $
y : 470832.0 $
```

Die korrekten Werte der obigen Terme für diese Werte von x und y sind:
z1 = z21 = z22 = z3 = 1

Nun zu einigen konkreten Auswertungsverfahren

2.1 Auswertung durch die gewöhnliche floating point Arithmetik von Maxima

```
x : 665857.0 $
y : 470832.0 $
z1 :  x*x*x*x - 4*y*y*y*y - 4*y*y;
z21 : (x^2)^2 - 4*(y^2)^2 - 4*y^2;
z22 : x^4 - 4*y^4 - 4*y^2;
z3 :  exp(4*log(x)) - 4* exp(4*log(y)) - 4* exp(2*log(y));
```

$1.1885568 * 10^{7}$

$1.1885568 * 10^{7}$

$1.1885568 * 10^{7}$

$-4.914309120002441 * 10^{8}$

2.2 Auswertung durch die Ganzzahl-Arithmetik von Maxima

```
xi : 665857 $
yi : 470832 $
z1 :  xi*xi*xi*xi - 4*yi*yi*yi*yi - 4*yi*yi;
z21 : (xi^2)^2 - 4*(yi^2)^2 - 4*yi^2;
z22 : xi^4 - 4*yi^4 - 4*yi^2;
z3 :  exp(4*log(xi)) - 4*exp(4*log(yi)) - 4*exp(2*log(yi));
```

1
1
1
1

2.3 Auswertung durch die Bigfloat-Arithmetik von Maxima (floating point precision: fpprec = 16 default value)

```
fpprec : 16 $
xb : bfloat(665857.0) $
yb : bfloat(470832.0) $
z1 :  xb*xb*xb*xb - 4*yb*yb*yb*yb - 4*yb*yb;
z21 : (xb^2)^2 - 4*(yb^2)^2 - 4*yb^2;
z22 : xb^4 - 4*yb^4 - 4*yb^2;
z3 :  exp(4*log(xb)) - 4*exp(4*log(yb)) - 4*exp(2*log(yb));
```

−6.97344b5
−6.97344b5
−6.97344b5
−1.223321600001831b8

2.4 Auswertung durch die Bigfloat-Arithmetik von Maxima (fpprec wird explizit auf 100 gesetzt)

```
fpprec : 100 $
xh : bfloat(665857.0);
yh : bfloat(470832.0);
z1 :  xh*xh*xh*xh - 4*yh*yh*yh*yh - 4*yh*yh;
z21 : (xh^2)^2 - 4*(yh^2)^2 - 4*yh^2;
z22 : xh^4 - 4*yh^4 - 4*yh^2;
z3 :  exp(4*log(xh)) - 4*exp(4*log(yh)) - 4*exp(2*log(yh));
```

6.65857b5
4.70832b5
1.0b0
1.0b0
1.0b0
1.00000000000000000000000000000000000000000000000000000000000000000000000000820436012735228967338 82
b0

```
fpprec : 100 $
xh : bfloat(665857);
yh : bfloat(470832) ;
z1 :  xh*xh*xh*xh - 4*yh*yh*yh*yh - 4*yh*yh;
z21 : (xh^2)^2 - 4*(yh^2)^2 - 4*yh^2;
z22 : xh^4 - 4*yh^4 - 4*yh^2;
z3 :  exp(4*log(xh)) - 4*exp(4*log(yh)) - 4*exp(2*log(yh));
```

6.65857b5

4.70832b5
1.0b0
1.0b0
1.0b0
1.000000000000000000000000000000000000000000000000000000000000000000000000000082043601273522896733882b0

3 Eine einfache Zählschleife

In der folgenden Zählschleife sind xb, yb gewöhnliche und xh, yh bigfloat Gleitkommazahlen. Rein mathematisch gesehen, sollte stets yb = yh sein - im Gegensatz zum Ausdruck des Programms (3. Spalte).

```
count(n) :=
 (xb : 0.01,
  xh : bfloat(1/100),
  yb : 0.0,
  yh : bfloat(0),
  i : 0,
  while i < n do
   (i : i+1,
    yb : yb+xb,
    yh : yh+xh,
    print(i, yb, yh, is(yb = yh ) ))
    );
```

$$count(n):=(xb:0.01, xh:\mathrm{bfloat}\left(\frac{1}{100}\right), yb:0.0, yh:\mathrm{bfloat}(0), i:0, \mathrm{while}\ i<n\ \mathrm{do}\ (i:i+1, yb:yb+xb, yh:yh+xh, \mathrm{print}(i, yb, yh, \mathrm{is}(yb=yh))))$$

```
count(10);
```

1 0.01 1.0b−2 ***false***
2 0.02 2.0b−2 ***false***
3 0.03 3.0b−2 ***false***
4 0.04 4.0b−2 ***false***
5 0.05 5.0b−2 ***false***
6 0.06 6.0b−2 ***false***
7 0.07 7.0b−2 ***false***
8 0.08 8.0b−2 ***false***
9 0.09 9.0b−2 ***false***

```
10  0.1  1.0b-1  false
        done
```

4 Eine einfache Iteration

In der folgenden Iteration sollte stets x = 0,2 sein.

```
simple_iteration_fp(n) :=
  (x : 0.2,
  for i:1 thru n do
    (x : 11*x-2,
     print(i, x)) ) ;
```

```
        simple_iteration_fp(n):=
(x:0.2,for i thru n do (x:11*x-2,print(i,x)))
```

```
simple_iteration_fp(10);
```

```
1  0.2000000000000002
2  0.200000000000002
3  0.2000000000000215
4  0.2000000000002364
5  0.2000000000026008
6  0.2000000000286084
7  0.2000000003146924
8  0.2000000034616169
9  0.2000000380777864
10  0.2000004188556508
        done
```

```
simple_iteration_bf(n) :=
  (fpprec : 16,
    x : bfloat(0.2),
   for i:1 thru n do
    (x : 11*x-2,
     print(i, "  ", x) ) ) ;
```

```
        simple_iteration_bf(n):=(fpprec:16,x:bfloat(0.2),for i
thru n do (x:11*x-2,print(i,  ,x)))
```

```
simple_iteration_bf(20);
```

```
1     2.000000000000001b-1
2     2.000000000000014b-1
3     2.00000000000148b-1
4     2.000000000001626b-1
5     2.00000000001788b-1
6     2.00000000196683b-1
```

```
7     2.000000002163511b−1
8     2.000000023798617b−1
9     2.000000261784782b−1
10     2.000002879632599b−1
11     2.000031675958591b−1
12     2.000348435544497b−1
13     2.00383279098947b−1
14     2.042160700884169b−1
15     2.463767709725854b−1
16     7.101444806984397b−1
17     5.811589287682836b0
18     6.19274821645112b1
19     6.792023038096232b2
20     7.469225341905855b3
          done
```

```
simple_iteration_bf2(n) :=
  (fpprec : 1000,
   xh : bfloat(2/10),
   for i:1 thru n do
     (xh : 11*xh - 2,
      print(i, "  ", xh) ) ) ;
```

$$\text{simple\_iteration\_bf2}(n):=(fpprec:1000,xh:\text{bfloat}\left(\frac{2}{10}\right),\text{for}$$

$$i \text{ thru } n \text{ do } (xh:11*xh-2,\text{print}(i, \quad ,xh)))$$

```
simple_iteration_bf2(10);
```

```
1     2.0b−1
2     2.0000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
```

00000000000000000000000000000000000000000000000000002b−1

3 2.0000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000023b−1

4 2.0000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000253b−1

5 2.0000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000

000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000002786b−1
6 2.0000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000030644b−1
7 2.0000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000337086b−1

8 2.0000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000003707944b−1

9 2.0000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000040787387b−1

10 2.0000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000

```
000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000448661255b−
1
        done
```

5 Rechengesetze

5.1 Das Assoziativgesetz für die Addition

```
(r1 : (0.1 + 0.2) + 0.3 ,
 r2 : 0.1 + (0.2 + 0.3) ,
 print(r1) ,
 print(r2) ,
 print(r1-r2),
 is(r1=r2)   );
```

0.6000000000000001

0.6

$1.110223024625157 * 10^{-16}$

false

```
(fpprec : 16,
 r1 : bfloat( (0.1 + 0.2) + 0.3 ) ,
 r2 : bfloat( 0.1 + (0.2 + 0.3) ),
 print(r1) ,
 print(r2) ,
 print(r1-r2),
 is(r1=r2)   ) ;
```

6.000000000000001b−1

6.0b−1 1.110223024625157b−16

false

```
(r1 : bfloat( (1/10 + 2/10) + 3/10 ) ,
 r2 : bfloat( 1/10 + (2/10 + 3/10) ) ,
 print(r1) ,
 print(r2) ,
 print(r1-r2),
 is(r1=r2)  ) ;
```

```
6.0b−1
6.0b−1
0.0b0
        true
```

5.2 Das Distributivgesetz

```
( r1 : 100 * (0.1 + 0.2),
  r2 : (100 * 0.1 + 100 * 0.2),
  print(r1),
  print(r2),
  print(r1-r2),
  is(r1=r2) );
```

```
30.0
30.0
```

$3.552713678800501 * 10^{-15}$

```
        false
```

```
( r1 : bfloat(100 * (1/10 + 2/10) ),
  r2 : bfloat(100 * 1/10 + 100 * 2/10 ),
  print(r1),
  print(r2),
  print(r1-r2),
  is(r1=r2) );
```

```
3.0b1
3.0b1
0.0b0
        true
```