Periodische Kommazahlen

Prof. Dr. Dörte Haftendorn, MuPAD 4, http://haftendorn.uni-lueneburg.de Aug.06

Automatische Übersetzung aus MuPAD 3.11, Mai 2004

Es fehlen nocht textliche Änderungen, die MuPAD 4 direkt berücksichtigen, das ist in Arbeit.

Web: http://haftendorn.uni-lueneburg.de www.mathematik-verstehen.de

++

```
DIGITS:=40   //Dezimalen-Anzeige vergrößern
```

40

```
float(1/17)
```

0.05882352941176470588235294117647058823529

So kann man die Periodenlänge ablesen.

Experimentiere mit 1/n , wie lang ist jeweils die Periode?

```
n:=13: [1/n,float(1/n)]
```

$\left[\frac{1}{13}, 0.07692307692307692307692307692307692307692\right]$

Versuche abbbrechende und nicht abbrechende Kommazahlen zu sortieren.

Bei welchen Nennern n bricht die Kommazahl 1/n ab?

```
n:=2^3*5^7: [1/n,float(1/n)]
```

$\left[\frac{1}{625000}, 0.0000016\right]$

```
//------------------------------------------------------------
```

Experimentiere: bei welchen Nennern n erhält man echte Vorperioden?

Vorperioden sind hinter dem Komma Ziffern ungleich Null, die nicht in der Periode vorkommen

```
n:=16*17: [1/n,float(1/n)]
```

$\left[\frac{1}{272}, 0.003676470588235294117647058823529411764706\right]$

```
//------------------------------------------------------------
```

Rückverwandlung von periodischen Kommazahlen in Brüche

```
DIGITS:=8:
[1/(10^n-1),float(1/(10^n-1))] $n=1..5
```

$\left[\frac{1}{9}, 0.11111111\right], \left[\frac{1}{99}, 0.01010101\right], \left[\frac{1}{999}, 0.001001001\right], \left[\frac{1}{9999}, 0.00010001\right], \left[\frac{1}{99999},\right.$

```
[23/(10^n-1),float(23/(10^n-1))] $n=2..5
```

$\left[\frac{23}{99}, 0.23232323\right], \left[\frac{23}{999}, 0.023023023\right], \left[\frac{23}{9999}, 0.00230023\right], \left[\frac{23}{99999}, 0.0002300023\right]$

```
[2347/(10^n-1),float(2347/(10^n-1))] $n=4..7
```

$\left[\frac{2347}{9999}, 0.23472347\right], \left[\frac{2347}{99999}, 0.023470235\right], \left[\frac{2347}{999999}, 0.0023470023\right], \left[\frac{2347}{9999999}, 0.000$

Also erzeugen die Nenner mit lauter Neunen Perioden, die genau dem Zähler entsprechen, aufgefüllt mit Nullen.
Die Periodenlänge ist gleich der Anzahl der Neunen.
Umgekehrt: **Für die Rückverwandlung sofort-periodischer Dezimalzahlen muss man offenbar die**
Periode als Zähler nehmen und als Nenner soviele Neunen wie die Periodenlänge angibt.

```
DIGITS:=40:
[0.123456712345671234567,1234567/9999999,float(1234567/9999999)]
```

$\left[0.123456712345671234567, \frac{1234567}{9999999}, 0.1234567123456712345671234567123456712346\right.$

Kommazahlen mit Vorperiode

Man muss den Vorperiodenteil abtrennen.

```
[0.345123456712345671234567,345,"+",1234567/9999999];
[345+1234567/9999999000,float(345+1234567/9999999)/1000]
```

$\left[0.345123456712345671234567, 345, \text{"+"}, \frac{1234567}{9999999}\right]$

$\left[\frac{3450000889567}{9999999000}, 0.3451234567123456712345671234567123456712\right]$

```
float(0.345+1234567/9999999000)
```

$0.3451234567123456712345671234567123456712$

Mit dem Befehl des CAS bekommt man nur Näherungslösungen:

```
numeric::rationalize(0.34512345671234567)
```

$\frac{34512345671234567}{100000000000000000}$

```
float(%)
```

0.34512345671234567

```
588235294117647/9999999999999999
```

$\frac{1}{17}$

```
//--------------------------------------------------------------------
```

Die Periodenlänge teilt **Phi(n)=Euler(n)**, wenn N teilerfremnd zu 10 ist.
Bestätigen Sie das für etliche Beispiele.
Beschaffung von EulerPhi

```
numlib::phi(3*11*13)
```

240

```
n:=3*11*13: [1/n,float(1/n)]
```

$\left[\frac{1}{429}, 0.002331002331002331002331002331002331002331\right]$

Periodenlänge 6 und 6 teilt tatsächlich 240.

Experimentierfeld:

```
float(1/137)
```

0.007299270072992700729927007299270072992701

```
float(1/(2*137));float(1/(16*137));
```

0.003649635036496350364963503649635036496350

0.0004562043795620437956204379562043795620438

```
float(1/17);float(1/(16*17));
```

0.05882352941176470588235294117647058823529

0.003676470588235294117647058823529411764706

```
float(1/11);float(1/(4*11));float(1/(44*11));
```

0.09090909090909090909090909090909090909091

0.02272727272727272727272727272727272727273

0.002066115702479338842975206611570247933884


```
numlib::phi(44)
```

20

```
DIGITS:=52
```
52

```
float(1/3);float(1/(4*3));float(1/(7*17));
```
0.3333333333333333333333333333333333333333333333333333

0.08333333333333333333333333333333333333333333333333333

0.008403361344537815126050420168067226890756302521008403

```
6*16
```
96

```
float(1/(11*11));float(1/(7*19));float(1/(11*17));
```
0.008264462809917355371900826446280991735537190082644628

0.007518796992481203007518796992481203007518796992481203

0.005347593582887700534759358288770053475935828877005348

```
numlib::phi(121)
```
110

```
float(1/(16*11));float(1/(4*3));float(1/(256*17));
```
0.005681818181818181818181818181818181818181818181818182

0.08333333333333333333333333333333333333333333333333333

0.0002297794117647058823529411764705882352941176470588235

```
float(1/256)
```
0.00390625

```
float(1/4);float(1/44)
```
0.25

0.02272727272727272727272727272727272727272727272727273


```
float(1/4);float(1/(4*17));float(1/17);float(1/34)
```

$$0.25$$

$$0.014705882352941176470588235294117647058823529411764 71$$

$$0.058823529411764705882352941176470588235294117647058 82$$

$$0.029411764705882352941176470588235294117647058823529 41$$

```
float(1/29);float(1/(4*29));float(1/(16*29));float(1/(3125*29))
```

$$0.034482758620689655172413793103448275862068965517241 38$$

$$0.008620689655172413793103448275862068965517241379310 345$$

$$0.002155172413793103448275862068965517241379310344827 586$$

$$0.000011034482758620689655172413793103448275862068965 51724$$

```
1/5^5;float(%)
```

$$\frac{1}{3125}$$

$$0.00032$$

```
1/99;float(%)
```

$$\frac{1}{99}$$

$$0.010101010101010101010101010101010101010101010101010 101$$

```
34*78
```

$$2652$$

```
2652/9900
```

$$\frac{221}{825}$$

```
float(221/825)
```

$$0.267878787878787878787878787878787878787878787878787 879$$