Disko-Problem: Simulation mit Maxima

Einige Hinweise:
Das folgende Maxima-Dokument enthält Maxima-Code-Zellen, Text-Zellen und Graphik-Zellen.
Maxima-Code ist in einer Festpunktschrift, reiner Text ist in einer Proportionalschrift (meist in "Times Roman") dargestellt.
Kommentare sind syntaktisch durch das "Klammerpaar" /* Kommentar-Beispiel */ gekennzeichnet.

```
/* Kommentare können sich auch über
   mehrere Zeilen erstrecken.   */
```

Das $ - Zeichen am Ende einer Eingabezelle verhindert, dass die Ausgabe ausgedruckt wird.

Die folgenden beiden Befehle dienen dazu, um verschiedene Darstellungsarten einzurichten. Sie haben nichts mit der eigentlichen Simulation zu tun.

```
(%i26) set_display('ascii)$  /* macht Zeilenumbruch am Seitenrand möglich */

(%i2)  set_display('xml)$    /* ermöglicht Graphik-Darstellung im Dokument */
```

Die folgenden Kommandos make_random_state und set_random_state dienen der Initiierung des Zufallszahlen-Generators von Maxima

```
(%i5)  randomstate1 :  make_random_state (654321) $
       set_random_state (randomstate1) $
       make_random_state(true) $

(%i6)  random(1000);
(%o6)  768
```

Die folgende Funktion RaumJ ermittelt, in welchem Raum sich Jens aufhält. Der Eingabeparameter p gibt an, mit welcher Wahrscheinlichkeit Jens in der Disko ist.

Die Werte der Variablen r1, r2, r3, r4, rn sind jeweils gleich 0 oder 1, je nachdem, ob Jens sich in dem Raum befindet oder nicht (r1=1: Jens ist in Raum 1, ... , rn=1: Jens ist nicht in der Disko).
Variable rog: Obergrenze für die random-Funktion
Die Ausgabe erfolgt als Liste: [r1, r2, r3, r4, rn], wobei genau ein Wert gleich 1 ist und alle anderen Werte gleich Null sind.

```
(%i7)  RaumJ(p) :=
         block([r1:0, r2:0, r3:0, r4:0, rn:0, rog : 1000000],
           r : random(rog),
           if r < (1-p)*rog then rn : 1                          /* J nicht in Disko */
             elseif r < (1-p)*rog + rog*p/4 then r1 : 1           /* J in Raum 1 */
                elseif r < (1-p)*rog + 2*rog*p/4 then r2 : 1      /* J in Raum 2 */
                   elseif r < (1-p)*rog + 3*rog*p/4 then r3 : 1   /* J in Raum 3 */
                     else r4 : 1,                                 /* J in Raum 4 */
           [r1, r2, r3, r4, rn] ) $
```

Einige Test-Aufrufe:

```
(%i8)  RaumJ(0.75);
(%o8)  [0,0,0,0,1]

(%i9)  RaumJ(1);
(%o9)  [1,0,0,0,0]

(%i10) RaumJ(0);
(%o10) [0,0,0,0,1]

(%i11) RaumJ(0.5);
(%o11) [0,0,0,0,1]
```

Die folgende Funktion SucheM simuliert einen einzigen Such-Vorgang. Zunäst wird durch den Aufruf von RaumJ(p) der Raum (mit Hilfe des Zufallszahlengenerators) festgelegt, in dem sich Jens befindet.
Variable n123: Jens befindet sich nicht in Raum 1 und nicht in Raum 2 und nicht in Raum 3.
Variable n123r4: Jens befindet sich nicht in Raum 1 und nicht in Raum 2 und nicht in Raum 3, sondern in Raum 4.
Ausgabe: Die Zweier-Liste [n123r4, n123] für die weitere Verarbeitung (Aggregierung der Daten).

```
(%i12) SucheM(p) :=
       block( [RJ : [0, 0 ,0 ,0, 0], n123:0, n123r4:0],
         RJ : RaumJ(p),
         if (RJ[1]=0 and RJ[2]=0 and RJ[3]=0)
            then n123 : n123+1,
         if (RJ[1]=0 and RJ[2]=0 and RJ[3]=0 and RJ[4]=1)
            then n123r4 : n123r4+1,
          [n123r4, n123] ) $

(%i13) SucheM(0.6);
(%o13) [0,0]
```

Der Einzel-Suchvorgang SucheM(p) wird im Folgenden L-mal hintereinander ausgeführt.
Die Ergebnisse von SucheM(p) werden in der Liste SumL aufaddiert.
SumL: 2-elementige Liste von der Struktur der Ausgabe von SucheM(p).
Ausgabe: Anteil von "Raum4" in der Menge
"(nicht Raum 1) und (nicht Raum 2) und (nicht Raum 3) und (Raum 4)"

```
(%i14) SimulationSuche(p, L) :=
         block( [SumL : [0,0]] ,
           for i:1 thru L do
             SumL : SumL + SucheM(p),
           if notequal(SumL[2],0) then return(float(SumL[1]/SumL[2]))
             else return("Bitte Simulation mit mehr Läufen wiederholen") ) $

(%i15) SimulationSuche(0.7,1000);
(%o15) 0.3622559652928417
```

Im Folgenden werden je 1000 Testläufe für die Wahrscheinlichkeiten von 0 bis 1 mit Schrittweite 0.1 durchgeführt.

```
(%i16) LT : makelist([0.1*i, SimulationSuche(0.1*i, 1000)], i, 0, 10, 1) $

(%i27) LT;
(%o27) [[0, 0.0], [0.1, 0.02939587160920782], [0.2, 0.05722359590250795],
[0.3, 0.09514013885317563], [0.4, 0.1370909615107395],
[0.5, 0.1921404412948945], [0.6000000000000001, 0.2641441441441442],
[0.7000000000000001, 0.3603622577927548], [0.8, 0.4960449094156673],
[0.9, 0.6942325297528227], [1.0, 1.0]]

(%i18) TabellenAusdruck(L) :=
        (print("               p              n123r4 / n123"),
         for K in L  do
           printf(true, "~10,2h ~12,4h  ~%",first(K), last(K) ) )  $
```

In der folgenden Tabelle sind in Abhängigkeit von der Wahrscheinlichkeit p die Simulationsergebnisse (d.h. die bedingten Wahrscheinlichkeiten
P(R4 | nicht(R1) und nicht(R2) und nicht(R3))
dargestellt.

```
(%i19) TabellenAusdruck(LT) $
```

p	*n123r4 / n123*
0.00	0.0000
0.10	0.0236
0.20	0.0590
0.30	0.1139
0.40	0.1312
0.50	0.1864
0.60	0.2956
0.70	0.3664
0.80	0.4600
0.90	0.7021
1.00	1.0000

Die (diskreten) Werte der obigen Tabelle sollen im Folgenden mit den Werten der (stetigen) Funktion f(p) verglichen werden. Zu diesem Zweck wird beides in einem gemeinsamen Schaubild eingetragen.

```
(%i20) f(p) := p/(4-3*p) $       /* Definition der Funktion f */
```

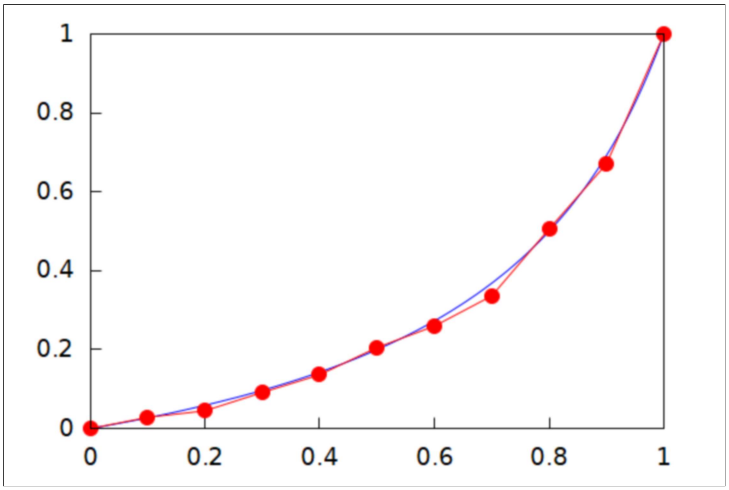
```
(%i21) load(draw);
```

(%o21) *C:/maxima-5.42.2/share/maxima/5.42.2/share/draw/draw.lisp*

Im Folgenden: Graphik-Darstellung einer Simulation mit jeweils 1.000 Läufen

```
(%i23) LT : makelist([0.1*i, SimulationSuche(0.1*i, 1000)], i, 0, 10, 1) $ ;
       DrawGraphik()
        (G1 : [color=blue, explicit(f(p), p,0,1)],
         G2 : [color=red, point_type=filled_circle, point_size=2,
         points_joined=true, points(LT)],
       wxdraw2d(G1, G2) )  $
```

(%t23)



Im Folgenden: Graphik-Darstellung einer Simulation mit jeweils 10.000 Läufen

```
(%i25) LT : makelist([0.1*i, SimulationSuche(0.1*i, 10000)], i, 0, 10, 1) $ ;
       DrawGraphik()
        (G1 : [color=blue, explicit(f(p), p,0,1)],
         G2 : [color=red, point_type=filled_circle, point_size=2,
         points_joined=true, points(LT)],
       wxdraw2d(G1, G2)) $ ;
```

(%t25)