Übungen zur Vorlesung „Physik auf dem Computer"
Sommersemester 2018

Übungsgruppenleiter:
Robin Bardakcioglu – `rhb@itp1.uni-stuttgart.de; Do. 14:00 - 15:30 Uhr, 5.331`
Johannes Reiff – `jreiff@itp1.uni-stuttgart.de; Di. 14:00 - 15:30 Uhr, 4.141`
Matthias Feldmaier – `fem@itp1.uni-stuttgart.de; Do. 14:00 - 15:30 Uhr, 4.141`

Übungsblatt 6 Ausgabe: 30. Mai 2018

Dienstagsübung: schriftliche Abgabe 03.06.18, Besprechung 05.06.18
Donnerstagsübungen: schriftliche Abgabe 05.06.18, Besprechung 07.06.18

Aufgabe 16.1: Tests für Sequenzen von Zufallszahlen

In dieser Aufgabe soll ein einfacher Pseudozufallszahlengenerator aus einem multiplikativen Algorithmus erzeugt und getestet werden. Die Folge der Zufallszahlen wird dabei durch

$$u_{i+1} = (c\,u_i) \mod p \tag{1}$$

gebildet, wobei c und p Konstanten sind. Im Intervall $[0, 1]$ gleichverteilte Zufallszahlen erhält man dann durch Division der u_i durch die maximale Periodenlänge p, also $z_i = u_i/p$.

a) Laden Sie das Programmpaket zur Aufgabe aus dem Web-Angebot zur Vorlesung. Sie finden darin das Programm `zufallszahlen.cc`. Dieses ist so vorbereitet, dass ein multiplikativer Generator von Pseudozufallszahlen in einer Funktion eingebaut werden kann und mit zwei verschiedenen Parametersätzen aufgerufen wird. Ergänzen Sie den notwendigen Quelltext zur Erzeugung gleichverteilter Pseudozufallszahlen im Intervall $[0, 1]$ nach Gleichung (1).

b) Das Programm erstellt drei Vektoren der Länge 100.000, in denen die Pseudozufallszahlen aus dem selbstgeschriebenen Generator und der Funktion `rand()` aus der C-Standardbibliothek abgespeichert werden. Schreiben Sie diese in eine Datei und erzeugen Sie anschließend Histogramme, die angeben wie viele Zufallszahlen in jedes Intervall der Länge 0,01 zwischen 0 und 1 fallen. In `gnuplot` geht das zum Beispiel für den Fall, dass Sie die Daten der ersten Spalte (`$1`) als Histogramm darstellen möchten, mit:

```
1 bin(x,width)=width*floor(x/width)
2 plot "Datendatei.dat" u (bin(\$1,0.01)):(1.0) smooth freq with boxes
```

ToDo: stellen Sie alle Histogramme vor.

(3 Punkt(e), Votier)

c) Speichern Sie die ersten 10.000 Pseudozufallszahlen aus jedem Vektor nach dem Muster

```
wuerfel << z_multiplikativ1[i] << "\t" << z_multiplikativ1[i+1] << "\t"
        << z_multiplikativ1[i+2] << ...
...
```

ab. Erzeugen Sie mit dieser Datei jeweils eine Abbildung für den Quadrat- und den Würfeltest für jede der drei Sequenzen. Für den Quadrattest tragen Sie z_{i+1} als Funktion von z_i auf. Der Würfeltest besteht darin, z_{i+2} in einem dreidimensionalen Diagramm als Funktion von z_{i+1} und z_i aufzutragen. Letzteres geht zum Beispiel in `gnuplot` mit:

```
splot "wuerfel.dat" u 1:2:3 w d
```

Dabei ist es wichtig, die Punkte nicht durch zu große Symbole darzustellen, was in `gnuplot` mit `w d` (`with dots`) erreicht werden kann. Achten Sie beim Würfeltest, dass Sie sich das Diagramm aus mehreren Richtungen anschauen und verwenden Sie eine gut geeignete für das Diagramm.

ToDo: Stellen Sie alle 6 Abbildungen vor.

(4 Punkt(e), Votier)

d) Unter vielen Unix-Betriebssystemen steht die Gerätedatei `/dev/random` zur Verfügung. Diese kann als Quelle von Zufallszahlen verwendet werden. Die Zufallszahlen werden gebildet, indem nicht aus einem Algorithmus stammende, vom Rechner meist nicht kontrollierbare Ereignisse (Netzwerkverkehr, Tastatureingaben, ...) gesammelt und verrechnet werden. Das erfolgt jedoch deutlich langsamer als das Bilden von Pseudozufallszahlen.

Fügen Sie Ihrem Programm die Zeilen

```
// Auslesen von 20 Zufallszahlen aus /dev/random
ifstream devrandom("/dev/random",ios_base::in);
for (int i = 0; i < 20; i++) {
  unsigned int Ergebnis;
  devrandom.read(reinterpret_cast<char*>(&Ergebnis),sizeof(Ergebnis));
  cout << Ergebnis << endl;
}
devrandom.close();
```

hinzu und beobachten Sie, wie die Zufallszahlen ausgegeben werden.

Überlegen Sie sich, in welchem Bereich die so gebildeten Zufallszahlen liegen. Berechnen Sie damit gleichverteilte Zufallszahlen in Intervall $[0, 1]$. Recherchieren Sie dazu, wie Sie die größte vorzeichenlose ganze Zahl vom Typ `int` ermitteln können.

ToDo: Stellen Sie das angepasste Programm vor.

(2 Punkt(e), Votier)

Aufgabe 17.1: Random walk

Ein *random walk* ist eine Zeitreihe von Ortskoordinaten $\boldsymbol{x}(t)$, die durch Aufsummierung von Zufallszahlen bzw. -vektoren $\boldsymbol{\gamma}_i$ entsteht: $\boldsymbol{x}(t) = \sum_{i=1}^{t} \boldsymbol{\gamma}_i$. In dieser Übung sollen Sie ein Programm

schreiben, das einen zweidimensionalen *random walk* erzeugt, indem mit gleicher Wahrscheinlichkeit zufällig jeweils einer der Vektoren $\boldsymbol{\gamma}_i \in \{(1,0),\ (0,1),\ (-1,0),\ (0,-1)\}$ generiert und dann zu einem zweidimensionalen *random walk* $\boldsymbol{x}(t)$ aufaddiert wird. Verwenden Sie einen geeigneten Zufallszahlengenerator um diese Auswahl zu treffen.

a) Schreiben Sie ein Programm, welche random walks der Länge 10, 100, 1.000, 10.000, und 100.000 berechnet und deren Trajektorien in Ausgabedateien schreibt.

b) Plotten Sie diese Trajektorien so, dass man den Verlauf der Walks in zwei Dimensionen sieht. **ToDo:** Stellen Sie diese Abbildungen vor.

(4 Punkt(e), Votier)

c) Berechnen Sie den Erwartungswert des Abstandsvektors $\langle \boldsymbol{R}(t) \rangle = \langle \boldsymbol{x}(t) - \boldsymbol{x}(0) \rangle$ und das mittlere Abstandsquadrat $\langle \boldsymbol{R}^2(t) \rangle$ aus 1000 random walks mit verschiedenen Zufallszahlensequenzen. **ToDo:** Plotten Sie den zeitlichen Verlauf von $|\langle \boldsymbol{R}(t) \rangle|$ und $\langle \boldsymbol{R}^2(t) \rangle$ und erklären Sie Ihre Beobachtung.

(6 Punkt(e), Votier)