Universität Bielefeld	Computerphysik	Prof. Dr. Jürgen Schnack
Fakultät für Physik	SS 2016	jschnack@uni-bielefeld.de

Aufgabenblatt 10

10.1 Monte-Carlo-Integration (Hausaufgabe + Email an die Tutoren, Einsendeschluss Mitternacht vor der Übung)

Sie können die folgenden Aufgaben in einer Programmiersprache Ihrer Wahl lösen. Bitte geben Sie an, welchen Zufallszahlengenerator Sie genutzt haben. Falls Sie keinen zur Hand haben, können Sie den Mersenne-Twister `mt19937ar.c` benutzen, den Sie in stud.ip finden.

1. Das folgende Integral kennen Sie: es ist die Fläche des Einheitskreises und somit π.

   Berechnen Sie das Integral der Funktion $4\sqrt{1-x^2}$ in den Grenzen von 0 bis 1 mit der Mittelwertmethode. Wählen Sie zunächst eine feste Anzahl von MC-Iterationen. Wiederholen Sie bei dieser Anzahl Ihre MC-Integration viele Male und sehen Sie sich die Streuung der Ergebnisse (einer einzelnen MC-Integration) in einem Histogramm an. Wiederholen Sie anschließend diesen Versuch mit einer höheren Anzahl von MC-Iterationen pro MC-Integration. Wie hängt die Streuungsbreite von der Zahl der Monte-Carlo-Iterationen ab?

   Tragen Sie Ihre Ergebnisse gegen die Anzahl der MC-Iterationen $N_{\mathrm{MC}} = 10,\ 100,\ 1000,\ 10000, \ldots$ auf. Stimmt die theoretisch vorhergesagte Skalierung der Genauigkeit mit N_{MC} in etwa?

2. Berechnen Sie das eindimensionale Integral

$$\int_0^1 \frac{x^2 e^x}{1 - x + x e^x} \mathrm{d}x \tag{1}$$

   mit der Mittelwertmethode, einmal mit und einmal ohne *Importance-Sampling*. Denken Sie sich dazu eine günstige Funktion $w(x)$ aus. Vergleichen Sie wie zuvor die Ergebnisse für verschiedene Anzahlen von MC-Iterationen mit Hilfe entsprechender Histogramme. Für genaue Analysen der Histogramme können Sie versuchen, an diese Gaußsche Normalverteilungen anzufitten.

   Können Sie feststellen, dass man mittels *Importance-Sampling* mit weniger MC-Iterationen die gleiche Genauigkeit erreicht?

Tipps zum Programmieren vom Experten (Matthias Götte):
Vor der main-Funktion:

```
#include <time.h>

//Funktionen müssen im eigenen Programm deklariert werden,
//da keine headerdatei vorhanden ist
void init_genrand();
double genrand_real1();
```

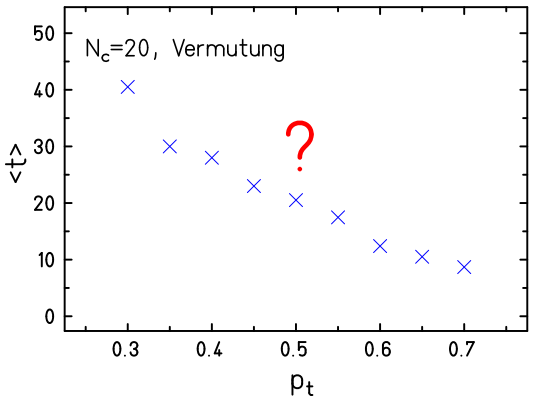
In der main-Funktion:

```
init_genrand(time(NULL));
//Um einen individuellen Seed zu generieren, basierend auf der aktuellen Zeit
```

Beim compilieren mt19937ar.c mit anhängen.

10.2 Metastasenbildung als Beispiel für Zufallsprozesse (Hausaufgabe + Bearbeiten in den Übungen)

Im Kolloquiumsvortrag von Prof. Timothy Newman am 4. Juni 2012 wurde folgende These untersucht: Krebsgeschwüre bilden im Körper Kolonien, sogenannte Metastasen. Diese gehen von einzelnen Zellen aus, die sich in anderen Organen einnisten. Eine Hypothese besagt, dass diese Zellen dafür sehr fit sein müssen, eventuell sogar hilfreiche Mutationen aufweisen müssen. Da das eher unwahrscheinlich ist, dauert die Metastasenbildung zum Glück recht lange. In der Biologie/Medizin sucht man deshalb nach diesen mutierten Superkrebszellen.
Es zeigt sich aber überraschenderweise, dass das nicht die einzige mögliche Erklärung ist. Dies können Sie selbst in dem folgenden Modell untersuchen: Ein Krebsgeschwür sendet Zellen aus, die sich woanders einnisten. Ob sie dort überleben, hängt davon ab, ob der Zellhaufen zu einer kritischen Größe N_c heranwachsen kann. Diesen Prozess kann man diskretisieren. In jedem Zeitschritt kann sich jede vorhandene Zelle mit der Wahlscheinlichkeit p_t teilen oder mit der Wahrscheinlichkeit p_s sterben. Es gilt $p_t + p_s = 1$. Die angesprochenen Superzellen sind durch ein hohes p_t gekennzeichnet, normale Krebszellen eher durch ein kleines p_t.
Man kann jetzt bei gegebenem p_t diesen stochastischen Prozess durch Zufallszahlen simulieren. Dazu würfelt man in jedem Zeitschritt für jede lebende Zelle eine zwischen 0 und 1 gleichverteilte Zufallszahl r. Wenn $r > p_t$, dann stirbt die Zelle, sonst teilt sie sich.
Die Frage ist, welche Zeit die erfolgreichen Kolonien als Funktion von p_t durchschnittlich brauchen, um die Größe N_c zu erreichen.



Untersuchen Sie dieses Verhalten, indem Sie für ein N_c Ihrer Wahl verschiedene Werte von p_t durchspielen. Die Graphik zeigt eine offensichtliche Vermutung, aber vielleicht ist ja alles ganz anders? Damit man in endlicher Zeit auch ein Resultat bekommt, sollte man N_c nicht zu groß wählen und sich von $p_t = 0.5$ schrittweise zu kleineren p_t vorarbeiten. Die Rechnungen für $p_t > 0.5$ gehen sehr schnell.

Ihre Untersuchungen sollten Sie nicht mit Mathematica durchführen; das ist zu langsam. Sie können den Zufallszahlengenerator Mersenne-Twister `mt19937ar.c` benutzen, den Sie in stud.ip finden.
Wenn Sie sich für den wissenschaftlichen Hintergrund interessieren, können Sie ein Blick in die folgende (auch in stud.ip abgelegte) Publikation werfen:
Luis H. Cisneros and Timothy J. Newman, *Quantifying metastatic inefficiency: rare genotypes versus rare dynamics*, Physical Biology 11 (2014) 046003.