Physik am Komputer 20. Juni 2010

Übung 05

Michael Kopp

Aufgabe 1: Kongruentieller Generator Der Algorithmus ist in `konrg01.cpp` realisiert.

Der Vorteil bei kleinem r ist, dass Ziffern auch doppelt vorkommen können: Möchte man eine Menge von N Zufallszahlen und ist $N > r$, so kommt jede Zahl nur einmal vor; das ist manchmal nicht unbedingt wünschenswert – manchmal aber auch nicht...

Was ich für problematisch halte, ist, dass r keine Primzahl ist: Es könnte sich irgendwann $0 = z_n = z_{n+1} = ...$ ergeben.

Bei zu vielen Punkten im Gnuplot-Diagramm ist nichts mehr zu erkennen, weil man nichts mehr sehen kann... Verwendet man dagegen 2 Dimensionen kann man noch ohne Probleme folgendes feststellen: Bei den Werten aus *Numerical Recipies* ist zwar eine gewisse Struktur festzustellen – bei leicht abweichenden jedoch eine viel stärkere. Vgl dazu Abb. 1, 2 und 3.

Verwendet man einen etwas langsameren PS-Rasterer / langsamen PC, so kann man sehen, wie sich die Zufallszahlen geordnet an ihre Positionen begeben...

Aufgabe 2: Gleichverteilung der Zufallszahlen Ich habe ein eigenes Histogramm-Programm in `histo01.cpp` geschrieben.

In Abb. 4 sieht man die Verteilung der Zufallszahlen aus Aufgabe 1.

Ich habe `histo01.cpp` zu `histo03b.cpp` geändert um Daten aus einer Datei einlesen zu können und diese verarbeiten zu können.

Das Programm `kirkpatrick.cpp` nimmt als Parameter die Anzahl an zu generierenden Zufallszahlen entgegen, `histo03b.cpp` dito. Mit

```
g++ -o histo03b histo03b.cpp
g++ -o kirkpatrick kirkpatrick.cpp PTSRandom.cc
./kirkpatrick 1000000 | ./histo03b 1000000
```

kann man so via Gnuplot Abb. 5 erzeugen.

Die mit diesem Zufallsgenerator erstellten Zufallszahlen sind ... komisch Verteilt: Nicht gleichmäßig um ihren Mittelwert sondern kleine Zahlen weniger häufig als große und die Verteilung ist linear.

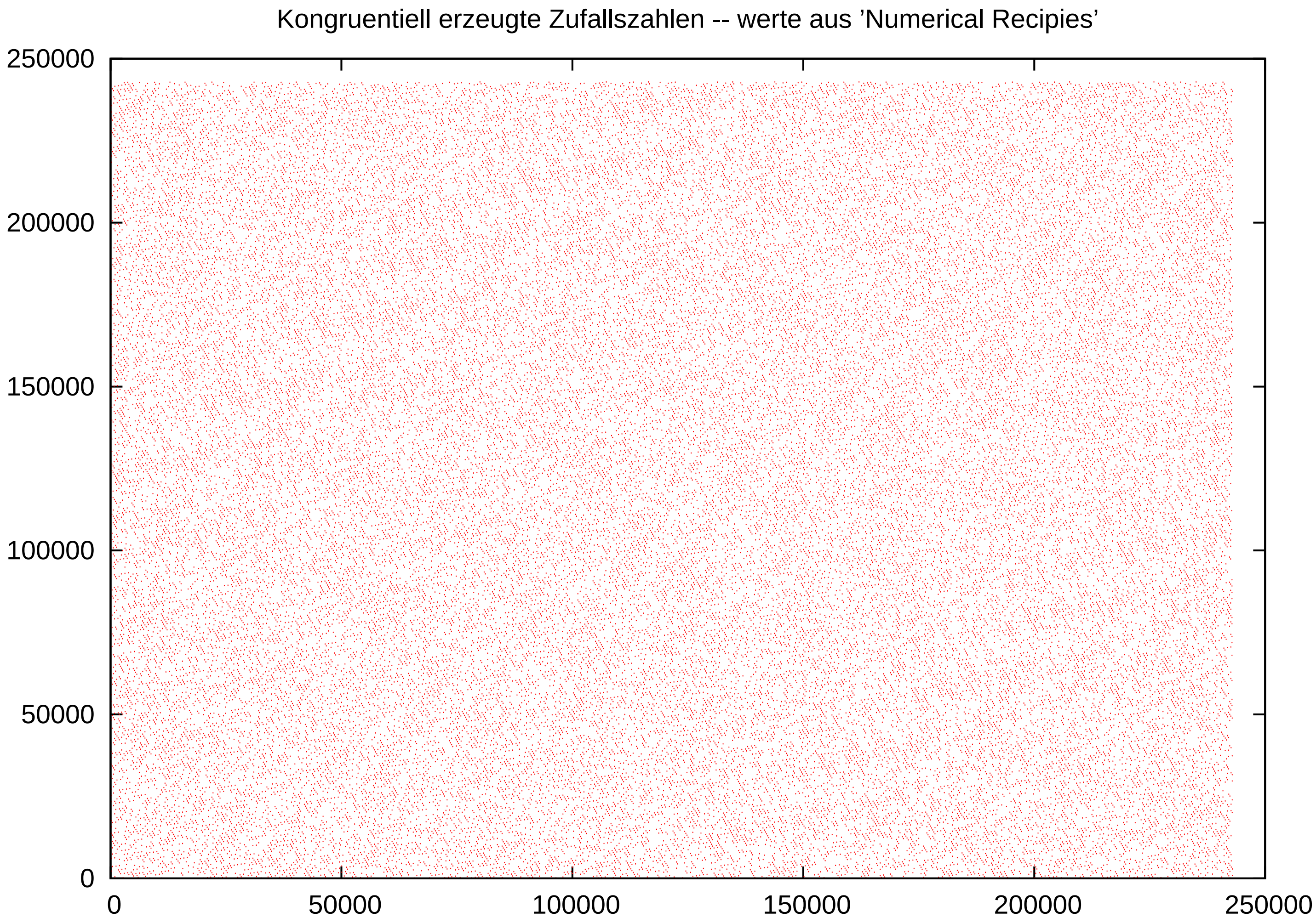


Abbildung 1: Zufallszahlen kongruentieller Generatoren – angegebener Wert $r = 243000$ verwendet

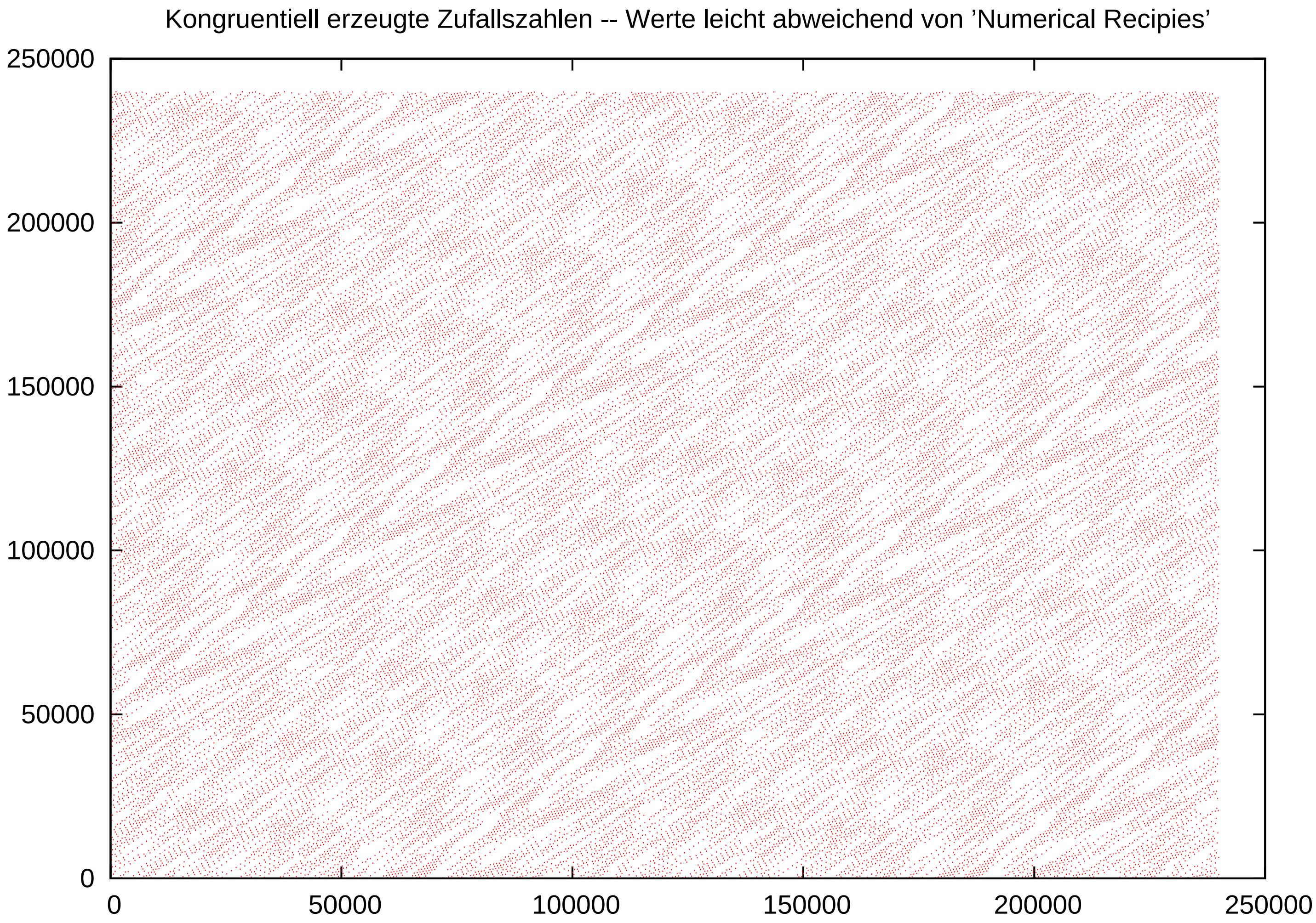


Abbildung 2: Zufallszahlen kongruentieller Generatoren – vergleichbarer Wert $r = 240000$ verwendet

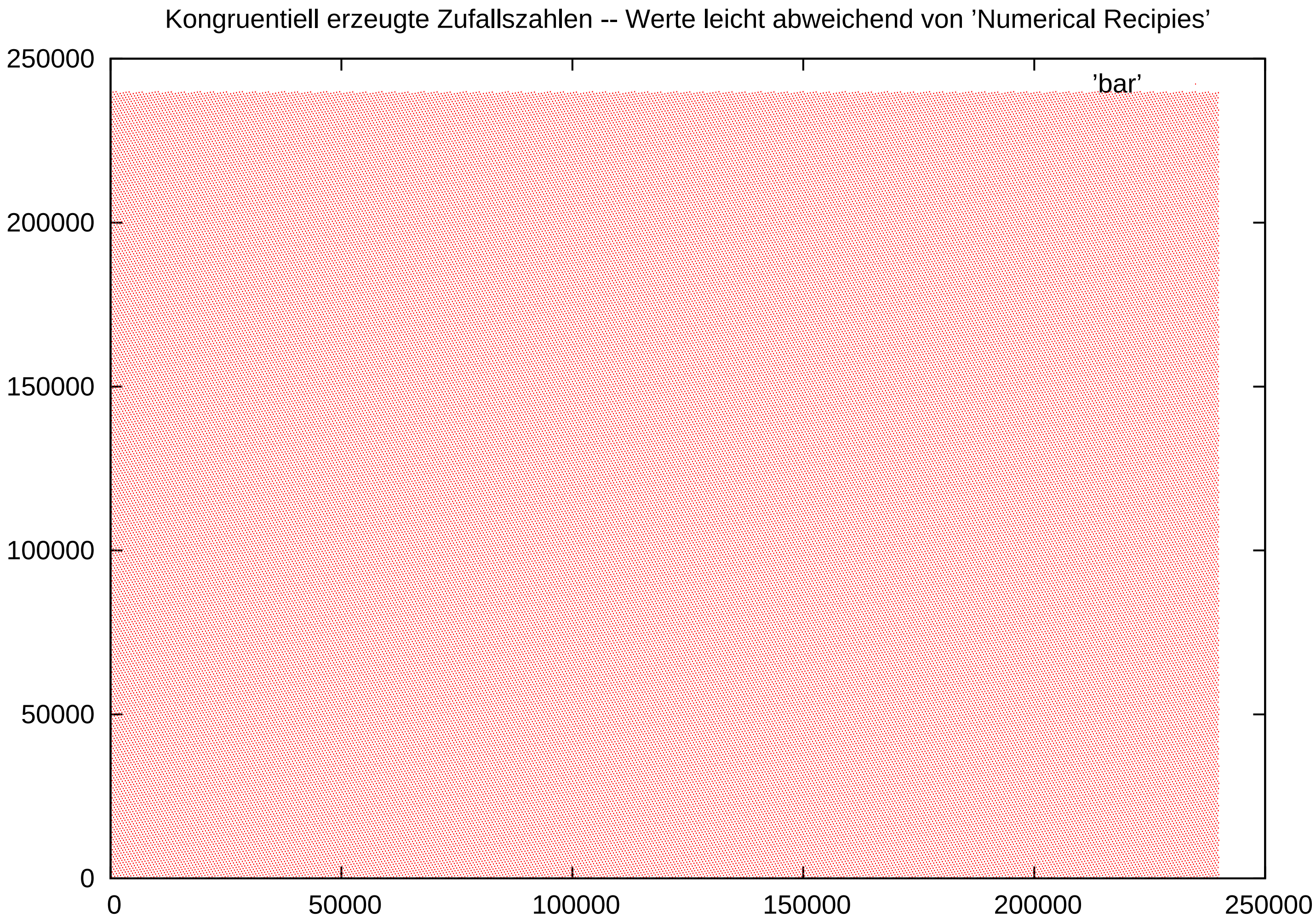


Abbildung 3: Zufallszahlen kongruentieller Generatoren – vergleichbarer Wert $r = 240000$ verwendet mit mehr Zahlen

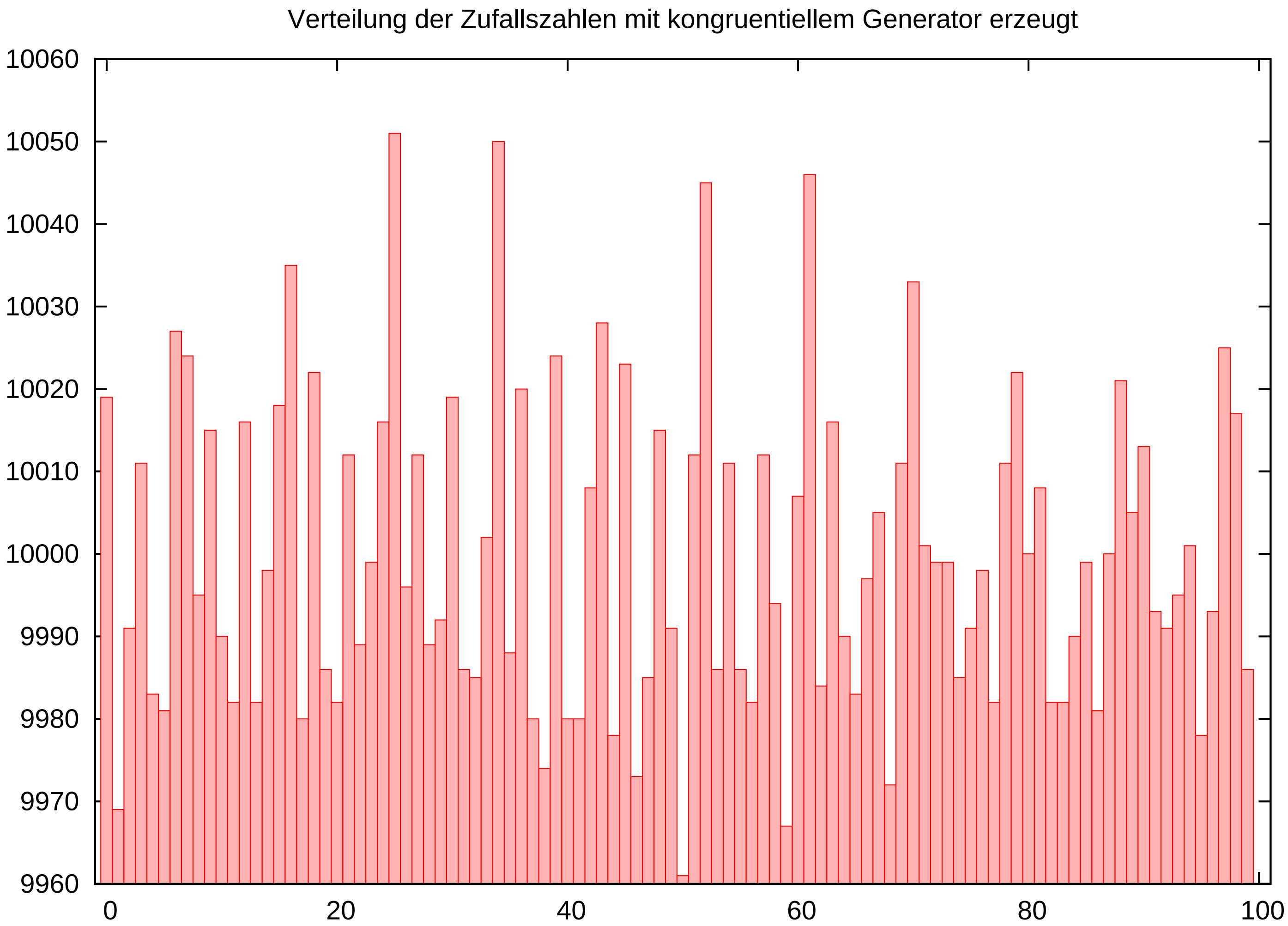


Abbildung 4: Verteilung von Zufallszahlen kongruentieller Generatoren – angegebener Wert $r = 243000$ verwende

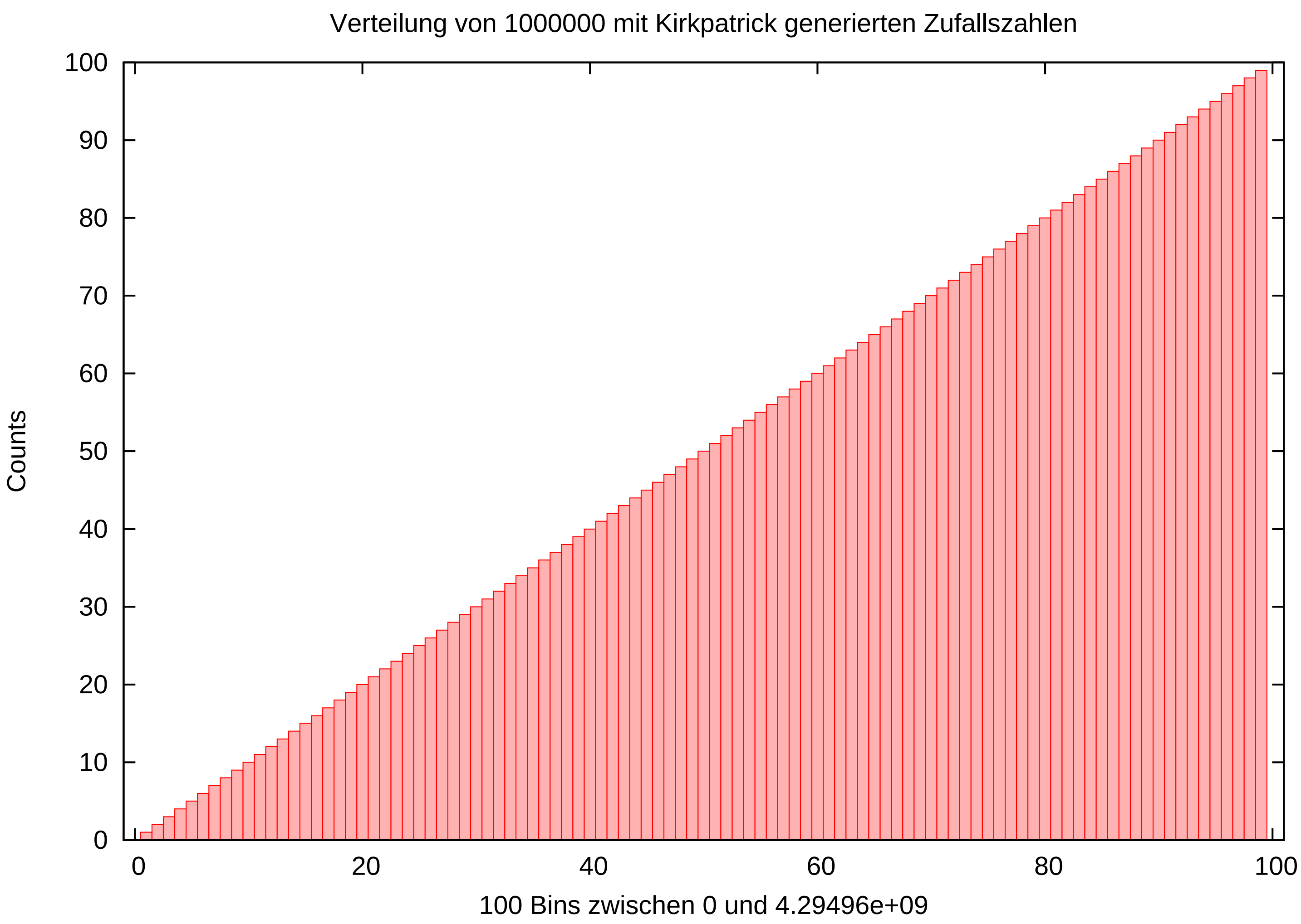


Abbildung 5: Verteilung von 1000000 mit Kirkpatrick generierten Zufallszahlen

Aufgabe 3: Random Walk Wie in der Übung abgesprochen verwende ich in `randomw01.cpp` den Zufallsgenerator aus `cstdlib`.

Im Quelltext ist ein alternativer Weg aufgezeigt, mit dem man auch Vektoren zulassen kann, bei denen beide Einträge von 0 verschieden sind.

Interessant ist: Verwendet man für die kleinen Zufallszahlen zwischen 0 und 3 rand() % 4 , so bekommt man für kurze Wege ziemlich langweilige Strecken: sie sind nur hin und her: Die Zufallszahlen durch Modulo sind sehr periodisch, wärend man mit (4 * rand()) / RAND_MAX bessere Zufallszahleb bekommt.

Die mit Gnuplot erzeugten Bilder der Walks sind in Abb. 6 zu finden; Zum Vergleich ist eine echte Zufallsbewegung – ein Kolloidteilchen – in Abb 7. Der Unterschied zwischen den beiden Abbbildungen ist, dass das simulierte Teilchen stets nur einen bestimmten Abstand gehen kann. Das Kolloidteilchen kann in einem Zeitschritt ein wenig springen – außerdem kann es einen Moment nicht von dem Messaufbau erfasst werden...

Mit `randomw02.cpp` sieht man, dass die Teilchen sich beim Random Walk im Mittel nicht vom Ausgangsort entfernen:

$$\langle \boldsymbol{R} \rangle = (0.151, 1.759) \ , \tag{1}$$

wobei die verschiedenen Endpunkte sehr breit verteilt sind:

$$\langle \boldsymbol{R}^2 \rangle = (499.113, 495.643) \ . \tag{2}$$

Um das Histogramm zu erzeugen, kann man `histo03.cpp` verwenden; hierzu muss man die Zahl der Moves eingeben. Mit bspw.

```
g++ -o histo03 histo03.cpp
g++ -o randomw02 randomw02.cpp
./randomw02 | ./histo03 1000
```

kann man so ein schönes Histogramm erzeugen; mit Gnuplot geplottet sind die Daten in Abb. 8 zu finden.

Verwendet man 500 000 Walks, so bekommt man eine schönere Gauß-Verteilung; dies ist in Abb. 9 angefittet.

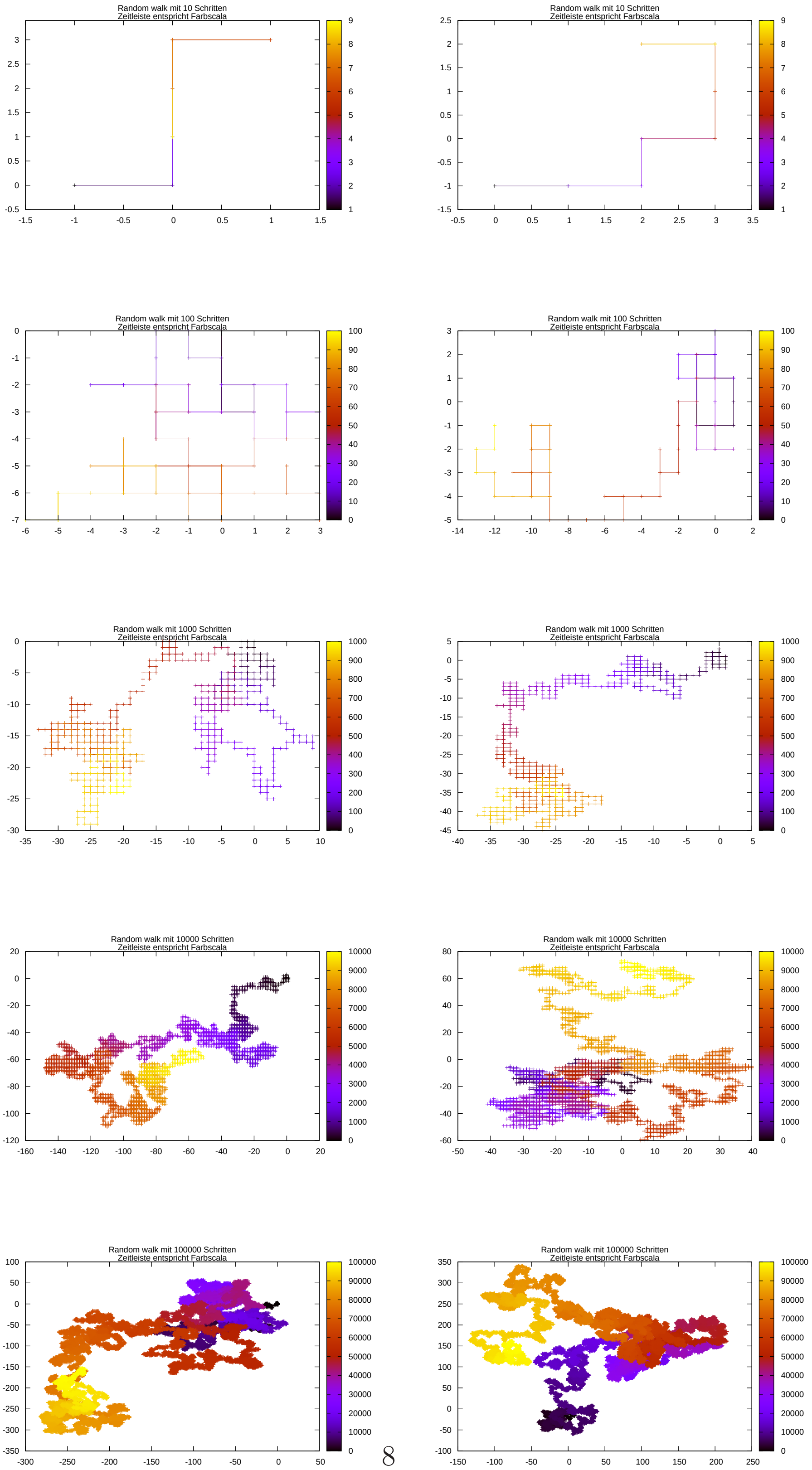
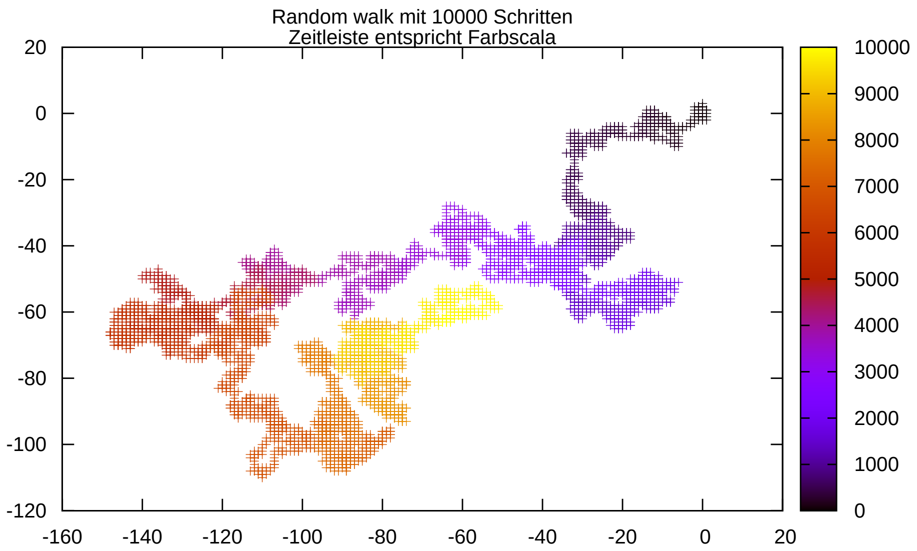


Abbildung 6: Random Walks; von oben nach unten $10^1, ..., 10^5$ Schritte



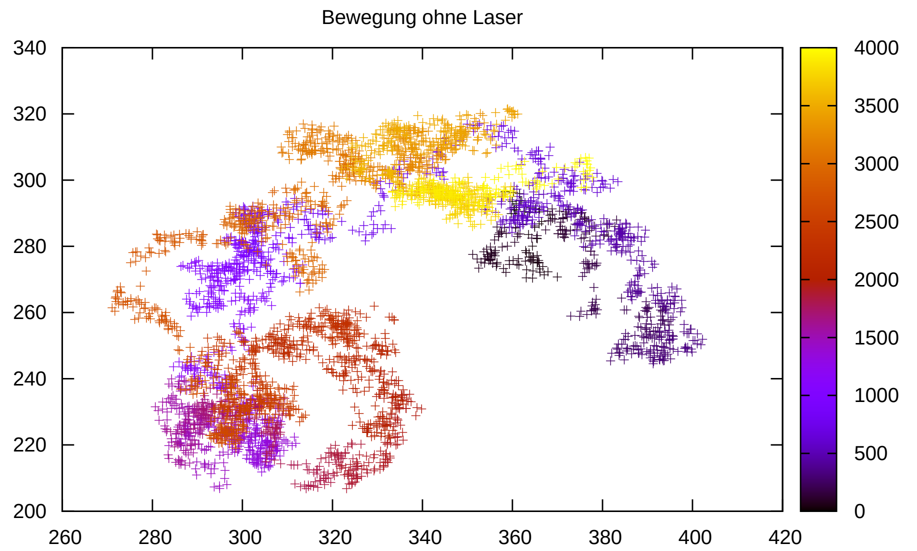


Abbildung 7: Vergleich der Simulierten und einer realen Bewegung mit Ranom Walk

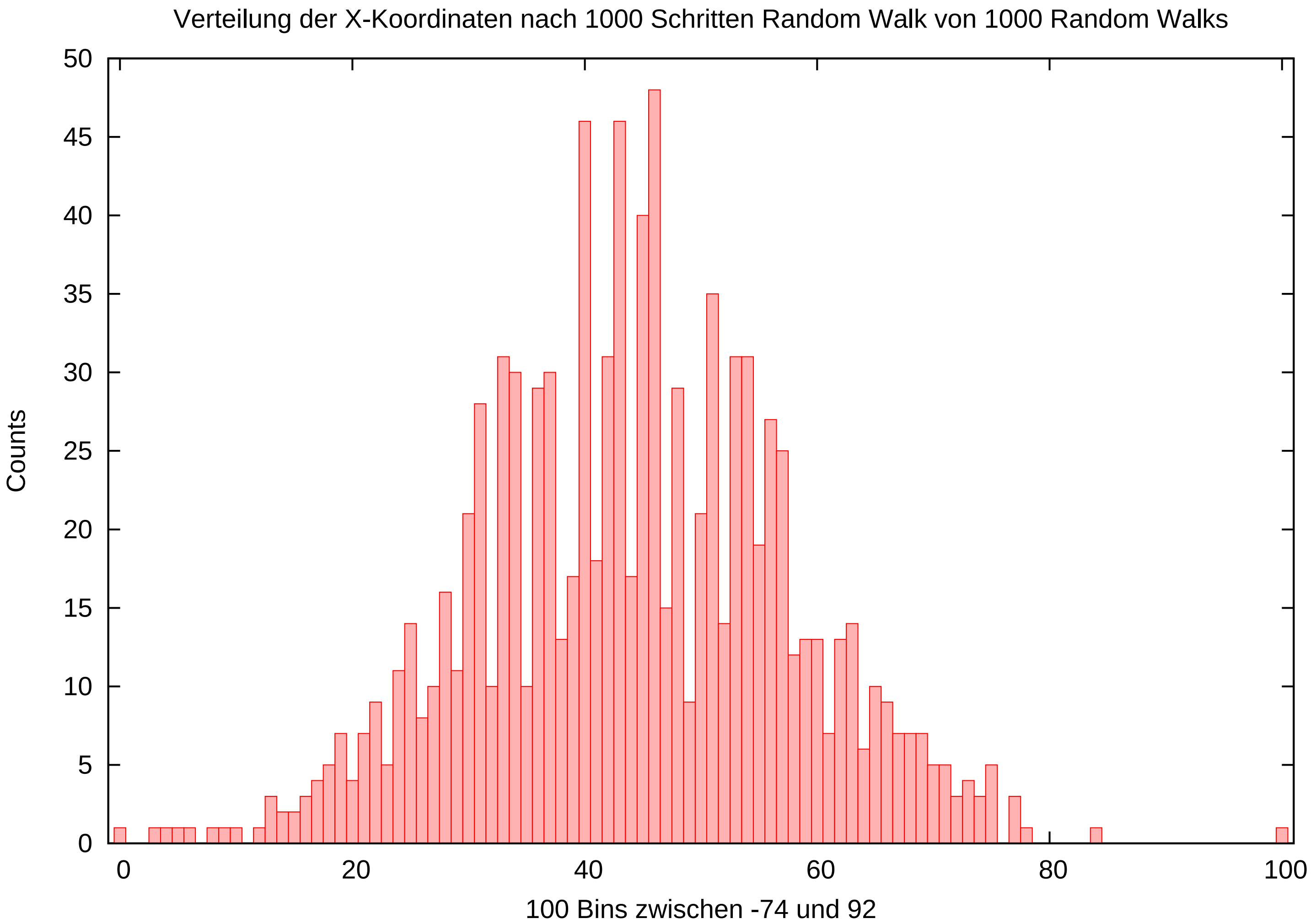


Abbildung 8: Verteilung der X-Koordinate nach 1000 Random Walks

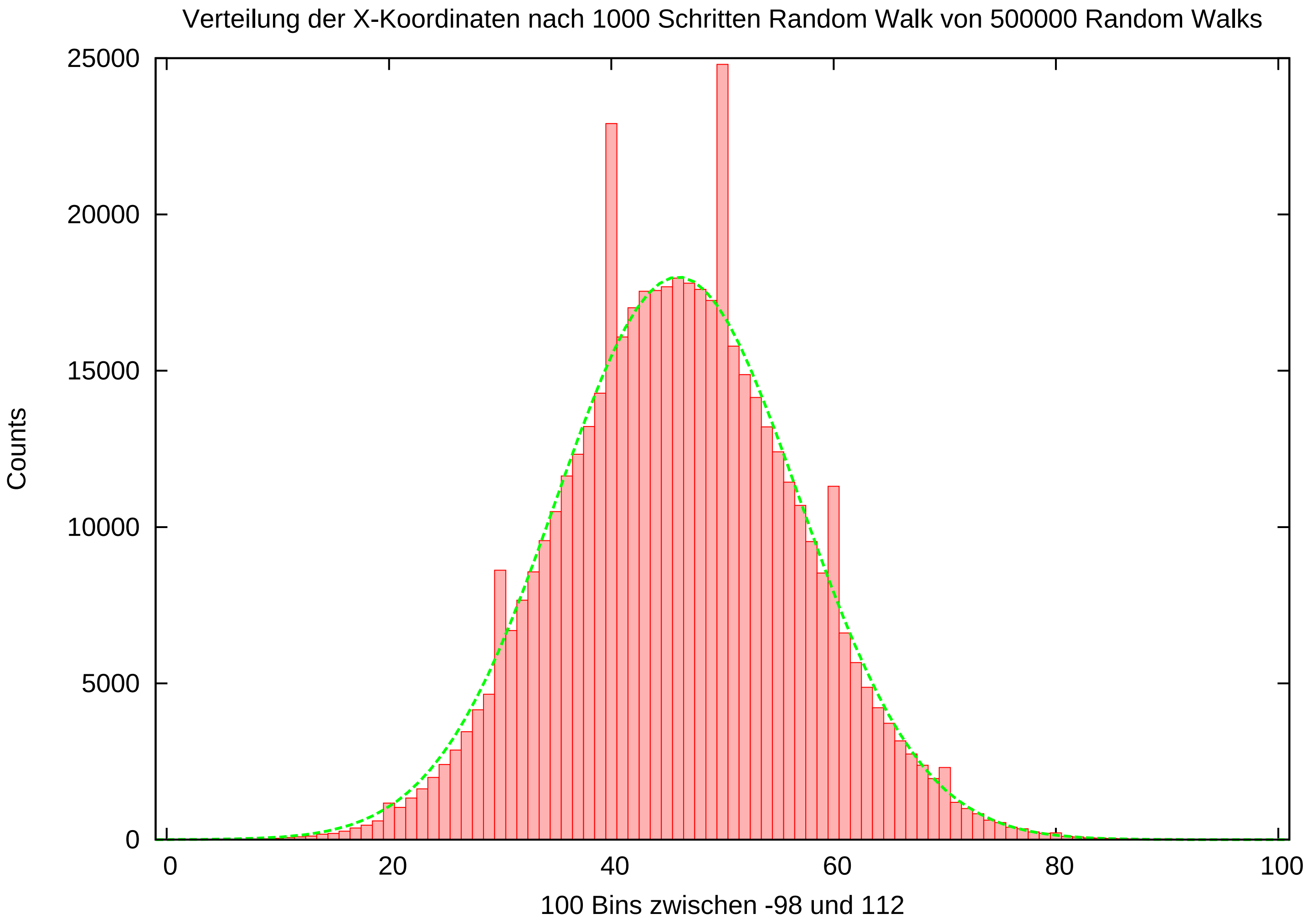


Abbildung 9: Verteilung der X-Koordinate nach 500000 Random Walks; Gaußglocke angefittet