Schriftliche Ausarbeitung zum Vortrag

Monte-Carlo-Simulation

Universität Hamburg
Fachbereich Informatik

Johannes Schlundt

20. März 2013

Inhaltsverzeichnis

1 Motivation

Monte-Carlo-Simulation ist eine Methode die dazu genutzt wird um eine Annäherung einer Simulation oder Berechnung zu unternehmen, mit den Vorteilen einer schnelleren Berechnung und einer guten Annäherung zum exaktem Resultat. Es ist geeignet eine Lösung zu liefern, wenn man viele Freiheiten in denn Eingabe Variablen hat. Da die Monte Carlo Simulation aus einem Wertebereich zufällig wählt, einsetzt und so zum Ergebnis kommt. Monte-Carlo-Simulation ist ein Verfahren aus der Stochastik, indem eine große Anzahl von Zufallsexperimenten als Basis dient. So wird es oft in physikalische und mathematischen Umgebungen angewandt. Monte-Carlo-Simulation löst dabei numerisch die Probleme, die durch einen deterministische Algorithmen schwer oder gar nicht lösbar wären. Da Monte-Carlo-Simulation auf eine große Anzahl von Zufallsexperimenten Basiert, ist diese Methode eher theoretischer Natur.

Ein groben Überblick wobei man die Monte Carlo Simulation anwenden kann.

- Analytisch unlösbare Probleme rein mathematischer Herkunft
  z.B Berechnung des Integrals einer Funktion
- Verteilungseigenschaften von Zufallsvariablen unbekannten Verteilungstyps
- die Nachbildung von komplexen Prozessen, die nicht geradlinig analysiert werden können
  z.B Wetter und Klima der Erde
- Unsicherheiten und statistische Verhalten simulieren
  z.b wo ein Tischtennisball landet, wenn er durch ein Nagelbrett fällt

2 Geschichte

Im Jahre 1934 war Enrico Fermi der erste der die Monte-Carlo-Simulation nutzte, bevor es einen Namen hatte *(Zeitzeuge berichtet davon[1])*. Er hat es für das Simulieren einer Neutronenstreuung genutzt. Im Jahre 1946 kam dann Stan Ulam drauf. Damit löste er sein Problem bei seinem Kartenspiel.

Stan Ulam: Kartenspiel

Ulam arbeitete im Jahre 1946 in einem geheimen Projekt Namens „Manhattan Project“. Als Ulam aufgrund einer Krankheit im Bett lag, spielte

er dabei gerne Canfield Solitär. Dabei stellte er sich die frage mit welcher Wahrscheinlichkeit, kann er in Canfield Solitär mit 52 Karten gewinnen. Am Anfang versuchte er alle mögliche Kombinationen durchzugehen. Dabei fiel im auf, dass es sehr Zeitaufwendig ist alle Kombinationen durchzugehen. Als nächstes versuchte er 100 Zufällige spiele zu spielen, dabei zählte Ulam die Gewonnen Spiele. Zum Schluss dividierte er die Anzahl der gewonnen Spiele durch 100. Dabei fiel ihm auf, dass er eine Annäherung eines schweren Kombinationsproblems in ein einfacheres Problem umgewandelt hat.[4] Dies berichtete er John Von Neumann, der auch im „Manhattan Project"tätig war. Von Neumann erkannte sofort das Potenzial dieser Methode, gemeinsame entwickelten sie auf dieser Basis mehrere Algorithmen und lösten damit viele Probleme in der Wissenschaft.

Namensherkunft *Bei meiner Recherche wurde oft erwähnt, dass John Von Neumann auf denn Namen kam. Es kommt von einer Anspielung auf Ulam's Onkel der sich im Gleichnamigen Casino Geld zum Spielen leihen würde, das Monte Carlo Casino.*

3 Monte Carlo Simulation

3.1 Definition

Wann eine Monte Carlo Simulation vorliegt ist meist nicht klar. Ich habe auf diversen Internet Seiten keine klare Eingrenzung gefunden. Eine Amerikanischer Professor namens Shomo Sawilowsky[3] erklärt es an folgenden Beispielen*(Ich versuche es schriftlich zu übersetzten)* :

Simulation

Simulation ist eine fiktive Darstellung der Realität.
Wir generieren **eine** zufällige Zahl eines Zufallszahlengenerator. Dies Zahl ist einheitlich in einem Bereich von $(0, 1]$ verteilt und kann als eine geworfene Münze angesehen werden: Wenn sie kleiner oder gleich 0.50 ist, dann ist es Kopf, aber wenn der Wert 0.50 überschreitet ist es Zahl.
Das genannte Beispiel ist eine Simulation, aber keine Monte Carlo Simulation.

Monte Carlo Methode

Eine Monte Carlo Methode ist eine Technik die benutzt wird, um ein mathematisches oder statistisches Problem zu lösen.

Eine irreguläre Figur auf einem einheitlichen Quadrat wird mit Dartpfeilen beworfen und dann wird das Verhältnis von Figur getroffenen zu insgesamt geworfenen Pfeilen berechnet.
Das ist ein Beispiel für eine Monte Carlo Methode auf einen eingegrenzten Bereich, aber keine Simulation.

Monte Carlo Simulation

Eine Monte Carlo Simulation nutzt das mehrmalige wiederholen des Versuchs aus, um ungewollte Phänomene oder Verhalten zu entfernen.
Wir generieren **eine große** Anzahl von Zufallszahlen. Die einheitlich aus dem Bereich von $(0, 1]$ kommen. Ist der Wert kleiner oder gleich 0.50 ist es Kopf und wenn größer 0.50 Zahl.
Das ist ein Beispiel für eine Monte Carlo Simulation die das Verhalten einer mehrmalige geworfenen Münze Simuliert.

Zwei weitere Personen, die Karlos und Whitlock heißen, haben erkannt das Monte Carlo Simulation nicht eindeutig eingegrenzt werden kann. Sie haben als Beispiel die Emission eines Radioaktiven Atoms genannt."Man kann dies direkt Simulieren oder das verhalten kann mit Stochastischen Gleichungen beschrieben und mit Monte Carlo Methode gelöst werden. *In der Tat, der selbe Computer Code kann gleichzeitig als 'natürliche Simulation' oder als eine Lösung der Gleichung für eine natürliche Abtastung betrachtet werden."*[3]

3.2 Allgemeine Eingrenzung

In denn meisten Erklärungen findet man, dass Monte Carlo Simulation in 4 grobe Schritte unterteilt werden kann. Die 4 Schritte werden hier aufgezählt, darauf hin nehme ich ein einfaches Beispiel und erkläre die Schritte kurz daran. Danach wird ein weiteres Beispiel diese 4 Schritte deutlich machen.

1. Wir grenzen die Möglichen Eingaben ein.
2. Wir generieren zufällige Eingaben aus denn möglichen Eingabebereich.
3. Danach führen wir eine deterministische Berechnung mit denn vorhanden eingaben aus.
4. Zuletzt werten wir das Ergebnis aus.

3.3 Beispiel: PI Approximation

Als ein einfaches Beispiel wird gerne die PI Approximation genannt. Diesen Versuch habe ich zur Verdeutlichung nachgestellt.
Hierbei wird als erstes eine Quadratische Fläche genommen und ein Viertelkreis in ihm gezeichnet (Siehe Abbildung 1).
(Schritt 1: Wir grenzen die Möglichen Eingaben ein.)

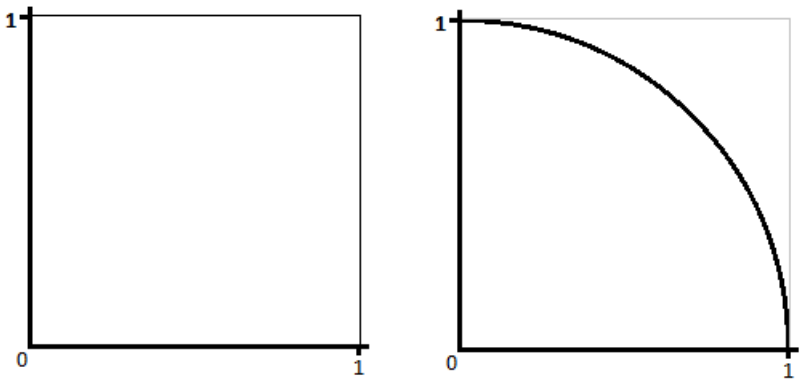


Abbildung 1: Vorbereitung

Danach werden gleichgroße Punkte $(x, y) \in \mathbb{R} : x, y = (0..1]$ zufällig innerhalb der Fläche generierte und jeder Punkt der innerhalb des Viertelkreises landet wird gezählt(Siehe Abbildung 2). *(Schritt 2: Wir generieren zufällige Eingaben aus denn möglichen Eingabebereich.)*
Da die Punkte zufällige Koordinaten bilden, auf die die Gleichverteilung zutrifft, kann man von Regen sprechen.

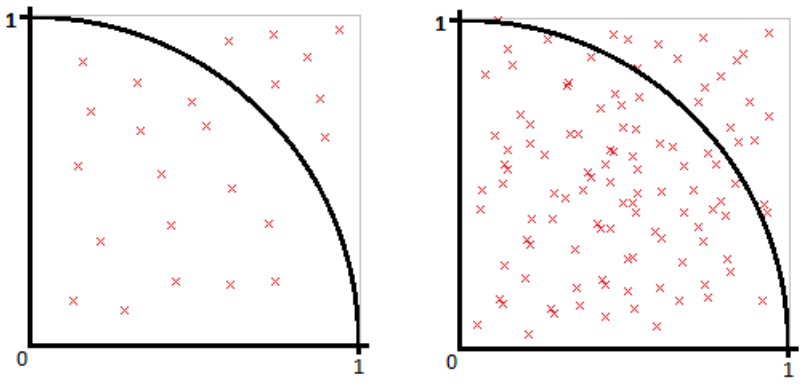


Abbildung 2: „Regnen“lassen

Zum Schluss werden die gezählte Punkte durch die gesamte Anzahl von erstellten Punkten dividiert *(Schritt 3: Danach führen wir eine deterministische Berechnung mit denn vorhanden eingaben aus.)*:

$$\frac{Innerhalb\ des\ Viertelkreises}{Gesamte\ Anzahl\ der\ Punkte} = \frac{\pi}{4}$$

Dabei bekommt man das Verhältnis des Viertelkreises zur Quadratische Fläche. Da wir nur ein Viertelkreisfläche haben, aber die gesamte Kreiszahl haben möchten, multiplizieren wir es noch mit 4.*(Schritt 4: Zuletzt werten wir das Ergebnis aus.)*

$$\frac{Innerhalb\ des\ Viertelkreises}{Gesamte\ Anzahl\ der\ Punkte} \cdot 4 = \pi$$

Ich habe diese Berechnung mit Ansteigender Anzahl von Punkten durchgeführt. Dabei ist mir aufgefallen, je höher die Anzahl der Zufälligen Punkten ist, umso genauer wird die PI Approximation(Siehe Abbildung 3).

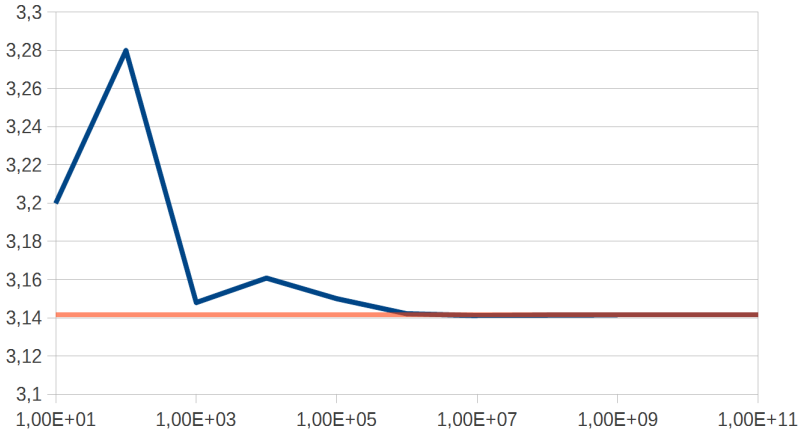


Abbildung 3: Ansteigende Tropfenzahl

Erklärung *Die Blaue Linie steht für meine Berechnungen. Jeder Punkt ist Logarithmisch zu der Anzahl der Punkten die ich erstellt habe. Es fängt von 10 Punkten an und geht bis zu 100 Milliarden Punkten. Die Rote Linie ist der Programmierte PI Wert in LibreOffice Calc.*
Meine Annäherung ist auf 4 Stellen nach dem Komma genau, nämlich ***3,1415****8731*
Die Rote Linie, also der genaue Wert hat als $\pi = 3,1415926536$

3.4 Ein Weiteres Beispiel: Nagelbrett

Wir nehmen ein Nagelbrett und lassen oberhalb einen Tischtennisball hineinfallen(Abbildung4). Jetzt gibt es einmal die Möglichkeit über die Mathematik mithilfe von Gauß und Pascal zu berechnen, mit welcher Wahrscheinlichkeit der Ball in welches Fach fallen wird. Eine weitere Möglichkeit wäre die Monte-Carlo-Simulation. Dabei lassen wir denn Ball mehrmals durch das Nagelbrett fallen und schauen uns an wo er landet. An jedem Nagel hat der Ball dann 50:50 Wahrscheinlichkeit nach links oder rechts zu fallen, also ein wahrscheinlichkeitsgewichteten Weg. Diesen Versuch wiederholen wir einige Tausende male, dadurch erhalten wir eine Verteilung, die in etwa der genauen mathematischen Berechnung entspricht.

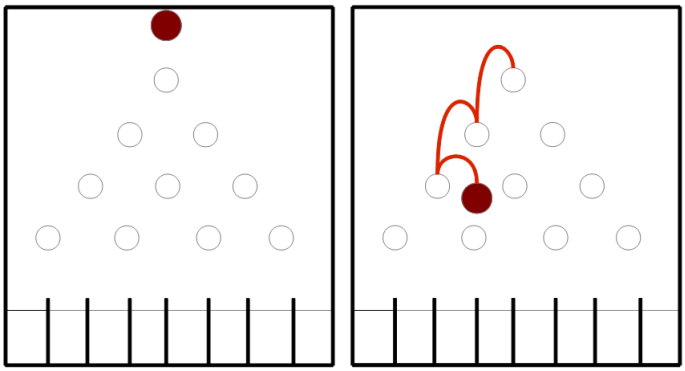


Abbildung 4: Nagelbrett

Die vier Schritte der Monte Carlo Simulation zum nachvollziehen.

1. Wir grenzen die Möglichen Eingaben ein.
   Je Stufe fällt der Ball links oder rechts.

2. Wir generieren zufällige Eingaben aus denn möglichen Eingabebereich.
   Der Ball fällt durch Zufall nach links oder rechts.

3. Danach führen wir eine deterministische Berechnung mit denn vorhanden eingaben aus.
   Wir Zählen wie oft er in welches Fach fällt und dividieren durch die Gesamtanzahl der Durchläufe.

4. Zuletzt werten wir das Ergebnis aus.
   Aus dem Ergebnis entwickelt man eine Verteilung. In welches Fach mit welcher Wahrscheinlichkeit der Ball fällt.

4 Exkurs: Zufallszahlengenerator

Als die Monte Carlo Simulation entwickelt wurde waren die Zufallszahlen das Herzstück einer Monte-Carlo- Simulation. Da es Anfang der 1946 Jahren noch keine Zufallszahlengenerator gab, hat John Von Neumann provisorisch einen entwickelt (Mittquadratmethode[5]). Der in jeder Iteration die neu erstellt Zahl quadriert und die mittleren Werte übernimmt, um damit die nächste Iteration zu entwickeln. Dieser Zufallszahlengenerator hat in denn Anfangszeiten für Monte Carlo Simulation noch gereicht, aber die großen Nachteile sind, z.B. das der Generator nach einigen Iteration einen Nullfolge bildet oder die generierten Zahlen nicht gleichmäßig verteilt liegen (Diese Ungleichverteilung kann sich im Beispiel der PI-Approximation negativ auswirken, wenn sich die Punkte vermehrt im Viertelkreis bilden, erhält man ein verfälschtes Resultat).
Die heutigen deterministischen Zufallszahlengeneratoren werden in vier Güterstufen eingeteilt, wohingegen die ersten zwei für Monte Carlo vollkommen ausreichen sind. Da die letzten zwei mit höherem Rechenaufwand Arbeiten und eher für die Kryptologie gedacht sind.

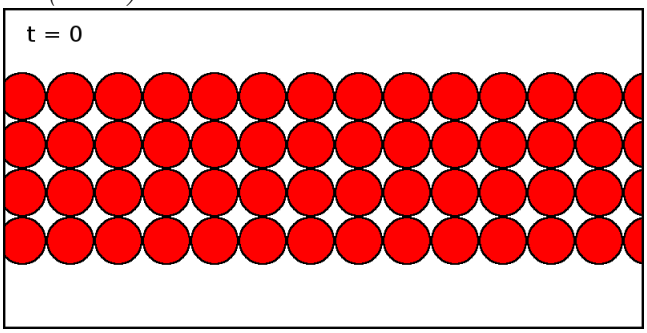
- K1 - Eine Folge von Zufallszahlen mit einer geringen Wahrscheinlichkeit von identischen aufeinander folgenden Zahlen.
- K2 - Eine Folge von Zahlen, die nicht von einem „wahren Zufallszahl“in einem statistischem Test unterschiedenen werden kann.
- K3 - Für jeden Angreifer sollte es unmöglich sein von jeder gegebener Sub-Sequenz, frühere oder zukünftige Werte in der Folge, noch einen inneren Zustand des Generators zu berechnen oder sonst zu erhalten.
- K4 - Für einen Angreifer sollte es unmöglich sein aus einem inneren Zustand des Generators, etwaige frühere Nummern in der Sequenz oder alle früheren inneren Generator Zustände zu berechnen oder zu erhalten.

Hinweis *Zu beachten ist das die Monte-Carlo-Simulation auf einen Zufallszahlengenerator zugreift und somit das Verhalten nicht deterministisch sein kann. Da die heutigen Rechner mit einem Zufallszahlengenerator arbeiten der mit einem Vordefinierten Seedwert startet. Man kann denn Versuch natürlich auch bei einem anderen Rechner zum selben Verhalten bringen, wenn man denn selben Seedwert verwendet. Das Ergebnis, im diesem Fall die Verteilung, sollte sich aber dennoch bei unterschiedlichen Seedwerten kaum unterscheiden.*

5 kinetischer Monte Carlo

Kinetischer Monte Carlo (kMC) ist eine Monte Carlo Simulation die häufig in der Physik genutzt wird. Mit kinetische Monte Carlo können wir Natürliche zeitabhängige Entwicklungen simulieren. Der Algorithmus hat immer noch Ähnlichkeiten mit dem typischen Aufbau der Monte Carlo Simulation, aber wird um einige Elemente erweitert.
Zum leichteren Verständnis, erkläre ich den kinetischen Monte Carlo mit dem Beispiel des Plateau-Rayleight Instability[7]. Dabei transformiert sich ein Zylindrisches-Gebilde in einzelne Tropfen.*Zu beachten ist, dass ich die Bilder in Freihand erstellt habe und nur zur Demonstration dienen*

1. Wir setzten die Zeit auf $t = 0$.
   Zu beachten ist, das man innerhalb des kMC in Monte Carlo Schritten(MCS) arbeitet.



2. Wir erstellen eine Liste aller beweglichen Objekte im System r_i.
   In meiner Zeichnung sind die Zustände wie sich die einzelnen Atome verhalten können. Die Atome können sich in allen Richtungen Bewegen, doch sie gewinnen an Trägheit je mehr unmittelbare Nachbarn sie haben.

3. Wir berechnen eine kumulative Funktion

$$R_i = \sum_{j=1}^{i} r_j : i = 1, ..., N$$

   N stellt die Gesamtzahl der Übergänge dar.

4. Wir wählen eine zufällige Zahl $u \in (0, 1]$ aus.

5. Wir wählen das Event i, dass wir mithilfe der Funktion

$$R_{i-1} < uR_N \leq R_i$$

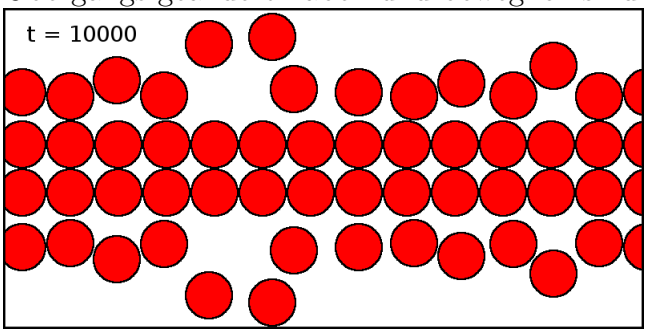
   finden.

6. Wir führen Event i aus.
   Das sind in meinem Beispiel die gewählte Richtung.

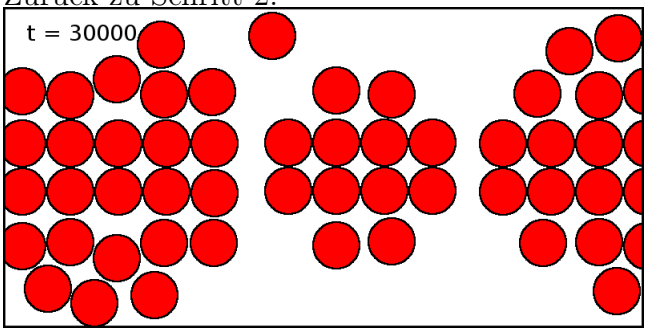
7. Wir wählen eine neue zufällige Zahl $u' \in (0, 1]$ aus.

8. Wir aktualisieren die Zeit mit $t = t + \Delta t$ $\Delta t = R_N^{-1} ln(1/u')$
   Man kann auch ein anderes Δt wählen, wenn das System ein anderes Bevorzugt

9. Wir Berechnen die Liste aller Objekte in r_i neu, die sich während des Übergangs geändert haben und beweglich sind.



10. Zurück zu Schritt 2.



Man kann mein gezeigtes Beispiel Animiert auf der Internet Seite http://www.roentzsch.org/Rayle
[Letzter zugriff:18.03.2013] angucken.
Ich hab es sehr einfach gehalten. Auf der Internetseite sind mehr Infos vorhanden wie der Autor der Seite das Beispiel Realisiert hat.

Literatur

[1] *Metropolis, Monte Carlo and the MANIAC*, Los Alamos Science, Nr. 14, 1986, S. 96–108, 1986.

[2] Monte_Carlo_methode

[3] Wikipedia: http://en.wikipedia.org/wiki/Monte_Carlo_method#Definitions

[4] *An Introduction to MCMC for Machine Learning, Andrieu, de Freitas, Doucet & Jardon*[Letzter zugriff: 14.03.13]

[5] http://de.wikipedia.org/wiki/Mittquadratmethode

[6] Wikipedia: http://en.wikipedia.org/wiki/Pseudorandom_number_generator

[7] Plateau-Rayleigh Instability : http://www.roentzsch.org/Rayleigh/

[8] Wikipedia: http://de.wikipedia.org/wiki/Monte-Carlo-Simulation

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <math.h>
#include <limits.h>

/*Zufallszahlengenerator von 0 bis 1*/
double zufall(){
   double number = rand()/((double)RAND_MAX + 1);
   return number;
}

/* Oginal Quellcode aus Wikipedia
 * http://de.wikipedia.org/wiki/Kreiszahl#Statistische_Bestimmung
 * Java Code auf C geaendert und fuer eine Logarithmische Ausgabe gesorgt, um es
     einfacher auszuwerten.
 */
int main (int argc, char *argv[])
{
  long tropfenzahl = 100000000000;  //Wie viele Punkte generiert werden sollen.
  srand ( time(NULL) );
  double pi = 0.0;
  long innerhalb = 0;
  long gesamt = ta;
  long zaehler = 0;

  double dotx = 0.0;
  double doty = 0.0;

  while (ta > 0) {
    dotx = zufall();
    doty = zufall();
    if (dotx * dotx + doty * doty <= 1.0) {
      innerhalb++;
    }
    ta--;
    zaehler++;
    if ((zaehler == 10)||
    (zaehler == 100)||
    (zaehler == 1000)||
    (zaehler == 10000)||
    (zaehler == 100000)||
    (zaehler == 1000000)||
    (zaehler == 10000000)||
    (zaehler == 100000000)||
    (zaehler == 1000000000)||
    (zaehler == 10000000000)||
    (zaehler == 100000000000)){
       pi = 4.0 * innerhalb/(gesamt - ta);
       printf("PI=%.8f␣n=%ld\n", pi, (gesamt - ta));}
  }
  return 0;
}
```

Mein Quelltext in C zu der PI-Approximation.