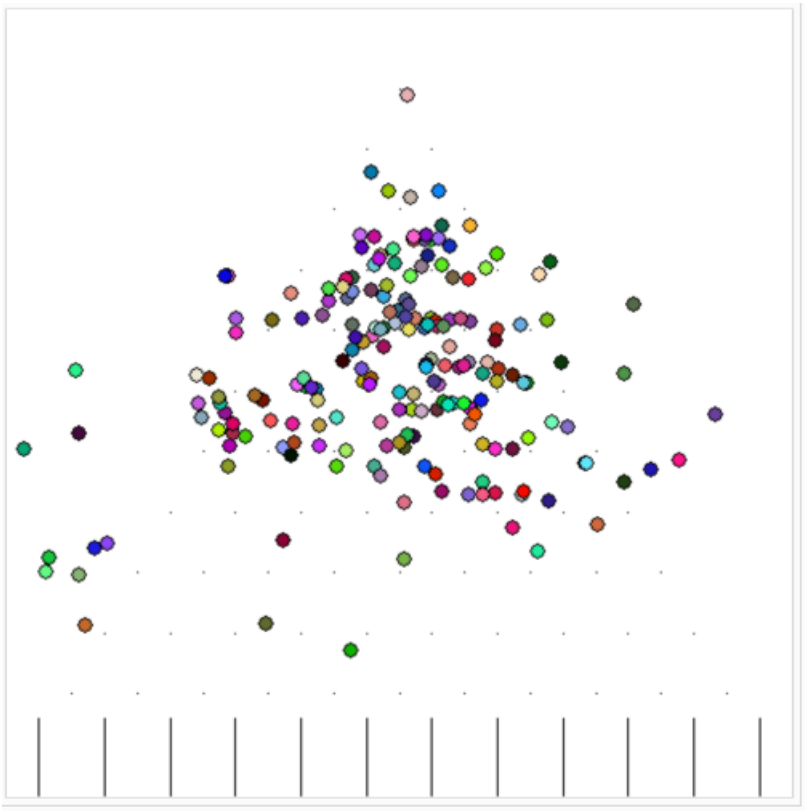
Berlin, den 25.03.2018
Moritz Hansemann

Mathesis Wintersemester 17/18

Galton-Fallbrett

Simulation in Python



Abstract:
Es gilt die Funktionsweise des Galton-Fallbretts vollständig in Python zu simulieren, sodass Statistikexperimente durchgeführt werden können. Dabei soll nicht nur das mathematische Modell simuliert werden, sondern auch verschiedene physikalische Aspekte.

Inhaltsverzeichnis

1 Einleitung **1**
1.1 Das Galton-Fallbrett 1
1.2 Zielsetzung 1

2 Protokoll **2**

3 Dokumentation **4**
3.1 Umgebung 4
3.2 Input 4
3.3 Output 4

4 Analyse **5**

5 Ausblick **6**

6 Zusammenfassung **7**

5 Literatur **7**

Abbildungsverzeichnis

1 Schematische Darstellung des Galton-Fallbretts[1] 1
2 Abweichung der Verteilung bei Veränderung des Kugelradius (2000 Kugeln) 5
3 Abweichung der Verteilung bei Veränderung der Kugelanzahl (Radius 5) 6

1 Einleitung

1.1 Das Galton-Fallbrett

Das Galton-Fallbrett ist ein vom britischen Naturforscher und Schriftsteller Francis Galton[2] entwickeltes Demonstrationsexperiment der Statistik. Mit dem Fallbrett lassen sich sehr einfach verschiedene Aspekte der Grundlegenden Statistik anschaulich zeigen. In der Regel ist das Fall-Brett wie folgt aufgebaut:

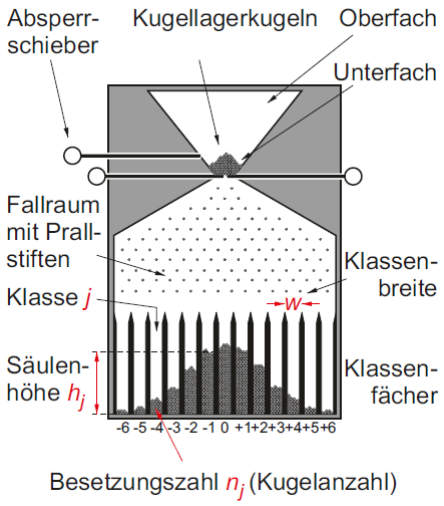


Abbildung 1: Schematische Darstellung des Galton-Fallbretts[1]

1.2 Zielsetzung

Ziel des Projekts ist die vollständige Simulation des Galton-Fallbretts in Python. Dazu soll zunächst das mathematische Modell simuliert werden als ein 'Prove of concept'. Anschließend gilt es die physikalische Realität möglichst nah abzubilden. Gesammt soll der Ablauf der Kugeln durch das Fallbrett visualisiert werden. außerdem soll der User die Möglichkeit haben die Parameter des Bretts frei zu adaptieren und so die Verschiedenen Eigenschaften des Bretts zu zeigen. Die adaptierbaren Paramet sollen Kugelanzahl so wie Größe, Anzahl der Pins und Fächer, Größer des Bretts, Fallbeschleunigung, Fallwinkel und Stoßparameter sein. Das Ergebnis eines Ablaufs soll gezählt werden und in einer Datei festgehalten werden. Außerdem soll ein Graphical User Interface (GUI) die Veränderung der Parameter ermöglichen und so dem User die Möglichkeit geben Code frei Experimente durchzuführen. Zusätzlich sollen einzelne Pins einen Imperfektion, eine Neigung aufweise die bestimmte Fallrichtung bevorzugt. Dies soll die Eigenheit eines bestimmten Bretts darstellen.

2 Protokoll

Donnerstag, 30.11.2017

Zunächst wurde begonnen das mathematische Modell ohne Visualisierung zu realisieren, um zu zeigen das das Gesamtkonzept funktioniert ('Prove of concept'). Dies war Erfolgreich.
Weiter wurde auch die Ausgabe als Textdatei realisiert. Diese weißt aber noch Bugs in der Fächerverteilung auf. Die Kugeln scheinen in der Regel ein Fach zu überspringen, aber nicht immer.

Donnerstag, 07.12.2017

Das mathematische Modell konnte vollständig realisiert werden.
Es wird mit der Recherche verschiedener Visualisierungen begonnen. Als engere Kandidaten kommen 'graphics' und 'pygame' in Frage. Die Entscheidung fällt auf 'graphics' aufgrund beweglicher Objekte. Außerdem wird überlegt zunächst die Trajektorien aller Objekte auszurechnen und diese anschließend mit einem beliebigen Programm zu realisieren. Diese wird später aber verworfen.
Als erstes Problem stellt sich heraus mehrere Objekte über eine Schleife zu erzeugen.

Donnerstag, 21.12.2017

Die dynamische Berechnung der Pin-Positionen, so wie die Fächer wurden visualisiert. Nun kann mit Hilfe des Strukturparameters *struc* dynamisch die Anzahl der Fächer und so die Anzahl der Pins verstellt werden. Die Darstellung des Bretts ist damit abgeschlossen.
Es wird mit der Animation der Kugeln begonnen.

Donnerstag, 11.01.2018

Die anfangs gerade Bewegung der Kugel wird in eine Springende Bewegung umgewandelt, dazu wird der Kugel ein Impuls zugeordnet. Außerdem wird eine Kollision-Abfrage eingebaut, die zunächst den Abstand zu allen Pins berechnet und für einen Pin unterhalb einer Abstandsgrenze den Impuls der Kugel umkehrt. Dies leitet den beginn des physikalischen Modells ein.

Donnerstag, 18.01.2018

Die Erzeugung der Pin geschah bisher frei und führte jedoch zu dem Problem das nur noch der zuletzt erzeugte Pin ansprechbar war. Dies wird behoben indem die Pins so wie alle Kugeln nun in Listen gespeichert werden.

Donnerstag, 01.02.2018

Die Kugeln werden nun in Tupellisten aus Ort und Impuls gespeichert. Es treten Probleme mit der Visualisierung auf, 'graphics' scheint mit größerer Kugelanzahl nicht zurecht zu kommen. Um die Perfomance zu verbessern wird auf die 'move'-Operation von 'graphics' verzichtet und nur noch Kugeln vor und nach ihrer Bewegung gezeichnet. Dies führt aber nicht zu einer deutlichen Verbesserung. Daraufhin wird entschieden auf 'graphics' zu verzichten und auf 'pygame' zu wechseln.

Donnerstag, 08.02.2018

Das Fallbrett wird vollständig in 'pygame' implementiert. Auch dies liefert aber keine deutliche Verbesserung der Performance. Das problem ist die Kollisionsabfrage, die bisher noch nur die Pins berücksichtigt, aber in jeder Bewegung alle Pins für jede Kugel zu kontrollieren ist einfach zu viel.
Als alternative wird ein kd-Tree aus 'scipy' implementiert. Auch dies führt zu keiner akzeptablen Verbesserung. Möglicherweise ist die Simulation von 2000 Kugeln und mehr zu viel um sie live durchzuführen.

Donnerstag, 15.02.2018

Die Performance kann deutlich verbessert werden nach dem die Leafsize des kd-Tree's reduziert wird. Die Kugel-Startposition wird über Zufallszahlen(RNG) leicht variiert. Dies erzeugt allerdings sehr symmetrische Muster beim Fallen der Kugeln.
Des weiteren wird die Auswertungsmethode und Dateiausgabe die bereits im mathematischen Modell benutzt wurde, auch in das physikalische Model implementiert.

Dienstag, 06.03.2018

Die Kugel-Kugel-Kollision wird mit einem weiteren kd-Tree hinzugefügt. Es wird eine Testbench gebaut um die Kugel-Kugel-Kollision genauer zu betrachten, da beim Fallen der Kugeln Probleme auffallen. Bisher sind alle Stöße Zentralstöße, die in manchen Situation zu seltsamen Verhalten der Kugeln führt.

Mittwoch, 07.03.2018

Um die Stöße realistische zu simulieren, werden nun auch dezentrale Stöße berücksichtigt. Die Kugeln werden bisher als Tupel in Listen geführt, dies wird nun durch eine Objektversion ersetzt. So wird auch ermöglicht Vektoroperationen aus 'scipy' zu benutzten, da nun Ort und Impuls als Vektorobjekte vorliegen. Allerdings gibt es hier Fehler bei der Berechnung die zu riesigen Impulsen der Kugeln führen.

3 Dokumentation

3.1 Umgebung

Das Programm wurde in Python 2.7.12 geschrieben. die Verwendeten Lybaries haben folgende Versionsnummern:

1. numpy 1.11.0
2. scipy 0.17.0
3. pygame 1.9.1release

Für die Visualisierung mit 'graphics' wurde Version graphics 5.0.1.post1 verwendet.

3.2 Input

Als Input erwartet das Programm eine Höhe und Breite, die in Pixel angegeben wird. Diese Bestimmen die Größe des Fensters. Diese können frei gewählt werden solange sie größer Null sind, aber auf Grund der Anschaulichkeit sollten hier Werte im vielfachen Hunderter Bereich gewählt werden. Des weiteren wird die Anzahl der Kugeln und der Strukturwert erwartet. Der Strukturwert legt die Anzahl der Fächer und somit die Anzahl der Pins fest, dieser Wert sollte mindestens Eins sein. Außerdem kann der Radius der Kugeln angegeben werden. Dieser ist für die Kollisionsabfragen und die Darstellung der Kugeln wichtig. Zuletzt kann der Modus: 0 = Fallbrett oder 1 = Testbench, sowie der Dateiname für die Output-Datei angegeben werden.

3.3 Output

Das Programm öffnet das 'pygame'-Fenster in der angegeben Größe und beginnt mit der Simulation. Zusätzlich öffnet sich ein Consolen-Fenster das zu Ende des Programms 'FIN' zeigt. Das Programm schreibt eine $\approx$ 130 byte große Text-Datei mit den Daten in das Verzeichnis des Programms. Diese enthält die zusammengezählten Daten wie folgt:

-6.0	12
-5.0	10
-4.0	5
-3.0	16
-2.0	15
-1.0	18
0.0	25
1.0	21
2.0	20
...	

4 Analyse

Schaut man sich die Ergebnisse der Simulation an so gibt es zwei entscheidende Parameter die die Verteilung in den Fächern beeinflussen. Die Kugelanzahl und der Kugelradius.

Kugelradius

Bei kleinen Radien ist die Verteilung deutlich um das mittlere Fach (Fach 0) zentriert und hat hier ein Maximum. Dies begründet sich darin, dass die Kugel weniger mit den Pins interagieren und so schneller durch das Brett fallen. Wird der Radius größer Verbreitert sich das Maximum, da sich nun die Kugeln stärker behindern und so die Verteilung von innen nach außen gedrückt wird. So fallen mit zunehmendem Radius auch deutlich mehr Kugeln in die äußersten Fächer (6 und -6).

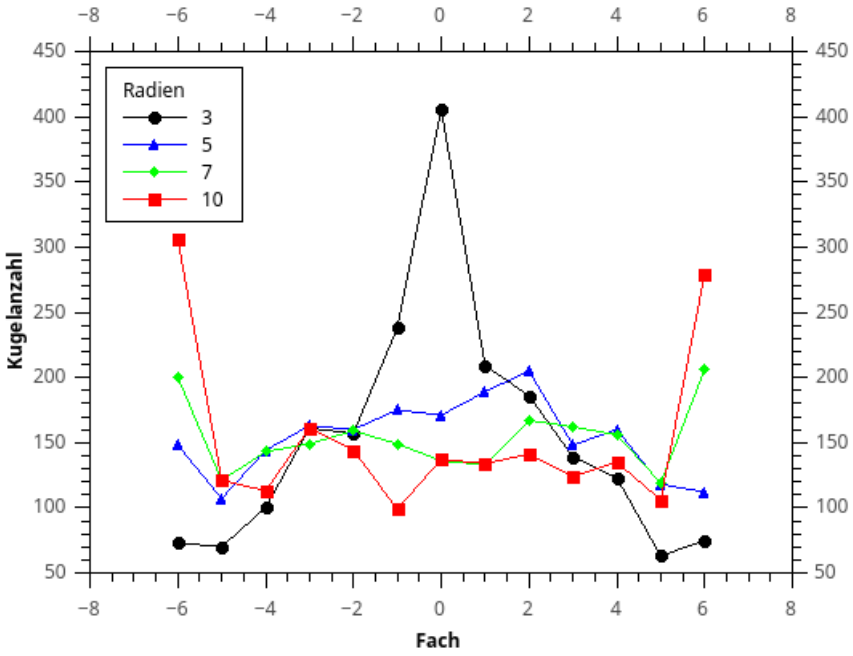


Abbildung 2: Abweichung der Verteilung bei Veränderung des Kugelradius (2000 Kugeln)

Kugelanzahl

Steigert man die Anzahl der Kugeln so zeigt sich deutlich die Verringerung der Varianz → die Kurve nähert sich der Gaußverteilung an. Dies ist ein Vorgang der mit der Natur übereinstimmt, was diesen Teil des Programms sehr schön die Realität nachempfinden lässt.

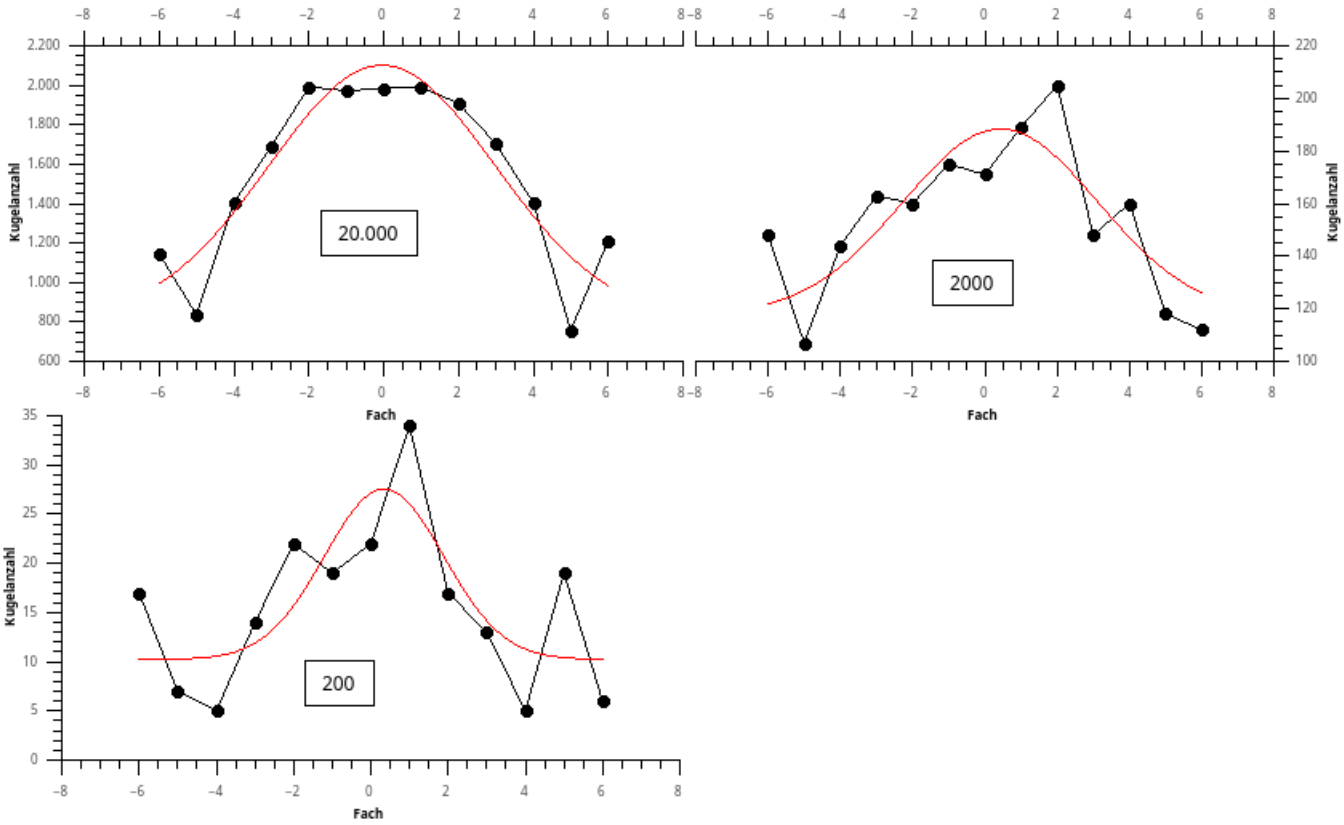


Abbildung 3: Abweichung der Verteilung bei Veränderung der Kugelanzahl (Radius 5)

5 Ausblick

In seinem jetzigen Stand hat das Programm schon viele seiner erwarteten Funktion, einige bleibe aber noch aus oder sind unvollständig. Es wurde bereits damit begonnen die die dezentrale Kollision auch für die Pins zu realisieren, dies ist aber bisher noch nicht erfolgreich. Des weiteren wurde mit der begonnen das Programm auf einen Objektorientierte Struktur umzustellen und Vektoren und Vektoroperation aus 'numpy' zu verwenden.
Zusätzlich wäre eine Integration des mathematischen Modells in das physikalische und eine Möglichkeit zwischen beide zu wechseln erstrebenswert. Auch das implementieren einer GUI ist bisher ausgeblieben und würde die User-experience deutlich verbessern. Außerdem wäre eine automatische Analyse der Daten eine sinnvolle Erweiterung.
Als Nebenprodukt hat dieses Programm, das eine stets bessere Physiksimulation darstellt, die Testbench hervorgebracht. Mit ihr lassen sich zum Beispiel ideale Gase simulieren. Dies könnte eine weitere zukünftige Verbesserung des Programms darstellen.

6 Zusammenfassung

Insgesamt kann das Projekt als Erfolg angesehen werden. Es konnte eine realitätsnahe Simulation des Galton-Fallbretts als mathematische, sowie physikalische Simulation erzeugt werden. Das Programm hat viele Aspekte der Realität hervorgebracht und ein tieferes Verständnis der physikalischen Abläufe hervorgebracht. Auch wenn noch einige Aspekte der Ursprünglichen Zielsetzung fehlen, so kann das Programm schon die meisten Ziele erfüllen.

5 Literatur

[1] H. J. Eichler, H.-D. Kronfeld, J. Sahm: *Das Neue physikalische Grundpraktikum.* 2. Aufl., Springer Verlag, 2006

[2] `https://de.wikipedia.org/w/index.php?title=Francis_Galton&oldid=168344956`