Mathematische Anwendungen der Monte-Carlo-Methode

Approximative Bestimmung mathematischer Größen und wiederholte Stichproben realer Zufallsexperimente mithilfe eigener Berechnungsprogramme

Vorwissenschaftliche Arbeit im Fach

Mathematik, Informatik

Verfasser:

Tobias Buchli

Maturajahrgang: 2023

Klasse: 8itm

Eingereicht am: 23.02.2023

am BRG/BORG Dornbirn-Schoren

Betreuende Lehrperson

Mag. Raimund Hermann

Abstract

Diese Arbeit beschäftigt sich mit der Monte-Carlo-Methode, welche in den verschiedensten Bereichen angewendet werden kann. Hauptaugenmerk liegt auf den einfachen, aber sehr anschaulichen Anwendungsbeispielen, die auf dem Prinzip der Monte-Carlo-Methode durchgeführt werden. Grundlage hierfür ist die Simulation diverser Zufallsexperimente. Im Zuge dieser Arbeit handelt es sich um die Simulation von Flächen, welche wir nicht exakt berechnen können. Dabei sprechen wir von Kreis- und Integralflächen. Auch ist es möglich, reale mathematische Experimente in einem wesentlich größeren Ausmaß durchzuführen. In Excel lassen sich mit der darin integrierten Skriptsprache VBA Versuche dieser Art mit eigenen Berechnungsprogrammen durchführen und grafisch darstellen. Während des Prozesses wird erkennbar, in welchem Umfang die Methode behilflich sein kann und welche Vorteile sie gegenüber konventionellen Berechnungen haben kann. Es wird ersichtlich, dass die Methode weniger für exakte Berechnungen, aber durchaus für das Verständnis verschiedener Zusammenhänge dient.

Vorwort

Schon früh wurde meine Faszination für die Mathematik in meiner Schullaufbahn erkennbar. Von meinem Umfeld in dieser Neugier unterstützt zu werden, ist dann natürlich sehr von Vorteil. Dies ermöglichte mir, meinen mathematischen Horizont über das von der Schule geforderte Maß hinaus erweitern zu können.

Beginnend möchte ich meinen Eltern danken, welche beide in ihren Berufen tagtäglich von der Mathematik Gebrauch machen und mir somit nie das Gefühl gegeben haben, die Mathematik wäre ein sinnloses Fach, ja möglicherweise würde man sie im späteren Leben nicht mehr benötigen. Dies ist ein guter Ansporn, meine vorwissenschaftliche Arbeit in genau diesem Gebiet zu schreiben.

Ein großes Danke gebührt ebenfalls meinem Betreuungslehrer und Mathematik-Professor Raimund Hermann, der mir anfangs dieses spannende Thema vorgeschlagen und mich während des Schreibprozesses fortlaufend unterstützt und motiviert hat. Ohne sein mathematisches Wissen und seine Hilfsbereitschaft wäre die Arbeit in dieser Form nicht möglich gewesen.

Im Laufe der Arbeit werden verschiedene Anwendungsbeispiele in Excel mitsamt dem dazugehörigen Programm in VBA vorgestellt und erläutert. Die dazugehörige Datei finden Sie unter diesem QR-Code oder dem Link darunter zum Download. Dort lassen sich die Zufallsexperimente noch einmal durchführen. Beachten Sie, dass die Makros nach dem Download zuerst noch in den Dokumenteigenschaften zugelassen werden müssen.

Dornbirn, am 23.02.2023 Tobias Buchli

tinyurl.com/vwa-buchli

Inhaltsverzeichnis

1. Einleitung 6

2. Definition der Monte-Carlo-Methode 8

2.1 Geschichte und Entstehung 8

2.2 Zufallszahlen 9

2.2.1 Die Rnd-Funktion und Randomize-Anweisung in VBA 10

2.3 Überblick über Anwendungen und ihre Beispiele 11

2.3.1 Anwendungen 11

2.3.2 Beispiele 11

3. Mathematische Anwendungen anhand von Beispielen mit Excel VBA 13

3.1 Probabilistische Bestimmung nicht berechenbarer Flächen 13

3.1.1 Berechnung der Kreisfläche und Kreiszahl π 13

3.1.2 Numerische Integration im Intervall [0; 1] 15

3.2 Das Nadelexperiment des Comte de Buffon 17

3.2.1 Entstehung und Definition 17

3.2.2 Das Experiment und die Kreiszahl π 18

3.3 Der Weg eines Betrunkenen 20

3.3.1 Der lineare Weg eines Betrunkenen 21

3.3.2 Der Weg eines Betrunkenen in der Ebene 23

3.4 Das Galtonbrett 25

3.4.1 Sir Francis Galton 26

3.4.2 Der Versuch 27

4. Zusammenfassung und Ausblick 31

5. Literatur- und Quellenverzeichnis 33

6. Abbildungsverzeichnis 35

7. Glossar 37

1. Einleitung

Früh lernt man, wie man den Flächeninhalt eines Kreises ausrechnet. Mithilfe der Kreiszahl π. Doch stellte sich mir die Frage, wie man diese, zumindest näherungsweise, berechnen und erfassen kann. Auf der Suche nach einer Antwort bin ich auf die sogenannte Monte-Carlo-Methode gestoßen. Sie ermöglicht dieses Vorhaben durch eine näherungsweise Berechnung, die auf dem Prinzip des Zufalls beruht und die Simulation realer Zufallsexperimente möglich macht. Mit ihr kann aber noch vieles mehr angestellt werden, wie beispielsweise in weiterer Folge die annäherungsweise Berechnung mathematischer Größen und unbekannter Flächen. Ein ähnliches Beispiel wäre dabei die Berechnung des Integrals einer Funktion, welches ebenfalls lediglich eine zu berechnende Fläche ist. Die Methode verwendet Pseudozufallszahlen, um Simulationen am Computer zu ermöglichen. Sie dient weniger als wissenschaftlicher Beweis, sehr wohl aber zum Verständnis diverser Zusammenhänge. Beinahe unlösbare mathematische Probleme lassen sich dabei annäherungsweise und einfach numerisch lösen.

Als Gegenstand dieser vorwissenschaftlichen Arbeit dient die Monte-Carlo-Methode als ein geeigneter Einstieg in die weite Welt der Mathematik, da sie weniger mit Theorie und Formeln, aber umso mehr mit Berechnungen und Experimenten zu tun hat. Es gilt, herauszufinden, wie die Methode funktioniert und in welchem Umfang es möglich ist, sie eigens anzuwenden. Dabei lassen sich folgende Leitfragen beantworten: Welche Vorteile bietet sie gegenüber konventionellen Berechnungen und welche weiteren praktischen Anwendungen offenbaren sich im Laufe des Bearbeitungsprozesses?

Bevor diese Fragen geklärt werden können, wage ich mich daran, eigene Programme auf dem Prinzip der Monte-Carlo-Methode zu schreiben, um die bereits beschriebenen Beispiele umsetzen zu können. Das Ziel soll es sein, diese Stichproben anhand mehrerer eigens entwickelter Computerprogramme durchzuführen und dabei die Vorteile der Verwendung einer solchen Methode sichtbar zu machen. Im ersten Abschnitt wird die Geschichte der Methode beschrieben und einige wichtige Definitionsbegriffe geklärt. Mit dem darin erlangten Vorwissen lassen sich dann im zweiten Abschnitt die ausgesuchten Anwendungsbeispiele verstehen.

Viele Dinge, die noch vor wenigen Jahren mit großem Aufwand verbunden waren, lassen sich mittlerweile relativ einfach realisieren. Das Microsoft-Office Paket bietet für unterschiedlichste Bereiche unkomplizierte Lösungen. Insbesondere das für diese Arbeit relevante Excel-Programm entwickelt sich immer mehr zu einem unvermeidbaren Arbeitsmittel. Was die meisten, die damit arbeiten, aber gar nicht wissen, sind die umfassenden Programmiermöglichkeiten in allen Office-Programmen, die für alle verfügbar sind und verwendet werden können. Mithilfe der Skriptsprache *Visual Basic for Applications* (VBA), die auch in Excel verfügbar ist, lassen sich diese Beispiele relativ einfach dokumentieren, analysieren und letztlich visualisieren. Durchaus bekannt sind Makros, die gerne genutzt werden, um Viren in den Dokumenten einfach zu verbreiten. Mittlerweile bietet aber Microsoft einen brauchbaren Schutz gegen Malware dieser Art. Denn eigentlich sind Makros eine Erweiterung zu den damit verknüpften Basis-Anwendungen und bieten damit einen hervorragenden Funktionsumfang. (vgl. Nahrstedt, 2015, S. VIIf)

Wenn wir in späteren Abschnitten mit den Berechnungsprogrammen beginnen, sollten zuvor einige Begriffe der Syntax von VBA geklärt werden. Mit der Anweisung *Dim* werden in VBA Variablen deklariert. Auch wird ihnen ein Datentyp zugewiesen. Wir arbeiten hauptsächlich mit *Long* und *Double*. Erstere sind lange Ganzzahlen, während letztere gebraucht werden, um Gleitkommazahlen mit doppelter Genauigkeit zu verwenden. Mit der Anweisung *Cells* kann auf Inhalte in Excel zugegriffen werden, um mit ihnen zu rechnen. Mit demselben Ausdruck lassen sich den Zellen auch Werte übergeben. (vgl. Microsoft)

Im letzten Abschnitt werden schlussendlich die Vorteile gegenüber herkömmlichen Berechnungen diskutiert und mit einem Fazit über die Methode abgeschlossen.

2. Definition der Monte-Carlo-Methode

Die Monte-Carlo-Methode ist ein Verfahren aus der Stochastik, bei der anhand von wiederholten Zufallsexperimenten analytisch nur recht aufwendig lösbare Probleme in der Mathematik numerisch durchgeführt werden können. Die Zufallsstichproben lassen sich real, aber auch mit Algorithmen am Computer durchführen. Diese verwenden zur Simulation des Zufalls sogenannte Pseudo-Zufallszahlen. (vgl. Mathepedia, kein Datum)

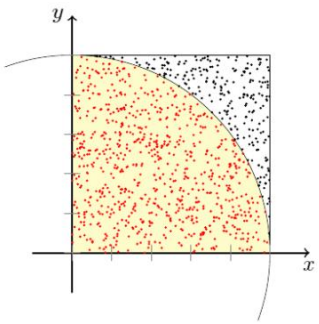

Abbildung 1: Berechnung der Kreiszahl π (Springob, 2004)

2.1 Geschichte und Entstehung

Die Anfänge der Monte-Carlo-Methode reichen bis ins Jahr 1733 zurück, als der französische Adlige und Naturforscher Georges Louis Leclerc, Comte de Buffon das berühmte Nadelproblem vor der Pariser Akademie der Wissenschaften vorstellte. Es behilft sich des Zufalls und ermöglicht eine experimentelle Annäherung an die Zahl π. Somit gilt sie als eine der ersten Anwendungen dieser Methode, da sich ein solches Experiment mithilfe der Monte-Carlo-Methode einfach simulieren lässt. (siehe 3.2) (vgl. Nahrstedt, 2015, S. 34)

Beteiligt an der Entwicklung waren schließlich im 20. Jahrhundert unter anderem der Physiker Enrico Fermi und später die Mathematiker Stanislaw Ulam sowie John von Neumann. 1930 hatte sich Fermi bereits mit der Simulation von Neutronenbewegungen beschäftigt. Während des Zweiten Weltkriegs sollen Ulam und von Neumann im Zuge des damals geheimen Manhattan-Projekts am *Los Alamos Scientific Laboratory* die Monte-Carlo-Methode angewendet haben, um damals hochkomplexe physikalische Probleme numerisch und simulativ zu lösen. Das Projekt führte zur Entwicklung der ersten Atombombe. Die Bezeichnung Monte Carlo wurde möglicherweise aufgrund der Beziehungen von Ulams Onkel zur Spielbank Monte Carlo verwendet und diente als Codename in diesem Projekt. (vgl. Nahrstedt, 2015, S. 1; RiskNET, kein Datum; Frey & Nießen, 2001, S. 15f)

Erst durch die Entwicklung der modernen Computertechnik konnten die Anwendungsmöglichkeiten durch Berechnungen mittels Monte-Carlo-Algorithmen deutlich erweitert werden. Experimente, die bis dahin nur real durchführbar waren, konnten nun mithilfe von Rechnern simuliert werden. Zusätzlich wurde es möglich, in kürzester Zeit eine beliebig hohe Anzahl an Zufallszahlen zu generieren. (vgl. Spektrum.de, 2017; Kohlas, 1971)

2.2 Zufallszahlen

Bevor man die Monte-Carlo-Methode ausreichend definieren kann, muss zuvor der Begriff der Zufallszahl geklärt werden. Wenn mit modernen Digitalrechnern gearbeitet wird, ist nichts dem Zufall überlassen. Die Simulation des Zufalls steht im Widerspruch zur Verwendung von modernen Rechnern. Somit ist es unmöglich, dass ein solcher Algorithmus echte Zufallszahlen generieren wird. Zufallsgeneratoren, wie sie in nahezu allen Programmiersprachen zu finden sind, erzeugen in Wahrheit also keine zufälligen Zahlen, sondern „*meist sogenannte Pseudo-Zufallszahlen, die von deterministischen Algorithmen erzeugt werden*." (Müller-Gronbach, Novak, & Ritter, 2012, S. 2) Bei diesen ist es möglich, dass sich nach einer bestimmten Länge Zahlenfolgen wiederholen werden. Dadurch wird im Zusammenhang mit der Monte-Carlo-Methode meist der Begriff Pseudo-Zufallszahl verwendet, da sie den Eigenschaften echter Zufallszahlen schon sehr nahekommen. Echte Zufälle lassen sich unter anderem durch das Digitalisieren von Rauschen oder durch das Ausnutzen von Quanteneffekten erzeugen, welche zeitlich oder technisch natürlich unnötig aufwendig für den Umfang dieser Arbeit sind. (vgl. Nahrstedt, 2015, S. 2; Lemieux, 2009, S. 57f; Dr. Lehner, 2017, S. 20f)

Die Begründung des Prinzips der Methode und somit eine wichtige Eigenschaft von Zufallszahlen ist das Gesetz der Großen Zahlen. Es handelt sich dabei um Aussagen über die Konvergenz der Mittelwerte von Zufallszahlen. Das Gesetz besagt, dass durch eine steigende Anzahl der Durchgänge eines Experiments sich auch die Wahrscheinlichkeit einer Gleichverteilung erhöht und die Genauigkeit des Ergebnisses dadurch zunehmend steigt. (vgl. Nahrstedt, 2015, S. 2; Frey & Nießen, 2001, S. 100; Meintrup & Schäffler, 2005, S. 150f; Dr. Lehner, 2017, S. 19ff)

Erklären lässt sich dies am einfachsten durch ein bekanntes Beispiel, dem Münzwurf mit Kopf und Zahl. Die theoretische Wahrscheinlichkeit, dass eines der beiden Ereignisse eintritt, ist ½. Wirft man die Münze ein paarmal, muss es nicht sein, dass beide Ereignisse gleich oft auftreten. Wirft man die Münze häufiger, so wird dies aber immer wahrscheinlicher, es nähert sich also der Quotient eines Ereignisses durch die Gesamtanzahl der Würfe dem Wert ½. Eine Garantie, dass sich beide Ereignisse ausgleichen werden, gibt es dabei nicht, ganz egal, wie oft man die Münze werfen wird. (vgl. Nahrstedt, 2015, S. 2; Meintrup & Schäffler, 2005, S. 150)

Erwähnt wurde diese Gesetzmäßigkeit das erste Mal 1733 durch den Schweizer Mathematiker Jakob Bernoulli. Er beschreibt in seiner *Ars Conjectandi* den einfachsten Fall des Gesetzes, ein Zufallsexperiment mit nur zwei Ausgängen, Erfolg und Misserfolg. Heute werden Experimente dieser Art als *Bernoulli-Experiment* bezeichnet. (vgl. Nahrstedt, 2015, S. 2)

2.2.1 Die Rnd-Funktion und Randomize-Anweisung in VBA

Auch VBA arbeitet mit Pseudozufallszahlengeneratoren. Um diese aufzurufen, wird die Funktion *Rnd()* verwendet, welche einen Wert zwischen einschließlich 0 und ausschließlich 1 liefert. Im Hintergrund erzeugt diese dabei eine Folge von Zufallszahlen und ruft die erste Zahl in der Liste auf. Welches die erste Zahl ist, wird durch den Anfangswert definiert. Dieser wird bei erstmaligem Aufrufen durch den Systemzeitgeber von Windows generiert. Dies bedeutet, dass für die Simulation des Zufalls aktuelles Datum und Uhrzeit verwendet werden. Die Genauigkeit reicht dabei sogar bis in den Bereich der Nanosekunden. Sollte man dieselbe Funktion mit identischem Anfangswert aufrufen, wäre auch dessen pseudozufällige Zahlenfolge ident. Ein erneuter Aufruf der Funktion gibt dann lediglich den nächsten Wert der Folge aus. Um bei jedem weiteren Aufruf auch die Folge neu zu generieren, setzt man im Code vor die Funktion eine *Randomize*-Anweisung, welche die Funktion zwingt, erneut den Systemzeitgeber und somit die aktuelle Uhrzeit abzufragen und einen neuen Startwert zu erzeugen. Dadurch werden nicht nur die Zahlen zufällig generiert, sondern bei jedem Aufruf auch der Anfangswert und damit die ganze Liste. (vgl. Microsoft, 2022; Minhorst, 2015)

2.3 Überblick über Anwendungen und ihre Beispiele

Mit der Monte-Carlo-Methode lassen sich unzählige Anwendungen und Probleme simulieren. Besonders in der Physik spielt sie eine signifikante Rolle. Da in weiterer Folge dieser Arbeit Anwendungen der Methode eine noch größere Rolle spielen, werden sie in diesem Kapitel nur zum Zwecke der Anschaulichkeit des weitreichenden Einsatzbereiches angeführt.

2.3.1 Anwendungen

Eine bereits beschriebene Anwendung der Monte-Carlo-Methode sind die Stichprobenwiederholungen (engl. Resampling), die mit ihrer Hilfe berechnet werden können. Da reale statistische Tests wegen ihres großen Aufwands nicht immer sinnvoll sind, können rechnergestützte Methoden viele Datensätze in kurzer Zeit erzeugen. Eng damit verbunden ist das Nachbilden von komplexen realen Prozessen, welche in vielen Bereichen der Wissenschaft Verwendung finden. (vgl. Mathepedia, kein Datum)

2.3.2 Beispiele

Ein bekanntes Beispiel ist das bereits bekannte Nadelexperiment. Weiters kann mit der Monte-Carlo-Methode das anschauliche Experiment des Galtonbretts, welches in 3.4 genauer beschrieben wird, simuliert werden. Bedeutende Anwendungsbeispiele sind außerdem die probabilistische Bestimmung der Kreiskonstante π anhand einem auf dem Einheitskreis liegenden Quadrat, sowie die auf dem gleichen Prinzip ruhende numerische Integration, welche beide in 3.1 anhand von einzelnen Beispielen gerechnet werden. Es lassen sich also Integrale approximieren, die Kreiszahl π lässt sich bestimmen und es ist möglich, Zufallsexperimente sowie die Normalverteilung zu simulieren. (vgl. Nahrstedt, 2015)

Ein wichtiges Feld, in dem die Monte-Carlo-Methode noch heute ihre Anwendung findet, sind die Naturwissenschaften mit ihren Bereichen der Physik und der Chemie. Die anfangs erwähnten Wissenschaftler Fermi, Ulam und von Neumann haben sich unter anderem mit der Untersuchung von Kernspaltungsprozessen beschäftigt. Da solche phy-

sikalischen Experimente aber aufgrund ihrer Gefahr, Kettenreaktionen und damit Explosionen auszulösen, zu riskant waren, wurde eine Alternative notwendig. Kernspaltungsprozesse sind von stochastischer Natur, eine analytische Berechnung ist nicht möglich, da sie von mehreren nichtlinearen Zusammenhängen abhängen. (vgl. Nahrstedt, 2015, S. 3)

Die Monte-Carlo-Methode kann auch in der Wirtschaft angewendet werden. Ein zentrales Stichwort ist der Aktienmarkt, der damit eine Voraussage treffen kann, welchen Kurs eine Aktie einschlagen wird. Vereinfacht kann mithilfe einer Reihe von Simulationen ein Trend abgeschätzt werden. Weiters finden sich Anwendungsbeispiele in Bereichen der Nuklearmedizin, der Risikoanalyse, sowie in der Versicherungsindustrie. (vgl. Kernbichler & Theis, 2002, S. 20ff; Frey & Nießen, 2001)

3. Mathematische Anwendungen anhand von Beispielen mit Excel VBA

Wie bereits im letzten Kapitel erwähnt, existieren in der Praxis zahlreiche Anwendungsmöglichkeiten der Monte-Carlo-Methode. Nachfolgend werden nun einige ausgewählte Beispiele genauer beschrieben. Durch die jeweiligen durchgeführten Berechnungen in der Programmierumgebung von Excel lässt sich die Thematik einfach und bildlich verstehen.

3.1 Probabilistische Bestimmung nicht berechenbarer Flächen

3.1.1 Berechnung der Kreisfläche und Kreiszahl π

Unser Ziel ist es, die Kreiszahl π mit eigenen Berechnungen möglichst genau zu bestimmen. Dabei ist uns der Flächeninhalt eines Einheitskreises ebenfalls unbekannt. Die Monte-Carlo-Methode sieht nun vor, auf diesen Kreis eine bekannte Fläche, und zwar die eines Quadrates, zu legen, um darin zufallsbedingte Punkte zu generieren. Zur Vereinfachung wird dabei ein Viertelkreis wie in Abbildung 2 betrachtet. Die Punkte verwenden wir, um damit den Flächeninhalt des Kreises zu simulieren, da wir π, welches für diese Berechnung essentiell wäre, nicht kennen.

x	y
0,87779	0,23261
0,026947	0,483348
0,707619	0,205335
0,441618	0,651413
0,559734	0,606338
0,498161	0,005474
0,720799	0,684896
0,157973	0,061664
0,531459	0,594046
0,034181	0,300691
0,499138	0,942555
0,968992	0,911476
0,331899	0,433876
0,539636	0,324895
0,088268	0,396929
0,109872	0,229234
0,700839	0,092456

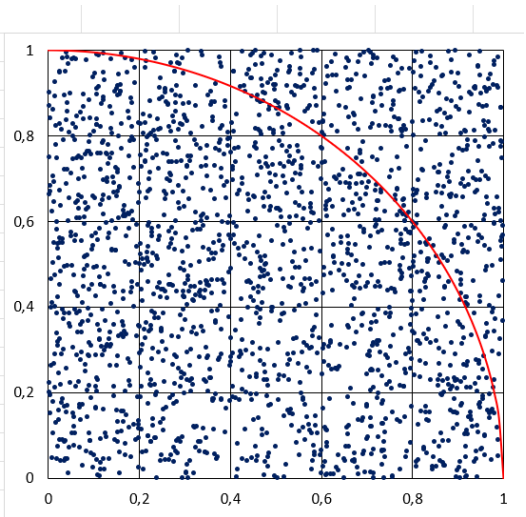

Abbildung 2: 2000 zufällig generierte Punkte im Viertelkreis (eigene Abbildung)

Für diese Koordinaten lässt sich nun folgende Abfrage über den Abstand aller Punkte zum Mittelpunkt, anhand des Satzes von Pythagoras, durchführen:

$$x_i^2 + y_i^2 \leq r^2$$

Da der namensgebende Radius des Einheitskreises 1 beträgt, können wir in diesem Fall nach 1 abfragen. Ist diese Ungleichung nun korrekt, befindet sich der Punkt innerhalb des Viertelkreises, da dafür der Abstand zum Mittelpunkt kleiner gleich 1 sein muss. Geht man davon aus, dass die erzeugten Pseudo-Zufallszahlen gleichverteilt sind, sollten sich nachfolgende Verhältnisse annähern.

$$\frac{\text{Punkte im Viertelkreis}}{\text{Anzahl aller Punkte}} \approx \frac{\text{Viertelkreisfläche}}{\text{Quadratfläche}}$$

Dabei gilt wieder, dass durch die zunehmende Menge der Punkte auch die Genauigkeit der Berechnung steigen wird. (vgl. Nahrstedt, 2015, S. 30)

Mit diesem Wissen lässt sich nun in VBA ein solches Zufallsexperiment in einem viel größeren Umfang durchführen. Das Ziel ist es, 1 Million Punkte zu generieren, und diesen Durchgang hundertmal zu wiederholen, um anschließend genügend Vergleichswerte zu erhalten.

Berechnung	Abweichung
3,1408	0,0008167
3,1450	0,0034433
3,1397	0,0018807
3,1416	0,0000207
3,1408	0,0008247
3,1400	0,0016207
3,1412	0,0003767
3,1438	0,0022513
3,1404	0,0011687
3,1428	0,0012473
3,1424	0,0007713
3,1417	0,0001233
3,1412	0,0003927
3,1390	0,0025487
3,1414	0,0002127
3,1417	0,0001073
3,1429	0,0012833
3,1424	0,0008553
3,1425	0,0008673

Berechne π

Abbildung 3: Zwanzig Durchläufe des Programms in Excel (eigene Abbildung)

```
Sub MCM_Pi()
    Dim x As Double
    Dim y As Double
    Dim i As Long
    Dim punkte As Long
    Dim treffer As Long
    Dim durchlaeufe As Long

    'Anzahl Durchläufe und Punkte
    durchlaeufe = 100
    punkte = 1000000

    'Zufallsgenerator
    Randomize
    For k = 1 To durchlaeufe
        treffer = 0
        For i = 1 To punkte
            'Zufallszahlen x und y
            x = Rnd()
            y = Rnd()
            'Abstand abfragen
            If x * x + y * y <= 1 Then
                'Punkte im Viertelkreis zählen
                treffer = treffer + 1
            End If
        Next i
    'In Excelzellen schreiben
    Cells(k + 1, 1).Value = 4 * (treffer / punkte)
    Next k
End Sub
```

Abbildung 4: Quellcode des Algorithmus in VBA (eigene Abbildung)

Zunächst werden zwei Zufallszahlen im Viertelkreis zwischen 0 und 1 generiert. Diese sind die Koordinaten der Punkte. Durch das Quadrieren der Koordinaten und der anschließenden Addition lässt sich nach dem Satz des Pythagoras das Quadrat des Abstands zum Mittelpunkt berechnen. Ist der Abstand kleiner oder gleich 1, befindet sich der Punkt im Kreis und die Variable *treffer* erhöht sich um eins. Dieser Vorgang wird wiederholt, bis alle Punkte generiert und überprüft wurden. Da wir bereits wissen, dass das Verhältnis zwischen den beiden Flächen und den darauf erzeugten Punkten annähernd gleich sein sollte, lässt sich dabei die Naturkonstante π mit

$$\frac{\pi}{4} \approx \frac{\text{Punkte im Viertelkreis}}{\text{Anzahl aller Punkte}}$$

approximativ berechnen. Zur mathematischen Herleitung der Verhältnisse dient nachfolgende Formel als eine verständliche Gedächtnishilfe. Es werden dabei der Flächeninhalt des gesamten Kreises, sowie der dessen Quadrats verwendet.

$$\frac{A_{Kreis}}{4 \cdot A_{Quadrat}} = \frac{\pi \cdot r^2}{4 \cdot r^2} = \frac{\pi \cdot 1}{4 \cdot 1} = \frac{\pi}{4}$$

3.1.2 Numerische Integration im Intervall [0; 1]

Auf dem nahezu selben Prinzip funktioniert die numerische Berechnung eines Integrals. Als Beispiel nehmen wir

$$F(x) = \int_0^x e^{-t^2}\, dt,$$

welches eine hohe Bedeutung in der Statistik und Wahrscheinlichkeitsrechnung hat. Es handelt sich um die Standardnormalverteilung. Da es nicht möglich ist, das Integral durch einen analytischen Funktionsausdruck zu beschreiben, müssen wir uns mit einem numerischen Näherungsverfahren behelfen. Bestens geeignet dafür ist die Monte-Carlo-Methode. Zur Vereinfachung betrachten wir die Funktion im Intervall [0; 1] und legen es in ein Quadrat mit Seitenlänge 1, um wiederum genügend Punkte darauf zu generieren. (vgl. Nahrstedt, 2015, S. 32)

Berechnung	Abweichung
0,7472	0,0003329
0,7465	0,0003371
0,7470	0,0001579
0,7465	0,0003141
0,7458	0,0010051
0,7468	0,0000129
0,7472	0,0003269
0,7476	0,0008239
0,7468	0,0000341
0,7468	0,0000741
0,7474	0,0005719
0,7461	0,0007131
0,7461	0,0006981
0,7468	0,0000501
0,7471	0,0002399
0,7473	0,0004929
0,7469	0,0000669
0,7474	0,0006089
0,7466	0,0002481

Integration

Abbildung 6: Zwanzig Berechnungen des Integrals in Excel (eigene Abbildung)

```
Sub MCM_Integral()
    Dim i As Long
    Dim x As Double
    Dim y As Double
    Dim f As Double
    Dim treffer As Long
    Dim punkte As Long
    Dim durchlaeufe As Long

    durchlaeufe = 100
    punkte = 1000000

    Randomize
    For k = 1 To durchlaeufe
        treffer = 0
        For i = 1 To punkte
            x = Rnd()
            y = Rnd()
            'Definition der Funktion für x-Wert
            f = Exp(-x * x)
            'Abfrage y-Wert <= Funktionswert
            If y <= f Then
                treffer = treffer + 1
            End If
        Next i
    Cells(k + 1, 1).Value = treffer / punkte
    Next k
End Sub
```

Abbildung 5: Quellcode des Programms in VBA (eigene Abbildung)

Wir integrieren dabei im Intervall [0; 1], wessen exakter Wert wie folgt beträgt:

$$\int_0^1 e^{-x^2} = 0{,}746824132812427$$

In der obigen Abbildung wird ersichtlich, dass auch dieser Wert nur mit einer geringen Abweichung zum exakten Wert bestimmt werden kann. Der Weg ist dabei ein leicht anderer. Statt den Abstand abzufragen, betrachten wir den jeweiligen Funktionswert. Dieser muss größer sein als der zufällig generierte y-Wert, um für das Programm einen *treffer* zu erzielen.

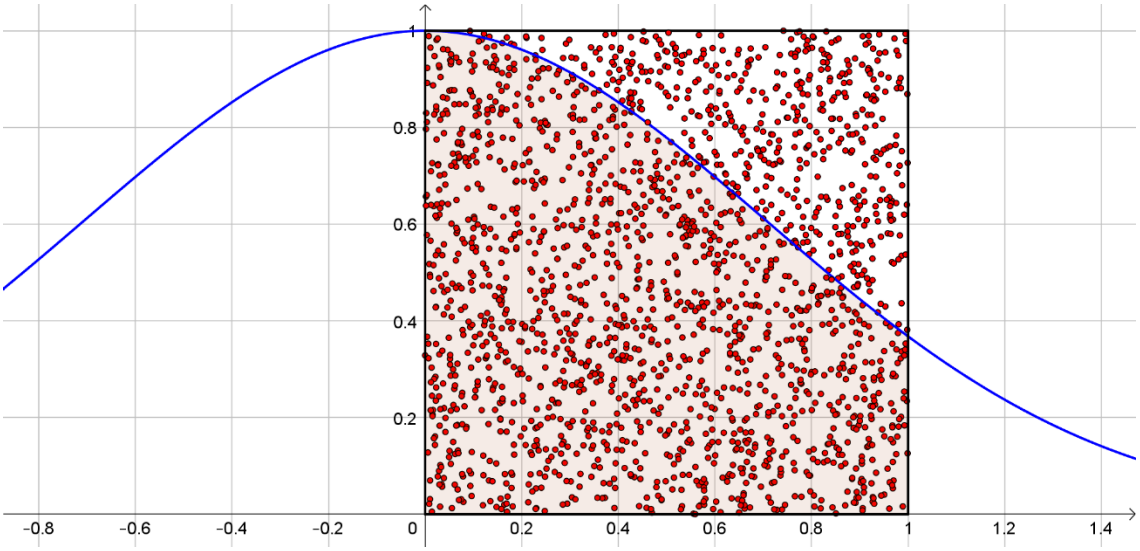

Abbildung 7: Zweitausend zufällig generierte Punkte auf dem Integral in GeoGebra (eigene Abbildung)

3.2 Das Nadelexperiment des Comte de Buffon

Das Buffonsche Nadelproblem behandelt eigentlich eine wahrscheinlichkeitstheoretische Frage, gewinnt aber durch das Auftreten von π große Relevanz.

3.2.1 Entstehung und Definition

Schon viele Völker vor Christi Geburt haben bereits versucht, die Zahl π zu bestimmen. Erst Archimedes erkannte, dass das Verhältnis zwischen Durchmesser und Umfang eines Kreises dem Verhältnis zwischen Fläche und Quadrat des Radius gleicht. Er konnte dieses Verhältnis aber damals nicht exakt bestimmen. Später bekam die Zahl ihren heutigen Namen und ein regelrechtes Rennen um die exakte Bestimmung der Kreiszahl begann. Zu jener Zeit war jedoch noch nicht gewiss, ob die Anzahl der Nachkommastellen endlich ist oder nicht. Dabei entstanden verschiedenste Methoden, π zu berechnen. Ebenso versucht hatte es Georges Louis Leclerc de Buffon mit seinem durchaus bekannten Nadelproblem. Einer Erzählung nach soll er zur Demonstration ein französisches Baguette über seine Schultern auf einen Dielenfußboden geworfen haben. Das Baguette sind dabei Nadeln der Länge l, die auf parallele Linien mit einem Abstand von $a \geq l$ geworfen werden. Dabei kommen sie unter dem Winkel φ zum Liegen. Gesucht ist nun die Wahrscheinlichkeit, dass die Nadeln auf einer der Linien liegen bleiben. Um dieses Experiment einfach durchführen zu können, kommt wiederum die Monte-Carlo-Methode zum Einsatz. Sie ermöglicht es, eine große Anzahl von Nadeln zu werfen, wobei die Anzahl der Versuche nicht unbedingt zu einer höheren Genauigkeit der Approximation von π beitragen. (vgl. Zuber, 2014, S. 2ff; Havil, 2009, S. 70; Nahrstedt, 2015, S. 33f)

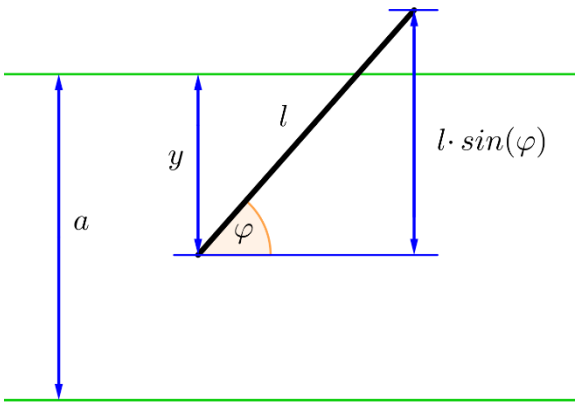

Abbildung 8: Das Modell zum Nadelexperiment: Die Nadel kreuzt eine Linie (eigene Abbildung)

3.2.2 Das Experiment und die Kreiszahl π

Folglich definieren wir in diesem Experiment also zwei Zufallsgrößen. Zum einen ist der Winkel φ zufällig, für ihn gilt $0 \leq \varphi \leq \pi$. Auch ist der Abstand zur oberen Linie, also die Position der Nadel y zufällig, für sie gilt $0 \leq y \leq a$, denn die Nadel darf nicht länger als der Abstand zwischen den beiden Parallelen sein. Wollen wir mit diesen Vorgaben Nadeln werfen, werden sie eine Linie berühren, wenn:

$$y < l \cdot \sin(\varphi)$$

Sollte der Abstand zur oberen Linie größer sein als zum obersten Punkt der Nadel, liegt sie ausschließlich im Bereich zwischen den Linien. Doch zwischen all den Nadeln ist es nicht klar ersichtlich, womit nun die Zahl π bestimmt werden kann. Diese Erklärung führt uns einen Schritt weiter, und es wird etwas komplexer. (vgl. Zuber, 2014, S. 6; Havil, 2009, S. 70)

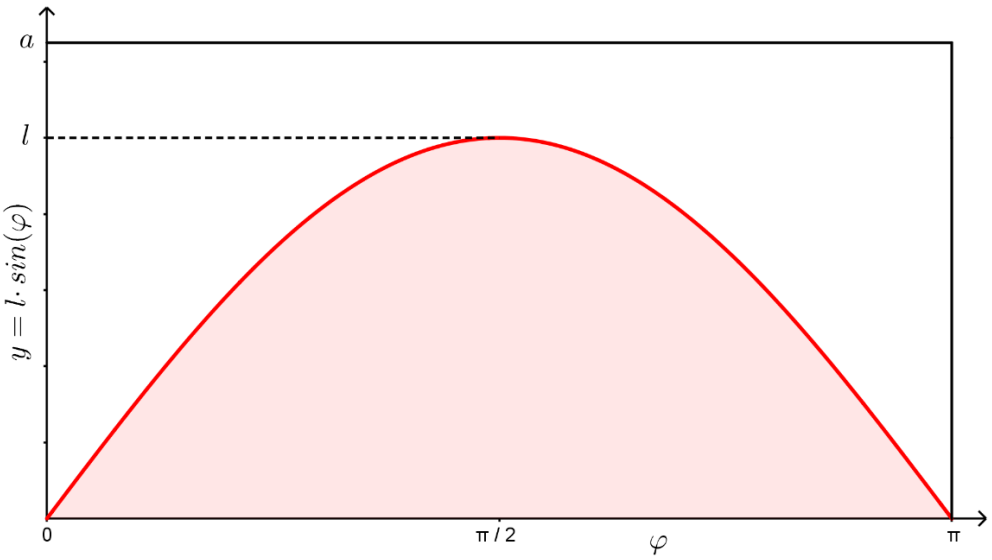

Abbildung 9: Grafische Darstellung aller Möglichkeiten des Nadelwurfs (eigene Abbildung)

Alle möglichen Varianten, wie die Nadel fallen könnte, lassen sich mit der Position und dem Winkel in einem Rechteck grafisch darstellen. Die Länge des Rechtecks ist dabei π, die Breite a. Dabei fällt auf, dass die Fläche unterhalb der Sinuskurve die Wahrscheinlichkeit für alle Ereignisse beschreibt, bei denen die Nadel auf einer Linie liegt. Auch bei diesem Beispiel bewegen wir uns also im Bereich der Flächen. Generiert man zwei Zufallsvariablen wie oben beschrieben, erhält man Punkte der Form $(\varphi|y)$ für die oben dargestellte Abbildung 9. Ist folglich unser y kleiner als der vertikale Abstand zwischen

unterer und oberer Position $l \cdot \sin(\varphi)$, wissen wir, dass sich der Punkt unterhalb der Kurve und somit auf der roten Fläche befindet. Ist die Bedingung unwahr, muss der Punkt außerhalb der Kurve liegen. (vgl. Zuber, 2014, S. 13)

Wir sind der Lösung schon sehr nahe. Die letzte Hürde ist es schließlich, das Verhältnis v zwischen der Fläche unter der Kurve zur Gesamtfläche des Rechtecks mit

$$v = \frac{\text{Fläche unter der Kurve}}{\text{Fläche des Rechtecks}} = \frac{\int_0^{\pi} l \cdot \sin(\varphi)\,\mathrm{d}\varphi}{\pi \cdot a} = \frac{2 \cdot l}{\pi \cdot a}$$

zu berechnen. Dabei erhalten wir die bemerkenswerte Schlussfolgerung, dass

$$\frac{2 \cdot l}{\pi \cdot a} \approx \frac{\text{Treffer}}{\text{Versuche}} \quad \rightarrow \quad \pi \approx \frac{2 \cdot l}{v \cdot a}$$

Mit diesem Verhältnis v haben wir im Endeffekt also die Wahrscheinlichkeit berechnet, dass eine Nadel auf einer Linie landet. Unser Ziel ist erreicht. Und weil es so kommen muss, lässt sich auch dieses Experiment in VBA einfach und anschaulich simulieren. (vgl. Havil, 2009, S. 71)

```
Sub MCM_Buffon()
    Dim laenge As Double
    Dim abstand As Double
    Dim winkel As Double
    Dim verhaeltnis As Double
    Dim i As Long
    Dim position_y As Double
    Dim treffer As Long
    Dim nadeln As Long
    Dim durchlaeufe As Long

    'Anzahl Durchlaufe und Nadeln, Abstand = 4, Länge = 3
    durchlaeufe = 100
    nadeln = 1000000
    abstand = 4
    laenge = 3

    Randomize
    For k = 1 To durchlaeufe
        treffer = 0
        For i = 1 To nadeln
            'Zufallszahlen für Winkel und Position y
            winkel = 3.14159 * Rnd()
            position_y = abstand * Rnd()
            'Abfrage, dass eine Nadel die Linie berührt
            If position_y < laenge * Sin(winkel) Then
                treffer = treffer + 1
            End If
        Next i
    verhaeltnis = treffer / nadeln
    Cells(k + 1, 1).Value = treffer
    Cells(k + 1, 2).Value = 2 * laenge / abstand / verhaeltnis
    Next k
End Sub
```

Abbildung 10: Quellcode des Nadelexperiments in VBA (eigene Abbildung)

Treffer	Berechnung	Abweichung
478156	3,1370515	0,0045411
477991	3,1381344	0,0034583
476976	3,1448123	0,0032197
478168	3,1369728	0,0046199
477743	3,1397634	0,0018292
476647	3,1469830	0,0053903
477104	3,1439686	0,0023760
477038	3,1444036	0,0028109
478076	3,1375765	0,0040162
477808	3,1393363	0,0022564
477567	3,1409205	0,0006721
477325	3,1425130	0,0009203
478242	3,1364874	0,0051053
476909	3,1452541	0,0036615
476465	3,1481851	0,0065924
477345	3,1423813	0,0007886
477367	3,1422365	0,0006438
478296	3,1361333	0,0054594
477988	3,1381541	0,0034386
477232	3,1431254	0,0015327
477801	3,1393823	0,0022104
477374	3,1421904	0,0005977
476897	3,1453333	0,0037406
476936	3,1450761	0,0034834
477355	3,1423155	0,0007228
477775	3,1395531	0,0020395
478216	3,1366579	0,0049347
476940	3,1450497	0,0034570
477930	3,1385349	0,0030577
477939	3,1384758	0,0031168

Nadelwurf

Abbildung 11: Die ersten 30 Durchläufe des Programms (eigene Abbildung)

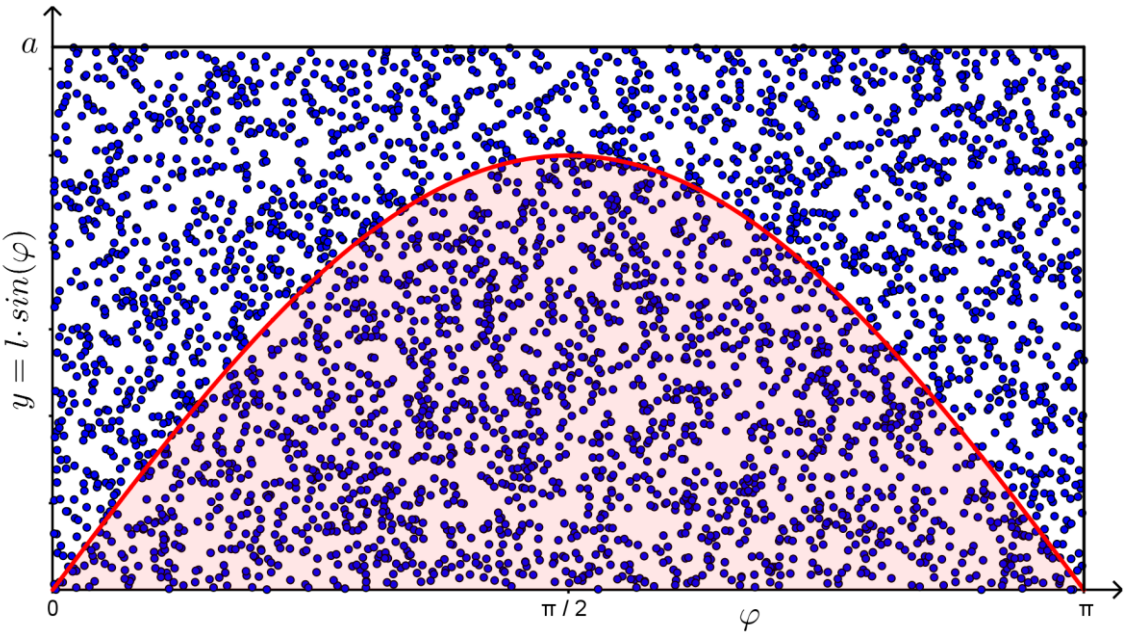

Abbildung 12: Dreitausend zufällig generierte Punkte im Phasenraum des Nadelwurfs (eigene Abbildung)

3.3 Der Weg eines Betrunkenen

Nach einer Legende soll die Monte-Carlo-Methode entstanden sein, als ein Mathematiker in Monte Carlo einen Betrunkenen beobachtete, der sich von einer Laterne wegbewegte. Dabei schwankte er so stark, dass er nicht wirklich vorwärtskam. Der Mathematiker fragte sich dann, wie weit sich der Betrunkene nach einer bestimmten Anzahl von Schritten wohl vom Laternenpfahl entfernen wird. Um eine Aussage über diese Wahrscheinlichkeit tätigen zu können, bedarf es aber einer Vielzahl von Beobachtungen in der gleichen Situation. Der Mathematiker wählte dann den eleganten Weg der Simulation, und nannte dieses Modell die Monte-Carlo-Methode. Hätten damals schon Computer existiert, hätte der Mathematiker die Bewegungen des Betrunkenen problemlos simulieren können. (vgl. Nahrstedt, 2015, S. 1, 5; Schneider, 2009)

3.3.1 Der lineare Weg eines Betrunkenen

Mit nachfolgendem Beispiel wird diese Beobachtung simuliert. Dabei gibt es zwei Möglichkeiten, wie sich der Betrunkene bewegen kann. Er kann entweder vorwärts- oder rückwärtsgehen. Vergleichen lässt sich dies mit dem Münzwurf aus 2.2, schließlich ist es nichts anderes als ein klassisches *Bernoulli-Experiment*. Wir können uns vorstellen, dass der Betrunkene vor jedem Schritt, den er setzt, eine Münze wirft. Kopf oder Zahl entscheiden darüber, in welche Richtung er schwanken wird. In unserem Programm entscheidet keine Münze die Richtung, sondern der Zufallsgenerator von VBA. Er erzeugt zufällig die Zahlen 0 und 1. Erstere lässt den Betrunkenen rückwärts bewegen, letztere vorwärts. Wir schicken nun den ersten Kandidaten von der Laterne aus los. Sein Weg lässt sich grafisch darstellen. Die vertikale Achse bezeichnet die Zeit in Schritten, die horizontale seine Position. Der Laternenpfahl befindet sich im Nullpunkt. (vgl. Nahrstedt, 2015, S. 5ff)

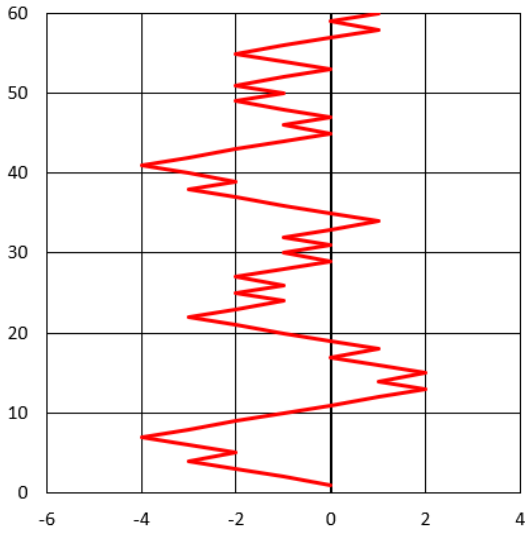

Abbildung 13: Simulierter Weg des ersten Betrunkenen (eigene Abbildung)

```
Sub MCM_Betrunkener()
    Dim b As Long 'Position des Betrunkenen
    Dim x As Double 'Zufallsvariable
    Dim anzahl_Betrunkene As Long
    Dim schritte As Long

    schritte = 100
    anzahl_Betrunkene = 1000

    Randomize
    For k = 1 To anzahl_Betrunkene
        b = 0
        For i = 1 To schritte
            x = Int(Rnd() * 2)
            If x = 0 Then
                b = b - 1
            Else
                b = b + 1
            End If
        Next i
    Cells(k, 1).Value = b
    Next k
End Sub
```

Abbildung 14: Programmcode für die Simulation von 1000 Betrunkenen in VBA (eigene Abbildung)

Wir wollen nun eintausend Betrunkene auf den Weg schicken, die sich alle jeweils hundert Schritte vom Laternenpfahl entfernen. Die Werte aller Endpositionen werden wieder in eine Excel-Tabelle übertragen, um deren Häufigkeit herauszufinden. Dies geschieht, indem wir zwischen -40 und 40 alle Endpositionen zählen. Erwartet wird, dass der Betrunkene wieder zu seiner Anfangsposition, der Laterne, zurückfindet. In der Praxis ist dies aber natürlich nicht der Fall. Auch handelt es sich lediglich um eine Stichprobe, eine erneute Durchführung würde neue Werte generieren. Die Verteilung würde wahrscheinlich ähnlich aussehen, eine Garantie dafür gibt es aber nicht. (vgl. Nahrstedt, 2015, S. 7)

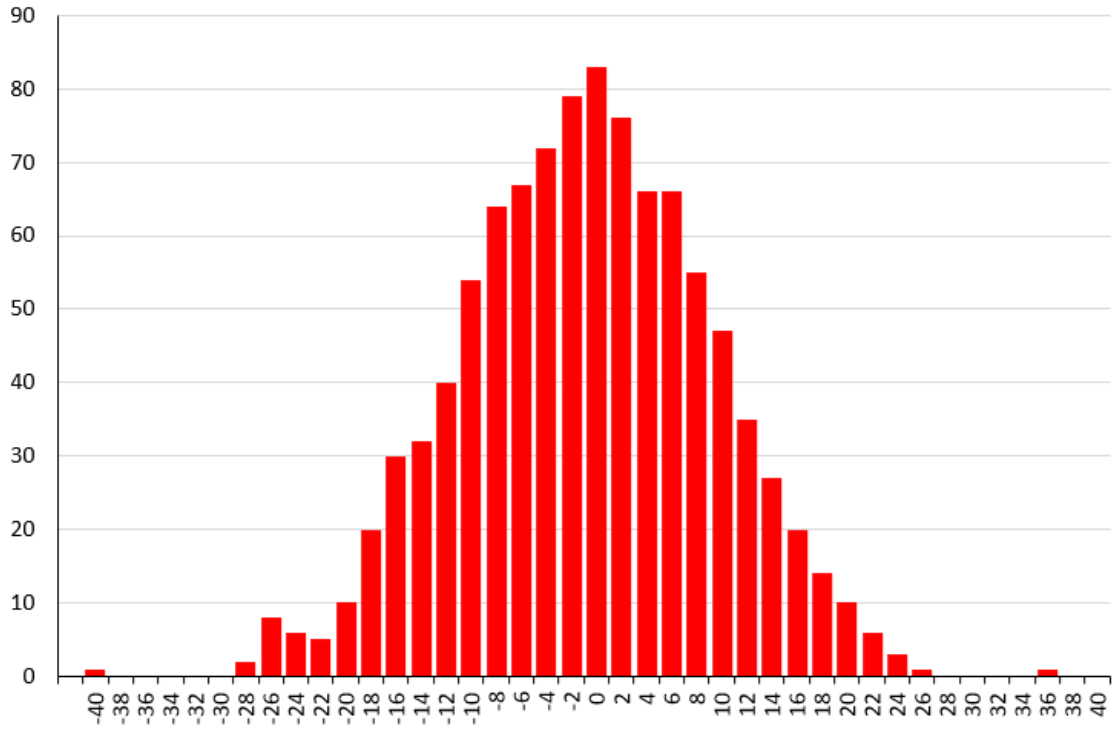

Abbildung 15: Häufigkeitsverteilung der Endpositionen von 1000 Betrunkenen (eigene Abbildung)

Wir sehen, dass der Erwartungswert tatsächlich der häufigste Wert im Diagramm ist. Die Endpositionen werden links und rechts davon aber immer weniger. Auch Ausreißer sind möglich, welche gut zu erkennen sind. Würde man die Anzahl der Betrunkenen erhöhen, nähert sich die Verteilung einer Normalverteilung an.

3.3.2 Der Weg eines Betrunkenen in der Ebene

Auch wenn die Ausgänge dieses Experiments schon recht spannend sind, ist es doch noch ein weniger fern von der Realität. Das lässt sich ändern, wenn wir zu den Richtungsmöglichkeiten noch links und rechts hinzufügen. Dadurch wird es dem Betrunkenen möglich, auch auf die Seite zu schwanken.

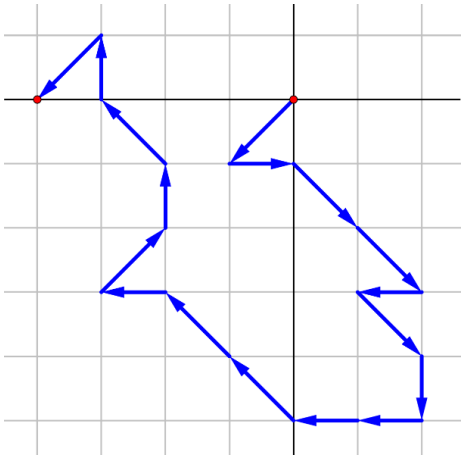

Abbildung 16: Zufällig generierter Weg des ersten Betrunkenen in der Ebene (eigene Abbildung)

Möglich wird das, wenn wir anstatt einer gleich zwei Pseudozufallszahlen erzeugen. Beide Zahlen können die Werte -1, 0 oder 1 annehmen. Sie bestimmen die Richtung des Betrunken im Koordinatensystem als negative oder positive Bewegung. Den Fall, dass er sich auf einer Achse nicht bewegt, wird mit dem Ausgang 0 beschrieben. Dass beide Zufallszahlen 0 sind und sich der Betrunkene somit nicht bewegt, ist jedoch ausgeschlossen. Er wird sich also mit jedem Schritt in eine Richtung bewegen. Der Weg des ersten Betrunkenen wurde zufällig generiert und wird in Abbildung 16 dargestellt. Möglich wäre es genauso, dass er sich zu einem späteren Zeitpunkt wiederholt auf derselben Position befindet. Dieser Fall lässt sich aber nur schwer abbilden. (vgl. Nahrstedt, 2015, S. 10f)

x	y
3	-11
-1	3
8	1
-2	15
18	-8
-21	10
12	-5
18	-6
-4	-3
-13	1
20	6
2	-3
12	-6
-6	4
7	3
-2	9
-7	3
0	-4
10	4
-15	-6
5	-1
12	7
-17	8
4	3
-8	12

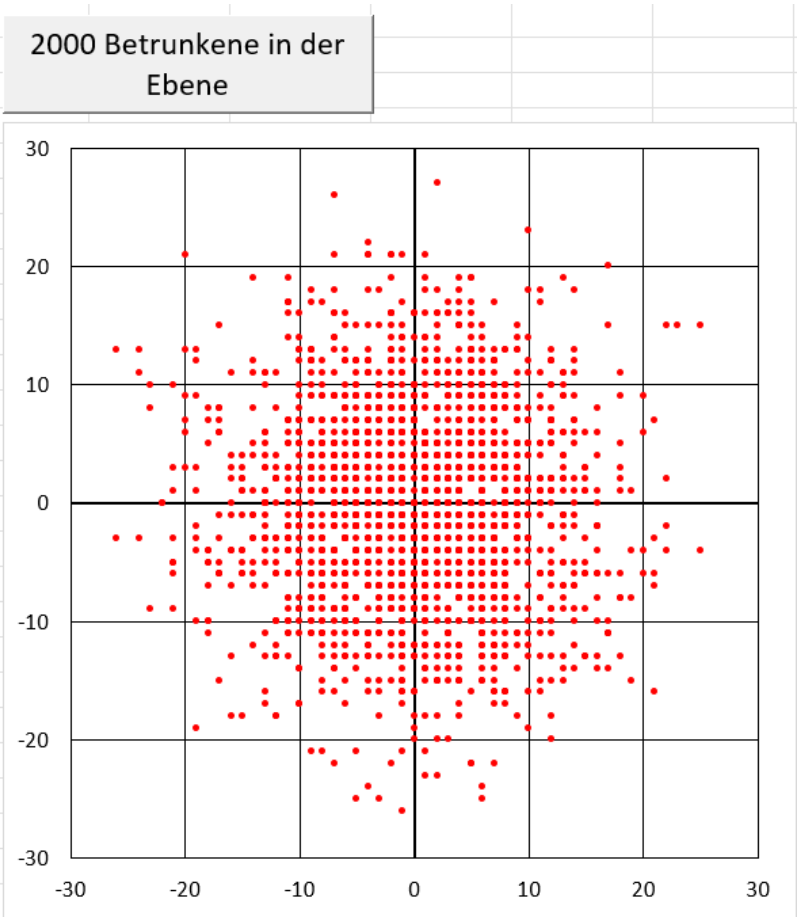

Abbildung 17: Simulation von 2000 zufällig generierten Endpositionen in Excel (eigene Abbildung)

Im Programm werden die Koordinaten der Endpositionen von zweitausend Betrunkenen erfasst. Die Endposition erreicht er nach hundert Schritten. Obwohl er so viele Schritte gesetzt hat, ist er meistens nicht weit von der Laterne zum Stehen gekommen. Dies hat einen einfachen Grund. Würde man die Endpositionen aller Kandidaten in einem Diagramm aufzeichnen, ließe sich wiederum sehr wahrscheinlich eine Verteilung um den Nullpunkt erkennen. Der Großteil wird also wieder am Laternenpfahl ankommen.

```
Sub MCM_Betrunkener_Ebene()
    Dim bx As Long 'x-Position des Betrunkenen
    Dim by As Long 'y-Position des Betrunkenen
    Dim x As Double
    Dim y As Double
    Dim anzahl_Betrunkene As Long
    Dim schritte As Long

    schritte = 100
    anzahl_Betrunkene = 2000

    Randomize
    For k = 1 To anzahl_Betrunkene
        bx = 0
        by = 0
        For i = 1 To schritte
            x = Int(Rnd() * 3 - 1)
            y = Int(Rnd() * 3 - 1)
            If x = 0 And y = 0 Then
                'Keine Veränderung
                i = i - 1
            Else
                bx = bx + x
                by = by + y
            End If
        Next i
    Cells(k, 1).Value = bx
    Cells(k, 2).Value = by
    Next k
End Sub
```

Abbildung 18: Programmcode für die Simulation der Betrunkenen in der Ebene in VBA (eigene Abbildung)

3.4 Das Galtonbrett

Wir haben bereits in vorherigen Beispielen die Normalverteilung kennengelernt. Sie ist eine stetige Verteilung und zeichnet sich dadurch aus, dass ihr Verlauf mit einer Funktion beschrieben werden kann und mithilfe der Integralrechnung die Wahrscheinlichkeiten berechnet werden können. Die Fläche unter jeder Kurve solcher Art beträgt 1, da sie alle Wahrscheinlichkeiten umfasst. Die Binomialverteilung hingegen ist eine diskrete Verteilung, bei der jeder Versuch jeweils nur zwei mögliche Ergebnisse hat, in folgendem Beispiel Rechts oder Links. Es ist also jeder Teilversuch der Binomialverteilung ein uns schon bekanntes *Bernoulli-Experiment* mit der Erfolgswahrscheinlichkeit p und deren Gegenwahrscheinlichkeit $q = 1 - p$. Die Anzahl aller Versuche wird mit n beschrieben, während k die Anzahl der Erfolge beschreibt. Das Galtonbrett ist ein Experiment, das entwickelt wurde, um mit dessen Hilfe die Binomialverteilung anschaulich zu demonstrieren. Es ist bestens dazu geeignet, die Begriffe der Wahrscheinlichkeitsrechnung kennenzulernen. (vgl. Nahrstedt, 2015, S. 37)

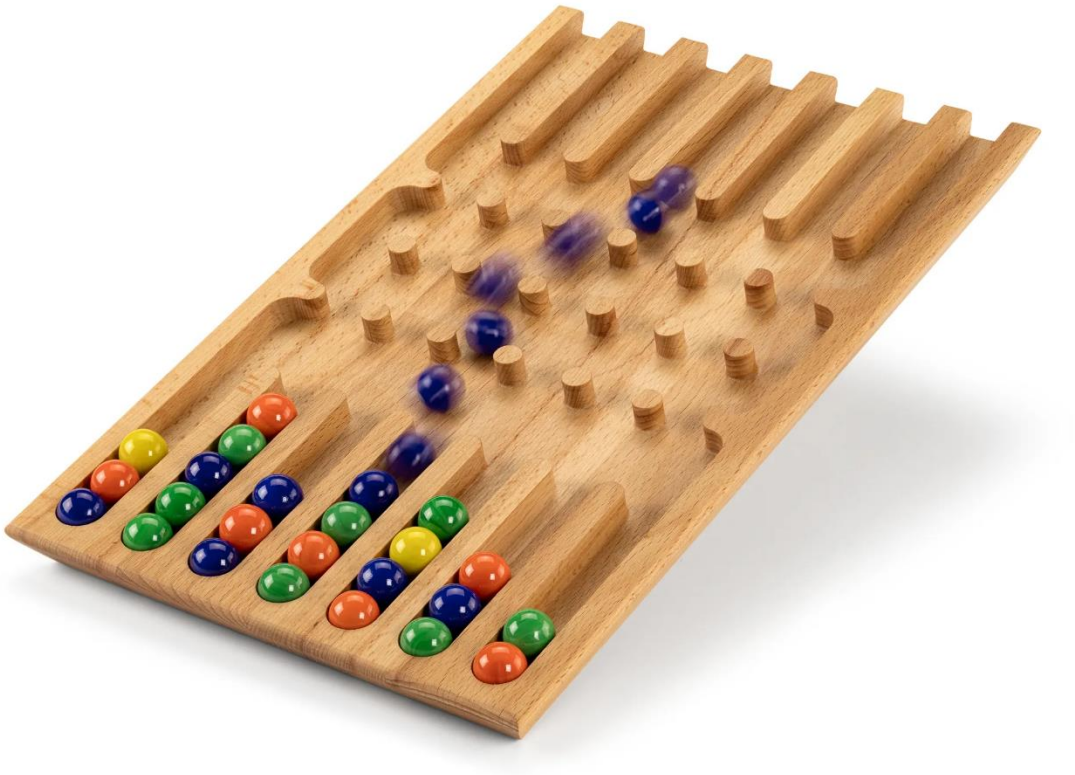

Abbildung 19: Reales Galtonbrett (Manufactum Handels GmbH)

3.4.1 Sir Francis Galton

Namensgeber des Experiments ist Sir Francis Galton, ein britischer Naturforscher und Schriftsteller, der von 1822 bis 1911 lebte und wirkte. Er war Cousin des berühmten Charles Darwin, der mit seiner Evolutionstheorie weltweite Bekanntheit erlangte. Galtons Tätigkeitsbereiche waren sehr weit gestreut. Er studierte zuerst Medizin, war aber in verschiedensten Bereichen, wie der Meteorologie, Statistik, Anthropologie oder Psychologie tätig und berichtete ausführlich über seine vielzähligen Afrikareisen. Besonders bekannt ist er als Begründer der Eugenik. Sie ist die Lehre, menschliches Erbgut zu verbessern. 1909 wurde er von König Eduard VII. für seine Verdienste zum Ritter geschlagen. (vgl. Kritische Psychologie Marburg, kein Datum)

3.4.2 Der Versuch

Beim Galtonbrett handelt es sich nun um ein Brett, auf dem Stifte in einem Dreieck befestigt sind, das auch als Pascalsches Dreieck bekannt ist. Dieses zeichnet sich dadurch aus, dass die Stifte in einer Ebene jeweils die Summe der zwei darübergelegenen Stifte sind und wie gleichmäßige Dreiecke angeordnet werden. Es werden dann eine beliebige Anzahl Kugeln auf den ersten Stift geworfen. Bei jedem Stift haben die Kugeln die Möglichkeit, nach Links oder Rechts zu fallen, beides mit einer Wahrscheinlichkeit von ½. Unter den untersten Stiften befinden sich dann Auffangbehälter, in diese die Kugeln fallen. (vgl. Runge, 2020, S. 142; Nahrstedt, 2015, S. 37)

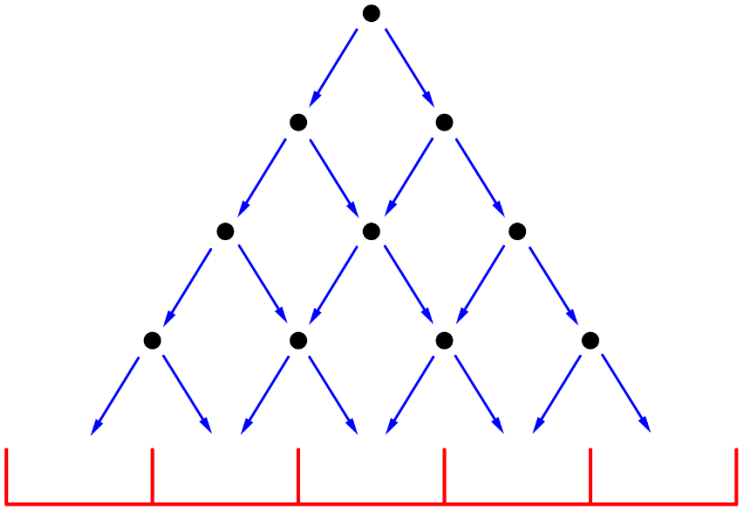

Abbildung 20: Das Galtonbrett mit vier Ebenen (eigene Abbildung)

Wirft man nun eine genügende Anzahl Kugeln ein, werden die meisten Kugeln in den mittleren Behältern liegen bleiben. In den äußeren Fächern wird die Anzahl geringer ausfallen, weil die Wahrscheinlichkeit dafür kleiner ist. Die Wahrscheinlichkeit, dass eine Kugel in ein bestimmtes Fach fällt, lässt sich mit der Binomialverteilung beschreiben. In VBA lässt sich dann ein Galtonbrett mit n Ebenen simulieren, in das zum Beispiel 5000 Kugeln geworfen werden. Da wir von einem symmetrischen Brett ausgehen, beträgt unsere Erfolgswahrscheinlichkeit $p = 0{,}5$. Die Wahrscheinlichkeiten lassen sich mit der allgemeinen Formel der Binomialverteilung beschreiben.

$$P(X = k) = \binom{n}{k} \cdot p^k \cdot (1-p)^{n-k}$$

Da die Wahrscheinlichkeit, sowie deren Gegenwahrscheinlichkeit ident sind, lässt sich die Formel mithilfe der Potenzregeln vereinfachen. Potenzen gleicher Basis werden multipliziert, indem man die Hochzahlen addiert. Das k fällt somit weg und wir erhalten die Formel:

$$P(X = k) = \binom{n}{k} \cdot 0{,}5^n$$

Würden wir beispielsweise mit einem schiefen Galtonbrett rechnen, würden sich die Wahrscheinlichkeiten ändern und diese Vereinfachung wäre nicht möglich. Wir kommen aber in diesen Genuss und fahren fort.

Wir erzeugen wieder eine binomialverteilte Zufallszahl, die die Werte 0 und 1 annehmen kann. Dass die Kugel nach links fällt, wäre dabei 0, während 1 das Ereignis beschreibt, dass die Kugel nach rechts fällt. Beide Ausgänge sind gleich wahrscheinlich und wird durch den Zufallsgenerator in VBA simuliert. Sie entscheidet über die Fallrichtung der Kugel. Zum Verständnis nummerieren wir die Fächer und beginnen links mit dem Wert 1. Dass eine Kugel in dieses Fach fällt, wird mit $k = 0$ beschrieben. Fällt eine Kugel in allen Ebenen nach links, wird sie in diesem Fach landen. Fällt die Kugel aber einmal nach rechts, wird im Programm ihre Position um 1 erhöht. Anhand der Position lässt sich k, und damit die Nummer des Faches bestimmen. Somit lässt sich für jede Kugel ihr Durchlauf durch das Brett simulieren und ihr Fach bestimmen, in das sie fällt. An jeder Endposition, also in jedem Fach, werden dann die Kugeln gezählt. Dabei ist die Anzahl der Ebenen n durch den Benutzer frei wählbar. Um zu überprüfen, wie weit die Simulation von den jeweiligen Erwartungswerten abweicht, lassen sich diese mit obiger Formel für jedes Fach k berechnen. Dazu muss im Programm für alle n Ebenen der Binomialkoeffizient berechnet werden. Am einfachsten geschieht dies mit einer rekursiven Darstellung. Unser Anfangswert ist $b = 1$, welcher für die Berechnung der jeweiligen Fächer notwendig ist. In der Ausgabe werden dann damit die erwarteten Häufigkeiten bestimmt. (vgl. Pollok, 1999; Wagner, 2006, S. 9f)

In Excel wird das Experiment mit 5000 Kugeln, die durch 10 Ebenen fallen, simuliert. In der Grafik ist erkennbar, dass die Anzahl Kugeln in den Fächern nicht stark von der erwarteten Anzahl abweicht. Außerdem lässt sich bereits eine Normalverteilung erkennen. Denn mit größerer Anzahl der Kugeln und Ebenen würden die Häufigkeiten gegen die Normalverteilung konvergieren. Man spricht dabei vom zentralen Grenzwertsatz von *Moivre-Laplace*. Eine Binomialverteilung nähert sich also der Normalverteilung, je größer die Ereigniszahl ist. Diese spannende Erkenntnis lässt sich wie in Abbildung 21 mit einem Programm in VBA durchführen. (vgl. Nahrstedt, 2015, S. 38)

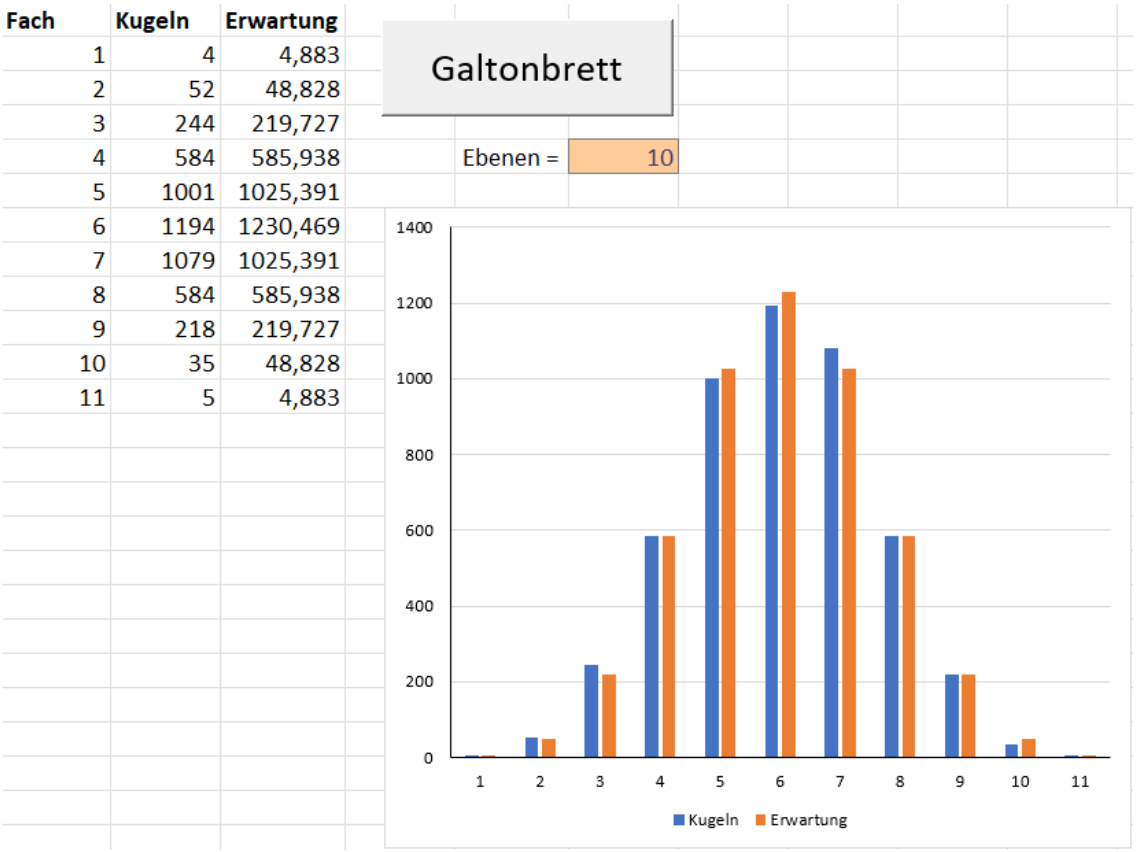

Abbildung 21: Ein Ergebnis des Galton-Experiments in Excel (eigene Abbildung)

```
Sub MCM_Galton()

Dim h() As Long
Dim Faecher() As Long
Dim n As Long
Dim k As Long
Dim p As Double

n = Cells(5, 6).Value
ReDim Faecher(n) As Long
ReDim h(n) As Long

    kugeln = 5000
    ebenen = n

    Range("A2:C100").ClearContents

    Randomize
    For k = 1 To kugeln
    position = 0
        For i = 1 To ebenen
            'Zufallszahl (0; 1)
            x = Int(Rnd() * 2)
            If x = 1 Then
                position = position + 1
            End If
        Next i
        'Zähler für die jeweilige Position der Kugel
        Faecher(position) = Faecher(position) + 1
    Next k

    'Berechnung des Binomialkoeffizienten
    For k = 0 To n
        b = 1
        For i = 1 To k
            b = b * (n + 1 - i) / i
        Next i
        h(k) = b
    Next k

    'Ausgabe
    For j = 1 To n + 1
        Cells(j + 1, 1).Value = j
        'Anzahl Kugeln
        Cells(j + 1, 2).Value = Faecher(j - 1)
        'Berechnung der erwarteten Häufigkeit
        Cells(j + 1, 3).Value = h(j - 1) * 0.5 ^ n * kugeln
    Next j

End Sub
```

Abbildung 22: Programmcode des Galtonbretts in VBA (eigene Abbildung)

4. Zusammenfassung und Ausblick

Wir befinden uns nun im letzten Kapitel dieser Arbeit und wollen abschließend die Erkenntnisse zusammenfassen, welche sich während des Schreibprozesses ergeben haben. Mit einfachen Anwendungsbeispielen wurde erkennbar, dass die Monte-Carlo-Methode ein geeignetes Mittel ist, Zusammenhänge zu erforschen und selbst Zufallsexperimente durchzuführen. Die Wahrscheinlichkeitsrechnung und die Statistik haben sich als zwei sehr spannende Bereiche der Mathematik erwiesen, in denen verschiedenste Experimente mithilfe des Zufalls möglich werden. Auch geschichtlich besitzt es eine bemerkenswerte Relevanz. Mit einer so simplen Methode war es schon sehr früh möglich, die Kreiskonstante π zu approximieren. Auch die Binomialverteilung und ihr Konvergenzverhalten zur Normalverteilung ist ein spannendes Feld, in der die Monte-Carlo-Methode angewendet werden kann. Außerdem bin ich während meiner Recherche auf unzählige weitere Anwendungsgebiete gestoßen, die ich zum Teil gar nicht erwähnt habe, geschweige denn im Detail behandeln konnte. Dafür reicht der Umfang einer VWA nicht aus. Besonders in der Physik gäbe es Anwendungen von hoher Bedeutung, wie zum Beispiel der Atombombe oder einer einfachen Simulation von Neutronenbewegungen. In all diesen Bereichen vereinfacht die Monte-Carlo-Methode die Arbeit in der Forschung. Diese sind im Gegensatz zu den theoretisch-mathematischen Beispielen praktische Anwendungen, die tatsächlich verwendet werden, allerdings deswegen umso komplexer sind.

Zusammenfassend ermöglicht die Monte-Carlo-Methode also eine schnelle und kostengünstige Berechnung diverser Probleme und Fragestellungen. Die Anzahl der Wiederholungen kann dabei beliebig variiert werden. Dadurch ergibt sich ein rasches Verständnis verschiedener Sachverhalte, und es können Aussagen darüber getroffen werden. Sie dient nicht unbedingt einer exakten Berechnung, dafür ist sie leider zu ungenau.

Mein Vorhaben war es ursprünglich, die Berechnungsprogramme in alternativen Programmiersprachen, beispielsweise Python oder JavaScript umzusetzen, da ich darin bereits einige Erfahrungen gesammelt habe. Mir wurde allerdings sehr rasch bewusst, dass neben den Einschränkungen, die VBA besitzt, ein wesentlicher Vorteil entscheidend ist. In allen anderen Sprachen wäre ich angewiesen gewesen, eigene Benutzeroberflächen

zu erstellen, um die Ergebnisse übersichtlich darzustellen. Dadurch, dass VBA in Excel integriert ist, konnte ich auf dessen Oberfläche zurückgreifen und problemlos programmieren. Zusätzlich wäre die Arbeit zu groß gewesen, eigene Grafiken zu erstellen, welche in Excel mit überschaubarem Aufwand visualisierbar sind. Deswegen war schon am Anfang der Arbeit absehbar, dass VBA die unkomplizierteste Lösung sein wird.

5. Literatur- und Quellenverzeichnis

Dr. Lehner, H. (20. November 2017). *Zufall / Programmieren mit Matlab.* Von Departement Informatik, ETH Zürich: https://lec.inf.ethz.ch/baug/informatik1/2017/slides/BAUG_Matlab1.pdf abgerufen

Frey, H. C., & Nießen, G. (2001). *Monte Carlo Simulation: Quantitative Risikoanalyse für die Versicherungsindustrie.* München: Gerling Akademie Verlag.

Havil, J. (2009). *Verblüfft?! Mathematische Beweise unglaublicher Ideen.* Berlin Heidelberg: Springer-Verlag.

Kernbichler, W., & Theis, C. (2002). Grundlagen der Monte Carlo Methoden. TU Graz: Institut für Theoretische Physik.

Kohlas, J. (1971). Die Monte Carlo Methode. Zürich: Springer-Verlag Berlin.

Kritische Psychologie Marburg. (kein Datum). *Sir Francis Galton: Begründer der Differenziellen Psychologie – Begründer der Eugenik*. Abgerufen am 13. Februar 2023 von https://www.kritische-psychologie.de/2007/sir-francis-galton-begruender-der-differenziellen-psychologie-begruender-der-eugenik

Lemieux, C. (2009). *Monte Carlo and Quasi-Monte Carlo sampling.* New York: Springer Science+Business Media, Inc.

Manufactum Handels GmbH. (kein Datum). Gesellschaftsspiel Galtoni. Abgerufen am 20. Februar 2023 von https://assets.manufactum.de/p/208/208329/208329_02.jpg/gesellschaftsspiel-galtoni.jpg

Mathepedia. (kein Datum). *Monte-Carlo-Methode*. Abgerufen am 30. Dezember 2022 von https://mathepedia.de/Monte-Carlo-Methode.html

Meintrup, D., & Schäffler, S. (2005). *Stochastik: Theorie und Anwendungen.* Berlin: Springer Verlag.

Microsoft. (7. April 2022). *Randomize-Anweisung*. Abgerufen am 14. Januar 2023 von https://learn.microsoft.com/de-de/office/vba/language/reference/user-interface-help/randomize-statement

Microsoft. (6. April 2022). *Rnd-Funktion*. Abgerufen am 14. Januar 2023 von https://learn.microsoft.com/de-de/office/vba/language/reference/user-interface-help/rnd-function

Minhorst, A. (1. Juni 2015). *Zufallszahlen generieren und verwenden*. Abgerufen am 14. Januar 2023 von https://access-basics.de/index.php/Zufallszahlen_generieren_und_verwenden.html

Müller-Gronbach, T., Novak, E., & Ritter, K. (2012). *Monte Carlo-Algorithmen.* Berlin Heidelberg: Springer.

Nahrstedt, H. (2015). *Die Monte-Carlo-Methode: Beispiele unter Excel VBA* (1. Ausg.). Wiesbaden: Springer Vieweg.

Pollok, B. (1999). *Das Galton - Brett: Ein schneller Zugang zu Binomialverteilungen.* Von http://www.kmk-format.de/material/Mathematik/5-2-Unterrichtsentwicklung-Szenarien_fuer_die_Fachgruppensitzung/5-2-1-3_Herrn_Galtons_rollende_Kugeln-ein_Weg_zur_Wahrscheinlichkeit/5-2-1-3-5_Einfuehrung_Grundlagenwissen_zum_Galtonbrett.pdf abgerufen

RiskNET. (kein Datum). *Monte-Carlo-Simulation*. Abgerufen am 30. Dezember 2022 von https://www.risknet.de/wissen/rm-methoden/monte-carlo-simulation/

Runge, B.-U. (17. Januar 2020). *Versuche zur Mechanik: Galton-Brett.* Von Universität Konstanz: https://ap.physik.uni-konstanz.de/ap-public/Anleitungen/Galton-Brett.pdf abgerufen

Schneider, G. (2009). *Die Monte Carlo Simulation.* München: GRIN Verlag.

Spektrum.de. (2017). *Monte-Carlo-Methode*. Abgerufen am 4. Januar 2023 von https://www.spektrum.de/lexikon/mathematik/monte-carlo-methode/6528

Springob. (2004). *Statistische Berechnung von Pi: Wikipedia CC BY-SA 3.0*. Abgerufen am 30. Dezember 2022 von https://de.wikipedia.org/wiki/Monte-Carlo-Simulation

Wagner, S. (Dezember 2006). *Kombinatorik.* Von TU Graz: https://www.math.tugraz.at/~wagner/KombSkr.pdf abgerufen

Zuber, A. (Januar 2014). *Das Nadelproblem von Buffon.* Von Universität Graz: https://imsc.uni-graz.at/baur/lehre/WS2013-Seminar/S15.pdf abgerufen

6. Abbildungsverzeichnis

Abbildung 1: Berechnung der Kreiszahl π (Springob, 2004) 8

Abbildung 2: 2000 zufällig generierte Punkte im Viertelkreis (eigene Abbildung) 13

Abbildung 3: Zwanzig Durchläufe des Programms in Excel (eigene Abbildung) 14

Abbildung 4: Quellcode des Algorithmus in VBA (eigene Abbildung) 14

Abbildung 5: Quellcode des Programms in VBA (eigene Abbildung) 16

Abbildung 6: Zwanzig Berechnungen des Integrals in Excel (eigene Abbildung) 16

Abbildung 7: Zweitausend zufällig generierte Punkte auf dem Integral in GeoGebra (eigene Abbildung) 16

Abbildung 8: Das Modell zum Nadelexperiment: Die Nadel kreuzt eine Linie (eigene Abbildung) 17

Abbildung 9: Grafische Darstellung aller Möglichkeiten des Nadelwurfs (eigene Abbildung) 18

Abbildung 10: Quellcode des Nadelexperiments in VBA (eigene Abbildung) 19

Abbildung 11: Die ersten 30 Durchläufe des Programms (eigene Abbildung) 19

Abbildung 12: Dreitausend zufällig generierte Punkte im Phasenraum des Nadelwurfs (eigene Abbildung) 20

Abbildung 13: Simulierter Weg des ersten Betrunkenen (eigene Abbildung) 21

Abbildung 14: Programmcode für die Simulation von 1000 Betrunkenen in VBA (eigene Abbildung) 21

Abbildung 15: Häufigkeitsverteilung der Endpositionen von 1000 Betrunkenen (eigene Abbildung) 22

Abbildung 16: Zufällig generierter Weg des ersten Betrunkenen in der Ebene (eigene Abbildung) 23

Abbildung 17: Simulation von 2000 zufällig generierten Endpositionen in Excel (eigene Abbildung) 24

Abbildung 18: Programmcode für die Simulation der Betrunkenen in der Ebene in VBA (eigene Abbildung) 25

Abbildung 19: Reales Galtonbrett (Manufactum Handels GmbH) 26

Abbildung 20: Das Galtonbrett mit vier Ebenen (eigene Abbildung) 27

Abbildung 21: Ein Ergebnis des Galton-Experiments in Excel (eigene Abbildung) 29

Abbildung 22: Programmcode des Galtonbretts in VBA (eigene Abbildung) 30

7. Glossar

Algorithmus:	Rechenvorgang nach einem bestimmten Schema
Analytisch:	auf einem logisch zergliedernden Verfahren beruhend
Approximation:	Näherungsweise Berechnung
Determinismus:	Bestimmbarkeit
Makros:	zusammengefasste Folge von Anweisungen oder Deklarationen in einem Programm
Malware:	Computerprogramme, die Schaden anrichten
Numerisch:	Verwendung von Zahlen
Probabilismus:	Die Wahrscheinlichkeit betreffend
Pseudozufallszahlen, pseudozufällig:	Zahlen scheinen zufällig zu sein, sind aber berechenbar
Simulation:	Nachbildung von realen Szenarien
VBA (siehe Seite 7):	Visual Basic for Applications, Skriptsprache in Office-Programmen

Selbstständigkeitserklärung

Name: Tobias Buchli

Ich erkläre, dass ich diese vorwissenschaftliche Arbeit eigenständig angefertigt und nur die im Literaturverzeichnis angeführten Quellen und Hilfsmittel benutzt habe.

Dornbirn, am 23.02.2023