Zufallskurven



Der britische Botaniker Robert Brown (links; 1773-1858) beschrieb 1827 die unregelmäßige, zuckende Bewegung von Pollen in einem Wassertropfen. Bereits 1785 hatte der holländische Arzt und Botaniker Jan Ingenhousz (1730-1799) beobachtet, dass Kohlenstaub-Teilchen in Alkohol sich ebenso verhalten. Die Bewegung ist wärmeabhängig und wurde erst 1905 von Albert Einstein theoretisch verstanden.



Die Brownsche Molekularbewegung kannst Du unter dem aufgebauten Mikroskop ansehen. Ein Video dazu findest Du unter http://www.youtube.com/watch?v=ra1mRVzqqck. Die Bewegung der Teilchen entsteht zufällig durch zahlreiche Zusammenstöße mit Wassermolekülen. Der Pfad eines Teilchens ist daher ein klassisches Beispiel einer Zufallskurve.

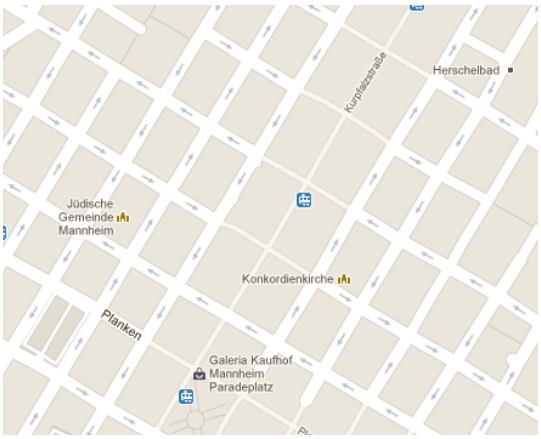
Durch die Zufallsbewegung werden die kleinen Teilchen auch über weitere Strecken transportiert. Öffnet man zum Beispiel einen Parfümflakon in einem Raum so erreicht der Duft nach einer gewissen Zeit jeden anderen Punkt in dem Raum, allein getrieben durch die zufällige Bewegung durch Stöße mit Luftmolekülen[1].

In diesem Modul kannst Du solche und andere Zufallsbewegungen mathematisch untersuchen und mit dem Computer simulieren. Um alles zu vereinfachen, benutzt man häufig das Modell einer zufälligen Bewegung auf einem Gitternetz. Diese Modellierung eines so genannten Random Walks auf einem Gitter kannst Du im nächsten Abschnitt anhand der zufälligen Bewegung einer völlig betrunkenen Person in einem Straßengitter kennen lernen. Dabei ist unter anderem interessant, wie weit so eine Person in einer bestimmten Zeit kommt (Luftlinie). Diese Erkenntnis wird dann auf die Ausbreitung von Krankheiten angewandt. Anschließend erfährst Du, wie man die Ausbreitung von Krankheiten realistischer modelliert. Reduziert auf eine Dimension finden Zufallsbewegungen Anwendung in der Spieltheorie und Finanzmathematik. Beim Glücksspiel ist zum Beispiel die Ruinwahrscheinlichkeit interessant. Diese kannst Du nach dem Durcharbeiten dieses Moduls exakt berechnen und auch in Maple simulieren. Zum Abschluss erfährst Du noch, wie man mit der Zahl π eine Zufallsbewegung programmiert.

[1] Dabei ist allerdings wichtig,andere Transportmechanismen wie Konvektion (Luftströmung)auszuschließen.

3.1 Ein desorientierter Fußgänger - Random Walks

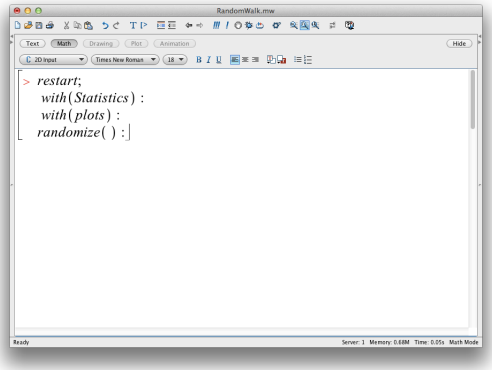




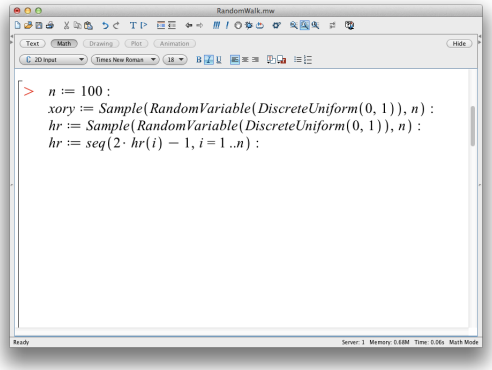
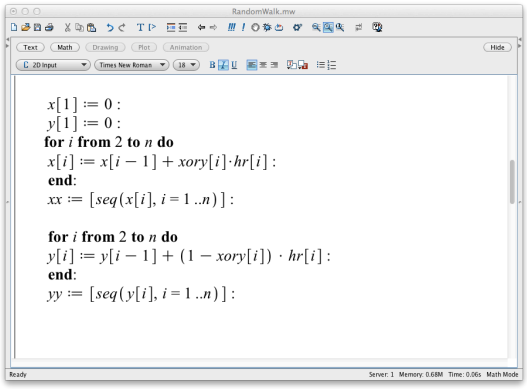
Obelix ist arg angeheitert, doch er muss noch nach Hause gehen. Er befindet sich in Mannheim, wo der Stadtplan einem regelmäßigen Straßengitter entspricht. In seinem Zustand ist Obelix völlig orientierungslos, so dass er an jeder Kreuzung herumtorkelt, und dann rein zufällig und gleichwahrscheinlich eine der 4 möglichen Richtungen einschlägt. Eventuell kehrt er also auch um und geht zurück. So eine Bewegung wird im Folgenden Random Walk genannt.

Das erste Ziel unserer Untersuchungen soll eine Simulation und eine graphische Darstellung der Bewegung des betrunkenen Obelix durch diesen Stadtplan sein. Dazu kannst Du mit 2 Münzwürfen an jeder Kreuzung zufällig die Richtung bestimmen. Schneller geht es aber mit dem Computer.

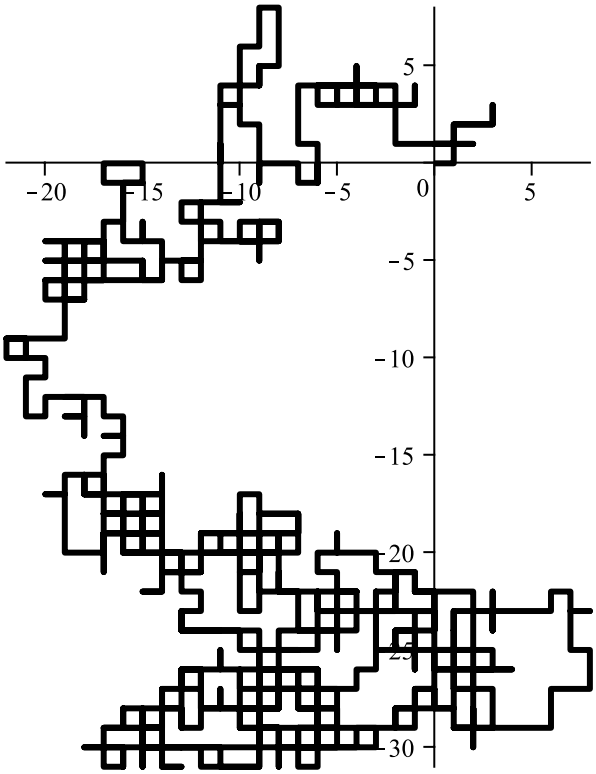
Simulation in Maple



Mit dem Kommando *restart;* wird Maple in den Anfangszustand versetzt. Mit Shift-Return kommt man eine Zeile weiter, ohne dass die Kommandos sofort ausgeführt werden. Da in dem Programm Statistik- und Plot-Befehle verwendet werden, lädt man zunächst die entsprechenden Pakete. Der Doppelpunkt schließt die Befehle so ab, dass kein Output angezeigt wird. Zum Schluss der Vorbereitung werden die Zufallszahlen, die im folgenden erzeugt werden, mit dem Befehl *randomize():* quasi durchgeschüttelt.

	*n:=100:* soll die Anzahl der Schritte (Kreuzungen) in diesem Beispiel sein. Nun werden für jeden Schritt 2 Zufallszahlen gezogen. In der Liste *xory* steht eine *1* für die x-Richtung und eine *0* für die y-Richtung. In der Liste *hr* (für hoch oder runter) soll eine *1* für die + Richtung und eine *-1* für die - Richtung stehen. Dazu zieht man zunächst *n* Zufallszahlen 0 oder 1, multipliziert sie dann mit 2 und subtrahiert 1: fertig!
	Der Random Walk kann starten. Bei (0l0) geht es los. In der Schleife **for** ... **end** werden zunächst die x-Koordinaten berechnet. Jedes Mal, wenn *xory(i)* gleich 1 ist, geht es einen Schritt nach links oder rechts. Ist *xory(i)* gleich 0, verändert sich die x-Koordinate nicht. In der 2. Schleife geschieht das gleiche für die y-Koordinate. Allerdings verwendet man nun *(1-xory(i))*.

Mit dem Befehl *pointplot(xx,yy,style=line,thickness=3);* kannst Du die Zufallsbewegung schließlich graphisch darstellen.



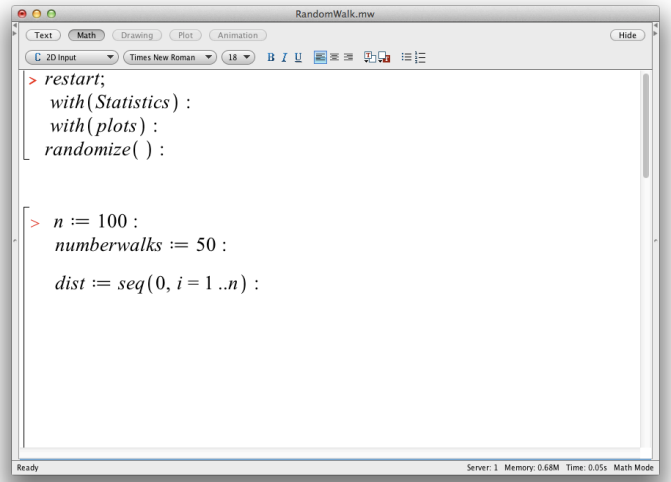
Übung:
Variiere das Maple-Programm. Erzeuge verschiedene Random Walks und studiere, welcher Anteil zum Startpunkt zurückkehrt. Variiere die Anzahl der Schritte und schätze grob ab, wie weit man in n Schritten kommt. (Genauer danach.)

Forschungsproblem
Wie weit reicht ein Random Walk im Mittel mit n Schritten (Luftlinie; für große Werte von n)?

Anders ausgedrückt:

Es dauert nicht mehr lange, und Obelix fällt sturzbetrunken zu Boden und bleibt wo er ist. Vorher schafft er noch n Schritte. Bestimme mit einem Maple-Programm, wie weit er sich im Mittel von seinem Startpunkt entfernt hat.

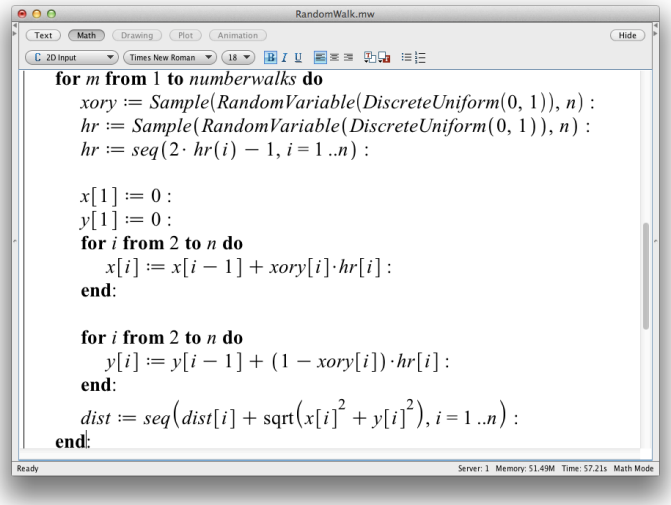
Dazu wird eine Anzahl von Random Walks in einem Maple-Programm erzeugt, jeweils die Entfernung vom Startpunkt bestimmt und der Mittelwert berechnet.



Zunächst muss die Anzahl der Schritte *n* und die Anzahl der Random Walks, über die gemittelt wird, definiert werden.

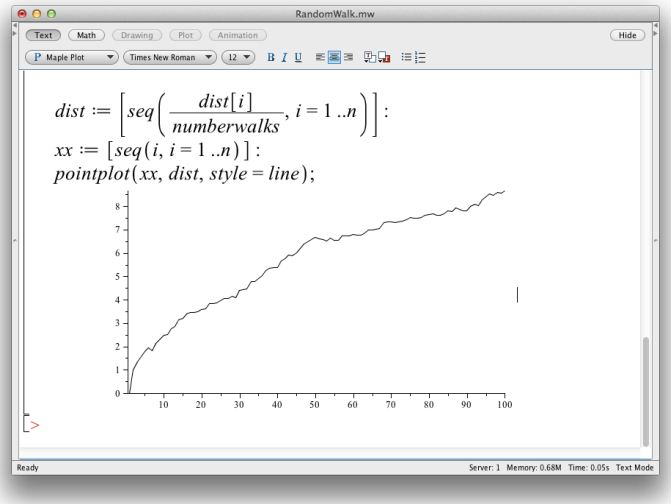
dist:=seq(0,i=1..n):

Damit setzt Du alle Distanzen zu Beginn der Simulation gleich *0*. Anschließend kannst Du nach jedem simulierten Random Walk die Distanzen berechnen, aufaddieren und zum Schluss durch die Anzahl der Pfade teilen.



dist:=seq(dist[i] + sqrt(x[i]²+y[i]²), i=1..n):

Der Abstand eines Punktes $P(x\,|\,y)$ vom Ursprung ist gleich $\sqrt{x^2+y^2}$. *sqrt* ist eine Abkürzung für Square Root, also Quadratwurzel.



dist := [seq(dist[i]/numberwalks, i = 1 .. n)]:

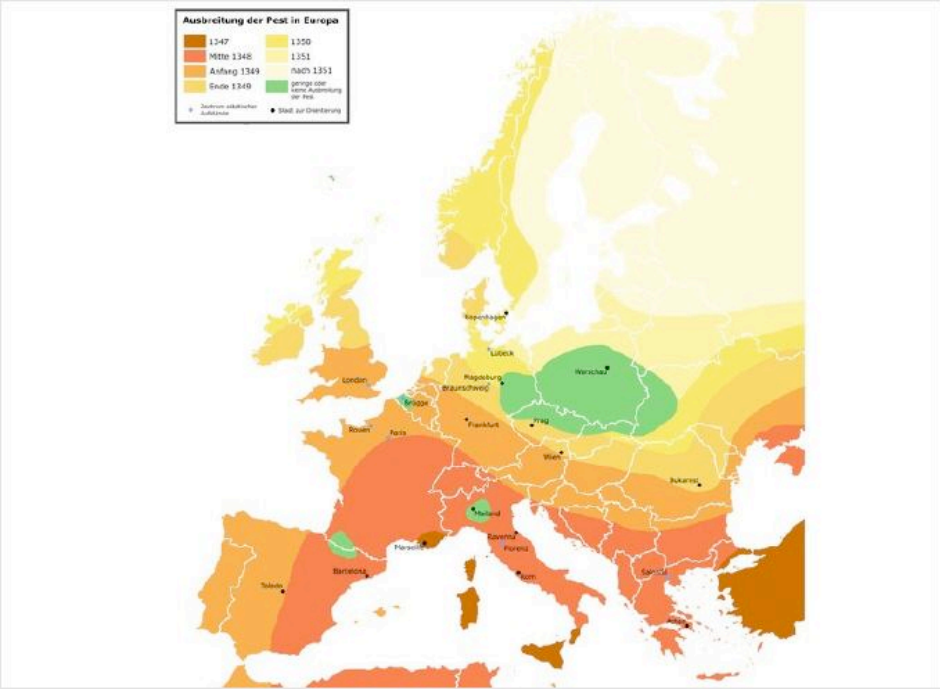
Damit berechnest Du den Mittelwert der Abstände vom Anfangspunkt.

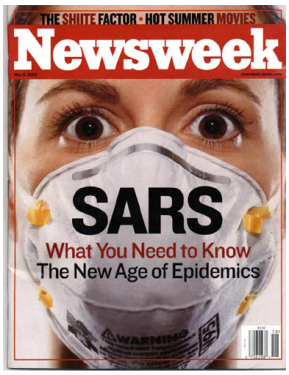
Übung: Berechne mit dem Programm die Abstände mit einer größeren Genauigkeit indem Du über mehr Zufallspfade mittelst (z.B. 1000).

***Übung*:** An welche Funktion erinnert Dich der Verlauf der Zuordnung
Anzahl der Schritte → Entfernung vom Startpunkt? Zeige, dass die Entfernung für eine große Anzahl von Schritten *n* proportional zu $\sqrt{n}$ ist. Nutze dazu die Daten aus Deinem Maple-Programm.

Ergebnis:
Die Entfernung, die eine einfache Zufallsbewegung (Random Walk) in n Schritten überbrückt, ist proportional zu $\sqrt{n}$. Dies stellt einen starken Gegensatz zu einer zielgerichteten Bewegung dar, bei der die zurückgelegte Entfernung in n Schritten proportional zu dieser Anzahl von Schritten ist. Die Entfernung bei einem Random Walk ist also deutlich geringer. Das Gesetz oben kann zum Beispiel bei der Brownschen Molekularbewegung nachgewiesen werden. Eine andere Situation kannst Du nun studieren.

3.2 Ausbreitung von Krankheiten - Der Lévy Flug

Ansteckende Krankheiten breiten sich in einer Gesellschaft ohne schnelle Reisemöglichkeiten nach dem Prinzip der Zufallsbewegung aus. Dies galt zum Beispiel für die erste Pandemie, die schwarze Pest im 14. Jahrhundert.	

	Diese Art der Krankheitsausbreitung ist recht langsam. In n Zeiteinheiten kommt die Krankheit nur $\sqrt{n}$ räumliche Einheiten weit. Infiziert die Krankheit nach 1 Woche zum Beispiel einen 10km großen Landkreis, so erreicht die Krankheit erst in 4 Wochen einen 20km großen Landkreis, und für die Ansteckung einer Region mit 100km Durchmesser würden 100 Wochen benötigt. In der Tat dauerte die Pest-Pandemie im Mittelalter 4 Jahre, bis schließlich ganz Europa durchseucht war, vgl. http://de.wikipedia.org/wiki/Schwarzer_Tod.
	Heute stellt sich die Situation jedoch anders dar. Mit modernen Verkehrsmitteln lassen sich in kurzer Zeit sehr große Entfernungen zurücklegen. Unsere Bewegung entspricht daher eher Random Walks, bei denen sich auch die Schreitweite zufällig ändert. Sehr große Schritte sind dabei zwar selten aber eben möglich - man spricht in diesem Zusammenhang von Lévy[2]-Flügen. Dadurch breiten sich Krankheiten heute sehr viel schneller aus und die Entstehung von Pandemien wie SARS wird begünstigt.

Bei der Erforschung der Ausbreitung von Seuchen aber auch bei der Erforschung von Gegenmaßnahmen spielen numerische Simulationen eine große Rolle. Im Folgenden kannst Du eine Simulation von Zufallspfaden mit variabler Schrittlänge schreiben und untersuchen.

```
# Vorbereitungen
restart:
with(Statistics):
with(plots):
randomize():

# Anzahl der Schritte
n := 1000:

# Ziehung der zufälligen Richtungswinkel (Bogenmaß)
alpha := Sample(RandomVariable(Uniform(0, 2*Pi)), n):

# Ziehung der zufälligen Schrittlängen
# Dazu wird die sogenannte Cauchy Verteilung genutzt,
# bei der große Zahlen mit einer gewissen Wahrscheinlichkeit
# vorkommen - abhängig von den Parametern , hier 1 und 0.05.
r := Sample(RandomVariable(CauchyDistribution(1, 0.05)), n):

# Berechnung der Schritte
dx := seq(r[i]*cos(alpha[i]), i = 1 .. n):
dy := seq(r[i]*sin(alpha[i]), i = 1 .. n):

# Berechnung des Levy Fluges
x[1] := 0:
y[1] := 0:
for i from 2 to n do
     x[i] := x[i-1]+dx[i]:
     y[i] := y[i-1]+dy[i]:
end:

# Graphische Ausgabe
xx := [seq(x[i], i = 1 .. n)]:
yy := [seq(y[i], i = 1 .. n)]:
pointplot(xx, yy, style = line);
```

Übung:
Gib das Programm in Maple ein und vergleiche den Lévy-Flug mit dem Random Walk eines Betrunkenen.

Übung:
Verändere das Simulationsprogramm so, dass die mittlere Entfernung nach n Schritten berechnet wird. Orientiere Dich beim Vorgehen an der Simulation des einfachen Random Walks oben.

Lévy-Flüge sind bei der Beschreibung vieler weiterer Phänomene interessant. Beim Sehen tasten die Augen Details genau ab, bis der Blickpunkt zum nächsten interessanten Fokus springt. Auf ähnliche Weise jagen Haie ihre Beute, erkunden Wespen ihre Umgebung, verläuft das Kinderspiel „Verstecken“.

Zum Weiterlesen
http://www.weltderphysik.de/de/7161.php
http://www.pro-physik.de/details/news/prophy10516news/news.html?laid=10516
http://en.wikipedia.org/wiki/Lévy_flight

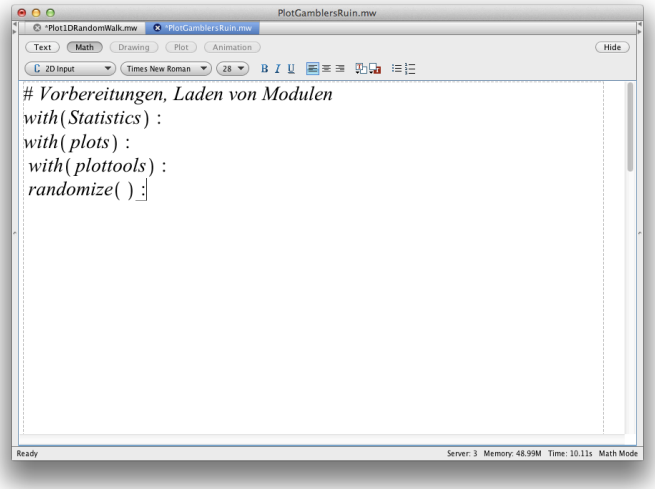
3.3 Gambler‘s ruin - Der eindimensionale Random Walk

Beim Roulette beobachtet man häufig die folgende Strategie. Ein Spieler setzt immer wieder den gleichen Einsatz auf *Rouge*. Fällt die Kugel auf eine der 18 roten Zahlen, so bekommt der Spieler das Doppelte seines Einsatzes ausgezahlt. Fällt die Kugel auf *noir* (ebenfalls 18 Zahlen) oder *Zéro*, so verliert der Spieler den Einsatz. Der Spieler spielt so lange, bis er entweder sein gesamtes Vermögen verspielt hat, also ruiniert ist, oder einen vorher avisierten Gewinn erzielt hat. Die Wahrscheinlichkeit, in einer Runde zu gewinnen ist $p = \frac{18}{37}$, die Wahrscheinlichkeit in einer Runde zu verlieren ist $q = \frac{19}{37}$.

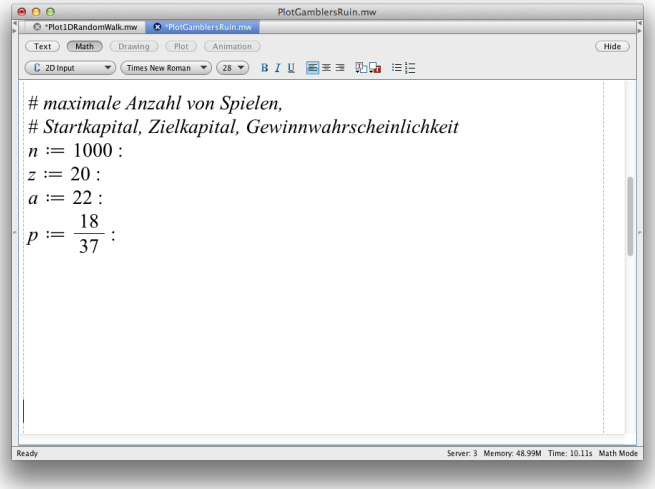


Etwas abstrakter ist die Spielbeschreibung so:0 Zwei Spieler spielen gegeneinander um Geld. Das Spiel geht über mehrere, eventuell viele Runden. In jeder Runde gewinnt Spieler A mit der Wahrscheinlichkeit p, und er verliert mit der Wahrscheinlichkeit q. Dabei ist $p+q=1$. Das Vermögen von Spieler A zu Beginn sei z, das Vermögen von beiden Spielern zusammen sei a. Die beiden Spieler spielen so lange, bis einer von beiden ruiniert ist, oder, anders ausgedrückt, bis einer der beiden das Vermögen a erreicht hat.

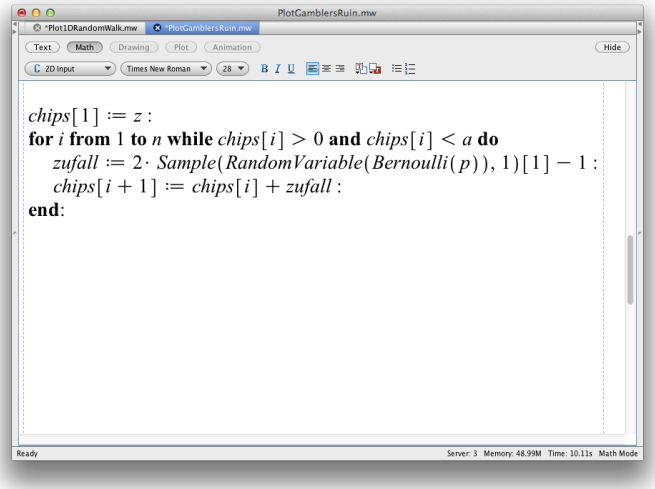
Vor einer Reihe theoretischer Überlegungen kannst Du nun zunächst das Spiel mit Maple simulieren. Neue Befehle benötigst Du kaum. Die Wahrscheinlichkeitsverteilung für einen Versuch heißt Bernoulli-Verteilung. Mit der Wahrscheinlichkeit p ist eine entsprechend verteilte Zufallsvariable gleich 1 und mit der Wahrscheinlichkeit $q = 1 - p$ ist sie gleich 0. In Maple kannst Du *n* solcher Zufallszahlen mit dem Kommando
Sample(RandomVariable(Bernoulli(p)),n);
erzeugen. Die Nullen müssen noch durch -1 ersetzt werden. Dies macht man am besten durch Multiplikation mit 2 und Subtraktion um 1.



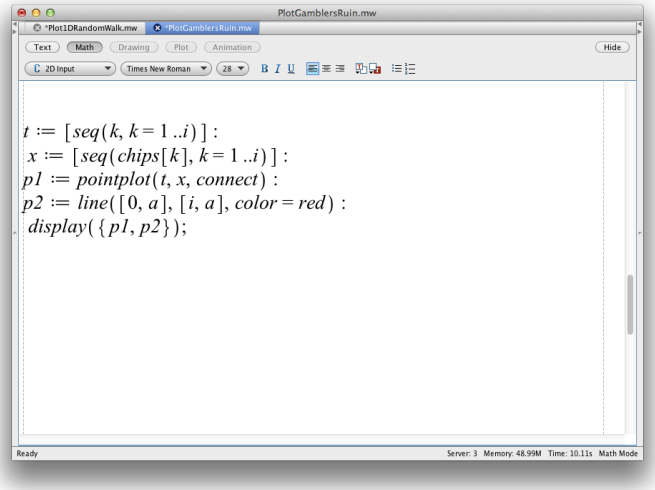
Zunächst müssen einige Befehle in Modulen geladen werden. Neu ist das Modul *plottools*, in dem das Grafikobjekt „Linie“ definiert ist, mit dem in der Ausgabe unten die Gewinngrenze angezeigt wird.



Dann legt man die Konstanten der Simulation fest. Diese kannst Du später verändern und so unterschiedliche Spiele analysieren. *n* ist die maximale Anzahl von Spielrunden, *z* ist das Startkapital, *a* ist das Zielkapital und *p* ist die Gewinnwahrscheinlichkeit pro Runde.

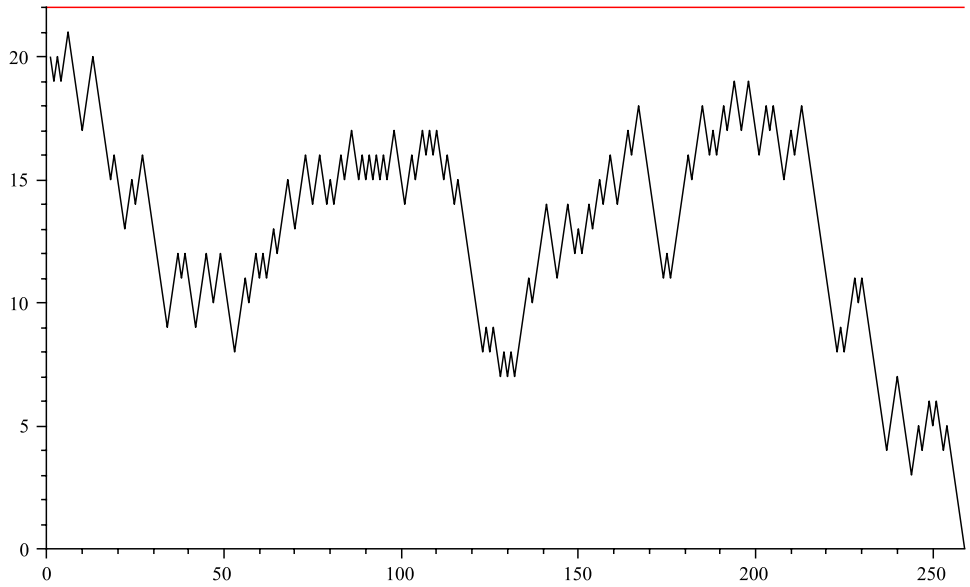


Hier geschieht die eigentliche Mathematik. Zunächst hat man *z* Chips. In der Schleife **for** .. **to** .. **while** .. **do** ... **end** wird jeweils eine Zufallszahl aus der Bernoulli-Verteilung wie oben beschrieben ermittelt. In dem gleichen Befehl wird eine 0 durch -1 ersetzt. Dann wird das Vermögen nach dieser Spielrunde bestimmt. Die Schleife ist beendet, wenn entweder die maximale Anzahl Spielrunden erreicht ist, oder der Spieler ruiniert ist oder das Zielvermögen erreicht ist.



Für die Ausgabe in einem Diagramm werden Listen erzeugt. *p1:= pointplot(t,x,connect):* bildet dann ein Grafikobjekt. Wenn Du den : durch ein ; ersetzt, wird das Diagramm auch direkt angezeigt. Hier wird aber erst ein weiteres Grafikobjekt erzeugt. Anschließend werden beide mit *display({p1,p2});* gemeinsam angezeigt.

Die Abbildung unten zeigt ein Beispiel für einen Spielverlauf beim Roulette, in dem der Spieler mit 20€ startet, das Ziel hat, 22€ zu gewinnen und nach 260 Spielrunden ruiniert ist.



Übung:

Mit dem Maple Programm kannst Du zahlreiche Forschungsfragen lösen. Diese Übung soll Dir als Anregung dienen, selber solche Fragen zu stellen und zu bearbeiten. Zunächst ist interessant, mit welcher Wahrscheinlichkeit der Spieler ruiniert wird bzw. sein Gewinnziel erreicht. Dies wird unten theoretisch hergeleitet, Du kannst aber auch in Maple mit einer weiteren **for**-Schleife eine Reihe von Spielverläufen erzeugen und die o.a. Wahrscheinlichkeit näherungsweise bestimmen. Wenn dies gelungen ist, kannst Du auch verschiedene Spielstrategien untersuchen und die Gewinnwahrscheinlichkeit in Abhängigkeit der Strategie bestimmen. Sehr interessant ist auch die mittlere Spieldauer.

Die Anzahl der Spielrunden ist gleich dem Parameter *i* nach der **for**...**end**-Schleife in dem Programm oben. Wenn Du also eine weitere **for**-Schleife programmierst, kannst Du den Mittelwert der Anzahl der Spielrunden einfach berechnen.

Theorie

Im Folgenden sollen Deine numerischen Untersuchungen mit einem theoretischen Teil unterfüttert werden. Dazu sei $Q(z)$ die Wahrscheinlichkeit dafür, dass Spieler A sein Vermögen z verliert. $P(z)$ bezeichnet die Wahrscheinlichkeit dafür, dass Spieler A - beginnend mit dem Vermögen z - Spieler B ruiniert.

Das Vermögen von Spieler A wächst im ersten Schritt mit der Wahrscheinlichkeit p von z auf $z+1$. Mit der Wahrscheinlichkeit q fällt das Vermögen von z auf $z-1$. Daher gilt für die gesamte Verlustwahrscheinlichkeit von Spieler A

$$Q(z) = pQ(z+1) + qQ(z-1). \tag{1}$$

Natürlich hat der Spieler mit einem Vermögen 0 verloren und mit einem Vermögen a gewonnen. Daher gelten die sogenannten Randbedingungen

$$Q(0) = 1, \quad Q(a) = 0. \tag{2}$$

Das Ziel der folgenden Überlegungen ist es, Gleichung (1) zu lösen. Dazu sei - wie beim Roulette - zunächst $p \neq q$. Dann hat Gleichung (1) zwei verschiedene fundamentale Lösungen,

$$a)\ Q(z) = 1,$$

$$b)\ Q(z) = \left(\frac{q}{p}\right)^z.$$

Übung:
Zeige durch Einsetzen, dass diese beiden Gleichungen wirklich Lösungen der Gleichung (1) darstellen.
Übung:
Zeige, dass Vielfache von a) und b) sowie Summen von solchen Vielfachen dann auch Lösungen von Gleichung (1) sind.

Die allgemeine Lösung von Gleichung (1) lautet

$$Q(z) = A + B\left(\frac{q}{p}\right)^z, \tag{3}$$

wobei die Konstanten A und B nun so zu bestimmen sind, dass die Randbedingungen (2) erfüllt sind.

Übung:
Bestimme die Konstanten A und B der allgemeinen Lösung (3) so, dass die Randbedingungen (2) erfüllt sind. (Erst selber versuchen, dann weiterlesen!)

Lösung:
Da $Q(0)=1$ ist, folgt aus (3)

$$A+B=1.$$

Da andererseits $Q(a)=0$ ist, folgt aus (3) auch

$$A+B\left(\frac{q}{p}\right)^a=0.$$

Durch Bilden der Differenz bekommst Du nun

$$B\left(1-\left(\frac{q}{p}\right)^a\right)=1, \quad \Rightarrow \quad B=\frac{1}{1-\left(\frac{q}{p}\right)^a}.$$

Die Berechnung von A ist dann einfach:

$$A=1-B=\frac{-\left(\frac{q}{p}\right)^a}{1-\left(\frac{q}{p}\right)^a}.$$

Die gesuchte Lösung für die Ruinwahrscheinlichkeit von Spieler A ist damit gefunden, denn man muss die Konstanten A und B nur noch in die Lösung (3) einsetzen,

$$Q(z)=\frac{\left(\frac{q}{p}\right)^a-\left(\frac{q}{p}\right)^z}{\left(\frac{q}{p}\right)^a-1}. \qquad (4)$$

Dies ist ein simples aber wirksames Ergebnis, das es Dir nun erlaubt, die Verlustwahrscheinlichkeit beim Roulette zu berechnen.

Beispiel:
Angenommen, ein Casinospieler setzt wie oben beschrieben immer den gleichen Einsatz von 1€ auf *Rouge*. Er startet mit 200€. Sein Ziel ist bescheiden, denn er strebt nur eine Rendite von 10% an, stoppt also, wenn er 220€ Vermögen hat. Also gilt $z=200,$ und $a=220.$ Außerdem ist $p=\frac{18}{37}$, $q=\frac{19}{37}$, und daher $\frac{q}{p}=\frac{19}{18}\approx 1{,}05556.$ Die Verlustwahrscheinichkeit ist daher

$$Q(z)=\frac{\left(\frac{19}{18}\right)^{220}-\left(\frac{19}{18}\right)^{200}}{\left(\frac{19}{18}\right)^{220}-1}=0{,}661.$$

Mit einer Wahrscheinlichkeit von 66,1%, also in 2 von 3 Casinobesuchen, verzockt dieser Spieler also mit der angegebenen Strategie sein ganzes Vermögen von 200€ und nur in 1 von 3 Fällen nimmt er den bescheidenen Gewinn von 20€ mit nach Hause - nicht gut! Eine Verdoppelung von 200€ auf 400€ ist mit der beschriebenen Strategie so gut wie aussichtslos.

Der Spieler ändert daher seine Strategie. Nun setzt er jedes Mal 10€ auf *Rouge*, während sein Ziel und sein Vermögen gleich bleiben. Mathematisch entspricht der neue Einsatz einer Skalierung des Vermögens um den Faktor 10. Die Ruinwahrscheinlichkeit des Spielers ist dann

$$Q(z) = \frac{\left(\frac{19}{18}\right)^{22} - \left(\frac{19}{18}\right)^{20}}{\left(\frac{19}{18}\right)^{22} - 1} = 0{,}147.$$

Nun verliert der Spieler nur noch in 1 von 7 Casinobesuchen sein Vermögen und kommt in 6 von 7 Besuchen mit einem Gewinn von 20€ nach Hause!

Dennoch: Besonders gut ist das auch nicht. Der erwartete Gewinn ist nämlich gleich

$$\mathrm{EW}_{\mathrm{Gewinn}} = a \cdot P(z) - z = a \cdot \big(1 - Q(z)\big) - z.$$

Im ersten Fall ist das gleich -125,42€ pro Casinobesuch, während mit der zweiten Strategie -12,34€ Gewinn pro Casinobesuch erwartet werden können. Natürlich gewinnt die Bank auf lange Sicht immer!

Übung:
Überlege weitere Anwendungsbeispiele der erarbeiteten Formeln. Berechne weitere Beispiele.

Nun muss noch die Gleichung (1) für den Fall $p = q = \frac{1}{2}$ gelöst werden. In diesem Fall heißt das Spiel *fair*. Dann Fall wird aus Gleichung (1)

$$Q(z) = \frac{1}{2}\big(Q(z+1) + Q(z-1)\big),$$

mit den fundamentalen Lösungen

$$a)\ Q(z) = 1,$$
$$b)\ Q(z) = z.$$

Übung:
Zeige, dass a) und b) Lösungen der obigen Gleichung sind.

Die allgemeine Lösung für die Verlustwahrscheinlichkeit ist daher

$$Q(z) = A + Bz.$$

Übung:

Bestimme aus den Randbedingungen (2) die Konstanten A und B und setze sie in $Q(z)$ ein.

Lösung:
Aus $Q(0)=1$ folgt

$$A=1,$$

und aus $Q(a)=0$ folgt

$$1+Ba=0, \quad \Rightarrow \quad B=-\frac{1}{a}.$$

Die Verlustwahrscheinlichkeit im fairen Fall ist daher

$$Q(z)=1-\frac{z}{a}.$$

Hinweis:
Die Überlegungen zum Ruin eines Spielers folgen im wesentlichen dem Klassiker *An Introduction to Probability Theory and its Applications*, Volume I, John Wiley & Sons von William Feller.

Noch mehr Theorie
Eine unbeschränkte Zufallsbewegung mit gleich langen Schritten nach links (Wahrscheinlichkeit q) bzw. rechts (Wahrscheinlichkeit p) ist mathematisch auch hochinteressant und in vielen Details alles andere als einfach.

Zunächst ist aber klar, dass im Mittel von n Schritten $n\cdot p$ nach rechts gehen und $n\cdot q$ nach links. Das heißt, dass eine solche zufällige Bewegung im Mittel in n Schritten die Entfernung $n\cdot(p-q)$ zurücklegt, wenn $p\neq q$ ist. In diesem Fall ist die durchschnittlich zurückgelegte Entfernung also proportional zur Anzahl der Schritte.

Für den Fall $p=q=\frac{1}{2}$ ist keine der beiden Richtungen bevorzugt. Daher ist natürlich der Mittelwert der Endkoordinate gleich 0, wenn die Bewegung im Ursprung startet. Trotzdem wird absolut (also betragsmäßig) betrachtet eine bestimmte Entfernung zurückgelegt. Diese kann man einfach berechnen. Ein Schritt der Zufallsbewegung entspricht dem Münzwurf mit einer fairen Münze. Das zufällige Ergebnis (1 für Kopf, -1 für Zahl) wird mit X_1 für die erste Münze, X_2 für die zweite Münze usw. bezeichnet. Der Erwartungswert einer Zufallsgröße X wird mit $\langle X\rangle$ bezeichnet. Natürlich ist der Erwartungswert $\langle X_1\rangle=0$, da Kopf und Zahl gleichberechtigt fallen. Andererseits ist aber $\langle X_1^2\rangle=1$, da das quadrierte Ergebnis des Münzwurfs immer positiv ist.

Für den Pfad zum Endpunkt der Zufallsbewegung nach n Schritten gilt

$$\text{Pfad Endpunkt} = X_1 + X_2 + \ldots + X_n .$$

Der Erwartungswert des Quadrates der Pfadlänge ist daher

$$\left\langle \left(X_1 + X_2 + \ldots + X_n \right)^2 \right\rangle .$$

Wenn man das Quadrat der Klammer berechnet, erhält man eine Summe mit den n Quadraten der Zufallsvariablen X_i sowie allen möglichen Produkten zwischen verschiedenen Zufallsvariablen $X_i \cdot X_j$, mit $i \neq j$. Diese Produkte haben den Wert +1 für das Wurfergebnis Kopf-Kopf und Zahl-Zahl und den Wert -1 für die Wurfergebnisse Kopf-Zahl und Zahl-Kopf. Der Erwartungswert eines jeden dieser gemischten Produkte ist daher gleich 0. Also gilt

$$\begin{aligned} \left\langle \left(X_1 + X_2 + \ldots + X_n \right)^2 \right\rangle &= \left\langle X_1^2 + X_2^2 + \ldots + X_n^2 \right\rangle \\ &= 1 + 1 + \ldots + 1 = n. \end{aligned}$$

Anders ausgedrückt heißt das, dass bei einem symmetrischen eindimensionalen Random Walk die Wurzel aus dem Erwartungswert des Quadrates der zurückgelegten Entfernung wie $\sqrt{n}$ mit der Anzahl der Schritte n wächst. In 2 Dimensionen hast Du oben ein ähnliches Statement numerisch belegt. Die zurückgelegte Entfernung wächst damit für große Werte von n deutlich langsamer als im Fall $p \neq q$.

Bonbon

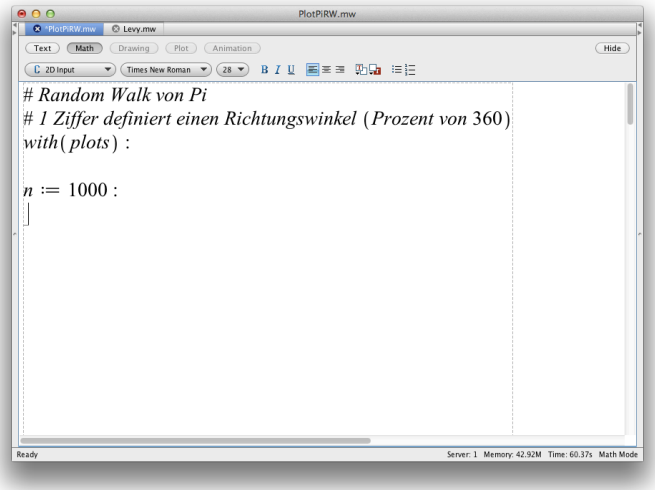
Die Kreiszahl $\pi = 3{,}14149265\ldots$ fasziniert seit jeher alle mathematisch Interessierten, und daher können wir nicht anders, als ihrem Charme auch zu erliegen. Klar ist, dass π irrational ist, also nicht als Bruch dargestellt werden kann, ebenso wurde bewiesen, dass π transzendent ist, also nicht einmal die Nullstelle eines Polynoms mit ganzzahligen Koeffizienten sein kann. Aber ist π auch normal?

„Als **normale Zahl** wird in der Mathematik eine reelle Zahl bezeichnet, unter deren Nachkommaziffern für jedes $k \geq 0$ alle möglichen *k*-stelligen Ziffernblöcke mit gleichen asymptotischen relativen Häufigkeiten auftreten.“ (Wikipedia).

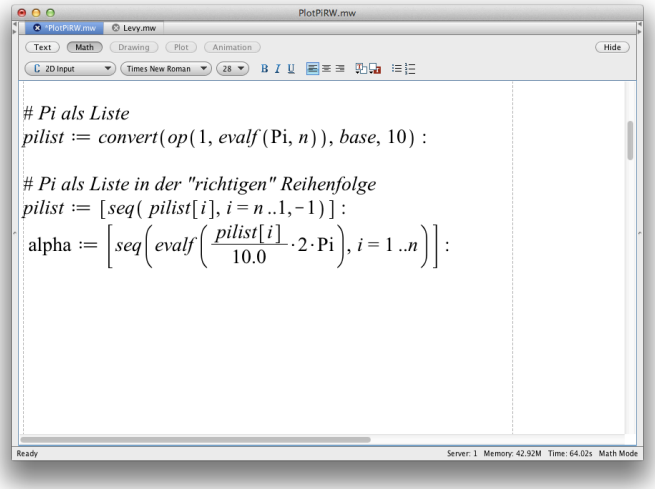
Eine normale Zahl enthält also jede Ziffernkombination am Stück, und sei sie auch noch so lang. Man könnte also die Bibel in Ziffern kodieren und erwarten, dass der gesamte Text in der Zahl vorkommt. Bei unendlichen *zufälligen* Ziffernfolgen würde man dies ebenfalls erwarten.

Allerdings weiß man nicht, ob π wirklich eine normale Zahl ist. Viele Forscher sind aber der Meinung, dass dies der Fall ist. 2005 haben Physiker die ersten 100 Millionen Ziffern von π untersucht und genau die gleichen Eigenschaften wie bei statistischen Zufallszahlen entdeckt.

Daher ist es reizvoll, die Ziffern der Zahl π auch zur Erzeugung von zufälligen Ereignissen nutzen und zur Programmierung eines Random Walks verwenden.

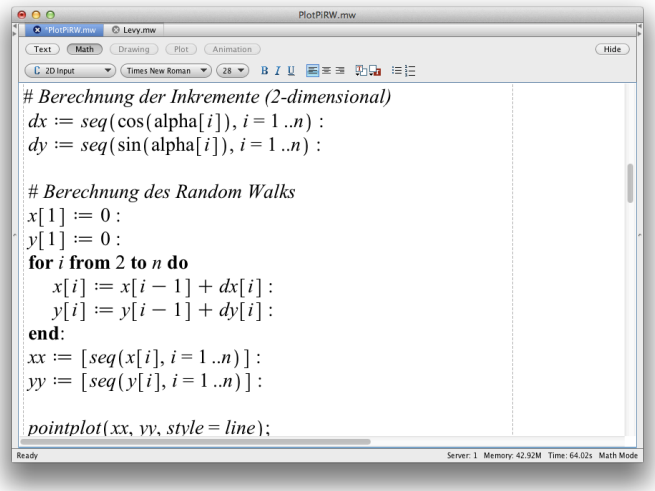


In dem Random Walk der Zahl π sollen allen Schritte die gleiche Länge 1 haben. Jede Ziffer kodiert eine von 10 möglichen Richtungen. Zunächst wird das Modul plots geladen und die Anzahl der Schritte definiert.



Interessant ist es, wie man in Maple auf die Ziffern der Zahl π zugreift.

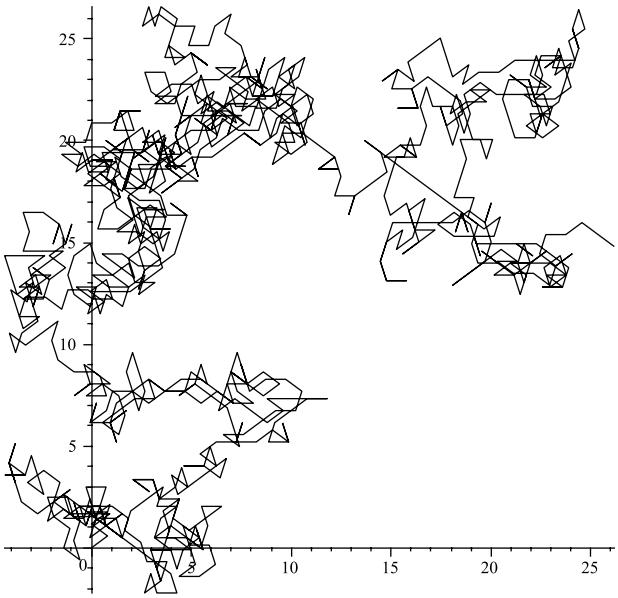
Dazu macht man sich die spezielle Art, wie Maple reelle Zahlen abspeichert, zunutze. *op(1,Zahl)* gibt eine ganze Zahl mit der gleichen Ziffernfolge zurück. Diese kann dann mit *convert(...)* in die Folge der einzelnen Ziffern zerlegt werden. Allerdings muss man dann noch die Reihenfolge umkehren. Anschließend werden dann schon die Richtungswinkel in Bogenmaß berechnet.



Der Rest ist wie beim Levy-Flug. Für die jeweiligen Winkel werden die richtigen Schritte berechnet und Schritt für Schritt zusammen addiert.

Mit *pointplot* wird der Random Walk dann dargestellt.

Das Ergebnis eines Random Walks der Zahl π für die ersten 1000 Ziffern ist in der folgenden Abbildung dargestellt.



Übung:
Natürlich gibt es viele weitere Methoden, einen Random Walk der Zahl π zu programmieren. Zum Beispiel könnte ein Winkel aus jeweils 2 Ziffern berechnet werden. Oder die Schrittlänge könnte variieren. Spiele mit dem Programm und wandele es nach einer ab, so dass ein anderer Random Walk der Zahl π entsteht.

Für weitere Anregungen kannst Du auch die Internetseite http://www.random-walk.com/ besuchen. Dort hat ein Designstudent dieses Projekt grafisch aufgearbeitet.