Chapter 9

確率実験とランダムウォーク

コインを投げたりサイコロを振って確率の実験をするかわりに，*Mathematica* に実験をさせてみよう．また後半では，現代の数学や物理学において重要な研究対象である「ランダムウォーク」や，複雑な数値積分に用いられる「モンテカルロ法」を紹介する．

数値計算やグラフの描画も面白いが，「実験」，いわゆるシミュレーションも *Mathematica* の得意とするところである．たとえば，こんな問題を考えてみよう：

ゲームの価格設定問題．サイコロを 5 回投げて，

$$\{(\text{出た目の最大値}) + (\text{出た目の最小値})\} \times 100 \text{ 円}$$

をもらえる有料のゲームを作りたい．このゲームの損益分岐価格（儲けも損もしない価格設定）を決定せよ．ただしゲームの運営コストは考えず，お客は無限に来るものとする．

ひとつの方法は，高校で学んだ「期待値」の知識を用いて，大学入試さながらに解くことだろう．もうひとつは，実際にサイコロを何千回も振ってゲームをシミュレートし，価格を経験的に割り出すことである．

ちなみに筆者は数学を専門とする大学生たちに上の問題を解いてもらったことがあるが，正しい答えを予想することすら難しい，という状況であった．というわけで，前者の方法はあまりオススメではない．

一方，後者のシミュレーションは退屈な単純作業に見えるかもしれない．しかし *Mathematica* に仮想的なサイコロを設定してやりさえすれば，一瞬のうちに結果が出るのである．

こうしたコンピューターによる統計的シミュレーション（「確率実験」）が生命保険などの価格設定にも有効であることは，すぐに想像がつくと思う．

問題 9.1 (株価っぽいグラフ) 株価 (stock) っぽいグラフを描こう．

(1) `x := RandomReal[{-1, 1}]` とする．`dif = Table[x, {100}]` に対し，`stock = Accumulate[dif]` はどんなリストを与えるのか説明せよ．

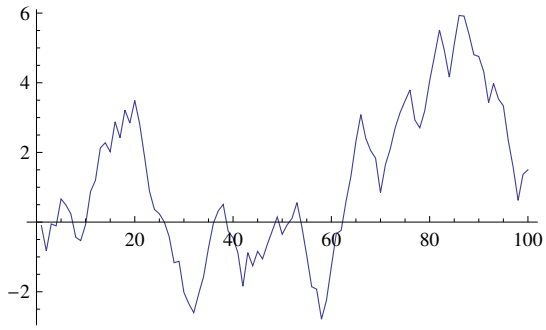
(2) `ListLinePlot[stock]` としてグラフを描け．

【解答】 (1) `x := RandomReal[{-1, 1}]` は区間 $[-1, 1]$ 内の実数値をとる乱数，それを 100 個並べたリストが `dif = Table[x, {100}]` である．さらに，そのリストを前から足し上げて得られたリストが `stock = Accumulate[dif]` となる．

(2) 適宜入力すれば，次のようになる（リストの出力は ; で省略した．）

```
In[ ]:= x := RandomReal[{-1, 1}]; dif = Table[x, {100}];
        stock = Accumulate[dif]; ListLinePlot[stock]
```

Out[ ]=



■

9.3　確率実験（コイン投げ）

コインを 10 回投げて (toss)，表が出た回数と同じ数だけリンゴ（apples）がもらえるとする．このゲームをシミュレーションしてみよう．コインの表を 1，裏を 0 とする．

[13] コインを 10 回投げた結果をリストにする変数 `toss` を定義：

```
In[13]:= toss := Table[RandomInteger[], {10}];
```

[14] たとえば：

```
In[14]:= toss
Out[14]= {0, 1, 0, 0, 1, 0, 1, 0, 1, 0}
```

[15] `toss` を実行し，`Total` でもらえるリンゴの数をカウントする：

```
In[15]:= apples := Total[toss];
```

`Total` はリストの要素の総和を与える組込み関数である．

[16] たとえば：

```
In[16]:= apples

Out[16]= 7
```

この値は [14] の toss の結果と合わないことに注意しよう．遅延的定義（:=）によって再度 toss 部分が計算されたからである．

[17] ゲームを 100 人にやらせる．すなわち，apples を 100 回実行しリストにする：

```
In[17]:= n = 100; trials = Table[apples, {n}]

Out[17]= {6, 7, 5, 4, 5, 8, 6, 4, 6, 6, 7, 6, 4, 7, 6, 5,
  5, 6, 6, 5, 2, 6, 4, 6, 6, 6, 6, 5, 9, 4, 4, 6,
  4, 5, 3, 6, 3, 7, 5, 3, 3, 5, 4, 2, 7, 4, 3, 5, 6,
  5, 4, 4, 5, 6, 7, 5, 7, 4, 6, 2, 5, 5, 5, 6, 7, 3,
  4, 3, 2, 2, 3, 4, 3, 4, 4, 3, 5, 5, 3, 5, 5, 1, 7,
  4, 4, 4, 6, 4, 6, 7, 5, 6, 2, 5, 7, 6, 6, 2, 3, 4}
```

今度は即時的定義（=）を用いたので，trials の値は以後固定される．

[18] trials 中のリンゴを k 個もらった人をカウントしたリスト num を作る：

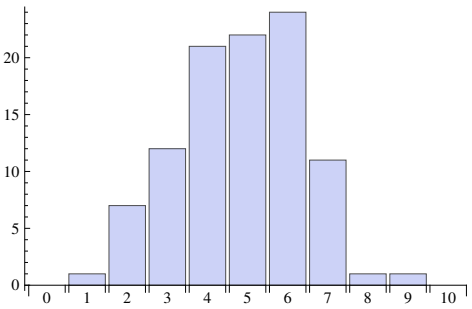
```
In[18]:= num = Table[Count[trials, k], {k, 0, 10} ]

Out[18]= {0, 1, 7, 12, 21, 22, 24, 11, 1, 1, 0}
```

Count は組込み関数で，Count[trials, k] でリスト trial の要素に k という値がいくつ含まれるかを数えてくれる．

[19] BarChart で結果を**棒グラフ**にする：

```
In[19]:= BarChart[num, ChartLabels -> Range[0, 10] ]
```

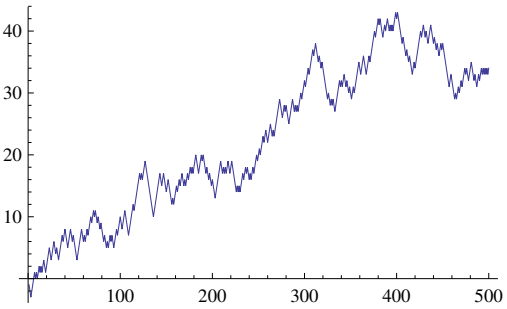
Out[19]=



オプション ChartLabels -> Range[0, 10] はグラフの横軸に 0 から 10 までの数字を打つためのものである．詳しくはドキュメントセンターを参照してほしい．

では冒頭の価格設定問題を解いてみよう．

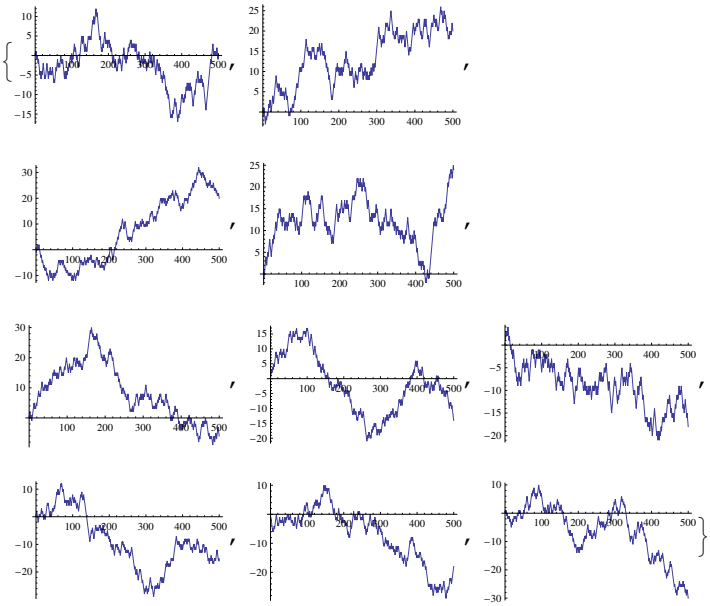
Out[26]=



[27] r[500] を 10 回実行しリストにする：

In[27]:=
```
tab = Table[r[500], {10}]
```

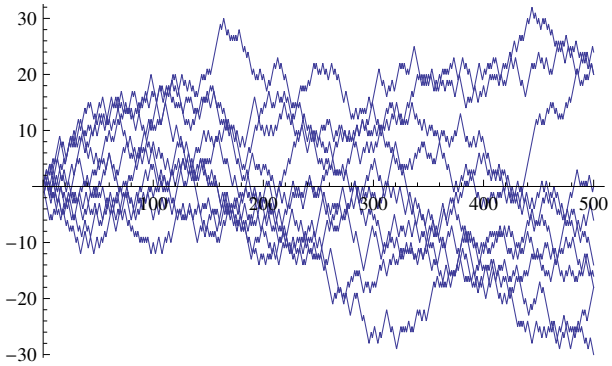
Out[27]=



[28] Show でグラフを重ねる：

In[28]:=
```
Show[tab, PlotRange -> All]
```

Out[28]=



問題 9.3 (グラフの色分け) 重ねたグラフはとても見づらいので，10 回の試行それぞれが別の色のグラフになるようにせよ．

【解答】　ドキュメントセンターを見ると，ListLinePlot の線の色を変えるオプションは PlotStyle だとわかる．たとえば PlotStyle -> Red のように書き加えればグラフの線を赤

くできるが，10 個のグラフそれぞれに色を指示するのは面倒なので，Hue という組込み関数を用いる．Hue は 0 から 1 の間の実数を図 9.2 のような色に対応付ける．

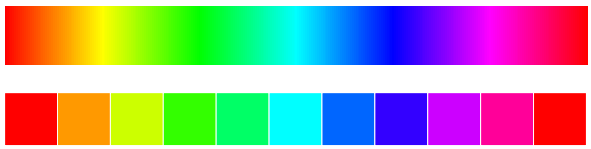


図 9.2 上は $[0,1]$ 区間内の実数 t に対し Hue[t] が指定する色．下は $0.0, 0.1, 0.2, \ldots, 0.9, 1.0$ に対応する色．実数 t が $[0,1]$ 区間にない場合はその小数部分に対応する色が選択される．

さて [25] に手を加えて，

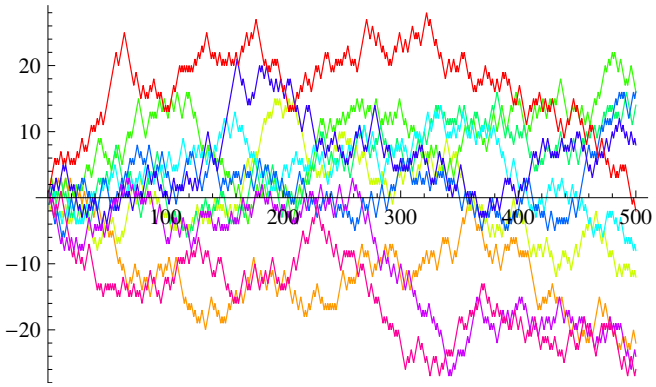
```
In[ ]:=  r[n_, c_] := ListLinePlot[q[n], PlotStyle -> Hue[c/10]];
```

とする．c が 1 から 10 まで変化すれば c/10 は $0.1, 0.2, \ldots, 0.9, 1.0$ と変化し，図 9.2 下段の色でグラフが着色される，というわけであえる．さらに [27] を，

```
In[ ]:=  tab = Table[r[500, c], {c, 1, 10}]
```

に変えて [28] を実行すれば，次のような出力が得られる:

Out[ ]=



■

9.5 2 次元ランダムウォーク

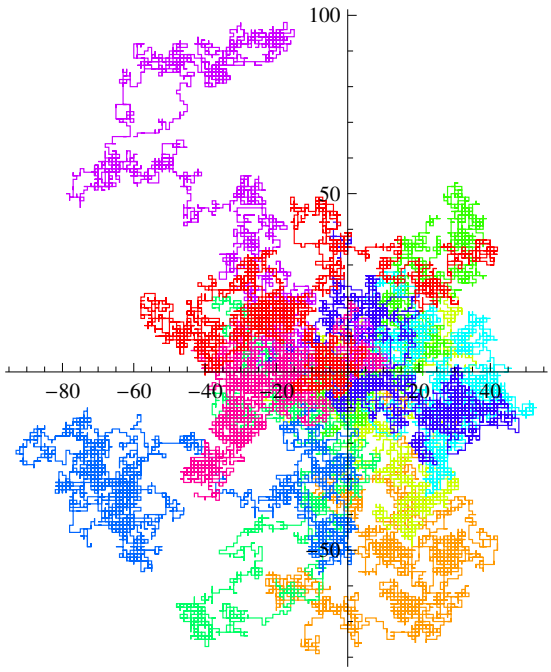
今度は平面上のランダムウォークをシミュレートしてみよう．図 9.3 のように 2 次元平面の原点からスタートし，「上下左右の方向に，それぞれ確率 1/4 で 1 移動」という操作を繰り返す．たとえば，図中の青い矢印たちは { 右，上，上，左，下，上，左，下，左 } に対応する経路を示している．この操作を *Mathematica* で自動化して，経路を観察してみよう．

[29] 上下左右の方向 (direction) に対応するベクトルのリスト dir を作る：

[38] tab で作ったグラフを重ねて（同一の座標に）表示：

```
In[38]:= Show[tab, PlotRange -> All]
```

Out[38]=



問題 9.4 (偏りのあるランダムウォーク) [29] の dir をうまく修正し，上下に進む確率がそれぞれ 1/3 ，左右に進む確率がそれぞれ 1/6 となるようにせよ．その上で [37][38] を実行せよ．

【解答】 上下方向に進む確率が左右のそれの 2 倍になるのだから，dir に上下の要素を追加して

```
In[ ]:= dir = {{0, 1}, {0, 1}, {0, -1}, {0, -1}, {1, 0}, {-1, 0}};
```

とすればよい．あとは同じ関数が再利用できる．（結果は略.）　■

9.6　研究：モンテカルロ法

乱数を用いて円周率の近似値を求めることができる：まず，xy 平面上に単位円板を含む正方形 $\{|x|, |y| \leq 1\}$ をとる．この正方形内にランダムに点をうっていくと，単位円板の中に入っている点の割合は（円板の面積）÷（正方形の面積）$= \pi/4$ に近づくであろう．円板に入っているかどうかは原点との距離の 2 乗で判定できる．また，第一象限だけで実験すればよい．

このような手法は**モンテカルロ法** (Monte Carlo method) とよばれる．

[39] 区間 $[0, 1]$ に値をとる乱数：

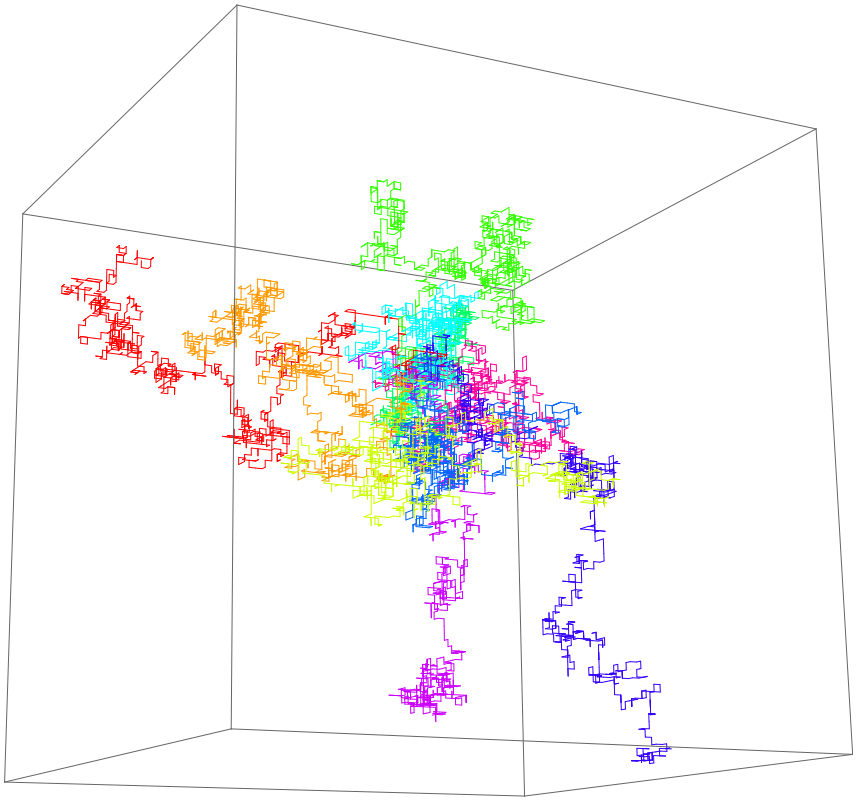


図 9.4 3 次元ランダムウォーク

9.8 研究：2 次元ランダムウォークいろいろ

2 次元ランダムウォークは，まず平面に正方形のタイルを敷き詰め，その頂点の間をランダムに移動する点の動きを表現したものだといえる．

同様に正 6 角形や正 3 角形も平面を埋め尽くすことができるから，その頂点の間でランダムウォークを考えることができる．たとえば次の問題を考えてみよう：

問題 9.5 (六方格子上のランダムウォーク) 正 6 角形のタイルで埋め尽くされた平面を**六方格子** (hexagonal lattice) とよぶ．[29] の `dir` と [31] の `p[n_]` をうまく修正し，六方格子上のランダムウォークを作成せよ．その上で [37][38] を実行せよ．(Hint. `dir` 進んだら次は `-dir` 進む．)

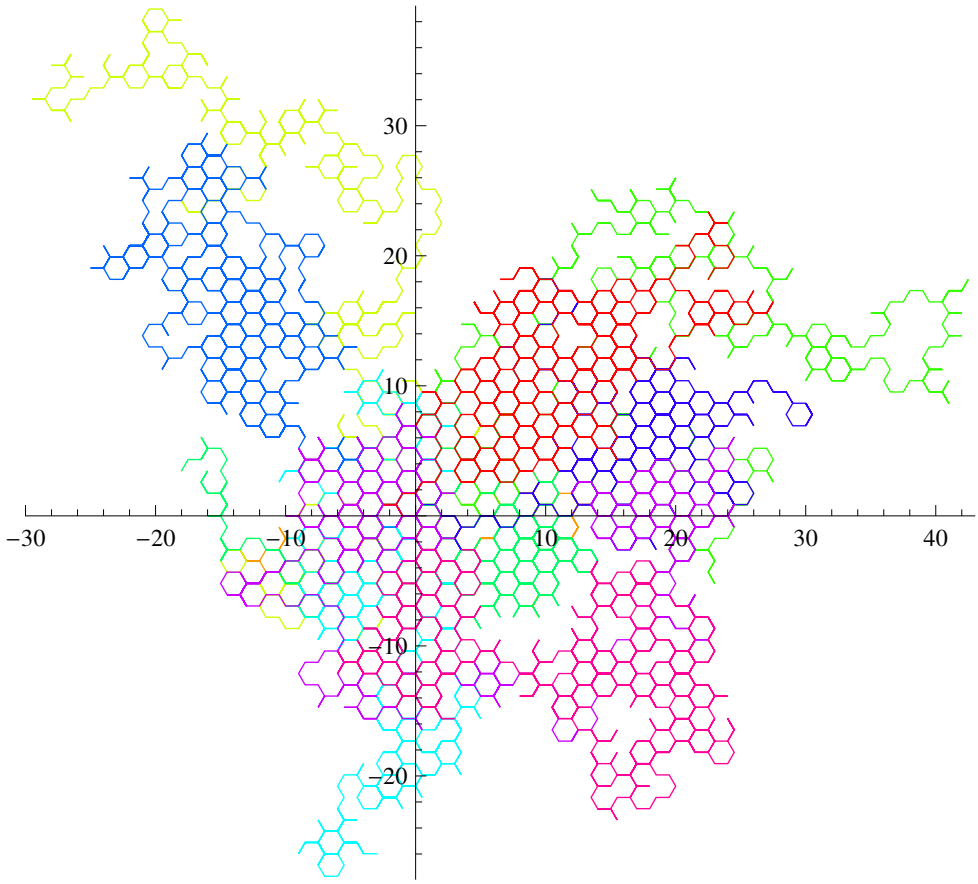
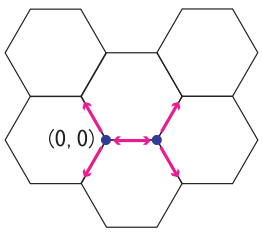


図 9.5　六方格子上のランダムウォーク.

【解答】　まず dir を次の 3 方向に変える.

```
In[ ]:=  dir = {{1, 0}, {-1/2, Sqrt[3]/2}, {-1/2, -Sqrt[3]/2}};
```

さらにステップ数の偶奇に応じて方向を 180 度回転させるよう p[n_] を修正する：




```
In[ ]:=  x := RandomChoice[dir];
         p[n_] := Table[(-1)^k * x, {k, 1, n}];
```

あとは同じ関数を再利用すればよい. ■

平面が正 3 角形で埋め尽くされた場合のランダムウォークはもっと簡単なので，ぜひ独力で考えてみていただきたい.