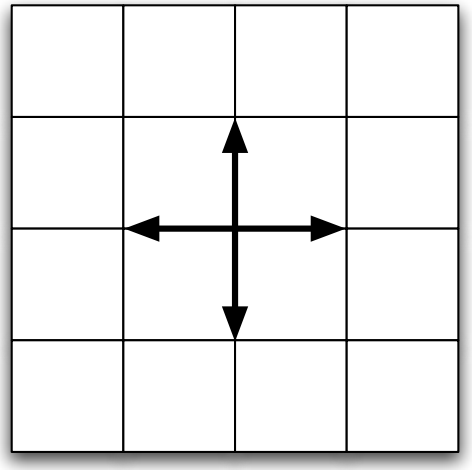
モンテカルロ法

Monte Carlo method

- 乱数を用いてシミュレーションや数値計算を行う手法の総称
- 物理学や生物学などのシミュレーションに良く用いられる
- 具体例：コイン投げ、ランダムウォーク（乱歩）、出生死亡過程など、確率論的な事象の変化をアルゴリズムとして記述して実行する

格子上のランダムウォーク

- 2 次元格子空間を考える。各個体は格子上に存在し、単位時間内に隣接する4つの格子のいずれかへ等しい確率 1/4 で移動する。
- 初期分布として原点に N 個体存在する状態を考える。時刻 t での個体の空間分布はどのようなものか？ (t = 0, 1, 2, 3, ...)
- また、時刻 t と原点から最も離れた個体の距離との関係は？



格子上のランダムウォーク

- *N* 個体の位置を 2次元配列で表現するランダムウォークに取り組む
- 初期状態として全ての個体は原点に位置するとする
- 以下を繰り返す（時刻のループ）
  - 次のアルゴリズムに従って、*N* 個体の座標を変化させる

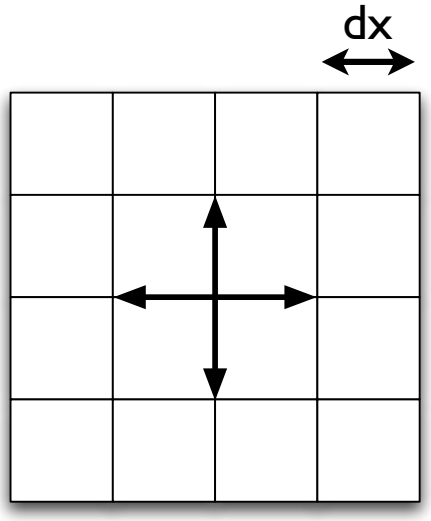
各個体の x, y 座標は

確率 1/4 で x += dx

確率 1/4 で x -= dx

確率 1/4 で y += dx

確率 1/4 で y -=dx



プログラム骨格

```
#define N 100
#define DX 1.0

double x[N], y[N]; // 個体の座標


int main(void)
{
    initialize();    // 初期化：すべての個体は原点にある
    process();    // シミュレーション実行
    finish();       // 後始末
}

void process()
{
    for(int step=0; step<STEP; step++){
        move_indivs();   // 個体を移動させる関数
        write_data();     // 個体の位置をファイルに書き出す
    }
}
```

プログラム骨格

```
void move_indivs()
{
  int i;
  double rv;

  for(i=0; i<N; i++){

    rv = genrand_real2();

    if( rv < 0.25 )    // rv の値に応じてx, y座標を更新
      x[i] += DX;
    else if( rv < 0.50 )
      x[i] -= DX;
    else if( rv < 0.75 )
      y[i] += DX;
    else
      y[i] -= DX;
  }
}

void write_data()
{
  int i;

  for(i=0; i<N; i++)
    fprintf(fp, "%f %f ", x[i], y[i]);

  fprintf(fp, “\n");
}
```

計算結果の表示

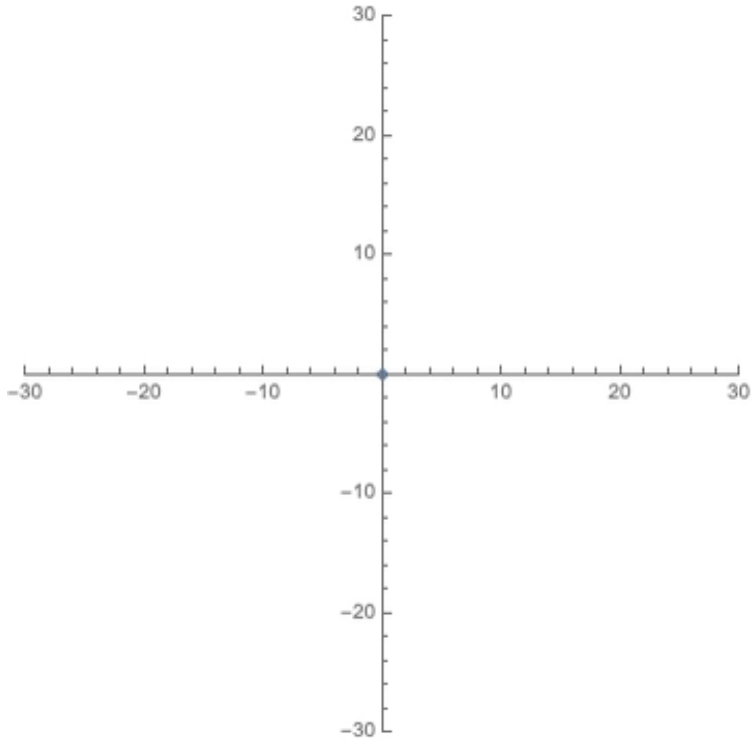
python で可視化する

```
# ランダムウォークモデルの可視化

import numpy as np
import matplotlib.pyplot as plt
import matplotlib.animation as animation

# ファイルからデータを読み込む

data = np.loadtxt('data_rw')                    <— データファイルを指定
print('Data length is ', len(data))

rangeMax = 3
rangeMin = -rangeMax                            <— y座標の範囲を設定

fig, ax = plt.subplots()

ims = []
for t in range(len(data)):
   seqx = []
   seqy = []
   for i in range(0, len(data[t]), 2):
      seqx.append(data[t][i])
      seqy.append(data[t][i+1])

   plt.xlim(rangeMin, rangeMax)
   plt.ylim(rangeMin, rangeMax)
   plt.gca().set_aspect('equal', adjustable='box')
   im = plt.scatter(seqx, seqy, color='black', s = 10)     <— 個体を描画
   ims.append([im])

anim = animation.ArtistAnimation(fig, ims, interval=100)  <— アニメーション描画
plt.show()
```

2次元格子上の乱歩

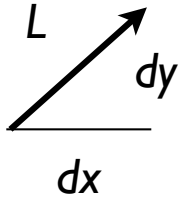
```
data = ReadList[ "data", Real, RecordLists -> True ];
data2 = Map[ Partition[#, 2] &, data];
Length[data2]
{ Max[data2], Min[data2] }

Do[
  ListPlot[ data2[[i]], PlotRange -> {{-40, 40}, {-40, 40}}, AspectRatio -> 1 ]//Print, {i, 1,
Length[data2]}
]
```

Mathematica を用いた可視化



問題 7

- 以下のルールに従い N 個体が移動する様をシミュレートする
- 各個体は格子空間ではなく 2次元連続空間上にあるとする（x, y 座標は実数）
- 離散時間 t = 0, 1, 2, 3 で考える
- 初期状態として全ての個体を原点に配置する
- 各個体は同期して距離 *L* だけ移動し、方角は任意の方向に等しい確率で移動
- t ステップ後に原点から最も離れた個体までの距離はどうなっているか？



角度 θ = U[0, 2π)

$$dx = L\cos\theta$$
$$dy = L\sin\theta$$

問題 8

- 1次元格子空間上のランダムウォークを考える（格子サイズは 1、左右に同じ確率 1/2 で移動）
- 原点から出発する1個体が最初に目的地 L > 0 に到達するまでの時間の分布を L = 1, 2, 3, 4, 5 などについて求めよ
- 2次元格子空間上で同様の解析を行ってみよ
  - 目的地は点 (L, 0) や領域 (x 座標 > L など) とする