♣♢♠♡♣♢♠♡♣♢♠♡♣♢♠♡♣♢♠♡♣♢♠♡♣♢♠♡♣♢♠♡♣♢♠♡♣♢♠♡

確率モデル及び演習

♣♢♠♡♣♢♠♡♣♢♠♡♣♢♠♡♣♢♠♡♣♢♠♡♣♢♠♡♣♢♠♡♣♢♠♡♣♢♠♡

先端理工学部・３年次以降配当・４Q・２単位[1]

阪井 一繁（先端理工学部　数理・情報科学課程）

sakai@math.ryukoku.ac.jp

成績評価

小テスト（12/23, 1/27 実施予定（５時間目開始時 20 分程度））40 点
＋
演習 60 点

0　確率モデル？

次の方程式は，太陽の引力を受けて運動するある惑星の運動方程式（記号の説明は省略）で，惑星の運行という現象の**数理モデル**です。

$$m\,\frac{d^2\boldsymbol{r}}{dt^2} = -\frac{GmM}{|\boldsymbol{r}|^3}\,\boldsymbol{r}$$

実は，この惑星は他の惑星からの引力も受けているので，方程式はもっと複雑で，場合によっては「カオス」という予測不可能な事態が起こることが知られています。しかし，方程式の中には不確定な部分や**ランダム**な要素は含まれていません。このように不確定な要素を含まない数理モデルを「**決定論的**な（数理）モデル」と言います。これの対義語が「**確率**（的な）モデル」です。いくつかの例で確率モデルの「感じ」をつかみましょう。

ブラウン運動　1827 年，イギリスのブラウン (R.Brown) という人が，水に浸かった花粉が破裂して出てきた粒子がずっとピョコピョコと動いていることを発見しました。原子や分子の存在は知られていない時代なので，ブラウンは生命現象かも，と考えたそうです。現在では，粒子をとりまく水分子が不規則に衝突することでこのピョコピョコ（ブラウン運動）が起こることがわかっています。仕組みがわかっているなら，惑星の運動のように粒子の運動方程式が書けそうですが，水分子が多すぎるうえに不規則に衝突するので，それはちょっと無理です。そこで確率モデルの登場です。把握不可能な水分子の衝突の**統計的**な傾向を考慮して，粒子の運動を（こちらも統計的に）調べるのです。ちなみに，この講義では，ブラウン運動の簡単なモデルである「**ランダムウォーク**」から始め，それに関連する，物質の**拡散現象**の数理モデルについて解説します。

[1]理工学部生は「応用プログラミング及び実習・後期・４単位」

待ち行列 人気ラーメン店に行列ができています。行列はどのくらいのペースで伸びたり縮んだりするでしょう？正確にはわからないですよね。客がどのタイミングで店に来るか，何を注文してどれくらいの時間で食べ終わるか，などを事前に知るのは無理です。しかし，こんな場合も客の到着や食事時間の統計的な傾向がわかれば，確率モデルを使ってある程度の解析が可能です。ラーメン店に限らず，行列の伸び縮みを調べる理論は実際にあって，まんま「**待ち行列理論**」と言います。

株価や為替レートの変動 これが決定論的にわかれば大金持ちですね。しかし，残念ながらそううまくはいきません。これらの変動の原因をすべて把握することは不可能なので，せいぜいおおまかな傾向がわかる程度です。これも確率モデルの守備範囲です（いろんな理論や方程式が知られていますが，実際にお金儲けをしている人たちはどんなモデルを使っているのだろう …)。

1 疑似乱数の生成・モンテカルロ数値積分

確率モデルを作って何らかの現象を調べるのに，計算機シミュレーションは強力な武器になります。そのとき，計算機の中で「ランダム」を作り出す必要がありますが，そこで使われるのが「**乱数**」です。C 言語の `rand()` 関数や Java の `Math.random()` メソッドなどを使ったことがあるかも知れません。しかし，コンピュータで次々に作られる（生成するとも言います）乱数は，**ランダムに見える数列**に過ぎません。この意味で，コンピュータで作られる乱数は「**疑似**」**乱数**と呼びます。ここでは,（たぶん）最も簡単な疑似乱数の発生方法である**線形合同法**を紹介します。

線形合同法

- 乱数の「種」と呼ばれる整数 x_0 を選ぶ。
- 整数 A, B と自然数 M を決めて（何らかの条件はつきます），漸化式

$$x_{n+1} = (Ax_n + B)\%M \quad (n = 0, 1, 2, \ldots)$$

にしたがって，疑似乱数の列 $x_1, x_2, \ldots$ を作っていく（$\%M$ は M で割った余りを取ることを表します)。

例えば，$A = 15, B = 5, M = 7$ としましょう。$x_0 = 2$ を種とする乱（?）数列は,

$$x_0 = 2 \ \to \ 0 \ \to \ 5 \ \to \ 3 \ \to \ 1 \ \to \ 6 \ \to \ 4 \ \to$$

となります。$M = 7$ なので，現れる整数はもともと 7 種類しかないのですが，それらがなんとなくランダムに現れているように見えなくもないです。しかし，数列のもう一つ先は $2 (= x_0)$ となり，それ以降は周期 7 の繰り返しになってしまうことがわかります。もちろん M にはずっとずっと大きな数が使われて，周期は（A, B を適切に選べば）M まで長くすることができますが，それでも（他の欠

点もあって）形合同法は「実用的には使いものにならない」とされています。この講義で使用するプログラミング言語 Python には，疑似乱数を生成するのに，**メルセンヌ・ツイスター (MT)** と呼ばれる，もっと高度で実用に耐える方法が標準装備されているそうなので，上記のような問題を心配する必要はほぼないのですが，どんな高度な方法も「何らかの漸化式にしたがってランダムに見える数列を作っている」ことは知っておくとよいでしょう。

演習 1.1（5 点） RandomNumber.py は，等しい（と思われる）確率で発生させた 0〜9 の整数値を n 個（個数は実行時に入力）用意し，ヒストグラムを描いて，実際にほぼ等確率であることを確認するプログラムです。空欄を埋めてプログラムを完成し，出力される画像ファイルを提出しなさい。空欄を除いたソースコードは manaba のコースコンテンツにアップしているので，ダウンロードして利用してください（今後も同様)。なお，複数の乱数のリストを一気に生成できるので，numpy.random モジュールの randint() 関数が便利です（使い方はネットなどで調べましょう)。

提出方法 保存される画像ファイルの名前を rd.png から，学籍番号が y200000 なら，rd_y200000.png に変更し，期限内（**次回講義開始時まで**）に manaba のレポートの回答としてアップしてください。

RandomNumber.py

```
import random
import numpy as np
import matplotlib.pyplot as plt

n = int(input("Enter number of steps:"))

x =

plt.hist(x, range = (0,10), rwidth = 0.9, align = 'left')
plt.savefig("rd.png")
plt.show()
```

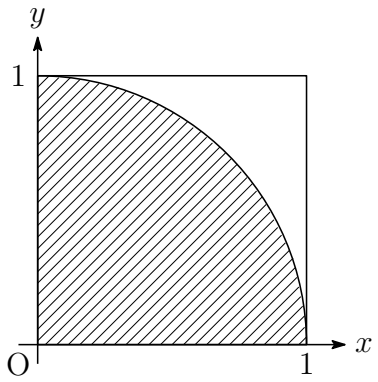
発展 1.1（ボーナス 2 点) 0〜9 の整数値を等確率で次々に生成し，0 の次に生成される整数値の頻度をヒストグラムを描いて調べなさい。線形合同法では，ある数値の後はいつも同じ数値が出ます。

提出方法 担当教員にプログラムと実行結果を (内容を説明しつつ) 見せてください。期限は最終回の終了時とします。

乱数を用いて，何らかの量を求めたり，現象のシミュレーションを行う方法を「**モンテカルロ法**」と呼びます。例えば，モンテカルロ法を使って，以下のように円周率 π の近似値を求めることができます。

モンテカルロ法による円周率の近似計算

- 座標平面の $[0,1) \times [0,1)$ に n 個の点をランダムに（正確に言えば，一様分布するように）とる。
- それらの点のうち，単位円の内部に含まれているものの個数 m を数える。
- 比率 m/n は $\pi/4$ に近いと考えられるので，$4m/n$ を π の近似値とする。



演習 1.2（5 点） MonteCarlo.py は，上記のアルゴリズムで円周率の近似値を求めるプログラムです。加えて，単位円内に入った点は赤色，入らなかった点は青色でプロットし，画像に近似値を表示しています。空欄を埋めてプログラムを完成し，出力される画像ファイルを提出しなさい。

提出方法 保存される画像ファイルの名前を mc.png から，学籍番号が y200000 なら，mc_y200000.png に変更し，期限内（**次回講義開始時まで**）に manaba のレポートの回答としてアップしてください。

発展 1.2（ボーナス 2 点)

後述の「**モンテカルロ数値積分**」の説明を読み，その方法を応用して，π の近似値を求めなさい。

提出方法 担当教員にプログラムと実行結果を (内容を説明しつつ) 見せてください。期限は最終回の終了時とします。

MonteCarlo.py

```
import random
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.patches as patches

plt.figure(figsize = (5,5))
ax = plt.axes()

n = int(input("Enter number of steps:"))

cnt = 0

for i in range(n):
    x = np.random.rand()
    y = np.random.rand()

    if [          ]:

        cnt = cnt + 1
        plt.plot(x, y, marker = ',', color = 'r')
    else:
        plt.plot(x, y, marker = ',', color = 'b')

circle = patches.Arc(xy = (0, 0), width = 2.0, height = 2.0,\
theta1 = 0.0, theta2 = 90.0, color = 'black')
ax.add_patch(circle)
plt.grid()
plt.axis()
plt.text(0.01, 0.01, str(n)+" steps  "+"pi ~ "+str(4*cnt/n))
plt.savefig("mc.png")
plt.show()
```

モンテカルロ数値積分

モンテカルロ法を用いて，定積分 $\int_0^1 f(x)\,dx$ の近似値を計算しましょう（背景にある原理は，円内に入る点の個数を数えて円の面積を求めたのと同じなのですが，見た目は違います)。

- $[0,1]$ で一様分布する確率変数 X を考える。その確率密度関数は,

$$p(x) = \begin{cases} 1 & 0 \leq x \leq 1 \\ 0 & \text{それ以外} \end{cases}$$

となる。

- 確率変数 $f(X)$ の期待値 $E[f(X)]$ は,

$$E[f(x)] = \int_{-\infty}^{\infty} f(x)\, p(x)\, dx = \int_{0}^{1} f(x)\, p(x)\, dx = \int_{0}^{1} f(x)\, dx$$

となる。

- $x_1, x_2, \ldots$ を $[0,1]$ での**一様乱数**の列とする。$f(x_1), f(x_2), \ldots$ の平均について,

$$\frac{f(x_1) + f(x_2) + \cdots + f(x_n)}{n} \longrightarrow E[f(x)] = \int_{0}^{1} f(x)\, dx \quad (n \to \infty)$$

という**大数の法則**が成り立つので，n を大きくとり，$f(x_1), f(x_2), \ldots, f(x_n)$ の平均を計算すれば，それが $\int_0^1 f(x)\, dx$ の近似値となる。