# 確率変数の変換

- 一様乱数 $X = U[0, 1)$ $(0 \leq X < 1)$ から様々な確率分布に従う乱数を生成する
- 指数分布に従う乱数、正規分布に従う乱数、などなど
- 具体例：一様乱数 $X = U[0, 1)$ に対して、
  - $Y = X + 1$ は $U[1, 2)$ に従う（自明）
  - $Y = 2X$ は $U[0, 2)$ に従う（自明）
  - $Y = X^2$ はどんな分布になるか？
  - 任意の変数変換 $Y = f(X)$ により、$Y$ はどのような分布に従うのか？

# U[0, 1) の生成と一様性の確認

## メルセンヌ・ツイスタ Mersenne twister を用いて U[0, 1) を生成する

```
% cc sample-MT-exp.c ran-MT.c        コンパイル・リンク
% ./a.out > data-uniform   実行結果をファイルに書き出す
```

## python を用いてヒストグラム（頻度分布）を描く

```
# ファイルからデータを読み込んで頻度分布を描くコード
import numpy as np
import matplotlib.pyplot as plt

data = np.loadtxt('data-uniform')

print(min(data))  # 最小値を表示
print(max(data))  # 最大値を表示

fig = plt.figure()# 図を書き出すための fig

# 範囲 0, 1、分割数20で頻度分布を描く。面積1に正規化。
plt.hist(data, range=(0, 1), bins=10, density=True, edgecolor='black')

plt.show()

fig.savefig('fig_uniform.eps') # 図を eps 形式で保存
```

# 確率変数の変換

- 一様乱数 X = U[0, 1) を変換 Y = f(X) により変換する
- Y の分布はどのようなものになるか？（確率変数の変数変換）
- X の分布（確率密度関数） p(x) と Y の分布 q(y) の関係

$$\mathrm{Prob}[x < X < x + dx] = p(x)dx \qquad p(x) = 1 \quad 0 \leq x < 1$$

$$\mathrm{Prob}[y < Y < y + dy] = q(y)dy$$

$$p(x)dx = q(y)dy$$

$$q(y) = p(x)\left|\frac{dx}{dy}\right|$$

# 指数分布する乱数

- 一様乱数 X = U(0, 1] を Y = – (log X)/γ で変換して得られる乱数 Y はパラメータ γ の指数分布に従う。log 0 = – ∞ なので 0 は除く

$$Y = -\frac{\log U(0,1]}{\gamma} = -\frac{\log(1 - U[0,1))}{\gamma}$$

$$\mathrm{Prob}[y < Y < y + dy] = q(y)dy$$

$$q(y) = \gamma e^{-\gamma y} \quad y \geq 0$$

$$E[Y] = \int_0^\infty yq(y)dy = \frac{1}{\gamma}$$

# 指数分布する乱数の生成

```c
#include <stdio.h>
#include <time.h>
#include <math.h>

#define GAMMA 2.0     <— 指数分布のパラメータ γ を設定

extern void init_genrand(long);
extern double genrand_real2(void);

int main(void)
{
  long seed;
  int i;
  double rand, rand2;

  seed = (long)time(NULL); // seed の設定

  init_genrand(seed); // seed で初期化

  for(i=0; i<10000; i++){
    rand = 1 - genrand_real2(); <— 1 – U[0, 1) = U(0, 1]
    rand2 = -log(rand)/GAMMA;
    printf("%.20f\n", rand2);
  }

  return 0;
}
```

# 指数分布する乱数の生成

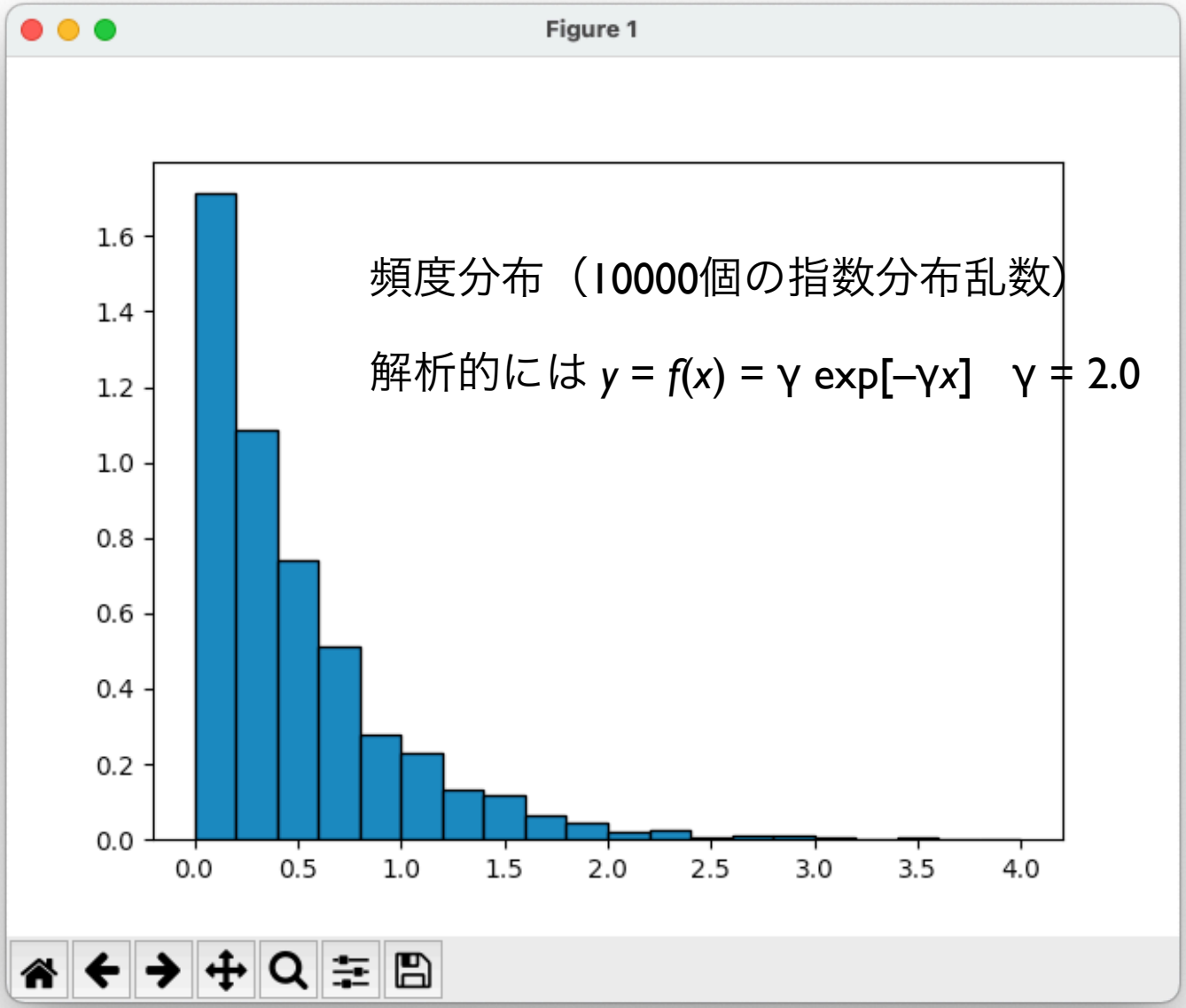
```
% cc sample-MT-exp.c ran-MT.c
```
コンパイル・リンク

```
% ./a.out > data-exp
```
実行結果をファイルに書き出す

## python を用いてヒストグラム（頻度分布）を描く

```
# ファイルからデータを読み込んで頻度分布を描くコード
import numpy as np
import matplotlib.pyplot as plt

data = np.loadtxt('data-exp')

print(min(data))  # 最小値を表示
print(max(data))  # 最大値を表示

fig = plt.figure()# 図を書き出すための fig

# 範囲 0, 1、分割数20で頻度分布を描く。面積1に正規化。
plt.hist(data, range=(0, 4), bins=20, density=True, edgecolor='black')

plt.show()

fig.savefig('fig_exp.eps')  # 図を eps 形式で保存
```

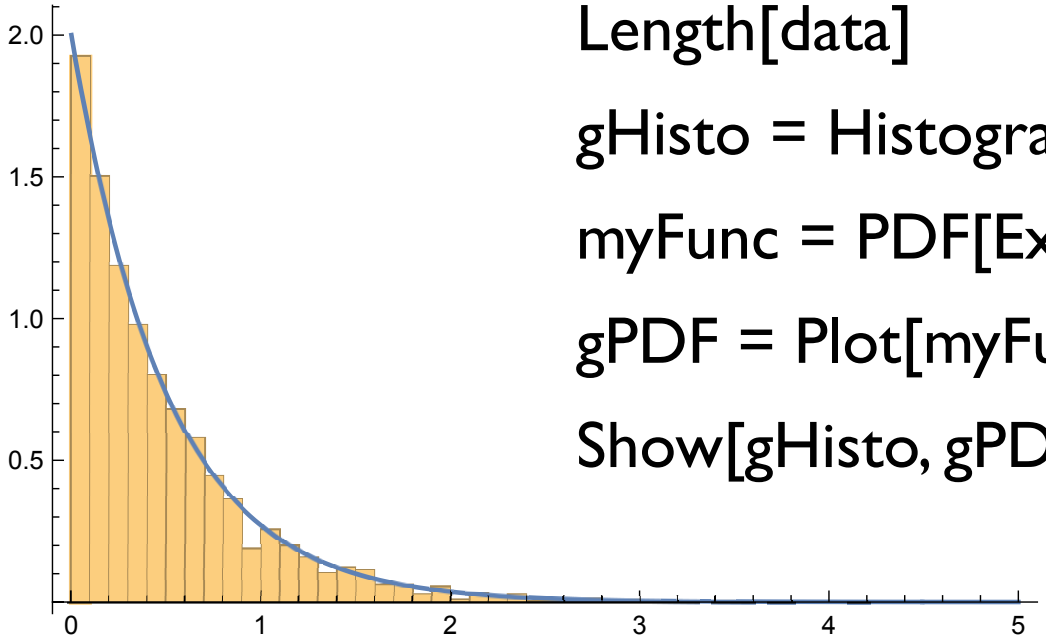
# 指数分布する乱数の生成



# 指数分布する乱数の生成

Mathematicaを用いて視覚化（参考資料を参照）

```
SetDirectory["/Users/takasu/計算機実験 2023/“]

data = ReadList["data-exp", Real];

Length[data]

gHisto = Histogram[data, {0, 5, 0.1}, “PDF"]

myFunc = PDF[ExponentialDistribution[2], x]

gPDF = Plot[myFunc, {x, 0, 5}, PlotRange -> All]

Show[gHisto, gPDF]
```



# 乱数の変数変換

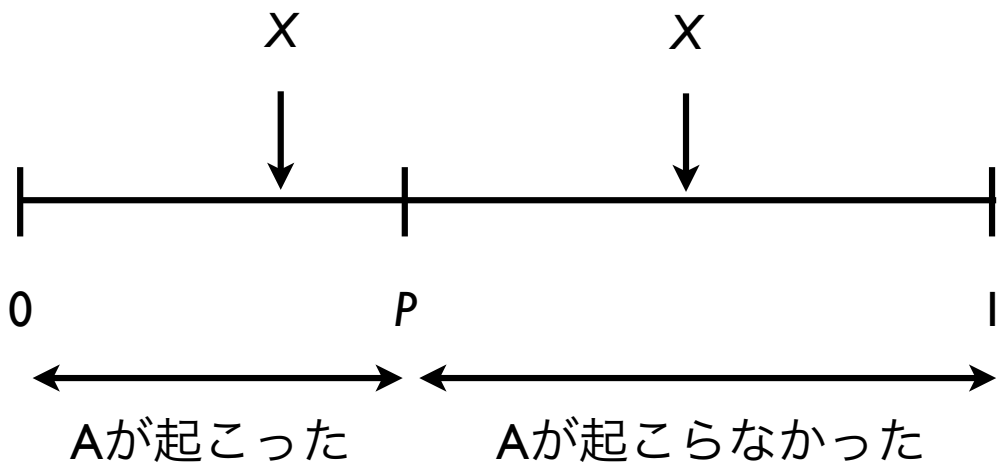
- 一様乱数 X = U[0, 1) を様々な関数Y = f(X) で変換し、乱数Yの分布について調べてみる
- 正規分布する乱数 N(0, 1)（平均0、分散1） —> $N(m, \sigma^2)$ を生成する（Box Muller 法：後述）
- 関数Y = f(X) の逆関数が解析的に与えられる場合、Yが従う分布（確率密度関数）を解析的に求め、擬似乱数生成シミュレーションと比較する（後述）

# モンテカルロ法

- 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）、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

# 確率的な事象のプログラム実装

- 確率 $P$ で起こる事象 A をプログラムとして実装（$0 \leq P \leq 1$）
  - 疑似一様乱数 $X = U[0, 1)$ を生成
  - $X < P$ なら、事象 A が起こったと見なす。そうでなければ起こらなかったと見なす（ベルヌイ試行 Bernoulli trial）




$X = U[0, 1)$ は区間 $[0, 1)$ で一様

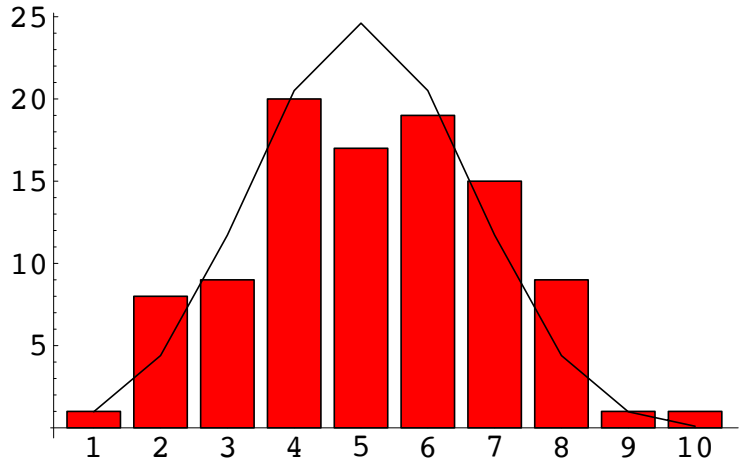
# 二項分布する乱数生成（コイン投げ）

- 正しく造られたコインは裏表がでる確率は $P = 1/2$ である
- $n$ 回コインを投げたとき、$i$ 回表がでる確率（$i = 0, 1, 2, \ldots, n$）は、二項分布 binomial distribution で与えられる（ベルヌイ試行の複数回繰り返し）

$$P_n(i) = \quad {}_n\mathrm{C}_i \left(\frac{1}{2}\right)^n$$

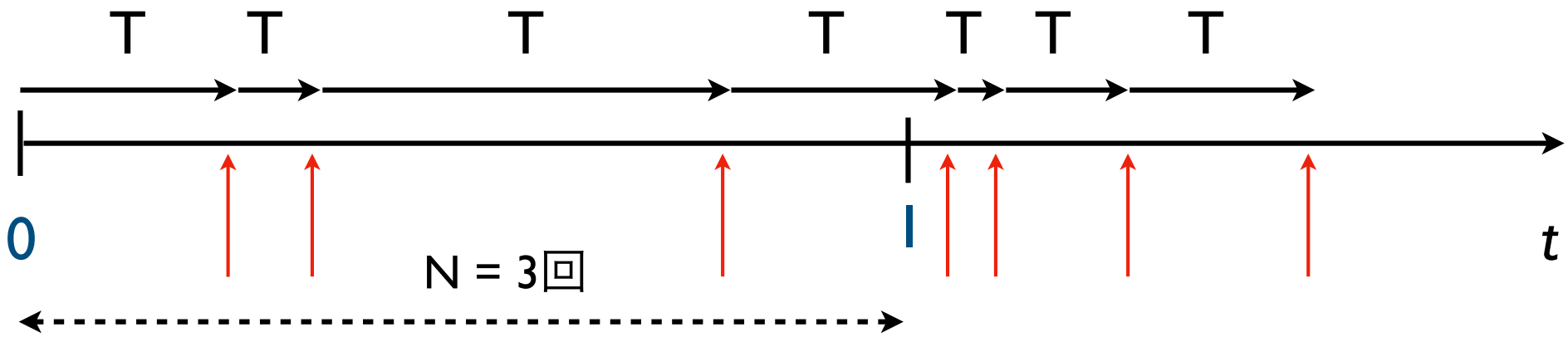
具体例：$P = 1/2$, $n = 10$ を 100 回繰り返したとき、表が出た回数

{7, 7, 9, 8, 4, 7, 8, 3, 7, 3, 1, 4, 2, 4, 5, 5, 6,
3, 8, 7, 5, 2, 6, 4, 4, 6, 3, 3, 5, 6, 2, 5, 8, 2,
6, 2, 6, 4, 6, 7, 10, 4, 5, 8, 5, 4, 4, 5, 4, 5, 3,
3, 8, 5, 5, 6, 5, 5, 4, 6, 6, 4, 2, 7, 2, 6, 6, 6,
4, 7, 5, 6, 3, 6, 7, 3, 6, 6, 8, 5, 6, 4, 7, 4, 2,
4, 5, 8, 7, 4, 4, 5, 7, 4, 4, 7, 7, 8, 6, 7}



# ポアソン分布する乱数

- 発生率（単位時間の発生回数）が γ > 0 である出来事が起こるまでの待ち時間 T はパラメータ γ の指数分布に従う（Poisson過程）




単位時間内に起こる回数 N はパラメータ γ のポアソン過程に従う

$$P_\gamma(N=n)=\frac{e^{-\gamma}\gamma^n}{n!}\quad n=0,1,2,\cdots$$

# 問題 3

- 0 から 1 の一様乱数 X = U[0, 1) を下記の変数変換を施して得られる確率変数 Y が満たす確率密度関数 q(y) を求めよ。疑似一様乱数を用いて、この結果を確かめよ
  - Y = – log X
  - Y = √X
- 2つの一様乱数 X = U[0, 1), Y = U[0, 1) を用いて以下の変換で得られる確率変数 Z は、平均が0、分散が1の正規分布に従う（ボックス - ミュラー法）。Z を多数を生成し確率分布を描いて確認せよ

$$Z = \sqrt{-2 \log X} \cos 2\pi Y$$

# 問題 4

- ベルヌイ試行を再現する関数を C 言語で実装せよ
- 引数として、出来事が起こる確率を double 型の値で与える
- 戻り値として、出来事が起こった場合 int 1 を、起こらなかった場合 int 0 を返却するものとする

```
int rv_bernoulli(double prob)
{
  int event;

  .....
  return event;
}
```

# 問題 5

- コインを $n$ 回投げる試行を $k$ 回繰り返す。各試行で表が出た回数を $i$ とする。$i$ をファイルに書き出せ
- 試行数 $k$ を十分大きくとったとき、$i$ の分布図を描け。また理論分布と比較せよ。$n$ の値は適当でよい

1回の試行で表が出た回数を戻り値として返す関数を定義して使うこと

```
int rv_binomial(int num, double prob)
{

  .....
  return count;
}
```

# 問題 6

- パラメータ λ > 0 で指数分布する乱数 T を生成する事により、パラメータ λ に従うポアソン分布する乱数 N を多数（数千個）生成し、理論値と比較せよ
- 理論値との比較（確率分布の比較）には、$\lambda$ =1, 5, 10 の3通りの値を用いよ

下記の関数を定義して用いよ

```
int rv_poisson(double lambda)
{

  .....
  return count;
}
```