Lineární kongruentní generátor

smíšený generátor

$$I_{j+1} = a\, I_j + c \qquad (\mathrm{mod}\, m)$$

$$a, c, m \in \boldsymbol{N}$$

$$x_j = I_j / m$$

$$0 \leq x_j < 1 \qquad x_j \in U(0,1)$$

- perioda nejvýše $m - 1$ ($m \approx 2^{32}$)
- semínko I_0
- pozor na korelaci a přetečení dat

čistě multiplikativní generátor

$$I_{j+1} = a\, I_j \qquad (\mathrm{mod}\, m)$$

$$a, m \in \boldsymbol{N}$$

$$x_j = I_j / m$$

$$0 \leq x_j < 1 \qquad x_j \in U(0,1)$$

- příklad $a = 7^5 = 16807$

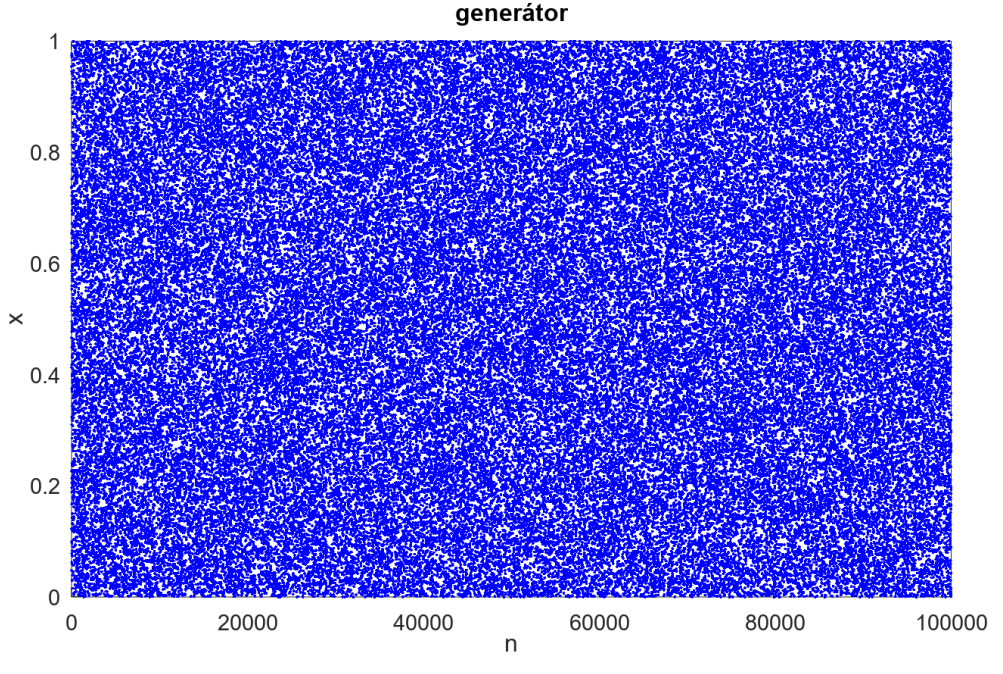
  $m = 2^{31} - 1 = 2147483647$

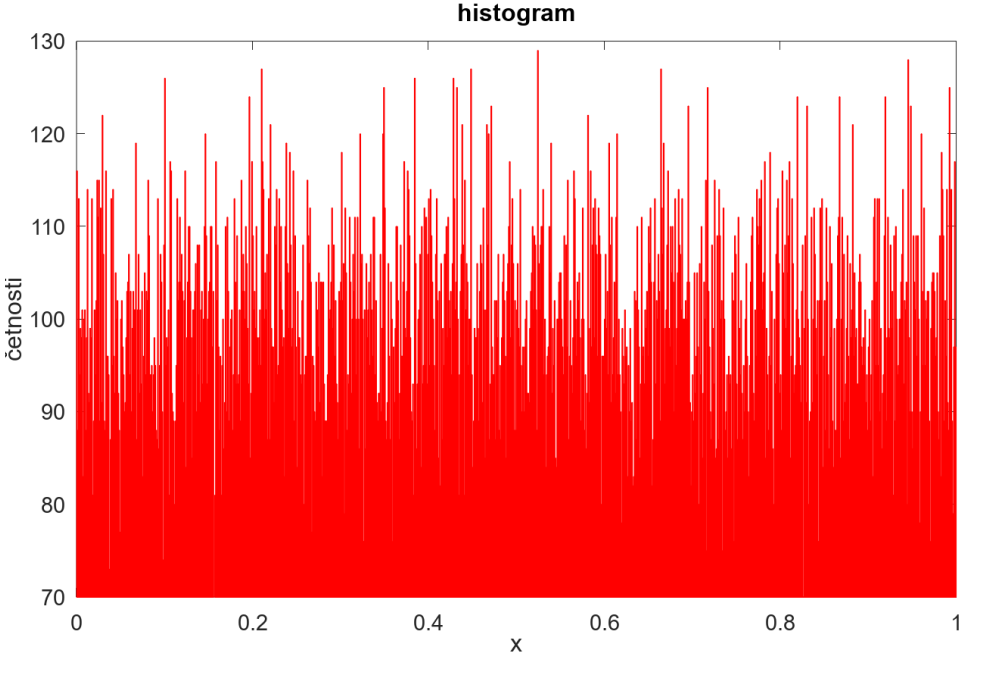
  perioda: $2^{31} - 2 \approx 2.1 \times 10^9$

Lineární kongruentní generátor

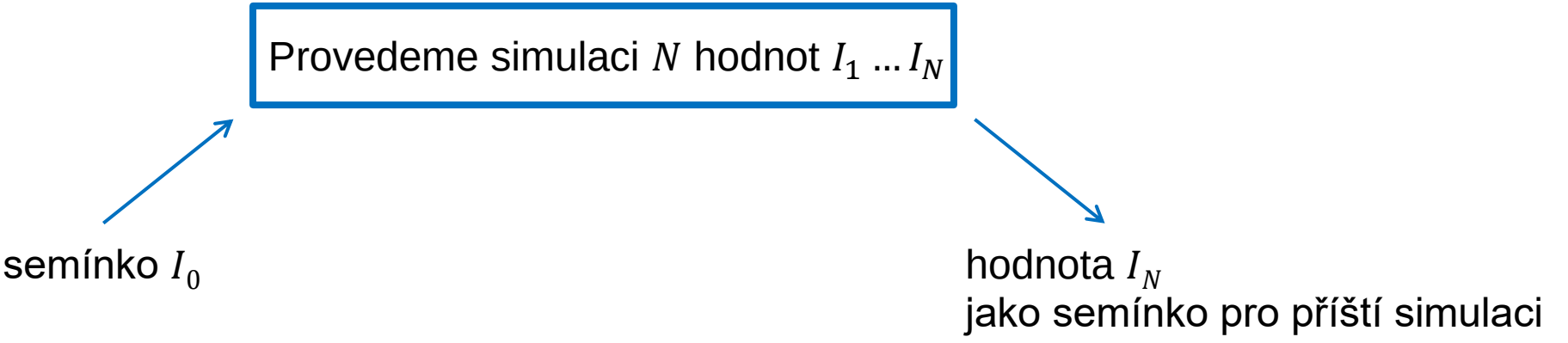
$N = 100\,000$

- multiplikativní generátor, $a = 16807$, $I_0 = 41369$, $m = 2^{31} - 1$





Monte Carlo simulace



- pokud jde o úplně první simulaci, vymyslím si semínko I_0

  jinak

- načtu semínko jako poslední uloženou hodnotu z předchozí simulace

  nebo

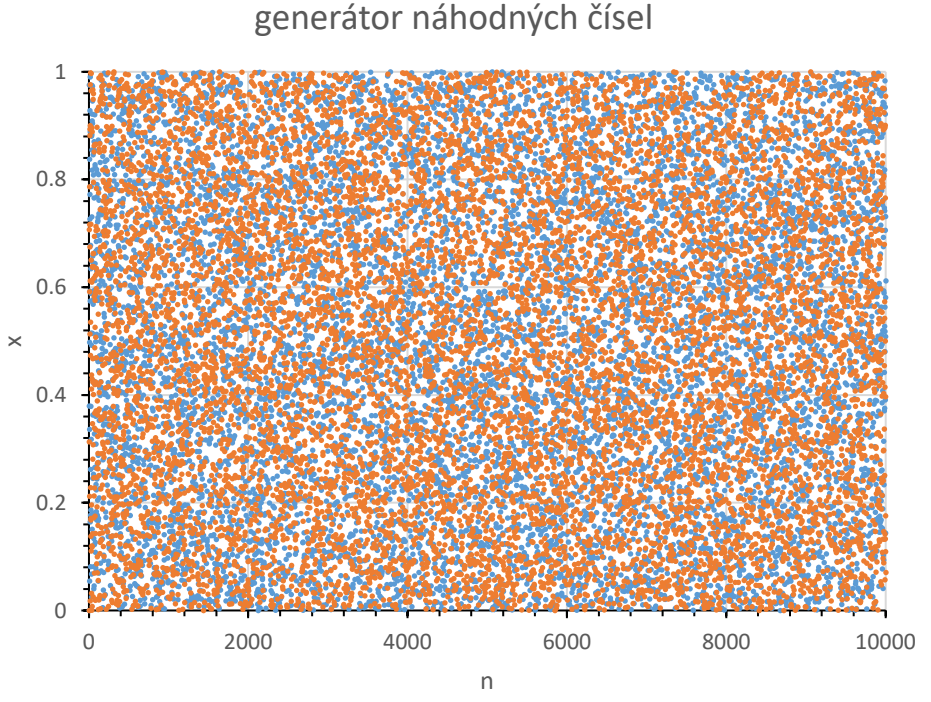
- semínko nějak vytvořím, aby to bylo pokaždé jiné číslo (např. aktuální datum, čas atd.)

Monte Carlo simulace v Excelu

`generator.xlsx`

$N = 10\,000$

- multiplikativní generátor $a = 16807,$ $I_0 = 48541, m = 2^{31} - 1$
- smíšený generátor $a = 16807, c = 4136,$ $I_0 = 48541, m = 2^{31} - 1$



generátor náhodných čísel U(0,1)

```
=NÁHČÍSLO()
```

histogram

```
=ČETNOSTI(D2:D10001,R1:R100)
```

data biny

maticové vzorce v Excelu:

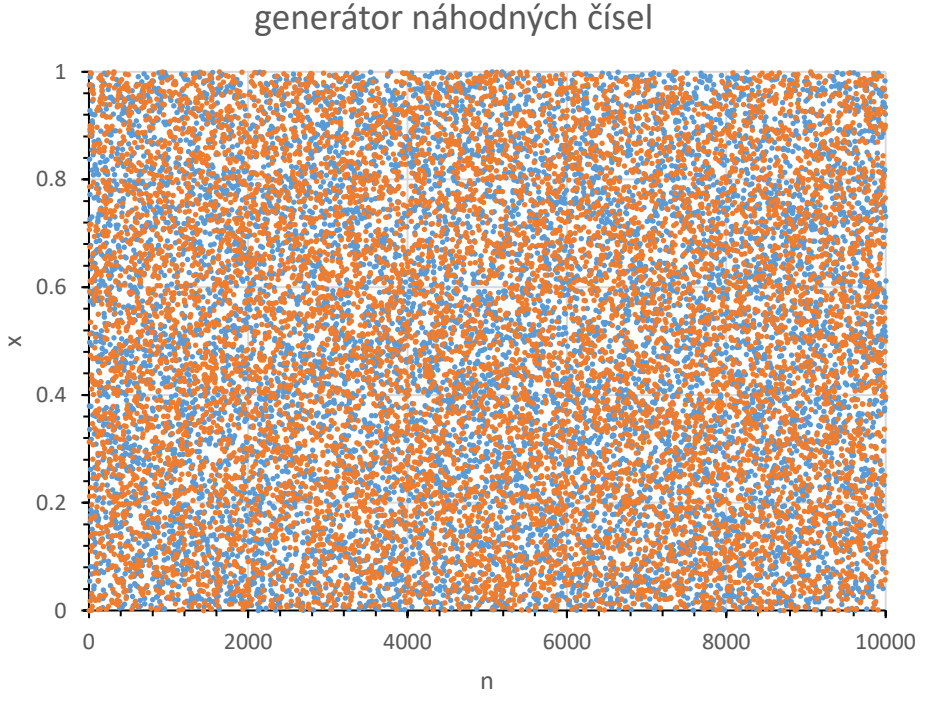
1. označit výstupní oblast
2. napsat vzorec a stisknout Ctrl+Shift+ENTER

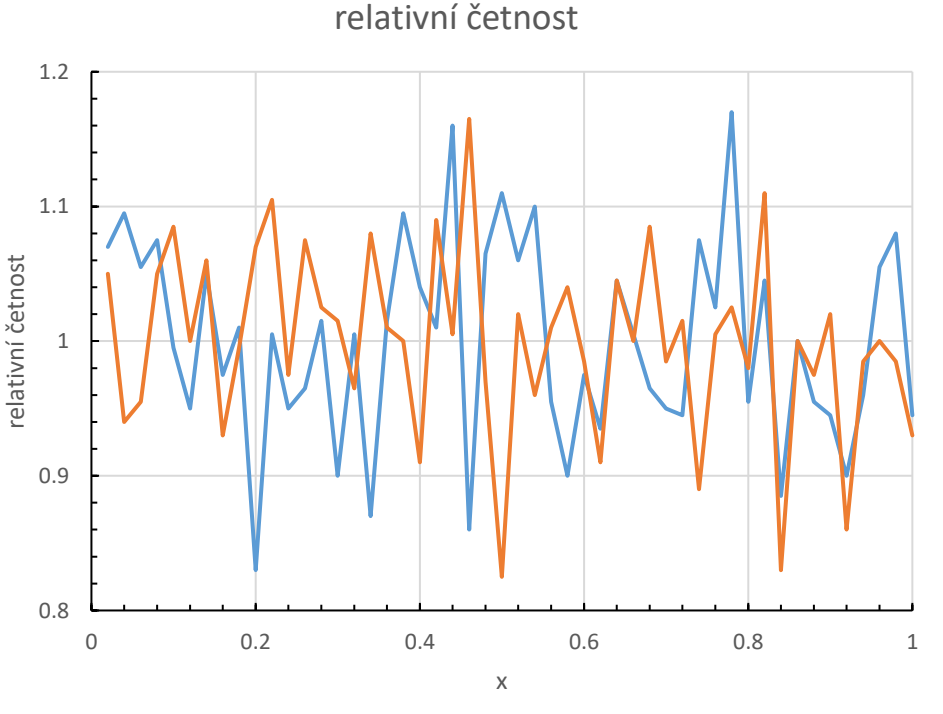
Monte Carlo simulace v Excelu

`generator.xlsx`

$N = 10\,000$

- multiplikativní generátor $a = 16807,$ $I_0 = 48541, m = 2^{31} - 1$
- smíšený generátor $a = 16807, c = 4136,$ $I_0 = 48541, m = 2^{31} - 1$



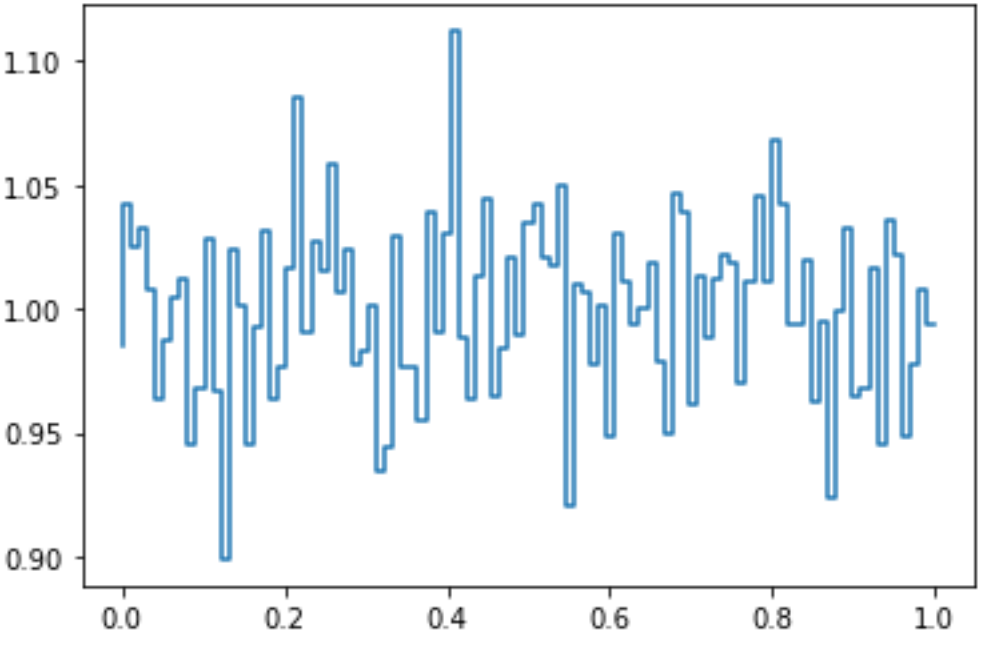


Monte Carlo simulace v Pythonu

`rand-histogram-1.py`

histogram „bod po bodu“

```
1   import numpy as np
2   import matplotlib.pyplot as plt
3   n=100000 #pocet dat
4   nbins=100 #pocet binu
5   x=np.linspace(0,1,nbins) #x-ova souradnice grafu
6   y=np.zeros(nbins) #y-ova souradnice grafu
7   for i in range(0,n):
8       ibin=int(nbins*np.random.random()) #generuje nahodne cislo
9       y[ibin]=y[ibin]+1 #inkrementace histogramu
10  y=y/(n/nbins) #normalizace
11  plt.step(x,y) #vykresleni grafu
```

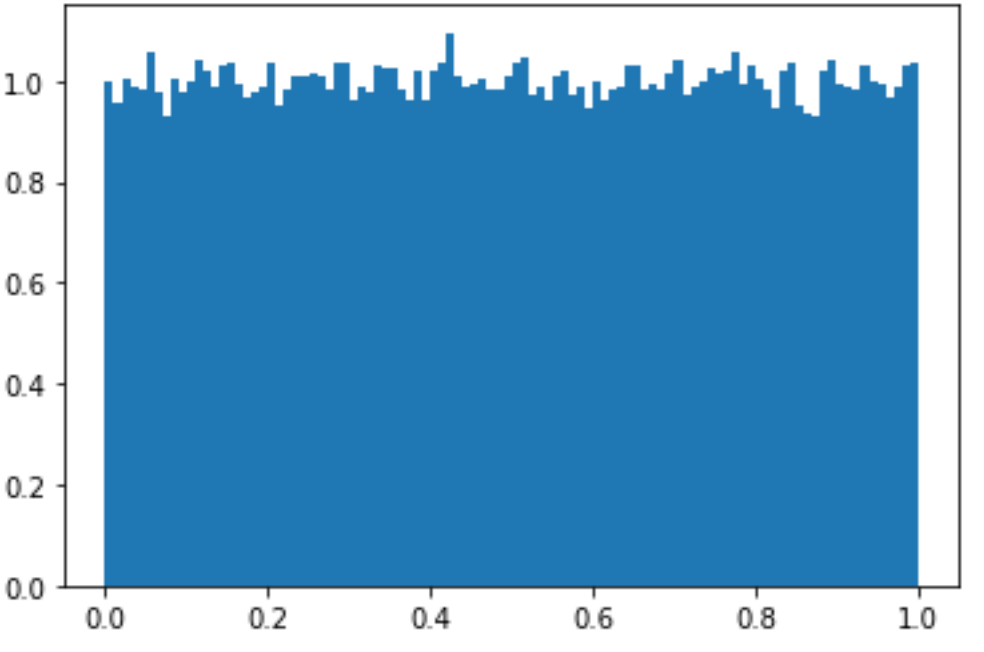


Monte Carlo simulace v Pythonu

`rand-histogram-2.py`

histogram metodou plt.histogram

```
1  import numpy as np
2  import matplotlib.pyplot as plt
3  n=100000 #pocet dat
4  nbins=100
5  data=np.array(n) #deklarace pole x-souradnic
6  data=np.random.random_sample(n) #naplni pole data nahodnymi hodnotami
7  plt.hist(data,bins=nbins,density=True) #udela a vykresli histogram
```

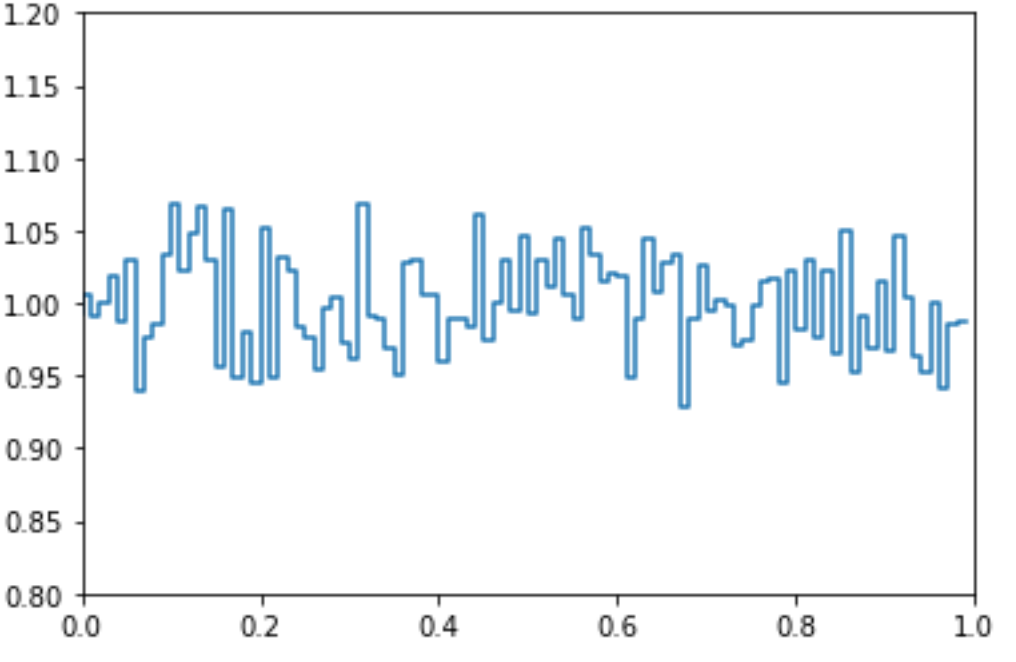


Monte Carlo simulace v Pythonu

`rand-histogram-3.py`

histogram metodou np.histogram

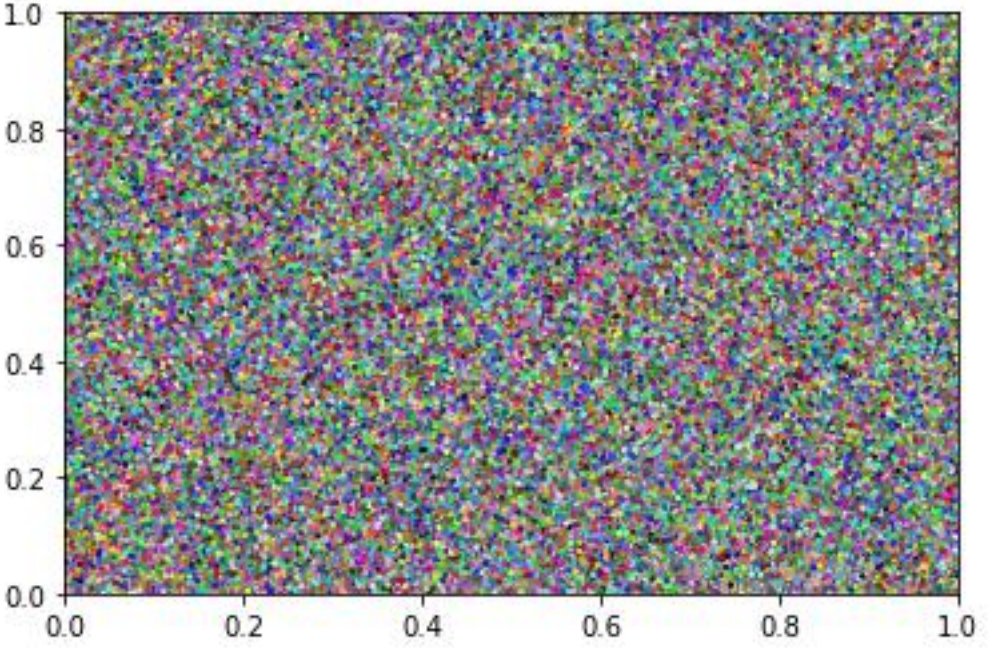
```
1   import numpy as np
2   import matplotlib.pyplot as plt
3   n=100000 #pocet dat
4   nbins=100
5   data=np.array(n) #deklarace pole x-souradnic
6   data=np.random.random_sample(n) #naplni pole nahodnymi hodnotami
7   hist,bin_edges=np.histogram(data,bins=nbins,density=True) #udela histogram
8   x=bin_edges[0:nbins] #x-ova souradnice
9   y=hist  #y-ova souradnice
10  fig,ax=plt.subplots() #vytvoreni obrazku
11  ax.step(x,y) #vykresleni grafu
12  plt.xlim(0,1) #nastaveni rozmezi osy x: 0,1
13  plt.ylim(0.8,1.2) #nastaveni rozmezi osy y> 0.5,1.5
```



Barevný test v Pythonu

randomcolours.py

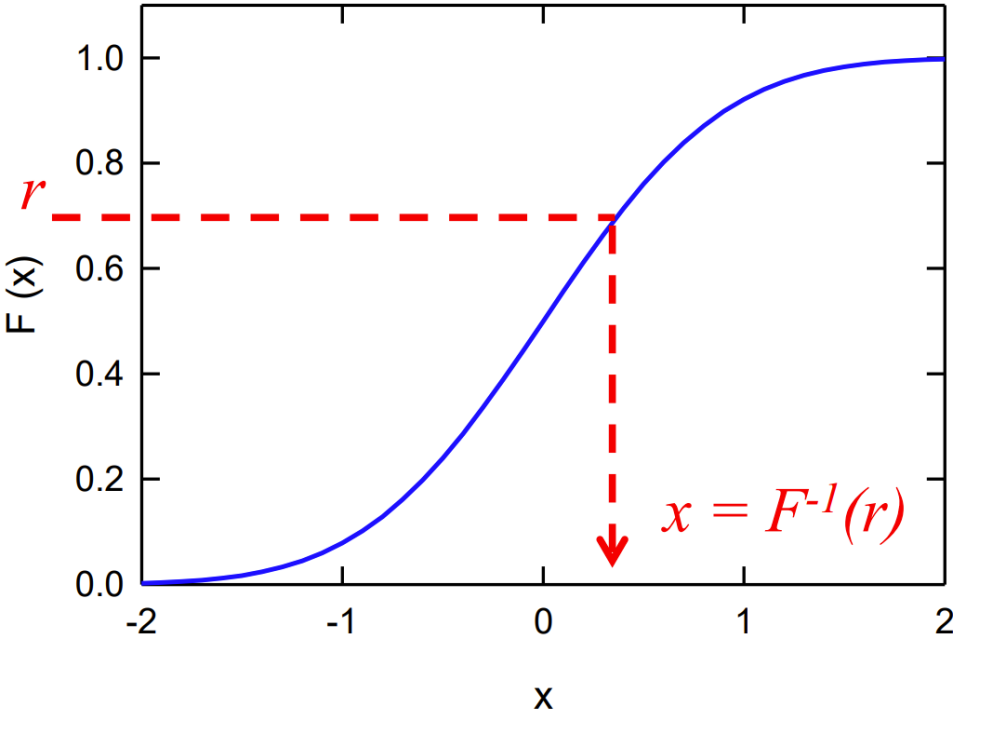
```
1  import numpy as np #knihovna numpy
2  import matplotlib.pyplot as plt #knihovna matplot.lib.pyplot
3  n=100000
4  x=np.array(n) #deklarace pole x-souradnice
5  y=np.array(n) #deklarace pole y-souradnice
6  colours=np.array([n,3]) #deklarace pole barva
7  x=np.random.random_sample(n)#vygeneruje 100000 nah. cisel U(0,1)
8  y=np.random.random_sample(n) #vygeneruje 100000 nah. cisel U(0,1)
9  colours=np.random.random_sample([n,3]) #vygeneruje 100000 trojic nah. cisel U(0,1)
10 plt.scatter(x,y, s=5, c=colours, edgecolors="none") #nakresli graf
11 #nastaveni os od 0 do 1
12 ax=plt.gca()
13 ax.set_xlim(left=0,right=1)
14 ax.set_ylim(bottom=0,top=1)
15 plt.draw()
16 #ulozeni do souboru formatu PNG
17 plt.savefig("randomcolours.png",dpi=150)
```



Monte Carlo simulace – metoda inverzní funkce

Nechť x je náhodná proměnná s rozdělením popsaným hustotou pravděpodobnosti $f(x)$ a distribuční funkcí $F(x)$.

Potom má nová náhodná proměnná $r = F(x)$ rovnoměrné rozdělení $U(0,1)$.



Metoda inverzní funkce

1. Vygeneruj náhodnou proměnnou $r \in U(0,1)$.

2. Vypočítej $x = F^{-1}(r)$ inverzní funkci k distribuční funkci $F(x)$ požadovaného rozdělení.

Monte Carlo simulace – metoda inverzní funkce

Exponenciální rozdělení

1. hustota pravděpodobnosti $f(t) = \frac{1}{\tau} e^{-\frac{t}{\tau}}$

2. distribuční funkce $F(t) = \int_0^t \frac{1}{\tau} e^{-\frac{z}{\tau}} \, \mathrm{d}z \Rightarrow F(t) = 1 - e^{-\frac{t}{\tau}}$

3. náhodná proměnná s rovnoměrným rozdělením $r \in U(0,1)$

4. inverzní funkce k distribuční funkci $F^{-1}(r) = -\tau \ln(1 - r)$

5. náhodná proměnná s exponenciálním rozdělením $t = -\tau \ln(1 - r)$

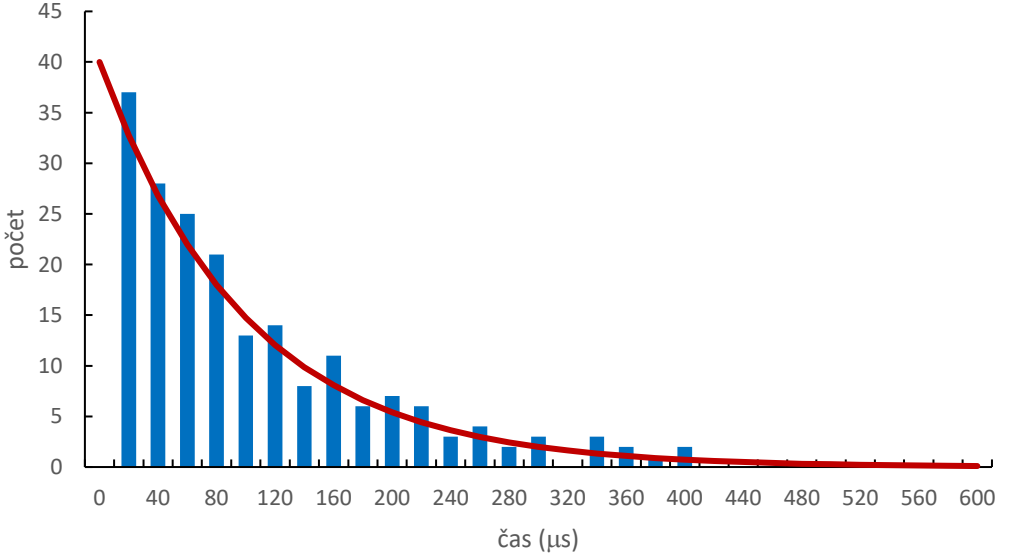
6. ekvivalentně $t = -\tau \ln r$

Monte Carlo simulace

simulace-exp-rozpad.xlsx

exp-rozpad2.py

1. Doba života radionuklidu X je $100\ \mu s$. Proveďte v Excelu a v Pythonu simulaci měření radioaktivního rozpadu (200 hodnot). Nakreslete histogram naměřených hodnot.



generátor náhodných čísel $U(0,1)$

```
Ai = NÁHČÍSLO()
```

exponenciální rozdělení $\tau = 100$ metodou inverzní funkce

```
Bi = -100*LN(Ai)
```

histogram

```
{=ČETNOSTI(B2:B199,D2:D32)}
```

oblast vygenerovaných dat

biny

maticové vzorce v Excelu:

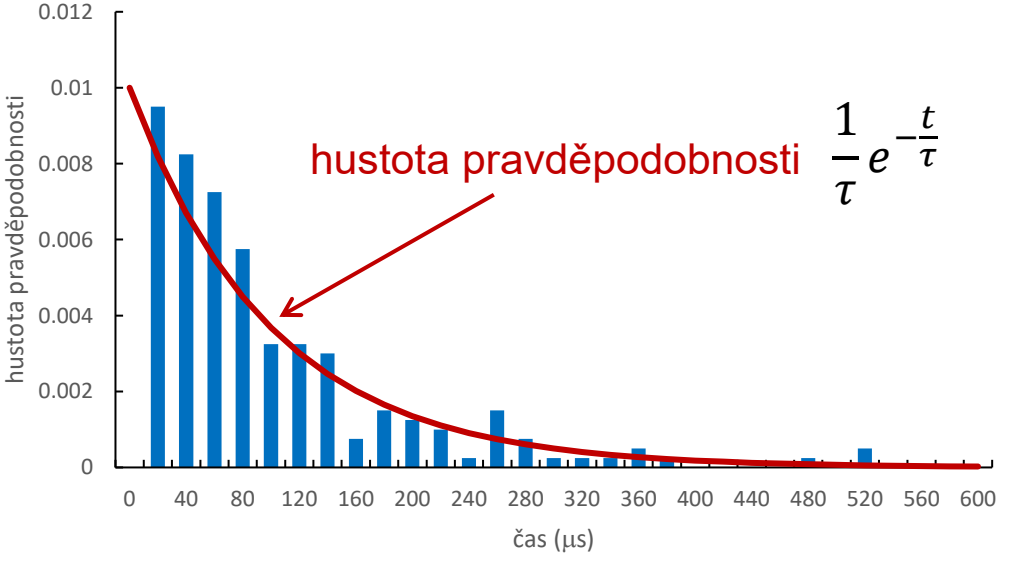
1. označit výstupní oblast
2. napsat vzorec a Ctrl+Shift+Enter

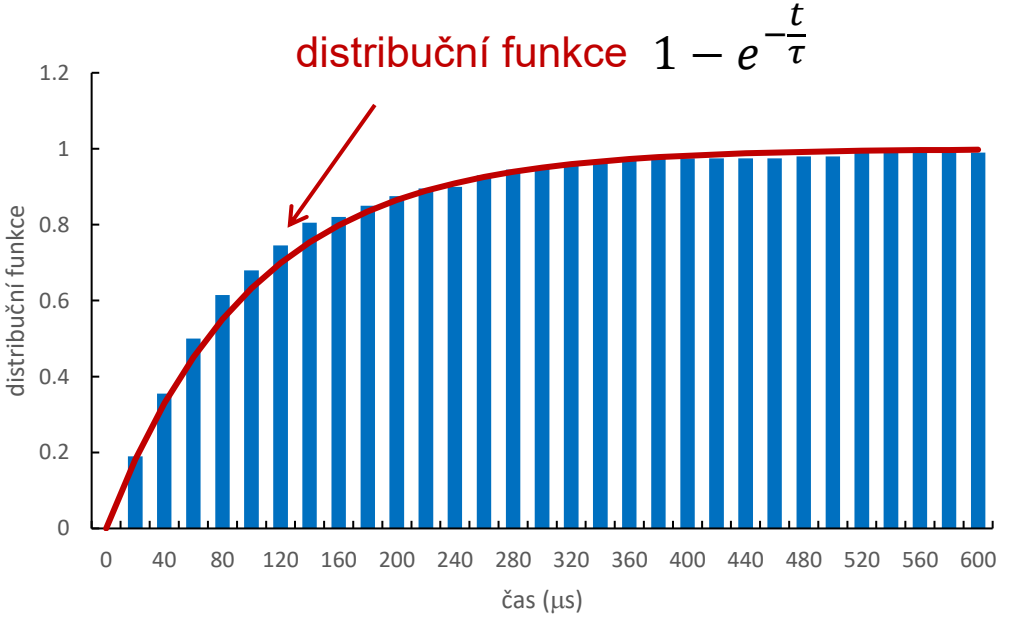
Monte Carlo simulace

simulace-exp-rozpad.xlsx

exp-rozpad2.py

2. Z dat vygenerovaných v předchozím příkladu udělejte normovaný histogram a srovnejte s hustotou pravděpodobnosti exponenciálního rozdělení a kumulovaný histogram, který srovnejte s distribuční funkcí exponenciálního rozdělení.





3. Jaká je pravděpodobnost, že radionuklid bude žít déle než $200\ \mu s$?