Problémy s reprezentací reálných čísel

February 27, 2020

1 Problémy s reprezentací reálných čísel

1.1 Určitý integrál

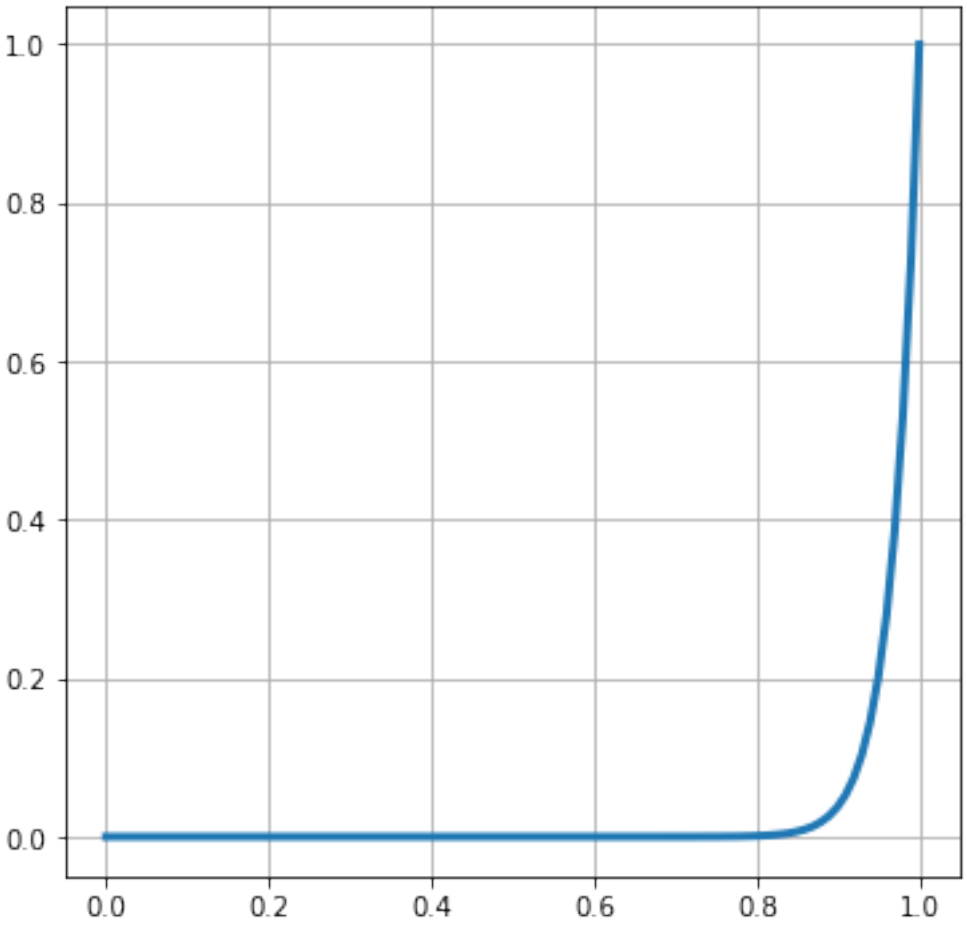
Spočítejte hodnotu integrálu

$$\int_0^1 x^{30} e^{x-1} \, \mathrm{d}x$$

[1]:
```
import math
import numpy as np
import matplotlib.pyplot as plt
```

[2]:
```
xs = np.linspace(0, 1, 100)
ys = xs ** 30 * np.exp(xs - 1)
plt.figure(figsize=(6, 6))
plt.grid()
plt.plot(xs, ys, linewidth=3)
```

[2]:
```
[<matplotlib.lines.Line2D at 0x7fea223dde50>]
```



$$J_n = \int_0^1 x^n e^{x-1}\,\mathrm{d}x \tag{1}$$

$$J_n = 1 - n\int_0^1 x^{n-1} e^{x-1}\,\mathrm{d}x \tag{2}$$

$$J_n = 1 - nJ_{n-1} \tag{3}$$

$$J_0 = 1 - 1/e \tag{4}$$

[3]:
```
J = {}
J[0] = 1 - 1 / math.e
J[0]
```

[3]:
```
0.6321205588285577
```

[4]:
```
for n in range(1, 31):
    J[n] = 1 - n * J[n - 1]
```

[5]:
```
J[30]
```

[5]:
```
-3296762455608386.5
```

1.2 Jednoduchá porovnání

[6]:
```
1 + 1 == 2
```

[6]:
```
True
```

[7]:
```
1.5 + 0.5 == 2
```

[7]:
```
True
```

[8]:
```
0.1 + 0.2 == 0.3
```

[8]:
```
False
```

1.3 Jednoduchá porovnání - řešení

[9]:
```
0.1 + 0.2
```

[9]:
```
0.30000000000000004
```

$$|x - y| < \epsilon$$

[10]:
```
abs((0.1 + 0.2) - 0.3) < 10 ** (-6)
```

[10]:
```
True
```

1.4 Určitý integrál - řešení

Výpočet transformujeme:

$$J_{n-1} = \frac{1 - J_n}{n}$$

Odhad:

$$J_{35} = 0$$

[11]:
```
J = {}
J[35] = 0
for n in reversed(range(30, 36)):
    J[n - 1] = (1 - J[n]) / n
```

[12]:
```
J[30]
```

[12]:
```
0.03127967462644882
```

1.5 Platné číslice

Počet platných číslic je zhruba 16:

[13]:
```
1234567890 + 0.000001
```

[13]:
```
1234567890.000001
```

[14]:
```
1234567890 + 0.0000001
```

[14]:
```
1234567890.0
```

1.6 Limity reprezentace

Nejvyšší hodnota, kterou lze uložit:

[15]:
```
import sys
sys.float_info.max
```

[15]:
```
1.7976931348623157e+308
```

1.6.1 Geometrický průměr

$$G(a_i) = \sqrt[n]{\prod_{i=1}^{n} a_i}$$

Vygenerujeme 1000 náhodných čísel od 0 do 10:

[16]:
```
N = 1000
array = np.random.random_sample(N) * 10
```

Aritmetický průměr hodnot:

[17]:
```
array.mean()
```

[17]:
```
4.989164316069559
```

Geometrický průměr hodnot:

[18]:
```
np.prod(array) ** (1 / N)
```

```
/home/krab1k/.anaconda3/lib/python3.7/site-
packages/numpy/core/fromnumeric.py:90: RuntimeWarning: overflow encountered in
```

```
reduce
  return ufunc.reduce(obj, axis, dtype, out, **passkwargs)
```

[18]:
```
inf
```

Transformace užitím logaritmů:

$$G(a_i) = exp\left(\frac{1}{n}\sum_{i=1}^{n}\log a_i\right)$$

[19]:
```
np.exp(sum(np.log(array)) / N)
```

[19]:
```
3.6798755862535586
```

1.7 Asociativita sčítání reálných čísel

Obecně neplatí:

$$a + (b + c) = (a + b) + c$$

[20]:
```
array = np.random.random_sample(10)
array
```

[20]:
```
array([0.06220636, 0.96272605, 0.74346763, 0.39376898, 0.33936434,
       0.48068666, 0.9848948 , 0.5023336 , 0.45191039, 0.49248488])
```

[21]:
```
array2 = np.random.permutation(array)
array2
```

[21]:
```
array([0.45191039, 0.96272605, 0.39376898, 0.74346763, 0.33936434,
       0.48068666, 0.9848948 , 0.06220636, 0.49248488, 0.5023336 ])
```

[22]:
```
array3 = np.random.permutation(array)
array3
```

[22]:
```
array([0.06220636, 0.39376898, 0.74346763, 0.48068666, 0.5023336 ,
       0.9848948 , 0.49248488, 0.96272605, 0.33936434, 0.45191039])
```

[23]:
```
sum(array) == sum(array2)
```

[23]:
```
False
```

[24]:
```
sum(array) == sum(array3)
```

[24]:
```
True
```

[25]:
```
sum(array), sum(array2), sum(array3)
```

[25]:
```
(5.413843689955182, 5.413843689955181, 5.413843689955182)
```

Správným řešením (zejména jsou-li mezi čísly řádové rozdíly) je seřadit čísla od nejmenšího po největší (v absolutní hodnotě) a pak teprve sečíst. Např.:

[26]:
```
sum(sorted(array, key=lambda x: abs(x)))
```

[26]:
```
5.413843689955182
```

1.8 Stabilita algoritmu

Vyřešte následující systém lineárních rovnic:

$$10^{-10}x_1 + x_2 = 1 \tag{5}$$
$$x_1 + 2x_2 = 3 \tag{6}$$

[27]:
```
# Naivní implementace Gaussovy eliminační metody -> nepoužívat

def GEM(matrix, vector):
    A = np.copy(matrix)
    b = np.copy(vector)
    n = A.shape[0]
    for i in range(n):
        for j in range(i + 1, n):
            mult = A[j][i] / A[i][i]
            A[j] -= mult * A[i]
            b[j] -= mult * b[i]
    for i in reversed(range(n)):
        for j in range(i - 1, -1, -1):
            mult = A[j][i] / A[i][i]
            b[j] -= mult * b[i]

    return b / np.diagonal(A)
```

$$A \cdot x = b$$
$$\begin{bmatrix} 10^{-10} & 1 \\ 1 & 2 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 3 \end{bmatrix}$$

[28]:
```
A = np.array([[10 ** (-10), 1], [1, 2]], dtype=np.float32)
b = np.array([1, 3], dtype=np.float32)
A, b
```

[28]:
```
(array([[1.e-10, 1.e+00],
        [1.e+00, 2.e+00]], dtype=float32),
 array([1., 3.], dtype=float32))
```

Naivní vs. správné řešení:

[29]:
```
GEM(A, b)
```

[29]:
```
array([0., 1.], dtype=float32)
```

[30]:
```
np.linalg.solve(A, b)
```

[30]:
```
array([1., 1.], dtype=float32)
```