Выберите многомерную функцию, реализуйте на Python для анализа чувствительности с библиотекой SALib, проведите анализ чувствительности, измерьте вреся -- 3 балла

In [1]:

```
import time
import numpy as np
from numba import njit, prange
from SALib.sample import saltelli
from SALib.analyze import sobol
```

In [2]:

```
a = 2
b = 0.7
```

In [3]:

```
def evaluate_model(x):
    return a * np.sin(x[2]) ** 2 * np.cos(x[2]) + b * np.sin(x[0])**3 * np.cos(x[1])**3
```

In [4]:

```
problem = {
    'num_vars' : 3,
    'names' : ['x1', 'x2', 'x3'],
    'bounds' : [[-6, 6],
                [-6, 6],
                [-6, 6]
               ]
}
```

In [5]:

```
n = 100000
start = time.time()
param_values = saltelli.sample(problem, n)
print("samples generation took %s seconds" %(time.time() - start))
Y = np.zeros([param_values.shape[0]])

start = time.time()
for i, X in enumerate(param_values):
    Y[i] = evaluate_model(X)

print("model evaluation took %s seconds" %(time.time() - start))

start = time.time()
Si = sobol.analyze(problem, Y)

print("SA took %s seconds" %(time.time() - start))
# first-order indices
print(Si['S1'])
# total indices for each variable
print(Si['ST'])
# second order indices
print("x1-x2:", Si['S2'][0,1])
print("x1-x3:", Si['S2'][0,2])
print("x2-x3:", Si['S2'][1,2])
```

```
samples generation took 10.129864931106567 seconds
model evaluation took 10.090608835220337 seconds
SA took 14.289999961853027 seconds
[8.26645488e-04 1.87719597e-04 8.51950868e-01]
[0.14816062 0.14701181 0.8520375 ]
x1-x2: 0.1470339970177656
x1-x3: 0.0003233657682393254
x2-x3: -0.00018170574726417144
```

Ускорьте вычисления любым из способов с лекции (PyBind11, ctypes, cython, numba) -- 3 балла

In [6]:

```
@njit
def opt_evaluate_model(x):
    return a * np.sin(x[2]) ** 2 * np.cos(x[2]) + b * np.sin(x[0])**3 * np.cos(x[1])**3
```

In [7]:

```
n = 100000
start = time.time()
param_values = saltelli.sample(problem, n)
print("samples generation took %s seconds" %(time.time() - start))
Y = np.zeros([param_values.shape[0]])

start = time.time()
for i, X in enumerate(param_values):
    Y[i] = opt_evaluate_model(X)

print("model evaluation took %s seconds" %(time.time() - start))

start = time.time()
Si = sobol.analyze(problem, Y)

print("SA took %s seconds" %(time.time() - start))
```

```
samples generation took 5.160666227340698 seconds
model evaluation took 1.2633419036865234 seconds
SA took 12.311583042144775 seconds
```

Используйте параллелизм -- 4 балла

Распараллелить я решил опять таки через Numba, пробовал через пул процессов не давало сильного прироста, как через prange

In [8]:

```
n = 100000
start = time.time()
param_values = saltelli.sample(problem, n)
print("samples generation took %s seconds" %(time.time() - start))
Y = np.zeros([param_values.shape[0]])

start = time.time()
for i in prange(100000):
    Y[i] = opt_evaluate_model(param_values[i])

print("model evaluation took %s seconds" %(time.time() - start))

start = time.time()
Si = sobol.analyze(problem, Y)

print("SA took %s seconds" %(time.time() - start))
```

```
samples generation took 4.898255825042725 seconds
model evaluation took 0.1169431209564209 seconds
SA took 12.18838381767273 seconds
```