[6]:
```python
#Третья функция
import jax.numpy as jnp
from jax import grad, jit, vmap
from jax import random
from jax import jacfwd, jacrev
import numpy as np
import math


n = 10
A = np.random.rand(n,n)
b = np.random.rand(n)
c = np.random.rand(n)

x_test = np.random.rand(n)
def f(x):
    j = np.linalg.norm(A@x - b)
    result = 0.5*j*j
    return result


def analytical_df(x_test):
    bruh = np.dot(x_test,x_test)
    return 2*math.exp((-1)*bruh)*x_test
    # 2*math.exp((-1)np.dot(x,x))@x


def analytical_ddf(x):
    # Your code here
    return ((A + A.T)*0.5)

def autograd_df(x):
    return jacrev(f)(x)

def autograd_ddf(x):
    # Your code here
    return jacfwd(jacrev(f))(x)

print(f'Analytical and autograd implementations of the gradients are close: {np.allclose(analytical_df(x_test), autograd_df(x_tes
#print(f'Analytical and autograd implementations of the hessians are close: {np.allclose(analytical_ddf(x_test), autograd_ddf(x_t
```

```python
import jax.numpy as jnp
from jax import grad, jit, vmap
from jax import random
from jax import jacfwd, jacrev
import numpy as np
```

```python
n = 10
A = np.random.rand(n,n)
b = np.random.rand(n)
c = np.random.rand(n)

x_test = np.random.rand(n)
def f(x):
    result = 0.5*x.T@A@x + b.T@x + c
    return result


def analytical_df(x):

    return (A + A.T)*0.5@x + b

def analytical_ddf(x):
    # Your code here
    return ((A + A.T)*0.5)

def autograd_df(x):
    return jacrev(f)(x)

def autograd_ddf(x):
    # Your code here
    return jacfwd(jacrev(f))(x)

print(f'Analytical and autograd implementations of the gradients are close: {np.allclose(analytical_df(x_test), autograd_df(x_test))}')
print(f'Analytical and autograd implementations of the hessians are close: {np.allclose(analytical_ddf(x_test), autograd_ddf(x_test))}')
print(f(x_test), autograd_df(x_test))
```

```
Analytical and autograd implementations of the gradients are close: True
Analytical and autograd implementations of the hessians are close: True
```