Task3_Prazdnichnykh

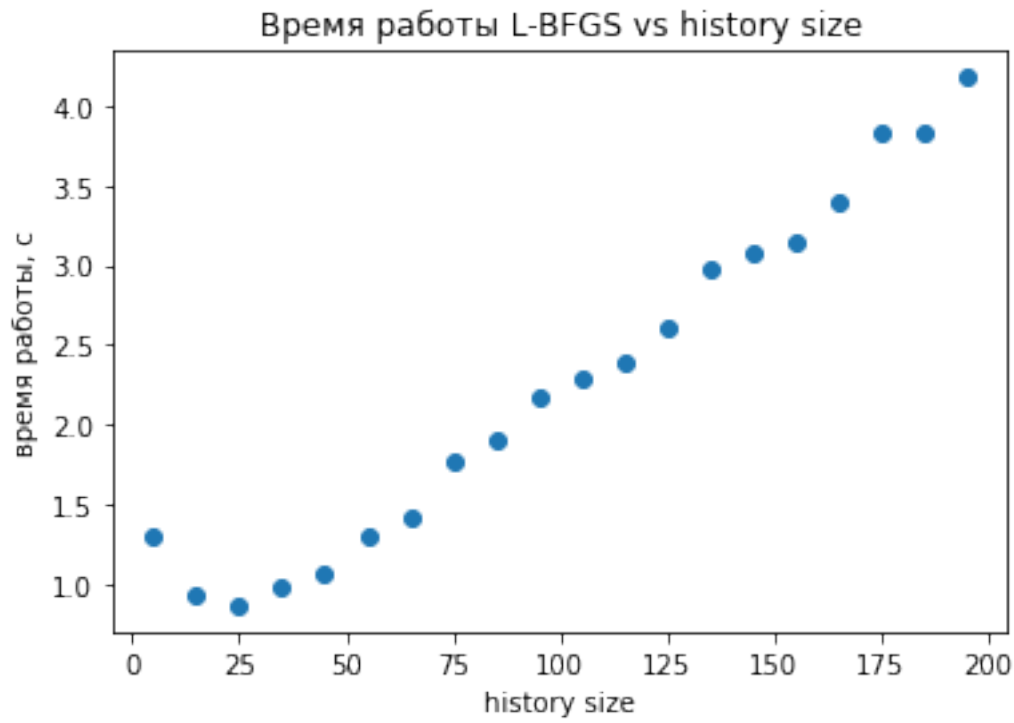
December 8, 2020

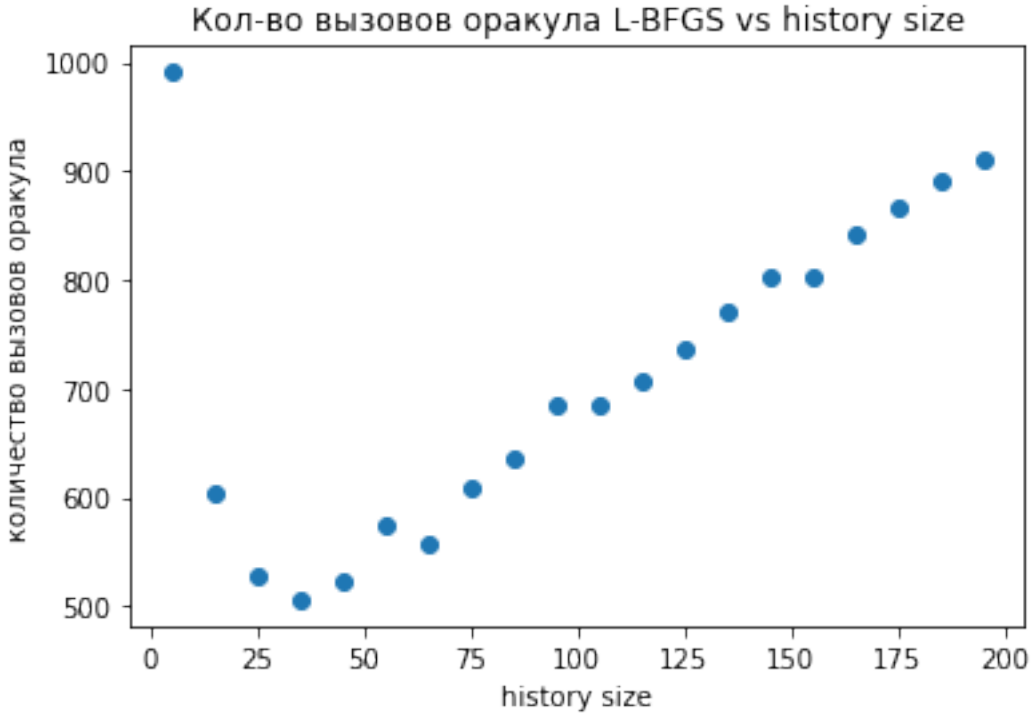
[1]:
```
import numpy as np
import scipy.sparse as sp
from sklearn.datasets import load_svmlight_file
from oracle import Oracle, make_oracle
import scipy as sc
from methods import *
import matplotlib.pyplot as plt
```

[2]:
```
orac = make_oracle('a1a.txt')
x, y = load_svmlight_file('a1a.txt', zero_based=False)
m = x[0].shape[1] + 1
```

, L-BFGS history size a1a.

[9]:
```
w0 = np.zeros(m).reshape((-1, 1))
ls = list(range(5, 200, 10))
times = []
orac_calls = []
for l in ls:
    optimizer = OptimizeLBFGS(l)
    optimizer(orac, w0)
    times.append(optimizer.times[-1])
    orac_calls.append(optimizer.orac_calls[-1])

plt.plot(ls, times, 'o')
plt.xlabel('history size')
plt.ylabel('       ,  ')
plt.title('        L-BFGS vs history size')
plt.show()

plt.plot(ls, orac_calls, 'o')
plt.xlabel('history size')
plt.ylabel('                ')
plt.title('  -           L-BFGS vs history size')
plt.show()
```


Время работы L-BFGS vs history size
время работы, с
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0
25
50
75
100
125
150
175
200
history size


Кол-во вызовов оракула L-BFGS vs history size
количество вызовов оракула
1000
900
800
700
600
500
0
25
50
75
100
125
150
175
200
history size

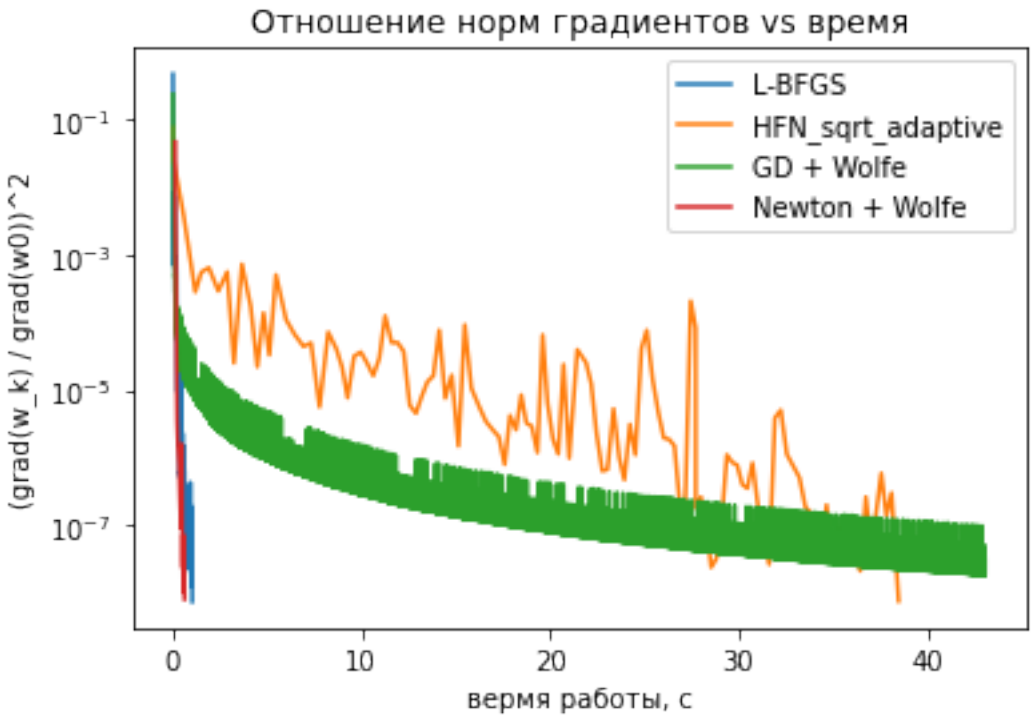
[10]:
```
true_val = sc.optimize.minimize(lambda w: orac.value(w), w0).fun
true_val
```

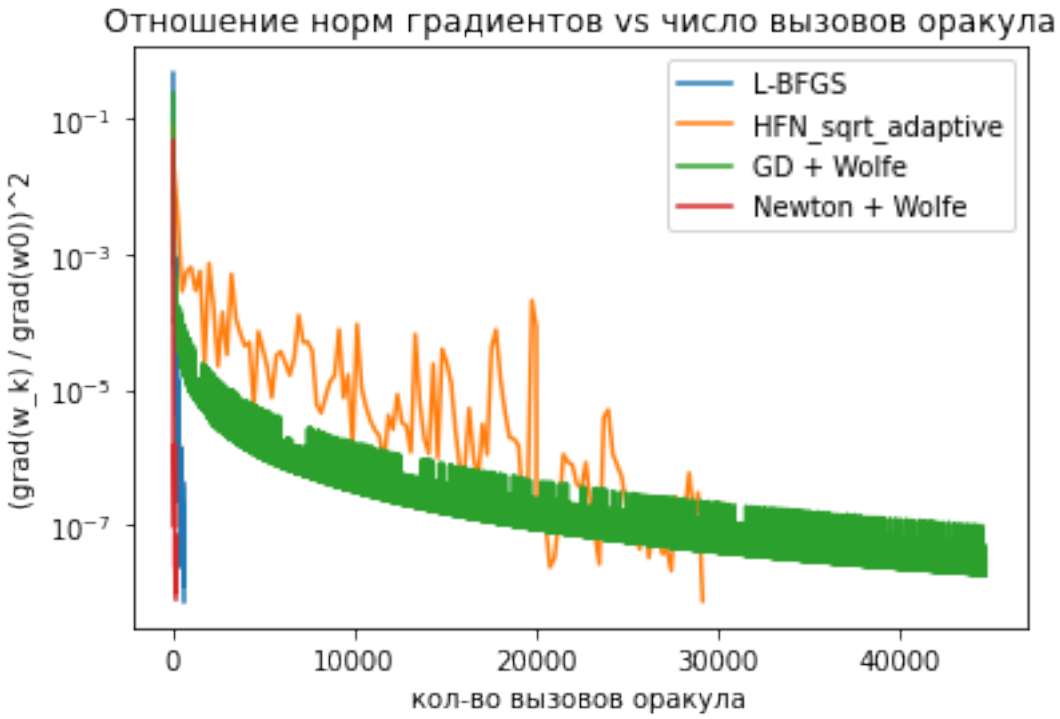
[10]:
```
0.29787605181219556
```

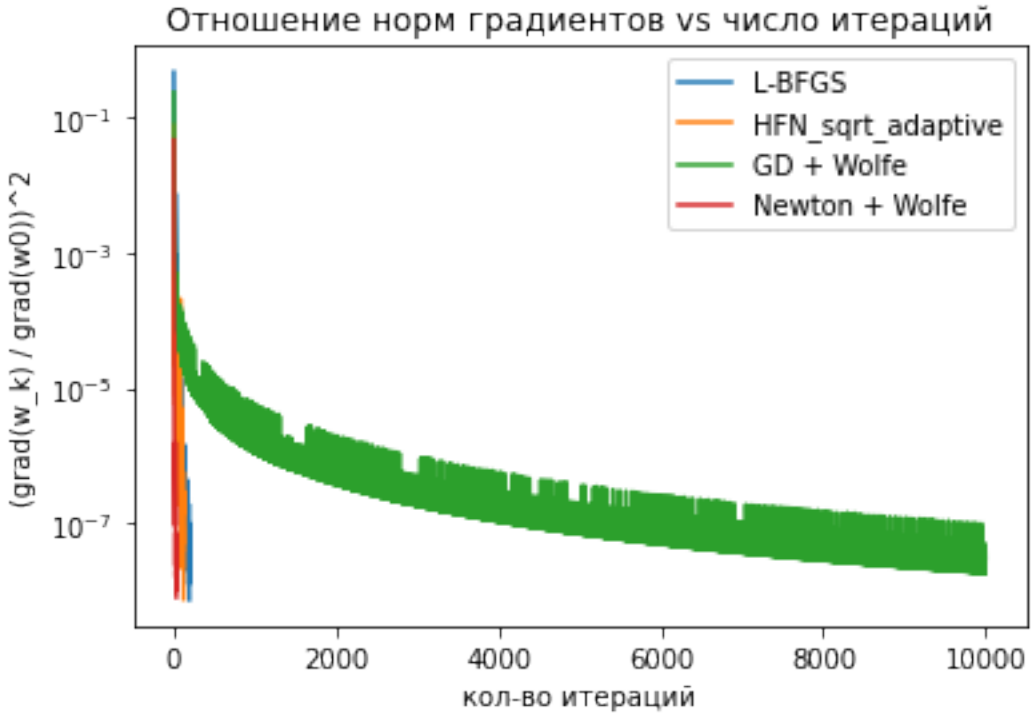
[13]:
```
def plotter(oracle, w0, true_val):
    optimizer = OptimizeHFN()

    rel_errs = []
    vals = []
    times = []
    oracle_calls = []
    n_its = []

    labels = ['L-BFGS', 'HFN_sqrt_adaptive', 'GD + Wolfe', 'Newton + Wolfe']

    n = len(labels)

    optimizer = OptimizeLBFGS(l=15)
    point = optimizer(oracle, w0)
    rel_errs.append(optimizer.rel_errs)
    vals.append(optimizer.values)
    times.append(optimizer.times)
    oracle_calls.append(optimizer.orac_calls)
    n_its.append(optimizer.n_iter)

    optimizer = OptimizeHFN()
    point = optimizer(oracle, w0, 'sqrt_adaptive')
    rel_errs.append(optimizer.rel_errs)
    vals.append(optimizer.values)
    times.append(optimizer.times)
    oracle_calls.append(optimizer.orac_calls)
    n_its.append(optimizer.n_iter)

    optimizer = OptimizeGD()
    point = optimizer(oracle, w0, WolfeLineSearch())
    rel_errs.append(optimizer.rel_errs)
    vals.append(optimizer.values)
    times.append(optimizer.times)
    oracle_calls.append(optimizer.orac_calls)
    n_its.append(optimizer.n_iter)

    optimizer = OptimizeNewton()
    point = optimizer(oracle, w0, WolfeLineSearch())
    rel_errs.append(optimizer.rel_errs)
    vals.append(optimizer.values)
    times.append(optimizer.times)
```

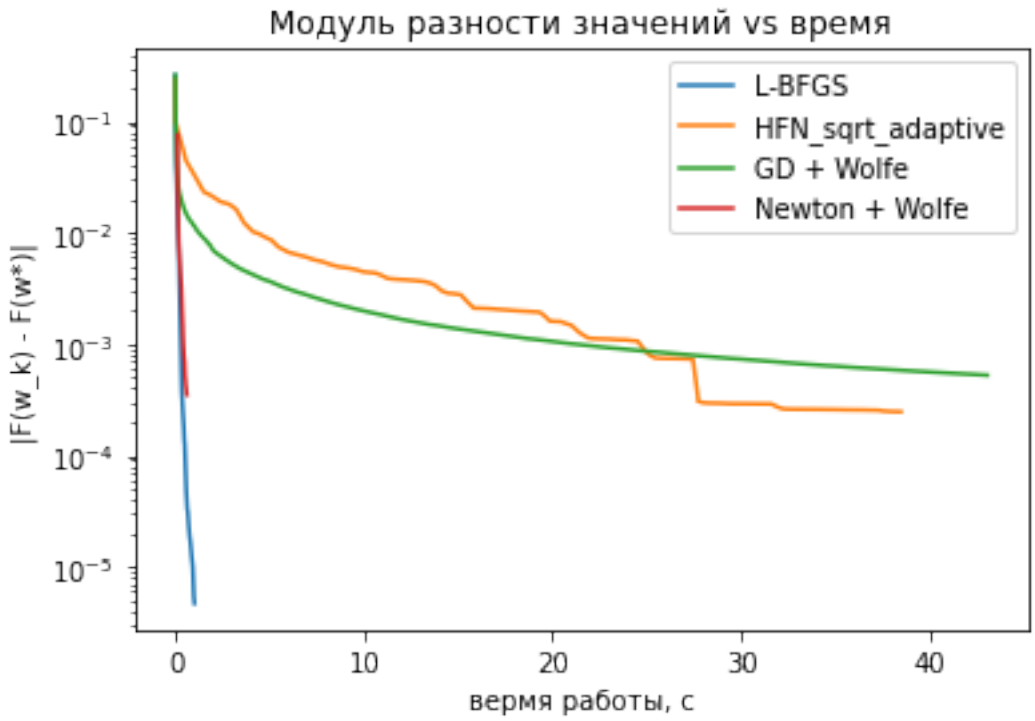
```
    oracle_calls.append(optimizer.orac_calls)
    n_its.append(optimizer.n_iter)

    plt.title('               vs    ')
    for i in range(n):
        plt.plot(times[i], rel_errs[i], label=labels[i])
    plt.legend()
    plt.yscale('log')
    plt.xlabel('       ,  ')
    plt.ylabel('(grad(w_k) / grad(w0))^2')
    plt.show()

    plt.title('               vs             ')
    for i in range(n):
        plt.plot(oracle_calls[i], rel_errs[i], label=labels[i])
    plt.legend()
    plt.yscale('log')
    plt.xlabel('  -           ')
    plt.ylabel('(grad(w_k) / grad(w0))^2')
    plt.show()

    plt.title('               vs         ')
    for i in range(n):
        plt.plot(list(range(1, n_its[i] + 1)), rel_errs[i], label=labels[i])
    plt.legend()
    plt.yscale('log')
    plt.xlabel('  -       ')
    plt.ylabel('(grad(w_k) / grad(w0))^2')
    plt.show()

    plt.title('               vs    ')
    for i in range(n):
        plt.plot(times[i], abs(np.array(vals[i]) - true_val), label=labels[i])
    plt.legend()
    plt.yscale('log')
    plt.xlabel('       ,  ')
    plt.ylabel('|F(w_k) - F(w*)|')
    plt.show()

    plt.title('               vs             ')
    for i in range(n):
        plt.plot(oracle_calls[i], abs(np.array(vals[i]) - true_val),␣
↪label=labels[i])
    plt.legend()
    plt.yscale('log')
    plt.xlabel('  -           ')
    plt.ylabel('|F(w_k) - F(w*)|')
```

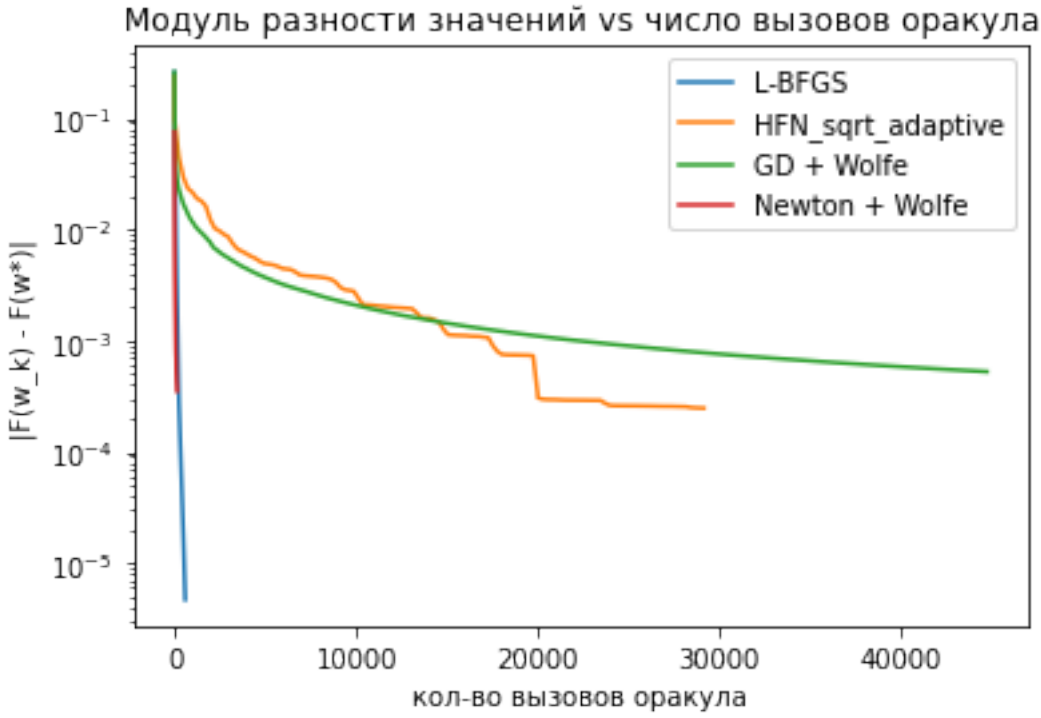
```
    plt.show()

    plt.title('               vs          ')
    for i in range(n):
        plt.plot(list(range(1, n_its[i] + 1)), abs(np.array(vals[i]) -␣
↪true_val), label=labels[i])
    plt.legend()
    plt.yscale('log')
    plt.xlabel('  -       ')
    plt.ylabel('|F(w_k) - F(w*)|')
    plt.show()

plotter(orac, w0, true_val)
```

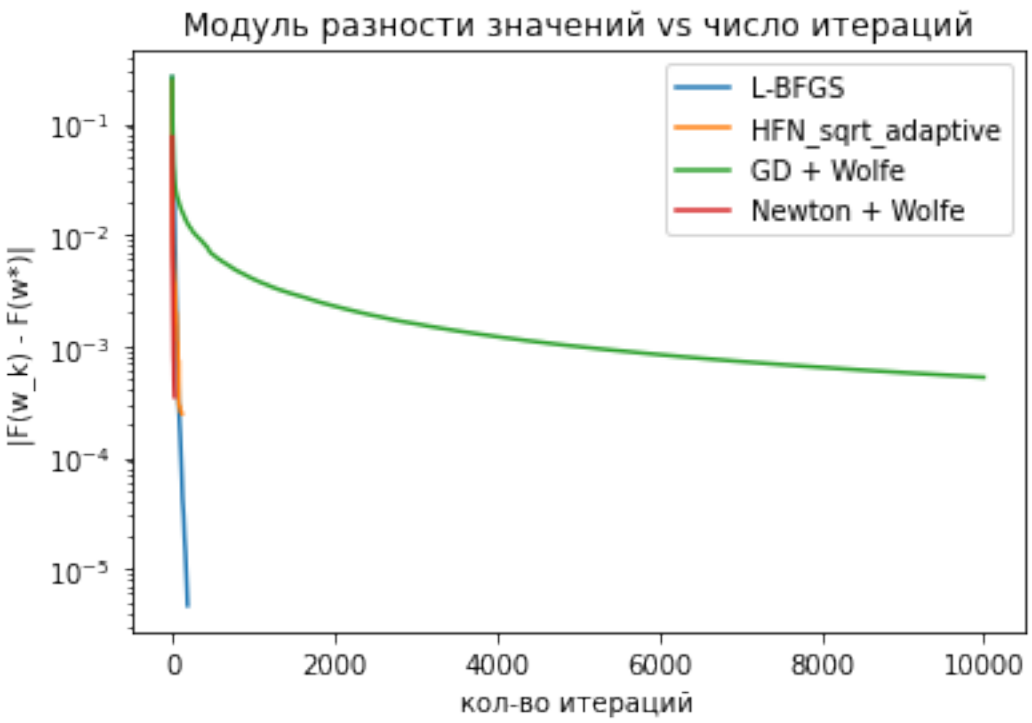



Отношение норм градиентов vs число вызовов оракула
L-BFGS
HFN_sqrt_adaptive
GD + Wolfe
Newton + Wolfe
(grad(w_k) / grad(w0))^2
кол-во вызовов оракула


Отношение норм градиентов vs число итераций
L-BFGS
HFN_sqrt_adaptive
GD + Wolfe
Newton + Wolfe
(grad(w_k) / grad(w0))^2
кол-во итераций


Модуль разности значений vs время
L-BFGS
HFN_sqrt_adaptive
GD + Wolfe
Newton + Wolfe
|F(w_k) - F(w*)|
вермя работы, с


Модуль разности значений vs число вызовов оракула
L-BFGS
HFN_sqrt_adaptive
GD + Wolfe
Newton + Wolfe
|F(w_k) - F(w*)|
кол-во вызовов оракула


Модуль разности значений vs число итераций
L-BFGS
HFN_sqrt_adaptive
GD + Wolfe
Newton + Wolfe
|F(w_k) - F(w*)|
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
0
2000
4000
6000
8000
10000
кол-во итераций

,