Matheus Diógenes Andrade

228117

Exercício 3

In [1]:
```
from scipy.optimize import minimize, rosen, rosen_der
from scipy.optimize import line_search, minimize_scalar
from sympy import symbols, diff
import numpy as np
import time
import pybobyqa

import matplotlib.pyplot as plt
#import autograd.numpy as np
from mpl_toolkits.mplot3d import Axes3D
from matplotlib.colors import LogNorm
from autograd import elementwise_grad, value_and_grad
from scipy.optimize import minimize
from collections import defaultdict
from itertools import zip_longest
from functools import partial
```

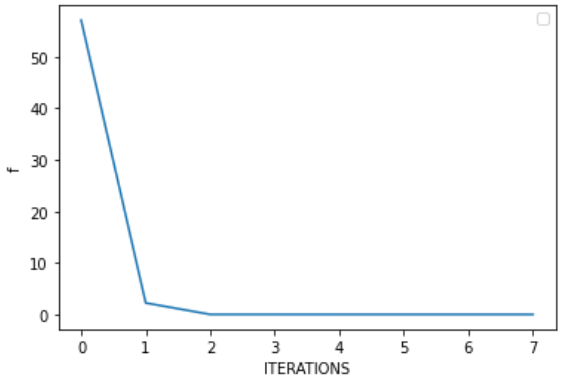
In [2]:
```
# method info
class MethodInfo:
    def __init__ (self, bestSol, time, nFCalls, nGradientCalls):
        self.bestSol = bestSol
        self.time = time
        self.nFCalls = nFCalls
        self.nGradientCalls = nGradientCalls
# params
INIT_SOL = np.array([4., 4.])
xmin, xmax, xstep = -4., 4., .02
ymin, ymax, ystep = -4., 4., .02
# counters
NUM_OBJ_FUN_CALLS = 0
NUM_GRAD_CALLS = 0
PATH = []
# scipy callback
scipy_callback = lambda xk : PATH.append(xk)
# obj. function
def f(x):
    global NUM_OBJ_FUN_CALLS
    NUM_OBJ_FUN_CALLS += 1
    return (x[0]**2 + x[1] - 11)**2 + (x[0] + x[1]**2 - 7)**2
# partial derivatives
pd_x1 = lambda x : 4 * x[0] * (x[0]**2 + x[1] - 11) + 2 * x[0] + 2 * x[1]**2 - 14
pd_x2 = lambda x : 2 * x[0]**2 + 4 * x[1] * (x[0] + x[1]**2 - 7) + 2 * x[1] - 22

def pd_x(x):
    global NUM_GRAD_CALLS
    NUM_GRAD_CALLS = NUM_GRAD_CALLS + 1
    return np.array([pd_x1(x), pd_x2(x)])
def plot_graph():
    # graph
    plt.xlabel('ITERATIONS')
    plt.ylabel('f')
    plt.plot(range(len(PATH)), [f(path) for path in PATH])
    plt.legend()
    plt.show()
#GD
def gradient_descent_line_search(VERBOSE = False):
    global PATH
    iter = 1
    x_old = INIT_SOL
    x_new = None
    PATH = [x_old]
    while (iter <= 50000):
        grad = - pd_x(x_old)
        res = line_search(f, pd_x, x_old, grad)
        x_new = x_old + res[0] * grad
        PATH.append(x_new)
        if VERBOSE:
            print("ITER {}\n{} {}".format(iter, x_old, x_new))
        if np.linalg.norm(x_old-x_new)/np.linalg.norm(x_new) < 1e-5:
            if VERBOSE:
```

```
                print("TOLERANCE < ", 1e-5)
            break
        iter += 1
        x_old = x_new
    return x_old
```

1. Conjugado gradiente

In [3]:
```
NUM_OBJ_FUN_CALLS = 0
NUM_GRAD_CALLS = 0
PATH = []
start_time = time.time()
res = minimize(f, INIT_SOL, method='CG', jac = pd_x, callback = scipy_callback)
incurr_time = time.time() - start_time
conjugatedGradient = MethodInfo([res.x[0], res.x[1]], incurr_time, NUM_OBJ_FUN_CALLS, NUM_GRAD_CALLS)
print("o ponto de mínimo e o valor da função neste ponto f({}, {}) = {}".format(conjugatedGradient.bestSol[0],
                                                                                 conjugatedGradient.bestSol[1],
                                                                                 f(conjugatedGradient.bestSol)))
print("tempo total de execução: {}s".format(conjugatedGradient.time))
#print("n chamadas para a função: {}".format(res.nfev))
print("n chamadas para a função: {}".format(conjugatedGradient.nFCalls))
#print("o número de chamadas para o gradiente: {}".format(res.njev))
print("o número de chamadas para o gradiente: {}".format(conjugatedGradient.nGradientCalls))
#res
plot_graph()
```

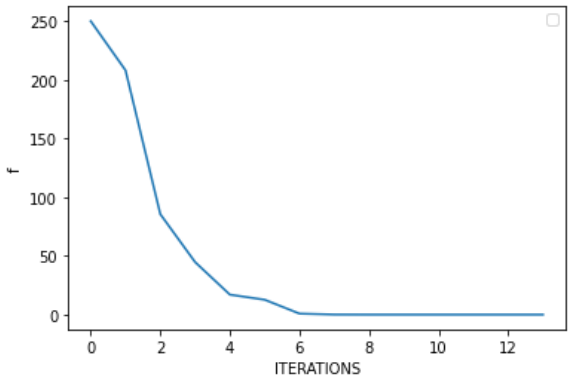
```
No handles with labels found to put in legend.
o ponto de mínimo e o valor da função neste ponto f(2.999999999989263, 1.9999999999893328) = 8.490750396096654e
-21
tempo total de execução: 0.009206295013427734s
n chamadas para a função: 17
o número de chamadas para o gradiente: 17
```



2. Descida do gradiente com busca em linha

In [4]:
```
NUM_OBJ_FUN_CALLS = 0
NUM_GRAD_CALLS = 0
start_time = time.time()
res = gradient_descent_line_search()
incurr_time = time.time() - start_time
lineSearch = MethodInfo([res[0], res[1]], incurr_time, NUM_OBJ_FUN_CALLS, NUM_GRAD_CALLS)
print("o ponto de mínimo e o valor da função neste ponto f({}, {}) = {}".format(lineSearch.bestSol[0],
                                                                                 lineSearch.bestSol[1],
                                                                                 f(lineSearch.bestSol)))
print("tempo total de execução: {}s".format(lineSearch.time))
#print("n chamadas para a função: {}".format(res.nfev))
print("n chamadas para a função: {}".format(lineSearch.nFCalls))
#print("o número de chamadas para o gradiente: {}".format(res.njev))
print("o número de chamadas para o gradiente: {}".format(lineSearch.nGradientCalls))
#res
plot_graph()
```

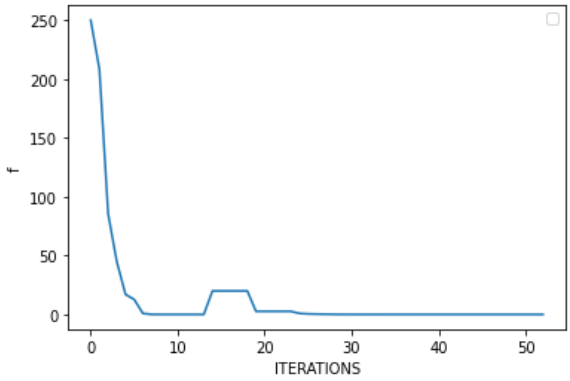
```
No handles with labels found to put in legend.
o ponto de mínimo e o valor da função neste ponto f(-3.7793170798233486, -3.2831674795196806) = 2.1397202414115
147e-08
tempo total de execução: 0.003698110580444336s
n chamadas para a função: 89
o número de chamadas para o gradiente: 39
```



3. Nelder-Mead

In [5]:
```
NUM_OBJ_FUN_CALLS = 0
NUM_GRAD_CALLS = 0
start_time = time.time()
res = minimize(f, INIT_SOL, method='Nelder-Mead', jac = pd_x, callback = scipy_callback,
               options={'initial_simplex': [[-4,-4], [-4,1], [4,-1]]})
incurr_time = time.time() - start_time
nelderMead = MethodInfo([res.x[0], res.x[1]], incurr_time, NUM_OBJ_FUN_CALLS, NUM_GRAD_CALLS)
print("o ponto de mínimo e o valor da função neste ponto f({}, {}) = {}".format(nelderMead.bestSol[0],
                                                                                 nelderMead.bestSol[1],
                                                                                 f(nelderMead.bestSol)))
print("tempo total de execução: {}s".format(nelderMead.time))
#print("n chamadas para a função: {}".format(res.nfev))
print("n chamadas para a função: {}".format(nelderMead.nFCalls))
#print("o número de chamadas para o gradiente: {}".format(res.njev))
print("o número de chamadas para o gradiente: {}".format(nelderMead.nGradientCalls))
#res
plot_graph()
```

```
/home/matheusdiogenesandrade/.local/lib/python3.6/site-packages/scipy/optimize/_minimize.py:518: RuntimeWarnin
g: Method Nelder-Mead does not use gradient information (jac).
  RuntimeWarning)
No handles with labels found to put in legend.
o ponto de mínimo e o valor da função neste ponto f(3.5844144912564206, -1.8481158805325921) = 1.06865669961686
41e-08
tempo total de execução: 0.20351839065551758s
n chamadas para a função: 77
o número de chamadas para o gradiente: 0
```



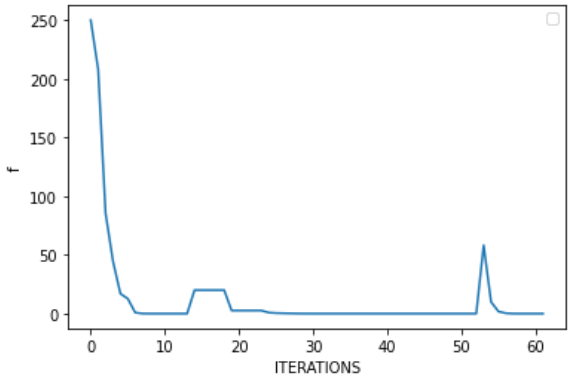
4. BFGS

4.1. BFGS com a função do gradiente

In [6]:
```
NUM_OBJ_FUN_CALLS = 0
NUM_GRAD_CALLS = 0
start_time = time.time()
res = minimize(f, INIT_SOL, method='L-BFGS-B', jac = pd_x, callback = scipy_callback)
incurr_time = time.time() - start_time
bfgsWithGradient = MethodInfo([res.x[0], res.x[1]], incurr_time, NUM_OBJ_FUN_CALLS, NUM_GRAD_CALLS)
print("o ponto de mínimo e o valor da função neste ponto f({}, {}) = {}".format(bfgsWithGradient.bestSol[0],
                                                                                 bfgsWithGradient.bestSol[1],
                                                                                 f(bfgsWithGradient.bestSol)))

print("tempo total de execução: {}s".format(bfgsWithGradient.time))
#print("n chamadas para a função: {}".format(res.nfev))
print("n chamadas para a função: {}".format(bfgsWithGradient.nFCalls))
```

```
#print("o número de chamadas para o gradiente: {}".format(res.njev))
print("o número de chamadas para o gradiente: {}".format(bfgsWithGradient.nGradientCalls))
#res
plot_graph()
```

```
No handles with labels found to put in legend.
o ponto de mínimo e o valor da função neste ponto f(2.999999857067003, 2.0000001906373814) = 8.287611020495648e
-13
tempo total de execução: 0.048629045486450195s
n chamadas para a função: 10
o número de chamadas para o gradiente: 10
```

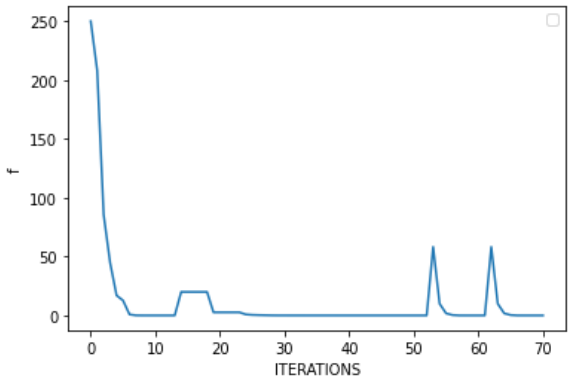


4.1. BFGS sem a função do gradiente

In [7]:

```
NUM_OBJ_FUN_CALLS = 0
start_time = time.time()
res = minimize(f, INIT_SOL, method='L-BFGS-B', callback = scipy_callback)
incurr_time = time.time() - start_time
bfgsWithoutGradient = MethodInfo([res.x[0], res.x[1]], incurr_time, NUM_OBJ_FUN_CALLS, NUM_GRAD_CALLS)
print("o ponto de mínimo e o valor da função neste ponto f({}, {}) = {}".format(bfgsWithoutGradient.bestSol[0],
                                                                                 bfgsWithoutGradient.bestSol[1],
                                                                                 f(bfgsWithoutGradient.bestSol)))
print("tempo total de execução: {}s".format(bfgsWithoutGradient.time))
#print("n chamadas para a função: {}".format(res.nfev))
print("n chamadas para a função: {}".format(bfgsWithoutGradient.nFCalls))
#print("o número de chamadas para o gradiente: {}".format(res.njev))
print("o número de chamadas para o gradiente: {}".format(bfgsWithoutGradient.nGradientCalls))
#res
plot_graph()
```

```
No handles with labels found to put in legend.
o ponto de mínimo e o valor da função neste ponto f(2.999999852725913, 2.000000188192229) = 8.502778926721376e-
13
tempo total de execução: 0.012542486190795898s
n chamadas para a função: 30
o número de chamadas para o gradiente: 10
```



5. BOBYQA

In [8]:

```
NUM_OBJ_FUN_CALLS = 0
res = pybobyqa.solve(f, INIT_SOL)
bobyqa = MethodInfo([res.x[0], res.x[1]], incurr_time, NUM_OBJ_FUN_CALLS, 0)
print("o ponto de mínimo e o valor da função neste ponto f({}, {}) = {}".format(bobyqa.bestSol[0],
                                                                                 bobyqa.bestSol[1],
                                                                                 f(bobyqa.bestSol)))
print("tempo total de execução: {}s".format(bobyqa.time))
print("n chamadas para a função: {}".format(bobyqa.nFCalls))
```

```
o ponto de mínimo e o valor da função neste ponto f(2.9999999999976628, 2.0000000000094835) = 1.287703554675167
e-21
tempo total de execução: 0.012542486190795898s
n chamadas para a função: 58
```

	Conjugated GD	Line Search	Nelder-mead	BFGS with gradient	BFGS without gradient	BOBYQA
BestSol	[2.99, 1.99]	[2.99, 1.99]	[3.58, -1.84]	[2.99, 2.00]	[2.99, 2.00]	[2.99, 2.00]
BestSolCost	8.4907503e-21	8.490750e-21	1.0686566e-08	8.287611020495e-13	8.502778926721376e-13	1.287703554675167e-2
Time	0.00831770896	0.0248918533	0.01173400878	0.0046205520629882	0.009778738021850586	0.009778738021850586
NObjFunCalls	17	212	77	10	30	58
NGradientCalls	17	191	0	10	0	0