Homework 1

August 31, 2021

Solve the following problem using Python SciPy.optimize. Please attach your code and results. Specify your initial guesses of the solution. If you change your initial guess, do you find different solutions? (**100 points**)

$$\begin{aligned}
\text{minimize:} \quad & (x_1 - x_2)^2 + (x_2 + x_3 - 2)^2 + (x_4 - 1)^2 + (x_5 - 1)^2 \\
\text{subject to:} \quad & x_1 + 3x_2 = 0 \\
& x_3 + x_4 - 2x_5 = 0 \\
& x_2 - x_5 = 0 \\
& -10 \leq x_i \leq 10, \ = 1, \ldots, 5
\end{aligned}$$

Note:

1. Please learn how to use **break points** to debug. **I will not address your programming questions if you have not learned how to debug your code.**
2. I recommend PyCharm as the IDE. If you are new to Python, you can also start with Google Colab without installing anything.
3. If you are on Windows, the Anaconda version of Python 3 is highly recommended.

Here are the steps to push a homework submission:

1. Clone the course repo: First click on **Code** to get the Git address (e.g., the HTTPS address). Then use your IDE to clone (download) the repo using this address. PyCharm tutorial on using Git.
2. You will find the homework in the **Homework** folder.
3. For analytical problems (e.g., proofs and calculations), please use Markdown to type up your answers. Markdown Math. For Latex users, you can convert tex to markdown using Pandoc.
4. For coding problems, please submit a Jupyter Notebook file with your code and final results. You can also add a URL to your Jupyter or Colab Notebook in README.md if you use online notebooks.

5. For each homework, please submit a single notebook file (or link) that combines the markdown solutions, the codes, and the computation results, and name the file according to the homework.

6. **IMPORTANT** Please push (upload) the notebook file every time you work on the homework and add comments when you push, e.g., "finished problem 1, still debugging problem 2". This way I know you worked on your own.

[1]:
```
import scipy.optimize as opt
fun = lambda x: (x[0] - x[1])**2 + (x[1] + x[2] - 2)**2 + (x[3] - 1)**2 + (x[4]␣
 ↪- 1)**2
cons = (
    {'type': 'eq', 'fun': lambda x: x[0] + 3*x[1]},
    {'type': 'eq', 'fun': lambda x: x[2] + x[3] - 2*x[4]},
    {'type': 'eq', 'fun': lambda x: x[1] - x[4]}
)
bnds = ((-10,10), (-10,10), (-10,10), (-10,10), (-10,10))

res = opt.minimize(fun, (1,1,1,1,1), method = 'SLSQP', bounds = bnds,␣
 ↪constraints = cons)
res
```

[1]:
```
     fun: 4.09302326452976
     jac: array([-2.04664832, -0.18578869, -2.23243701, -2.23257673,
-1.48833793])
 message: 'Optimization terminated successfully'
    nfev: 38
     nit: 6
    njev: 6
  status: 0
 success: True
       x: array([-0.76749312,  0.25583104,  0.62795044, -0.11628835,
0.25583104])
```

[2]:
```
# Changing the initial guess
x_init = (2,3,1,4,5)
fun = lambda x: (x[0] - x[1])**2 + (x[1] + x[2] - 2)**2 + (x[3] - 1)**2 + (x[4]␣
 ↪- 1)**2
cons = (
    {'type': 'eq', 'fun': lambda x: x[0] + 3*x[1]},
    {'type': 'eq', 'fun': lambda x: x[2] + x[3] - 2*x[4]},
    {'type': 'eq', 'fun': lambda x: x[1] - x[4]}
)
bnds = ((-10,10), (-10,10), (-10,10), (-10,10), (-10,10))

res = opt.minimize(fun,x0 = x_init, method = 'SLSQP', bounds = bnds,␣
 ↪constraints = cons)
res
```

[2]:
```
      fun: 4.093023288479049
      jac: array([-2.04646897, -0.18635827, -2.23282731, -2.2323209 , -1.4883827
])
  message: 'Optimization terminated successfully'
     nfev: 43
      nit: 7
     njev: 7
   status: 0
  success: True
        x: array([-0.76742588,  0.25580863,  0.62777771, -0.11616046,
0.25580863])
```

Changing the initial guess does not change the minimum