Lab 10: Optimization

10.1 Introduction

This lab focuses on solving for the coefficients of nonlinear fits and on finding extrema of 1D and 2D functions. The common thread here is using the `optimize` module to let Python solve problems that require function minimization.

For some of the problems you will be presenting graphical representations of the models as well as calculating statistical information to quantify the goodness of fit. For some of the problems, you will also be making predictions based on the models. Each problem has a component on the Connect system - typically you will be entering coefficients, statistical values, extrema locations, or extrema values. For several of the problems, you will also be required to generate a graph of the original data points and the model. These graphs should be saved as .png files and then imported to a PDF document to upload to Sakai along with the scripts.

10.2 Getting Started

Edit: To access the Box folder for this lab, you must complete the Sakai "test" for Lab 10/11 to confirm your understanding that everything in this lab is to be done ***individually***. The PDF and starter files are in the Box folder whose link you will get once you have confirmed your understanding of the nature of the assignment. The rest of the process is the same as he have used for the semester and there is a support page at:

https://pundit.pratt.duke.edu/wiki/EGR_103/Fall_2021/Lab_10

10.3 Assignment

10.3.1 Chapra 15.11, p. 400

Script name: `chapra_15_011.py`
Graph name: `chapra_15_011_plot.png`

Use *nonlinear regression* to solve for the maximum photosynthesis rate P_m and the optimal solar radiation I_{sat}. Note that you will need to be sure to give the optimization routine a good starting guess for these values to obtain realistic results. Once you have the coefficients and print them out, make a plot with the original data as red diamonds and the model equation as a green line. Include axis labels with units and a grid; no need for estimates, a title, or a legend. Determine the S_t, S_r, and r^2 values and include these in your lab report. Note that the values should show that this is a very good mathematical fit!

Coding hints: Start with the nonlinear regression example and make sure you understand how it works. Carefully note the *names* of the variables you are trying to find and figure out how you might make a reasonable first-guess as to what each might be. *Note:* `[0, 0]` is a very, very poor guess.[1] You should get a very good r^2 value if you choose good initial values. In the lab report, describe what values you used for your initial guesses and why.

10.3.2 Chapra 15.22

Script name: `chapra_15_022.py`
Graph name: `chapra_15_022_plot.png`

Use *nonlinear regression* to solve for the a and b values. Once you have the coefficients and print them out, make a plot with the original data as purple squares and the model equation as a yellow line. Include axis labels and a grid; no need for estimates, a title, or a legend. Determine an estimate of y at the requested x value on Connect. Also determine the S_t, S_r, and r^2 values and include these in your lab report. Note that the values should show that this is a very good mathematical fit!

[1] `[1, 1]` isn't so hot either.

Copyright 2021, Gustafson et al.

10.3.3 Chapra 15.29

Script name: `chapra_15_029.py`
Graph name: `chapra_15_029_plot.png`

Use *nonlinear regression* to solve for the A, and B, and C values. *Note:* use `x=T` for your independent value and `y=np.log(p)` for your dependent value. Once you have the coefficients and print them out, make a plot of $\ln(p)$ as a function of T in Kelvin with the original data as cyan hexagons and the model equation as a dashed black line. Include axis labels and a grid; no need for estimates, a title, or a legend. Also determine the S_t, S_r, and r^2 values and include these in your lab report. You will also need to report the standard error – since this model has three coefficients, note that:

$$s_{y/x} = \sqrt{\frac{S_r}{n-3}}$$

Hints: (1) This is a really good model, (2) the graph has a domain of about 50 to about 130, (3) the graph has a range of about 4 to about 15, and (4) the model is increasing at a decreasing rate (concave down).

10.3.4 Chapra 7.16, p. 216

Script name: `chapra_7_016.py`
Graph name: `chapra_7_016_plot.png`

First make a plot of the temperature as a function of the insulation thickness for thicknesses between 1 mm and 12 mm. The main reason for making the graph is to get an idea of the insulation thicknesses that can be used as boundaries to find the minimum temperature. Be sure to properly label and title your graph and use a grid. Then write the code needed to find and print the insulation thickness that yields the minimum temperature for the wire. Solve for the value twice in your code - once using `fminbound` and boundary values and again using `fmin` and an estimate of 10 mm. Note that the two methods may not come up with the exact same value but they should be close enough to each other and to the Connect answer to work. Report both in your lab and include your graph.

10.3.5 Chapra 7.25, 7.26, 7.27 (b), p. 217

Script name: `extremes.py`
Graph Names: `chapra_7_025_plot.png`, `chapra_7_026_plot.png`, `chapra_7_027_plot.png`

You can put the code for all three of these problems in a single script file. Use the `fmin` function in `scipy.optimize` to find the extreme value asked for. To help get you started, here is some code that finds the actual minimum for the function given in Chapra Example 7.4:

```
def f(x, y):
    return 2 + x - y + 2*x**2 + 2*x*y + y**2

min_loc = opt.fmin(lambda vec: f(vec[0], vec[1]), [0, 0])

min_val = f(*min_loc)
```

Note that `fmin` searches based on changing the entries of a single *vector* - that is why the x and y values in the `fmin` command are given as entries within the `vec` vector. The initial condition also must have two entries - an initial guess for x and an initial guess for y. Make a wireframe plot of each equation using a domain that goes from -4 to 4 in each direction with 20 points in each direction. Be sure to properly label and title your plot and include the plots as well as the locations and values of the extrema in your lab report.

Copyright Fall 2021, Gustafson et al.

10.3.6 Chapra 7.31, p. 218

Script name: `chapra_7_031.py`
Graph name: `chapra_7_031_plot.png`

First make a plot of the dissolved oxygen concentration as a function of the time in days for 0 to 20 days. The main reason for making the graph is to get an idea of the times that can be used as boundaries to find the minimum oxygen concentration. Be sure to properly label and title your graph and use a grid. Then write the code needed to find and print time in days that yields the minimum oxygen concentration for the river. Solve for the value twice in your code - once using `fminbound` and boundary values and again using `fmin` and an estimate of 4 days. Note that the two methods may not come up with the exact same value but they should be close enough to each other and to the Connect answer to work. Report both values in your lab.

10.3.7 Chapra 7.34, p. 219

Script name: `chapra_7_034.py`

The plot has already been made for you in Figure P7.34. Write the code needed to find and print the slip value that yields the maximum torque for the motor. Solve for the value twice in your code - once using `fminbound` and boundary values and again using `fmin` and an estimate of 2 for the slip. Note that the two methods may not come up with the exact same value but they should be close enough to each other and to the Connect answer to work. Report both values in your lab.

10.3.8 Chapra 7.36, p. 219

Script name: `chapra_7_036.py`

You are not required to make a graph for this one. Assume that the value of x is between 0 and 4. Write the code needed to find and print the value of x where the maximum force occurs. Solve for the value twice in your code - once using `fminbound` and boundary values and again using `fmin` and an estimate of 2 for the location. Note that the two methods may not come up with the exact same value but they should be close enough to each other and to the Connect answer to work. Report both values in your lab.

Copyright Fall 2021, Gustafson et al.

Lab 11: Finding Roots

11.1 Introduction

This (mini)lab focuses on finding roots of equations. For some of the problems you will be presenting graphical representations of the models as well as calculating roots. Note that in some cases the Chapra book may specify a method for finding a root - you will generally ignore that and instead use either `brentq` or `root`.

Each problem has a component on the Connect system - typically you will be entering a root. For some of the problems, you will also be required to generate a graph. These graphs should be saved as .png files and they will be uploaded to a Sakai assignment along with the scripts. Each problem will specify names for the scripts and for the graphs - please be sure to use these names to assist with compiling your PDF!

11.2 Getting Started

Edit: To access the Box folder for this lab, you must complete the Sakai "test" for Lab 10/11 to confirm your understanding that everything in this lab is to be done ***individually***. The PDF and starter files are in the Box folder whose link you will get once you have confirmed your understanding of the nature of the assignment. The rest of the process is the same as he have used for the semester and there is a support page at:

https://pundit.pratt.duke.edu/wiki/EGR_103/Fall_2021/Lab_11

11.3 Assignment

11.3.1 Chapra 6.13, p. 191

Script name: `chapra_6_013.py`
Graph name: `chapra_6_013_plot.png`

The main reason for making the graph is to get an idea of temperatures that can be used to bracket the root. Remember that to make this a root finding problem, you need to have some equation you want to set equal to 0. Use `brentq` here. For the graph, use a solid line of whatever color you would like and be sure to include axis labels and a title.

11.3.2 Chapra 6.14, p. 191

Script name: `chapra_6_014.py`
Graph name: `chapra_6_014_plot.png`

Make a plot of K as a function of x with p_t equal to the value specified on Connect. The x values should start at 0 and end where the K values shown are negative for about half of the graph (try a maximum x of 0.05 to start. Be sure your graph has proper labels and a title. Then use the graph to determine good brackets for the root and use `brentq` to solve.

11.3.3 Chapra 6.16, p. 192

Script name: `chapra_6_016.py`

Think about what a reasonable set of bracketing values will look like. You are finding the height of fluid in a cylinder that is on its side - what is the physically possible range of values for that height of fluid?

11.3.4 Chapra 6.33, p. 195

Script name: `chapra_6_033.py`

Use `root` for this rather than programming Newton Raphson. Be sure to give a reasonable initial guess.

Copyright 2021, Gustafson et al.