Assignment #1

Due Wednesday, 7 September 2022, at the start of class

Make sure you have a copy of the textbook:

> Griva, Nash, and Sofer, *Linear and Nonlinear Optimization*, 2nd ed., SIAM Press 2009.

Please read Chapter 1; note that Section 1.7 can be read lightly. Also read Appendix B.4 and Sections 2.1–2.4.

DO THE FOLLOWING EXERCISES from page 47 of the textbook:

- Exercise 2.1
- Exercise 2.3
- Exercise 2.4
- Exercise 2.5 (*Also specify the feasible set for your example.*)

DO THE FOLLOWING PROBLEMS. These problems are based on the *5 example optimization problems* notes, handed-out in class and posted at

`bueler.github.io/opt/assets/examples/fiveexamples.pdf`

Do *not* use optimization-related black boxes, such as the MATLAB commands `fzero`, `fsolve`, `fminsearch`, or `fminbnd`, for these, or any other, exercises or problems, unless this is specifically requested.

Problem P1. Solve `calcone` by implementing a strategy for solving this and similar one-variable optimization problems on bounded, closed intervals. (*There are many correct answers, i.e. strategies. Choose one and implement and understand it.*) Describe the strategy in brief and well-written english. Your strategy will necessarily be iterative, and will not get the exact answer but only an approximate one. Discuss any issues about the general performance/success of your strategy, emphasizing how it might fail on problems of this type.[1] Implement the strategy as a MATLAB or other code, using elementary programming structures such as variables, arrays, `for` loops, `if` conditionals, and such. Argue that your answer to this particular problem is given to at least 6-digit accuracy, and, along the way, visualize the function $f(x)$.

Note that your solution strategy for `calcone` should *not* be based on human interaction with a figure window, e.g. repeated zooming into a figure. This is because higher-dimensional problems are un-visualizable by humans. Programs must run autonomously to be useful.

[1] Note that every numerical procedure can be made to fail by careful input (i.e. problem) design. In optimization, professionals know how to break what they build.

Problem P2. Solve `fit`.

Follow essentially the same rules as above: Describe a strategy (algorithm) for solving this and similar problems. Discuss any issues about the general performance/success of your strategy. Implement your strategy as a MATLAB code using elementary programming. Demonstrate 6 digit accuracy. Plot the solution curve on the same graph as the data.

Please try to *avoid* copying formulas from books or online. *Avoid* recipes you do not understand. Though problems like `fit` are standard in the statistics and linear algebra courses, I want you to start from scratch and understand what you are doing.

Problem P3. Solve `salmon`.

In fact this problem is embarassingly simple to solve, so start by writing a few clear sentences stating and justifying the solution. Then visualize, in 3D and probably with pencil and paper, the set of feasible solutions; mark and label the solution on this plot.

Also use a straightforward substitution to eliminate the equality constraint, and then re-visualize the feasible set and solution in 2D.

Is this problem discrete? Is it fair to interpret it as continuous? Comment.

Problem P4. Complete the following classification table for the example problems:

name	discrete	constrained	linear	quadratic	dimension
`calcone`					
`fit`					
`salmon`					
`tsp`					
`glacier`					

Directions. Except for the last column, use a check (✓) if the property is true, leave blank if it is not, or write "NA" for not applicable. In the last column give an integer for the dimension, or ∞, or "NA".

Regarding the "linear" and "quadratic" columns, first check the form of the objective function; is it linear or quadratic or neither? Next note that an optimization problem is called linear or quadratic if *both* the objective function and the constraints have that property. Finally, note that the class of quadratic functions has linear functions as a subset.