Assignment 1 Report

AEROSP 588 – Dr. Joaquim Martins

University of Michigan

September 5, 2025

Austin Leo Thomas

I. Problem 1.1

The primary goal of this problem was to graphically determine the location and classification of minima of the function defined by Eq. (1).

$$f(x) = \frac{1}{12}x^4 + x^3 - 16x^2 + 4x + 12 \quad (1)$$

For this purpose, a Python script was written to plot the function over a reasonable domain, recognizing that the function should tend towards positive infinity as x becomes much larger or smaller. The resultant plot is shown in Fig. (1). A larger domain of x than that which is shown in the figure was explored to ascertain if other minima exist, but no such points were found.

The problem also asked for consideration of the number of significant digits which could be reasonably estimated for the location of these minima, via purely graphical means. From Fig. (1) alone one can only place the locations of the minima along the x-axis with a precision of only one to two significant figures. One might estimate the global minimum to be roughly $x_g^* = -16 \pm 1$ and the local minimum to be roughly $x_l^* = 6 \pm 1$; these are the estimated solutions to the problem.

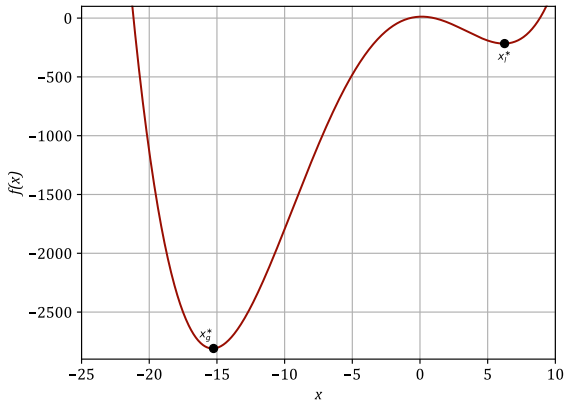

Figure 1. Plot of the function defined by Eq. (1) over the domain $x \in [-25,10]$. Two minima are shown: a local minima, denoted by x_l^*, and a global minima, denoted by x_g^*.

However, since this function is plotted with a computer, rather than by hand, the opportunity exists to achieve much greater precision by simply adjusting the domain and range upon which the function is plotted. Such an attempt was made; the resultant "zoomed-in" plots are shown in Fig. (2) and (3) for the global and local minima, respectively.

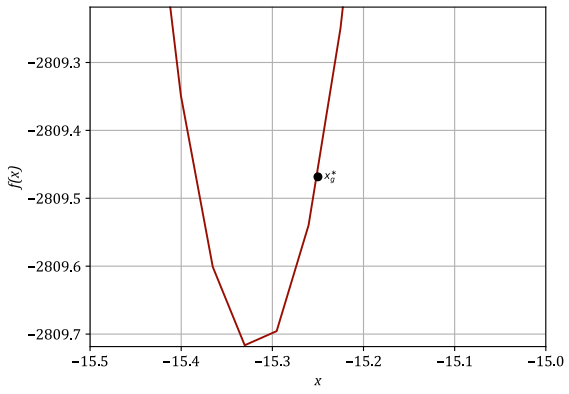

Figure 2. Plot of the function defined by Eq. (1) over the domain $x \in [-15.5, -15.0]$. The estimated position of the global minimum, x_g^*, is shown.

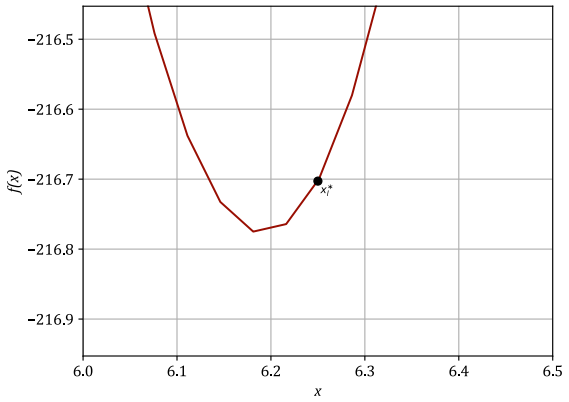

Figure 3. Plot of the function defined by Eq. (1) over the domain $x \in [6.0,6.5]$. *The estimated position of the global minimum,* x_1^*, *is shown.*

Examination of Fig. (2) and (3) reveals that the estimations made of the minima locations based on Fig. (1) were not altogether accurate: though these estimations were near the true minima, by zooming in on the function's plot is becomes clear that they are not themselves minimum values. From Fig. (2) and (3) more accurate estimations of the minima values may be made, with a precision of two to three significant figures.

It must be noted, of course, that estimations of minima locations based on Fig. (2) and (3) will be limited by the precision of the numerical evaluation of the function: that is, the function appears to be discontinuous in these plots, revealing the discretized nature of computational plotting. This may easily be remedied, however, by adjusting the computational resolution: that, by computing $f(x)$ for x-values closer together.

This process of "zooming in" the plot and adjusting the computational resolution of the plotting script may be repeated continuously until eventually limited by machine precision. In this way, graphical approximations of minima locations may become very precise indeed. If $f(x)$ is plotted for values of x with significant figures on the order of machine precision (up to 17 significant figures in double-precision computing), one might reasonably estimate the location of the minimum with a precision at most one order of magnitude less than the computations. As such, with sufficient computing power, one might accurately estimate the location of these minima to 16 significant figures.

II. Problem 1.2

The primary goal of this problem was to graphically determine the location and classification of minima of the function defined by Eq. (2).

$$f(x_1, x_2) = x_1^3 + 2x_1x_2^2 - x_2^3 - 20x_1 \qquad (2)$$

For this purpose, a Python script was written to plot the function contours over a reasonable domain. The resultant plot is shown in Fig. (4).

In analyzing contour plots the issue of precision when estimating a minimum location graphically becomes even more severe. In examining Fig. (4) it is clear that there are a pair of correlating local minimum and maximum values; the function then seems to venture towards positive and negative infinity for more extreme values of x_1 and x_2. Though the true local minimum may exist anywhere within the circular contour line enclosing the x^* point, mathematical intuition leads one to believe it is near the center of the circle. Additionally, given that the point seems to lie very near the x_1-axis, it is reasonable to conclude that x_2^*, the x_2-value of the local minimum, is likely zero.

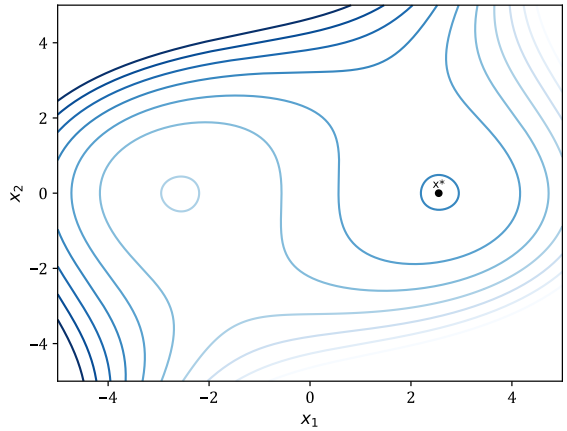

Figure 4. Contour plot of the function described by Eq. (2). Contour lines are evenly distributed with darker lines representing smaller f-values. The estimated local minimum value is denoted by the point x^*.

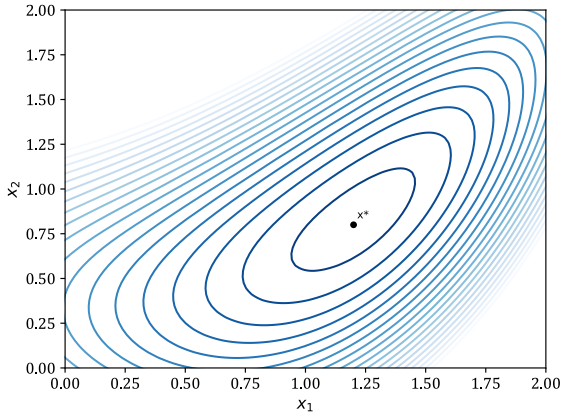

Figure 5. Contour plot of the function described by Eq. (3). Contour lines are evenly distributed with darker lines representing smaller f-values. The estimated local minimum value is denoted by the point x^*.

The x_1-value of the local minimum is much more difficult to confidently estimate: as a best guess, one might say that $x_1^* = 2.5$. This leaves a single local minimum at $(x_1, x_2) = (2.5,0)$ as a final solution to the problem.

III. Problem 1.3

The primary goal of this problem was to determine the location and classification of minima of the function defined by Eq. (3).

$$f(x_1, x_2) = (1 - x_1)^2 + \cdots$$
$$\cdots (1 - x_2)^2 + \frac{1}{2}(2x_2 - x_1^2)^2 \qquad (3)$$

The problem was first approached graphically, as before. A Python script was employed to generate a contour plot of the function over a reasonable domain, and this plot, shown in Fig. (5), was analyzed to estimate the location of any minima.

Examination of the function over a larger domain than is shown in Fig. (5) revealed that the point x^* is indeed a global minimum: as x_1 and x_2 diverge from this point, $f(x_1, x_2)$ tends everywhere towards positive infinity.

As before, it is reasonable to assume that the most likely location for this global minimum, based on the information provided by Fig. (5), is in the center of the smallest oval-shaped contour line enclosing x^*: based on this assumption, the best guess for the coordinates of x^* are $(x_1, x_2) = (1.2,0.8)$.

However, this need not be the end of the problem . Instead, more sophisticated, numerical-based optimization tools were employed to obtain a more accurate estimation of the location of this global minimum. Specifically, a built-in optimization tool included in the SciPy module was employed: the *scipy.minimize* function.

This Python function offers a number of optimization algorithms which may be

applied to a minimization problem such as the one posed here. For this problem, the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm was applied. This algorithm was selected based on the algorithm-selection decision tree created by Dr. Joaquim Martins [1]. The function described by Eq. (3) is clearly continuous and differentiable, and the problem is unconstrained. This makes the BFGS algorithm an ideal approach to this problem.

The efficacy of this approach was first evaluated by providing the algorithm with an initial guess point relatively near the graphically-approximated minimum (1.2,0.8); the initial guess point, or start point, (0,0) was provided and the algorithm was run.

Two plots were generated to illustrate the optimization procedure: a contour plot showing the path of the algorithm through the problem space, with points indicating the algorithm's current guess for each iteration,

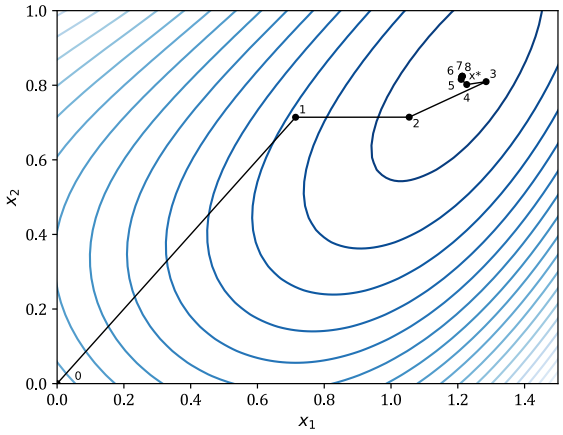

Figure 6. Contour plot of the function described by Eq. (3). Contour lines are evenly distributed with darker lines representing smaller f-values. Optimum values at each iteration are denoted by the iteration number; the final optimum point yielded is denoted by the point x^*.

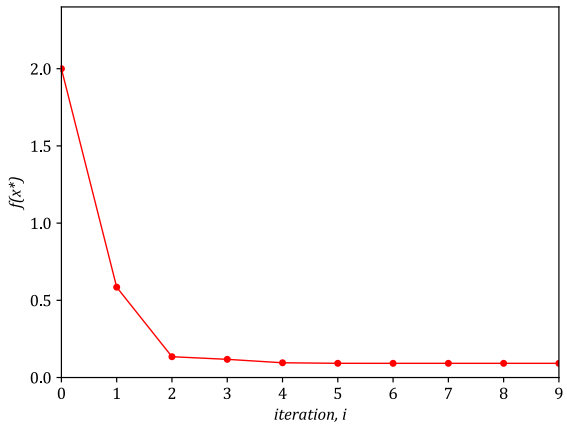

Figure 7. Plot of the function value, as evaluated by Eq. (3), given the current optimum point, as shown in Fig. (6), for each iteration of BFGS optimization from a start point of (0,0)*.*

and a two-dimensional plot of the function value with each iteration. These plots are shown in Fig. (6) and (7), respectively.

The algorithm was able to converge to an optimum point (global minimum) of (1.213,0.8241) in nine iterations. Examination of either Fig. (6) or (7) shows that after only four iterations the algorithm had achieved a value very near this optimum.

Additionally, it is clear that the algorithm converged to a value very near the graphically-approximated point (1.2,0.8). As such, there is no cause for fear that the algorithm's result is erroneous.

With this initial optimization complete, attempts were made to challenge the algorithm by providing start points which were not near the true optimum. Given that the function has no local minima in the region of interest (this was confirmed graphically) the algorithm should not struggle to converge to the optimum point already found. Of interest, however, is the number of iterations required to do so.

The next optimization procedure used a start point (30,30). The same plots were generated for this process as before, and these plots are shown in Fig. (8) and (9).

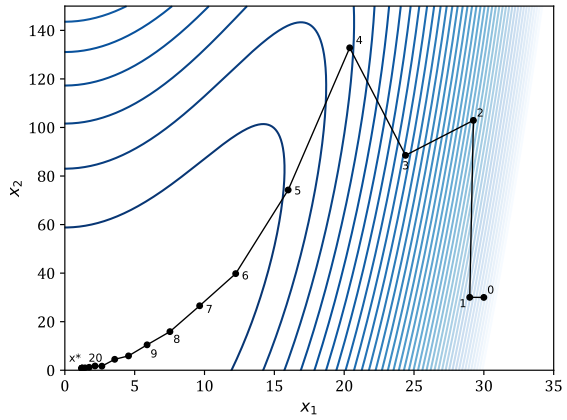

Figure 8. Contour plot of the function described by Eq. (3). Contour lines are evenly distributed with darker lines representing smaller f-values. Optimum values at each iteration are denoted by the iteration number; the final optimum point yielded is denoted by the point x^*.

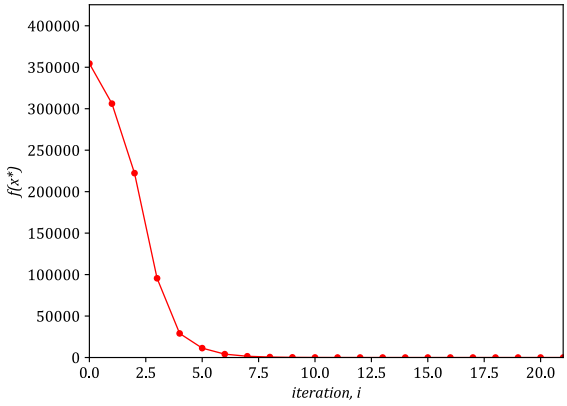

Figure 9. Plot of the function value, as evaluated by Eq. (3), given the current optimum point, as shown in Fig. (8), for each iteration of BFGS optimization from a start point of (30,30).

Here the algorithm again converged to an optimum at (1.213,0.8241). This time, however, the optimum value was found on the twenty-first iteration, with significantly depreciating returns after the eleventh iteration.

The process was repeated once more with a start point of $(-10, -10)$. The same plots were generated and are shown in Fig. (10) and (11). The algorithm again did not struggle to converge to the same optimum at (1.213,0.8241), accomplishing it now on the nineteenth iteration with significantly depreciating returns after the eleventh iteration as well.

Table (I) reflects a summary of the results of these three optimization runs, providing a sense of the computational differences depending on the distance between the optimum and the start point.

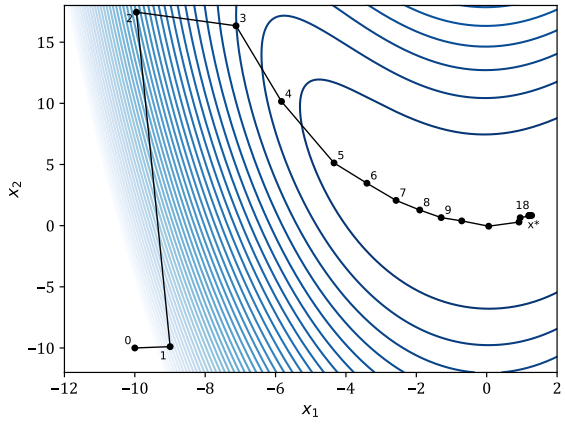

Figure 10. Contour plot of the function described by Eq. (3). Contour lines are evenly distributed with darker lines representing smaller f-values. Optimum values at each iteration are denoted by the iteration number; the final optimum point yielded is denoted by the point x^*.

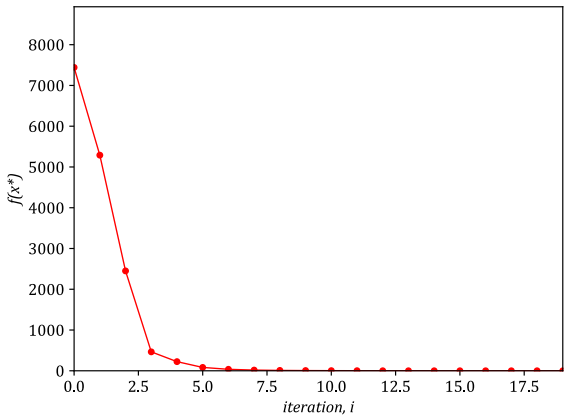

Figure 11. Plot of the function value, as evaluated by Eq. (3), given the current optimum point, as shown in Fig. (10), for each iteration of BFGS optimization from a start point of $(-10, -10)$.

TABLE I
SUMMARY OF OPTIMIZATION PROCESSES

Start Point	Distance from Optimum Point	Iterations to Converge
(0,0)	1.4665	9
(30,30)	40.9869	21
$(-10, -10)$	15.5850	19

IV. Problem 1.4

The paper to be discussed summarizes an effort to optimize the performance of aircraft engines which have been semi-submerged in the aircraft body [2]. The ducts of these engines are often subject to unfavorable boundary layer ingestion which subsequently reduces inlet pressure recovery. Inlet pressure loss is highly unfavorable for engine performance, and indeed it is this pressure loss which the researchers chose to minimize in their optimization problem. Given the ever-evolving field of aircraft design optimization, and the seeming difficulty in addressing this design concern (evidenced by numerous fruitless attempts), this research is both highly relevant and important.

As mentioned, the objective function for this problem was inlet pressure loss. The problem was approached in two parts: first, the inlet duct geometry was optimized while keeping inlet vertical offset and duct length fixed (i.e. as equality constraints). Additional constraints included those imposed by aerodynamics, namely certain boundary conditions, and physics, such as restrictions on mass flow rate and flow distortion.

The optimum converged to under this problem was not appreciably better than the initial design. As such, the problem was then expanded to include the upstream surface geometry in the design space; this optimization procedure yielded highly fruitful results.

As a whole, this problem appears well-posed. When it became clear that the current design space did not offer enough room for significant optimization, the researchers correctly reformulated the problem by expanding the design space. The problem may most readily be classified as a continuous, single-objective, constrained problem [1]. The objective and constraint functions are best classified as generally continuous, nonlinear, multimodal, nonconvex, and deterministic [1].

The optimization method applied was an adjoint-based approach, which is a limited version of gradient-based optimization. Previous attempts at more expansive gradient-based optimization for this problem were unable to sufficiently explore the design space. The adjoint-based approach used here utilizes sensitivity derivatives and gradient vectors to achieve high accuracy without becoming computationally prohibitive. The optimization

algorithm employed was SU2, an open-source CFD software written in C++. This algorithm could best be described as a first-order, local, mathematical, direct, deterministic, and static model [1]. As a whole, this model seems suited to this problem: a second-order approach would be computationally prohibitive, as would stochastic modelling. Surrogate approaches have been taken before to little success, so direct modelling is appropriate here.

The result of the second optimization procedure employed was a novel boundary layer diverting inlet configuration consisting of a "surface groove forking over a slight bump at the inlet throat" [2]. This result appears to be physically and aerodynamically reasonable, and the model indicates its employment would significantly reduce pressure loss at the inlet.

One important consideration is, of course, whether minimizing pressure loss is indeed the correct objective function. The researchers concede that "boundary layer ingestion causes a reduction in the ram drag and possibly increases the overall propulsion system efficiency" [2]. Indeed, if boundary layer ingestion may have desirable attributes, then minimizing pressure loss by minimizing boundary layer ingestion may not be truly optimizing engine performance. A closer evaluation of the effects of this novel inlet design on other design attributes and other possible objective functions may be worthwhile.

V. Disclosures

As required by the syllabus, a disclosure is included here regarding AI use on this assignment. U-M GPT was used as an aide when writing the Python scripts employed in this assignment. The tool was used primarily in place of a search engine to look up syntax for the various modules used in the scripts. The tool was also used in the debugging process when other measures failed. AI did not write any blocks of code submitted for this assignment, nor was the tool used in any part of the authoring of this report.

VI. References

[1] J. R. Martins and A. Ning, Engineering Design Optimization, Cambridge: Cambridge University Press, 2020.

[2] U. C. Kucuk and I. H. Tuncer, "Adjoint based aerodynamic shape optimization of a semi-submerged inlet duct and upstream inlet surface," *Optimization and Engineering,* vol. 25, no. 4, pp. 2205 - 2228, 2024.