Assignment 4 Report

AEROSP 588 – Dr. Joaquim Martins

University of Michigan

October 21, 2025

Austin Leo Thomas

I. Problem 4.1

We let the dimensions of a rectangle be x_1 and x_2 and its fixed perimeter be P. The problem formulation is then:

maximize: $f = x_1 x_2$

subject to: $2x_1 + 2x_2 - P = 0$

We formulate the Lagrangian for this problem as:

$$\mathcal{L} = x_1 x_2 + \lambda(2x_1 + 2x_2 - P) \qquad (1)$$

We then define the partial derivatives of the Lagrangian and set them equal to zero:

$$\frac{\partial \mathcal{L}}{\partial x_1} = x_2 + 2\lambda = 0 \qquad (2)$$

$$\frac{\partial \mathcal{L}}{\partial x_2} = x_1 + 2\lambda = 0 \qquad (3)$$

$$\frac{\partial \mathcal{L}}{\partial \lambda} = 2x_1 + 2x_2 - P = 0 \qquad (4)$$

Solving the system of equations defined by Eq. (2) through (4) yields a singular result: $(x_1, x_2, \lambda) = (0.25P, 0.25P, -0.125P)$. This result corresponds to a square with side lengths equal to one-quarter of the perimeter of the rectangle.

We must now determine if this is a maximum or minimum point by evaluating the Hessian of the Lagrangian, which is:

$$H_{\mathcal{L}} = \begin{bmatrix} \frac{\partial^2 \mathcal{L}}{\partial x_1^2} & \frac{\partial^2 \mathcal{L}}{\partial x_1 \partial x_2} \\ \frac{\partial^2 \mathcal{L}}{\partial x_2 \partial x_1} & \frac{\partial^2 \mathcal{L}}{\partial x_2^2} \end{bmatrix} \qquad (5)$$

$$H_{\mathcal{L}} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

Evaluating the determinant of the Hessian of the Lagrangian shows that:

$$\det(H_{\mathcal{L}}) = -1 < 0 \qquad (6)$$

Thus the Hessian of the Lagrangian is negative definite at the point of interest, indicating that $(x_1, x_2) = (0.25P, 0.25P)$ is a maximum point. Thus we have proven that the area of a rectangle is maximized when the rectangle is a square.

The Lagrange multiplier λ indicates the degree to which the objective function, f, varies with the constraint. In this case, the objective function has units of area while the constraint has units of length. Thus, the units of the Lagrange multiplier are area over length, or simply length. The value of the Lagrange multiplier, $\lambda = -0.125P$, combined with the result that $f = x_1 x_2 = 4\lambda$, reveals the following relation between the optimum area, $f_{\max}$, and the constraint, P:

$$f_{\max} = -0.5P \qquad (7)$$

The role the Lagrange multiplier plays in relating the objective function to the constraint is made clear by in Eq. (7): utilization of the Lagrange multiplier allows for a direct relation between the optimum objective function value and any constraints.

We may arrive at the same solution via solution of an alternative problem:

minimize: $f = 2x_1 + 2x_2$

subject to: $x_1 x_2 - A = 0$

Here, we seek to minimize the perimeter while constraining the area, A; the design variables for this problem are the same as for the previous one. The Lagrangian for this problem is then:

$$\mathcal{L} = 2x_1 + 2x_2 + \lambda(x_1 x_2 - A) \qquad (8)$$

And its partial derivatives, set equal to zero, are:

$$\frac{\partial \mathcal{L}}{\partial x_1} = 2 + 2x_2\lambda = 0 \qquad (9)$$

$$\frac{\partial \mathcal{L}}{\partial x_2} = 2 + 2x_1\lambda = 0 \qquad (10)$$

$$\frac{\partial \mathcal{L}}{\partial \lambda} = x_1 x_2 - A = 0 \qquad (11)$$

Solving the system of equations defined by Eq. (9) through (11) yields two solutions: $(x_1, x_2, \lambda) = \left(\pm\sqrt{A}, \pm\sqrt{A}, \mp\frac{1}{\sqrt{A}}\right)$. Of course only one of these solutions is physically realistic: $(x_1, x_2, \lambda) = \left(\sqrt{A}, \sqrt{A}, -\frac{1}{\sqrt{A}}\right)$. Here again we've found a single extreme value corresponding to a square with side lengths equal to the positive square root of the constrained area, A. We construct the Hessian of the Lagrangian to determine if this is a maximum or minimum value:

$$H_{\mathcal{L}} = \begin{bmatrix} \frac{\partial^2 \mathcal{L}}{\partial x_1^2} & \frac{\partial^2 \mathcal{L}}{\partial x_1 \partial x_2} \\ \frac{\partial^2 \mathcal{L}}{\partial x_2 \partial x_1} & \frac{\partial^2 \mathcal{L}}{\partial x_2^2} \end{bmatrix} \qquad (12)$$

$$H_{\mathcal{L}} = \begin{bmatrix} 0 & 2\lambda \\ 2\lambda & 0 \end{bmatrix}$$

Evaluating the determinant of the Hessian of the Lagrangian shows that:

$$\det(H_{\mathcal{L}}) = -4\lambda = \frac{4}{\sqrt{A}} > 0 \qquad (13)$$

Thus the Hessian of the Lagrangian is positive definite at the point of interest, indicating that $(x_1, x_2) = \left(\sqrt{A}, \sqrt{A}\right)$ is a minimum point.

The combination of these two solutions proves that for a rectangle of a given perimeter, a square shape provides the maximum possible area, and that additionally that constrained perimeter is the minimum possible perimeter of any given rectangle of that area.

II. Problem 4.2

We formulate the problem by substituting the following values into the provided formulation: $h = 0.25$, $b = 0.125$, $\sigma_{\text{yield}} = 2 \cdot 10^8$, $\tau_{\text{yield}} = 1.16 \cdot 10^8$, $l = 1$, and $P = 1 \cdot 10^5$. Thus all values are in SI units and

the solution will be in SI units as well. The problem formulation is then:

$$\text{minimize: } 0.25t_b + 0.25t_w$$

$$\text{by varying: } t_b, t_w$$

$$\text{subject to: } \frac{1.25\cdot10^4}{I} - 2\cdot10^8 \leq 0$$

$$\frac{6\cdot10^4}{t_w} - 1.16\cdot10^8 \leq 0$$

Here, I is given by:

$$I = \frac{t_w}{768} + \frac{t_b^3}{48} + \frac{t_b}{256} \tag{14}$$

We define the Lagrangian as:

$$\begin{gathered}\mathcal{L} = 0.25t_b + 0.25t_w + \\ \cdots \sigma_1\left(\frac{1.25\cdot10^4}{I} - 2\cdot10^8 + s_1^2\right) + \\ \cdots \sigma_2\left(\frac{6\cdot10^5}{t_w} - 1.16\cdot10^8 + s_2^2\right)\end{gathered} \tag{15}$$

Finding the partial derivatives of the Lagrangian and setting them equal to zero yields the following equations:

$$\begin{gathered}\frac{\partial\mathcal{L}}{\partial t_b} = 0.25 - \cdots \\ \frac{12500\sigma_1}{I^2}\left(\frac{3}{48}t_b^2 + \frac{1}{256}\right) = 0\end{gathered} \tag{16}$$

$$\begin{gathered}\frac{\partial\mathcal{L}}{\partial t_w} = 0.25 - \frac{12500\sigma_1}{768I^2}\cdots \\ \cdots - \frac{600000\sigma_2}{t_w^2} = 0\end{gathered} \tag{17}$$

$$\frac{\partial\mathcal{L}}{\partial\sigma_1} = \frac{1.25\cdot10^4}{I} - 2\cdot10^8 + s_1^2 = 0 \tag{18}$$

$$\frac{\partial\mathcal{L}}{\partial\sigma_2} = \frac{6\cdot10^4}{t_w} - 1.16\cdot10^8 + s_2^2 = 0 \tag{19}$$

$$\frac{\partial\mathcal{L}}{\partial s_1} = 2\sigma_1 s_1 = 0 \tag{20}$$

$$\frac{\partial\mathcal{L}}{\partial s_2} = 2\sigma_2 s_2 = 0 \tag{21}$$

This is a system of six unknowns and six equations and can thus be solved directly. The SymPy symbolic solver was used to this end. Seven solutions results, though only a single solution was fully real. That solution is:

$$(t_b, t_w) = (0.01426, 0.00517)\text{ m} \tag{22}$$

$$\sigma_1 = \sigma_2 = s_1 = s_2 = 0 \tag{23}$$

This corresponds to the case where both constraints are active. This solution may be confirmed graphically by plotting the function contours and the constraints; such a plot is shown in Fig. (1).

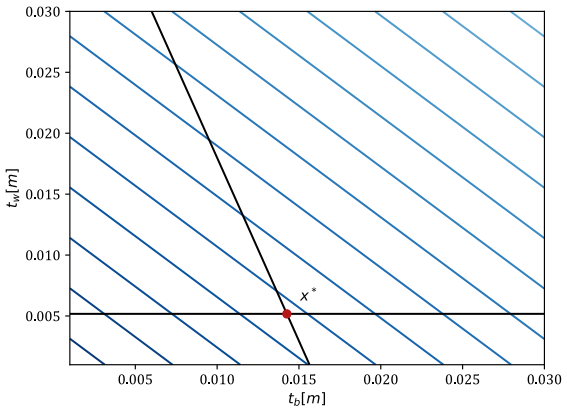

Figure 1. Contour plot showing the variation of cross-sectional area of the beam with the design variables. The constraint functions are plotted here as equality constraints since they are both active. The x^* here is from the analytic solution.

Evaluation of the plot shown in Fig. (1) reveals that the analytic approach to determining the optimum beam dimensions was successful in minimizing the cross-sectional area of the beam given the constraints: the analytic solution x^* is clearly located at the constrained minimum.

III. Problem 4.3

Exterior and interior penalty method algorithms were developed in Python. For the exterior penalty method, a quadratic penalty was applied in the form given by Eq. (5.43) in *Engineering Design Optimization* [1]. For this implementation, $\mu_h = \mu_g = \mu$ and were thus varied in lockstep. For the interior penalty method, the equality constraints were still penalized by a quadratic penalty, but the inequality constraints were now penalized by a logarithmic penalty, per Eq. (5.57) in the text. Here, μ_h and μ_g were varied independently, since $\mu_h \to \infty$ while $\mu_g \to 0$ as the optimization progressed.

Both algorithms were designed to run until either $\|\nabla_x \mathcal{L}\|_\infty < \tau_{\text{opt}}$ or $\|h\|_\infty < \tau_{\text{feas}}$, where $\tau_{\text{opt}} = 1 \cdot 10^{-6}$ and $\tau_{\text{feas}} = 0.1$. This relatively high value for τ_{feas} was chosen so as to prevent μ from becoming too large or too small, which often led to the optimizers failing during testing.

These algorithms were put to use to solve Example (5.4) in the text. This problem is given as:

$$\text{minimize: } f = x_1 + 2x_2$$

$$\text{subject to: } \frac{1}{4}x_1^2 + x_2^2 - 1 \leq 0$$

The problem was attempted with an initial guess $x_0(0,0)$. For the exterior penalty method, $\mu_{\text{init}} = 1$ and $\rho = 2$. For the interior penalty method, on the other hand, $\mu_{\text{init}} = 1$, $\rho_h = 2$, and $\rho_g = 0.5$.

The exterior penalty algorithm succeeded in converging the constrained optimum for this problem; the interior penalty algorithm did not. The optimization paths for both solutions are shown in Fig. (2) and (3).

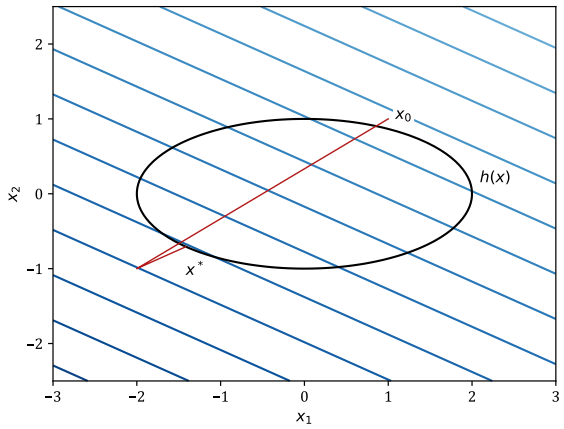

Figure 2. Optimization path for the exterior penalty method, attempting to solve Ex. (5.4) in the text. The inequality constraint is shown here as an equality constraint since it is active at the optimum. The initial guess is $x_0(1,1)$.

Examination of Fig. (2) reveals that the algorithm succeeded in finding the optimum, converging to $x^*(-1.416, -0.708)$ after ten iterations, with an optimum value $f^* = -2.83$. The algorithm was terminated not as a result of meeting the convergence criteria but rather due to a manually-set threshold on the μ-value. During testing it was found that μ-values exceeding 1000 began to fail the line search step of the algorithm, resulting in a never-ending line search necessitating manual termination of the program.

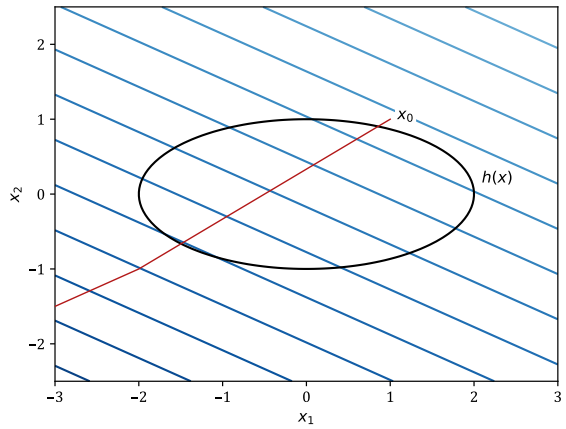

Figure 3. Optimization path for the interior penalty method, attempting to solve Ex. (5.4) in the text. The inequality constraint is shown here as an equality constraint since it should be active at the optimum. As seen, the algorithm failed to converge. The initial guess is $x_0(1,1)$.

Because of this limitation, there is considerable error in the result: at even three-decimal precision we see deviation from the true result of $x^*(-1.414, -0.707)$. We also se that the converged result is on the infeasible side of the constraint. This makes logical sense, given that the algorithm is approaching from the infeasible side and that we expect infeasible results when using the exterior penalty method. Nonetheless, the combination of a relatively inaccurate and infeasible result, and the inability to push closer to the optimum without breaking the algorithm, makes this result very disappointing indeed.

Even more disappointing is the result of the interior penalty method, which failed to converge at all. This algorithm also terminated after hitting a μ-value threshold: the limit that $\mu_g > 10^{-3}$. Despite many hours of tuning the algorithm and looking for any possible errors, the algorithm could never converge. Pushing the value of μ_g lower than this threshold caused the line search to fail, but it seems that with this limited range of μ_g-values, the interior penalty method cannot succeed. Close analysis of the problem led to the conclusion that the line search being implemented (a strong Wolfe line search) may not be well-programmed enough to be applied well to penalty methods. Reformulating the line search, however, is beyond the scope of this investigation (and unnecessary, given that it is sufficient for the soon-to-be-discussed and far superior SQP method).

Other starting points were attempted as well. With these other initial guesses, the exterior penalty method was still able to converge, though the interior method still failed. One such result is illustrated in Fig. (4), the result of the exterior penalty algorithm applied with an initial guess $x_0(-2,2)$.

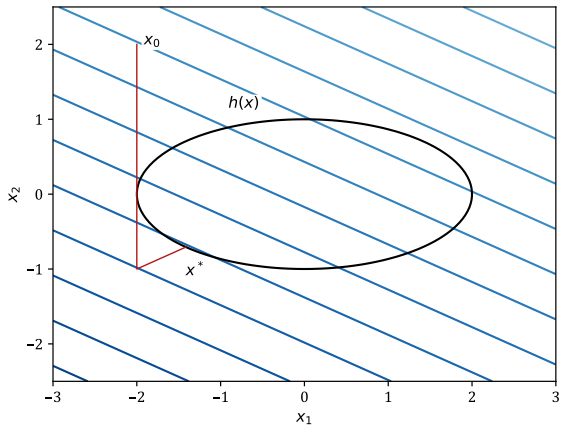

Figure 4. Optimization path for the exterior penalty method, attempting to solve Ex. (5.4) in the text. The inequality constraint is shown here as an equality constraint since it is active at the optimum. The initial guess is $x_0(-2,2)$.

This investigation illustrates the significant disadvantages of penalty methods as a whole: converged points are often inaccurate, due to the limitations of the size of

μ-values, and in the case of the exterior penalty method, solutions may be infeasible as well. For these reasons, penalty methods are not seriously considered for most engineering problems.

IV. Problem 4.4

An SQP optimizer was constructed in Python to handle equality-constrained problems. This optimizer made use of a merit function to perform it's line search, defined in Eq. (5.89) and (5.90) of *Engineering Design Optimization* [1]. A backtracking line search was employed for simplicity and robustness: the constrained nature of problems which are solved via SQP optimization may lead to difficulty in satisfying the sufficient curvature conditions, hence why a backtracking line search was chosen over a line search based in the strong Wolfe conditions.

A damped BFGS update was employed to generate approximate Hessians in each iteration. The Hessian approximation was initialized as the identity matrix. Additionally, whenever $r_k^{\mathrm{T}} s_k < 0$ the approximation was reset to the identity matrix.

The algorithm was designed to run until either $\|\nabla_x \mathcal{L}\|_\infty < \tau_{\text{opt}}$ or $\|h\|_\infty < \tau_{\text{feas}}$, where $\tau_{\text{opt}} = \tau_{\text{feas}} = 1 \cdot 10^{-6}$. For each line search, an initial step size of $\alpha_{\text{init}} = 1$ was utilized, corresponding to a full Newton step. For the merit function employed for the line search, a value of $\mu = 10$ was applied to Eq. (5.89) in *Engineering Design Optimization* [1].

This algorithm was first applied to Example (5.4) in in the text, with a modification to the constraint to convert it from an inequality constraint to an equality constraint. The problem was then given as:

$$\text{minimize: } f = x_1 + 2x_2$$

$$\text{subject to: } \frac{1}{4}x_1^2 + x_2^2 - 1 = 0$$

This problem was readily solved with 16 iterations and 97 function calls. The optimum value was found to be $x^*(-1.414, 0.707)$ and the optimum function value was found to be $f^* \sim -2.828$. The optimization path taken by the SQP optimizer is shown in Fig. (5).

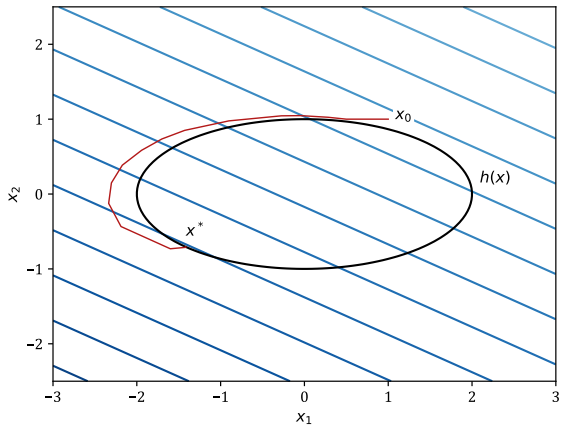

Figure 5. Optimization path taken by the SQP optimizer in solving Example (5.4) in the text.

Evaluation of Fig. (5) reveals that the optimizer was quite efficient in locating the correct optimum point. Interestingly, it is easy to see from the plot the effect the curvature of the constraint has on the optimization path: the curvature of the optimization path roughly follows the curvature of the constraint curve up until the final step. This reveals a potential inefficiency of the SQP algorithm: by following this curvature, the optimizer failed to take a more direct route which would cross the constraint curve, running orthogonal to the constraint curvature and in the direction of steepest descent of the objective function.

With this simple problem solved well, we proceed to test the robustness of the algorithm against increasingly difficult problem configurations. To this end, we look to the Rosenbrock function. The general n-dimensional Rosenbrock function, defined by Eq. (D.3) in *Engineering Design Optimization*, was utilized as the objective function. A single constraint function was also applied:

$$\sum_{i=1}^{n_x} x_i = 0 \qquad (24)$$

The decision to apply only a single constraint was an intentional one. Because the SQP algorithm developed here works only for equality-constrained problems, the problem actually becomes simpler as more constraints are added. The addition of equality constraints effectively lowers the dimensionality of the problem, reducing the degrees of freedom with which the design variables may be changed. In the extreme case, there is only one feasible point and the problem becomes an issue of solving a nonlinear system rather than actually optimizing the objective function. Thus, in testing the robustness of the algorithm, applying only one constraint is preferable.

Seven test cases were applied, as described in Table (I). First the dimensionality of the problem was increased while holding the starting point (effectively) constant and near the expected optimum. Once the dimensionality of the problem was raised sufficiently (Cases I through IV), the starting point was moved progressively further from the expected optimum (Cases V through VII). The results of this study are summarized in Table (II).

Table I
Rosenbrock Test Cases for SQP Alg.

Case	n_x	x_0
I	2	(2,2)
II	4	(2,2, … ,2)
III	8	(2,2, … ,2)
IV	16	(2,2, … ,2)
V	16	$(-1,-1,\dots,-1)$
VI	16	(5,5, … ,5)
VII	16	(25,25, … ,25)

Table II
Rosenbrock Test Results

Case	x^*	f^*	Iter.
I	$(0.01,-0.01)$	0.99	10
II	(0,0, … ,0)	2.99	20
III	(0,0, … ,0)	6.99	93
IV	(0,0, … ,0)	14.99	321
V	(0,0, … ,0)	14.99	176
VI	(0,0, … ,0)	14.99	340
VII	(0,0, … ,0)	14.99	902

Evaluation of the results shown in Table (II) reveal that the algorithm is rather robust indeed. For low-dimensions problems, such as Cases I and II, the algorithm converges to a viable and presumably optimal point in relatively few iterations. As problem dimensionality increases, as in Cases III and IV, the number of iterations required for convergence also increases, but not prohibitively so. Moving the starting point away from the optimum, as in Cases V through VII, does begin to become computationally expensive, though the algorithm still converges to the same optimum in all cases. Even when the starting point is very far from the optimum, as in Case VII, the algorithm still eventually converges. For a challenging high-dimensional problem such as the 16-variable Rosenbrock function, such performance is

impressive and indicative of a very strong algorithm.

The two-dimensional case, Case I, was plotted to allow for a visualization of the SQP algorithm's path through the design space. This plot is shown in Fig. (6).

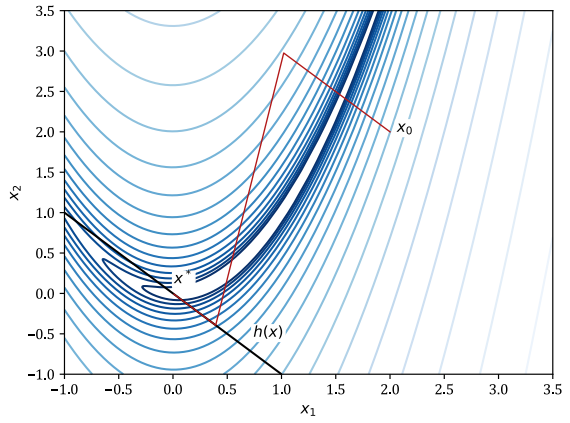

Figure 6. Optimization path taken by the SQP optimizer in solving Case I in Table (I): the two-dimensional Rosenbrock function with a single constraint, defined in Eq. (24), with $x_0(2,2)$.

Examination of Fig. (6) reveals some interesting characteristics of the algorithm's performance. Most notably, it is seen that the algorithm only changes direction twice, despite running for ten iterations. This indicates that the lien search is not as efficient as it could be. If larger steps were taken, or if the strong Wolfe conditions were enforced, it is possible that the algorithm could converge in just three steps. However, convergence in ten iterations is not bad by any means, so tuning is not necessarily required. Given the complexity of the problem, Fig. (6) reflects a well-conditioned optimization path and leaves no cause for concern when applying the algorithm to higher-dimensional problems.

Finally, the results of this homemade SQP algorithm were briefly compared to the results of SciPy's off-the-shelf SQP tool. Expectedly, the performance of the off-the-shelf tool was significantly better than the performance of the homemade algorithm. The SciPy tool was given Cases IV through VII of the n-dimensional Rosenbrock study just discussed, and though the results for both my algorithm and the SciPy algorithm were approximately identical (all optimum design variables were some very small number), the SciPy tool used only a small fraction (several order of magnitude difference for Case VII) of the function calls my algorithm used. This is indicative of some underlying inefficiencies in my algorithm, which is expected given the lack of emphasis on computational efficiency.

Also of note is that the optimum values listed in Table (II) which were found with my algorithm tended to be ~0.01 below the true value (i.e. reporting 14.99 rather than 15). The SciPy results did not have this deficiency, indicating a noticeable increase in accuracy of the reported optimum function value despite no noticeable discrepancy in the reported optimum design variable values. This is likely due to a stricter convergence tolerance employed by the SciPy optimizer, and is not reflective of a poor algorithm on my part.

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.