Assignment 2 Report

AEROSP 588 – Dr. Joaquim Martins

University of Michigan

September 16, 2025

Austin Leo Thomas

I. Problem 1.1

The first goal of this problem is to solve Kepler's equation, shown in Eq. (1), via both Newton's method and fixed-point iteration for a mean anomaly $M = \frac{\pi}{2}$ and eccentricity scalar $e = 0.7$.

$$E - e \cdot \sin E = M \qquad (1)$$

A Python script was employed to this end. The algorithms were written with a stop criteria based on the relative error between subsequent E-values as defined by Eq. (3.21) in *Engineering Design Optimization* [1]. For this stop criteria to be achieved and the algorithm terminated, the relative error had to fall below $\varepsilon = 1 \cdot 10^{-16}$. This provides our solution with double precision and makes use of the full reach of machine precision: accuracy to sixteen digits. Table (I) summarizes the results of solving Eq. (1) for the provided M- and e-values via both Newton's method and fixed-point iteration.

TABLE I
Kepler's Equation Solutions

Method	Iteration	Solution, E [rad]
Newton's	6	2.1547852 … … 93101842
fixed-point	38	2.1547852 … … 93101842

Clearly both approaches yielded the same result, though Newton's method arrived at the solution significantly faster. This leads to the question of comparing the rates of convergences of these two approaches. Newton's method converges quadratically ($p = 2$) while fixed-point iteration converges linearly ($p = 1$) [1]. The convergence constants for these two cases may be found according to Eq. (3.16) in *Engineering Design Optimization*, using the second- and third-to-last E-values for x_{k+1} and x_k, respectively. These values were both computed to be zero, meaning that both methods have identical convergence constants. Since $\gamma \to 0$ and $p \geq 1$ for the fixed-point iteration case, the method is superlinear for this problem.

The next goal of the problem was to evaluate the robustness of the methods by inputting different initial guesses for the E_0. Since the stop criteria for the algorithms will not change, both approaches will yield the exact solution shown in Table (I) for all E_0-values. Of interest, however, is the number of iterations required to achieve that solution. The solutions already found made use of an initial guess of $E_0 = \pi$. The number of iterations required for convergence for this and other E_0-values are shown in Table (II).

TABLE II

Robustness Evaluation Summary

E_0	Method	Iterations
π	Newton's	6
	fixed-point	38
$\frac{\pi}{36}$	Newton's	8
	fixed-point	38
$\frac{35\pi}{36}$	Newton's	6
	fixed-point	38

Clearly both methods – especially fixed-point iteration, which remained completely unaffected – are quite robust, converging in a similar number of iterations regardless of the proximity of the initial guess to the true solution.

The next goal of the problem was to plot E as a function of $M \in [0,2\pi]$ for eccentricity scalars $e = \{0,0.1,0.5,0.9\}$. This plot is shown in Fig. (1).

The final goal of the problem was to analyze the numerical noise associated with

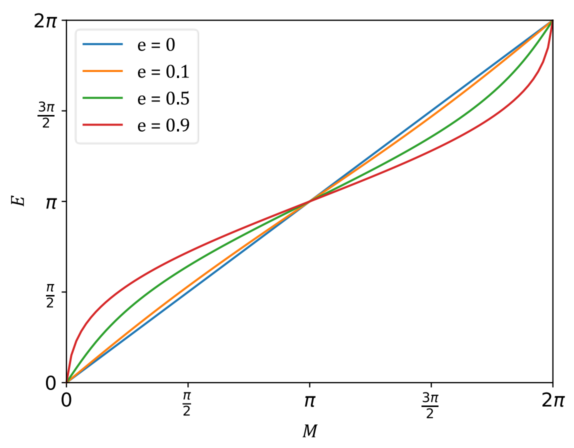

Figure 1. Plot of eccentric anomaly, E, *versus mean anomaly,* M, *for various eccentricity values,* e, *as found via Newton's method for numerically solving Kepler's equation.*

Newton's method. This was done by applying the algorithm for one hundred M-values in the range $M \in \left[\frac{\pi}{2} - 0.01, \frac{\pi}{2} + 0.01\right]$. Here the stop criteria was changed to one based on convergence tolerance: once the value of the absolute value of the residual of Kepler's equation passed below 0.01, the algorithm was terminated. Additionally, the initial guess values for this solution were varied randomly across a range $E_0 \epsilon [M_i - 0.5, M_i + 0.5]$, where M_i was the M-value for the current computation. The resultant plot is shown in Fig. (2).

Clearly there is non-negligible noise when employing Newton's method for solution of Kepler's equation. This noise could be reduced by enforcing a more strict convergence tolerance on the algorithm, or by use of a more sophisticated method of numerical approximation.

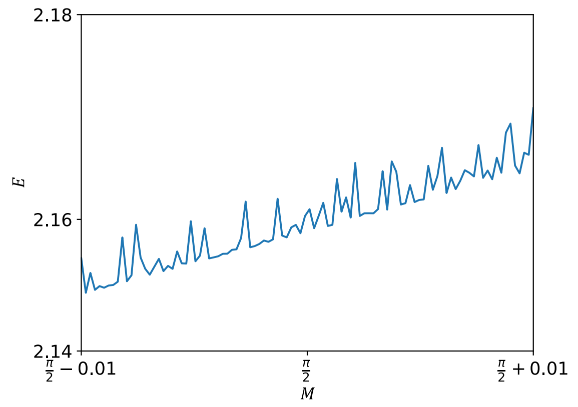

Figure 2. Numerical noise in the solution of Kepler's equation via Newton's method in the region $M \in \left[\frac{\pi}{2} - 0.01, \frac{\pi}{2} + 0.01\right]$.

II. Problem 1.2

The goal of this problem was to analytically identify and classify the critical points of the two-variable function f, defined in Eq. (2), and then plot the function and points in a contour plot.

$$f(x_1, x_2) = x_1^4 + 3x_1^3 + 3x_2^2 - \cdots \\ \cdots 6x_1x_2 - 2x_2 \tag{2}$$

Critical points of the equation may be readily found by setting the partial derivatives of the function, $\frac{\partial f}{\partial x_1}$ and $\frac{\partial f}{\partial x_2}$, equal to zero and solving for x_1 and x_2. This procedure is shown in Eq. (3) and (4), and the solutions of these equations are summarized in Table (III).

$$\frac{\partial f}{\partial x_1} = 4x_1^3 + 9x_1^2 - 6x_2 = 0 \tag{3}$$

$$\frac{\partial f}{\partial x_2} = 6x_2 - 6x_1 - 2 = 0 \tag{4}$$

TABLE I

Critical Points of Eq. (2)

Point	x_1	x_2	$f(x_1, x_2)$
A	$-\frac{1}{4}$	$\frac{1}{12}$	-0.0638
B	$-1-\sqrt{3}$	$-\frac{2}{3}-\sqrt{3}$	-22.7
C	$-1+\sqrt{3}$	$-\frac{2}{3}+\sqrt{3}$	-1.94

To classify these critical points, it is necessary to consider the Hessian matrix and its eigenvalues when evaluated at each critical point. The Hessian matrix is found according to Eq. (4.10) in *Engineering Design Optimization* and is evaluated for this problem per Eq. (5). The Hessian matrix for each critical point and their corresponding eigenvalues are shown in Table (IV).

The Hessian matrix of Point $A\left(-\frac{1}{4}, \frac{1}{12}\right)$ has both a positive and a negative eigenvalue, making it a saddle point. The Hessian matrices

TABLE IV

Hessian Matrices of Critical Points

Point	Hessian	Eigenvalues
A	$\begin{bmatrix} -3.75 & -6 \\ -6 & 6 \end{bmatrix}$	$\lambda_1 = 8.95$ $\lambda_2 = -6.20$
B	$\begin{bmatrix} 40.4 & -6 \\ -6 & 6 \end{bmatrix}$	$\lambda_1 = 41.4$ $\lambda_2 = 4.98$
C	$\begin{bmatrix} 19.6 & -6 \\ -6 & 6 \end{bmatrix}$	$\lambda_1 = 21.9$ $\lambda_2 = 3.73$

of both Points $B\left(-1-\sqrt{3},-\frac{2}{3}-\sqrt{3}\right)$ and $C\left(-1+\sqrt{3},-\frac{2}{3}+\sqrt{3}\right)$ have two positive eigenvalues, making them local minima. However, since $f \to +\infty$ as $x_1, x_2 \to \pm\infty$, one of them is also a global minima. An examination of the function evaluations at these points, shown in Table (III), reveals that Point B is the global minimum. Fig. (3) shows these solutions graphically via a contour plot.

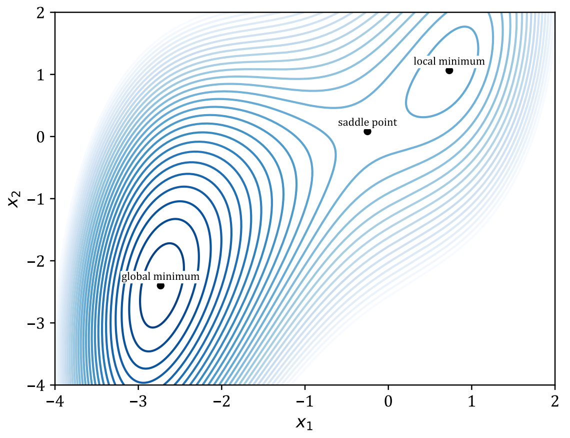

Figure 3. Contour plot showing critical points of Eq. (2). Darker lines indicated larger f*-values.*

III. Problem 1.3

The goal of this problem was to generate line search algorithms based on both backtracking and the strong Wolfe criteria, and to apply these algorithms to a number of optimization problems.

The first equation to be dealt with is described in Example 4.8 of *Engineering Design Optimization* [1]. This example, along with Example 4.9, detail the parameters required for recreation of the solutions presented in the text. Those parameters are: $x_0 = (-1.25, 1.25)$, $p = \langle 4, 0.75\rangle$, $\mu_1 = 10^{-4}$, $\rho = 0.7$, $\alpha_{init,large} = 1.2$, $\alpha_{init,small} = 0.05$, $\sigma = 2$, and $\mu_2 = 0.9$.

Plots were generated showing the line search over the α-path (that is, a plot of f versus α) and over the two-dimensional design space (i.e. on a contour plot). These plots are shown in Fig. (4) to (11).

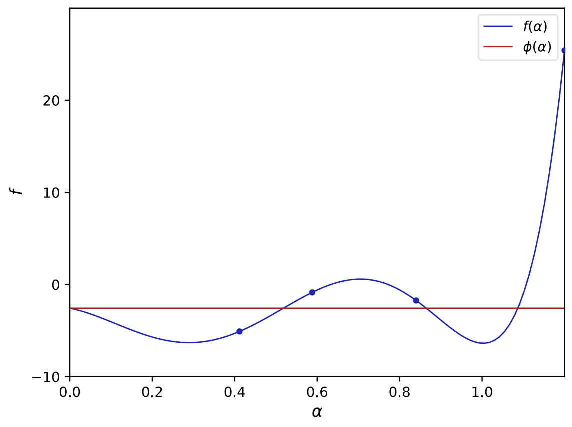

Figure 4. Backtracking line search path in sliced design space, $\alpha_{init} = 1.2$.

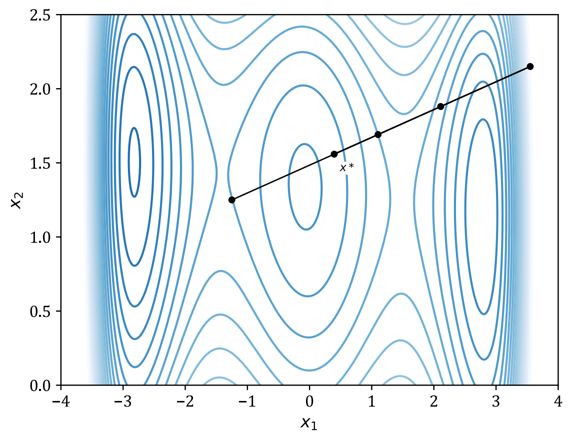

Figure 5. Backtracking line search path in full design space, $\alpha_{init} = 1.2$.

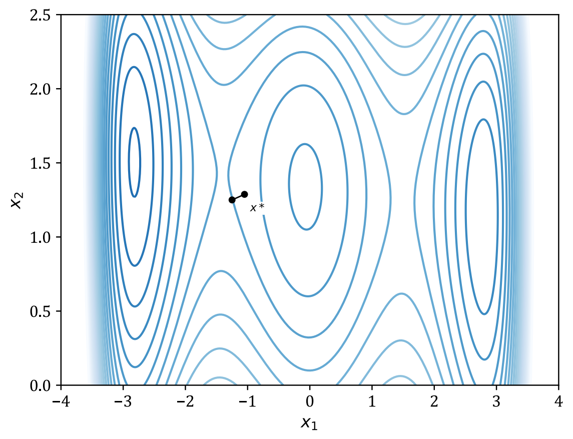

Figure 7. Backtracking line search path in full design space, $\alpha_{init} = 0.05$.

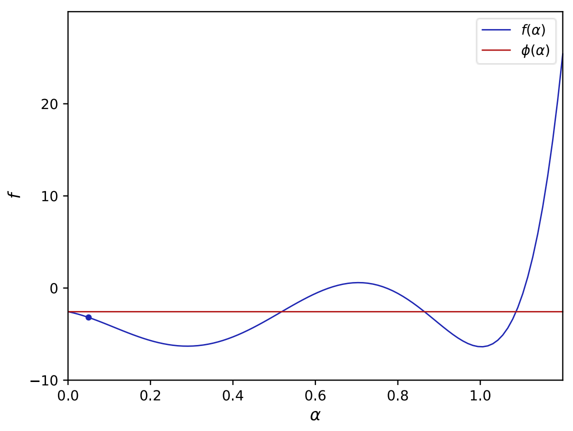

Figure 6. Backtracking line search path in sliced design space, $\alpha_{init} = 0.05$.

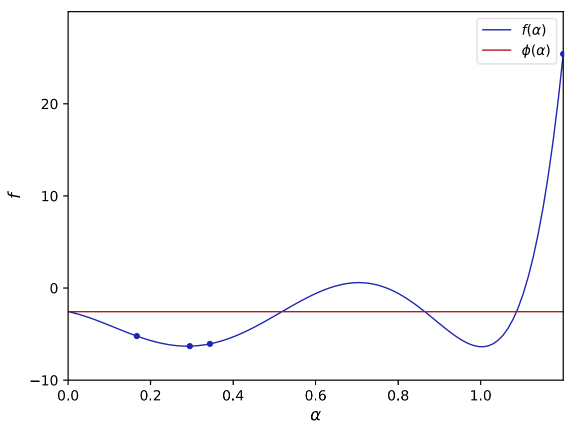

Figure 8. Strong Wolfe line search path in full design space, $\alpha_{init} = 1.2$.

By varying μ_2, the strong Wolfe algorithm may be tuned to eventually achieve an exact line search, at the cost of increasing computational expense. Such a well-tuned algorithm was found, in this instance, with the large initial step size ($\alpha_{init,large} = 1.2$), to be $\mu_2 = XXX$. In theory, by setting μ_2 closer and closer to 10^{-16}, the precision of the line search will approach one order of magnitude above machine precision (that is, 10^{-15}).

The next goal of this problem was to apply these algorithms to a different two-dimensional function. For this aim, the function defined by Eq. (2), which was

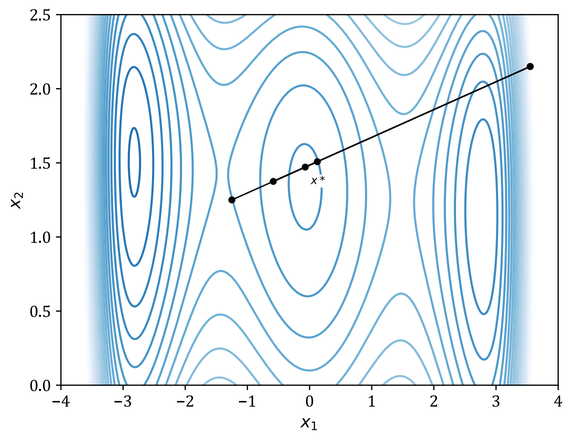

Figure 9. Strong Wolfe line search path in full design space, $\alpha_{init} = 1.2$.

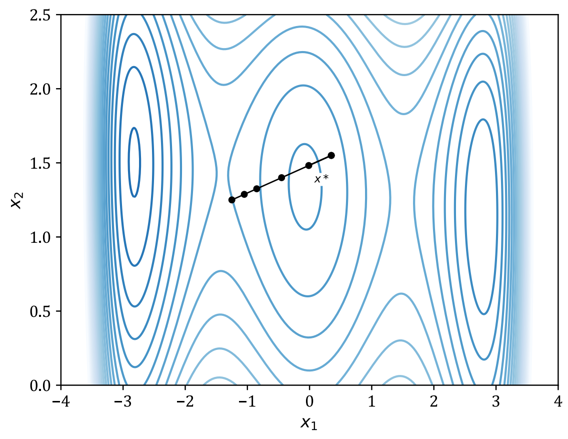

Figure119. Strong Wolfe line search path in full design space, $\alpha_{init} = 0.05$.

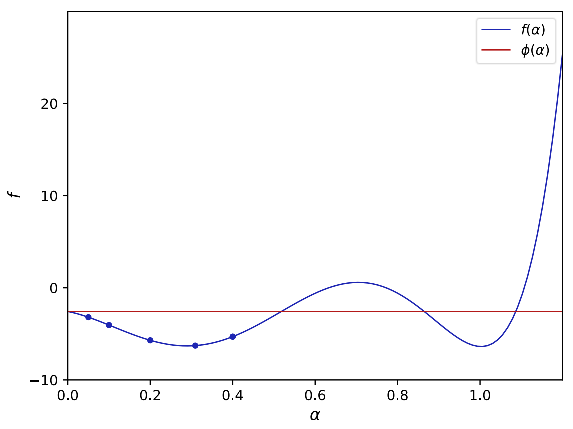

Figure 10. Strong Wolfe line search path in full design space, $\alpha_{init} = 0.05$.

previously minimized analytically, was selected. For this exploration, the following parameters were utilized: $x_{0,base} = (-1,1)$, $x_{0,exp} = (0,1)$ $p_{base} = \langle -\sqrt{2}, -\sqrt{2} \rangle$, $p_{exp} = \langle 1,0 \rangle$, $\mu_1 = 10^{-4}, \rho = 0.7, \alpha_{init} = 1, \sigma = 2$, and $\mu_2 = 0.9$. Thus rather than varying the initial step size, as before, now the initial point and direction are being varied. Plots of the line search process in the sliced design space were generated for both algorithms and evaluate the robustness of these approaches when utilized with non-ideal parameters. The plots are shown in Fig. (12) to (15).

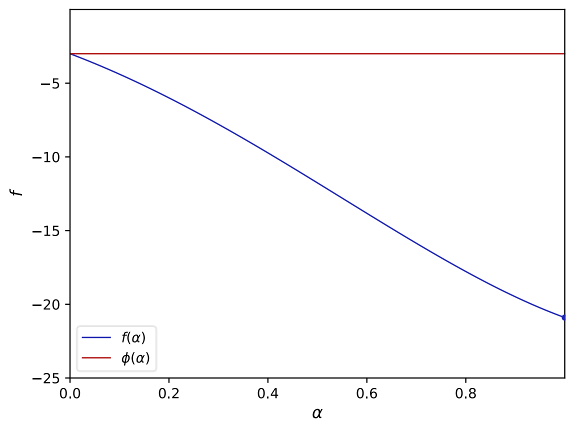

Figure 12. Backtracking line search path in sliced design space under 'base' conditions.

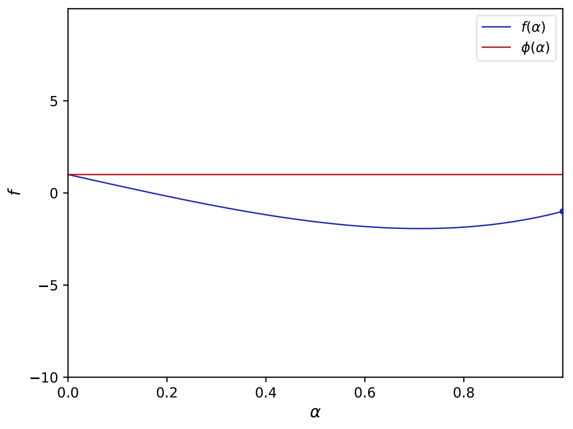

Figure 13. Backtracking line search path in sliced design space under 'experimental' conditions.

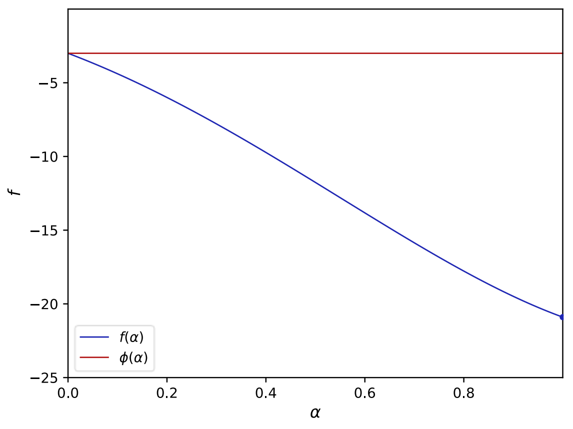

Figure 14. Strong Wolfe line search path in sliced design space under 'base' conditions.

Evaluation of these plots offers excellent insight into the utility of the strong Wolfe criteria. It is seen that under 'base conditions', both algorithms perform identically: there is a constant downward slope in the path of the line search and so both

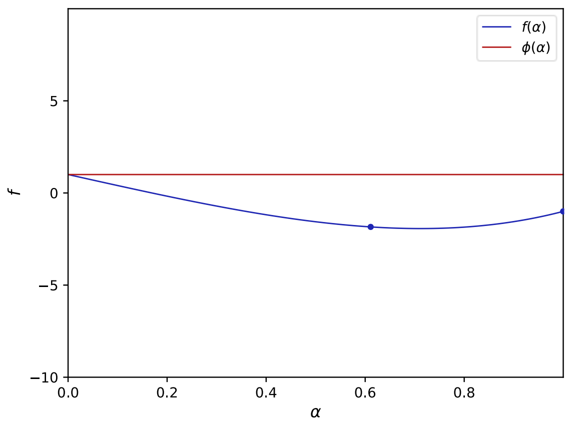

Figure 15. Strong Wolfe line search path in sliced design space under 'experimental' conditions.

algorithms converge to the furthest possible point downhill. In such a case, nothing is gained from the additional effort of enforcing the strong Wolfe criteria.

However, when applying these same algorithms to the 'experimental' conditions, the strong Wolfe approach proves much better. The backtracking approach again converges to the furthest possible point: this time, however, this is not the lowest point on the path. The strong Wolfe criteria allows for an additional iteration of convergence further than the backtracking approach does, and as a result the strong Wolfe algorithm yields a solution very near the minima on the path.

It is important to note that the search directions for both cases were selected based on the contour plot of the function, shown in Fig. (3). Had care not been taken to select direction vectors pointing downhill of the starting point, these algorithms would not have worked.

As an additional point of inquiry, the relationship between μ_2, ρ, and the number of iterations for convergence was explored under the experimental conditions. These results are summarized in Table (V).

TABLE V

Varying μ_2 and ρ for Eq. (2) Solution

Method	Value	Iterations
backtracking	$\rho = 0.2$	1
	$\rho = 0.4$	1
	$\rho = 0.6$	1
	$\rho = 0.8$	1
strong Wolfe	$\mu_2 = 0.2$	102
	$\mu_2 = 0.4$	1
	$\mu_2 = 0.6$	1
	$\mu_2 = 0.8$	1

Evaluation of Table (V) make evident that the backtracking method is quite robust with respect to the selection of ρ-value, at least in the case where the entire path lies below the initial point. The strong Wolfe method, on the other hand, is rather sensitive to selection of the μ_2-value: small values lead to massive numbers of iterations for convergence. This makes sense, as smaller μ_2-values equate to a near-perfect line search, which is computationally expensive and not always attainable.

The next goal of this problem was to apply these algorithms to the two-dimensional Rosenbrock function, so as to challenge them. The same approach was taken as for Eq. (2): two different starting points and two different direction vectors were used. The parameters are: $x_{0,base} = (0,1)$, $x_{0,exp} = (1,0)$ $p_{base} = \langle\sqrt{2},\sqrt{2}\rangle$, $p_{exp} = \langle -1,0\rangle$, $\mu_1 = 10^{-4}$, $\rho = 0.7$, $\alpha_{init} = 2$, $\sigma = 2$, and $\mu_2 = 0.9$. Plots were again generated of the sliced design space, shown in Fig. (16) to (19).

A similar study of the robustness of these algorithms when faced with varying ρ- and μ_2-values was conducted, the results of which are summarized in Table (VI).

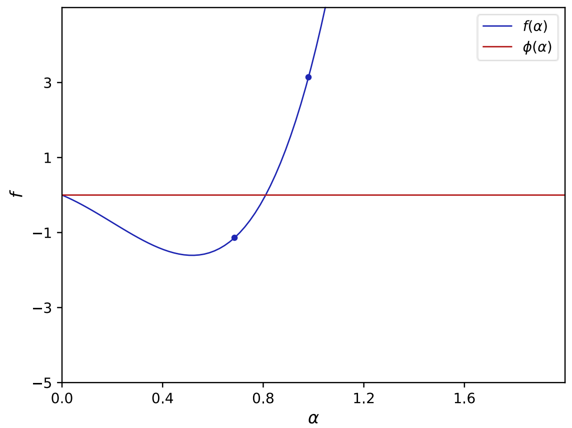

Figure 16. Backtracking line search path in sliced design space under 'base' conditions.

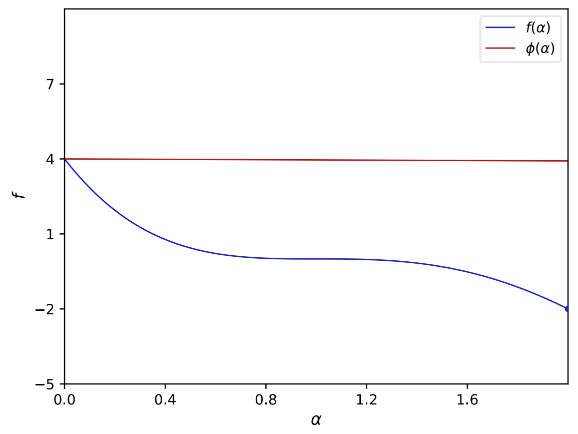

Figure 17. Backtracking line search path in sliced design space under 'experimental' conditions.

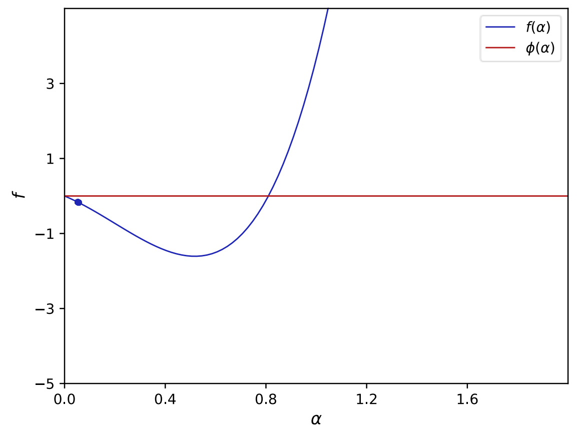

Figure 18. Strong Wolfe line search path in sliced design space under 'base' conditions.

Examination of Table (VI) makes clear that the results found for the previous robustness study are valid: varying of ρ and μ_2 does not have a significant effect on the number of iterations required for convergence.

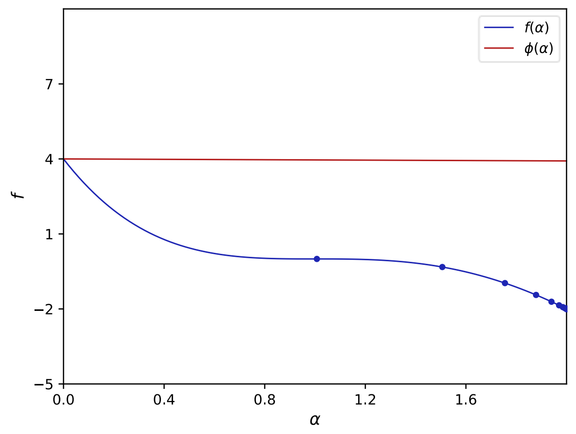

Figure 19. Strong Wolfe line search path in sliced design space under 'experimental' conditions.

TABLE VI

Varying μ_2 and ρ for Rosenbrock Solution

Method	Value	Iterations
backtracking	$\rho = 0.2$	1
	$\rho = 0.4$	1
	$\rho = 0.6$	1
	$\rho = 0.8$	1
strong Wolfe	$\mu_2 = 0.2$	102
	$\mu_2 = 0.4$	102
	$\mu_2 = 0.6$	102
	$\mu_2 = 0.8$	102

Examination of Fig. (16) to (19) further illustrates the power of these algorithms. Even when faced with an intentionally difficult optimization problem, they succeed in finding acceptable (albeit non-ideal) points of descent in the line search method, despite non-ideal parameters.

The final goal of this problem was to test these algorithms with a non-two-dimensional problem. The six-dimensional Rosenbrock function was used for this goal. Of course, no plots could be made for such a problem, but data was collected regarding the minimum point along the path and the number of iteration required to arrive at that point for each algorithm. The following parameters were used: $x_0 = (0,0,0,0,0,0)$, $p = \langle\sqrt{2},\sqrt{2},\sqrt{2},\sqrt{2},\sqrt{2},\sqrt{2}\rangle$, $\mu_1 = 10^{-4}$, $\rho = 0.7$, $\alpha_{init} = 2$, $\sigma = 2$, and $\mu_2 = 0.9$.

The backtracking method yielded a minimum function value along the path of 0.424 after four iterations. The strong Wolfe method yielded a minimum function value along the path of 4.97 after 3 iterations. The algorithms did not get stuck, nor did they take unreasonably long to converge, Without graphical confirmation it is difficult to say with certainty how well they performed, but it is reasonable to conclude, based on other results, that they were at least partially successful in their line search efforts.

IV. 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 to generate the gradient function of the 6-dimensional Rosenbrock function, since I could not find a good source online for that calculation. 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.

V. References

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