This project will involve applying two optimization algorithms to the so-called Rosenbrock function:

\[ f(x) = 100\left( x_2 - x_1^2 \right)^2 + (1 - x_1)^2 \]

In a previous assignment, you found expressions for the gradient and Hessian of this function, showed that \( x^* = (1, 1)^T \) is the only local minimizer of \( f \), and that \( F(x^*) \) is positive definite.

Your "product" (what you turn in) for this project should be a project report that has (at minimum) an introduction section and sections corresponding to items 1 - 3 below. You should strive for clarity and good organization, but also for conciseness.

1. Implement the Steepest Descent algorithm and apply it to the Rosenbrock function, using the Golden Section line search with the Strong Wolfe stopping conditions. Recall that you developed a program for this line search algorithm in Project 1, so if that line search program was correct, you can simply use it in this project.

A complete algorithm also needs a stopping condition to determine when we are "close enough" to the final solution (when to stop the high-level iterations). For this project, we will use a fairly simple condition, using our knowledge that the objective function gradient is zero at a local minimum point. Based on this, terminate the algorithm when the length of the gradient vector is 0.2 or less, that is, when \( (g^T g)^{1/2} \leq 0.2 \).

Also, you should normalize each direction vector to have a length of 1.0 before passing it to the line search algorithm. That is, you should divide the direction vector by its length before executing the line search. This is to make the value of \( \alpha \) a direct measure of the length of the line search step.

a. For this first part, you will use what you learned from Assignment 4 to select parameter values. Specifically, you should find a single set of fixed values for \( c_1 \) and \( c_2 \) (Wolfe parameters) and \( \alpha_{\text{max}} \) (Golden Section parameter) that allow all line searches to terminate for all of the following starting points: \((1.2, 1.2)^T\), \((-1.2, 1.0)^T\), \((10, 0)^T\), and \((1.5, 15)^T\).

Your report should discuss the process that you used to determine your set of parameters. Even though the process will probably have an element of trial-and-error to it, the process should be guided by theory and experimental results.

For each of the starting points, report at least the following information: final point in 2-space, final objective function value, number of steepest descent steps taken (high-level iterations), and \( \text{total} \) number of line search iterations (low-level iterations). That is, each line search will require a certain number of iterations to find an appropriate value for \( \alpha \), then you should add up all of these values for each time that the line search portion is executed to get this total. The latter two values are a measure of computational requirements: the number of steepest descent steps represents the number of objective function gradient evaluations needed, and the number of line search iterations represents...
the number of function evaluations and Wolfe condition tests needed.

You may include additional information that you think will shed light on the behavior of the optimization algorithm. One example of such information might be plots of objective function value as a function of iteration.

b. Once you have found a set of parameters that "works" for the above 4 starting points, try to find at least one starting point for which your parameter set results in a line search that does not terminate. Here are a couple of suggestions for such points: \((1 \ 15)^T\) and \((0 \ 15)^T\).

c. Finally, modify the algorithm so that it dynamically adjusts one or more of the three parameter values with the following dual goals: (i) allow the algorithm to complete (without intervention) for a larger set (perhaps all?) starting points, and (ii) improve the computational efficiency of the algorithm as measured by number of high-level iterations and/or number of low-level iterations. Describe your modification(s) and report on how well the above goals have been met.

2. In this second part, you will implement either a Newton-based Trust Region (TR) method or the Fletcher-Reeves (FR) variant of the Nonlinear Conjugate Gradient method (your choice) for the above objective function.

If you choose the TR method, use the Hessian \(F_k\) for the matrix \(B_k\). Also, use Algorithm 4.1 from the handout for updating the trust region radius. To solve the sub-problem at each iteration, you should use a full step whenever that is possible. If a full step is not possible, use a Cauchy point step. You will need to choose (or find) values for the three parameters of Algorithm 4.1: maximum value of radius (\(\hat{\Delta}\)), initial value of radius (\(\Delta_0\)), and sufficient improvement parameter (\(\eta\)). You will also need to choose Wolfe parameters (\(c_1\) and \(c_2\)), which can be the same values you used for the steepest descent part.

If you choose the FR method, you will only need to choose Wolfe parameters (\(c_1\) and \(c_2\)) and the maximum line search parameter (\(\alpha_{\text{max}}\)), which can be the same values you used for the steepest descent part.

Regardless of your choice, execute your chosen algorithm on the same 4 starting points and use the same high-level stopping condition as in part 1 above. Report the same results as in part 1a, with one exception. If you choose the TR method, report the breakdown between full-step and Cauchy-step iterations (how many of each), the average number of iterations per Cauchy-step line search, and the number of times that the algorithm discards the solution to the sub-problem (the last "else" in Algorithm 4.1). As in part 1, you should also discuss the behavior of your chosen algorithm, using whatever other data from the experiment that seems relevant to you.

3. Compare and contrast these two algorithms based on your experience working with them in this project. Include such factors as performance, ease of implementation, etc.