Introduction to Numerical Optimization

Assignment 2

April 2021

General Instructions:

- Submissions in pairs only.
- You are free to use either MATLAB or Python for any coding exercise.
- Sections labeled by [**Report**] should be included in your final report, and sections labeled with [**Code**] include a short coding task.
- Sections with no associated label include explanatory text only.
- More details about what should be included in your final report and your submission file are available in sections 3 and 4.
- Due date - May 21th, 2021 (23:59).

1 Convex Sets and Functions [Report]

Solve the following questions:

1.1 Question 1

Show that if f_1 and f_2 are convex functions on a convex domain C, then $g\left(x\right) = \max\limits_{i=1,2} f_i\left(x\right)$ is a convex function.

1.2 Question 2

Let $f\left(x\right)$ be a convex function defined over a convex domain C. Show that the level set $L = \left\{x \in C : f\left(x\right) \leq \alpha\right\}$ is convex.

1.3 Question 3 (Bonus - 5 Points)

Let $f\left(x\right)$ be a smooth and twice differentiable convex function over a convex domain C. Prove that $g\left(x\right) = f\left(Ax\right)$ is convex, where A is a matrix of appropriate size. Show explicitly that the hessian $\nabla^2 g$ is positive semi-definite for all $x \in C$. **Correction (18/5/2021)**: Let $f : \mathbb{R}^m \to \mathbb{R}$ be a smooth and twice differentiable convex function over $\mathbb{R}^m$, and let $A \in \mathbb{R}^{m\times n}$. Prove that $g : \mathbb{R}^n \to \mathbb{R}$, that is defined as $g\left(x\right) = f\left(Ax\right)$, is a convex function over $\mathbb{R}^n$. Show explicitly that the hessian $\nabla^2 g$ is positive semi-definite for all $x \in \mathbb{R}^n$.

1.4 Question 4

Formulate and prove the Jensen's inequality.

1.5 Question 5

Using Jensen's inequality, prove the arithmetic/geometric mean inequality:

$$\frac{x_1 + x_2 + \ldots + x_n}{n} \geq \sqrt[n]{x_1 \cdot x_2 \cdot \ldots \cdot x_n}$$

Where $\forall i : x_i > 0$.

2 Multivariate Optimization - Gradient Descent and and Newton's Method

In this section you will implement and experiment with two fundamental multivariate line-search optimization algorithms:

1. Gradient Descent
2. Newton's Method

For your convenience, you can find a great written reference on those two methods in Convex Optimization by Boyd and Vandenberghe on pages 466 (Gradient Descent) and 487 (Newton's Method).

Please use the following settings for the line search routine implementation:

- **Stopping Criteria**: $||\nabla f(x_k)|| < 10^{-5}$
- **Inexact Line Search**: Backtracking with Armijo Rule (Step-size decrease factor $\beta = 0.5$, Slope decrease factor $\sigma = 0.25$, Initial step-size $\alpha_0 = 1$)

2.1 Quadratic Form

In this section we're given the following quadratic form:

$$f(x) = \frac{1}{2} x^T Q x \tag{1}$$

Where $Q \in \mathbb{R}^{n \times n}$ and $x \in \mathbb{R}^n$.

2.1.1 Gradient and Hessian [Report]

Derive the analytical gradient and Hessian of equation 1.

2.1.2 Directional Derivative [Report]

Find the value of α which minimizes equation 1 along a line that passes through a given point x and heading in some direction d. In other words, find the minimum point of $g_{x,d} : \mathbb{R} \to \mathbb{R}$, which is given as follows:

$$g_{x,d}(\alpha) = f(x + \alpha d) \tag{2}$$

2.2 The Rosenbrock Function

In this section we're given a non-convex function known as the Rosenbrock function:

$$f\left(x_1, x_2, \dots, x_n\right) = \sum_{i=1}^{n-1} \left(\left(1 - x_i\right)^2 + 100\left(x_{i+1} - x_i^2\right)^2\right) \tag{3}$$

It was introduced by Howard H. Rosenbrock in 1960, and is used as a performance test problem for optimization algorithms.

2.2.1 Gradient and Hessian [Report]

Derive the analytical gradient and Hessian of equation 3.

2.3 Gradient Descent Method Implementation [Code]

Implement a gradient descent optimization routine, which can be configured (by an input argument) to execute inexact or exact line-search at each iteration. Use the step-length expression you have found in section 2.1.2 for the exact line-search implementation. You are free to design your code however you want.

2.4 Newton's Method Implementation [Code]

Implement a Newton's Method optimization routine, which can be configured (by an input argument) to execute inexact or exact line-search at each iteration. Use the step-length expression you have found in section 2.1.2 for the exact line-search implementation. You are free to design your code however you want.

2.4.1 LDL Decomposition and Forward/Backward Substitution

Use LDL decomposition to find the Newton direction, i.e., to solve the following set of linear equations $\nabla^2 f(x_k)\, d_k = -\nabla f(x_k)$. Once you replace the Hessian $\nabla^2 f(x_k)$ with its LDL^T decomposition, use forward/backward substitution to solve for d_k. We have provided you a routine that computes the Modified Cholesky Factorization of a symmetric matrix. There is a Python version (**mcholmz.py**) and a MATLAB version (**mcholmz.m**). For more information about solving linear system of equations, see mldivide for MATLAB, and numpy.linalg.solve for Python.

2.5 Find the Minimum of a Quadratic Form [Code and Report]

Complete the following items:

1. Run your **gradient descent** and **Newton's method** implementations to find an approximation of the minimum point of the quadratic form given in equation 1, with each of the following settings sets:

   - Settings Set #1:
     (a) Line-Search Type: **Exact**
     (b) Starting Point: $x_0 = (-0.2, -2)$
     (c) Quadratic Form Matrix: $Q = \begin{bmatrix} 10 & 0 \\ 0 & 1 \end{bmatrix}$
   - Settings Set #2:

(a) Line-Search Type: **Exact**

(b) Starting Point: $x_0 = (-0.2, -2)$

(c) Quadratic Form Matrix: $Q = \begin{bmatrix} 3 & 0 \\ 0 & 3 \end{bmatrix}$

- Settings Set #3:

  (a) Line-Search Type: **Exact**

  (b) Starting Point: $x_0 = (-2, 0)$

  (c) Quadratic Form Matrix: $Q = \begin{bmatrix} 10 & 0 \\ 0 & 1 \end{bmatrix}$

- Settings Set #4:

  (a) Line-Search Type: **Inexact**

  (b) Starting Point: $x_0 = (-0.2, -2)$

  (c) Quadratic Form Matrix: $Q = \begin{bmatrix} 10 & 0 \\ 0 & 1 \end{bmatrix}$

- Settings Set #5:

  (a) Line-Search Type: **Inexact**

  (b) Starting Point: $x_0 = (-0.2, -2)$

  (c) Quadratic Form Matrix: $Q = \begin{bmatrix} 3 & 0 \\ 0 & 3 \end{bmatrix}$

- Settings Set #6:

  (a) Line-Search Type: **Inexact**

  (b) Starting Point: $x_0 = (-2, 0)$

  (c) Quadratic Form Matrix: $Q = \begin{bmatrix} 10 & 0 \\ 0 & 1 \end{bmatrix}$

2. **For each settings set, plot your results as instructed in section 2.6.**

2.6 Plot Instructions

Each plot should include two figures, with the following details:

1. Left Figure: (3D Figure)
   - **Domain**: Your plot's domain should range from $(-2, -2)$ to $(2, 2)$.
   - **3D Surface**: You should plot your function's values above the its domain, which should form a 3D surface.
   - **Contour Lines**: You should plot your function's level sets (contour lines) on its domain.
   - **Trace of Descent**: You should plot the trace of descent of your algorithm on the function's domain. For example, if your algorithm converges into the function's minimum point by a series of points $x_0, x_1, x_2, \ldots$, then you should plot a piecewise-linear path that starts at x_0, then goes to x_1, next to x_2, and so on.
2. Right Figure: (2D Figure)
   This is a bird's-eye view of the left 3D figure. It should include the same elements as above, but excluding the 3D surface plot.

Please find an output example in figure 1.

2.6.1 MATLAB Implementation References

You might find the following links helpful:

- meshgrid - Create a grid of sample-points of the function's 2D domain.
- subplot - Create a figure with multiple subplots.
- contour - Plots the contour lines of a given functions evaluation.
- surf - Plots the 3D surface of a given function evaluation.
- surfc - Plots a 3D surface of a given function evaluation, with 2D contour lines below - This can also be achieved by using **surf** and **contour** instead.
- plot - Plots a 2D piecewise linear curve. Can be used to plot the Trace of Descent for both the 2D and 3D plots.

2.6.2 Python Implementation References

We have no experience with implementing this exact exercise in Python, but we believe that you might find the following links helpful:

- Surface Plots.
- Contour Plots.
- 2D Plots in 3D.
- 2D Contour Demo.
- 2D and 3D Axes in same Figure.

2.7 Find the Minimum of the Rosenbrock Function [Code and Report]

Use the gradient descent method and Newton's method to find the minimum point of the Rosenbrock function given in section 2.2, when defined over $\mathbb{R}^{10}$. Use the following settings for both methods:

1. Line-Search Type: **Inexact**
2. Starting Point: $x_0 = (0, 0, \ldots, 0) \in \mathbb{R}^{10}$.

For each method, plot the convergence curve $f(x_k) - f^*$ (the y-axis) as function of the iteration number k (the x-axis), where f^* is the optimal/minimal value of the Rosenbrock function (Take a minute to figure out the value of f^*, or just look it up in Wikipedia). Use logarithmic scale for the y-axis.

3 What Should Be Included In Your Written Report?

1. Solutions to all questions of section 1.
2. Detailed derivations of sections 2.1.1, 2.1.2 and 2.2.1.
3. All 12 plots of section 2.5. Please provide a **<u>short</u>** explanation for each plot that describe the observed results. If you can provide a common explanation for a group of plots, please do so.

4. The 2 plots of section 2.7. Please provide a **short** explanation for the difference between the plots.

4 Assignment Submission

1. Please name your report's PDF file as **hw2_report_ID1_ID2.pdf**, and submit it through Coursera. **Only ONE individual of each pair has to submit this file**.
2. Please zip the source code files that you have written in sections 2.3, 2.4, 2.5 and 2.7 into a file named **hw2_code_ID1_ID2.zip**, and submit your file through Coursera. **Only ONE individual of each pair has to submit this file**.
3. Please make a video recording with oral explanations to each of the following items:
   - Documentation to the code that you have written in section 2.3.
   - Documentation to the code that you have written in section 2.4.
   - Documentation to the code that you have written in section 2.5.
   - Documentation to the code that you have written in section 2.7.

   You are free to record each video on your own pace and style.
   Please zip all your videos into a file named **hw2_videos_ID.zip**, and upload it through Coursera. **Please note, EACH individual has to record his own set of videos, and upload them SEPARATELY to Coursera**. In case you experience problems with uploading the videos to Coursera, you can

alternatively upload them to YouTube as unlisted videos, and submit instead a text file named **hw2_videos_ID.txt** with links to your YouTube videos. You are free to remove the videos from YouTube by the end of the semester.

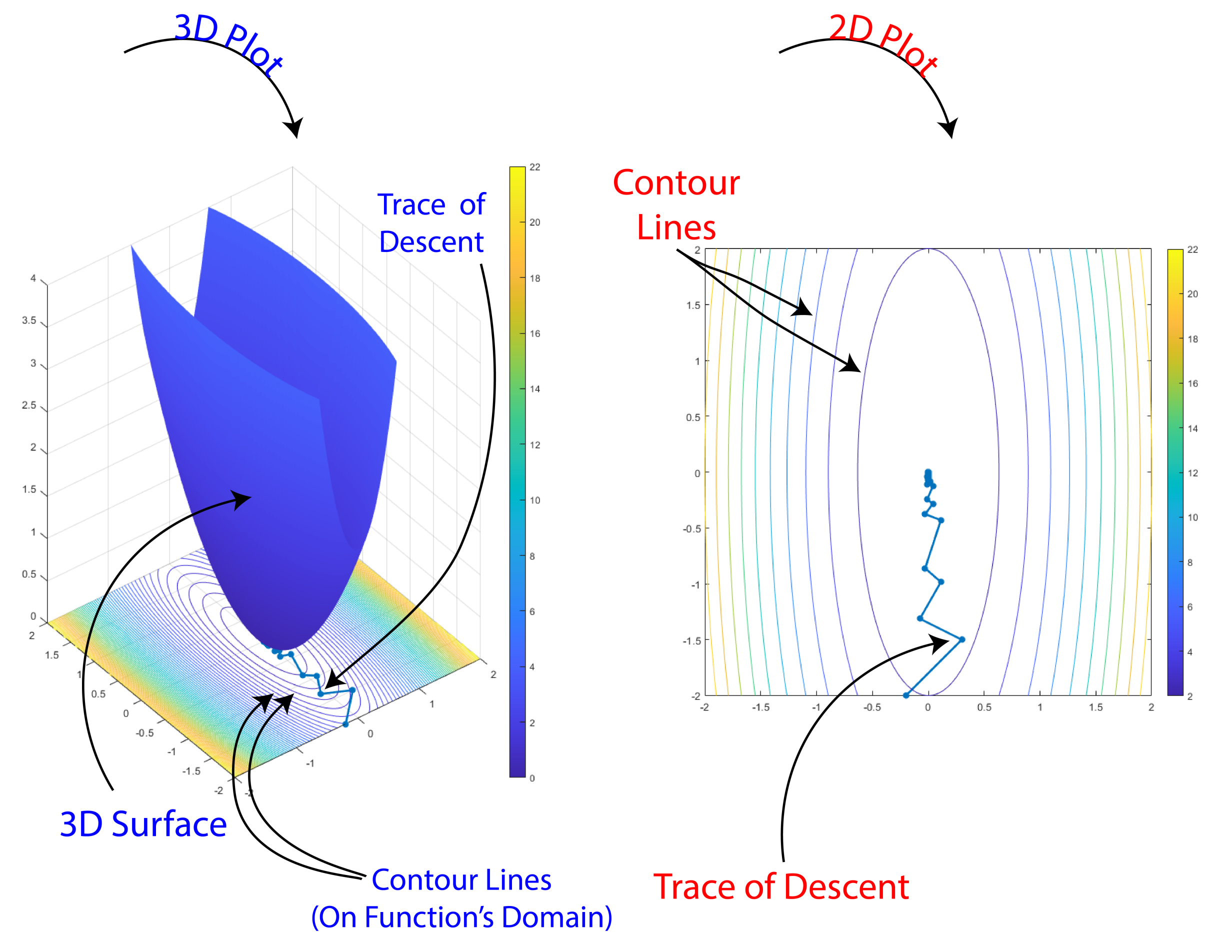

Figure 1: An example for a possible plot output for section 2.5. You should not necessary stick to this exact presented style. Design your plot by your taste, just make sure you include the 3 main elements - 3D Surface, Contour Lines and Trace of Descent.