Instructions: (Please read carefully and follow them!)

Try to solve all problems on your own. If you have difficulties, ask the instructor or TAs.

In this session, we will understand scalability issues noticed in the implementation of Newton's method to solve problems of the form $\min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x})$. Further in this lab, we shall design methods which can try to resolve the scalability issues associated with Newton's method.

The implementation of the optimization algorithms in this lab will involve extensive use of the `numpy` Python package. It would be useful for you to get to know some of the functionalities of `numpy` package. For details on `numpy` Python package, please consult https://numpy.org/doc/stable/index.html

For plotting purposes, please use `matplotlib.pyplot` package. You can find examples in the site https://matplotlib.org/examples/.

Please follow the instructions given below to prepare your solution notebooks:

- Please use different notebooks for solving different Exercise problems.
- The notebook name for Exercise 1 should be `YOURROLLNUMBER_IE684_Lab5_Ex1.ipynb`.
- Similarly, the notebook name for Exercise 2 should be `YOURROLLNUMBER_IE684_Lab5_Ex2.ipynb`, etc.
- Please post your doubts in MS Teams Discussion Forum channel so that TAs can clarify.

There are only 2 exercises in this lab. Try to solve all problems on your own. If you have difficulties, ask the Instructors or TAs.

Only the questions marked [**R**] need to be answered in the notebook. You can either print the answers using `print` command in your code or you can write the text in a separate text tab. To add text in your notebook, click `+Text`. Some questions require you to provide proper explanations; for such questions, write proper explanations in a text tab. Some questions require the answers to be written in LaTeX notation. Please see the demo video (posted in Lab 1) to know how to write LaTeX in Google notebooks. Some questions require plotting certain graphs. Please make sure that the plots are present in the submitted notebooks. Please include all answers in your `.pynb` files.

After completing this lab's exercises, click File → Download .ipynb and save your files to your local laptop/desktop. Create a folder with name `YOURROLLNUMBER_IE684_Lab5` and copy your `.ipynb` files to the folder. Then zip the folder to create `YOURROLLNUMBER_IE684_Lab5.zip`. Then upload only the `.zip` file to Moodle. There will be extra marks for students who follow the proper naming conventions in their submissions.

Please check the **submission deadline announced in moodle.**

Exercise 1: Scalability issues in Newton's method

In the last few labs, we have been discussing about gradient descent and Newton's methods to solve problems of the form $\min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x})$. We discussed the importance of scaling the direction along which the descent step is taken during the update. This scaling helped in better conditioning of the problem. Further we noticed that Newton's method might be fast in most cases when compared to gradient descent. This feature of Newton's method is attractive but is available only under certain assumptions.

In the implementation of Newton's method, we noticed that the inverse of Hessian is required at each iteration.

1. Consider the problem $\min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\top\mathbf{A}\mathbf{x}$ where A is a $n \times n$ symmetric positive definite matrix. Note that $\nabla^2 f(\mathbf{x}) = \mathbf{A}$. We will now investigate the computational effort needed to compute the inverse of the Hessian matrix A for large n values. First let us create a useful candidate for $\mathbf{A}$. Then we will check the time taken to compute the inverse of matrix $\mathbf{A}$.

2. [**R**] Choose $n \in \{500, 1000, 2500, 5000, 7500, 10000\}$ and find the time taken to compute the inverse of $\mathbf{A}$ for each possible value of n. Comment on your observations. Do you observe similar trends in the time taken to test the positive definiteness of $\mathbf{A}$? Comment and provide appropriate reasons for your observations.

3. However if we use Newton's method to solve $\min_{\mathbf{x}\in\mathbb{R}^n} 0.5\mathbf{x}^\top\mathbf{A}\mathbf{x}$ then the inverse of the Hessian remains invariant to the iterates $\mathbf{x}^k$ and hence can be computed once for all iterations. The situation becomes worse when the inverse of the Hessian needs to be computed afresh at every iteration. To demonstrate this fact, let us try to use Newton's method to solve the problem $\min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x})$ where $f(\mathbf{x}) = f(x_1, x_2, \ldots, x_n) = \sum_{i=1}^{n-1}\left[2(x_i^2 - x_{i+1})^2 + (x_i - 1)^2)\right]$.

4. [**R**] Implement Newton's method to solve the problem $\min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x})$ where we have $f(\mathbf{x}) = f(x_1, x_2, \ldots, x_n)$ $= \sum_{i=1}^{n-1}\left[2(x_i^2 - x_{i+1})^2 + (x_i - 1)^2)\right]$. Take $n \in \{1000, 2500, 5000, 7500, 10000\}$ and compute the time taken by the Newton's method to find the minimizer for each value of n. Take the starting point to be $\mathbf{x}^0 = (0, 0, \ldots, 0)$. Choose $\eta^k = 1, \forall k$ in the implementation of Newton's method. Prepare a graph where you plot the time taken by Newton's method vs n.

Exercise 2: BFGS Method

$k \leftarrow 0$;
Start with a suitable point $\mathbf{x}^k \in \mathbb{R}^n$;
while *not converged* **do**
  $\mathbf{p}^k \leftarrow -(B^k)\nabla f(\mathbf{x}^k)$;
  $\alpha^k \leftarrow \text{argmin}_{\alpha \geq 0} f(\mathbf{x}^k + \alpha \mathbf{p}^k)$;
  $\mathbf{x}^{k+1} \leftarrow \mathbf{x}^k + \alpha^k \mathbf{p}^k$;
  $\mathbf{s}^k = \mathbf{x}^{k+1} - \mathbf{x}^k$;
  $\mathbf{y}^k = \nabla f(\mathbf{x}^{k+1}) - \nabla f(\mathbf{x}^k)$;
  Implement the update rule to update B^{k+1}. ;
  $k \leftarrow k + 1$;
end

Algorithm 1: BFGS Method

1. **[R]** What is the initial choice of B (denoted by B^0)?
2. **[R]** Implement the modules of BFGS method to solve the problem $\min_{\mathbf{x} \in \mathbb{R}^n} f(\mathbf{x})$ where we have $f(\mathbf{x}) = f(x_1, x_2, \ldots, x_n) = \sum_{i=1}^{n-1} \left[2(x_i^2 - x_{i+1})^2 + (x_i - 1)^2)\right]$. Take $n \in \{1000, 2500, 5000, 7500, 10000\}$ and compute the time taken by the Newton's method to find the minimizer for each value of n. Take the starting point to be $\mathbf{x}^0 = (0, 0, \ldots, 0)$, and use $\eta^k = 1 \forall k$ in the implementation of BFGS Method. Compare the time taken by BFGS method against the time taken by Newton's method for each value of n. Plot the time taken vs n using different colors. Comment on your observations.
3. **[R]** Use backtracking line search with $\alpha^0 = 0.9, \rho = 0.5, \gamma = 0.5$ in the implementation of BFGS method. Take the starting point to be $\mathbf{x}^0 = (0, 0, \ldots, 0)$. Compare the time taken by BFGS method with backtracking line search against the time taken by Newton's method for each value of n. Plot the time taken vs n using different colors. Comment on your observations.