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 start with implementation of algorithms for solving nonlinear optimization problems without constraints. Today, we will discuss gradient descent to solve problems of the form $\min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x})$.

Gradient descent is one of the oldest algorithms (dating back to Cauchy), yet simple and popular in many scientific communities even today. Gradient descent works iteratively where in each iteration a suitable descent direction and an appropriate step length are chosen to update the current iterate.

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 `ROLLNUMBER-lab02-ex1.ipynb`. `ROLLNUMBER-lab02-ex2.ipynb`, etc for others.
- Please ask your doubts to TAs or instructors or post in Moodle Discussion Forum channel.
- You should upload on the .ipynb files on Moodle (one per exercise).

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 on paper. Write legible and to-the-point explanations. The work-sheet on which you write needs to be submitted before leaving the session.

Some other questions require plotting graphs (histograms, trajectories, level-sets etc). Please make sure that the plots are present in the submitted ipython notebooks.

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

Exercises today aim to help you learn the plotting of the level sets of the function and implementation of **optimization algorithms** such as first order (**Gradient Descent**) and second order (**Newton's method**). We will try zero-order methods later in the course.

Exercise 1 [5 marks]
Plot the good-looking visuals of level sets for each of the following functions in (1) to (4)-:

1. $f(\mathbf{x}) = f(x_1, x_2) = 100x_2^2 + 100x_1^4 - 200x_2x_1^2 + x_1^2 - 2x_1 + 1$
2. $f(\mathbf{x}) = f(x_1, x_2) = x_1^4 + x_2^4 - 20x_1^3 - 20x_2^3 + 100x_1^2 + 100x_2^2$
3. $f(\mathbf{x}) = f(x_1, x_2) = -\cos(x_1)\cos(x_2)\exp(-((x_1 - \pi)^2 + (x_2 - \pi)^2))$
4. $f(\mathbf{x}) = f(x_1, x_2) = 2x_1^2 - 1.05x_1^4 + \frac{x_1^6}{6} + x_1x_2 + x_2^2$
5. **[R]** What do you observe from the level sets so obtained in (1) to (4)?, Are they useful in characterizing the optimal solution visually, give reasoning via finding minima of each of the functions in (1) to (4) using usual first and second order optimality conditions.

Note-: Please use the theory references wisely related to the level sets, you need to explore the equivalent Python commands over the internet. Please use the axes range at least -4 to 4 and also choose levels in log or linear space wisely depending on particular function.

Exercise 2 [5 marks] **Gradient descent** is one of the oldest algorithms (dating back to Cauchy), yet simple and popular in many scientific communities even today. Gradient descent works iteratively where in each iteration a suitable descent direction and an appropriate step length are chosen to update the current iterate.

We will start with a procedure which helps to find a minimizer of the function $f(\mathbf{x}), \mathbf{x} \in \mathbb{R}^n$.

We will use the following gradient descent type algorithm:

Algorithm 1 Gradient Descent Procedure with Constant Step Length

Require: Starting point $\mathbf{x_0}$, Tolerance level τ, Step length η
1: Initialize $k = 0$
2: **while** $\|\nabla f(\mathbf{x_k})\|_2 > \tau$ **do**
3: $\quad \mathbf{x_{k+1}} \leftarrow \mathbf{x_k} - \eta\nabla f(\mathbf{x_k})$
4: $\quad k \leftarrow k + 1$
5: **Output:** $\mathbf{x_k}$

1. **[R]** What is the minimizer and minimum function value of $f(\mathbf{x}) = f(x_1, x_2) = (a + 1 - x_1)^2 + b \cdot (x_2 - x_1^2)^2$? Assume a is the last digit of your Roll Number and b is 100 if a is an even number, else b is 10. Use these values a, b for other problems that follow.
2. **[R]** Write the gradient of all the above four functions. You will need this gradient in the next steps.
3. With the starting point $x_0 = (-1.5, 1.5)$ and $\eta = 0.001$, we aim to analyze the behavior of the Algorithm 2 for different tolerance values (τ). We set $\tau = 10^{-p}$ where $p = 1, 2, \ldots, 13$. For each τ, record the final minimizer, objective function value at termination, and the number of iterations required for convergence in a tabular form. Generate a plot, illustrating the relationship between the number of iterations and τ values. Comment on the observations. Comment about the minimizers and objective function values obtained for different choices of the tolerance values.

4. Plot the level sets of the function in (1) and also plot the trajectory of the optimization on the same plot and report your observations. "In optimization, a trajectory refers to the path or sequence of points that a numerical optimization algorithm traverses while iteratively updating the solution in search of an optimal point".

Bibiography

(1) Y. Nesterov, Introductory lectures on convex optimization: A basic course. Springer Science & Business Media, 2003, vol. 87.
(2) https://people.eecs.berkeley.edu/~jiantao/227c2022spring/material.html