EPFL Fall 2019
Doctoral School
Civil and Environmental Engineering

Scientific Programming
for Engineers

Week 4 — *Homework: Conjugate Gradient*

The goal of this exercise is to get familiar with scipy and numpy module by implementing an iterative solver and visualizing the results. We will use Python for this exercise. A README file has to be included and should describe how to execute the code as well as its structure and dependencies. The following points will be considered for the grading:

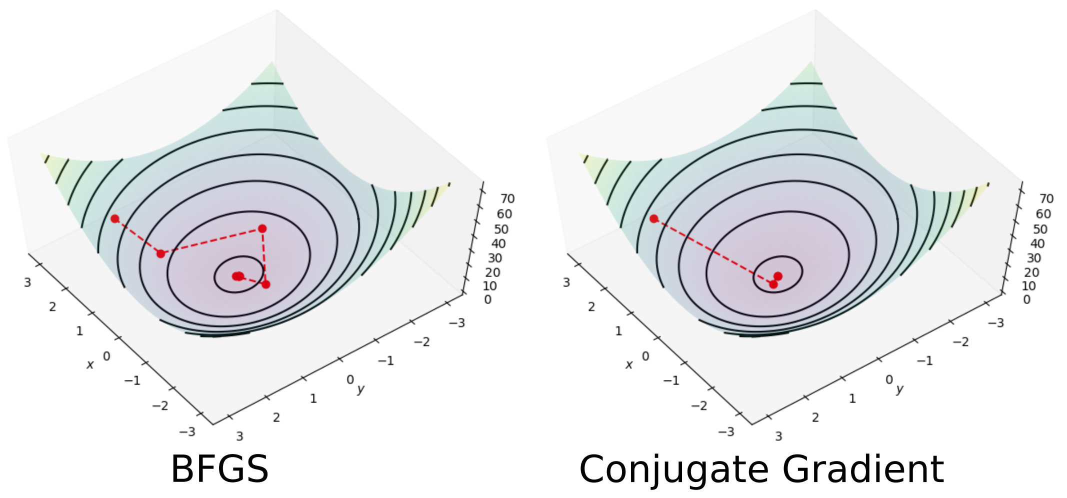
- Proper usage of git (meaningful comments, several commits with developments steps, use of .gitignore)
- Code works as intended
- Readability of the code (meaningful variable names, comments)
- Minimal documentation: README file

Exercise 1: *Scipy optimization*

In this exercise, we will use the optimization library of **Scipy** module. The goal of this exercise is to find the value of $\underline{x}^T = (x, y)$ which minimizes the quadratic function $S(\underline{x})$ defined by the equation:

$$S(\underline{x}) = \underline{x}^T \begin{pmatrix} 4 & 0 \\ 1 & 3 \end{pmatrix} \underline{x} - \underline{x}^T \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

Figure below shows the surface plot for $S(\underline{x})$. The red line shows the iterative path taken by two different solvers (BFGS, Conjugate gradient) to locate the minima of the function.



1. Create a **optimizer.py** file. In this file, create a function which takes $S(\underline{x})$ as a input argument in form of a python functor and returns the minimizer. Use `scipy.optimize.minimize` routine to solve the minimization problem. The type of solver can be specified through `method` parameter. For more help, refer to the documentation : Scipy optimize
2. Implement another routine which plots the $S(\underline{x})$ and the solution at each iteration step similar to figure shown above. Use **Matplotlib** module for plotting. To get the solution at each iterative step from `scipy.optimize.minimize`, you will have to use the parameter `callback`.

Exercise 2: ***Conjugate Gradient***

The conjugate gradient (CG) method is one of the many known iterative techniques for solving sparse symmetric definite systems of linear equations. For this exercise, implement the CG method to find the minimizer of the quadratic function

$$S(\underline{x}) = \frac{1}{2}\underline{x}^T \underline{\underline{A}}\, \underline{x} - \underline{x}^T \underline{b}$$

The minimizer of the above function is also the solution of the linear system of equation

$$\underline{\underline{A}}\, \underline{x} = \underline{b}$$

1. Create a **conjugate_gradient.py** file which implements the CG method. Use `einsum` from the **Numpy** module to perform different matrix and vector operations such as inner product, dot product etc. The implementation should be independent of matrix and vector size. For help, please refer to the wikipedia page Conjugate Gradient
2. Use `argparse` to input the coefficients of the function $S(x, y)$ defined in the previous exercise via the command-line and to construct a quadratic function usable to find the minimum of the function $S(x, y)$. Use argparse also to specify which type of minimizer to be used (yours or the one from scipy.optimize).
3. Use the plotting routine developed in the previous exercise to plot the solution at each iteration step and compare the method with the **Scipy** optimization method chosen by you in the previous exercise. Use argparse to make the plot optional.