Exercises 1st week

Tuesday, 25 February 2025

Learning outcomes

Exercise 1 (Programming):

- Understand how to use Python's API to create vectors and arrays
- Be able to create and edit rudimentary functions
- Be able to use basic plotting tools
- Be able to see how iterative linear solvers can depend on the tolerance parameters etc.
- Be able to use NumPy library in Python

Exercise 2 (Handwritten):

- Be able to choose the appropriate method for solving a linear system.
- Be able to apply basic linear algebraic principles

Exercise 1 (Programming)

The aim of this exercise is to get you familiar with using matrix operations.

a) Write a function *generate_tridiagonal_matrix()* that takes in as input an integer n and returns a tridiagonal matrix with -2 on the main diagonal and 1 on the upper and lower diagonals : $\boldsymbol{A} = \begin{matrix} -2 & 1 & 0 & 0 \\ 1 & \ddots & \ddots & \ddots \\ 0 & \ddots & \ddots & 1 \\ 0 & \ddots & 1 & -2 \end{matrix}$

b) Using this function, create an arbitrary linear system of size 100. Do this by first creating the tridiagonal matrix, **A** = *generate_tridiagonal_matrix(100)*. Then generate a random vector, **x**. Create the vector **b** using **Ax** = **b**. Now attempt to recover the vector **x** by using *LU Decomposition.* Compare the error of the recovered solution compared to the original solution.

<u>Hint:</u> Use the function numpy.*ones()* to generate the vectors of diagonals and then the function *scipy.sparse.diags()* to create the upper,lower and main diagonals before assembling the matrix.

Exercise 1 (Programming)

c) Using the same **A** and **b** from the previous exercise, try to solve the system using the Week_1_gauss_seidel.py file provided to you. Edit the file to take in as input the tolerance value and the starting vector **x0**. Play around with the tolerance value and the starting vector (generated randomly here but can also be all zeroes) to see how the number of iterations, and the accuracy of the solution, changes. If you would like to time the function, import the *time* module.

d) Edit the function Week_1_gauss_seidel.py to return the norm of the residual at each iteration along with the final solution. Once this is done, plot the norm and see how it evolves over the iterations. Does a normal plot show you the reduction in the residual norm clearly? What about a log plot ?

e) Loading files: Load the matrix **A** and vector **b** provided in the.npy files and solve the system using the aforementioned methods. Does the Gauss-Seidel method work? If not, why? (check the condition for convergence of the Gauss-Seidel method)

Exercise 2 (Handwritten)

A production line manufactured a large batch of cameras with faulty lenses that produce blurry images (Figure below, left). To avoid the recall of cameras, the company decided to upgrade the camera software and perform numerical deblurring of the images as soon as they are captured (Figure below, right). The deblurring is performed by solving a system of linear equations $Ax = b$, where A is the deblurring matrix that captures the lens imperfections.

Blurred image (b)

Deblurred image ($x = A^{-1}b$)

Exercise 2 (Handwritten)

a) Assuming that the deblurring matrix is dense and non-symmetric, propose a method that would be the most efficient for this task. Explain your choice.

b) Apply the proposed method to compute $\boldsymbol{x}$ for

$$\boldsymbol{A} = \begin{bmatrix} 3 & 4 & -5 \\ 6 & -3 & 4 \\ 8 & 9 & -2 \end{bmatrix} \quad \boldsymbol{b_1} = \begin{bmatrix} 1 \\ 9 \\ 9 \end{bmatrix} \quad and \quad \boldsymbol{b_2} = \begin{bmatrix} 3 \\ 5 \\ 4 \end{bmatrix}$$

c) Compute the determinant of the matrix A.