Programming Assignment Requirements:

- All parts of the assignment should be emailed to me. My email can be found on Blackboard.
- The subject line must contain the following text: *MATH7015 - Assignment #1*
- Send me a copy of your code and a written report in either PDF or Word format.
- Provide enough detail in your written report so I can understand what you have done and why without having to search through your code.
- Do not put your source code in your written report. It should be in separate files so I can easily run your code.
- Make any plots and figures professional looking and readable. Label things appropriately (axes, legends, etc.) and present the data is a clear manner.
- Make your code clean and readable, including comments where appropriate.
- Do not cut and paste code. You should reuse your code by placing common code fragments into their own functions.
- Do not copy code that you find online. For the assignments, you are expected to create your own solution to the problems. It is not difficult to detect plagiarised code.
- Try to make your code efficient but do not spend too much time optimizing the code for speed.
- Do **NOT** use symbolic toolboxes or libraries.
- Do not assume that I will have any toolboxes or libraries besides the vanilla MATLAB version or NumPy and SciPy for Python.

---

1. [5 points] For the first programming assignment, you will test your MATLAB/Python installation. You only have to do one of the following:

   For Python, do the following:

   (a) Install a version of Python 3.7. I highly suggest using Anaconda.
   (b) Download the book Numerical Python by Robert Johannson from the SpringerLink website. You must be on the UC network to download it for free.
   (c) Practice with Jupyter, qtConsole, VSCode, and the Spyder IDE if you are not already familiar with Python or these tools.
   (d) Run the Python test script (PythonTest.py) and copy its output into your report.

   For MATLAB, do the following:

   (a) Download the book Numerical Computing with MATLAB by Cleve Moler.
   (b) Read the introduction to MATLAB if you are not already familiar.
   (c) Run the MATLAB test script (MATLAB_Test.m) and copy its output into your report.

2. [20 points] Write a function that computes the solution $\boldsymbol{x} \in \mathbb{R}^N$ of the linear problem $\mathbf{A}\boldsymbol{x} = \boldsymbol{b}$ for a given square matrix $\mathbf{A} \in \mathbb{R}^{N\times N}$ and a set of vectors $\boldsymbol{b}_k \in \mathbb{R}^N$ for $k = 1, \ldots, M$ using the built-in linear solver (`mldivide` (backslash operator) in MATLAB or `numpy.linalg.solve` in NumPy).
   (a) This function must be called `MultiLinearSolve`.
   (b) It takes two parameters, `A` and `bs`, which are a $N \times N$ array containing the matrix $\mathbf{A}$ and a $N \times M$ array containing the vectors $\boldsymbol{b}_k$ for $k = 1, 2, \ldots, M$.
   (c) It will return a $N \times M$ array containing the solution vectors $\boldsymbol{x}_k$ for $k = 1, 2, \ldots, M$.

3. [20 points] Write a function that computes the solution $\boldsymbol{x} \in \mathbb{R}^N$ of the linear problem $\mathbf{A}\boldsymbol{x} = \boldsymbol{b}$ for a given square matrix $\mathbf{A} \in \mathbb{R}^{N\times N}$ and a set of vectors $\boldsymbol{b}_k \in \mathbb{R}^N$ for $k = 1, \ldots, M$ using the built-in LU factorization code (`lu` (backslash operator) in MATLAB or `scipy.linalg.lu` in SciPy).
   (a) This function must be called `MultiLUSolve`.
   (b) It takes two parameters, `A` and `bs`, which are a $N \times N$ array containing the matrix $\mathbf{A}$ and a $N \times M$ array containing the vectors $\boldsymbol{b}_k$ for $k = 1, 2, \ldots, M$.
   (c) It will return a $N \times M$ array containing the solution vectors $\boldsymbol{x}_k$ for $k = 1, 2, \ldots, M$.
   (d) This function should compute the LU factorization of the matrix $\mathbf{A}$ once and then use forward and backward substitution to solve for $\boldsymbol{x}$. You can either use a built-in triangular matrix solver or write your own forward/backward substitution codes but this function must be faster than `MultiLinearSolve` for large N and M.