The Python code I have attached is designed to solve system of equations denoted as:

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

It is designed to solve an arbitrary banded sparse system using TDMA algorithm. TDMA algorithm is a simplified form of Gaussian elimination that can be used to solve tridiagonal system which can be expressed in both equations and matrix as shown below.

$$\begin{bmatrix} b_1 & c_1 & & & 0 \\ a_2 & b_2 & c_2 & & \\ & a_3 & b_3 & \cdot & \\ & & \cdot & \cdot & c_{n-1} \\ 0 & & & a_n & b_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ x_n \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ \cdot \\ \cdot \\ d_n \end{bmatrix} \qquad a_i x_{i-1} + b_i x_i + c_i x_{i+1} = d_i$$

where a_1=0 and c_n=0.

The algorithm I chose for my code have two phases: forward elimination phase and backward substitution phase as shown below.

Forward elimination phase

for k = 2 step until n do

$$m = \frac{a_k}{b_{k-1}}$$
$$b_k = b_k - mc_{k-1}$$
$$d_k = d_k - md_{k-1}$$

end loop (k)

Backward substitution phase

$$x_n = \frac{d_n}{b_n}$$

for k = n-1 stepdown until 1 do

$$x_k = \frac{d_k - c_k x_{k+1}}{b_k}$$

end loop (k)

```python
# -*- coding: utf-8 -*-
"""
FILE TDMAsolver.py
Solves TDMA

Name: Kyu Mok Kim
Student Number: 998745381

MIE1210 Project 1

@author: kimkyu4

This code is designed to solve the system of equations denoted as: Ax=b,
using TDMA method using a,b,c,d arrays.
Refer to my report for detailed explanation.
"""

#imprting numpy for convenience
import numpy as np

#TDMA Solver
def TDMAsolver(aa,bb,cc,dd):
    a,b,c,d = map(np.array, (aa,bb,cc,dd))  #preventing overwriting original
    n = len(b)          #number of rows/equations

    for i in range (1,n):
        e = a[i-1]/b[i-1]
        b[i]=b[i]-e*c[i-1]
        d[i]=d[i]-e*d[i-1]

    x = b
    x[-1]=d[-1]/b[-1]

    for j in range (n-2,-1,-1):
        x[j]=(d[j] - c[j] *x[j+1])/b[j]

    return x
```

The Python code for the TDMA solver is attached and also shown above.

In my code I have utilized NumPy, which is a library for Python which allows creating and dealing with multi-dimensional arrays and matrices along with high-level mathematical functions to operate on these arrays.My code also allows the rank (number of equations) to vary and also the diagonal coefficients to vary.

This method, as it is a simplified form, requires less calculations and less storage than Gaussian Elimination. The cost per unknown is independent of the number of unknowns, which leads to good scaling with respect to iterative methods. However, there are some disadvantages to this method: round off error is significant, may not be suitable for non-linear problems unless equations can be linearized, and not stable in general but it is stable if the matrix is diagonally dominant ($|b_i| > |a_i| + |c_i|$) or symmetric positive definite.

To verify the code works, I have used the example from class to put in the following values:

```
##For checking purposes using the example given in class
aa = np.array([-5.,-5.,-5.,-5.])      #diagonal column below the middle
bb = np.array([20.,15.,15.,15.,10.])   #middle diagonal column value
cc = np.array([-5.,-5.,-5.,-5.])      #diagonal column above the middle
dd = np.array([1100.,100.,100.,100.,100.]) #the right column (b)

print (TDMAsolver(aa,bb,cc,dd))
```

Which represents

$$\begin{bmatrix} 20 & -5 & 0 & 0 & 0 \\ -5 & 15 & -5 & 0 & 0 \\ 0 & -5 & 15 & -5 & 0 \\ 0 & 0 & -5 & 15 & -5 \\ 0 & 0 & 0 & -5 & 10 \end{bmatrix} \begin{bmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \\ \phi_4 \\ \phi_5 \end{bmatrix} = \begin{bmatrix} 1100 \\ 100 \\ 100 \\ 100 \\ 100 \end{bmatrix}$$

The output of the code gave:

```
In [6]: runfile('//srvd/Homes$/kimkyu4/Downloads/MIE1210 CFD HT Project/
TDMAsolver.py', wdir='//srvd/Homes$/kimkyu4/Downloads/MIE1210 CFD HT Project')
[ 64.22764228  36.91056911  26.50406504  22.60162602  21.30081301]
```

Which matches the answer to the example:

$$\begin{aligned} \phi_5 &= 0 + 21.30 \\ &= 21.30 \\ \phi_4 &= 0.3816 \times 21.30 + 14.4735 \\ &= 22.60 \\ \phi_3 &= 0.3793 \times 22.60 + 17.9308 \\ &= 26.50 \\ \phi_2 &= 0.3636 \times 26.50 + 27.2727 \\ &= 36.91 \\ \phi_1 &= 0.25 \times 36.91 + 55 \\ &= 64.22 \end{aligned}$$