Assignment: Solving a System of Linear Equations

Gauss Elimination without Pivoting

Instructions:

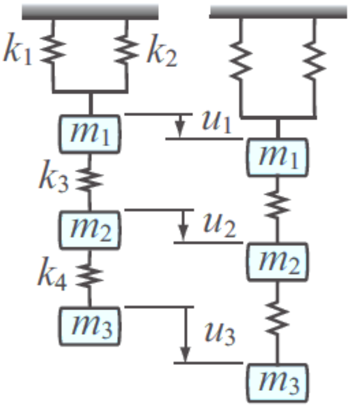
- Declare and define the numerical method functions in your header file “myNM.h”, “myNM.cpp”, “myMatrix.h”, “myMatrix.cpp”, respectively.
- The main program file, ‘Assignmen_gaussElim.cpp’, contains the main function and calls yours NM functions to solve the assignment problems.
- You must submit the report and the program files: “myNM.h”, “myNM.cpp”, “myMatrix.h”, “myMatrix.cpp”, “Assignmen_gaussElim.cpp” on Hisnet.
- You can use dynamic memory allocation for 2D array and structures for matrix variables. It is not mandatory.
- Check if your function is accepting only square matrices
- You should insert exceptional/error handling(e.g. giving error message when square matrix is not used, div by zero etc)
- For this assignment, you do not need to use partial pivoting.

Problem: Solve the following linear systems of Ax=b, using Gauss elimination [20pt]

Q1. Determine the displacement of the three masses

They are in the equilibrium states, and u_1,u_2, u_3 are the relative displacement for each mass.

m_1=2 kg, m_2=3kg, m_3=1.5kg , g=9.81 m/s^2
k_1= 30 [N/m], k_2=25 [N/m], k_3=20 [N/m], k_4=15 [N/m]



$$
\begin{aligned}
(k_1+k_2+k_3)u_1 - k_3u_2 &= m_1g \\
-k_3u_1 + (k_3+k_4)u_2 - k_4u_3 &= m_2g \\
-k_4u_2 + k_4u_3 &= m_3g
\end{aligned}
$$

Procedure

- Review how to define, initialize and use 2D arrays and how to pass 2D array to a function in C/C++. (See Tutorial #2)
- If you want, you can apply partial pivoting in the program.
- Add exceptional/error handling for when **A** is not square, dimension of **A**, **b** are not appropriate, division by zero and so on
- *We will only consider square matrix A (n by n) for this assignment.

1. Gauss elimination method without partial pivoting

First write a pseudocode

Create a C/C++ function that processes the Gauss elimination without partial pivoting

- Input: matrix **A**(n x n), vector **b**(n x1)
- Output: matrix **U**, vector **d** (modified b after Gauss elimination).
  (option)permutation matrix **P**
- You can use either 2D array with fixed dimension or structure Matrix
- Declare in “myNM.h” and define in “myNM.c”

```
void gaussElim(Matrix _A, Matrix _b, Matrix _U, Matrix _d);
```

Show you code here

2. Create back-substitution function to solve Ux=d

First write a pseudocode

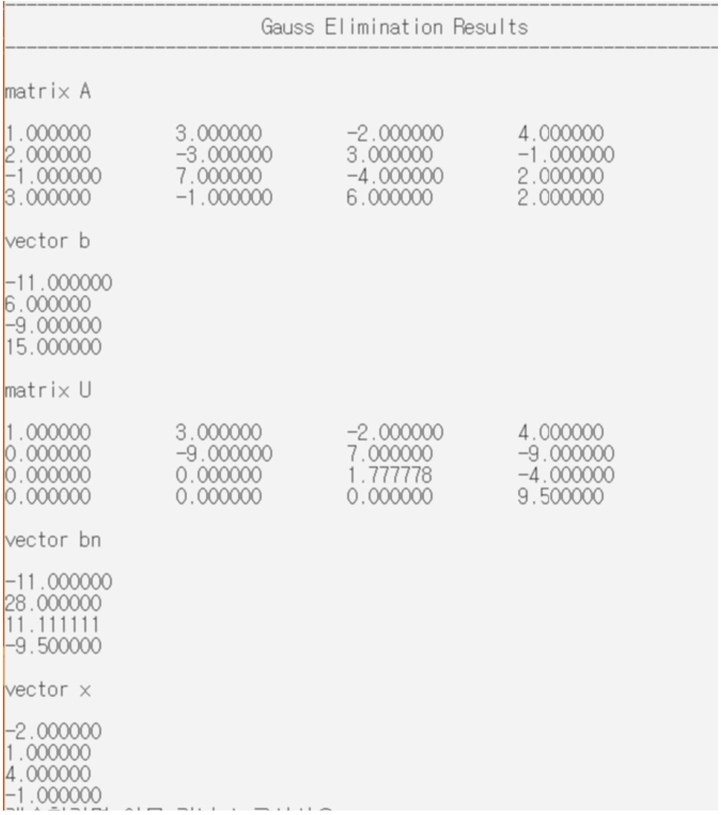
Then, create a C/C++ function

```
void backsub(Matrix _U, Matrix _d, Matrix _x);
```

- Input: matrix **U**(n x n), vector **d**(n x1)
- Output: vector **x** (nx1)
- You can use either 2D/1D array with fixed dimension or structure Matrix
- Declare in “myNM.h” and define in “myNM.c”

3. Show the output results

(example)



```
---------------------------------------------------------------------------
                         Gauss Elimination Results
---------------------------------------------------------------------------

matrix A

1.000000        3.000000        -2.000000       4.000000
2.000000        -3.000000       3.000000        -1.000000
-1.000000       7.000000        -4.000000       2.000000
3.000000        -1.000000       6.000000        2.000000

vector b

-11.000000
6.000000
-9.000000
15.000000

matrix U

1.000000        3.000000        -2.000000       4.000000
0.000000        -9.000000       7.000000        -9.000000
0.000000        0.000000        1.777778        -4.000000
0.000000        0.000000        0.000000        9.500000

vector bn

-11.000000
28.000000
11.111111
-9.500000

vector x

-2.000000
1.000000
4.000000
-1.000000
```

4. Check your answer with the output from MATLAB