**Assignment: Solving a System of Linear Equations**

**With Gauss Elimination**

# What you need to submit

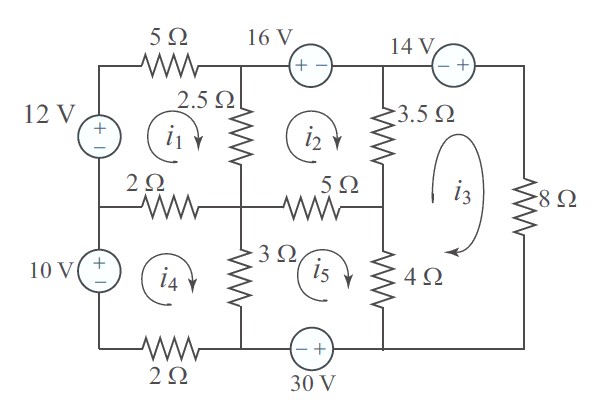
* Submit the report+source files as a zip file: **Assignment\_gaussElim\_Name\_ID.zip**
* **Report:** with pseudocode, output result and source code as instructed
* **Src Code Files:**
  + - Assignment\_gaussElim\_ID.cpp
    - myNP.h, myNP.cpp, myMatrix.h, myMatrix.cpp

**Tip:**

* Review how to use structure type for using 2D array
  + [Tutorial & Exercise](https://ykkim.gitbook.io/ec/c-programming/c-programming-review/dynamic-alloc)
  + [Tutorial Video](https://youtu.be/wJcecYauQvk)
* You should insert exceptional/error handling(e.g. giving error message when square matrix is not used, div by zero etc). \*We will only consider square matrix A (n by n) for this assignment.

# Problem: Solve the following linear systems of Ax=b

**Q1. Analyze the current in each mesh in the given circuit.**



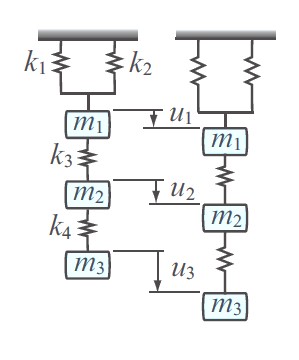
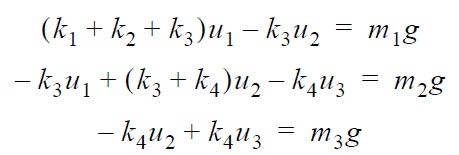
텍스트이(가) 표시된 사진

자동 생성된 설명

**Q2. Determine the displacement (u1, u2, u3)**

They are in the equilibrium states, and *u1,u2, u3* are the relative displacement for each mass.

m1=2 kg, m2=3kg, m3=1.5kg , g=9.81 m/s^2 k1= 30 [N/m], k2=25 [N/m], k3=20 [N/m], k4=15 [N/m]



# Procedure

**1. Gauss elimination method without partial pivoting**

## 1a) Write a pseudocode for Gauss Elimination [5pt]

### // pseudocode goes here

텍스트이(가) 표시된 사진

자동 생성된 설명

## 1b) Create a C function for Gauss elimination without partial pivoting [10pt]

void gaussElim(Matrix A, Matrix b, Matrix U, Matrix d);

### - Input: matrix **A**(n x n), vector **b**(n x1) - Output: matrix **U**, vector **d** // Ax=b → Ux=d

for (k = 0; k < U.rows; k++)

{

for (i = k + 1; i < U.rows; i++) {

if (U.at[k][k] == 0) {

printf("There's 0 in pivot element"); // 예외 처리 (0으로 나눌 때)

return;

}

m = U.at[i][k] / U.at[k][k];

for (j = k; j < U.cols; j++) {

U.at[i][j] = U.at[i][j] - (m \* U.at[k][j]);

}

d.at[i][0] = d.at[i][0] - m \* d.at[k][0];

} // Gauss-Elimination

}

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

## 2a) First write a pseudocode [5pt]

텍스트이(가) 표시된 사진

자동 생성된 설명

## 2b) Then, create a C/C++ function for back substitution [10pt]

void backsub(Matrix U, Matrix d, Matrix x);

### - Input: matrix **U**(n x n), vector **d**(n x1) - Output: vector **x** (nx1)

### // your code

Matrix backSub(Matrix U, Matrix d, Matrix x)

{

Matrix Out = createMat(d.rows, 1);

int i, j;

for (i = U.rows - 1; i >= 0; i--)

{

double sub = 0;

for (j = i + 1; j < U.rows; j++)

{

sub += U.at[i][j] \* Out.at[j][0];

}

Out.at[i][0] = (d.at[i][0] - sub) / U.at[i][i];

}

return Out;

}

**3. Show the output results for Q1 and Q2 systems [10pt]**

## - matrix A, b, U, d

- solution **x**

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

**4. Compare your answer with MATLAB [5pt]**

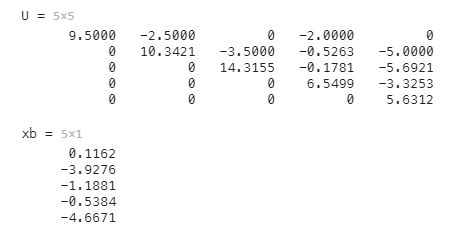
### Write Matlab code

### // your MATLAB code

텍스트이(가) 표시된 사진

자동 생성된 설명텍스트이(가) 표시된 사진

자동 생성된 설명

// MATLAB output

텍스트이(가) 표시된 사진

자동 생성된 설명

# Extra-credit (Optional)

**5. Gauss elimination method with partial pivoting [30pt]**

Modify your Gauss elimination code in include partial pivoting

void gaussElim(Matrix A, Matrix b, Matrix U, Matrix d, Matrix P);

- Input: matrix **A**(n x n), vector **b**(n x1) - Output: matrix **U**, vector **d,** matrix **P** // PAx=Pb → Ux=d

## a) First write a pseudocode [10pt]

### // your pseducode 텍스트이(가) 표시된 사진 자동 생성된 설명

## b) Then, create a C/C++ function [15pt]

### // your code

void gaussElim(Matrix A, Matrix b, Matrix U, Matrix d, Matrix P)

{

if (A.rows != A.cols || b.rows != A.rows || A.rows != U.rows || A.cols != U.cols || A.rows != P.rows || A.cols != P.cols || d.rows != U.rows || b.rows != d.rows || b.cols != 1 || d.cols != 1) {

printf("Square Matrix is not Used || Vector's rows are not same with Matrix's rows || Vector's cols are not 1");

return;

} // 예외 처리 (행렬, 벡터 에러)

int i, j, k;

double max\_a;

double temp;

double m;

copyVal(eye(P.rows, P.cols), P); // Permutation matrix

copyVal(A, U);

copyVal(b, d);

for (k = 0; k < U.rows; k++)

{

max\_a = fabs(U.at[k][k]);

for (i = k+1; i < U.rows; i++)

{

if (max\_a < fabs(U.at[i][k]))

{

max\_a = fabs(U.at[i][k]); // 가장 큰 행렬 찾기

for (j = 0; j < U.rows; j++) {

temp = U.at[i][j];

U.at[i][j] = U.at[k][j];

U.at[k][j] = temp;

temp = P.at[i][j];

P.at[i][j] = P.at[k][j];

P.at[k][j] = temp;

}

}

} // 행렬 재정렬

for (i = k + 1; i < U.rows; i++) {

if (U.at[k][k] == 0) {

printf("There's 0 in pivot element"); // 예외 처리 (0으로 나눌 때)

return;

}

m = U.at[i][k] / U.at[k][k];

for (j = k; j < U.cols; j++) {

U.at[i][j] = U.at[i][j] - (m \* U.at[k][j]);

}

d.at[i][0] = d.at[i][0] - m \* d.at[k][0];

} // Gauss-Elimination

}

### }

## c) Validate your code with the test matrix A [5pt]

텍스트이(가) 표시된 사진

자동 생성된 설명 텍스트이(가) 표시된 사진

자동 생성된 설명텍스트이(가) 표시된 사진

자동 생성된 설명텍스트이(가) 표시된 사진

자동 생성된 설명