ECE 204 Simulation Assignment 1

Phoebe Luo
University of Waterloo

Introduction

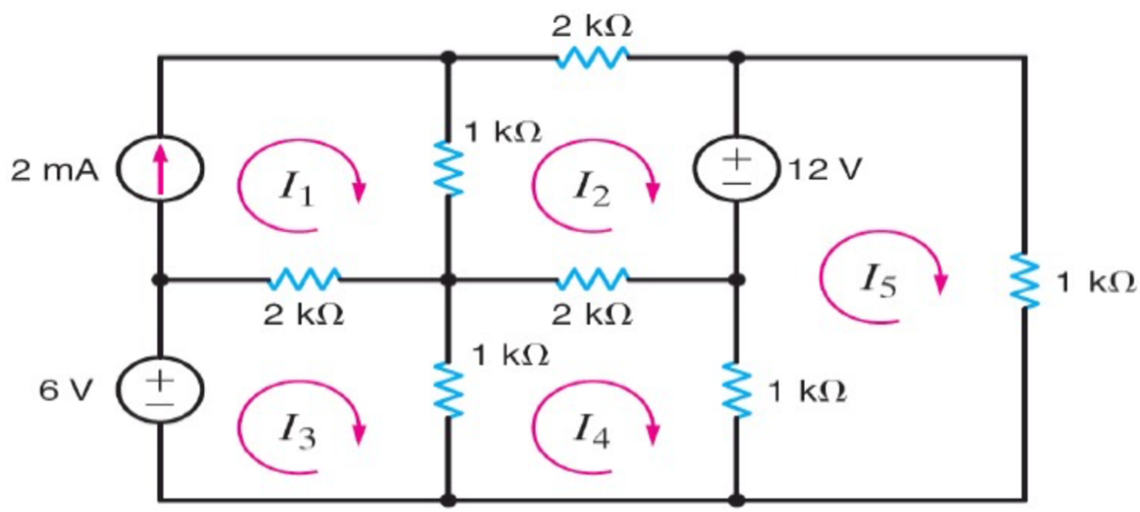
The main problem of Simulation Assignment 1 is to solve the current of the given circuit and also develop the algorithm to be general enough for any n*n matrix using algorithms:

a) Simple Matrix Inversion

b) Gaussian Elimination with Partial Pivoting

c) Gauss-Seidel Iteration

The circuit is



The system of equations will be

(1) $I_1 = 2mA$

$I_1 = 2$

(2) $1(I_2 - I_1) + 2(I_2) + 12 + 2(I_2 - I_4) = 0$

$5I_2 - I_1 - 2I_4 = -12$

(3) $-6 + 2(I_3 - I_1) + 1(I_3 - I_4) = 0$

$3I_3 - 2I_1 - I_4 = 6$

(4) $1(I_4 - I_3) + 2(I_4 - I_2) + 1(I_4 - I_5) = 0$

$4I_4 - I_3 - 2I_2 - I_5 = 0$

(5) $-12 + 1I_5 + 1(I_5 - I_4) = 0$

$2I_5 - I_4 = 12$

The matrix for the above circuit will be

```
1 0 0 0 0
-1 5 0 -2 0
-2 0 3 -1 0
0 -2 -1 4 -1
0 0 0 -1 2
```

A.txt

```
2
-12
6
0
12
```

B.txt

Results and Discussion

Part 1 – Solve system of equations with the required methods, number of sig figs, and the absolute approximate relative error.

- The code for general format, file reading, and global variable:

```
format shortg
clc;
A = readmatrix("A.txt");
B = readmatrix("B.txt");
row_num = size(A, 1); % total number of rows
```

a) Simple Matrix Inversion

- The code is

```
% Simple Matrix Inversion
disp("****** Simple Matrix Inversion ******")
S1 = inv(A)*B; % solution matrix
S1 = round(S1,5,'significant');
disp(S1);
```

- The output is

```
        2
  -1.0986
   4.0845
   2.2535
   7.1268
```

b) Gaussian Elimination with Partial Pivoting

- The code is

```
% Gaussian Elimination with Partial Pivoting
disp("****** Gaussian Elimination with Partial Pivoting ******")
% forward elimination
A2 = [A B]; % combine A, B together to do operation
row_swap = 1; % keep track of the row number that will be swapped with the
cur_row
```

```
ope_num = row_num - 1; % the number of operations in total
cur_row = 2; % keep track of the current row of subtraction for this
operation (initially at row 2)
cur_col = 1; % keep track of the current column of subtraction (initially at
column 1)
count = cur_row; % keep track of the current row being manipulated

for a = 1:ope_num
    count = cur_row;
    while count < (row_num+1)
          A2(count,:) = A2(count,:)-
A2(a,:).*(A2(count,cur_col)/A2(a,cur_col));
          count = count + 1;
    end
    % row swap
    [~,row_swap] = max(abs(A2(cur_row:row_num,cur_col)));
    temp = A2(cur_row,:);
    A2(cur_row,:) = A2(row_swap+cur_row-1, :);
    A2(row_swap+cur_row-1,:) = temp;
    cur_row = cur_row + 1;
    cur_col = cur_col + 1;
end

% back substitution
S2 = zeros (row_num,1); % solution matrix
x = row_num; % x for solving from the bottom right diagonally
num_sub = 0; % number of subtraction required for A(x, row_sum)
cur_sub = num_sub; % keep track of needed subtraction left for A(x, row_sum)

while x > 0
    while cur_sub > 0
        n = row_num - (cur_sub - 1);
        A2(x,row_num+1) = A2(x, row_num+1) -  S2(n, 1)*A2(x, n);
        cur_sub = cur_sub - 1;
    end
    S2(x,1) = A2(x,row_num+1)/A2(x,x);
    num_sub = num_sub + 1;
    cur_sub = num_sub;
    x = x - 1;
end
S2 = round(S2,5,'significant');
disp(S2);
```

- The output is

```
        2
  -1.0986
   4.0845
   2.2535
   7.1268
```

c) Gauss-Seidel Iteration

- The code is

  (example code produces output of error = 1%. To produce output of approximate relative error less than 1, 0.5, 0.1 and 0.01%:

  change line `error_req = 1; % error required`

  to `error_req = 0.5; % error required`

  `error_req = 0.1; % error required`

  `error_req = 0.01; % error required`)

```
% Gauss-Seidel Iteration
disp("****** Gauss-Seidel Iteration ******")
% 1% error
S3_cur = zeros(row_num,1); % store values in the current operation
S3_last = zeros(row_num,1); % store values in the last operation
ite_req = 0; % keep track of number of iterations required
error_req = 1; % error required
stop = 0; % change to 1 when the loop can be stopped

while stop == 0
    ite_req = ite_req + 1;
    for b = 1:row_num
        x = B(b,1);
        for c = 1:row_num
            if c ~= b
                x = x - A(b,c)*S3_cur(c,1);
            end
        end
        x = x/A(b,b);
        x = round(x,5,'significant');
        S3_cur(b,1) = x;
    end
    % check for error
    count_2 = 0; % count if all the elements fits error required
    for d = 1:row_num
        error = abs((S3_cur(d,1) - S3_last(d,1))/S3_cur(d,1)*100);
        if error < error_req
            count_2 = count_2 + 1;
        end
        if count_2 == row_num
            stop = 1;
        end
    end
```

```
    S3_last = S3_cur;
end

disp(S3_cur);
disp(ite_req);
```

- The outputs are

2	2	2	2
-1.1031	-1.1004	-1.0989	-1.0986
4.0808	4.083	4.0843	4.0845
2.249	2.2517	2.2532	2.2535
7.1245	7.1259	7.1266	7.1267
8	9	11	13
(1%)	(0.5%)	(0.1%)	(0.01%)

Part 2 - Assume the values of all resistors increased by 5%, calculate the new value of the loop currents using Gaussian elimination, and evaluate if the system of equations is in ill-condition.

- The system of equations will be

1. $I_1 = 2mA$

   $I_1 = 2$

2. $1.05(I_2 - I_1) + 2.1(I_2) + 12 + 2.1(I_2 - I_4) = 0$

   $-1.05 I_1 + 5.25 I_2 - 2.1 I_4 = -12$

3. $-6 + 2.1(I_3 - I_1) + 1.05(I_3 - I_4) = 0$

   $-2.1 I_1 + 3.15 I_3 - 1.05 I_4 = 6$

4. $1.05(I_4 - I_3) + 2.05(I_4 - I_2) + 1.05(I_4 - I_5) = 0$

   $-2.1 I_2 - 1.05 I_3 + 4.2 I_4 - 1.05 I_5 = 0$

5. $-12 + 1.05 I_5 + 1.05(I_5 - I_4) = 0$

   $-1.05 I_4 + 2.1 I_5 = 12$

- The matrix for the new circuit will be:

```
1 0 0 0 0
-1.05 5.25 0 -2.1 0
-2.1 0 3.15 -1.05 0
0 -2.1 -1.05 4.2 -1.05
0 0 0 -1.05 2.1
```

A.txt

```
2
-12
6
0
12
```

B.txt

- The code is the same as Gaussian Elimination with Partial Pivoting

- The output is

```
      2
-1.0101
 3.9678
 2.1891
 6.8089
```

- The increase percentage in the output is [0.00% 8.76% 2.94% 2.94% 4.67%]. The system of equations is in ill condition if little change (i.e. 5%) in the input results in large change in the output (i.e. 10% or 20%). The system of equations is not in ill condition since the maximum increase percentage is 8.76% < 10%.