Numerical Methods: Computer Assignment #3

Hong Chenhui
drredthered@gmail.com

Zhejiang University — March 18, 2024

Info: This pieces of work is only intended for a rough exercise,not to be a complete analysis.

1 Problem

1. Develop a Naive Gaussian Elimination algorithm.
2. Use the algorithm to solve the following linear system:

$$\begin{bmatrix} 1 & 2 & 4 & 8 \\ 1 & 3 & 9 & 27 \\ 1 & 4 & 16 & 64 \\ 1 & 5 & 25 & 125 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} 8 \\ 27 \\ 64 \\ 125 \end{bmatrix}$$

3. Change 125 in coefficient matrix to 124.5 and solve the system again. Explain the result.

1.1 Theoretical viewpoint

Question

To perfor Naive Gaussian Elimination, we need to first convert the augmented matrix to an upper triangular matrix, then solve the system by back substitution. Pseudocode as follows:

Algorithm 1: Naive Gaussian Elimination

Data: Augmented matrix A
Result: Solution vector x
for $k = 1$ *to* $n - 1$ **do**
  for $i = k + 1$ *to* n **do**
    $r = \frac{A[i][k]}{A[k][k]}$;
    for $j = k$ *to* $n + 1$ **do**
      $A[i][j] = A[i][j] - r \cdot A[k][j]$;
    end
  end
end
for $i = n$ *to* 1 **do**
  $x[i] = \frac{A[i][n+1]}{A[i][i]}$;
  for $j = i + 1$ *to* n **do**
    $x[i] = x[i] - \frac{A[i][j] \cdot x[j]}{A[i][i]}$;
  end
end

2 implementation

2.1 Original Equation

$$\begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

2.2 Modified coefficient matrix

$$\begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} -0.8276 \\ 0.8966 \\ -0.3103 \\ 1.0345 \end{bmatrix}$$

3 Analysis

One could distinguish the problem as apparently ill-conditioned, for small changes in coefficients result in large changes in the solution. Several methods could be used to interpret the situation.

3.1 Determinant

Calculate the determinant of the coefficient matrix when scaling down:

$$\begin{vmatrix} \frac{1}{8} & \frac{1}{4} & \frac{1}{2} & 1 \\ \frac{1}{27} & \frac{1}{9} & \frac{1}{3} & 1 \\ \frac{1}{64} & \frac{1}{16} & \frac{1}{4} & 1 \\ \frac{1}{125} & \frac{1}{25} & \frac{1}{5} & 1 \end{vmatrix} = 6.9444 \times 10^{-6} \approx 0$$

Which means it is a terribly ill-conditioned problem.

3.2 Inverts

$$A_{mod}^{-1} = \begin{bmatrix} 80 & -540 & 960 & -500 \\ -62.6667 & 513 & -992 & 541.6667 \\ 16 & -148.5 & 320 & -187.5 \\ -1.3333 & 13.5 & -32 & 20.8333 \end{bmatrix}$$

with pratically every elements several orders of magnitude greater than one. High possibility that the problem retains ill-conditioned.

3.3 Matrix Condition Number

$$A_{mod} = \begin{bmatrix} \frac{1}{8} & \frac{1}{4} & \frac{1}{2} & 1 \\ \frac{1}{27} & \frac{1}{9} & \frac{1}{3} & 1 \\ \frac{1}{64} & \frac{1}{16} & \frac{1}{4} & 1 \\ \frac{1}{125} & \frac{1}{25} & \frac{1}{5} & 1 \end{bmatrix}$$

$$Cond[A_{mod}] = ||A_{mod}||_\infty \cdot ||A_{mod}^{-1}||_\infty = \frac{15}{8} \times \frac{6326}{3} = 3953.75$$

The fact that the condition number is considerably greater than unity suggests that the system is ill-conditioned.

4 Codes

//fuction

```
[naiveGauss.m]
function x = naiveGauss(A, b)
    % Check if the matrix is square
    [m, n] = size(A);
    if m ~= n
        error('Matrix A must be square');
    end

    % Check if the number of rows of A matches the number of elements in b
    if n ~= length(b)
        error('Dimensions of A and b are inconsistent');
    end

    % Augmented matrix [A|b]
    Ab = [A, b];

    % Forward elimination
    for k = 1:n-1
        for i = k+1:n
            factor = Ab(i,k) / Ab(k,k);
            Ab(i,k:n+1) = Ab(i,k:n+1) - factor * Ab(k,k:n+1);
        end
    end

    % Back substitution
    x = zeros(n, 1);
    x(n) = Ab(n,n+1) / Ab(n,n);
    for i = n-1:-1:1
        x(i) = (Ab(i,n+1) - Ab(i,i+1:n)*x(i+1:n)) / Ab(i,i);
    end
end
```

Command Line

```
>>A = [1, 2, 4, 8; 1, 3, 9, 27; 1, 4, 16, 64; 1, 5, 25, 125]; //125 to 124.8
>>b = [8; 27; 64; 125];
>>x = naiveGauss(A, b);
>>disp(x);
```