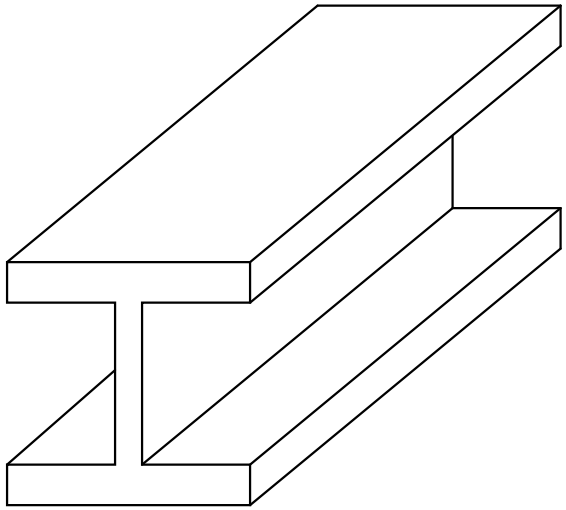
CEE384—Numerical Methods

Naive Gauss Elimination

Solving Linear Systems of Equations



Arizona State University

Brian Chevalier

Updated: October 21, 2019

Contents

1 Naive Gauss Elimination

Naive Gauss Elimination is a linear algebra method to solve matrix equations of the following form:

$$\mathbf{Ax} = \mathbf{b} \tag{1}$$

This method is similar to a method called Reduced Row Echelon Form that you may be familiar with from previous courses.

$$\mathbf{A} = \begin{bmatrix} A_{0,0} & A_{0,1} & A_{0,2} & A_{0,3} \\ A_{1,0} & A_{1,1} & A_{1,2} & A_{1,3} \\ A_{2,0} & A_{2,1} & A_{2,2} & A_{2,3} \\ A_{3,0} & A_{3,1} & A_{3,2} & A_{3,3} \end{bmatrix} \tag{2}$$

Note: the indexing used here is zero-indexing. That is, numbering begins at zero, not at one like typical linear algebra. That is because almost all programming languages use zero-indexing.

1.1 Forward Elimination

Forward elimination is the process by which a matrix is converted into an upper triangular matrix, which is where all elements below the diagonal are zero.

Let's take a look at the first step in the process. We want to make element $A_{1,0}$ equal to zero such as in Equation 3

$$\mathbf{A} = \begin{bmatrix} A_{0,0} & A_{0,1} & A_{0,2} & A_{0,3} \\ A_{1,0} & A_{1,1} & A_{1,2} & A_{1,3} \\ A_{2,0} & A_{2,1} & A_{2,2} & A_{2,3} \\ A_{3,0} & A_{3,1} & A_{3,2} & A_{3,3} \end{bmatrix} \tag{3}$$

To accomplish this, we take the row with the element we want to eliminate (row 1), and subtract row 0 by some scaled amount that will guarantee $A_{1,0}$ will become zero.

$$A_{1,:} = A_{1,:} - \frac{A_{1,0}}{A_{0,0}} A_{0,:} \tag{4}$$

Note: the notation $A_{1,:}$ reads as "row one, all of the columns". Similarly, $A_{:,0}$ would read as "all of the rows of the zero column". In linear algebra classes this may be written as "R_1", however the notation used here makes it easier to jump into programming the logic later.

We are now left with a zero in position $A_{1,0}$.

$$\mathbf{A} = \begin{bmatrix} A_{0,0} & A_{0,1} & A_{0,2} & A_{0,3} \\ 0 & A_{1,1} & A_{1,2} & A_{1,3} \\ A_{2,0} & A_{2,1} & A_{2,2} & A_{2,3} \\ A_{3,0} & A_{3,1} & A_{3,2} & A_{3,3} \end{bmatrix} \tag{5}$$

Next, eliminate $A_{2,0}$ with the same method as before.

$$A_{2,:} = A_{2,:} - \frac{A_{2,0}}{A_{0,0}} A_{0,:} \tag{6}$$

$$A_{i,:} -= \frac{A_{i,j}}{A_{j,j}} A_{j,:} \quad (1 \leq i < n) \quad (0 \leq j < i) \tag{7}$$

$$b_i -= \frac{A_{i,j}}{A_{j,j}} b_j \quad (1 \leq i < n) \quad (0 \leq j < i) \tag{8}$$

1.2 Backward Substitution

$$x_i = \frac{b_i - \sum_{j=i+1}^{n} A_{i,j} x_j}{A_{i,i}} \quad (n-1 \leq i < 0) \tag{9}$$

1.3 Limitations

There may be instances during the process of forward elimination where the order of the rows in the matrix A require division by zero (i.e. when $A_{0,0}$ is zero). To avoid this problem we use a method called "partial pivoting". This means rearranging the rows such that division by zero errors are not encountered.

Furthermore, there may be instances where division by very small (almost zero) values occur. This will cause greater numerical errors that could be significant. We can use "full pivoting" to arrange the rows such division always occurs with the greatest possible denominator to minimize this error.

Finally, there are times we may want to analyze a system of equations with many $\mathbf{b}$ vectors, as is the case when a structure is subjected to many load cases. Using Gauss Elimination would require redoing work each time a new load case should be analyzed.

References

[1] Grant Sanderson. (2017) Taylor Series. [Online]. Available: https://www.youtube.com/watch?v=3d6DsjIBzJ4