1 February 2nd, 2021

1.1 Class Logistics

This course is a continuation of MATH5311. In MATH5311, we introduced how to discretize differential equations. In the course, we focus on the algebraic side of numerical analysis, introducing some more advanced solvers for these numerical methods.

1.1.1 Assessment Scheme

Biweekly Homework - 60%

Take-home Final Exam - 40%

1.1.2 Reference Books

Lecture Notes - Main content that will be covered

Youset Saad: Iterative Methods for Sparse Linear Systems - Free ebook can be found on Canvas

Golub, Van Lean: Matrix Computations

Trefethen, Bau: Numerical Linear Algebra

1.2 Direct Methods for Systems of Lienar Equations

The goal of this chapter is very simple, solve:

$$Ax = b$$

where

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix} \in \mathbb{R}^{n \times n}$$

and:

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \in \mathbb{R}^n$$

which is the unknown vector, and:

$$b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} \in \mathbb{R}^n$$

is the right hand side (RHS).

The solution of system of linear equations is a fundamental task in numerical analysis. If we want to solve a non-linear equation, we can do it iteratively, where each step is solving a linear system. In this chapter, we will introduce **Direct Methods**.

Definition 1.1. A **Direct Method** is a method in which, if there is no truncation error in the computer, can solve the system exactly in finite time.

One example of a direct method is **Gaussian Elimination**.

Example 1.2

Consider:

$$\begin{cases} 2x_1 + x_2 - x_3 = 1 \\ 4x_1 + 5x_2 - 3x_3 = -3 \\ -2x_1 + 5x_2 - 2x_3 = -8 \end{cases}$$

To perform Gaussian Elimination, we eliminate x_1 from Equations 2 and 3 by subtracting Equation 1, and then x_2 from Equation 3 by subtracting Equation 2. Doing so, we get:

$$\begin{cases} 2x_1 + x_2 - x_3 = 1 \\ 3x_2 - x_3 = -5 \\ -x_3 = 3 \end{cases}$$

After this, we can solve x_3 and then using its value solve x_2 and x_3.

Typically we use matrix terminology to reformulate the methods. We do this by writing the **augmented matrix** of the above linear system:

$$\left[\begin{array}{ccc|c} 2 & 1 & -1 & -1 \\ 4 & 5 & -2 & -3 \\ -2 & 5 & -3 & -8 \end{array}\right]$$

Lemma 1.3

Row Operations are equivalent to left matrix multiplications.

Example 1.4

In Example 1.2, the row operation to eliminate x_1 for Equations 2 and 3 can be represented as:

$$\begin{bmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \left[\begin{array}{ccc|c} 2 & 1 & -1 & -1 \\ 4 & 5 & -2 & -3 \\ -2 & 5 & -3 & -8 \end{array}\right] = \left[\begin{array}{ccc|c} 2 & 1 & -1 & -1 \\ 0 & 3 & -1 & -5 \\ 0 & 6 & -3 & -7 \end{array}\right]$$

As such, if A is the augmented matrix and we want to solve the system $Ax = b$, we have:

$$L_{n-1} \dots L_1 A x = L_{n-1} \dots L_1 b$$

and until the LHS is an upper triangular matrix.

Remark 1.5 — Note that the row operations L_i are lower triangular matrix. In particular, it is a rank 1 modification of the identity matrix:

$$L_i = I - ve_i^T$$

where $v \in \mathbb{R}^n$ and $v_1 = v_2 = \ldots = v_i = 0$ and e_i is the i-th unit vector.

Definition 1.6. L_i are called **Gauss Transformations**.

In summary, Gaussian Elimination consists of the following steps:

1. Left multiply Gaussian Transformations to reduce A to an upper triangular matrix:

$$A^{(1)} = A = \begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \ldots & a_{1n}^{(1)} \\ a_{21}^{(1)} & a_{22}^{(1)} & \ldots & a_{2n}^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1}^{(1)} & a_{n2}^{(1)} & \ldots & a_{nn}^{(1)} \end{bmatrix} \text{ and } b^{(1)} = b = \begin{bmatrix} b_1^{(1)} \\ b_2^{(1)} \\ \vdots \\ b_n^{(1)} \end{bmatrix}$$

   In step 1, we have:

$$L_1 = \begin{bmatrix} 1 & & & \\ -\frac{a_{21}^{(1)}}{a_{11}^{(1)}} & 1 & & \\ \vdots & & \ddots & \\ -\frac{a_{n1}^{(1)}}{a_{11}^{(1)}} & 0 & \ldots & 1 \end{bmatrix} = I - \ell_1 e_1^T, \quad \ell 1 = \begin{bmatrix} 0 \\ \frac{a_{21}^{(1)}}{a_{11}^{(1)}} \\ \vdots \\ \frac{a_{n1}^{(1)}}{a_{11}^{(1)}} \end{bmatrix}$$

   After step 1, we have:

$$L_1 A^{(1)} = A^{(2)} = \begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \ldots & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & \ldots & a_{2n}^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & a_{n2}^{(2)} & \ldots & a_{nn}^{(2)} \end{bmatrix} \text{ and } L_1 b^{(1)} = b^{(2)} = \begin{bmatrix} b_1^{(1)} \\ b_2^{(2)} \\ \vdots \\ b_n^{(2)} \end{bmatrix}$$

   We can continue this process until the LHS is upper left triangular. The pseudocode of this first part is given in 1.

2. For part 2, we solve the resulting upper triangular linear system:

$$A^{(n)} x = b^{(n)}$$

   This is quite straightforward, as we just start from the last row and solve each equation, since we are only introducing one unknown with each equation. The pseudocode of this second part is given in 2.

Remark 1.7 — Note that a naive implementation would take $O(n^3)$ memory, as three indexes are used for $a_{ij}^{(k)}$.

We can implement this more efficiently by dropping the superscript, since we do not change it. In other words, we use $a_{ij}^{(k)}$ to overwrite a_{ij}. This will take up the upper triangular part of the matrix. Since the lower triangular part of the matrix is 0, we can use write ℓ_{ik} to this memory. If we do not need b anymore, we can overwrite it as well, meaning that we can do Gaussian Elimination without requiring additional memory.

Algorithm 1 Gaussian Elimination Part 1

```
for  k = 1 : n − 1 do
    for i = k + 1 : n do
        ℓ_ik = a_ik^(k) / a_kk^(k)
    end for
    for i = k + 1 : n do
        for j = k + 1 : n do
            a_ij^(k+1) = a_ij^(k+1) − ℓ_ik a_kj^(k)        ▷ Compute A^(k+1) = L_k A^(k)
        end for
    end for
    for i = k + 1 : n do
        b_i^(k+1) = b_i^(k+1) − ℓ_ik b_i^(k)        ▷ Compute b^(k+1) = L_k b^(k)
    end for
end for
```

Algorithm 2 Gaussian Elimination Part 2

```
for  k = n : −1 : 1 do
    for j = n : −1 : k + 1 do
        y_k = y_k − u_kj x_j
    end for
    x_k = y_k / a_kk
end for
```