Numerical Methods

Lev Leontev

February 17, 2023

Contents

1 Linear Systems of Equations

***Definition* 1.** A linear system of equations is given by

$$Ax = b,\ A \in \mathbb{R}^{m\times n},\ x \in \mathbb{R}^n, b \in \mathbb{R}^m$$

I.e. the matrix A has m rows and n columns, x is a vector with n unknowns, b has m entries, thus the system has m equations.

Remark. The system of equations is called *linear*, because the degree of all x_i is equal to one.

Remark. If $n = m$, the system is called *square.* (As the matrix is square).

Remark. We can also write the system as a sum:

$$\sum_{j=1}^{n} a_{ij}x_j = b_i,\ i = 1, \ldots, m$$

Example.

$$\begin{cases} a_{11}x_1 + a_{12}x_2 = b_1 \\ a_{21}x_1 + a_{22}x_2 = b_2 \end{cases} \quad n = 2,\ m = 2$$

Linear systems of equations arise in a lot of problems:

- Geometrical problems (coordinate transforms, 3D matrices).
- Electrical circuits, Kirchhoff's laws/Ohm's laws.
- Solving differential equations.
- GPS.

1.1 Gaussian Elimination (GE)

Assume that $m = n$ (square system). The idea of Gaussian Elimination: do row operations to produce an upper triangular matrix (echelon form). Then do backward substitution to solve the system.

Allowed row operations:

1. Swap rows.
2. Scale rows, i.e. multiply a row by a scaler.
3. Add multiples of one row to another.

Example.

$$A = \begin{bmatrix} 6 & -2 & 2 & 4 \\ 12 & -8 & 6 & 10 \\ 3 & -13 & 9 & 3 \\ -6 & 4 & 1 & -18 \end{bmatrix}, \quad b = \begin{bmatrix} 16 \\ 26 \\ -19 \\ -34 \end{bmatrix}$$

Step 1. Do GE in a systematic way:

$$\text{Augmented matrix} = \left[\begin{array}{cccc|c} 6 & -2 & 2 & 4 & 16 \\ 12 & -8 & 6 & 10 & 26 \\ 3 & -13 & 9 & 3 & -19 \\ -6 & 4 & 1 & -18 & -34 \end{array}\right]$$

The 6 here is the pivot element, and the first row is the pivot row.

$$\left[\begin{array}{cccc|c} 6 & -2 & 2 & 4 & 16 \\ 12 & -8 & 6 & 10 & 26 \\ 3 & -13 & 9 & 3 & -19 \\ -6 & 4 & 1 & -18 & -34 \end{array}\right] \begin{array}{l} \leftarrow \text{ pivot row} \\ \leftarrow (-2)\cdot R_1 + R_2 \\ \leftarrow (-1/2)\cdot R_1 + R_3 \\ \leftarrow 1\cdot R_1 + R_4 \end{array}$$

$$\hookrightarrow \left[\begin{array}{cccc|c} 6 & -2 & 2 & 4 & 16 \\ 0 & -4 & 2 & 2 & -6 \\ 0 & -12 & 8 & 1 & -27 \\ 0 & 2 & 3 & -14 & -18 \end{array}\right] \begin{array}{l} \\ \leftarrow \text{ pivot row} \\ \leftarrow (-3)\cdot R_2 + R_3 \\ \leftarrow (1/2)\cdot R_2 + R_4 \end{array}$$

Always consider the factor, e.g.

$$-3 = -\left(\frac{-12}{-4}\right)$$
$$\frac{1}{2} = -\left(\frac{2}{-4}\right)$$

Eventually, we end up with a triangular form (using diagonal elements as pivots).

$$\left[\begin{array}{cccc|c} 6 & -2 & 2 & 4 & 16 \\ 0 & -4 & 2 & 2 & -6 \\ 0 & 0 & 2 & -5 & -9 \\ 0 & 0 & 0 & -3 & -3 \end{array}\right]$$

Step 2. Backward substitution:

- Last row: $-3x_4 = -3 \iff x_4 = 1$.
- Second last row:

$$2x_3 - 5x_4 = -9$$
$$2x_3 - 5 = -9 \iff x_3 = -2$$

- ...finally: $x_1 = 3,\ x_2 = 1,\ x_3 = -2,\ x_4 = 1$.

The algorithm again:

1. Input $A \in \mathbb{R}^{n\times n},\ b \in \mathbb{R}^n$.

   Forward substitution:

2. For $k = 1, \ldots, n-1$ (for all pivot rows, except the last one):

3. For $i = k+1, \ldots, n$ (for all rows below the pivot row):

4. For $j = k, \ldots, n$ (for all columns from the pivot one):
   $a_{ij} := a_{ij} - \frac{a_{ik}}{a_{kk}} a_{kj}$
   End for.
   $b_i := b_i - \frac{a_{ik}}{a_{kk}} b_k$

$\bar{3}$. End for.

$\bar{2}$. End for.

Backward substitution:

5. $x_n = \frac{b_n}{a_n n}$ (last unknown)

6. For $i = n-1, \dots, 1$ (return back row by row)
   rhs $:= b_i$

7. For $j = n, \dots, i+1$ (for all columns up to the pivot element)
   rhs $:=$ rhs $- a_{ij} x_i$ (all x_j are already known)

$\overline{7}$. End for. $x_i := \frac{\text{rhs}}{a_{ii}}$

$\overline{6}$. End for.

GE can be used whenever the pivots don't vanish.

Example.

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

But addition of rows will give us:

$$\begin{cases} x_3 = 1 - \text{here we have a missing pivot} \\ x_2 + x_3 = 0 \end{cases} \qquad \left(\begin{array}{ccc|c} 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 2 & 2 & 1 \end{array}\right)$$

We already get into trouble with very small pivot elements.

Example. Let $\varepsilon > 0$ and consider

$$\begin{cases} \varepsilon x_1 + x_2 = 1 \\ x_1 + x_2 = 2 \end{cases} \iff \begin{pmatrix} 2 & 1 \\ 1 & 1 \end{pmatrix} \vec{x} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$

For $\varepsilon \ll 1$, the actual solution is $x_1 \approx x_2 \approx 1$. However, GE yields

$$x_2 = \frac{2 - \frac{1}{\varepsilon}}{1 - \frac{1}{\varepsilon}} \overset{\varepsilon \ll 1}{\approx} \frac{-\frac{1}{\varepsilon}}{-\frac{1}{\varepsilon}} = 1$$

WIth finite precision we will get through backward substitution: $x_2 = 1$ and $x_1 = \frac{1-x_2}{\varepsilon} = 0$ which is wrong. The pivot is too small. But change order of equations.

$$\begin{cases} x_1 + x_2 = 2 \\ \varepsilon x_1 + x_2 = 1 \end{cases} \overset{\text{GE}}{\implies} \begin{cases} x_2 = \frac{1-2\varepsilon}{1-\varepsilon} \\ x_1 = 2 - x_2 \end{cases}$$

Now the answer is correct. The reason why the first one was incorrect is error amplification of x_2 by multiplication. $\frac{1}{\varepsilon}$ leads in the first case to a wrong result.

1.2 Scaled partial pivoting

Definition 2. *Pivoting* means that the pivot element is chosen appropriately, and not just row by row.

Definition 3. *Partial pivoting* means we will reorder rows (not columns, otherwise it would be full pivoting).

***Definition* 4.** *Scaled* means we look for best *relative* pivot, i.e. best ratio between pivot element and maximal entry of row (all in absolute values).

Remark. This will lead to minimal error propagation.

The algorithm:

1. Input $A \in \mathbb{R}^{n\times n},\ b \in \mathbb{R}^m$.
2. Find maximal absolute values of entries in rows $s \in \mathbb{R}^n$, such that $s_i = \max_{j=1}^n |a_{ij}|$.
   Forward elimination:
3. For $k = 1, \ldots, n-1$ (for all pivot rows).
4. For $i = k, \ldots, n$ (for all rows below pivot row)
   compute $\left|\frac{a_{ik}}{s_i}\right|$.

$\bar{4}$. End for.

5. Find row with the largest relative pivot element, name it row j.
6. Swap k with j.
7. Swap entries k and j in vector s.
8. Do skip of forward elimination in row k.

$\bar{3}$. End for.

Backward substitution is done as before, but with updated order.

Example.

$$\left[\begin{array}{cccc|c} 3 & -13 & 9 & 3 & -19 \\ -6 & 4 & 1 & -18 & -32 \\ 6 & -2 & 2 & 4 & 16 \\ -12 & -8 & 6 & 10 & 26 \end{array}\right]$$

Initial $s = (13, 18, 6, 12)$. Iterations:

1. • Relative pivots:

$$\left(\frac{3}{13}, \frac{6}{18}, \frac{6}{6}, \frac{12}{12}\right) = \left(\left|\frac{a_{ik}}{s_i}\right|\right)$$

• Rows 3 and 4 have pivot 1 greater than all others. Select for swapping rows 1 and 3.

$$\left[\begin{array}{cccc|c} 6 & -2 & 2 & 4 & 16 \\ -6 & 4 & 1 & -18 & -32 \\ 3 & -13 & 9 & 3 & -19 \\ 12 & -8 & 6 & 10 & 26 \end{array}\right]$$

• Swap entries $3 \leftrightarrow 1$ in s: $(6, 18, 13, 12)$.

• Forward elimination step (like in GE):

$$\left[\begin{array}{cccc|c} 6 & -2 & 2 & 4 & 16 \\ 0 & 2 & 3 & -14 & -18 \\ 0 & -12 & 8 & 1 & -27 \\ 0 & -4 & 2 & 2 & -6 \end{array}\right]$$

2. On the second iterations, $k = 2$.
   - Relative pivots (we don't care about the first row anymore, so just three rows left):
$$\left(\left| \frac{2}{18} \right|, \left| \frac{12}{13} \right|, \frac{4}{12} \right)$$
     The second ratio is the largest, and it corresponds to the third row.
   - So, we swap row 3 with row $k = 2$.
   - Swap entries in s.
   - Forward elimination. Then backward substitution on updated matrix as before.

Remarks:

- In efficient implementations, the step of row swapping can be omitted, just a permutation vector l needs to be stored to keep track of matrix rearrangements. This will result in "echelon form" that will look like e.g.
$$\begin{array}{c} 2 \rightarrow \\ 4 \rightarrow \\ 1 \rightarrow \\ 3 \rightarrow \end{array} \left[\begin{array}{cccc|c} 0 & * & * & * & * \\ 0 & 0 & 0 & * & * \\ * & * & * & * & * \\ 0 & 0 & * & * & * \end{array} \right]$$

- GE with scaled partial pivoting always works when matrix is invertible, i.e. there exists a A^{-1}, such that $AA^{-1} = I$.

  It will fail for a singular (i.e. not invertible) matrix, because eventually a division by 0 will occur.

- Doing Gaussian elimination has computational complexity of $\mathcal{O}(n^3)$, because we have three nested for-loops. Cubic behaviour n^3 is problematic for large n!

- Traditionally, only the multiplication/division operations were counted in the number of operations C. (Since addition is very cheap). On present-day hardware, however, the costs are nearly as "cheap" as addition or subtraction.

- We are missing costs due to exchange with memory. Therefore, estimates of time complexity and reality may diverge substantially.

- Backward substitution has order n^2, which does not affect the general estimate of n^3.

- Scaled partial pivoting leads to an increase in cost, but order stays n^3.