Solving Linear Systems

§3.4 Solving linear systems (i.e., actually solving $Ax = b$)

MA385/MA530 – Numerical Analysis 1

November 2019

We now know << Annotated slides >>>

- there are sufficient conditions that guarantee we can factorise A as LU;
- How to compute L and U.

But our overarching goal is to solve: "*find* $\boldsymbol{x} \in \mathbb{R}^n$ *such that* $A\boldsymbol{x} = \boldsymbol{b}$, *for some* $\boldsymbol{b} \in \mathbb{R}^n$". We do this by first solving $L\boldsymbol{y} = \boldsymbol{b}$ for $\boldsymbol{y} \in \mathbb{R}^n$ and then $U\boldsymbol{x} = \boldsymbol{y}$. Because L and U are triangular, this is easy. The process is called **back-substitution**.

Example 3.14

Use LU-factorisation to solve

$$\overset{A}{\begin{pmatrix} -1 & 0 & 1 & 2 \\ -2 & -2 & 1 & 4 \\ -3 & -4 & -2 & 4 \\ -4 & -6 & -5 & 0 \end{pmatrix}} \overset{x}{\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix}} = \overset{b}{\begin{pmatrix} -2 \\ -3 \\ -1 \\ 1 \end{pmatrix}}$$

Solution: In Example 4 of Section 3.3, we saw that

$$L = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 \\ 3 & 2 & 1 & 0 \\ 4 & 3 & 2 & 1 \end{pmatrix}, \quad U = \begin{pmatrix} -1 & 0 & 1 & 2 \\ 0 & -2 & -1 & 0 \\ 0 & 0 & -3 & -2 \\ 0 & 0 & 0 & -4 \end{pmatrix}$$

So then...

Set $y = Ux$, and solve $Ly = b$, i.e.,

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 \\ 3 & 2 & 1 & 0 \\ 4 & 3 & 2 & 1 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{pmatrix} = \begin{pmatrix} -2 \\ -3 \\ -1 \\ 1 \end{pmatrix}$$

Clearly, $y_1 = -2$. Substitute this back into the 2^{nd} equation $2y_1 + y_2 = -3$ to get $y_2 = 1$. Now use these in $3y_1 + 2y_1 + y_3 = -1$ to get $y_3 = 0$.

And so on... $y = \begin{pmatrix} -2 \\ -3 \\ 1 \\ 0 \end{pmatrix}$

Now solve for x from
$Ux = y$, ie

$$\begin{pmatrix} -1 & 0 & 1 & 2 \\ 0 & -2 & -1 & 0 \\ 0 & 0 & -3 & -2 \\ 0 & 0 & 0 & -4 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} -2 \\ 1 \\ 3 \\ 0 \end{pmatrix}$$

First solve for x_4, then, in order, x_3, x_2, x_1. Get

$$x = \begin{pmatrix} 1 \\ 0 \\ -1 \\ 0 \end{pmatrix}.$$

[Check!]

Example 3.15

Suppose we want to compute the LU-factorisation of

$$A = \begin{pmatrix} 0 & 2 & -4 \\ 2 & 4 & 3 \\ 3 & -1 & 1 \end{pmatrix}.$$

We can't compute l_{21} because $u_{11} = 0$. But if we swap rows 1 and 3, then we can (we did this as Example 3.4.3). This like changing the order of the linear equations we want to solve. If

$$P = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} \quad \text{then} \quad PA = \begin{pmatrix} 3 & -1 & 1 \\ 2 & 4 & 3 \\ 0 & 2 & -4 \end{pmatrix}.$$

This is called *Pivoting* and P is the permutation matrix.

Pivoting (not covered in class)

Definition 3.16

$P \in \mathbb{R}^{n \times n}$ is a *Permutation Matrix* if every entry is either 0 or 1 (it is a Boolean Matrix) and if all the row and column sums are 1.

Theorem 3.17

For **any** $A \in \mathbb{R}^{n \times n}$ there exists a permutation matrix P such that $PA = LU$.

For a proof, see p53 of text book.

How efficient is the method of LU-factorization for solving $A\boldsymbol{x} = \boldsymbol{b}$? That is, how many computational steps (additions and multiplications) are required? In Section 2.6 of the textbook, you'll find a discussion that goes roughly as follows:

Suppose we want to compute $l_{i,j}$. Recall the formula from Section 3.4:

$$l_{i,j} = \frac{1}{u_{jj}}\left(a_{ij} - \sum_{k=1}^{j-1} l_{ik}u_{kj}\right) \qquad i = 2, \ldots, n, \text{ and } j = 1, \ldots, i-1.$$

We see that this requires $j-2$ additions, $j-1$ multiplications, 1 subtraction and 1 division: a total of $2j-1$ operations.

In Exercise 3.13 we will show that

$$1 + 2 + \cdots + k = \frac{1}{2}k(k+1), \quad \text{and}$$

$$1^2 + 2^2 + \cdots k^2 = \frac{1}{6}k(k+1)(2k+1).$$

So the number of operations required for computing L is

$$\sum_{i=2}^{n}\sum_{j=1}^{i-1}(2j-1) = \sum_{i=2}^{n} i^2 - 2i + 1 =$$
$$\frac{1}{6}n(n+1)(2n+1) - n(n+1) + n \leq Cn^3$$

for some C.

A similar (slightly smaller) number of operations is required for computing U. (For a slightly different approach that yields cruder estimates, but requires a little less work, have a look at Lecture 10 of Stewart's *Afternotes on Numerical Analysis*).

. . . .

This doesn't tell us how long a computer program will take to run, but it does tell us how the execution time grows with n. For example, if $n = 100$ and the program takes a second to execute, then if $n = 1000$ we'd expect it to take about a thousand seconds.

Unlike the other methods we studied so far in this course, we shouldn't have to do an error analysis, in the sense of estimating the difference between the true solution, and our numerical one. That is because the LU-approximation approach should give us exactly the true solution.

However, things are not that simple. Unlike the methods in earlier sections, the effects of (inexact) floating point computations become very pronounced. In the next section, we'll develop the ideas needed to quantify these effects.

Exercise 3.10

Suppose that A has an LDU-factorisation (see Exercises 3.9). How could this factorization be used to solve $A\boldsymbol{x} = \boldsymbol{b}$?

Exercise 3.11

Prove that

$$1 + 2 + \cdots + k = \frac{1}{2}k(k+1), \quad \text{and}$$

$$1^2 + 2^2 + \cdots k^2 = \frac{1}{6}k(k+1)(2k+1).$$