1 Summary

We seek the solution $\mathbf{x}$ to

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{1}$$

where $n \times n$ matrix $\mathbf{A}$ and forcing function $\mathbf{b}$ are known. Our goal is to use row operations to obtain

$$\mathbf{U}\mathbf{x} = \mathbf{d} \tag{2}$$

where $\mathbf{U}$ is an upper-diagonal matrix and $\mathbf{d}$ is related to the forcing function. With an upper-diagonal matrix, the back-substitution algorithm quickly reveals the solution.

2 LU decomposition

2.1 Upper-triangular matrix U

Beginning with

$$\mathbf{A} \equiv \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{25} & \dots & a_{2n} \\ a_{31} & a_{32} & a_{33} & a_{34} & a_{35} & \dots & a_{3n} \\ a_{41} & a_{42} & a_{43} & a_{44} & a_{45} & \dots & a_{4n} \\ a_{51} & a_{52} & a_{53} & a_{54} & a_{55} & \dots & a_{5n} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & a_{n4} & a_{n5} & \dots & a_{nn} \end{bmatrix}$$

we perform a series of row operations to make the first column zero (except for the first entry in the column). This can be achieved via

$$\begin{aligned} \text{(second row)} &\rightarrow \text{(second row)} - \frac{a_{21}}{a_{11}}\text{(first row)} \\ \text{(third row)} &\rightarrow \text{(third row)} - \frac{a_{31}}{a_{11}}\text{(first row)} \\ \text{(fourth row)} &\rightarrow \text{(fourth row)} - \frac{a_{41}}{a_{11}}\text{(first row)} \end{aligned}$$

where the $\rightarrow$ can be interpreted as "becomes".[1] After proceeding through all n rows, we will have developed an intermediate matrix[2]

$$\mathbf{A}' \equiv \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} & \dots & a_{1n} \\ 0 & a'_{22} & a'_{23} & a'_{24} & a'_{25} & \dots & a'_{2n} \\ 0 & a'_{32} & a'_{33} & a'_{34} & a'_{35} & \dots & a'_{3n} \\ 0 & a'_{42} & a'_{43} & a'_{44} & a'_{45} & \dots & a'_{4n} \\ 0 & a'_{52} & a'_{53} & a'_{54} & a'_{55} & \dots & a'_{5n} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & a'_{n2} & a'_{n3} & a'_{n4} & a'_{n5} & \dots & a'_{nn} \end{bmatrix}.$$

[1] What happens if $a_{11} = 0$, even approximately (e.g., 0.000000017)? We need to rearrange the rows such that this is not the case. Let's not worry about it for now.

[2] Note the $'$ notation conflicts with MATLAB's implementation of the transpose – this is not a transpose!

If we proceed similarly with

$$\text{(new third row)} \rightarrow \text{(new third row)} - \frac{a'_{32}}{a'_{22}}\text{(new second row)}$$

$$\text{(new fourth row)} \rightarrow \text{(new fourth row)} - \frac{a'_{42}}{a'_{22}}\text{(first row)}$$

we will arrive at

$$\mathbf{A}'' \equiv \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} & \dots & a_{1n} \\ 0 & a'_{22} & a'_{23} & a'_{24} & a'_{25} & \dots & a'_{2n} \\ 0 & 0 & a''_{33} & a''_{34} & a''_{35} & \dots & a''_{3n} \\ 0 & 0 & a''_{43} & a''_{44} & a''_{45} & \dots & a''_{4n} \\ 0 & 0 & a''_{53} & a''_{54} & a''_{55} & \dots & a''_{5n} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & a''_{n3} & a''_{n4} & a''_{n5} & \dots & a''_{nn} \end{bmatrix}.$$

and can continue with this process until we arrive at

$$\mathbf{U} \equiv \mathbf{A}''''' \cdots \equiv \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} & \dots & a_{1n} \\ 0 & a'_{22} & a'_{23} & a'_{24} & a'_{25} & \dots & a'_{2n} \\ 0 & 0 & a''_{33} & a''_{34} & a''_{35} & \dots & a''_{3n} \\ 0 & 0 & 0 & a'''_{44} & a'''_{45} & \dots & a'''_{4n} \\ 0 & 0 & 0 & 0 & a''''_{55} & \dots & a''''_{5n} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \dots & a''''' \cdots_{nn} \end{bmatrix} \equiv \begin{bmatrix} u_{11} & u_{12} & u_{13} & u_{14} & u_{15} & \dots & u_{1n} \\ 0 & u_{22} & u_{23} & u_{24} & u_{25} & \dots & u_{2n} \\ 0 & 0 & u_{33} & u_{34} & u_{35} & \dots & u_{3n} \\ 0 & 0 & 0 & u_{44} & u_{45} & \dots & u_{4n} \\ 0 & 0 & 0 & 0 & u_{55} & \dots & u_{5n} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \dots & u_{nn} \end{bmatrix}.$$

This is the upper-triangular matrix **U** that will be useful in the back-substitution algorithm.

2.2 Lower-triangular matrix L

The row operations described above can be made mathematical by a multiplication by

$$\mathbf{L} \equiv \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & \dots & 0 \\ a_{21}/a_{11} & 1 & 0 & 0 & 0 & \dots & 0 \\ a_{31}/a_{11} & a'_{32}/a'_{22} & 1 & 0 & 0 & \dots & 0 \\ a_{41}/a_{11} & a'_{42}/a'_{22} & a''_{43}/a''_{33} & 1 & 0 & \dots & 0 \\ a_{51}/a_{11} & a'_{52}/a'_{22} & a''_{53}/a''_{33} & a'''_{54}/a'''_{44} & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1}/a_{11} & a'_{n2}/a'_{22} & a''_{n3}/a''_{33} & a'''_{n4}/a'''_{44} & a''''_{n5}/a''''_{55} & \dots & 1 \end{bmatrix}$$

as in, $\mathbf{LU} = \mathbf{A}$. Note that the lower-triangular entries are the same as the multipliers we used in the algorithm to get **U**. The creation of **L** is practically a by-product of generating **U**.

3 The backward-substitution algorithm

Beginning with Eq. (2), we multiply both sides by **L** to obtain

$$\mathbf{LU} = \mathbf{Ld}.$$

Since $\mathbf{LU} = \mathbf{A}$, this reveals that

$$\mathbf{Ld} = \mathbf{b}.$$

Because $\mathbf{L}$ is a lower-triangular matrix, determination of $\mathbf{d}$ is relatively straightforward via the "forward-substitution algorithm". Framed in terms of a generic lower-triangular matrix with entries ℓ_{ij}, the first three steps are:

$$\begin{aligned}
\ell_{11}d_1 &= b_1 \implies d_1 = \frac{1}{\ell_{11}}b_1 \\
\ell_{21}d_1 + \ell_{22}d_2 &= b_2 \implies d_2 = \frac{1}{\ell_{22}}\left(b_2 - \ell_{21}d_1\right) \\
\ell_{31}d_1 + \ell_{32}d_2 + \ell_{33}d_3 &= b_3 \implies d_3 = \frac{1}{\ell_{33}}\left(b_3 - \ell_{31}d_1 - \ell_{32}d_2\right)
\end{aligned}$$

and generally

$$d_i = \frac{1}{\ell_{ii}}\left(b_i - \sum_{j=1}^{i-1}\ell_{ij}d_j\right).$$

We use this to generate $\mathbf{d}$ from the given forcing function $\mathbf{b}$.

4 The backward-substitution algorithm

Beginning with Eq. (2), and since $\mathbf{U}$ is an upper-triangular matrix, and since we known $\mathbf{d}$ after going through the forward-substitution algorithm above, we can determine the solution $\mathbf{x}$ via the backward-substitution algorithm. Framed in terms of a generic upper-triangular matrix with entries u_{ij}, for a a matrix with n rows, the first three steps are:

$$\begin{aligned}
u_{nn}x_n = d_n &\implies x_n = d_n/u_{nn} \\
u_{(n-1)(n-1)}x_{n-1} + u_{(n-1)(n)}x_n = d_{n-1} &\implies x_{n-1} = \frac{1}{u_{(n-1)(n-1)}}\left(d_{n-1} - u_{(n-1)(n)}x_n\right) \\
u_{(n-2)(n-2)}x_{n-2} + u_{(n-2)(n-1)}x_{n-1} + u_{(n-2)(n)}x_n = d_{n-2} &\implies \\
x_{n-2} = \frac{1}{u_{(n-2)(n-2)}}&\left(d_{n-2} - u_{(n-2)(n-1)}x_{n-1} - u_{(n-2)(n)}x_n\right)
\end{aligned}$$

and generally

$$x_i = \frac{1}{u_{ii}}\left(d_i - \sum_{j=i+1}^{n}u_{ij}x_j\right).$$

This expression is mathematically correct (I think), but it's implementation is tricky: this algorithm must proceed backwards: $i = n$, then $i = n-1$, etc.

We use this to get our solution $\mathbf{x}$.