

HIGH-PERFORMANCE COMPUTING IN APPLIED MATHEMATICS

DIRECT SOLUTION TO THE LINEAR SYSTEM USING LU FACTORIZATION

Solution to the linear system

$$\mathbf{Ax} = \mathbf{b} \quad (1)$$

may be easily provided if a LU factorization of the matrix \mathbf{A} is available

$$\mathbf{A} = \mathbf{LU} \quad (2)$$

where \mathbf{L} is a lower diagonal matrix and \mathbf{U} is an upper diagonal matrix.

Solving boundary-value problems using finite difference methods involves the solution to linear systems with banded matrices.

Definition An $n \times n$ matrix is called a band matrix if for some integers p and q with $1 < p, q < n$ we have $a_{ij} = 0$, whenever $i + p \leq j$ or $j + q \leq i$. The bandwidth of the matrix is defined as $p + q - 1$.

Example: solving the finite difference equations associated to the 1D Poisson problem $-y''(x) = f(x)$

$$-\frac{y_{i-1} - 2y_i + y_{i+1}}{h^2} = f(x_i), \quad i = 1, 2, \dots, n \quad (3)$$

$$y_0 = \alpha, \quad y_{n+1} = \beta \quad (4)$$

requires the solution to the linear system

$$\mathbf{Au} = \mathbf{b} \quad (5)$$

where \mathbf{A} is a tridiagonal matrix

$$\mathbf{A} = \begin{bmatrix} 2 & -1 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & -1 & 2 \end{bmatrix}$$

In general we are interested to solve linear systems where the matrix \mathbf{A} takes the form

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & 0 & 0 & \dots & 0 \\ a_{21} & a_{22} & a_{23} & 0 & & \vdots \\ 0 & a_{32} & a_{33} & a_{34} & 0 & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ & & & & & a_{n-1,n} \\ 0 & \dots & & 0 & a_{n,n-1} & a_{n,n} \end{bmatrix},$$

Assuming that the LU factorization may be performed, the matrices L and U may be found as:

$$\mathbf{L} = \begin{bmatrix} l_{11} & 0 & \dots & \dots & 0 \\ l_{21} & l_{22} & 0 & & \vdots \\ 0 & l_{32} & l_{33} & 0 & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & & 0 & l_{n,n-1} & l_{n,n} \end{bmatrix}, \quad \mathbf{U} = \begin{bmatrix} 1 & u_{12} & 0 & 0 & \dots & 0 \\ 0 & 1 & u_{23} & 0 & & \vdots \\ \vdots & 0 & 1 & u_{34} & 0 & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & \dots & & & & u_{n-1,n} \\ 0 & \dots & & & 0 & 1 \end{bmatrix} \quad (6)$$

To identify the $(2n-1)$ undetermined entries of \mathbf{L} and the $(n-1)$ undetermined entries of \mathbf{U} we impose the matrix equality $\mathbf{A} = \mathbf{LU}$:

$$\begin{bmatrix} a_{11} & a_{12} & 0 & 0 & \dots & 0 \\ a_{21} & a_{22} & a_{23} & 0 & & \vdots \\ 0 & a_{32} & a_{33} & a_{34} & 0 & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ & & & & a_{n-1,n} & \\ 0 & \dots & & 0 & a_{n,n-1} & a_{n,n} \end{bmatrix} = \begin{bmatrix} l_{11} & 0 & \dots & \dots & 0 \\ l_{21} & l_{22} & 0 & & \vdots \\ 0 & l_{32} & l_{33} & 0 & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ & & & & 0 \\ 0 & \dots & & 0 & l_{n,n-1} & l_{n,n} \end{bmatrix} \begin{bmatrix} 1 & u_{12} & 0 & 0 & \dots & 0 \\ 0 & 1 & u_{23} & 0 & & \vdots \\ \vdots & 0 & 1 & u_{34} & 0 & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ & & & & & u_{n-1,n} \\ 0 & \dots & & & 0 & 1 \end{bmatrix}$$

Term-by-term identification of the entries provides the equations:

$$l_{11} = a_{11} \quad (7)$$

$$l_{i,i-1} = a_{i,i-1}, \quad i = 2 : n \quad (8)$$

$$l_{i,i-1}u_{i-1,i} + l_{ii} = a_{ii}, \quad i = 2 : n \quad (9)$$

$$l_{ii}u_{i,i+1} = a_{i,i+1}, \quad i = 1 : n - 1 \quad (10)$$

Equations (7-10) may be solved to obtain

ALGORITHM TRILU

```

l11 = a11
li,i-1 = ai,i-1,  i = 2 : n
u12 = a12/l11
for i = 2 : n - 1
    lii = aii - li,i-1ui-1,i
    ui,i+1 = ai,i+1/lii
end
ln,n = an,n - ln,n-1un-1,n

```

The number of arithmetic operations involved in the TRILU algorithm is:

- $n-1$ additions/subtractions
- $2n-2$ multiplications/divisions

Once the LU factorization is available, the linear system $\mathbf{Ax} = \mathbf{b}$ may be solved in two steps

- $\mathbf{Lz} = \mathbf{b}$ % forward substitution
- $\mathbf{Ux} = \mathbf{z}$ % backward substitution

The forward-backward substitution algorithm is:

ALGORITHM TRISOLVE

```
% solve  $\mathbf{Lz} = \mathbf{b}$ 
 $z_1 = b_1/l_{11}$ 
for  $i = 2 : n$ 
     $z_i = (b_i - l_{i,i-1}z_{i-1})/l_{ii}$ 
end
% solve  $\mathbf{Ux} = \mathbf{z}$ 
 $x_n = z_n$ 
for  $i = n - 1 : -1 : 1$ 
     $x_i = z_i - u_{i,i+1}x_{i+1}$ 
end
```

The number of arithmetic operations involved in the TRISOLVE algorithm is:

- $2n-2$ additions/subtractions
- $3n-2$ multiplications/divisions

Combining the TRILU and TRISOLVE algorithms it follows that when the matrix \mathbf{A} is tridiagonal we can solve the linear system (5) using

- $3n-3$ additions/subtractions
- $5n-4$ multiplications/divisions

which is extremely rapid, even when the dimension of the system is large e.g., $n = 10^6$.

It is noticed that matrix storage is not required but only the 3 vectors that define the matrix.

Both TRILU and TRISOLVE algorithms are serial and parallel programming may not be used to speedup the computations. However, we will use these algorithms in the parallel implementation of the semi-implicit finite difference numerical schemes.

Practical application: solution to the 1D-heat equation using the BTCS scheme

Consider the one-dimensional heat equation

$$\frac{\partial u}{\partial t} = k \frac{\partial^2 u}{\partial x^2} + f(x, t), \quad 0 < x < L, \quad t > 0 \quad (11)$$

with homogeneous boundary conditions

$$u(0, t) = 0 \quad (12)$$

$$u(L, t) = 0 \quad (13)$$

and the initial condition

$$u(0, x) = u_0(x) \quad (14)$$

Assume that the spatial domain $[0, L]$ is divided into n equal intervals of length $\Delta x = L/n$. Consider a constant discretization time step τ . Denote

$$x_j = j\Delta x, \quad t_m = m\tau$$

The exact solution $u(x_j, t_m)$ is numerically approximated as

$$u(x_j, t_m) \approx u_j^m$$

To discretize problem (11) at time $t_m, m \geq 1$ and gridpoint $x_j, j = 1 : n - 1$ we use a backward approximation of the time derivative and a central approximation of the space derivative. The resulting equations are:

$$\frac{u_j^m - u_j^{m-1}}{\tau} = k \frac{u_{j-1}^m - 2u_j^m + u_{j+1}^m}{(\Delta x)^2} + f_j^m, \quad j = 1 : n - 1, \quad m \geq 1 \quad (15)$$

Denoting the mesh parameter

$$s = k \frac{\tau}{(\Delta x)^2}$$

we may rearrange the terms in (15) to:

$$(1 + 2s)u_j^m - s(u_{j-1}^m + u_{j+1}^m) = u_j^{m-1} + \tau f_j^m, \quad j = 1 : n - 1, \quad m \geq 1 \quad (16)$$

Taking into account that from the boundary conditions $u_0^m = 0, u_n^m = 0, m \geq 0$ the solution to the equations (16) may be obtained by solving the linear system:

$$\begin{bmatrix} 1 + 2s & -s & 0 & 0 & \dots & 0 \\ -s & 1 + 2s & -s & 0 & & \vdots \\ 0 & -s & 1 + 2s & -s & 0 & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ & & & & -s & \\ 0 & \dots & & 0 & -s & 1 + 2s \end{bmatrix} \begin{bmatrix} u_1^m \\ \vdots \\ \vdots \\ \vdots \\ u_{n-2}^m \\ u_{n-1}^m \end{bmatrix} = \begin{bmatrix} u_1^{m-1} + \tau f_1^m \\ \vdots \\ \vdots \\ \vdots \\ u_{n-2}^{m-1} + \tau f_{n-2}^m \\ u_{n-1}^{m-1} + \tau f_{n-1}^m \end{bmatrix}$$

which may be solved efficiently using the algorithms TRILU and TRISOLVE. Notice that the LU factorization needs to be done only once, at the beginning of the time loop. At each time step only TRISOLVE needs to be applied such that the computational cost per time step is low, of the same order as the FTCS scheme.

See `btcs_heat1d.f`

- The major advantage of the BTCS scheme over the FTCS scheme is that BTCS is unconditionally stable: for a given spatial discretization BTCS may be implemented with any time step, whereas FTCS required the stability condition $s \leq 0.5$.
- The major drawback from the parallel implementation point of view is that BTCS is a serial scheme, not suitable for parallel programming.