Solving tridiagonal linear systems

Günay Doğan

December 16, 2008

The Thomas algorithm solves the following tridiagonal system in $O(n)$ time, where n is the number of unknowns $\{x_i\}_{i=0}^{n-1}$.

$$\begin{pmatrix} b_0 & c_0 & & & \\ a_1 & b_1 & c_1 & & \\ & a_2 & b_2 & \ddots & \\ & & \ddots & \ddots & c_{n-2} \\ & & & a_{n-1} & b_{n-1} \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ \vdots \\ x_{n-1} \end{pmatrix} = \begin{pmatrix} d_0 \\ d_1 \\ \vdots \\ \vdots \\ d_{n-1} \end{pmatrix}. \tag{1}$$

Equation (1) can solved using a simple procedure that consists of three for-loops executed in linear time. The three stages of the procedure are: LU-decomposition, forward substitution and backward substitution. The details are given in Algorithm 1.

Note that LAPACK includes an implementation of a tridiagonal solver, which should be used in practice.

Algorithm 1 The Thomas algorithm

$m_0 = b_0$
for $i = 0, 1, \dots, n-2$ **do**
  $l_i = a_i / m_i$
  $m_{i+1} = b_{i+1} - l_i c_i$
end for
$y_0 = d_0$
for $i = 1, 2, \dots, n-1$ **do**
  $y_i = d_i - l_{i-1} y_{i-1}$
end for
$x_{n-1} = y_{n-1} / m_{n-1}$
for $i = n-2, n-3, \dots, 0$ **do**
  $x_i = (y_i - c_i x_{i+1}) / m_i$
end for

Unfortunately the Thomas algorithm is not directly applicable to coefficient matrices of the form

$$A = \begin{pmatrix} b_0 & c_0 & & & a_0 \\ a_1 & b_1 & c_1 & & \\ & a_2 & b_2 & \ddots & \\ & & \ddots & \ddots & c_{n-2} \\ c_{n-1} & & & a_{n-1} & b_{n-1} \end{pmatrix}. \tag{2}$$

Nonetheless the Thomas algorithm can still be used as a building block for an algorithm to invert A. For this, we define

$$\tilde{A} = \begin{pmatrix} b_0 - a_0 & c_0 & & & 0 \\ a_1 & b_1 & c_1 & & \\ & a_2 & b_2 & \ddots & \\ & & \ddots & \ddots & c_{n-2} \\ 0 & & & a_{n-1} & b_{n-1} - c_{n-1} \end{pmatrix}, \quad u = \begin{pmatrix} a_0 \\ 0 \\ \vdots \\ \vdots \\ c_{n-1} \end{pmatrix}, \quad v = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix},$$

and note

$$A = \tilde{A} + uv^T.$$

Then the Sherman-Morrison formula enables us to write A^{-1} using $\tilde{A}^{-1}$, which is easier to compute

$$A^{-1} = \tilde{A}^{-1} - \frac{\tilde{A}^{-1} u v^T \tilde{A}^{-1}}{1 + v^T \tilde{A}^{-1} u}.$$

Taking advantage of this, we can write a simple three-step algorithm to solve $Ax = d$

- Solve $\tilde{A}z = u$ for z.
- Solve $\tilde{A}y = d$ for y.
- Compute $x = y - \frac{v^T y}{1 + v^T z} z$.