Tridiagonal Systems

Charles Kankelborg

2001-01-21

1 The Problem

A matrix equation $M\mathbf{u} = \mathbf{r}$ is just a way of writing N linear, algebraic equations. The ith equation is:

$$\sum_j M_{ij}\, u_j = r_i. \tag{1}$$

The standard problem is to solve for $\mathbf{u}$ given M and $\mathbf{b}$. A square matrix is said to be *tridiagonal* when

$$M_{ij} = 0, \quad |i - j| > 1. \tag{2}$$

For example,

$$\begin{pmatrix} b_1 & c_1 & 0 & 0 \\ a_2 & b_2 & c_2 & 0 \\ 0 & a_3 & b_3 & c_3 \\ 0 & 0 & a_4 & b_4 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{pmatrix} = \begin{pmatrix} r_1 \\ r_2 \\ r_3 \\ r_4 \end{pmatrix}. \tag{3}$$

2 Algorithm

Have a look *Numerical Recipes* Section 2.6. The *Numerical Recipes* function `tridag` is easily understood without LU decomposition.

To solve the 4×4 system in equation 3 for $\mathbf{u}$, we begin by using row 1 to eliminate a_2 from row 2. Similarly, we eliminate all the a_i row by row.

Here is how the `tridag` implementation works. The first row stands for the equation:

$$b_1\, u_1 + c_1\, u_2 = r_1.$$

$$\text{Let} \quad \beta_1 = b_1, \tag{4}$$

$$v_1 = \frac{r_1}{\beta_1}, \tag{5}$$

$$\gamma_2 = \frac{c_1}{\beta_1} \tag{6}$$

If we divide row 1 by β_1, its equation becomes

$$u_1 + \gamma_2\, u_2 = v_1. \tag{7}$$

The equation for the second row is:

$$a_2\, u_1 + b_2\, u_2 + c_2\, u_3 = r_2.$$

$$\begin{aligned} \text{Let} \quad \beta_2 &= b_2 - a_2\, \gamma_2, \\ v_2 &= \frac{(r_2 - a_2\, v_1)}{\beta_2}, \\ \gamma_3 &= \frac{c_2}{\beta_2}. \end{aligned}$$

We now subtract a_2 times row 1 from row 2. Dividing the result by β_2, our new row 2 is written:

$$u_2 + \gamma_3\, u_3 = v_2. \tag{8}$$

This process may be repeated over the range $i = 2, 3, \ldots N$, using the recursion formulas

$$\gamma_i = \frac{c_{i-1}}{\beta_{i-1}}, \tag{9}$$

$$\beta_i = b_i - a_i\, \gamma_i, \tag{10}$$

$$v_i = \frac{r_i - a_i\, v_{i-1}}{\beta_i}. \tag{11}$$

The result is that the matrix equation is rewritten in uper triangular form. For our 4×4 example,

$$\begin{pmatrix} 1 & \gamma_2 & 0 & 0 \\ 0 & 1 & \gamma_3 & 0 \\ 0 & 0 & 1 & \gamma_4 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{pmatrix} = \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{pmatrix}. \tag{12}$$

This problem is trivially solved by starting from the bottom and back substituting as you go up:

$$u_N = v_N, \tag{13}$$

$$u_i = v_i - \gamma_{i+1}\, u_{i+1}. \tag{14}$$

3 How to Break the Algorithm

If $\beta_i = 0$, for some i, the program quits to avoid a division by zero. The authors of *Numerical Recipes* assure us that this doesn't happen very often.

It can be shown that $\beta_i = 0$ cannot occur in the calculation of y_i'' for the cubic spline.