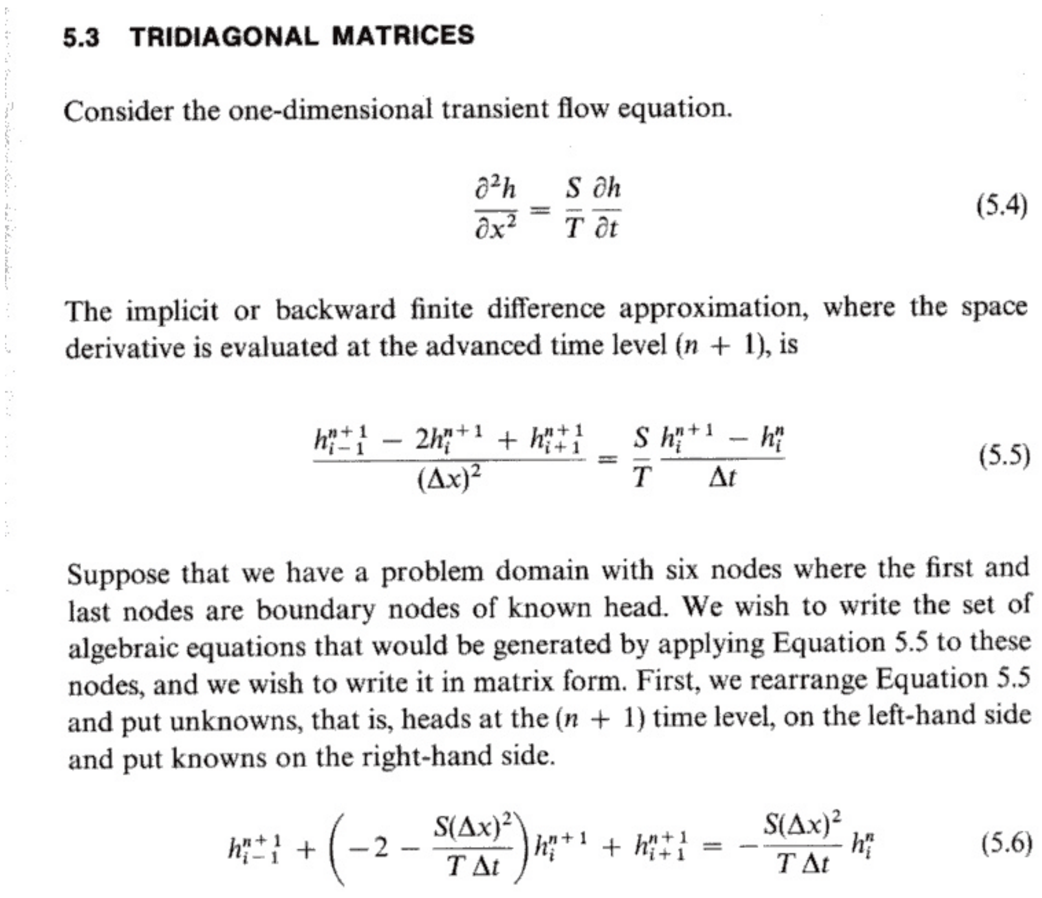
ALGORITMO DE THOMAS PARA MATRICES TRIDIAGONALES

CI66J/CI71T Modelación de Aguas Subterráneas
Profesor C. Espinoza
Semestre Otoño 2007

Wang and Anderson. Introduction to Groundwater Modelling. 1982.



5.3 TRIDIAGONAL MATRICES

Consider the one-dimensional transient flow equation.

$$\frac{\partial^2 h}{\partial x^2} = \frac{S}{T}\frac{\partial h}{\partial t} \tag{5.4}$$

The implicit or backward finite difference approximation, where the space derivative is evaluated at the advanced time level $(n + 1)$, is

$$\frac{h_{i-1}^{n+1} - 2h_i^{n+1} + h_{i+1}^{n+1}}{(\Delta x)^2} = \frac{S}{T}\,\frac{h_i^{n+1} - h_i^n}{\Delta t} \tag{5.5}$$

Suppose that we have a problem domain with six nodes where the first and last nodes are boundary nodes of known head. We wish to write the set of algebraic equations that would be generated by applying Equation 5.5 to these nodes, and we wish to write it in matrix form. First, we rearrange Equation 5.5 and put unknowns, that is, heads at the $(n + 1)$ time level, on the left-hand side and put knowns on the right-hand side.

$$h_{i-1}^{n+1} + \left(-2 - \frac{S(\Delta x)^2}{T\,\Delta t}\right)h_i^{n+1} + h_{i+1}^{n+1} = -\frac{S(\Delta x)^2}{T\,\Delta t}\,h_i^n \tag{5.6}$$

If the head values h_1 and h_6, which are known from the boundary conditions, are transferred to the right-hand side, then the matrix form of the set of algebraic equations for the six-node problem is

$$\begin{bmatrix} -2-\frac{S(\Delta x)^2}{T\Delta t} & 1 & 0 & 0 \\ 1 & -2-\frac{S(\Delta x)^2}{T\Delta t} & 1 & 0 \\ 0 & 1 & -2-\frac{S(\Delta x)^2}{T\Delta t} & 1 \\ 0 & 0 & 1 & -2-\frac{S(\Delta x)^2}{T\Delta t} \end{bmatrix} \begin{Bmatrix} h_2^{n+1} \\ h_3^{n+1} \\ h_4^{n+1} \\ h_5^{n+1} \end{Bmatrix} = \begin{Bmatrix} -\frac{S(\Delta x)^2}{T\Delta t} h_2^n - h_1 \\ -\frac{S(\Delta x)^2}{T\Delta t} h_3^n \\ -\frac{S(\Delta x)^2}{T\Delta t} h_4^n \\ -\frac{S(\Delta x)^2}{T\Delta t} h_5^n - h_6 \end{Bmatrix} \tag{5.7}$$

The coefficient matrix has nonzero entries only along the three center diagonals. This type of matrix is known as a tridiagonal matrix.

Thomas Algorithm

In general, a matrix equation such as Equation 5.2 can be solved by a technique known as Gaussian elimination. However, except for special types of coefficient matrices, Gaussian elimination requires a great deal of computer storage and computer time. A particularly efficient form of Gaussian elimination can be used to solve matrix equations which have a tridiagonal coefficient matrix. This form of Gaussian elimination utilizes the Thomas algorithm. Remson et al. (1971) present the details of both Gaussian elimination and the Thomas algorithm. We develop the Thomas algorithm for tridiagonal matrices by systematically solving the linear equations by row operations.

We begin with a set of four equations whose coefficients are in tridiagonal form.

$$\begin{aligned} b_1x_1 + c_1x_2 \qquad\qquad\qquad &= f_1 \\ a_2x_1 + b_2x_2 + c_2x_3 \qquad &= f_2 \\ a_3x_2 + b_3x_3 + c_3x_4 &= f_3 \\ a_4x_3 + b_4x_4 &= f_4 \end{aligned} \tag{5.8}$$

The notation is that a_i is a subdiagonal coefficient, b_i is a center diagonal coefficient, and c_i is a superdiagonal coefficient. The subscript i indicates row number. We will perform row operations in a systematic manner to eliminate the subdiagonal terms and to normalize the coefficients of diagonal terms to 1. The idea is to transform the original tridiagonal set of equations into an equivalent upper diagonal set.

$$\begin{aligned} x_1 + \beta_1x_2 \qquad\qquad &= y_1 \\ x_2 + \beta_2x_3 \qquad &= y_2 \\ x_3 + \beta_3x_4 &= y_3 \\ x_4 &= y_4 \end{aligned} \tag{5.9}$$

To put Equation 5.8 into the form of Equation 5.9, we need to find expressions for β_i and y_i. We do row operations such as one would use in solving the system of equations by hand. If we compare the first row of Equation 5.9 with that of Equation 5.8, we see that $\beta_1 = c_1/b_1$ and $y_1 = f_1/b_1$. We can obtain the form of the second row of Equation 5.9 if we eliminate x_1 between the first row of Equation 5.9 and the second row of Equation 5.8. The first row of Equation 5.9 is multiplied by a_2 and subtracted from the second row of Equation 5.8. This produces the equation

$$(b_2 - a_2\beta_1)x_2 + c_2x_3 = f_2 - a_2y_1 \tag{5.10}$$

We define the coefficient of x_2 in Equation 5.10 to be $\alpha_2 = b_2 - a_2\beta_1$. Dividing Equation 5.10 by α_2 gives

$$x_2 + \frac{c_2}{\alpha_2}x_3 = \frac{f_2 - a_2y_1}{\alpha_2} \tag{5.11}$$

Equation 5.11 is now in the form of the second row of Equation 5.9. We can therefore make the identification that $\beta_2 = c_2/\alpha_2$ and $y_2 = (f_2 - a_2y_1)/\alpha_2$. By continuing down the rows in this fashion, we find the general relations

$$\alpha_i = b_i - a_i\beta_{i-1} \tag{5.12}$$

$$\beta_i = c_i/\alpha_i \tag{5.13}$$

and

$$y_i = (f_i - a_i y_{i-1})/\alpha_i \tag{5.14}$$

If β_0 and y_0 are defined to be equal to zero, the recursive expressions also hold for $i = 1$. All the coefficients β_i and knowns y_i in Equation 5.9 are now defined. The solution of the original problem, Equation 5.8, is done systematically by backward substitution from the bottom row $x_4 = y_4$ to the top row. In general, $x_n = y_n$, where n is the number of equations, and, for $i < n$,

$$x_i = y_i - \beta_i x_{i+1} \tag{5.15}$$

The recursive relations, Equations 5.12 to 5.15, constitute the Thomas algorithm. The Thomas algorithm is programmed in *SUBROUTINE TRIDIA* (Figure 5.1).

Figure 5.1
Subroutine *TRIDIA*. Subroutine for solving a matrix equation with a tridiagonal coefficient matrix using the Thomas algorithm.

```
      SUBROUTINE TRIDIA(N)
C  THIS SUBROUTINE CONTAINS THE THOMAS ALGORITHM
C  THE SOLUTION IS CONTAINED IN THE X ARRAY
      COMMON A,B,C,X,F
      DIMENSION A(50),B(50),C(50),X(50),F(50)
      DIMENSION ALPHA(50),BETA(50),Y(50)
      ALPHA(1)=B(1)
      BETA(1)=C(1)/ALPHA(1)
      Y(1)=F(1)/ALPHA(1)
      DO 201 I=2,N
      ALPHA(I)=B(I)-A(I)*BETA(I-1)
      BETA(I)=C(I)/ALPHA(I)
201   Y(I)=(F(I)-A(I)*Y(I-1))/ALPHA(I)
C  BEGIN BACKWARD SUBSTITUTION FROM LAST ROW
      X(N)=Y(N)
      NU=N-1
      DO 203 I=1,NU
      J=N-I
203   X(J)=Y(J)-BETA(J)*X(J+1)
      RETURN
      END
```