Quick Introduction to MATLAB 7

S. Baskar

Department of Mathematics
Indian Institute of Technology Bombay
Powai, Mumbai 400076.

SI 512 (Spring 2011)

Suggested reference: MATLAB Programming for Engineers (3/e) by Stephen J. Chapman
Outline

1. Heat Equation
   - Tridiagonal System: Thomas Algorithm
   - Implicit Scheme (BTCS)
Recall the fully implicit scheme (BTCS) for the one dimensional heat equation with homogeneous Dirichlet boundary condition is given by

\[-\mu U_{j-1}^{k+1} + (1 + 2\mu) U_j^{k+1} - \mu U_{j+1}^{k+1} = U_j^k, j = 2, 3, \ldots, (n_x - 1).\]

Note here that we have used the index \( j \) running from 1 to \( n_x \), which is for the sake of convenience to MATLAB indexing.

The tridiagonal coefficient matrix \( A \) generated by this implicit scheme is

\[
\begin{pmatrix}
(1 + 2\mu) & -\mu & 0 & 0 & \cdots & 0 & 0 & 0 \\
-\mu & (1 + 2\mu) & -\mu & 0 & \cdots & 0 & 0 & 0 \\
0 & -\mu & (1 + 2\mu) & -\mu & \cdots & 0 & 0 & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \ddots & \ddots \\
0 & 0 & 0 & 0 & \cdots & -\mu & (1 + 2\mu) & -\mu \\
0 & 0 & 0 & 0 & \cdots & 0 & -\mu & (1 + 2\mu)
\end{pmatrix}
\]

This is an \( N \times N \) matrix, where \( N = n_x - 2 \). Here \( \mu = \Delta t / \Delta x^2 \).
Tridiagonal System: Thomas Algorithm

The solution vector for this tridiagonal system is

\[ u = (u_1, u_2, \cdots, u_N)^T = (U_2^{k+1}, U_3^{k+1}, \cdots, U_{n_x-1}^{k+1})^T, \]

and the right hand side vector is

\[ d = (d_1, d_2, \cdots, d_N)^T = (U_2^k, U_3^k, \cdots, U_{n_x-1}^k)^T. \]

Note that \( U_0^{k+1} = U_{n_x}^{k+1} = 0 \) due to boundary condition.

The Thomas algorithm is given by

\[
\begin{align*}
e_1 &= \frac{\mu}{1 + 2\mu}, & f_1 &= \frac{d_1}{1 + 2\mu}, \\
e_{j+1} &= \frac{\mu}{1 + (2 - e_j)\mu}, & f_{j+1} &= \frac{d_{j+1} + \mu f_j}{1 + (2 - e_j)\mu}, \quad j = 1, 2, \cdots, N - 1.
\end{align*}
\]

The back substitution is

\[ u_N = f_N, \quad u_j = f_j + e_j u_{j+1}, \quad j = N - 1, N - 2, \cdots, 1. \]
Function 1: Write a MATLAB function named \texttt{thomas} that takes the values $\mu$, $N$ and the vector $d$ as input, and returns the solution vector $u$ as an output.

```matlab
function u = thomas(mu,d,N)
%Coefficients
    b=1+2*mu;
    e(1) = mu/b; f(1) = d(1)/b;
    for j = 1:N-1
        den = 1+(2-e(j))*mu;
        e(j+1) = mu/den;
        f(j+1) = (d(j+1)+mu*f(j))/den;
    end
%Back substitution
    u(N) = f(N);
    for j = N-1:-1:1
        u(j) = f(j) + e(j)*u(j+1);
    end
end
```
**Function 2:** Write a matlab function named `heim` that takes the vector `u0`, the values of $\Delta x$, $\Delta t$, and $t$ as inputs. Then computes the numerical solution of the heat equation with the Dirichlet boundary condition using the fully implicit method BTCS. Finally, returns the vector $U$ as an output.

```matlab
function u = heim(u0, delx, delt, t)
  nx=size(u0,2); N=nx-2;
  mu = delt/(delx*delx);
  tv=0;
  while tv<=t+delt
    tv=tv+delt
    d=u0(2:nx-1);
    u=thomas(mu,d,N);
    u0(2:nx-1)=u;
  end
  u=u0;
end
```
Let us take the initial condition as

\[ u_0(x) = \begin{cases} 
1, & \text{if } 0.25 \leq x \leq 0.75 \\
0, & \text{elsewhere} 
\end{cases} \]

**Program 1:** Write a matlab function named `heq` that generates the vectors \( x, u_0 \). Then takes value of \( t \) and \( M \) as input from the user. First computes the approximate value of the Fourier series solution of the one dimensional heat equation with Dirichlet boundary condition. Then computes the approximate solution using BTCS method. Finally, plots the initial condition, Fourier series solution and the BTCS solution, all in one frame.
x=[0:0.01:1];
N=size(x,2);
for i = 1:N
if(x(i)>=0.25 && x(i)<=0.75)
u0(i) = 1;
else
u0(i) = 0;
end
end
t=input('Enter the value of t: ');
M=input('Enter the value of M: ');
U=hefss(x,t,M);
u=heim(u0,0.01,0.001,t);
plot(x,u0,'-*',x,U,x,u,'r');