SOLUTION OF BURGER'S EQUATION USING LOWER AND HIGHER ORDER METHODS

HOMEWORK-5

in

MEEN 689 – COMPUTATIONAL FLUID DYNAMICS

By

B SATYA AKHIL REDDY

826006779

Date: 11/26/2018

The initial velocity distribution of a compression wave is given as

$$u(x,0) = \begin{Bmatrix} 1 & ,x < 0.25 \\ 1.25 - x & ,0.25 \leq x \leq 1.25 \\ 0 & x > 1.25 \end{Bmatrix}$$

The inviscid Burger's equation is represented as

$$\frac{\delta u}{\delta t} + \frac{\delta E}{\delta x} = 0$$

Where $E = u^2$. The wave propagation was solved using Burger's equation within a domain of $0 < x < 4$. The solution was developed up to time $t = 6$. The boundary conditions are specified as

$$u(0,t) = 1$$

$$u(4,t) = 0$$

The problem was solved using both lower order and higher order methods. For the lower order method, explicit upwind scheme was used whereas for the higher order method, MacCormack method was used. Figure 1 shows the solution obtained using explicit upwind scheme at different time levels. At t=0, the wave represents the initial solution. As time progresses, the movement of the wave is observed. The solution looks exact since the solution moves along the characteristic line when CFL is equal to 1.

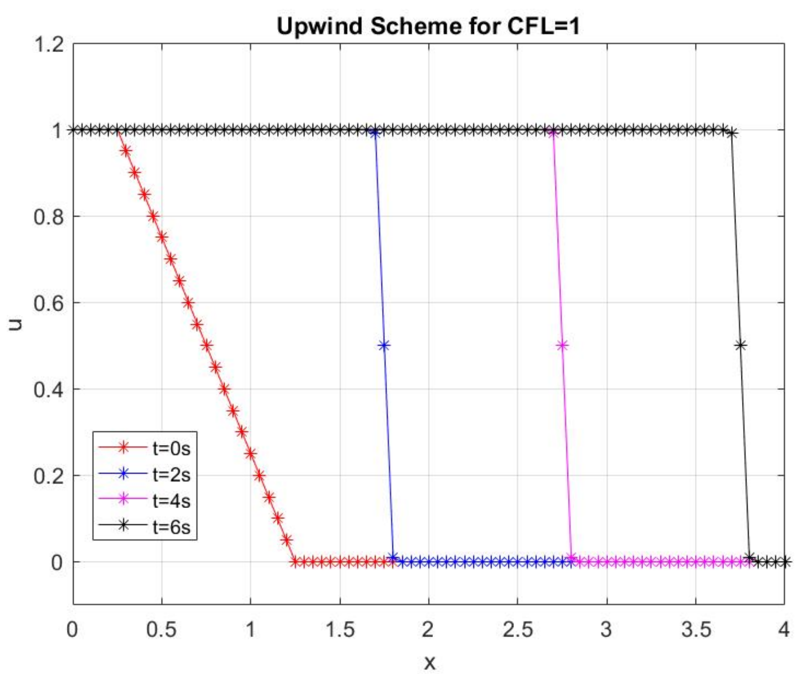

Figure 1 Upwind scheme at different time levels when CFL=1

Figure 2 shows the variation in wave velocity at t=2 for different CFL numbers. Explicit upwind is stable only when $CFL \leq 1$. At CFL=1, since the solution moves along the characteristic, the amplitude of the wave is equal to the given initial condition. But when we decrease the CFL number, the amplitude of the wave is decreased. This happens because lower order explicit upwind is highly dissipative, causing the solution to be dissipated to neighboring points. By the stability conditions, $CFL \geq 1$ gives unexpected results.

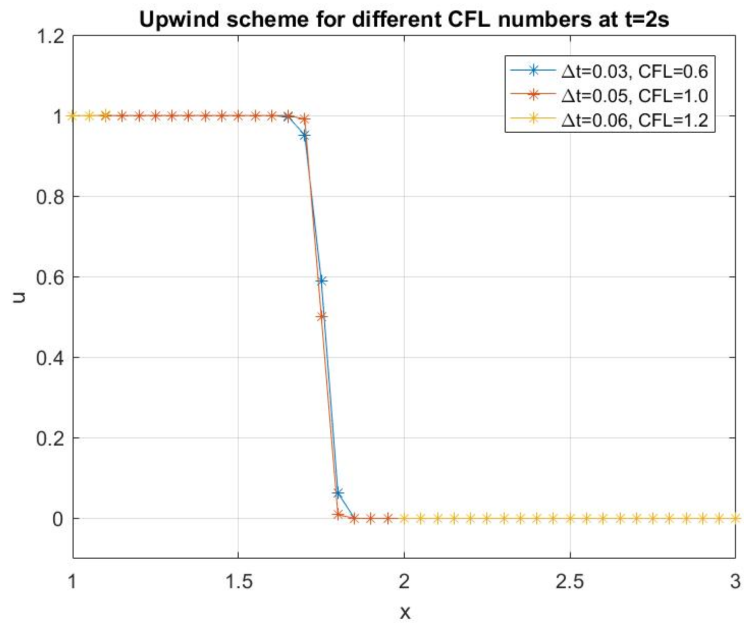

Figure 2 Explicit Upwind at t=2 for different CFL numbers

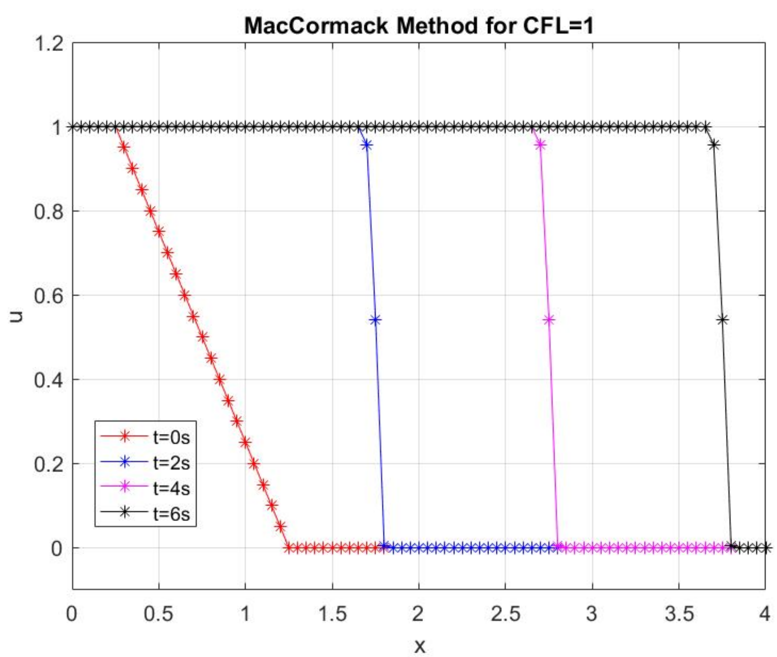

Figure 3 MacCormack method at different time levels when CFL=1

Figure 3 shows the solution obtained using MacCormack method at different time levels. This method has a stability requirement that $CFL \leq 1$. At t=0, like the previous case, it represents the initial solution. As expected, CFL=1 give the best results and are visible in Fig. 3.

Figure 4 shows the variation in wave velocity at t=2 for different CFL numbers. As discussed, the solution at CFL=1 gives the best results. Solution obtained with $CFL \geq 1$ is highly unstable. When $CFL \leq 1$, we see that ringing occurs near the sharp discontinuities. These oscillations are a by-product of the dispersion error introduced into the solution. Dispersion errors are inherent in second order accurate methods.

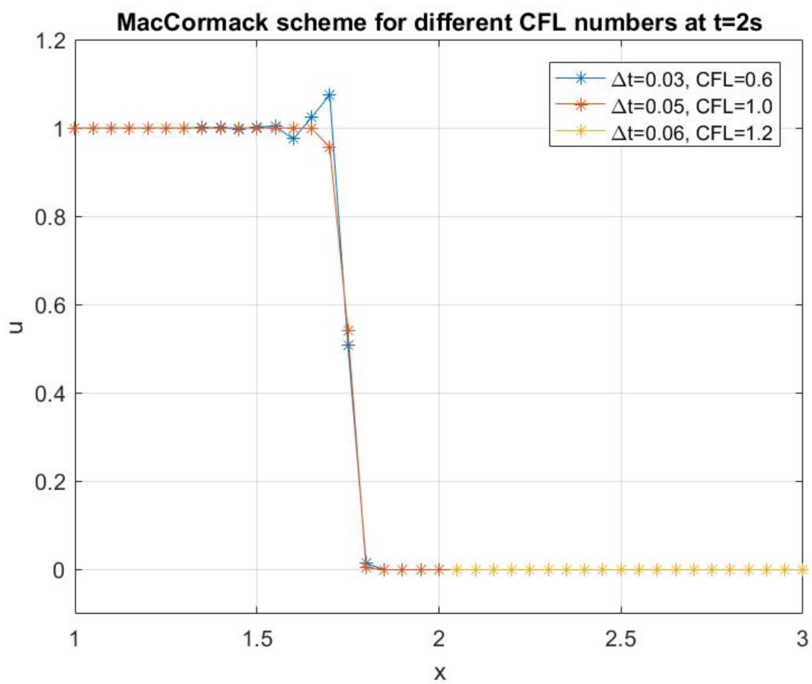

Figure 4 MacCormack solution at t=2 for different CFL numbers

APPENDIX

Lower Order – Explicit first order scheme

```
function u=lower_order(del_t)
    %% Number of nodes

    L=4;
    del_x=0.05;
    Nx=L/del_x+1;

    %% Courant Number

    T=6;
    Nt=T/del_t+1;
    cfl=del_t/del_x;

    %% Initialization and IC,BC

    u=zeros(Nt-1,Nx);

    % Initial Condition

    D=0:0.05:4;
    u(1,1:5)=1;
    u(1,6:26)=1.25-D(6:26);
    u(1,27:81)=0;

    %% Explicit Upwind

    for i=2:Nt
        for j=2:Nx-1
            u(i,j)=u(i-1,j)-(cfl/2)*((u(i-1,j))^2-(u(i-1,j-1))^2);
        end

        % Boundary Condition
        u(i,1)=1;
        u(i,Nx)=0;
    end
 end
```

Higher Order – MacCormack Method

```
function u=higher_order(del_t)
    %% Number of nodes

    L=4;
    del_x=0.05;
    Nx=L/del_x+1;

    %% Courant Number

    T=6;
    Nt=T/del_t+1;
```

```
    cfl=del_t/del_x;

    %% Initialization and IC,BC

    u=zeros(Nt,Nx);

    % Initial Condition

    D=0:0.05:4;
    u(1,1:5)=1;
    u(1,6:26)=1.25-D(6:26);
    u(1,27:81)=0;

    %% MacCormack Method

    u_str=zeros(1,Nx);
    u_str(1,1)=1;
    u_str(1,Nx)=0;

    for i=2:Nt

        % Predictor Step
        for j=2:Nx-1
            u_str(1,j)=u(i-1,j)-(cfl/2)*((u(i-1,j+1))^2-(u(i-1,j))^2);
        end

        % Corrector Step
        for j=2:Nx-1
            u(i,j)=0.5*(u(i-1,j)+u_str(1,j)-(cfl/2)*((u_str(1,j))^2-
(u_str(1,j-1))^2));
        end

        % Boundary Conditions
        u(i,1)=1;
        u(i,Nx)=0;
    end
end
```