Computational Fluid Dynamics

Course Assignment

Contents

Andrew N Jackson

14th March 1997

(1) Introduction

The aim of this assignment is to investigate the performance of a series of algorithms for handling the linear advection equation in one dimension:

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

Where 'u' is a general conserved quantity and the constant coefficient 'a' represents the velocity of a travelling wave profile (ie the wave profile being the distribution of u as a function of x).

The techniques being investigated are: Centred difference, first order upwind (donor cell), Lax-Wendroff, MacCormack, second and third order monotonicity preserving, and donor cell and SHASTA flux corrected transport schemes. These algorithms will be tested to see how they cope with the transport of sinusoidal wave profiles and with the transport of wave profiles containing discontinuities (more specifically, the transport of square and triangular wave profiles). Following this conclusions will be drawn as to which routines are suitable for solving fluid dynamics problems.

(2) Method Of Simulation

The one dimensional advection equation is evaluated over a finite difference grid representing the value of u as a function of x. The spatial domain is made effectively infinite by the use of periodic boundary conditions so that there is no limit to the period of time the simulation can run for. If the mesh spacing is Δx and the time step is Δt, then given the assumption that no fluid particle should be able to move more than one cell in each iteration period we have the Courant-Fredricks-Levy condition:

$$|\lambda| = |a| \Delta t/\Delta x \rfloor 1$$

Where a is the wave velocity from the advection equation.

In the case of this assignment the wave velocity is a constant for the whole system, and so we can modify the finite-difference equations to be expressed purely in terms of u_i and λ.

The code written to evaluate the advection equation can be broken down into two main parts: the definition of the initial conditions and the evaluation of the finite difference representation of the advection equation.

(2.1) Initial Conditions:

The evolution of the wave profile is analysed under two main classes of initial conditions, those for smooth wave transport and for discontinuous wave transport. In both cases 127 mesh points were used.

(2.1.1) Smooth Wave Transport:

In this case the quantity u is distributed along the mesh such that it represents a continuous sinusoidal wave (See fig. 2.1.1). In this assignment, the values of u_i were made to form two whole periods of a sinusoidal wave profile.

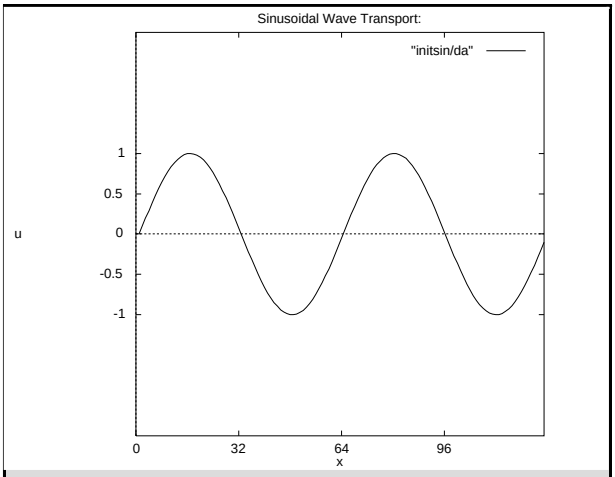

Figure (2.1.1): Initial mesh distribution for the smooth wave transport problem.

(2.1.2) Discontinuous Wave Transport:

Two different types of discontinuous wave are considered here, first-order discontinuous waves (square wave profiles, see fig. 2.1.2) and second-order discontinuous waves (triangular wave profiles, see fig. 2.1.3).

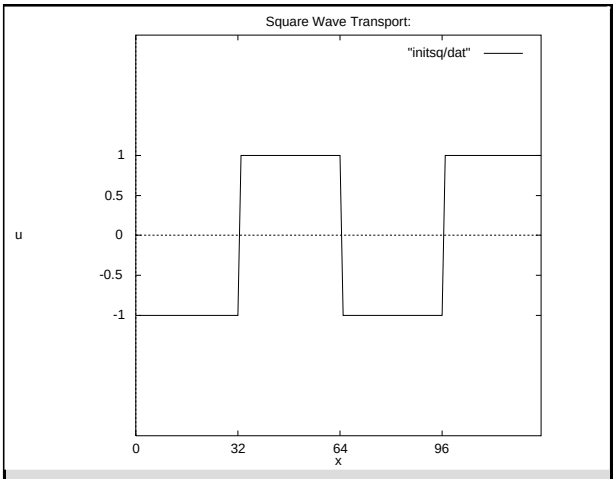

Figure (2.1.2): Initial mesh distribution for the first-order discontinuous wave transport problem.

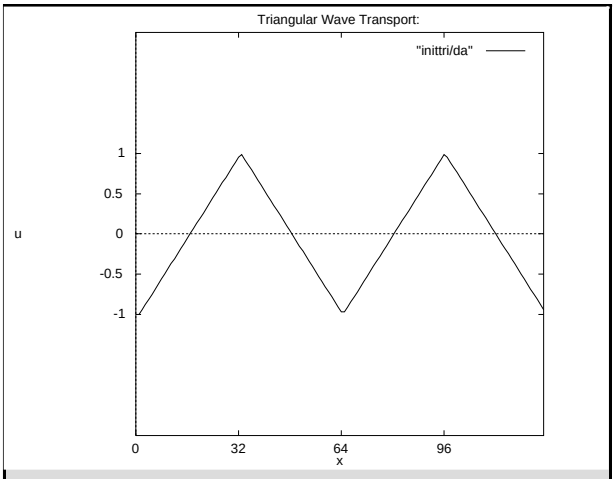

Figure (2.1.3): Initial mesh distribution for the second-order discontinuous wave transport problem.

(2.2) Finite Difference Algorithms:

All of the following are Eulerian (stationary mesh) finite-difference schemes for the solution of the advection equation, as opposed to Lagrangian schemes, where the mesh moves with the fluid.

(2.2.1) Centre Difference:

By simple (first-order) centred spatial differencing of the advection equation we obtain:

$$u_i^{n+1} = u_i^n - {}^{\lambda}/_{2} (u^n_{i+1} - u^n_{i-1})$$

where,

$$\lambda = a\, {}^{\Delta t}/_{\Delta x}$$

as before.

The stability of this algorithm can be ascertained by finding the amplification factor G, defined via the spatial fourier transform giving equations of the form:

$$U^n_k = U^0_k \{G[e^{-ikl\Delta x}, e^{-ik(l-1)\Delta x}, ...]\}^n$$

Clearly, for a stable algorithm

$$|G| \leq 1 + O(\Delta t)$$

However, the centre difference algorithm produces:

$$|G|^2 = 1 + \lambda^2 \sin^2 (k\Delta x)$$

Which must be greater than 1, and so centre differencing should be absolutely unstable for all values of k and λ.

(2.2.2) First Order Upwind (Donor Cell):

The stability problems of the centre difference scheme lie in it's failure to reflect the upstream generation of signals. If the spatial difference is calculated upstream, we find that:

$$u_i^{n+1} = u_i^n - \lambda (u_i^n - u^n_{i-1}) \quad \text{if } \lambda>0$$

$$- \lambda (u^n_{i+1} - u_i^n) \quad \text{otherwise.}$$

This leads to an amplification factor of:

$$|G|^2 = 1 - 2|\lambda|(1-|\lambda|)[1-\cos(k\Delta x)]$$

And so the algorithm is stable is $|\lambda| \leq 1$ (NB this means $|G| < 1$, and so the scheme must be diffusive). Also, if we consider the amplification factor for the exact solution:

$$G = e^{-\lambda k\Delta x}$$

which represents a phase shift of $\lambda k\Delta x$ per step (with no amplitude change), then comparing this to the phase shift of the centre difference scheme, which is:

$$\arg(G) = \arctan\{-|\lambda| \sin(k\Delta x)/[1-\cos(k\Delta x)]\}$$

we see that the scheme solution differs from the exact solution and corresponds to waves of differing k moving with different velocities, in other words the scheme is dispersive. NB the scheme loses it's dispersive properties when $|\lambda|={}^{1}/_{2}$, because at this point $\arg(G) = \pm {}^{k\Delta x}/_{2}$, which is the exact value.

(2.2.3) Lax-Wendroff:

The previous schemes were both first order, and so the lowest order term of the error in the expansion is a second order diffusion term, and this leads the the numerical diffusion problems that those schemes experience. To solve this problem, we must include second order terms in the numerical solutions. The Lax-Wendroff scheme was one of the first to do this effectively, and for the problem considered here, it has the following form:

$$u_i^{n+1} = u_i^n - \lambda/2\,(u^n_{i+1} - u^n_{i-1}) + \lambda^2/2\,(u^n_{i+1} - 2u^n_i + u^n_{i-1})$$

Which leads to an amplification factor of:

$$|G|^2 = 1 - \lambda^2(1-\lambda^2)[1-\cos(k\Delta x)]^2$$

and so the scheme is stable if $|\lambda| \leq 1$. While this scheme is not diffusive, the third order error can still lead to dispersive effects under certain conditions, eg where the wave profile is discontinuous.

(2.2.4) MacCormack:

This second order scheme is closely related to the Lax-Wendroff algorithm, although this is not clear from it's appearance. For the linear advection equation, it has the following two-step form:

$$\tilde{u}_i^{n+1} = u_i^n - \lambda(u^n_{i+1} - u^n_i)$$

$$u_i^{n+1} = \tfrac{1}{2}\,[\,u_i^n + \tilde{u}_i^{n+1} - \lambda(\tilde{u}^n_i - \tilde{u}^n_{i-1})]$$

This scheme has the same diffusion and dispersion properties as the Lax-Wendroff scheme.

(2.2.5) Second Order Monotonicity Preserving:

In order to handle dispersive effects, implicit artificial dissipation can be used to take advantage of the strong dissipation of the first order schemes where required, but the scheme must revert to a higher order form elsewhere. Given that for the advection equation considered here, the solution should preserve monotonicity (ie preserve the form of the wave profile of u), then a monotonicity preserving scheme could clip the flux to prevent overshoot in the regions of strong gradient (eg for shocks) to prevent the generation of short wavelength oscillations. If the change is u is defined by a flux g such that:

$$u_i^{n+1} = u_i^n - \lambda(g_{i+\frac{1}{2}} - g_{i-\frac{1}{2}})$$

Then a general monotonicity preserving scheme is defined as follows:

$$g_{i+\frac{1}{2}} = u_i^n - \tfrac{1}{2}(1-\lambda)\tilde{D}_{i+\frac{1}{2}} \quad \text{if } \lambda>0$$

$$g_{i+\frac{1}{2}} = u_{i+1}^n - \tfrac{1}{2}(1+\lambda)\tilde{D}_{i+\frac{1}{2}} \quad \text{otherwise}$$

where the switching is peformed by

$$D_{i+\frac{1}{2}} = s\,\mathrm{MIN}\{|D_{i+\frac{1}{2}}|, 2|u_{i+1} - u_i|, 2|u_i - u_{i-1}|\}$$

$$s = 0 \quad \text{if } \mathrm{sign}(u_{i+1} - u_i) \neq \mathrm{sign}(u_i - u_{i-1})$$

$$s = \mathrm{sign}(u_i - u_{i-1}) \quad \text{otherwise}$$

$$\tilde{D}_{i+\frac{1}{2}} = s'\,\mathrm{MIN}\{|D'_{i+\frac{1}{2}}|, 2|u_{i+2} - u_{i+1}|, 2|u_{i+1} - u_i|\}$$

$$s' = 0 \quad \text{if } \mathrm{sign}(u_{i+2} - u_{i+1}) \neq \mathrm{sign}(u_{i+1} - u_i)$$

$$s' = \mathrm{sign}(u_{i+2} - u_{i+1}) \quad \text{otherwise}$$

The second order (Lax-Wendroff) monotonicity preserving scheme is then defined by:

$$D_{i+½} = (u_{i+1} - u_i) = - D'_{i+½}$$

While it is not possible to find an amplification factor for this algorithm, the Courant-Fredricks-Levy condition should apply.

(2.2.6) Third Order Monotonicity Preserving:

This scheme is exactly the same as the second order scheme, except:

$$D_{i+½} = (u_{i+1} - u_i) - {}^1/_3\,(1+\lambda)(u_{i+1} - 2u_i + u_{i-1})$$

$$D'_{i+½} = (u_i - u_{i-1}) - {}^1/_3\,(1-\lambda)(u_{i} - 2u_{i+1} + u_{i+2})$$

(2.2.7) Flux Corrected Transport (Donor Cell):

The underlying philosophy of the flux corrected schemes is to perform a two step calculation, where the first step uses a first order diffusion scheme and the second step removes that diffusion, subject to an extremum condition. The first order scheme removes any ripples via diffusion, which is then removed without recreating the high-frequency waves. This philosophy can be used to take advantage of some of the useful properties of the first order upwind differencing scheme, leading to the following two step finite difference form:

$$\grave{u}_i^{n+1} = u_i^n - \{\lambda^- u^n_{i+1} + (\lambda^+ - \lambda^-)u^n_i - \lambda^+ u^n_{i-1}\}$$

where,

$$\lambda^+ = MAX\{0,\lambda\}$$

$$\lambda^- = MIN\{0,\lambda\}$$

The flux correction step then has the form:

$$u_i^{n+1} = \grave{u}_i^{n+1} - \phi_{i+½} + \phi_{i-½}$$

where,

$$\phi_{i+½} = s\,MAX\{0, MIN\{s\Delta_{i+½-1}, |\grave{\phi}_{i+½}|, s\Delta_{i+½+1}\}\}$$

and,

$$\Delta_{i+½} = \grave{u}_{i+1} - \grave{u}_i$$

$$s = sign(\Delta_{i+½})$$

$$\grave{\phi}_{i+½} = ½|\lambda|(1-|\lambda|)(\grave{u}_{i+1} - \grave{u}_i)$$

As with the monotonicity preserving schemes, it is difficult to identify an exact amplification factor for the flux corrected transport schemes. However, we can approximate G via:

$$G = G_A\,G_T$$

Where G_A and G_T are the amplification factors for the anti-diffusion and transport stages respectively. If we assume the anti-diffusion will limit shorter wavelengths only (ie $k\Delta x \sim \pi$), then:

$$G_A \text{ J } 1 + 2\eta[1-\cos(k\Delta x)]$$

where, $\eta = ½|\lambda|(1-|\lambda|)$

This lead to an amplification factor of:

$$|G|^2 \leq 1 - 12\eta^2[1-\cos(k\Delta x)]^2 - 16\eta^3[1-\cos(k\Delta x)]^3$$

and so the scheme should be stable for $0 \leq \eta \leq 3/8$, ie that $0 \leq \lambda \leq 1$. However, the scheme will only maintain positivity for $0 \leq \lambda \leq ½$. Also, this scheme cannot cope with regions of zero flow velocity (ie when $\lambda = 0$).

(2.2.8) Flux Corrected Transport (SHASTA):

This scheme is the same as the donor cell code above except that:

$$\tilde{u}_i^{n+1} = u_i^n - ½(Q^+ u^n_{i+1} - (Q^- + Q^+)u^n_i - Q^- u^n_{i-1})$$

where,

$$Q^+ = (½ - \lambda)^2$$

$$Q^- = (½ + \lambda)^2$$

and the flux correctior has the form:

$$\tilde{\phi}_{i+½} = 1/8(\tilde{u}_{i+1} - \tilde{u}_i)$$

With the same switching criteria as before. This scheme is stable if $|\lambda| \leq \bullet(7/12)$, and positivity is maintained for $|\lambda| \leq 1/2$. The SHASTA scheme is more flexible than the donor cell flux corrected scheme due to it's ability to cope with regions of zero flow velocity.

(3) Results

In the following results, the program was written to plot the wave front by taking into account the speed of the wave and the time elapsed. This makes the simulation effectively Lagrangian, and also allows easy comparison of the results with the initial conditions. All figures show the results of the simulation as a thin black line, and the expected results (ie the initial conditions) as a broad grey line.

(3.1) Transport of Smooth Wave Profiles

This section consists of a series of results from the analysis of the 8 algorithms when applied to the transport of smooth (sinusoidal) waves.

(3.1.1) Centre Difference Results:

As expected, the centre difference approach was found to be completely unstable for all values of λ. Figure (3.1.1) give an example (where λ = 0.9) and shows how the wave profile was destroyed by violent dispersion even after only 70 iterations. Also note that before the wave collapsed, the algorithm had caused the amplitude of the wave to grow, which is another symptom of the poor stability of the algorithm.

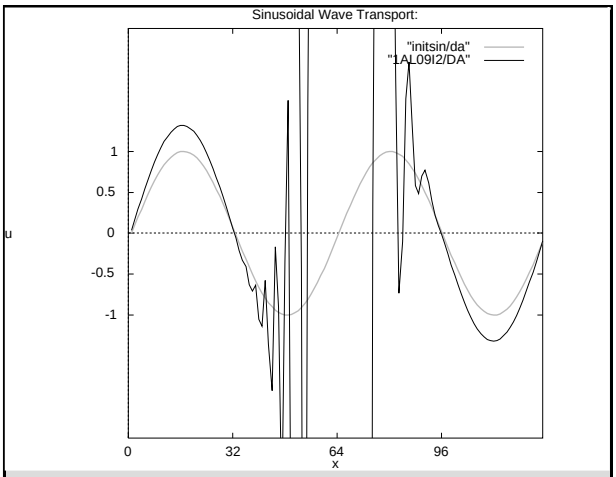

Figure (3.1.1): Centre difference results after 70 iterations for λ = 0.9.

(3.1.2) First Order Upwind Results:

The donor cell scheme produced a much better simulation than the centre difference algorithm, but as shown in figure (3.1.2a), the dissipative effects of the scheme are significant after a few hundred iterations. Experimentation also confirmed the stability condition of the upwind scheme, and figure (3.1.2b) gives an example for this instability at |λ| > 1.0.

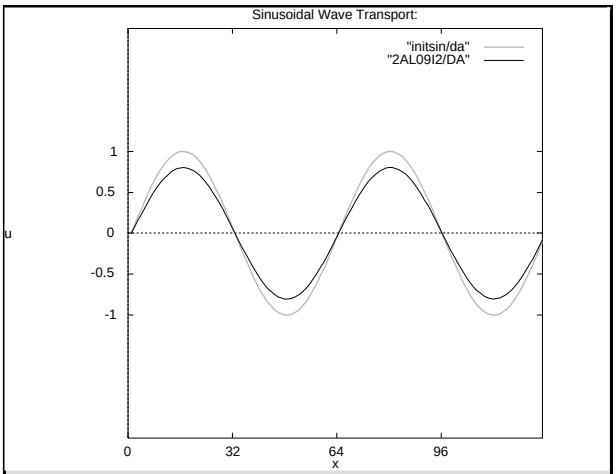

Figure (3.1.2a): First order upwind results after 500 iterations for λ = 0.9.

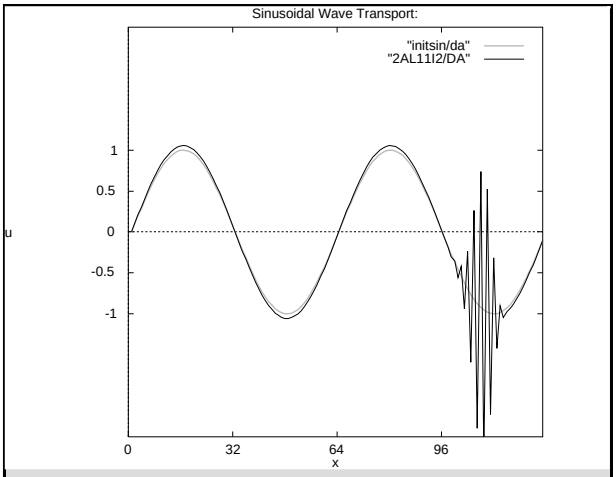

Figure (3.1.2b): First order upwind results after 100 iterations for λ = 1.1.

(3.1.3) Lax-Wendroff Results:

The results for the Lax-Wendroff algorithm show that it successfully solves the problem of rapid dissipation (see figure (3.1.3a)). Also, the stability condition for Lax-Wendroff was also confirmed (see figure (3.1.3b)).

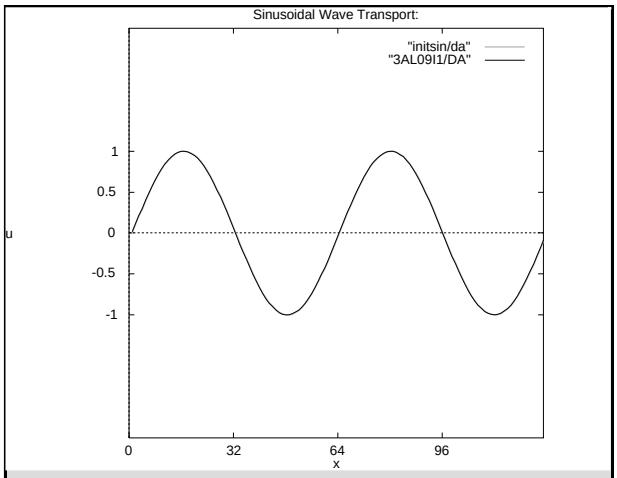

Figure (3.1.3a): Lax-Wendroff results after 500 iterations for $\lambda = 0.9$.

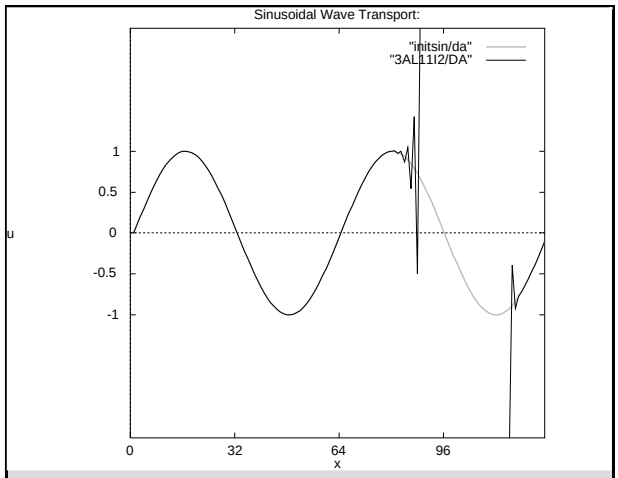

Figure (3.1.3b): Lax-Wendroff results after just 70 iterations for $\lambda = 1.1$.

(3.1.4) MacCormack Results:

As might be expected, the MacCormack algorithm has almost identical diffusion and stability characteristics as the Lax-Wendroff. Figures (3.1.4a) and (3.1.4b) give the MacCormack results for the same sets of conditions as figures (3.1.3a) and (3.1.3b).

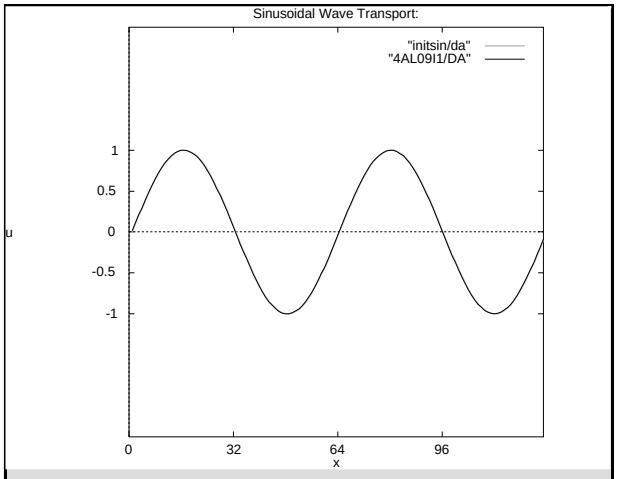

Figure (3.1.4a): MacCormack results after 500 iterations for $\lambda = 0.9$.

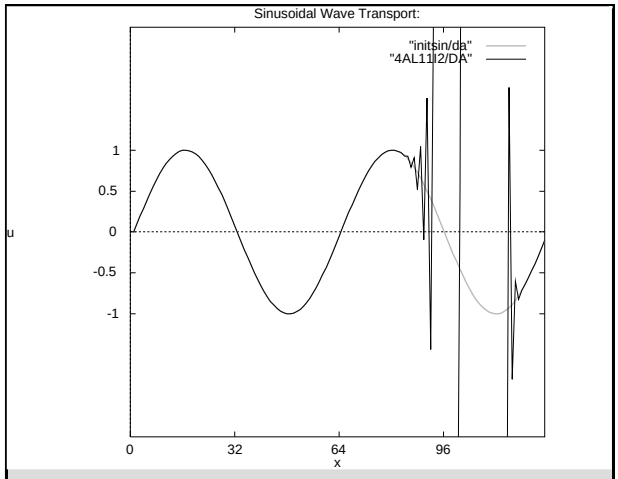

Figure (3.1.4b): MacCormack results after just 70 iterations for $\lambda = 1.1$.

(3.1.5) Second Order Monotonicity Preserving Scheme Results:

This routine was found to perform well, but as show in figure (3.1.5a), the wave profile begins to deform after a few thousand iterations (although reducing λ to $^1/_2$ gave some improvement). The stability condition was confirmed, with high gradient areas leading to dispersion for $|\lambda| > 1.0$ (see figure 3.1.5b)).

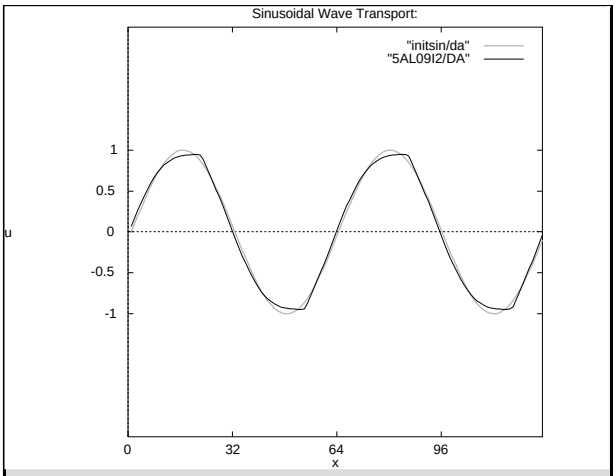

Figure (3.1.5a): Second order monotonicity preserving scheme results after 2000 iterations for $\lambda = 0.9$.

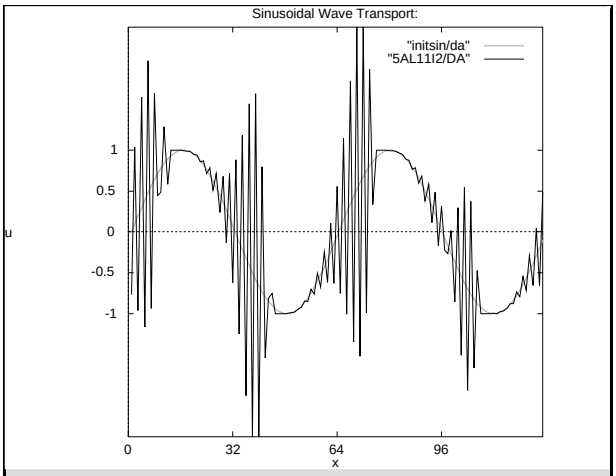

Figure (3.1.5b): Second order monotonicity preserving scheme results after 40 iterations for $\lambda = 1.1$.

(3.1.6) Third Order Monotonicity Preserving Scheme Results:

The third order scheme was found to perform very well. Figure (3.1.6a) illustrates how the algorithm gave very good results even after 5000 iterations. The stability condition was again confirmed to be that $|\lambda| \rfloor 1.0$ (eg figure (3.1.6b)).

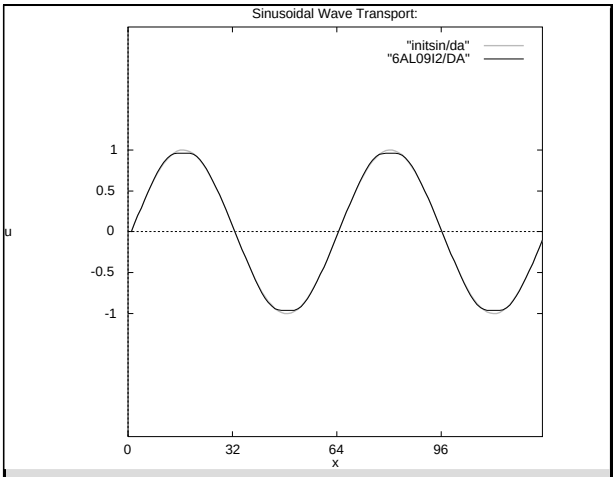

Figure (3.1.6a): Third order monotonicity preserving scheme results after 5000 iterations for $\lambda = 0.9$.

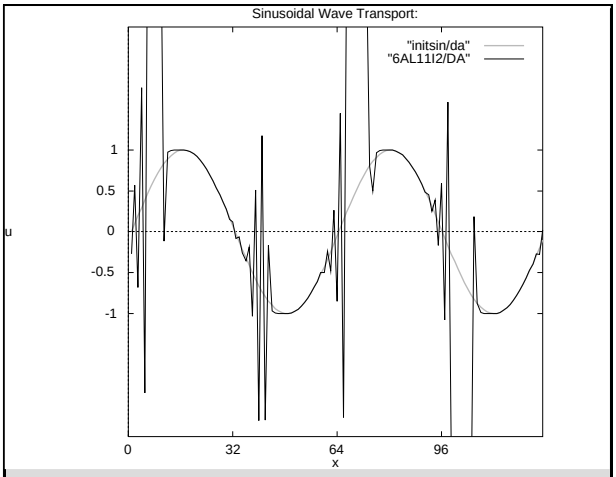

Figure (3.1.6b): Third order monotonicity preserving scheme results after 50 iterations for $\lambda = 1.1$.

(3.1.7) Flux Corrected Transport (Donor Cell) Results:

Both the flux corrected transport schemes were found to work well for 5000 iteration runs. Figure (3.1.7a) illustrated the stability of the donor cell form for $|\lambda| \rfloor 1.0$, and figure (3.1.7b) shows how the results improve for $|\lambda| \rfloor {}^{1}/_{2}$ (ie when the algorithm is preserving positivity).

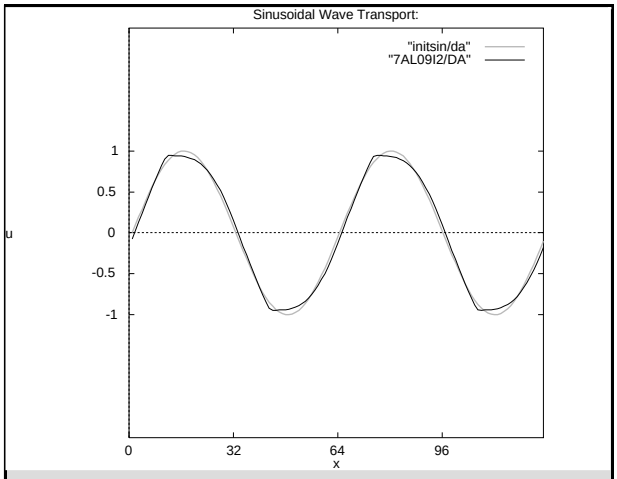

Figure (3.1.7a): Donor cell flux corrected transport scheme results after 5000 iterations for $\lambda = 0.9$.

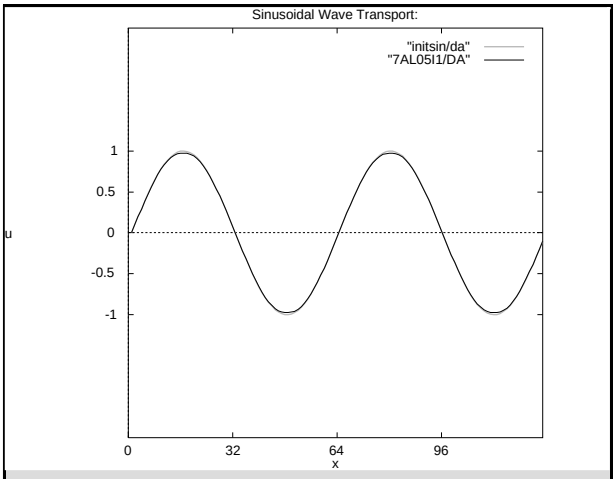

Figure (3.1.7b): Donor cell flux corrected transport scheme results after 5000 iterations for $\lambda = 0.5$.

(3.1.8) Flux Corrected Transport (SHASTA) Results:

This scheme was found to give very good results, with the limit on stability playing an important role in the quality of the simulation. Figure (3.1.8a) shows the algorithm's instability at $|\lambda| > \bullet({}^{7}/_{12})$, with figure (3.1.8b) illustrating the stability for values of $|\lambda|$ below $\bullet({}^{7}/_{12})$ and figure (3.1.8c) showing the improvement in results for $|\lambda| \rfloor {}^{1}/_{2}$.

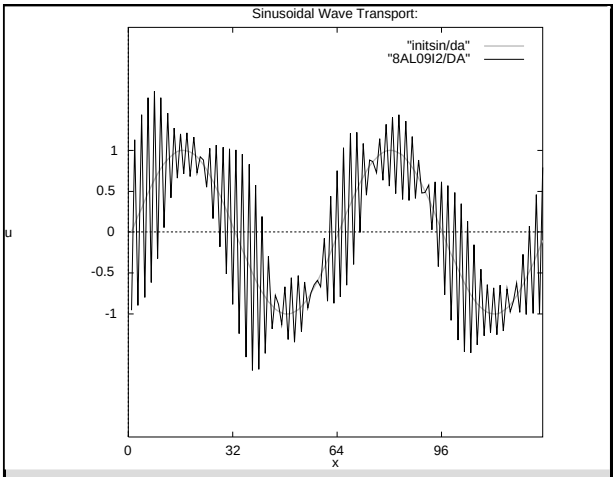

Figure (3.1.8a): SHASTA flux corrected transport scheme results after 5000 iterations for $\lambda = 0.9$.

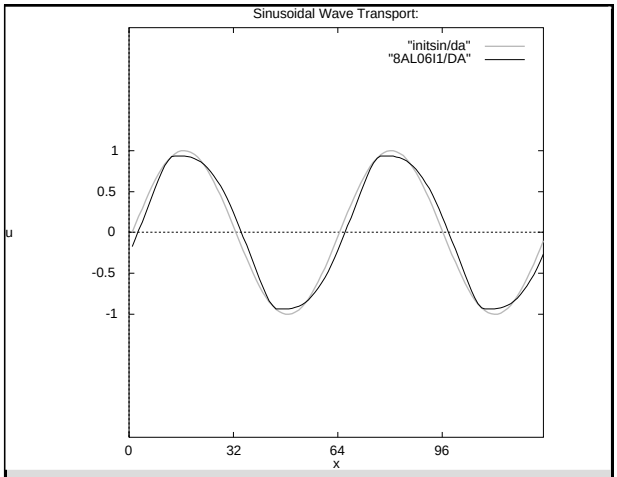

Figure (3.1.8b): SHASTA flux corrected transport scheme results after 5000 iterations for $\lambda = 0.6$.

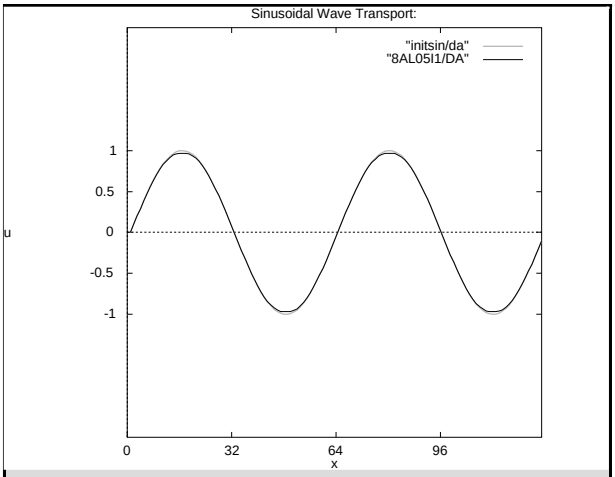

Figure (3.1.8c): SHASTA flux corrected transport scheme results after 5000 iterations for $\lambda = 0.5$.

(3.2a) Transport of Discontinuous (Square) Wave Profiles

This section consists of a series of results from the analysis of the 8 algorithms when applied to the transport of first-order discontinuous (square) waves. Also, λ was set to 0.5 throughout.

(3.2a.1) Centre Difference Results:

We have already established that the centre difference technique is completely unstable, and this was again found to be the case (see fig. 3.2a.1), although the algorithm failed even earlier than before.

(3.2a.2) First Order Upwind Results:

This scheme was found to very quickly dissolve the square wave profile and after 500 iterations had produced a sinusoidal profile suffering from significant diffusion (figure (3.2a.2)).

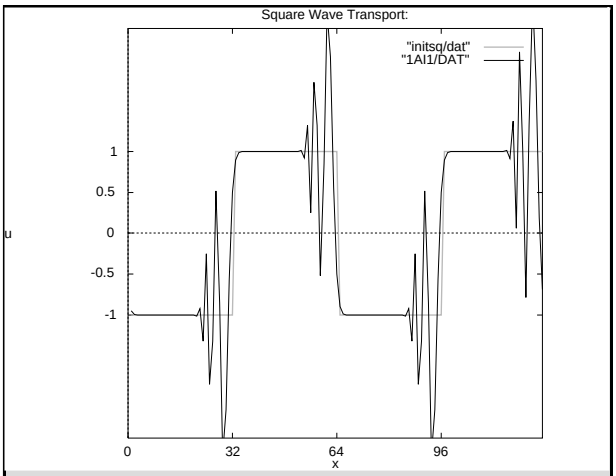

Figure (3.2a.1): Centre difference results after 10 iterations for square waves.

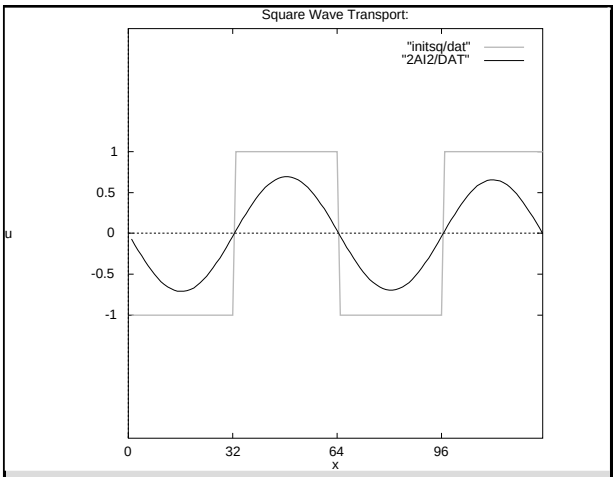

Figure (3.2a.2): First order upwind results after 500 iterations for square waves.

(3.2a.3) Lax-Wendroff Results:

Although this algorithm had previously performed well, the discontinuities in the profile quickly (after 50 iterations) lead to significant dispersion (figure (3.2a.3)), eventually leading to a smooth wave profile.

(3.2a.4) MacCormack:

This algorithm produced results almost exactly the same as those produced by the Lax-Wendroff algorithm (see fig. (3.2a.4)).

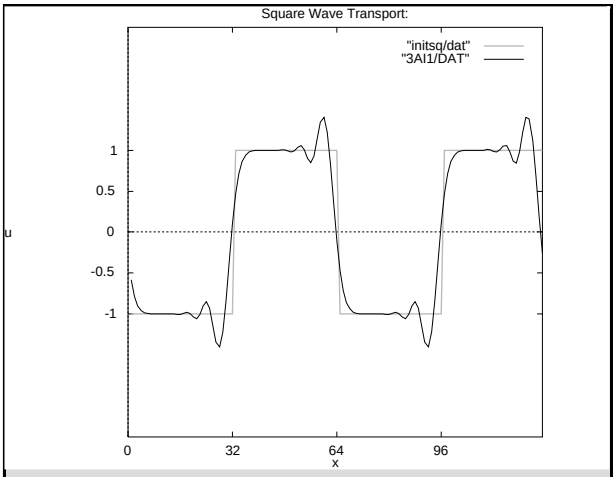

Figure (3.2a.3): Lax-Wendroff results after 50 iterations for square waves.

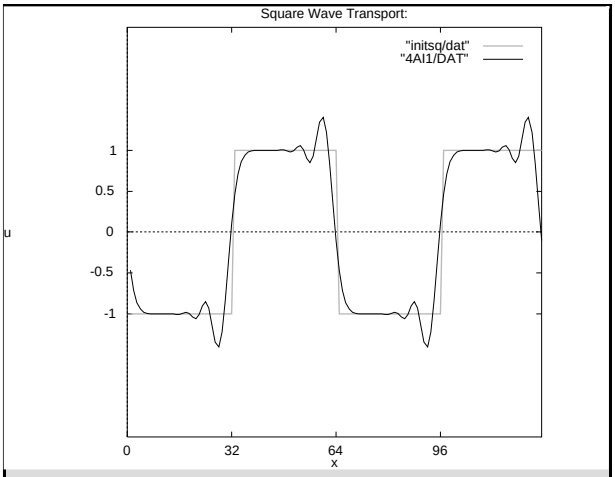

Figure (3.2a.4): MacCormack results after 50 iterations for square waves.

(3.2a.5) Second Order Monotonicity Preserving Results:

While this routine performed better than the pervious algorithms, there was still significant lop-sided deformation after 500 iterations (see fig. (3.2a.5)).

(3.2a.6) Third Order Monotonicity Preserving Results:

This routine performed better than the second order code, with only slight deformation of the wave profile due to the initial effect of the artificial dissapation (see fig. (3.2a.6)). The routine was found to still give reasonable results even after 5000 iterations.

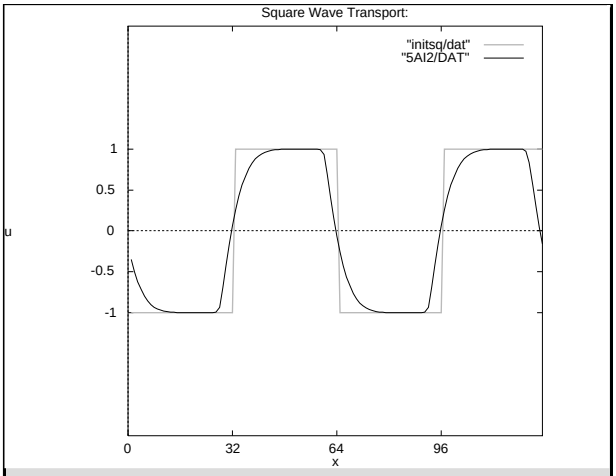

Figure (3.2a.5): Second order monotonicity preserving scheme results after 500 iterations for square waves.

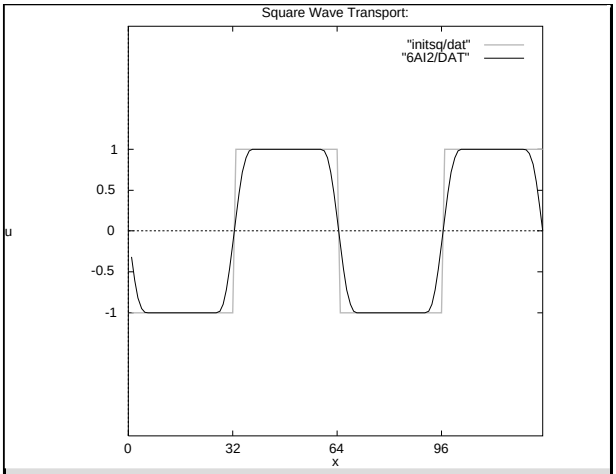

Figure (3.2a.6): Third order monotonicity preserving scheme results after 500 iterations for square waves.

(3.2a.7) Flux Corrected Transport (Donor Cell) Results:

As figure (3.2a.7) shows, this routine performed well, and in fact dealt with the shock as well as the third order monotonicity preserving scheme.

(3.2a.8) Flux Corrected Transport (SHASTA) Results:

This algorithm also coped well (see fig. (3.2a.8)), indeed just as well as the donor cell flux corrected transport and third order monotonicity preserving schemes.

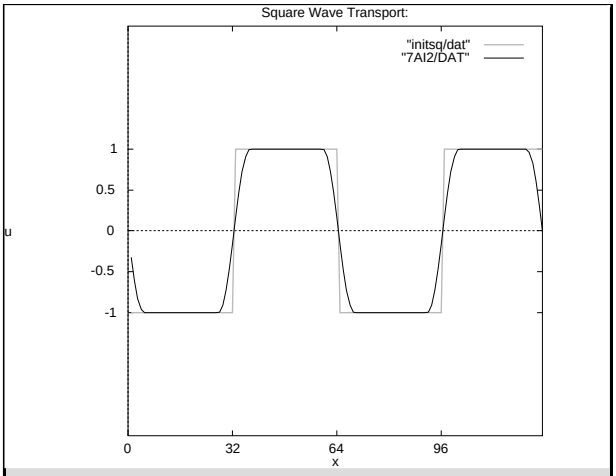

Figure (3.2a.7): Donor cell flux corrected transport scheme results after 500 iterations for square waves.

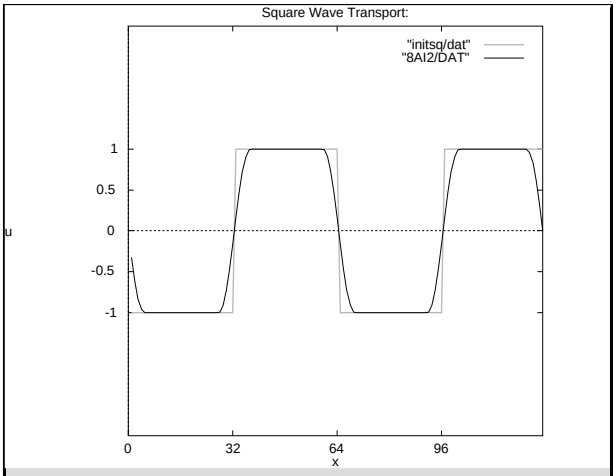

Figure (3.2a.8): SHASTA flux corrected transport scheme results after 500 iterations for square waves.

(3.2b) Transport of Discontinuous (Triangular) Wave Profiles

This section consists of a series of results from the analysis of the 8 algorithms when applied to the transport of second-order discontinuous (triangular) waves. As in the case of the square wave transport analysis, λ was set to 0.5 throughout.

(3.2b.1) Centre Difference Results:

Again this algorithm was found to be completely unstable (see fig. 3.2b.1).

(3.2b.2) First Order Upwind Results:

In this case, the routine performed better than with square waves, but still tended to quickly reduce the wave profile to a sinusoidal form (figure (3.2b.2)).

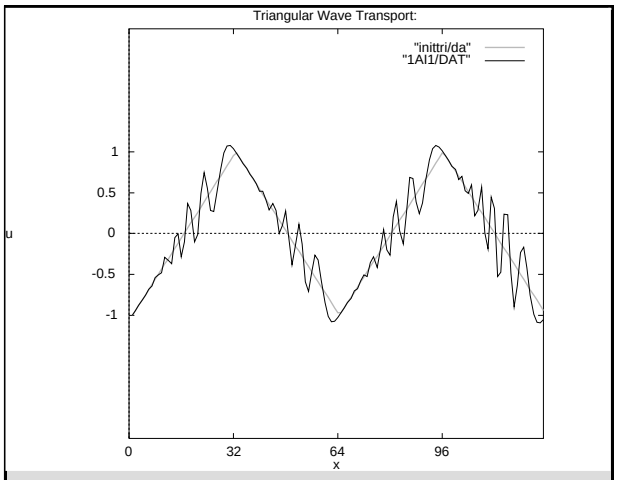

Figure (3.2b.1): Centre difference results after 30 iterations for triangular waves.

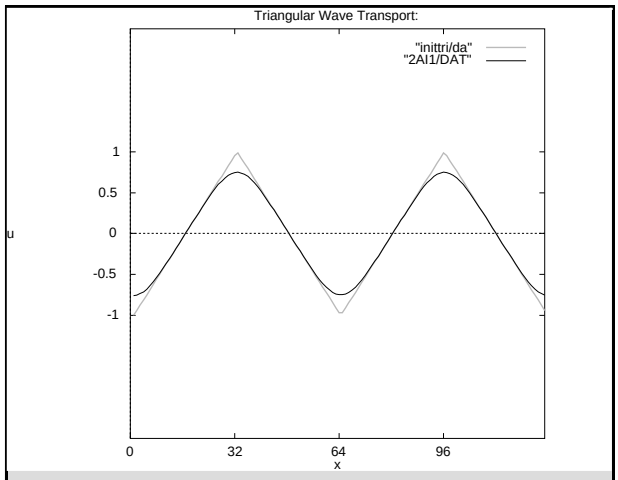

Figure (3.2b.2): First order upwind results after 100 iterations for triangular waves.

(3.2b.3) Lax-Wendroff Results:

While the dispersive properties of this algorithm did lead to some softening of the wave profile (see figure (3.2b.3)), the profile was stable for a few hundreds of iterations, and did not tend to a sinusoidal wave profile.

(3.2b.4) MacCormack:

Yet again this routine produced results very similar to those of the Lax Wendroff algorithm (see fig. (3.2b.4)).

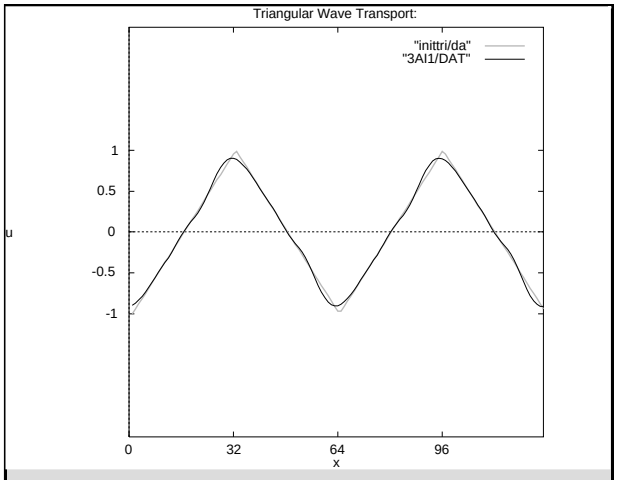

Figure (3.2b.3): Lax-Wendroff results after 200 iterations for triangular waves.

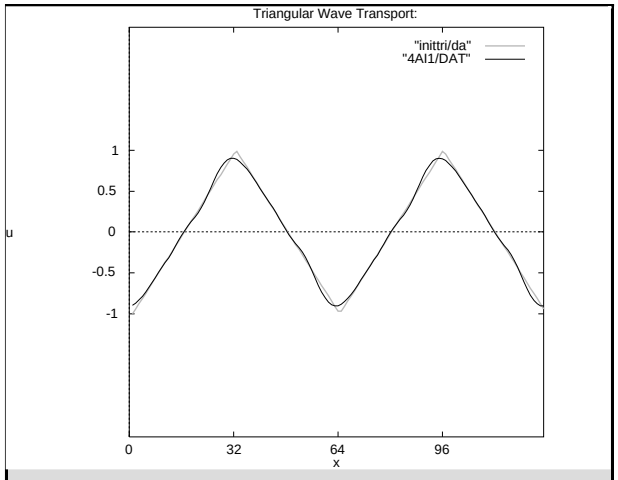

Figure (3.2b.4): MacCormack results after 200 iterations for triangular waves.

(3.2b.5) Second Order Monotonicity Preserving Results:

The lop-sided deformation apparently inherent in this algorithm appeared again for triangular waves, becoming significant after 200 iterations (see fig. (3.2b.5)).

(3.2b.6) Third Order Monotonicity Preserving Results:

Again this routine was found to perform well, with a slight rounding of the corners of the triangular waves allowing the profile to be consistent for a few hundred (see fig. (3.2b.6)) or even a few thousand iterations.

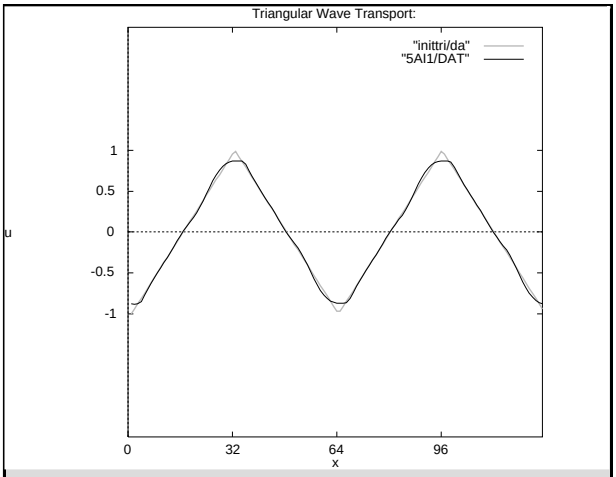

Figure (3.2b.5): Second order monotonicity preserving scheme results after 200 iterations for triangular waves.

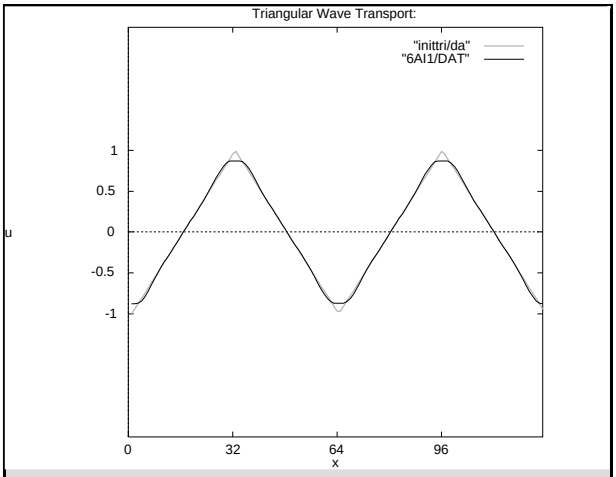

Figure (3.2b.6): Third order monotonicity preserving scheme results after 500 iterations for triangular waves.

(3.2b.7) Flux Corrected Transport (Donor Cell) Results:

Again this routine performed well (see fig. 3.2b.7)), and again it was fractionally better than the third order monotonicity preserving scheme.

(3.2a.8) Flux Corrected Transport (SHASTA) Results:

The SHASTA routine also coped well (see fig. (3.2b.8)), and again was not noticeably better than the donor cell flux corrected transport scheme.

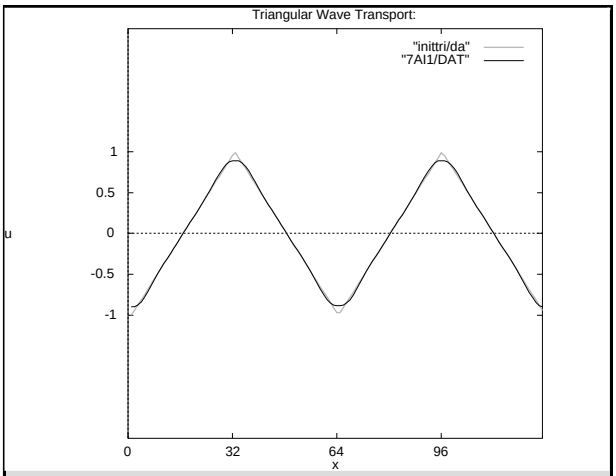

Figure (3.2b.7): Donor cell flux corrected transport scheme results after 500 iterations for triangular waves.

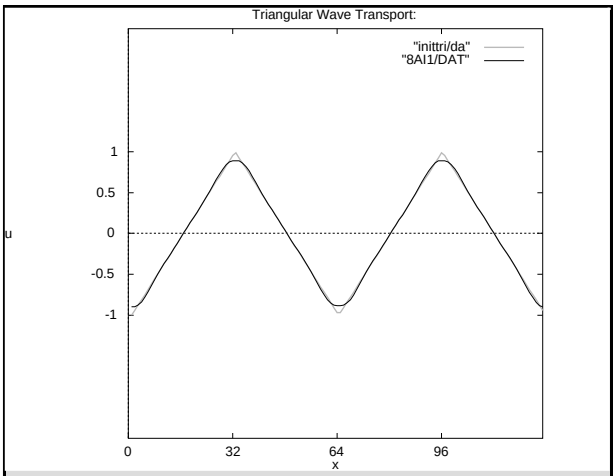

Figure (3.2b.8): SHASTA flux corrected transport scheme results after 500 iterations for triangular waves.

(4) Conclusions

When analysing the results of this assignment, it is important to remember that in realistic fluid flow situations, the actual problem would be more complex than the simple advection equation, (the main complication being that the velocity coefficient 'a' would no longer be constant).

(4.1) Smooth Wave Transport:

Clearly, the centre difference algorithm has no practical use. Also, while the first order upwind routine was stable, the effect of diffusion is too great for it to be used for any problem that requires any more than a few hundred iterations. The Lax-Wendroff and MacCormack routine both performed well under smooth conditions, and while other the routines also worked well, the cost of implementation brings no significant improvement in results. The straightforward second order routines are simpler (and so more flexible), quicker and less restricting (in terms of allowed values of λ).

(4.2) Square Wave Transport:

As one would expect from the theory, all those algorithms without some form of switched artificial dissipation performed badly under shock conditions. However, all of the algorithms with some form of artificial dissipation performed well, with the third order monotonicity preserving scheme and both flux corrected transport schemes all performed excellently. From this it is clear that for any fluid simulation running close to or above the sound speed for that fluid, one of the aforementioned three routines should be used. Unfortunately, due to the simplicity of this assignment, it is not possible to test the algorithms hard enough to determine which gives the highest accuracy.

(4.3) Triangular Wave Transport:

Predictably, the centre difference and first order upwind schemes both performed badly for triangular wave transport, and the monotonicity preserving and flux corrected transport schemes all performed well. The main point note is that under these conditions, the Lax-Wendroff and MacCormack routines both performed acceptably well, which implies that for situations where shock are not involved but there may be other sharp changes, such as a boundary layer, then the Lax-Wendroff/MacCormack approch will cope acceptably well and give reliable results (whilst being simple, flexible and fast). However, if such sharp changes are dominant in a simulation, or those sharp changes are to be analysed to a high degree of accuracy, then one of the routines recommended for shock conditions should be used.

Appendix A: FORTRAN program code:

```
C      PROGRAM CFD Assignment:
C      Andrew Jackson: v1.1 4th March 1997.
C
C Define common variables:
      PARAMETER (MS=128)
      DIMENSION U(-4:MS+4),U1(-4:MS+4),U2(-4:MS+4)
      COMMON /CFDREL/ U,U1,U2,lam,ALG,iters
      REAL*8 U,U1,U2,lam
      INTEGER*2 K
      INTEGER I,ALG,iters,stps
C
      K=0
         WRITE(*,*) 'CFD Assignment:'
         WRITE(*,*) '~~~~~~~~~~~~~~~'
     WRITE(*,*) 'Enter lambda:'
     READ(*,*) lam
     WRITE(*,*) ' '
     WRITE(*,*) '(1) - Centred difference.'
     WRITE(*,*) '(2) - First order upwind/donor cell.'
     WRITE(*,*) '(3) - Lax-Wendroff.'
     WRITE(*,*) '(4) - MacCormack.'
     WRITE(*,*) '(5) - 2nd order monotonicity preserving.'
     WRITE(*,*) '(6) - 3rd order monotonicity preserving.'
     WRITE(*,*) '(7) - FCT donor cell.'
     WRITE(*,*) '(8) - FCT SHASTA.'
     READ(*,*) ALG
C
      stps=10
      CALL InitConds
      CALL PlotSys
      CALL GET_KEY@(K)
      IF (K.EQ.Z'13B') GOTO 777
  666    DO I=1,stps
         CALL FluxCalc
        ENDDO
        IF (iters.EQ.10*stps) stps=10*stps
        CALL PlotSys
        CALL GET_KEY@(K)
      IF (K.NE.Z'13B') GOTO 666
 777  CALL TEXT_MODE@
      STOP
      END
C
C ---------------------------------------------------------------------------
C
      SUBROUTINE InitConds
C Define common variables:
      PARAMETER (MS=128)
      DIMENSION U(-4:MS+4),U1(-4:MS+4),U2(-4:MS+4)
      COMMON /CFDREL/ U,U1,U2,lam,ALG,iters
      REAL*8 U,U1,U2,lam,tmp
      INTEGER I,ALG,iters,wave
C
      WRITE(*,*) '(1) - Sin wave profile.'
      WRITE(*,*) '(2) - Square wave profile.'
      WRITE(*,*) '(3) - Triangular wave profile.'
```

```
      READ(*,*) wave
C
      iters=0
      DO I=-4,MS+4
C Sin wave:
       IF (wave.EQ.1) THEN
        U(I)=SIN(2.0d0*(I-1)*2.0d0*3.141592654/(MS-1))
       ENDIF
C Square wave:
       IF (wave.EQ.2) THEN
        tmp=REAL(I-1)/REAL(MS-1)
        tmp=tmp-INT(tmp)
        IF (tmp.LT.0.25 .OR. (tmp.GT.0.5.AND.tmp.LT.0.75)) THEN
        U(I)=-1.0d0
        ELSE
         U(I)=+1.0d0
        ENDIF
       ENDIF

C Trianglier wave:
       IF (wave.EQ.3) THEN
        tmp=REAL(I-1)/REAL(MS-1)
        tmp=tmp-INT(tmp)
        IF (tmp.LT.0.25) THEN
        U(I)=8.0d0*(tmp-0.125)
        ELSEIF (tmp.GE.0.25 .AND. tmp.LT.0.5) THEN
         U(I)=-8.0d0*(tmp-0.125-0.25)
        ELSEIF (tmp.GE.0.5 .AND. tmp.LT.0.75) THEN
         U(I)=8.0d0*(tmp-0.125-0.25-0.25)
        ELSEIF (tmp.GE.0.75) THEN
         U(I)=-8.0d0*(tmp-0.125-0.25-0.25-0.25)
        ENDIF
       ENDIF
      ENDDO
C
      RETURN
      END
C
C ---------------------------------------------------------------------------
C
      SUBROUTINE PlotSys
C Define common variables:
      PARAMETER (MS=128)
      DIMENSION U(-4:MS+4),U1(-4:MS+4),U2(-4:MS+4)
      COMMON /CFDREL/ U,U1,U2,lam,ALG,iters
      REAL*8 U,U1,U2,lam
      INTEGER*2 IH,IV,ICOL
      INTEGER I,ALG,iters,offset,J
C
      OPEN(UNIT=21,FILE='CFD.DAT',STATUS='UNKNOWN')
      WRITE(21,*) '# Results from algorithm number ',ALG,'.'
      WRITE(21,*) '# After ',iters,' iterations.'
      WRITE(21,*) '# lambda = ',lam
      WRITE(21,*) '#'
      CALL VGA@
      ICOL=15
C
      WRITE(*,*) iters
```

```
      offset=iters*lam
      offset=offset-INT(offset/(MS-1))*(MS-1)
C
      DO I=1,MS-1
       J=I+offset
       IF (J.GT.MS-1) J=J-(MS-1)
       IF (J.LT.1) J=J+(MS-1)
       IH=400*I/MS
       IV=200-50*U(J)
       WRITE(21,*) I,U(J)
       CALL SET_PIXEL@(IH,IV,ICOL)
      ENDDO
C
      IV=200
      DO I=1,MS-1,2
       IH=400*I/MS
       CALL SET_PIXEL@(IH,IV,ICOL)
      ENDDO
C
      IV=250
      DO I=1,MS-1,4
       IH=400*I/MS
       CALL SET_PIXEL@(IH,IV,ICOL)
      ENDDO
C
      CLOSE(UNIT=21)
      RETURN
      END
C
C ---------------------------------------------------------------------
C
      SUBROUTINE FluxCalc
C Define common variables:
      PARAMETER (MS=128)
      DIMENSION U(-4:MS+4),U1(-4:MS+4),U2(-4:MS+4),G(-1:MS+1)
      DIMENSION D(-1:MS+1)
      COMMON /CFDREL/ U,U1,U2,lam,ALG,iters
      REAL*8 U,U1,U2,lam,D,s,delta,plam,nlam,delt2,delt3
      REAL*8 s1,s2
      INTEGER I,ALG,iters

C Centre difference:
      IF (ALG.EQ.1) THEN
       DO I=1,MS-1
        U1(I) = U(I) - 0.5d0*lam*(U(I+1)-U(I-1))
       ENDDO

C First order upstream:
      ELSEIF (ALG.EQ.2) THEN
       DO I=1,MS-1
        IF (lam.GT.0.0) U1(I) = U(I) - lam*(U(I)-U(I-1))
        IF (lam.LT.0.0) U1(I) = U(I) - lam*(U(I+1)-U(I))
       ENDDO

C Lax-Wendroff:
      ELSEIF (ALG.EQ.3) THEN
       DO I=1,MS-1
        U1(I)=U(I)-0.5d0*lam*(U(I+1)-U(I-1))+
```

```
     +            0.5d0*lam*lam*(U(I+1)-2.0d0*U(I)+U(I-1))
       ENDDO

C MacCormack:
      ELSEIF (ALG.EQ.4) THEN
       DO I=1,MS-1
        U2(I) = U(I) - lam*(U(I+1)-U(I))
       ENDDO
       U2(MS)=U2(1)
       U2(0)=U2(MS-1)
       DO I=1,MS-1
        U1(I) = 0.5d0*(U(I)+U2(I) - lam*(U2(I)-U2(I-1)))
       ENDDO

C 2nd/3rd order monotonicity preserving:
      ELSEIF (ALG.EQ.5 .OR. ALG.EQ.6) THEN
       DO I=0,MS-1
c 2nd or 3rd order approx:
        IF (ALG.EQ.5) THEN
         D(I)=U(I+1)-U(I)
        ELSEIF (ALG.EQ.6) THEN
         IF (lam.GT.0.0) THEN
          D(I)=U(I+1)-U(I)-(1.0d0+lam)*(U(I+1)-2.0d0*U(I)+U(I-1))/3.0d0
         ELSE
          D(I)=U(I)-U(I-1)-(1.0d0-lam)*(U(I)-2.0d0*U(I+1)+U(I+2))/3.0d0
         ENDIF
        ENDIF
c Flux calc:
        IF (lam.GT.0.0) THEN
         s1=DSIGN(1.0d0,U(I+1)-U(I))
         s2=DSIGN(1.0d0,U(I)-U(I-1))
         IF (s1.GT.s2 .OR. s1.LT.s2) THEN
          s=0.0d0
         ELSE
          s=s2
         ENDIF
         D(I)=ABS(D(I))
         D(I)=s*MIN(D(I),2.0d0*ABS(U(I+1)-U(I)),2.0d0*ABS(U(I)-U(I-1)))
        ELSE
         s1=DSIGN(1.0d0,U(I+2)-U(I+1))
         s2=DSIGN(1.0d0,U(I+1)-U(I))
         IF (s1.GT.s2 .OR. s1.LT.s2) THEN
          s=0.0d0
         ELSE
          s=s1
         ENDIF
         D(I)=s*MIN(ABS(D(I)),2.0d0*ABS(U(I+2)-U(I+1)),
     +                        2.0d0*ABS(U(I+1)-U(I)))
        ENDIF
       ENDDO
C
       DO I=0,MS-1
         IF (lam.GT.0.0) G(I)=U(I)+0.5d0*(1.0d0-lam)*D(I)
         IF (lam.LE.0.0) G(I)=U(I+1)+0.5d0*(1.0d0+lam)*D(I)
       ENDDO

C Flux corrected transport: Donor cell & SHASTA:
      ELSEIF (ALG.EQ.7 .OR. ALG.EQ.8) THEN
```

```
        IF (ALG.EQ.7) THEN
         plam=MAX(0.0d0,lam)
         nlam=MIN(0.0d0,lam)
         DO I=-1,MS+1
          U2(I)=U(I)-(nlam*U(I+1)+(plam-nlam)*U(I)-plam*U(I-1))
         ENDDO
        ELSE
         plam=0.5d0-lam
         plam=plam*plam
         nlam=0.5d0+lam
         nlam=nlam*nlam
         DO I=-1,MS+1
          U2(I)=U(I)+0.5d0*(plam*U(I+1)-(nlam+plam)*U(I)+nlam*U(I-1))
         ENDDO
        ENDIF
        U2(MS)=U2(1)
        U2(MS+1)=U2(2)
        U2(MS+2)=U2(3)
        U2(0)=U2(MS-1)
        U2(-1)=U2(MS-2)
        U2(-2)=U2(MS-3)
        DO I=0,MS-1
         IF (ALG.EQ.7) THEN
          D(I)=ABS(0.5d0*ABS(lam)*(1.0d0-ABS(lam))*(U2(I+1)-U2(I)))
         ELSE
          D(I)=ABS((U2(I+1)-U2(I))/8.0d0)
         ENDIF
         delt3=U2(I)-U2(I-1)
         delta=U2(I+1)-U2(I)
         delt2=U2(I+2)-U2(I+1)
         s=DSIGN(1.0d0,delta)
         D(I)=s*MAX(0.0d0,MIN(s*delt3,D(I),s*delt2))
        ENDDO
        DO I=1,MS-1
         U1(I)=U2(I)-(D(I)-D(I-1))
        ENDDO
       ENDIF

       IF (ALG.EQ.5 .OR. ALG.EQ.6) THEN
C Flux calculation:
        DO I=1,MS-1
         U1(I)=U(I)-lam*(G(I)-G(I-1))
        ENDDO
       ENDIF

C Move new U1 into old U:
       DO I=1,MS-1
        U(I)=U1(I)
       ENDDO

C Fixing periodic boundary:
       U(MS)=U(1)
       U(MS+1)=U(2)
       U(MS+2)=U(3)
       U(0)=U(MS-1)
       U(-1)=U(MS-2)
       U(-2)=U(MS-3)
C Increment iteration counter:
```

```
      iters=iters+1
      RETURN
      END
C
C ---------------------------------------------------------------------
C ---------------------------------------------------------------------
C
```