Homework III : Coding of Finite Difference Approximations(Hyperbolic PDE) - Part 1 and 2

Dhruv Balwada

September 29, 2014

1 Statement of problem

We are required to solve the initial-boundary value problem :

$$PDE : Q_t + AQ_x \equiv \begin{bmatrix} u \\ v \end{bmatrix}_t + \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}_x = 0, x \in [0,1] \tag{1}$$

$$BCs : \{ \begin{matrix} u(0,t) = 0 \\ Radiation\ at\ x = 1 \end{matrix} \tag{2}$$

$$IC : Q(x,0) = Q_o(x) \tag{3}$$

where the initial conditions will be

$$\begin{bmatrix} u(x,0) \\ v(x,0) \end{bmatrix} = \begin{bmatrix} e^{-100(x-1/2)^2} \\ 0 \end{bmatrix} \tag{4}$$

using finite difference approximations. This has to be achieved using modular approach, where each module/subroutine should be presented and tested individually.

2 Description of The Mathematics

2.1 Hyperbolic systems and Compatibility conditions

The hyperbolic system presented here has already been solved analytically in homework1. Here we present the highlights of the analytical method and the analytical solution.

The main idea is to using the method of characteristics. This transforms the equations to a set of uncoupled equations where some property is conserved along some characteristic curves. For the system presented here the conserved properties are $w^+ = u + v$ and $w^- = u - v$. These properties stay constant along the characteristic curves $x - 3t$ and $x + t$ projecting out from the initial conditions. When these characteristic curves interact with the boundary they reflect based on the appropriate reflection or radiation conditions.

Transforming the equations into the characteristic equations is useful when finding analytical solutions but when the solutions are calculated numerically we need to solve the

system in terms of physical variables. The boundary conditions on the characteristics curves also need to be transformed to the appropriate boundary conditions on all the physical variables that are part of the system. These boundary conditions are called Compatibility conditions. These are derived from the characteristic equations for which the boundary conditions of reflection or radiation are easily imposed. From homework 1 we know that the compatibility conditions are

$$u(0) = u_t(0) = 0 \tag{5}$$

$$v_t(0) = -(u_x(0) - v_x(0)) \tag{6}$$

$$u_t(1) = v_t(1) = -\frac{3}{2}(u_x(1) + v_x(1)) \tag{7}$$

The analytical solution was obtained for different regions based on whether the region received information from the boundary points or initial points. The exact solution is the following

Region I ($x - 3t >= 0, x + t <= 1$)

$$u(x,t) = \frac{1}{2}[u_o(x-3t) + v_o(x-3t) + u_o(x+t) - v_o(x+t)] \tag{8}$$

$$v(x,t) = \frac{1}{2}[u_o(x-3t) + v_o(x-3t) - u_o(x+t) + v_o(x+t)] \tag{9}$$

Region 2 ($x - 3t >= 0, x + t >= 1$)

$$u(x,t) = \frac{1}{2}[u_o(x-3t) + v_o(x-3t)] \tag{10}$$

$$v(x,t) = \frac{1}{2}[u_o(x-3t) + v_o(x-3t)] \tag{11}$$

Region 3 ($x - 3t <= 0, x + t <= 1$)

$$u(x,t) = \frac{1}{2}[-u_o(t-x/3) + v_o(t-x/3) + u_o(x+t) - v_o(x+t)] \tag{12}$$

$$v(x,t) = \frac{1}{2}[-u_o(t-x/3) + v_o(t-x/3) - u_o(x+t) + v_o(x+t)] \tag{13}$$

Region 4 ($x - 3t <= 0, x + t >= 1, t - x/3 - 1 <= 0$)

$$u(x,t) = \frac{1}{2}[-u_o(t-x/3) + v_o(t-x/3)] \tag{14}$$

$$v(x,t) = \frac{1}{2}[-u_o(t-x/3) + v_o(t-x/3)] \tag{15}$$

Region 5 ($t - x/3 - 1 >= 0$)

$$u(x,t) = 0 \tag{16}$$

$$v(x,t) = 0 \tag{17}$$

To make the problem slightly simpler $v_o = 0$ and all the v_o terms from the solution.

2.2 Numerical approximations to spatial derivatives

The derivatives are approximated using center differences on the spatial grid except at the boundaries where a forward and backward difference are used on the left and right boundary respectively.

The order of the finite difference approximation can be found using Taylor series approximation and difference calculus.

$$\delta^o = \frac{S - S^{-1}}{2\triangle x} \tag{18}$$

Where S is the forward shift operator and $\triangle x$ is the grid spacing.

We know $S = e^{\triangle x D}$ where D is the derivative operator. This gives that

$$\delta^o = \frac{e^{\triangle x D} - e^{-\triangle x D}}{2\triangle x} = \frac{sinh(\triangle x D)}{\triangle x} = D + \frac{\triangle x^2 D^3}{6} + O(\triangle x^4) = D + O(\triangle x^2) \tag{19}$$

Going through a similar analysis we can see that $\delta^+ = D + O(\triangle x)$ and $\delta^- = D + O(\triangle x)$

This scheme should give the exact results at all the interior points for a 2nd or lower degree polynomial and exact results for a linear polynomial on the boundary points.

2.3 Runge-Kutta Solver

Once the spatial differencing scheme is applied to the PDEs a set of ODEs is obtained at each spatial grid point which can be progressed/solved using an ODE integrator. Here we are instructed to use a 3rd order - two storage level -Runge-Kutta scheme. This scheme is 3rd order accurate and thus will lead to exact solutions for ODEs which has a 3rd or lower order polynomial as the solution.

The coefficients of this solver are : $a = 0, -5/9, -153/128$, $b = 0, 1/3, 3/4$ and $g = 1/3, 15/16, 8/15$

3 Description of the Algorithm

The code for solving this homework was written in modular form. In this section I provide the algorithm for each individual module and then the algorithm for the main program that connects the modules.

I am starting my arrays from index of 1 in my codes. So all the spatial and time indices as shown in the problem statement will be shifted by 1.

3.1 xDeriv Algorithm

GET the vector that needs to be differentiated(u), a vector to hold the derivative(dudx), number of spatial points(N) and the size of the grid(dx) as input.

$dudx_1 = (u_2 - u_1)/dx$

for j = 2 to N **do**

$dudx_j = (u_{j+1} - u_{j-1})/(2 * dx)$

end for
$dudx_{N+1} = (u_{N+1} - u_N)/dx$

3.2 tDeriv Algorithm

This calculation is a simple matrix multiplication operation on all the interior points. At the boundary points we need to use the compatibility conditions(eqns 5,6 and 7).

GET Q(vector/array of physical variables), dQdt(array to return the tendencies), N, neqn, dx, dt
for j=1 to neqn **do**
  Call xDeriv(Q(:,j))
end for
Define the coefficient matrix for the hyperbolic pde system A or get this as input.
for j=1 to neqn **do**
  for i=2 to N **do**
    dQdt(j,i) = A(j,1)*Q(i,1) + A(j,2)*Q(i,2) + ... A(j,neqn)*Q(i,neqn)
  end for
end for
{Now use compatibility conditions at j = 1 and N+1}
dQdt(1,1) = 0
dQdt(1,2) = -dQdx(1,1) + dQdx(1,2)
dQdt(N+1,1) = -3.0/2.0*(dQdx(N+1,1)+dQdx(N+1,2))
dQdt(N+1,2) = dQdt(N+1,1)

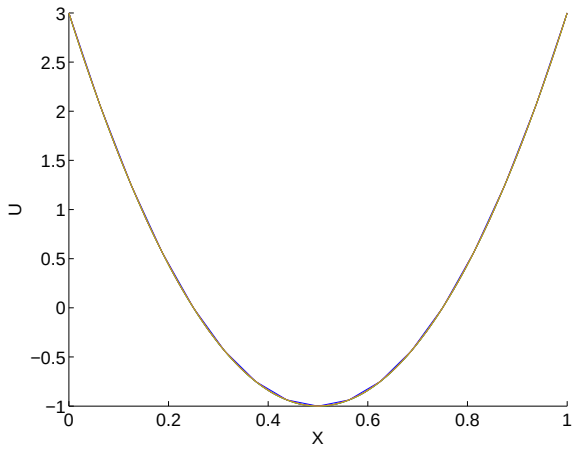
3.3 rkLs Algortihm

GET Q, neqn, N ,time, dx, dt
for k =1 to kmax **do**
  tk = time + dt*b(k)
  update or assign value to dQdt
  for l=1 to neqn **do**
    for j=1 to N+1 **do**
      G = a(k)*G + dQdt(tk,Q)
      Q(j,l) = Q(j,l) + g(k)*G*dt
    end for
  end for
end for

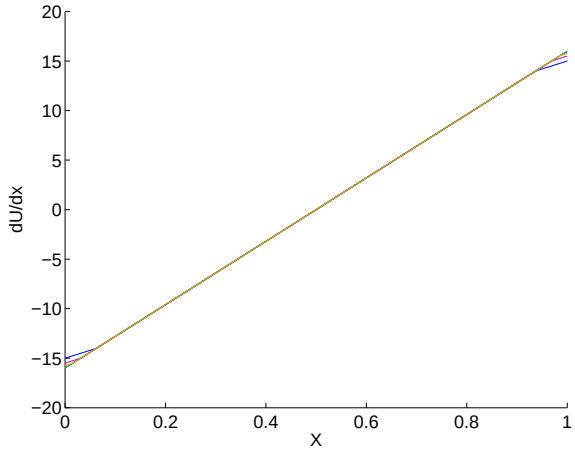
4 Results

4.1 xDeriv Testing

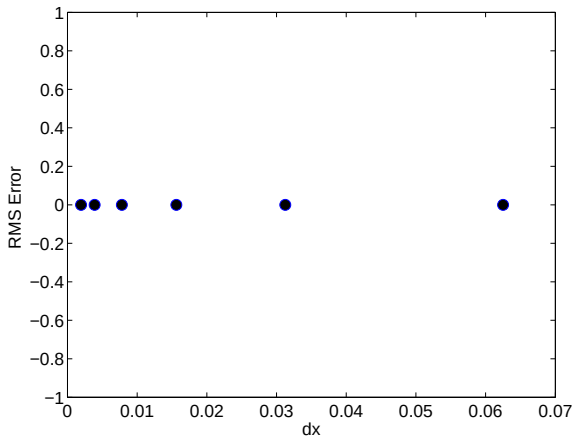
The xDeriv method is tested on a quadratic and a cubic polynomial. The quadratic function $f = 16x^2 - 16x + 3$ is plotted in Figure1a, the derivative of the function as calculated by



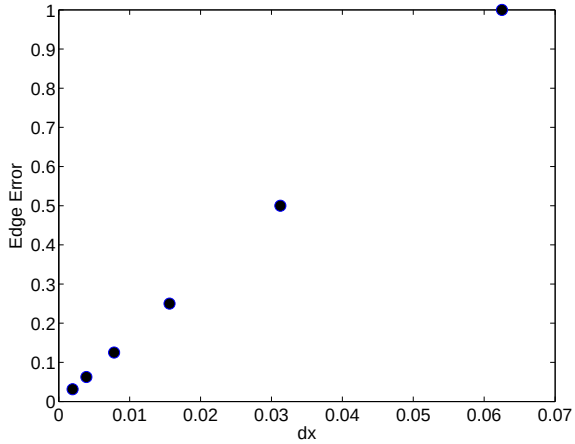
(a) $f = 16x^2 - 16x + 3$



(b) df/dx vs x for varying dx



(c) Interior RMS error vs dx



(d) Boundary error vs dx

Figure 1: Behavior of xDeriv on a quadratic input function

the module and for different dxs is plotted in Figure 1b. Figure 1c shows that the error in the calculation of the derivative of this polynomial away from the boundary points is zero. Figure1d shows the error on the boundary points varies linearly with dx.

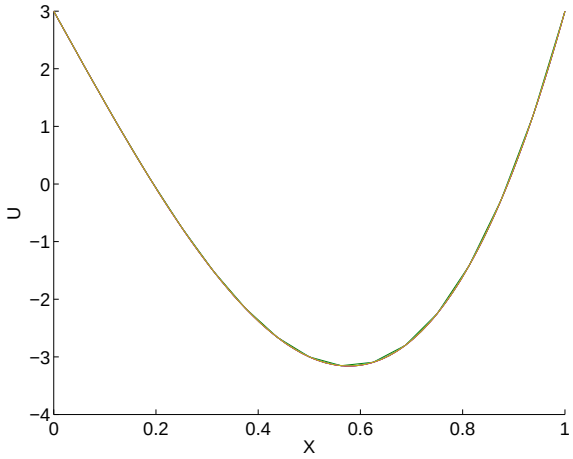
The cubic function $f = 16x^3 - 16x + 3$ is plotted in Figure2a, the derivative of the function as calculated by the module and for different dxs is plotted in Figure 2b. Figure 2c and shows the error on the interior points of the derivative increases quadratically.

It is clear from these results that the method is working as expected; with second degree accuracy in the interior and linear accuracy at the boundary points.

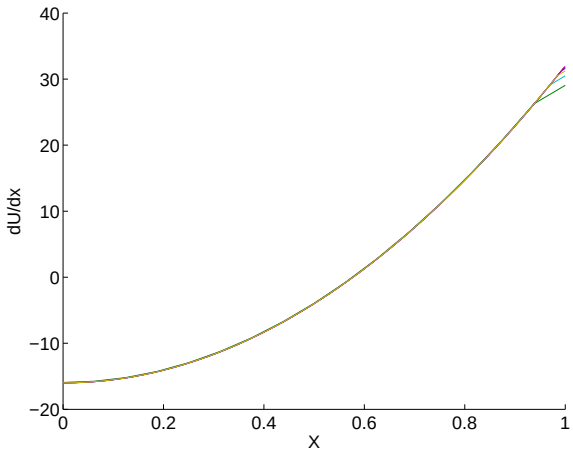
4.2 rkLs Testing

The rkLs method tested on ODEs with a cubic and 4th order polynomial as the solutions.

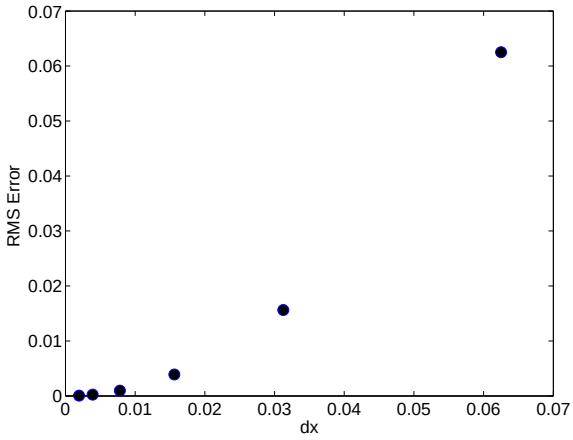
Figure 3a plots the solution to the ODE $du/dt = 10t^2$ where the solution is $u(t) = 10/3t^3$. Figure 3b shows that the ODE solver is exact for this ODE as the error is zero.



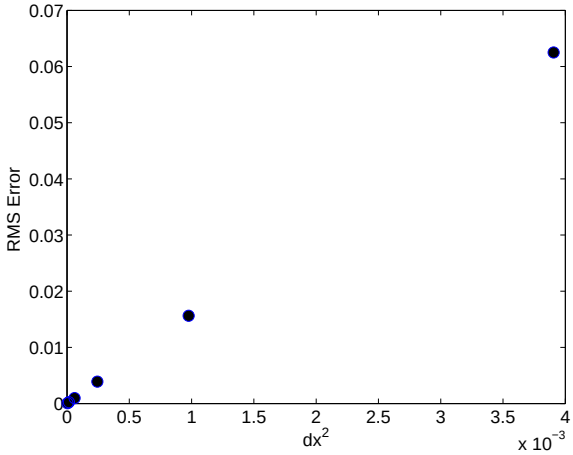
(a) $f = 16x^3 - 16x + 3$



(b) df/dx vs x for varying dx

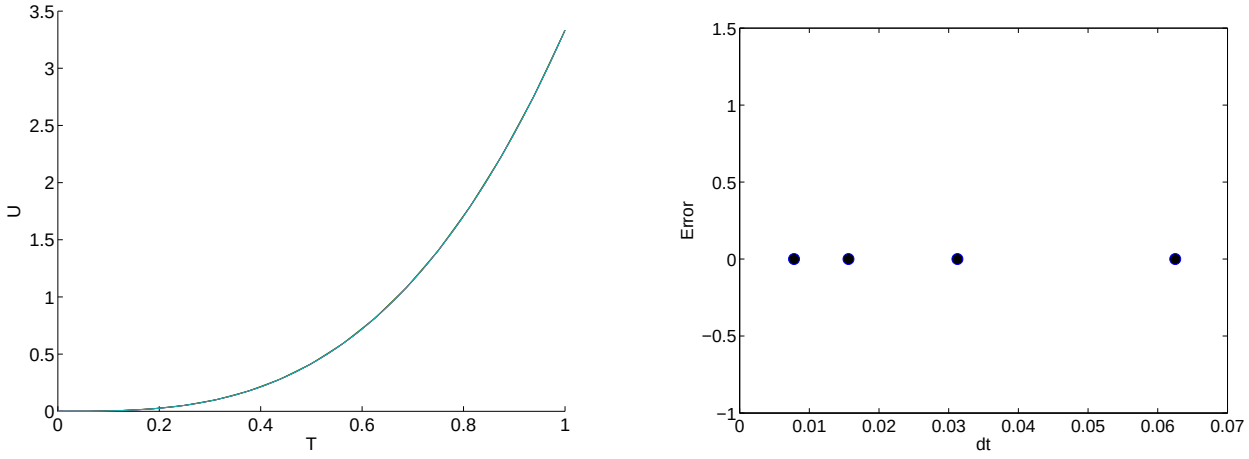


(c) RMS error vs dx



(d) RMS error vs dx^2

Figure 2: Behavior of xDeriv on a cubic input function



(a) (b)

Figure 3: Results for RKLS

Figure 4a plots the solution to the ODE $du/dt = 10t^3$ where the solution is $u(t) = 10/4t^4$. Figures 4b and c show that the error in the solution decreases as dt^3.

It is clear from these results that the Williamson's Runge-Kutta solver has third order accuracy.

4.3 Hyperbolic Solver Results

Figure 8c and d plot the numerical solution for N=128 and CFL=0.5(Nt=768) and Figure 8a,b plot the exact solution at the same grid points. We see that visually these solutions are almost indistinguishable. It is also clear that the solution has 2 characteristics. One going to the right with a slope of 3 and another one going to the left with a slope of 1. There are reflections at the left boundary and the sign is reversed. At the right boundary the radiation condition was imposed and the characteristics leave the domain with no reflection.

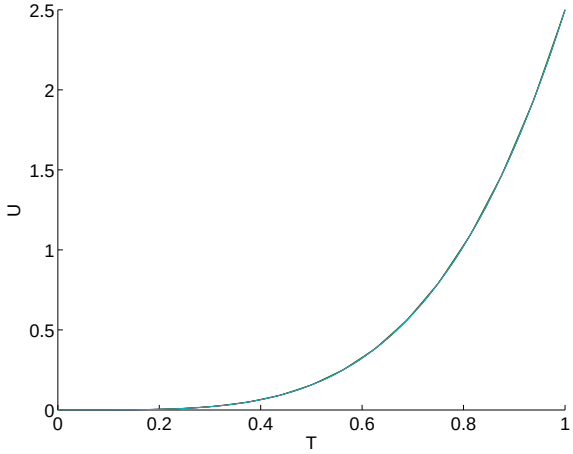
Next I follow Part 2 of the homework assignment to analyze the functioning and efficiency of code.

The solutions are computed upto t=1.0 for N=32, 64, 128 and 256 for a fixed CFL number $CFL = |\lambda_{max}|\triangle t/\triangle x = 0.5$. The $|\lambda_{max}|$ for this problem is 3. This gives us that $N_t = 6N$ and so the N_t = 192, 384, 768 and 1536. The CPU times for the numerical calculations are shown in Table 1 and one can see that the growth of CPU time is much faster than linear.

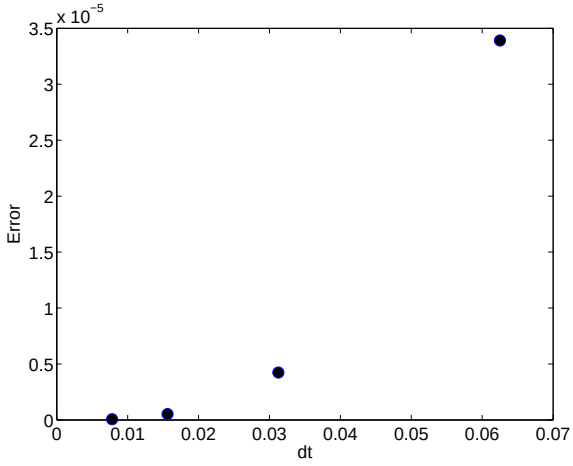
Figure 5 shows the comparison of U for different Ns at time 0.125. One can visually see that as the N increases the solution approaches the exact. The larger grid spacing solutions lag the exact solution in phase and are slightly smaller in amplitude. The solution for N=128 is almost indistinguishable from the exact solution.

Figure 6 shows that the error norm($||e||_2 = (\triangle x e_j e_j)^{0.5}$, where vector summation rules apply) as a function of N. As expected the error norm decreases with increasing N. The relationship between the error norm and dx is $||e||_2 = dx^1.92$, which is slightly smaller than the order of the center differencing in the interior 2.

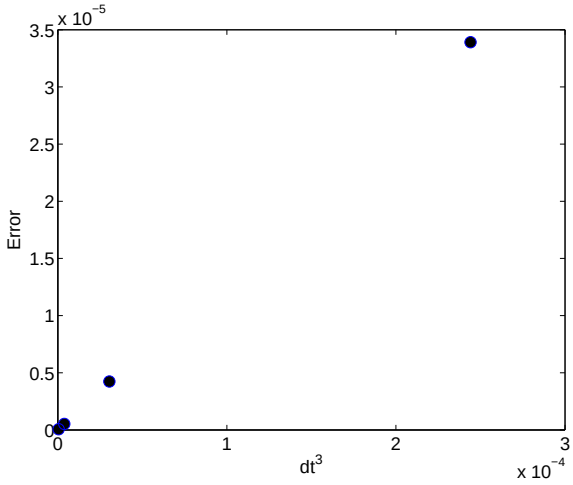
Then we test the effects of varying the courant number on the solution and the time



(a)



(b)



(c)

Figure 4: Results for RKLS

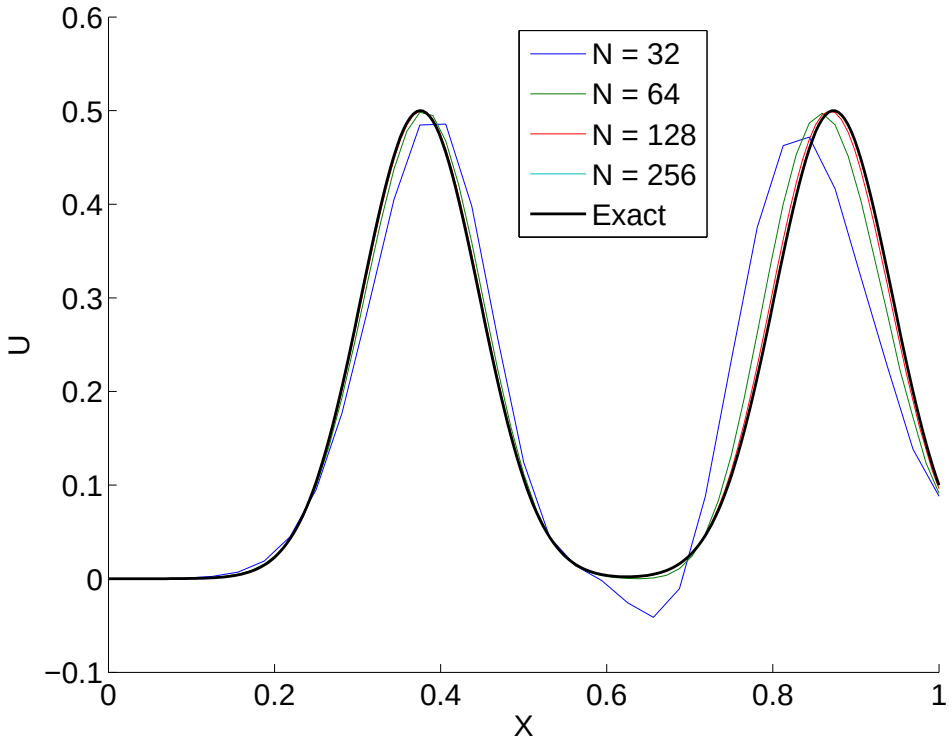


Figure 5: U as a function of X for time T=0.125 for different N

of blow up. We look at the case of N=256 as it is the best(in some sense) solution we have. We test on CFL number = 0.25 to 2.5 for which the Nt vary from 3072 to 308(where Nt has been rounded up from the value obtained by the relation between CFL, Nt and N, $CFL = 3N/N_t$). The time of blow up is defined as the time in a simulation when the error norm exceeds 10. We see that the solution does not blow up for $CFL < 2$. For CFL of 2 and higher the time of blow up decreases as what appears to be an exponential decrease.

N	CPU TIME(s)
32	0.003051
64	0.012299
128	0.044072
256	0.18715

Table 1: CPU times for different Ns

5 Conclusion

A solver for hyperbolic PDEs was implemented and tested. In addition general modules for differentiation, 3rd order RK ODE solver and calculating tendency terms were also implemented and tested. These could be used with future codes or swapped to change the

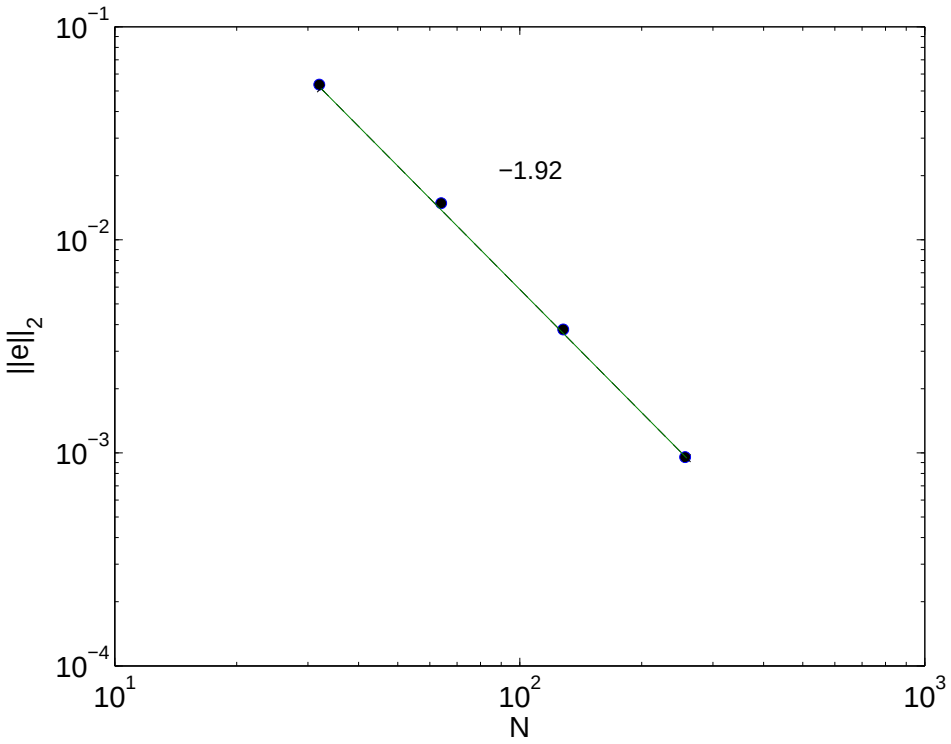


Figure 6: Error norm as a function of N. The slope is approx -1.94.

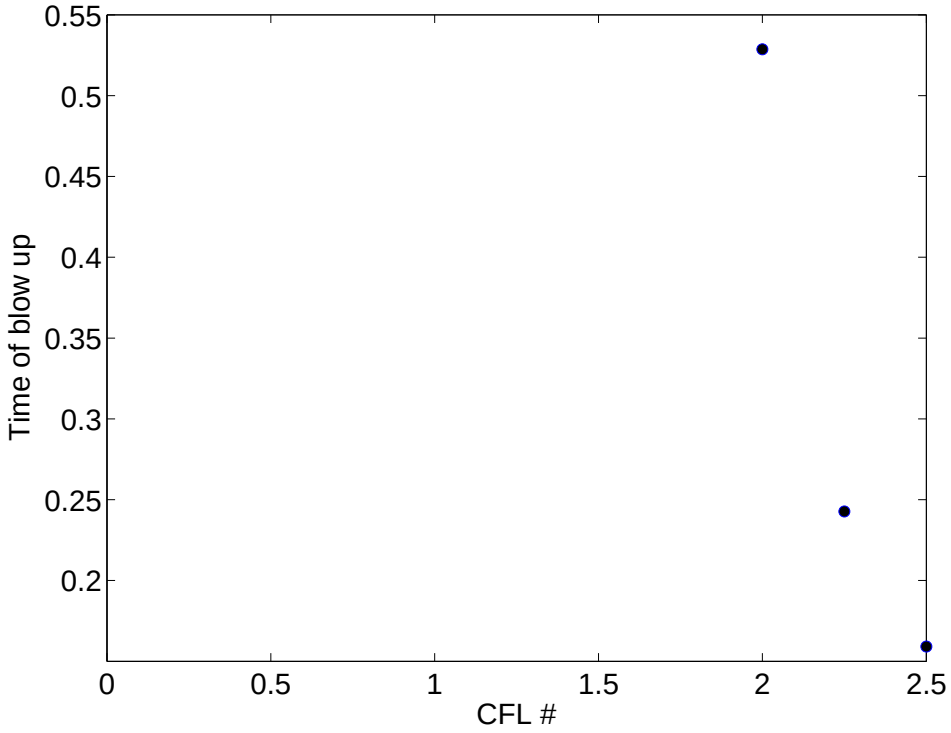
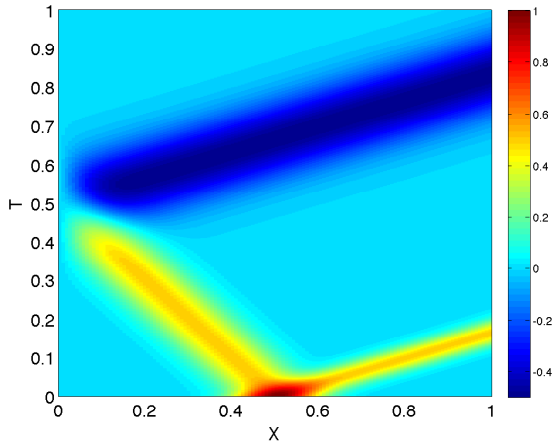
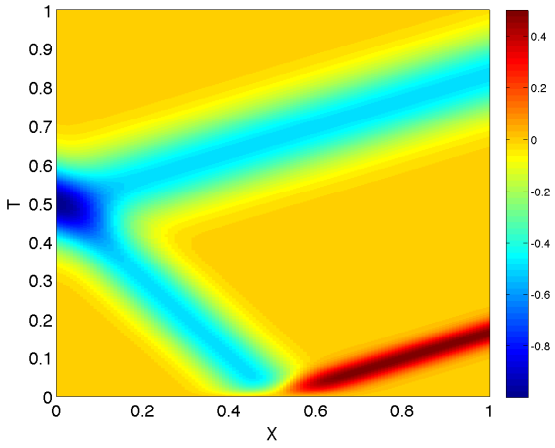


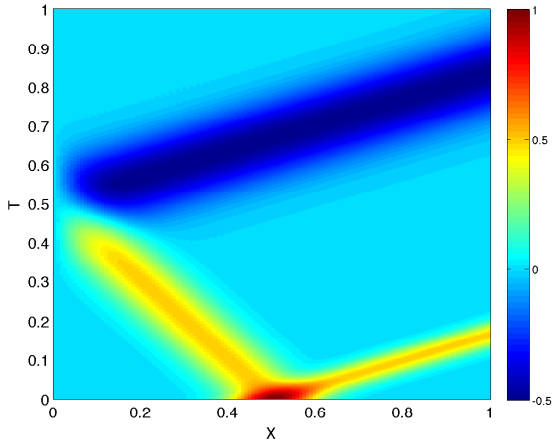
Figure 7: Time of blow up defined as time when error norm is greater than equal to 10 as a function of CFL number



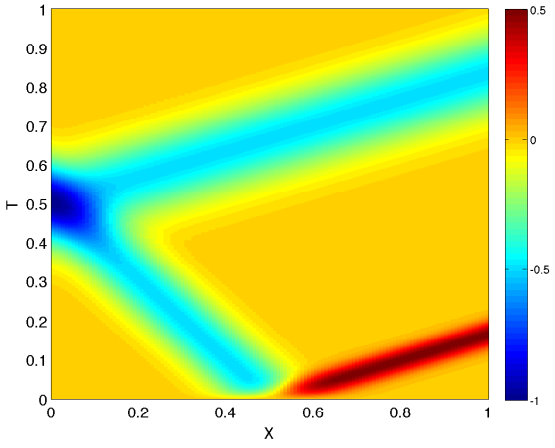
(a) Exact u



(b) Exact v



(c) Numerical u



(d) Numerical v

Figure 8

numerical properties of this solver. The solver works without any glitches. The characteristics travel left and right with the right wave speeds and no apparent loss of energy(in visual terms). It has an order of 1.92 in $\triangle x$ as observed by numerical testing. The solver is stable for grids with a CFL number smaller than 2 after which it blows up and the time of blow up decreases exponentially.

The side modules also worked as expected. The differencing operator gave second order accuracy in the interior and linear accuracy at the boundaries. The Williamson's Runge-Kutta solver showed a third order accuracy. The routine for calculating tendencies also worked as expected.