Numerical PDE II HW 5 : Finite Volume Methods

Dhruv Balwada

April 28, 2015

1 Statement of Problem

Solve the wave equation in two space dimension

$$Q_t + F_x + G_y = 0$$

$$Q = \begin{bmatrix} p \\ u \\ v \end{bmatrix}, F = AQ, G = BQ$$

$$A = \begin{bmatrix} 0 & \rho c^2 & 0 \\ 1/\rho & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, B = \begin{bmatrix} 0 & 0 & \rho c^2 \\ 0 & 0 & 0 \\ 1/\rho & 0 & 0 \end{bmatrix}$$

using an upwind finite volume method on a square domain. The boundary conditions and initial conditions will consist of :

- Reflecting wall along the right
- Non reflecting wall on the left, bottom and top
- Initial condition

$$p(x, y, 0) = exp[-ln(2)(\frac{x^2 + y^2}{0.06^2})$$
$$u(x, y, 0) = v(x, y, 0) = 0$$

2 Description of the Mathematics

Finite volume methods(FVM) are used to solve conservation law written in the conserved form :

$$\frac{d}{dt}\int_V \vec{u} dV = -\oint_S (\vec{f}, \vec{g}).\hat{n} dS$$

Where $\vec{u}$ is the conserved quantity and $\vec{f}$ is the flux in the x direction and $\vec{g}$ is the flux in the y direction.

For a quadrilateral the FVM is

$$V^k\dot{U^k} + \sum_{i=1}^{4}(\int_{s^i}(\vec{f}dS_x^i + \vec{g}dS_y^i) = 0$$

Where $U_k = \frac{1}{V^k}\int_{V^k}\vec{u}dV^k$

The integrals are approximated using quadratures.

$$d\vec{S^i} = \triangle Y_{i,i+1}\hat{X} - \triangle X_{i,i+1}\hat{Y}$$

where i is cyclic($i = 4 => i + 1 = 1$).

$$V^k = 0.5(\triangle X_{24}\triangle Y_{31} - \triangle X_{31}\triangle Y_{24})$$

where the sides are numbered counter clockwise.

The flux is calculated using an upwind Riemann solver for linear problems : If $f = B\vec{u}$ and $g = C\vec{u}$. Define $A = \alpha B + \beta C$, $\alpha = N_x, \beta = N_y$ and $\alpha^2 + \beta^2 = 1$. Where (N_x, N_y) defines the outward pointing normal. Then

$$F^*(\vec{U^L}, \vec{U^R}) = A^+\vec{U^L} + A^-\vec{U^R}$$

Where $A^\pm = \frac{A\pm|A|}{2}$.

On a rectangular grid with a constant $\triangle x$ spacing it can be shown that the error is order $\triangle x$. This analysis is similar to the results shown in class and homework worked out in the first semester and will not be repeated here.

A simple forward time solver was used to time step the solution.

2.1 Boundary Conditions

To impose a wall boundary condition($\vec{u}.\hat{N}$=0), we start with the characteristic variables:

$$S^{-1}\vec{u} = \begin{bmatrix} 1 & \alpha c & \beta c \\ 1 & -\alpha c & -\beta c \\ 0 & \beta & -\alpha \end{bmatrix} \begin{bmatrix} P \\ u \\ v \end{bmatrix} = \begin{bmatrix} w^+ \\ w^- \\ w^o \end{bmatrix}$$

Where the matrices have been chosen for the current problem at hand. Now applying $\vec{u}.\hat{N}$=0 gives $(w^+)^(int) = (w^-)^{ext}$. We also match up the tangential velocity between the internal and external states $(w^0)^{ext} = (w^0)^{int}$. Also set $P^{ext} = P^{int}$.

This gives

$$\vec{u_{wall}^{ext}} = \begin{bmatrix} P^{int} \\ [(\beta^2 - \alpha^2)u^{int} - 2\alpha\beta v^{int}] \\ -[(\beta^2 - \alpha^2)u^{int} + 2\alpha\beta v^{int}] \end{bmatrix}$$

One can apply the same ideas to an open boundary. In a simple fashion the treatment is to set $w^- = 0$.

2.2 Analytical Solution

The analytical solution to the IBVP is

$$p(x,y,t) = \int_0^\infty \frac{e^{-\omega^2/4b}}{2b}\omega J_o(r\omega)cos(\omega t)d\omega$$

where J_o is the Bessel function of the first kind of order zero, and $b = ln2/w^2$ with $w = 0.06$ until the reflection. To take care of the reflection the problem can be treated using a method of images.

I chose the domain to be $[0,1]X[0,1]$ and the initial Gaussian was centered at $x = 0.5, y = 0.5$. This doesn't change any of the results as it is simple coordinate transform in a coordinate invariant problem. So for this problem a "pseudo" gaussian needs to be placed at $x = 1.5, y = 0.5$ to properly represent a reflective right boundary.

$$p(x,y,t) = \int_0^\infty \frac{e^{-\omega^2/4b}}{2b}\omega J_o(r_1\omega)cos(\omega t)d\omega + \int_0^\infty \frac{e^{-\omega^2/4b}}{2b}\omega J_o(r_2\omega)cos(\omega t)d\omega$$

where $r_1 = \sqrt{(x-0.5)^2 + (y-0.5)^2}$ and $r_2 = \sqrt{(x-1.5)^2 + (y-0.5)^2}$. The same coordinate transform is also applied to the initial condition to get the initial guassian to be centered at $x = 0.5, y = 0.5$.

3 Description of the Algorithm

3.1 Grid

A simple grid generation routine was used to generate a grid that was returned with its nodes and connections. The connectivity was always arranged in a counter clockwise fashion. Two files were created, one with the nodes and the other with the connectivity.

3.2 Classes

3.2.1 Mesh class

Uses node class and element class. Has attributes nodes and elements. Has methods construct, destruct and read mesh. Construct reads in number of elements and number of nodes and allocates memory. Destructor deallocates the memory. Read mesh reads in node file name and connectivity file name. Read mesh will assign the nodes to the node attribute and the node ids to each element from the connectivity file.

3.2.2 Element Class

Has attributes nodeids, neighbour element ides, boundary condition flags, normal vectors, length of each side and volume.

3.2.3 Node Class

Has attributes with position of the node.

3.2.4 Edge Class

Has attributes node ids(that define the edge), element ids(on either side of the edge), element sides(which side of the elements is this edge on) and edgeid(to keep a count of number of edges). Has a procedure to construct edge.

Construct edge assigns the two node ids, the edge id and the element id and element side id for the first element to access this edge.

3.2.5 Linked list record class

Uses edge class. Has attributes of edge data(type:edge) and a pointer to the next record. Has a procedure to construct an edge record in the linked list. The constructor allocates memory for the next edge record and then calls the edge constructor to record data in the allocated memory for this record.

3.2.6 Linked list class

Uses linked list record class. Has attributes of two pointer pointing to head and tail elements of linked list. Has procedures construct, destruct, add, modify. Construct allocates the head and tail pointers and nullifies them. Destruct deallocates the head and tail. Add allocates memory for a new record and then calls the constructor for the newly allocated linked list record. It then assigns the the head, tail and next(part of the record) pointers appropriately as they should be in a linked list. Modify is used to modify an element of a linked list record when the second(not assigned so far) side of the edge is accessed. This procedure cycles through the linked list elements looking the record of the correct edge and then adds the second element id and element side to that record.

3.3 Main FVM code

Uses mesh class. Initialize all the variables including a mesh object. Call the read mesh routine with the appropriate file names. Then call the edge finding routine and the routine to calculate the volume. Then define the time step and total number of time steps required based on the choice of CFL number to be 0.5 and time of integration to be 1second. The allocate an array to store the solution in each element, for each variable and two time steps(past and current). Initialize the problem with appropriate boundary conditions, in this case gaussian like function at $x = 0.5, y = 0.5$.

for k=1 to Number of time steps **do**
  for i =1 to number of elements **do**
    sum of flux = 0
    for j=1 to 4(number of sides **do**
      read or assign $A, |A|, N_x, N_y$ for the current side
      assign the external state(either the neighbor or based on boundary condition)
      call the Riemann solver to get flux
      sum of flux = sum of flux+ flux
    end for
    $Q(i, :, 2) = Q(i, :, 1) - dt * (sumofflux)/(elementvolume)$

end for
Store at appropriate time steps
$Q(:,:,1) = Q(:,:,2)$
end for

3.4 Find edges

This subroutine gets the mesh variable from the main program. Then it allocates a matrix exists of size no. of nodesXno. of nodes and sets it to 0. Then make a node map. with node ids for 4 edges in counter clockwise order. set edgeid to 0.

for k=1 to number of elements **do**
for s=1 to 4(number of edges per element) **do**
$nodeid1 = mesh\_var\%elements(k)\%nodeids(nodemap(1,s))$
$nodeid2 = mesh\_var\%elements(k)\%nodeids(nodemap(2,s))$
i = max(nodeid1, nodeid2)
j = min(nodeid1, nodeid2)
if exists(i,j) ==0 **then**
edgeid = edgeid+1
add edge to the linked list
exists(i,j) = edgeid
else
eid = exists(i,j)
modify edge record in link list with edgeid== eid
end if
end for
end for
for k=1 to edgeid **do**
define neighbour element ids by indirection
fill flags for the appropriate boundary conditions
define the normals to each edge(2 normals for each element)
end for

3.5 Calculate volume

Calculate the volume as per the formula in the description of mathematics section.

3.6 Assign external boundary condition

Assign external states based on the results presented in the description of mathematics section.

4 Results

The system was run on three grids with 32X32, 64X64 and 128X128 mesh resolution on a square grid with a CFL number of 0.5.

First I present a comparison of the visual results between the finest(128X128) grid and the analytical solution in Figure 1. The simulation was run for 1s and the solution is shown at time of 0.2, 0.5 and 0.9s. At a first glance the numerical and analytical solution look very similar with some small differences. The first obvious difference is that the numerical solution has a smaller amplitude than the analytical solution and is more diffused. This would be expected as the numerical solution is diffusive(numerical diffusion due to error terms) while the original PDE has no diffusion terms in it.

The other difference is observed at the open boundaries(top, bottom and left). It can be seen in panels (e) and (f) that there are some small reflections at the open boundaries. At the open boundaries the boundary condition was chosen such that the external state was zero. This is not a perfect open boundary condition. As the wave approaches the boundary it sees a sharp gradient in the field variables. This can produce artificial reflections, as are being seen quite clearly in panel (e) and (f) of figure 1.However the amplitude of these reflections is small compared to the amplitude of the solution we are interested in, this warrants this boundary conditions choice as not being extremely bad.

Figure 2 shows the comparison of solutions at the line Y=0.5. For the numerical solution if the solutions at exactly Y=0.5 were not calculated then the closest grid point was chosen. The analytical solution was calculated at the Y that was chosen for the mid resolution run. Linear interpolation between the values at line just below and above Y=0.5 were considered but not performed as this problem is symmetric in Y and those two grid points will have the same values. A few point spline interpolation can be done but was not done here. The results are not greatly affected by this choice. It can be seen that the error is order of $\triangle x$ as expected. The order of the scheme would be less than order $\triangle x$ if a non constant grid had been chosen.

Looking at Figure 2 one can also see that the finest grid is the least dissipative and the coarsest grid has the highest dissipation. One can also see that the propagation speeds for the numerical solution are faster than the analytical and faster for coarser meshes. We present these results without any further theoretical investigation as deriving dispersion and dissipation rates for a FVM, for a system of equations in 2D is non trivial.

5 Conclusions

A finite volume solver was implemented for a quadrilateral grid. In this assignment a square grid was chosen however the code is versatile enough to work with unstructured meshes. The solver was then used to solve a simple initial boundary value problem with both open and reflective boundaries. The behavior of the open boundaries was found to be satisfactory but not perfect due to small amplitude reflections. For the square grid as chosen in this case the order of error was $\triangle x$ as expected. Dissipation and dispersion was also observed and noted but not analyzed.

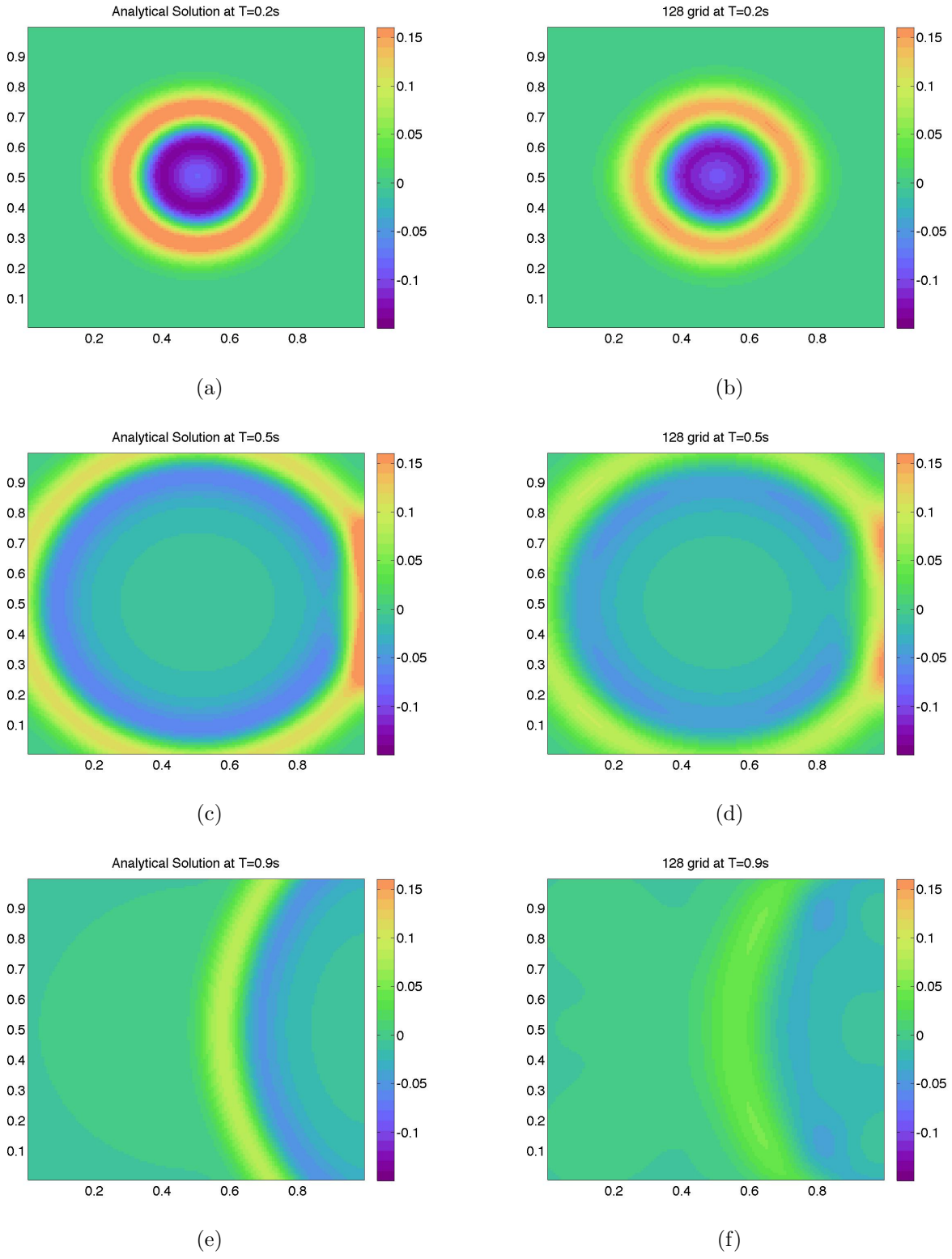

Figure 1: Analytical solutions(left column) and numerical solutions on a 128X128 mesh(right column) at different times

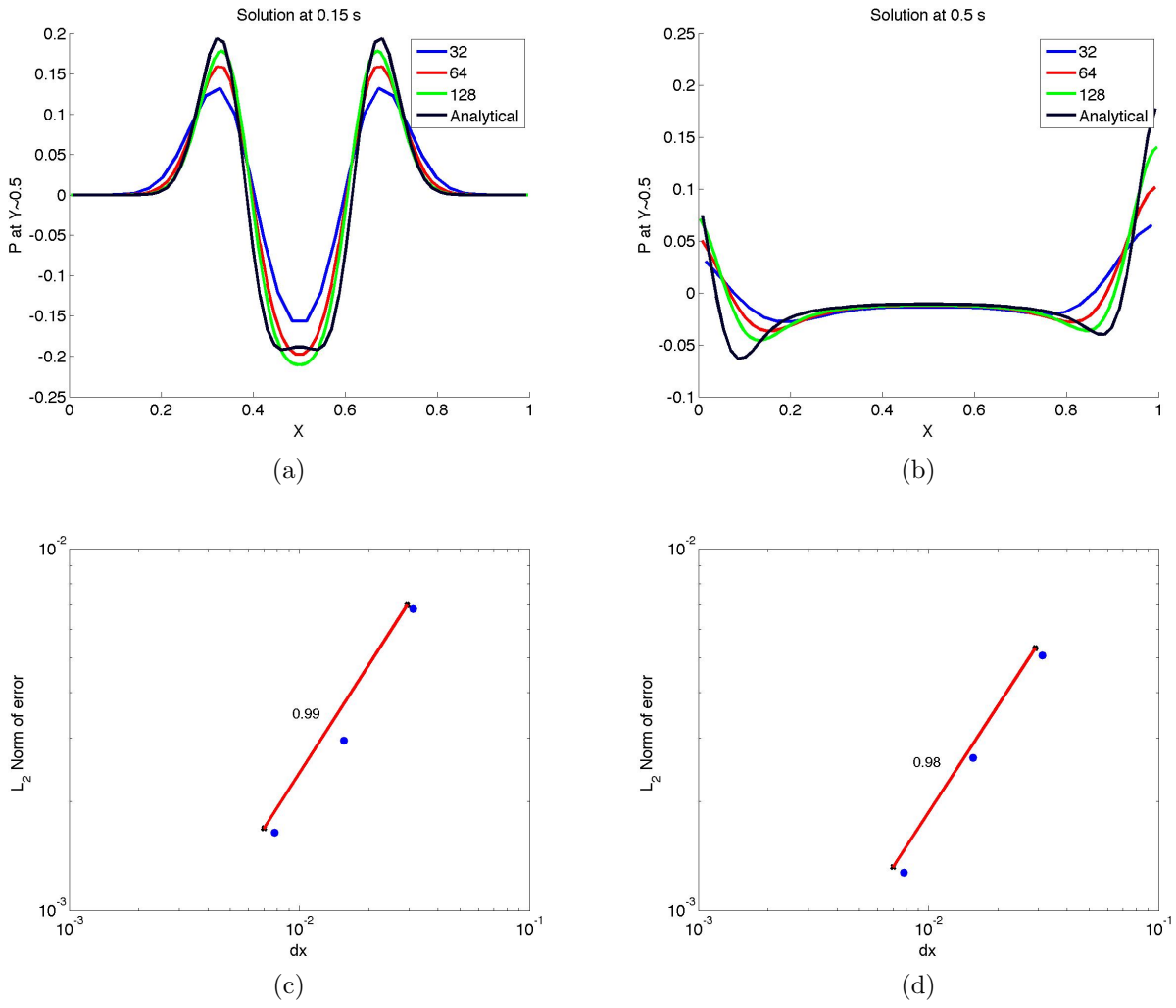

Figure 2: (a) and (b) Solutions at Y=0.5 for the grids on the 32X32, 64X64 and 128X128 mesh compared with the analytical solution at time 0.15s and 0.5s. (c) and (d) show the L_2 norm of error at the times corresponding to the times of panel (a) and (b)