NPDE 1 Homework 3

David Mandel

4 Oct 2013

1 Statement of the Problem

We solve the first order hyperbolic BVP

$$\begin{cases} \begin{pmatrix} u \\ v \end{pmatrix}_t + \begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix}_x = 0, \quad x \in [0,1]; \\ \\ u(0,t) = 0, \quad \text{radiation at } x = 1; \\ \\ \begin{pmatrix} u(x,0) \\ v(x,0) \end{pmatrix} = \begin{pmatrix} \mathrm{e}^{-100(x-1/2)^2} \\ 0 \end{pmatrix} \end{cases} \tag{1}$$

using a three-stage Runge-Kutta time integration and a centered space approximation. We compare the numerical solution with the analytical solution and show that the behavior of the numerical solution is consistent with that expected of it by the analysis.

2 Description of the Mathematics

Using the method of characteristics for the BVP (1), from Homework 1 we know the analytical solution depends on the region of the xt-plane that we want to evaluate it. However, since the w^+ and w^- waves are independent of each other, the BVP can be converted to the Cauchy problem using the method of images as follows, and this makes the analytical solution simpler to encode.

We know that all w^+ waves emanating from regions 1 and 2 in figure 1 are determined by their initial condition, $w_0^+(x-3t)$, and that the w^+ waves in regions 3 and 4 are reflections of w^- waves at the left boundary. So we seek an initial condition for all w^+ waves such that the ones in regions 3 and 4 have an initial condition that is equivalent to a reflection of a w^- wave.

Since $w^+(x,t) = -w_0^-(t - \frac{x}{3})$ in region 3 and 4, then at time $t = 0$, $w^+(x,0) = -w_0^-(-\frac{x}{3})$. Now, because of the initial condition of our problem, namely a narrow Gaussian function, the initial wave w_0^+ may be written as

$$\begin{aligned} w_0^+ &= f(x) + g(x), \\ \text{where } f(x) &= u_0(x) + v_0(x) = \mathrm{e}^{-100(x-1/2)^2} \\ \text{and } g(x) &= -w_0^-(-x/3) = -(u_0(-x/3) - v_0(-x/3)) = -\mathrm{e}^{-100((-x/3)-1/2)^2}. \end{aligned}$$

Note that this is acceptable because of the negligible contribution of either function to the other; see figure 1. This gives us

$$\begin{aligned} w^+(x,t) &= w_0^+(x-3t) = \mathrm{e}^{-100(x-3t-1/2)^2} - \mathrm{e}^{-100((-(x-3t)/3)-1/2)^2}; \\ w^-(x,t) &= w_0^-(x+t) = u_0(x+t) - v_0(x+t) = \mathrm{e}^{-100(x+t-1/2)^2} \end{aligned}$$

for all x, t, from which we may easily compute the solutions for u and v.

The boundary conditions are still needed to be specified in a convenient form for numerical computation, and for this we derived compatibility equations in Homework 1. The result was a PDE for both the left and right boundaries:

$$x = 0 : \begin{cases} v_t - (v-u)_x = 0 \\ u = 0 \end{cases}$$
$$x = 1 : \begin{cases} 2u_t + 3(v_x + u_x) = 0 \\ 2v_t + 3(v_x + u_x) = 0 \end{cases}$$

The stencils and their error for the spatial derivatives used in this approximation are

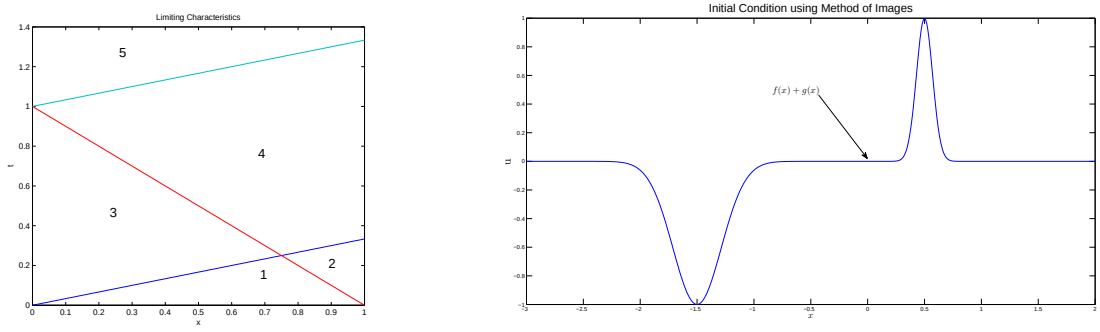
$$\delta^+ = \frac{S - I}{\Delta x} = D + \mathcal{O}(\Delta x),$$
$$\delta^0 = \frac{S - S^{-1}}{2\Delta x} = D + \mathcal{O}(\Delta x^2),$$
$$\delta^- = \frac{I - S^{-1}}{\Delta x} = D + \mathcal{O}(\Delta x),$$

where the first order stencils are used to prevent ghost points at the ends. These expected errors are verified in the Results section.

For a hyperbolic equation, a necessary condition for stability of a finite difference approximation is the Courant condition, which states that the numerical domain of dependence must at least include the physical domain of dependence of the PDE. For a given scheme, this is checked by the CFL number, defined as

$$CFL = |\lambda_{max}| \frac{\Delta t}{\Delta x} \tag{2}$$

where λ_{max} is the largest eigenvalue of the matrix in the BVP (1).



(a) Limiting characteristics

(b) Initial conditions

Figure 1: (a) Limiting characteristics for the PDE. (b) Initial condition using method of images. Note the negligible contribution of either function to the other

3 Description of the Algorithm

An object-oriented approach was taken in the implementation of the algorithm. The program is organized into an `Integrator` class, which contains the `Discretizer` class, i.e., a `Discretizer` object is instantiated whenever the `Integrator` class is instantiated. The program begins by discretizing the functions u and v at the grid points by evaluating the functions' initial condition at each point. The spatial derivatives for these initial values are then computed using three different approximations in the `xDeriv()` function:

$$\left.\frac{\partial u}{\partial x}\right|_{x_j} \approx \texttt{dudx[j]} = \begin{cases} \delta^+ u_0; \\ \delta^0 u_j, \quad j = 1, \ldots N-1; \\ \delta^- u_N, \end{cases}$$

and similarly for v. The lower order stencils are used at the left- and right-most points since they prevent ghost points, while the higher order centered stencil was used at all interior points.

The time derivatives were computed next by simply solving for u_t and v_t in the PDE and enforcing the compatibility equations at the boundaries:

$$\left.\frac{\partial u}{\partial t}\right|_{x_j} \approx \texttt{dudx[j]} = \begin{cases} 0, & j = 0; \\ -\texttt{dudx[j]} - 2\texttt{dvdt[j]}, & j = 1, \ldots, N-1; \\ -\frac{3}{2}\left(\texttt{dudx[j]} + \texttt{dvdx[j]}\right), & j = N \end{cases}$$

$$\left.\frac{\partial v}{\partial t}\right|_{x_j} \approx \texttt{dudt[j]} = \begin{cases} \texttt{dvdx[j]} - \texttt{dudx[j]}, & j = 0; \\ -2\texttt{dudx[j]} - \texttt{dvdx[j]}, & j = 1, \ldots, N-1; \\ -\frac{3}{2}\left(\texttt{dudx[j]} + \texttt{dvdx[j]}\right), & j = N. \end{cases}$$

At this point, we have the required data to integrate, which is done using a three-stage, two level storage Runge-Kutta routine. This integration routine is repeated until we reach the terminal time.

4 Results

4.1 Algorithm Component Testing

The `xDeriv()` function computes the finite difference approximation to the x derivative using the δ^+, δ^0 and δ^- operators at the left endpoint, interior points and right endpoint, respectively. To test this function, the finite difference approximations for a linear, quadratic, and exponential function were computed and the maximum error tabulated. Since the endpoint operators were derived from a linear interpolant, we expect exactness for a linear function, and likewise for a quadratic function for the interior points. The endpoint operators are expected to have error $\mathcal{O}(\Delta x)$ while the centered operator in the interior is expected to have error $\mathcal{O}(\Delta x^2)$. Table 1 verifies these expectations.

function:	x		x^2		$\mathrm{e}^{-100(x-1/2)^2}$	
N	ends	interior	ends	interior	ends	interior
32	0	0	0.03125	0	7.34E-9	0.59034
64	0	0	0.01563	0	1.86E-9	0.15521
128	0	0	0.00781	0	6.92E-10	0.03953
256	0	0	0.00391	0	3.02E-10	0.00990

Table 1: Maximum Error for xDeriv(). Note the interior error decreases at a rate proportional to Δx^2 while the endpoint error decreases on the order of Δx.

The Runge-Kutta routine `rkLs()` was tested using the ODE system

$$\begin{pmatrix} u'(t) \\ v'(t) \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \begin{pmatrix} u(t) \\ v(t) \end{pmatrix}, \qquad \begin{pmatrix} u(0) \\ v(0) \end{pmatrix} = \begin{pmatrix} 0 \\ 2 \end{pmatrix}, \tag{3}$$

which has the solution

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} 2\sin t \\ 2\cos t \end{pmatrix}.$$

The interval $t \in [0, 2\pi]$ was integrated using using shrinking grid spacings dt (which correspond with larger number of points, N), which resulted in shrinking errors as demonstrated in Table 2.

N	$\|E\|_\infty$
32	3.89E-3
64	4.94E-4
128	6.17E-5
256	7.72E-6

Table 2: Maximum error for rkLs() using the equation (3).

4.2 Analysis of Numerical Solution

The algorithm was applied to the BVP (1) at spatial discretizations of $N = 32, 64, 128, 256$, where N = number of grid points. To begin, the solutions for both components u and v were computed using a CFL number of 0.5 and plotted against the exact solution. Figures 5 - 7 provide a graphical view of the u solutions, and figures 8 - 10 do the same for the v solutions. It is clear that the numerical solution better approximates the exact solution as we shrink the grid spacings, at least for CFL = 0.5. It looks like $N = 128$ provides exact visual accuracy, which is why the solutions for $N = 256$ were not plotted. The CPU times used to integrate to the final time are graphed as a function of N in figure 3, which suggests $\mathcal{O}(N^2)$ relationship.

The norm of the error, defined as

$$||e|| = \sqrt{\Delta x \sum_{j=0}^{N} |e_j|^2},$$

where

$$|e_j|^2 = (u - u_{exact})_j^2 + (v - v_{exact})_j^2$$

was computed at eight time levels for the four values of N and plotted. It looks like the norm of the error is inversely proportional to the square of the step size, i.e., $||e|| = \mathcal{O}(\frac{1}{N^2})$. An interpolant of this type was fitted to two of the points of the $t = 0.5$ data and appears to be a good fit - see figure 2.

The Courant number (2) was varied from 0.5 to 2.5 at steps of 0.25 to test the stability of the algorithm. We defined a "blow up" of the solution if $||e|| > 10$ and were interested at combination of N and CFL numbers the solution blew up. It was found that $CFL < 2$ provided no blow up and $CFL = 2$ blew up only for $N = 256$, but $CFL > 2$ allowed the solution to blow up. The time t that the solution reached this point was noted and the results are tabulated in table 3.

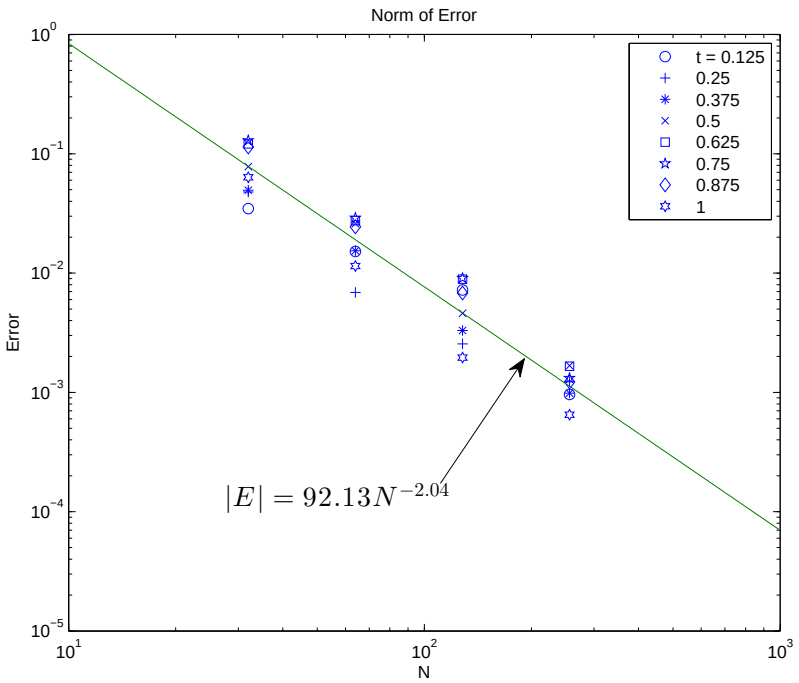


Figure 2: Norm of the error reported at various time levels as a function of N

CFL	N	time to "blowup"
2	256	0.528646
2.25	32	0.515625
	64	0.691406
	128	0.462891
	256	0.240234
2.5	32	0.338542
	64	0.429687
	128	0.30599
	256	0.159505

Table 3: Time until solution "blows up" as defined by $||e||^2 > 10$

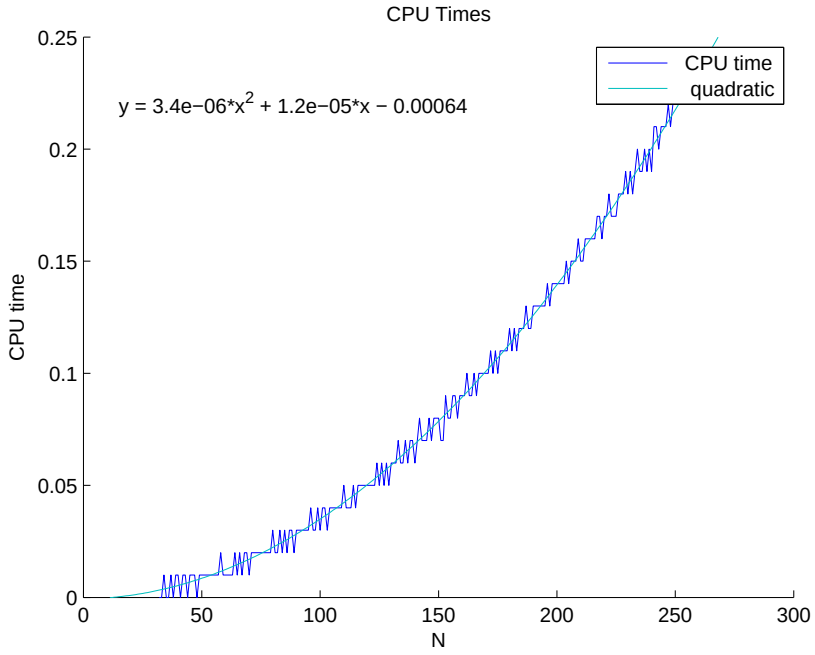


Figure 3: CPU time as a function of N with a quadratic fitted to the data.

5 Conclusions

The results from the `xDeriv()` routine were tested and were found to be consistent with the exactness and approximation order expected (see table 1). The Runge-Kutta routine `rkLs()` also performed as expected (see table 2).

For $CFL = 0.5$ and $N = 32$ a few interesting things are noticeable in the plots comparing the solutions (see figure 5). First, the amplitude of the numerical solution is smaller than the exact. Second and most notable is that the peak of the numerical wave is out of phase with the exact wave. In fact, it seems to be traveling more slowly than the exact wave. This is less noticeable for $N = 64$ and not noticeable for higher N. Finally, it's worth noting that there does not appear to be any diffusion in the numerical solution, as would be expected had we used the δ^+ operator. This is because there is no diffusion term in the expansion of the centered difference operator:

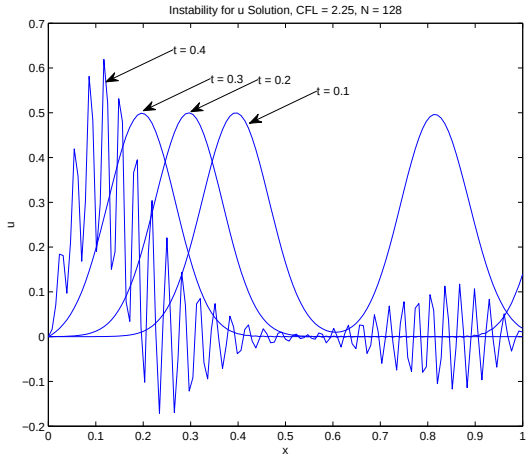
$$\delta^0 = D + \frac{\Delta x^2}{3!} D^3 + \mathcal{O}(\Delta x^4).$$

It's interesting to take a look at the behavior of the numerical solution as it is "blowing up", for example the u solution for $N = 128$ and $CFL = 2.25$. Table 3 shows the solution becoming large around $t = 0.43$, and a plot of these solutions around that time shows the instability (see figure 4). The solution appears to become unstable right at the point it hits the left boundary.

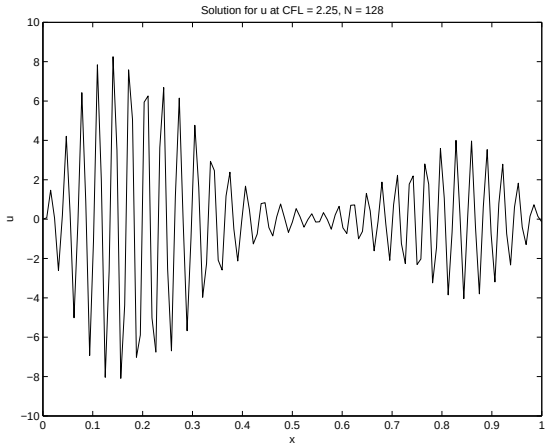
Questions worth answering: Why is the amplitude lower? Why is it traveling slower? Doesn't the CFL number have to be less than one for stability?

6 Program Listing

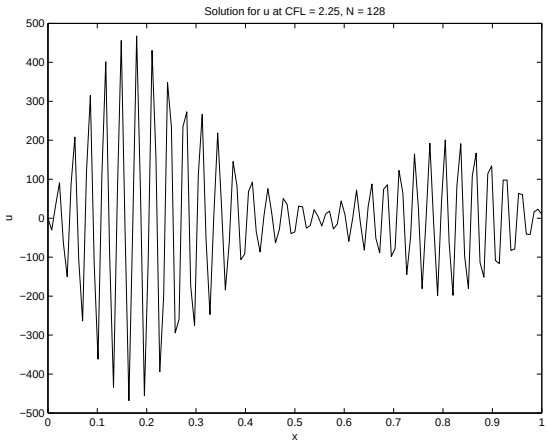
see attached.



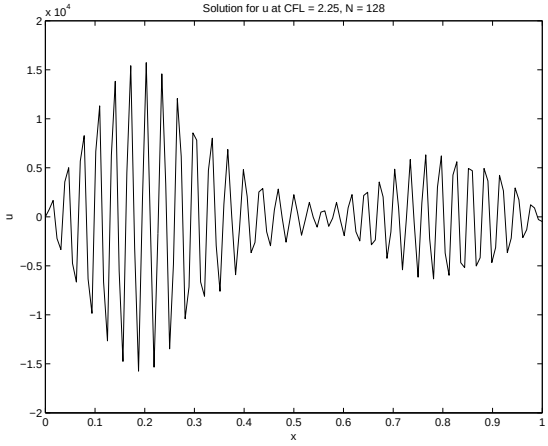
(a) initial times, t = 0.1, 0.2, 0.3, 0.4



(b) t = 0.45

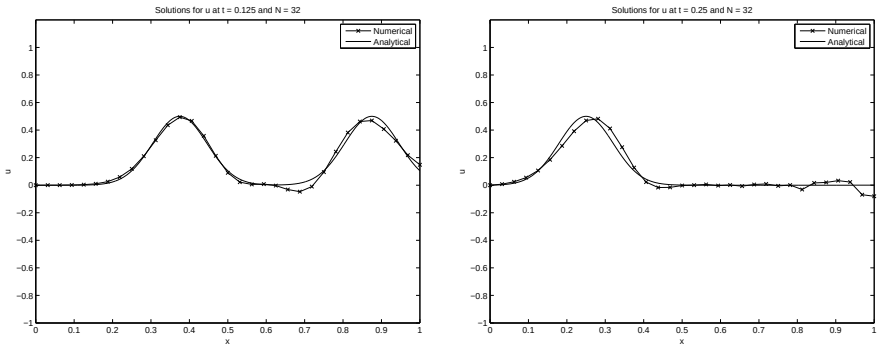


(c) t = 0.5



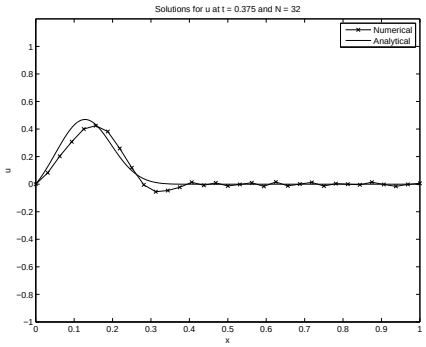
(d) t = 0.55

Figure 4: Instability for the u solution for $CFL = 2.25$, $N = 128$. Note the significant change in scales on the u - axis

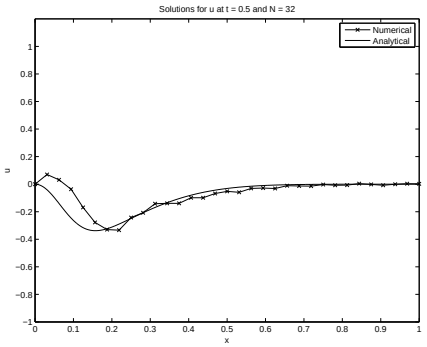


(a) t = 0.125

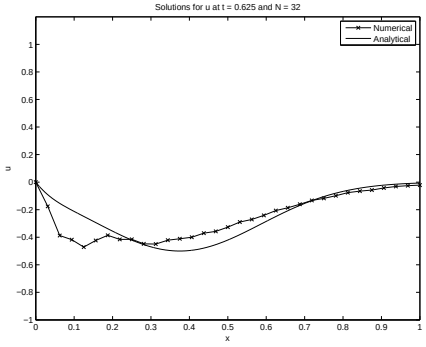
(b) t = 0.25



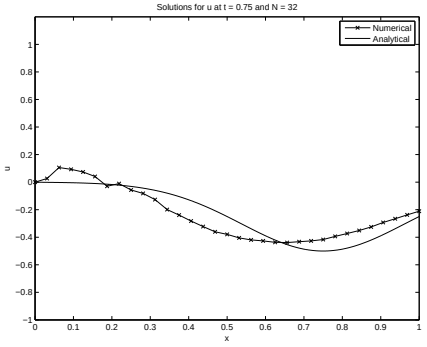
(c) t = 0.375



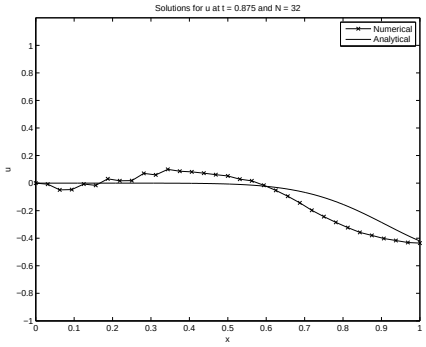
(d) t = 0.5



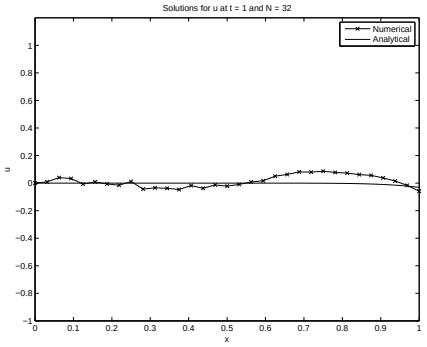
(e) t = 0.625



(f) t = 0.75

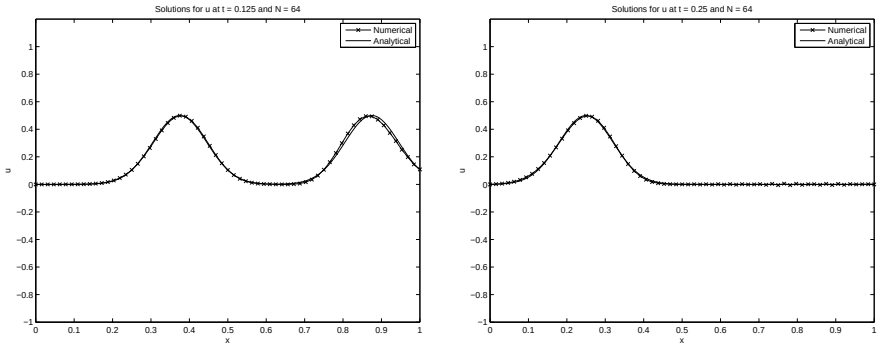


(g) t = 0.875



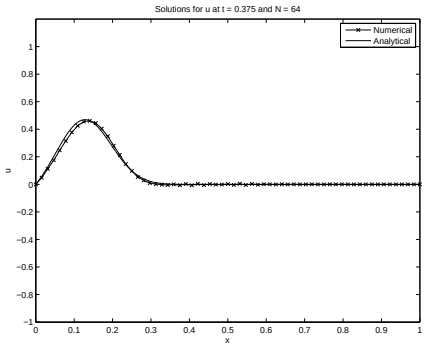
(h) t = 1

Figure 5: Solution comparison for u for $N = 32$ The initial condition was exact (at least graphically) so it is not shown.

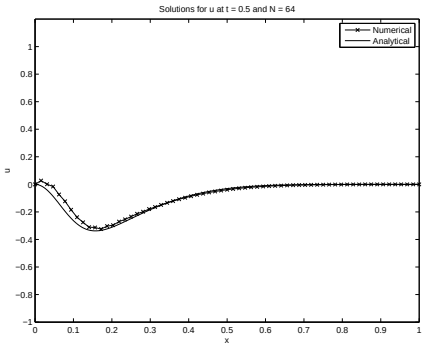


(a) t = 0.125

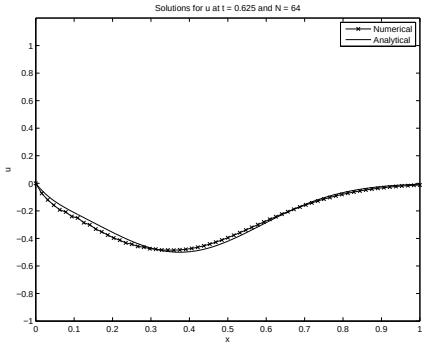
(b) t = 0.25



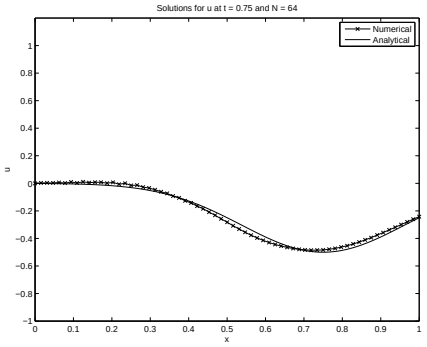
(c) t = 0.375



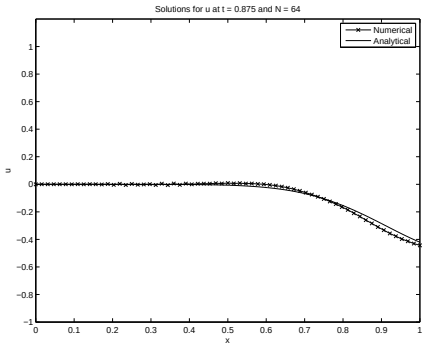
(d) t = 0.5



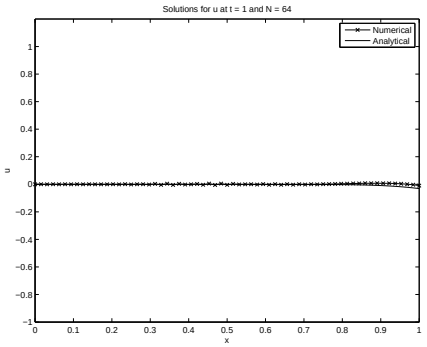
(e) t = 0.625



(f) t = 0.75

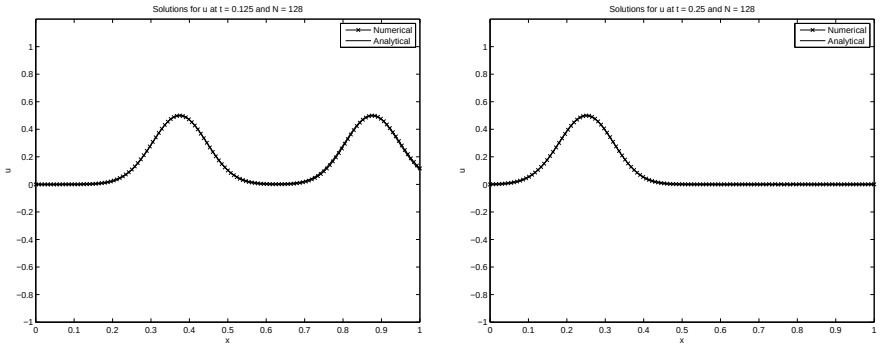


(g) t = 0.875



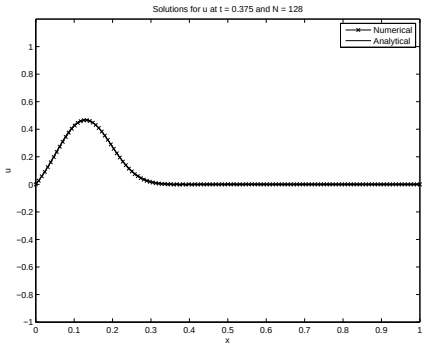
(h) t = 1

Figure 6: Solution comparison for u for $N = 64$.

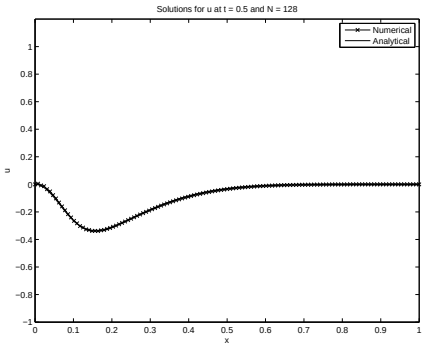


(a) t = 0.125

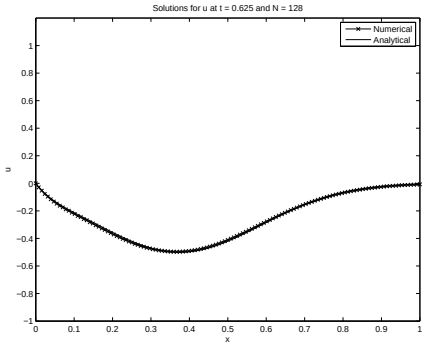
(b) t = 0.25



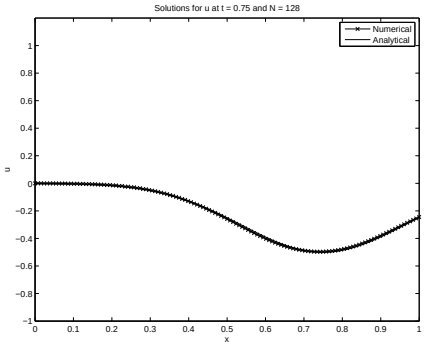
(c) t = 0.375



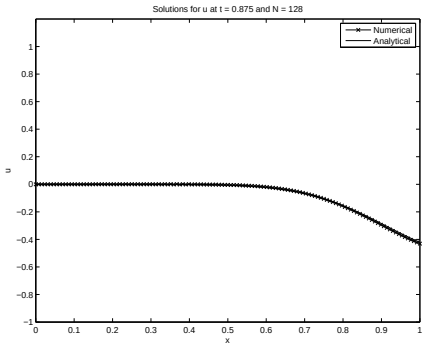
(d) t = 0.5



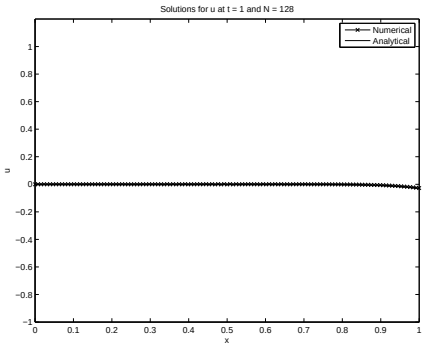
(e) t = 0.625



(f) t = 0.75

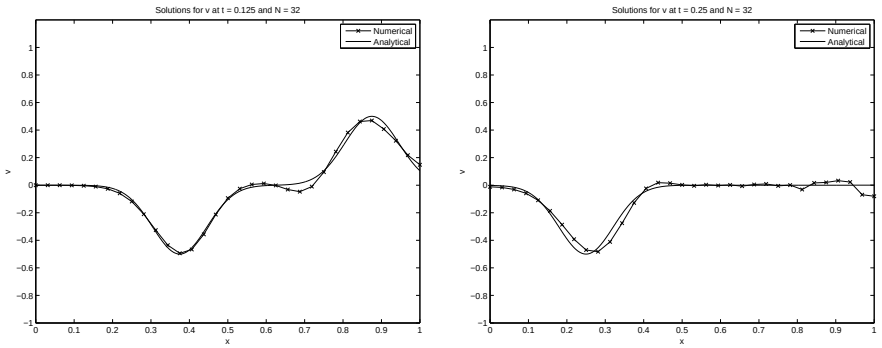


(g) t = 0.875



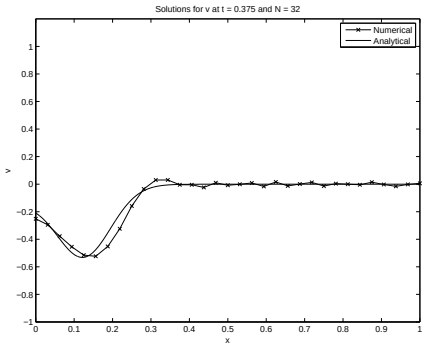
(h) t = 1

Figure 7: Solution comparison for u for $N = 128$.

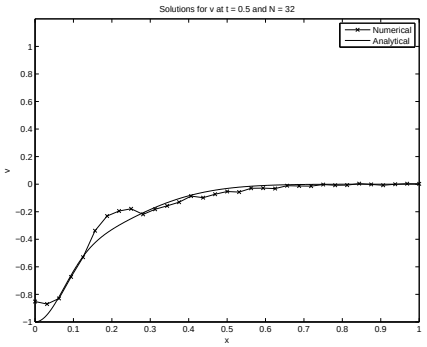


(a) t = 0.125

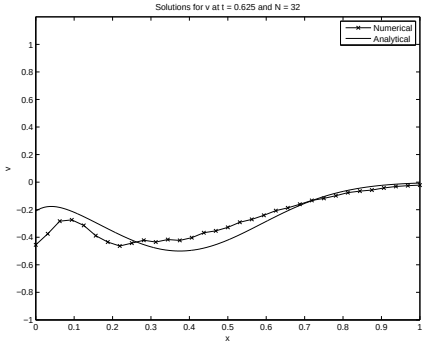
(b) t = 0.25



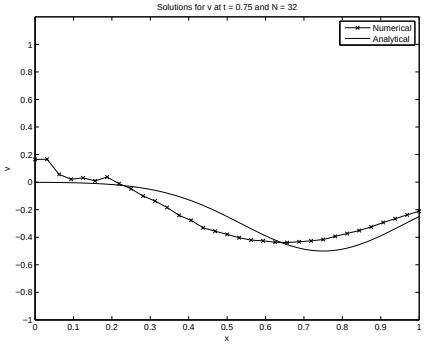
(c) t = 0.375



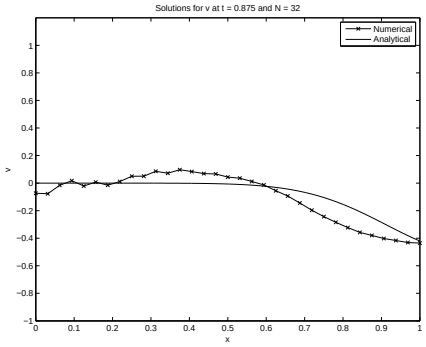
(d) t = 0.5



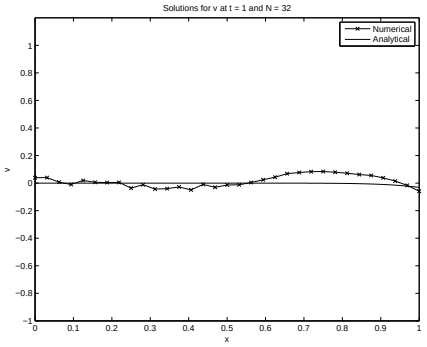
(e) t = 0.625



(f) t = 0.75

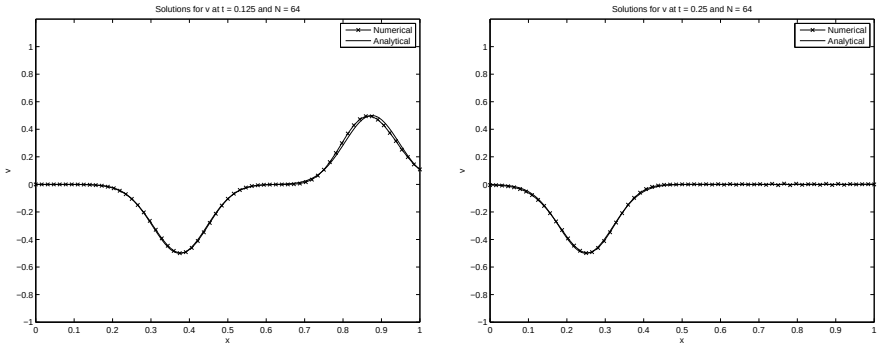


(g) t = 0.875



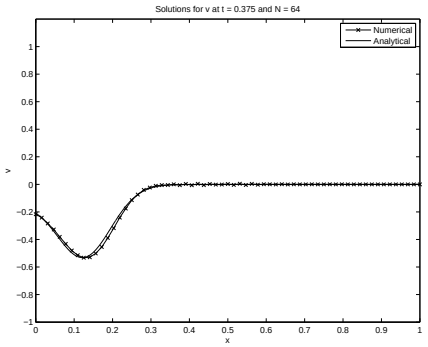
(h) t = 1

Figure 8: Solution comparison for v for $N = 32$ The initial condition was exact (at least graphically) so it is not shown.

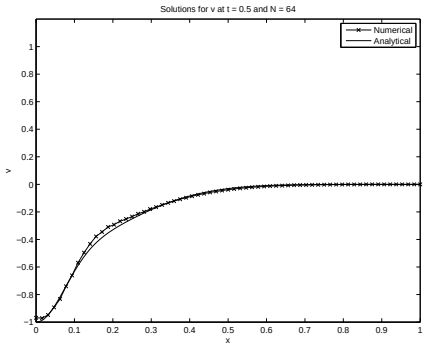


(a) t = 0.125

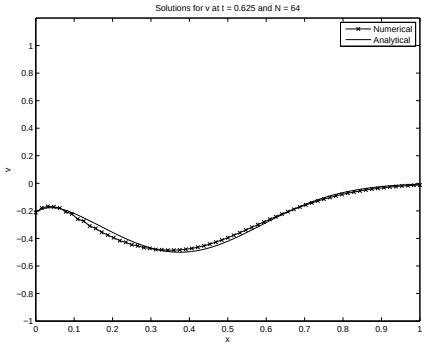
(b) t = 0.25



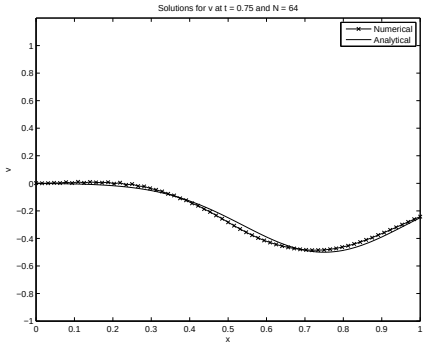
(c) t = 0.375



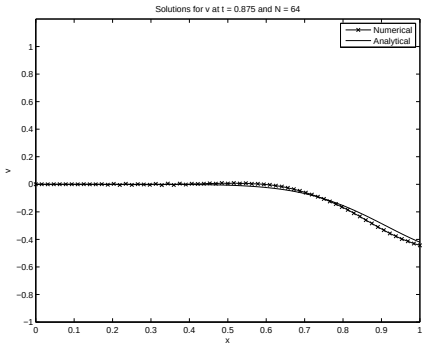
(d) t = 0.5



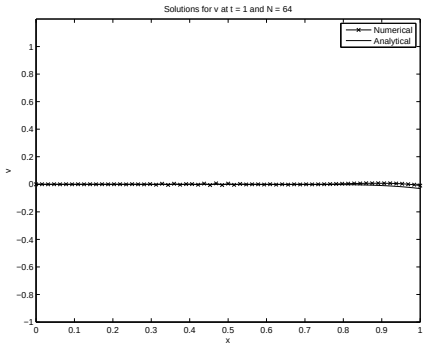
(e) t = 0.625



(f) t = 0.75

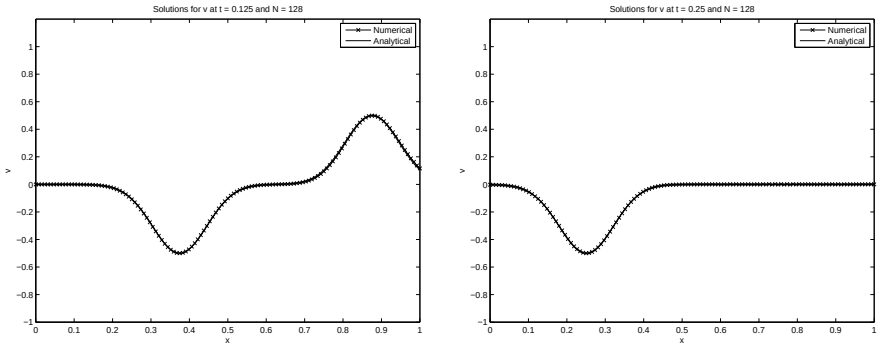


(g) t = 0.875



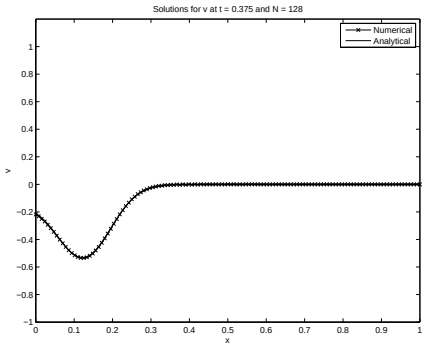
(h) t = 1

Figure 9: Solution comparison for v for $N = 64$.

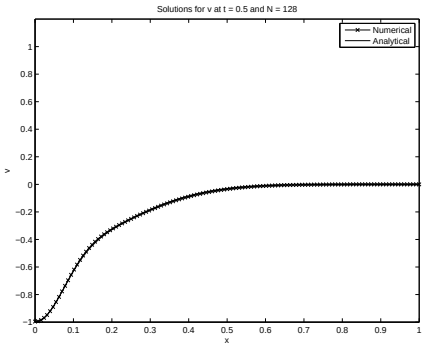


(a) t = 0.125

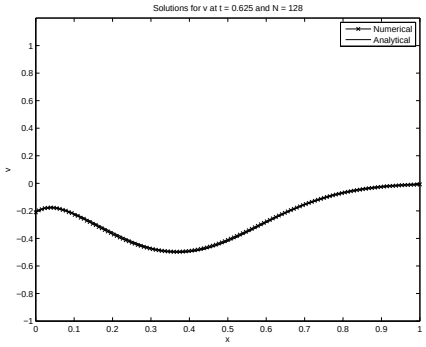
(b) t = 0.25



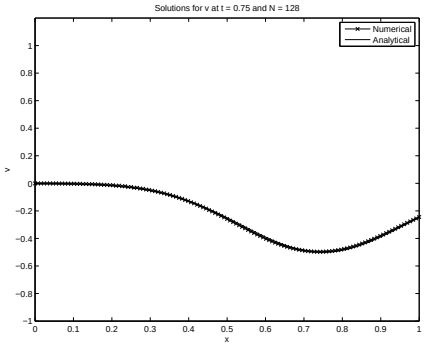
(c) t = 0.375



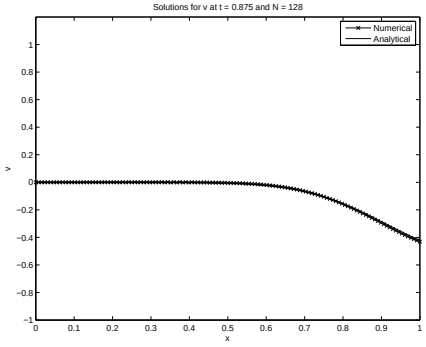
(d) t = 0.5



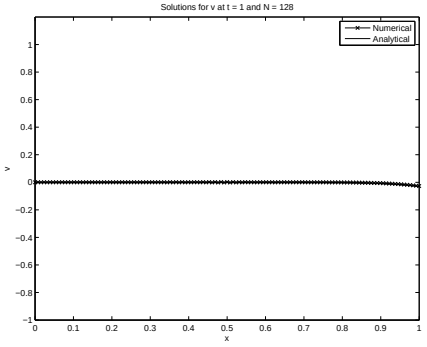
(e) t = 0.625



(f) t = 0.75



(g) t = 0.875



(h) t = 1

Figure 10: Solution comparison for v for $N = 128$.