387N Homework 2: Wave Equations

Group 2
Marcus Berg
Ethan Honda
Rob Jones

March 10, 1998

`http://wwwrel.ph.utexas.edu/~p387g2`

1 Preliminaries

We will consider the *1-d advection equation*

$$\frac{\partial u(x,t)}{\partial t} = \frac{\partial u(x,t)}{\partial x} \tag{1}$$

on the domain

$$\Omega = \{(x,t) \ \mid \ 0 \le x \le 1, \ 0 \le t \le T\} \tag{2}$$

subject to the initial condition

$$u(x,0) = u_0(x) \tag{3}$$

and to the periodic boundary conditions

$$u(0,t) = u(1,t). \tag{4}$$

We will solve (1)-(4) with a well-known fully-discretized finite difference method, *the leapfrog scheme.*

$$u_j^{n+1} = \lambda(u_{j+1}^n - u_{j-1}^n) + u_j^{n-1} \tag{5}$$

where $\lambda = \frac{\Delta t}{\Delta x}$ is the Courant parameter in this case.

We will show that this scheme is well chosen for this problem since it is stable and convergent for $\lambda \in (0,1]$, and is, in fact, *error-free* for $\lambda = 1$.

2 Problem 1a: Analytic Solution

By inspection, we see that any $g(x+t)$ will satisfy (1). Equation (3) demands that $u(x,0) = g(x) = u_0(x)$, so our solution, in general, must be

$$u(x,t) = u_0(x+t) \tag{6}$$

providing that the function u_0 satisfies the periodicity condition given by (4), *i.e.*

$$u_0(0) = u_0(1). \tag{7}$$

2.1 Alternate Solution Form

As an aside, we also note that a general solution to (1)-(4) may be written as $u(x,t) = u_0(x+t)$ where

$$u_0(x) = x(x-1)f(x) + C \tag{8}$$

and where $f(x)$ is any function defined on $[0, T+1]$, and C is a real constant. Thus,

$$u(x,t) = (x+t)(x+t-1)f(x+t) + C \tag{9}$$

satisfies (1)-(4) for any function f.

2.2 Description of Solutions

The exact solution (6) to the *1-d advection equation* is that of an evolution equation which "transports" its initial data to the left with unit speed.

This is not surprising, given that the form of equation (1) itself, essentially a "scaled-down" hyperbolic PDE, suggests that its solutions might be similar to those of the *wave equation.*

As an aside we note that in fact, the *wave equation* can be written as a pair of coupled *advection equations. i.e.* consider the following:

$$\begin{aligned}\frac{\partial u}{\partial t} &= \frac{\partial v}{\partial x}\\ \frac{\partial v}{\partial t} &= \frac{\partial u}{\partial x}\end{aligned}$$

Then we may write

$$\frac{\partial^2 u}{\partial t^2} = \frac{\partial}{\partial t}\left(\frac{\partial u}{\partial t}\right) = \frac{\partial}{\partial t}\left(\frac{\partial v}{\partial x}\right) = \frac{\partial}{\partial x}\left(\frac{\partial v}{\partial t}\right) = \frac{\partial}{\partial x}\left(\frac{\partial u}{\partial t}\right) = \frac{\partial^2 u}{\partial x^2} \tag{10}$$

3 Problem 1b: Difference Operators

Let us define the finite difference operators Δ_t^h and Δ_x^h on the discrete domain $\Omega^h = \{(x_j, t^n) \mid 0 \le x_j \le 1,\ 0 \le t^n \le T\}$

$$\begin{aligned}\Delta_t^h f(x,t) &\equiv \frac{f(x,t+\Delta t) - f(x,t-\Delta t)}{2\Delta t}\\ &\equiv \frac{f(x,t+\lambda h) - f(x,t-\lambda h)}{2\lambda h} \qquad (11)\\ \Delta_x^h f(x,t) &\equiv \frac{f(x+\Delta x,t) - f(x-\Delta x,t)}{2\Delta x}\\ &\equiv \frac{f(x+h,t) - f(x-h,t)}{2h} \qquad (12)\end{aligned}$$

With these definitions, we can write a finite-difference solution to (1) as

$$(\Delta_t^h - \Delta_x^h)u^h = 0 \tag{13}$$

Now, let us expand $f(x,t)$ in a Taylor series about $t + \lambda h$ and $t - \lambda h$.

$$\begin{aligned}f(x,t+\lambda h) &= f(x,t) + \lambda h\frac{\partial f}{\partial t} + \frac{(\lambda h)^2}{2}\frac{\partial^2 f}{\partial t^2} + \frac{(\lambda h)^3}{6}\frac{\partial^3 f}{\partial t^3} + \\ &\quad \frac{(\lambda h)^4}{24}\frac{\partial^4 f}{\partial t^4} + O(h^5)\\ f(x,t-\lambda h) &= f(x,t) - \lambda h\frac{\partial f}{\partial t} + \frac{(\lambda h)^2}{2}\frac{\partial^2 f}{\partial t^2} - \frac{(\lambda h)^3}{6}\frac{\partial^3 f}{\partial t^3} + \\ &\quad \frac{(\lambda h)^4}{24}\frac{\partial^4 f}{\partial t^4} + O(h^5)\end{aligned}$$

Combining these terms, we get

$$\begin{aligned}f(x,t+\lambda h) - f(x,t-\lambda h) &= 2\lambda h\frac{\partial f}{\partial t} + 2\frac{(\lambda h)^3}{6}\frac{\partial^3 f}{\partial t^3} + O(h^5)\\ \frac{f(x,t+\lambda h) + f(x,t-\lambda h)}{2\lambda h} &= \frac{\partial f}{\partial t} + \frac{(\lambda h)^2}{6}\frac{\partial^3 f}{\partial t^3} + O(h^4) \qquad (14)\end{aligned}$$

And comparing this with (11), we arrive at

$$\Delta_t^h = \partial_t + \frac{(\lambda h)^2}{6}\partial_{ttt} + O(h^4) \tag{15}$$

We will proceed similarly for Δ_x^t

$$\begin{aligned}f(x+h,t) &= f(x,t) + h\frac{\partial f}{\partial x} + \frac{h^2}{2}\frac{\partial^2 f}{\partial x^2} + \frac{h^3}{6}\frac{\partial^3 f}{\partial x^3} + \frac{h^4}{24}\frac{\partial^4 f}{\partial x^4} + O(h^5)\\ f(x-h,t) &= f(x,t) - h\frac{\partial f}{\partial x} + \frac{h^2}{2}\frac{\partial^2 f}{\partial x^2} - \frac{h^3}{6}\frac{\partial^3 f}{\partial x^3} + \frac{h^4}{24}\frac{\partial^4 f}{\partial x^4} + O(h^5)\end{aligned}$$

Thus, we have

$$\frac{f(x+h,t)-f(x-h,t)}{2h} = \frac{\partial f}{\partial x} + \frac{h^2}{6}\frac{\partial^3 f}{\partial x^3} + O(h^4) \tag{16}$$

And comparing this with (12), we see

$$\Delta_x^h = \partial_x + \frac{h^2}{6}\partial_{xxx} + O(h^4) \tag{17}$$

We may then write, by substitution into (13),

$$\Big[\partial_t + \frac{(\lambda h)^2}{6}\partial_{ttt} - \partial_x - \frac{h^2}{6}\partial_{xxx}\Big]u(x,t) + O(h^4) = 0 \tag{18}$$

If we now define the *local truncation error*, τ^h, via

$$\tau^h \equiv (\Delta_t^h - \Delta_x^h)u$$

we find that

$$\tau^h = (\partial_t - \partial_x)u(x,t) + \Big[\frac{(\lambda h)^2}{6}\partial_{ttt} - \frac{h^2}{6}\partial_{xxx}\Big]u(x,t) + O(h^4) \tag{19}$$

But the *advection equation* (1) tells us that the first-derivative terms are identically zero, thus we have

$$\begin{aligned} \tau^h &= \Big[\frac{(\lambda h)^2}{6}\partial_{ttt} - \frac{h^2}{6}\partial_{xxx}\Big]u(x,t) + O(h^4) \\ \tau^h &= O(h^2) \end{aligned} \tag{20}$$

Or, put more simply, the *leapfrog* scheme applied to the *advection equation* as given in (13) is of second-order.

4 Problem 1c: Richardson Expansion

Let's assume that the solution, u^h, of the difference equation (13) admits an asymptotic expansion of the form

$$u^h(x,t) = u(x,t) + h^2 e_2(x,t) + h^4 e_4(x,t) + O(h^6) \tag{21}$$

where the odd order terms in h have been omitted due to the symmetry of the finite difference operators (11) and (12). *i.e.* for the same reason that the odd order terms in h vanished from (14) and (16).

Substituting (15), (17), and (21) into (13) we find

$$\Big(\partial_t + \frac{(\lambda h)^2}{6}\partial_{ttt} - \partial_x + \frac{h^2}{6}\partial_{xxx}\Big)\Big(u + h^2 e_2 + h^4 e_4\Big) = 0 \tag{22}$$

which can be expanded fully to yield

$$\begin{aligned} &\frac{\partial u}{\partial t} + \frac{\lambda^2 h^2}{6}\frac{\partial^3 u}{\partial t^3} - \frac{\partial u}{\partial x} - \frac{h^2}{6}\frac{\partial^3 u}{\partial x^3} + h^2\frac{\partial e_2}{\partial t} + \frac{\lambda^2 h^4}{6}\frac{\partial^3 e_2}{\partial t^3} - h^2\frac{\partial e_2}{\partial x} - \\ &\qquad \frac{h^4}{6}\frac{\partial^3 e_2}{\partial x^3} + h^4\frac{\partial e_4}{\partial t} + \frac{\lambda^2 h^6}{6}\frac{\partial^3 e_4}{\partial t^3} - h^4\frac{\partial e_4}{\partial x} - \frac{h^6}{6}\frac{\partial^3 e_4}{\partial x^3} = 0 \end{aligned} \tag{23}$$

We now demand that all the terms in (23) vanish *order-by-order* so we may collect the terms of each order seperately.

The $O(1)$ terms yield the original *advection equation* (1).

$$\frac{\partial u}{\partial t} - \frac{\partial u}{\partial x} = 0 \tag{24}$$

The $O(h^2)$ terms then give us a partial differential equation for the leading-order error function e_2

$$\frac{\partial e_2}{\partial t} - \frac{\partial e_2}{\partial x} + \frac{\lambda^2}{6}\frac{\partial^3 u}{\partial t^3} - \frac{1}{6}\frac{\partial^3 u}{\partial x^3} = 0 \tag{25}$$

where the function $u(x,t) = u_0(x+t)$ is the same function defined by (6).

We can then, without loss of generality, assume that the solution to (25) is of the form

$$e_2(x,t) = At^\alpha f(x+t) + g(x+t) \tag{26}$$

where $f(x)$ and $g(x)$ are arbitrary functions and A and α are arbitrary (nonzero) real numbers. Then

$$\frac{\partial e_2}{\partial t} = A\alpha t^{\alpha-1} f(x+t) + At^\alpha f'(x+t) + g'(x+t) \tag{27}$$

$$\frac{\partial e_2}{\partial x} = At^\alpha f'(x+t) + g'(x+t) \tag{28}$$

Since we also demand that the initial condition (3) be an exact solution of (1), we know that $e_2(x,0) = 0$. Therefore, the function g in our *ansatz* for e_2 must be identically zero (since it must be zero for all x when $t = 0$). Then, substituting into (25), we have

$$\begin{aligned}\frac{\partial e_2}{\partial t} - \frac{\partial e_2}{\partial x} &= A\alpha t^{\alpha-1} f(x+t) \\ &= \frac{1-\lambda^2}{6} {u_0}'''(x+t)\end{aligned} \tag{29}$$

which can only be satisfied if

$$\begin{aligned} A &= \frac{1-\lambda^2}{6} \\ \alpha &= 1 \\ f(x+t) &= {u_0}'''(x+t) \end{aligned} \tag{30}$$

Thus the general solution to the PDE (25) is

$$e_2(x,t) = \frac{1-\lambda^2}{6}\, t\, {u_0}'''(x+t) \tag{31}$$

and the leading order error in the difference solution (13) is $h^2 e_2(x,t)$.

For the given initial data

$$u_0(x) = e^{\frac{-(x-\frac{1}{2})^2}{0.05^2}} \tag{32}$$

we have

$$u(x,t) = e^{\frac{-(x+t-\frac{1}{2})^2}{0.05^2}} \tag{33}$$

and

$$e_2(x,t) = \frac{160000(\lambda^2-1)t(197 - 1194(x+t) + 2400(x+t)^2 - 1600(x+t)^3)}{3e^{\frac{(x+t-\frac{1}{2})^2}{0.05^2}}} \tag{34}$$

5 Problem 1d: Leading Order Error Estimate

Let's assume that we have two separate solutions to the difference equation (13), u^h and u^{2h}, computed at resolutions h and $2h$ respectively. Then

$$\begin{aligned} u^h = u + h^2 e_2 + O(h^4) &\implies u = u^h - h^2 e_2 + O(h^4) \\ u^{2h} = u + 4h^2 e_2 + O(h^4) &\implies u = u^{2h} - 4h^2 e_2 + O(h^4) \end{aligned}$$

i.e. both give $O(h^4)$ approximations to the actual solution, u. Equating these approximations, we find

$$\begin{aligned} u^h - h^2 e_2 &= u^{2h} - 4h^2 e_2 \\ 3h^2 e_2 &= u^{2h} - u^h \\ e_2 &= \frac{1}{3h^2}\left(u^{2h} - u^h\right) \end{aligned} \tag{35}$$

6 Problems 1e and 1f: Computational Solution

We have written several computational solutions to the *1-d advection equation*: one C program based on the method described in the previous section, one RNPL implementation of the same *leapfrog* scheme, and a semi-discrete solution using the numerical method-of-lines in Fortran. We found that the programs agreed quite well with each other.

6.1 Fully discrete *leapfrog* solution program `advect1d`

We first solved the *advection equation* with the fully-discrete, explicit *leapfrog* integration scheme.

The `advect1d` program has the following usage:

```
advect1d <level> <olevel> <nc> <lambda>
```

where, the `level` parameter gives the grid size and spacing; the grid contains $2^{level} + 1$ points evenly spaced between 0 and 1. For our purposes, `lambda` takes on values in the 5-10 range. The `olevel` parameter controls output frequency, which occurs every $2^{level-olevel}$ timesteps. The `nc` parameter is a measure of the number of times a pulse would cross the periodic grid assuming a `lambda` ≈ 1. And `lambda` is the *Courant* number, or ratio $\frac{\Delta t}{\Delta x}$.

At each output step, we display the computed solution, the exact solution, the absolute error between these, and the leading order error term $h^2 e_2$. Internally, our program follows a simple schematic:

1. Read input values for `level`, `olevel`, `nc`, and `lambda` from the command line
2. Set initial data using the exact (analytic) solution obtained in section 2
3. Advance the solution in time using equation (13)
4. Output solution if necessary (every $2^{level-olevel}$ steps)
5. Repeat (3)-(4) until the final time is reached

For coarse grids, (`level` ≤ 6), the error is (naturally) larger than for finer grids, and the deviation of the computed solution from the exact solution grows roughly linearly in time (see figures (1) and (2).

We note that "course grid" is defined as a grid for which the initial pulse spans only a few grid points. To accurately represent the pulse, we demand that it cross at least 4 grids, and the more it crosses, the better the results will be.

6.2 RNPL `advect1d` program

We used RNPL to find a second solution to the *advection equation*. The very nature of RNPL made this a relatively simple task. We started our RNPL program with a `system parameter` statement that declared the amount of memory to be allocated to this program. We then declared the program parameters (which were all of type `float`) with a statement of the form `parameter float varname := default value`. We then defined the coordinates, the grid structure, and the grid variables where we chose rectilinear coordinates x and t (this is a 1-D problem, so on a uniform grid, there is no difference between rectilinear x and t versus polar r and t). Since we are

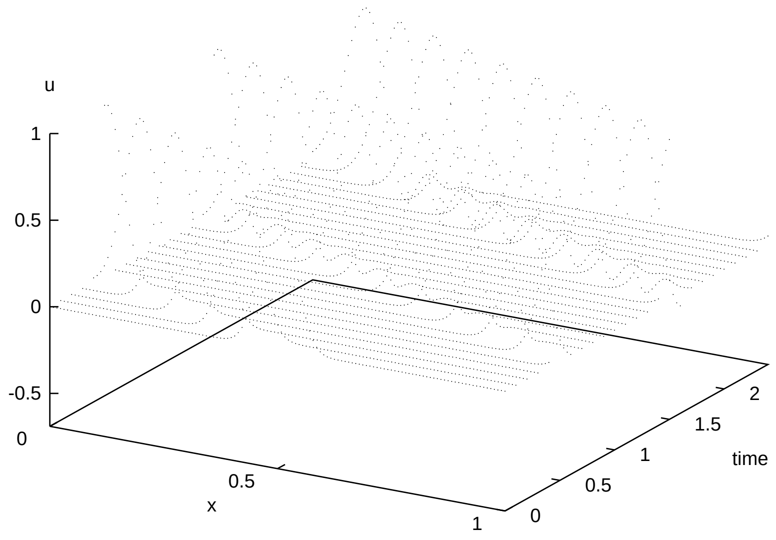

Figure 1: Leapfrog evolution for `level=7`, `nc=3`, `lambda=0.8`. Note the little “bump” growing behind the pulse.

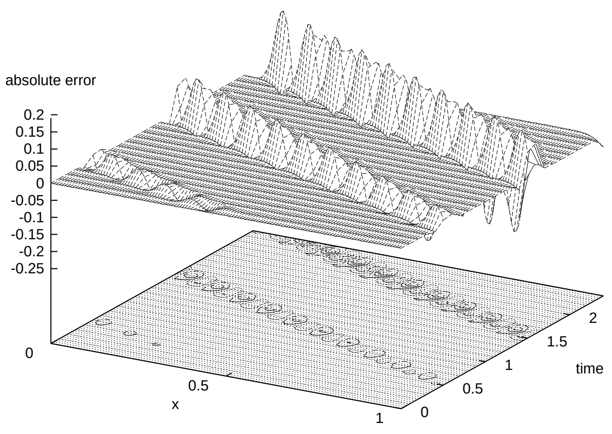

Figure 2: Error `level=7`, `nc=3`, `lambda=0.8`

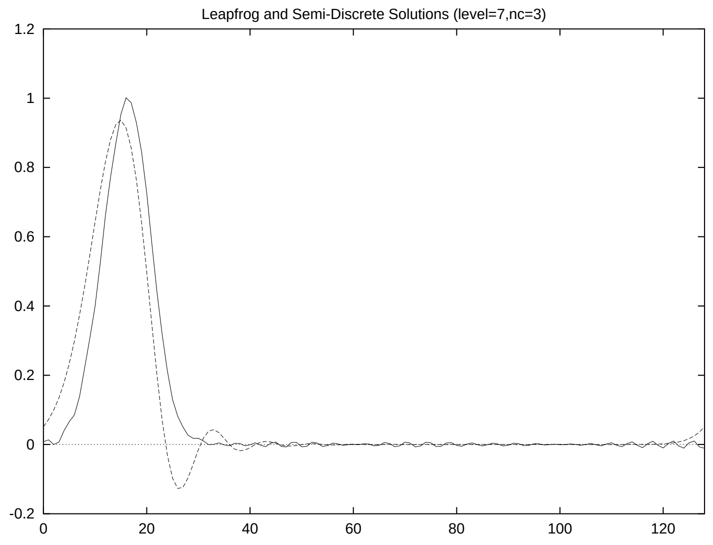

Figure 3: Comparison. The dashed plot is the *leapfrog* solution, and the solid line is the semi-discrete solution for `level=7`, `nc=3`, `lambda=0.8`

using a leap frog differencing scheme, we define the grid function `phi` at the retarded, current, and advanced time steps (-1,0,1). The operator statements define the time (and space) first derivative (second order center-differenced) operators. We initialize our grid function with the given initial data of `phi` being a Gaussian pulse. Finally, we ask RNPL to loop through solution iteratively and update the solution for `phi` automatically.

The usage of the RNPL program is `advect1d parfile` where parfile contains the parameters necessary for running the program. These include user defined parameters (which will over-write the values specified in the RNPL source) and parameters created and used by RNPL. The minimum and maximum x values for the domain are set to 0 and 1.0, respectively, and the initial data parameters for the gaussian (`amp`, `cent`, and `sigma`) are also set as specified in the problem set.

6.3 Semi-discrete Solution

The semi-discrete solution we have implemented uses higher order finite differences (a 5-point scheme: $\frac{1}{12h}\left(u_{j-2} - 8u_{j-1} + 8u_{j+1} - u_{j+2}\right)$ which has a leading order error of $\frac{1}{30}h^4$ for the spatial part of the *advection equtaion* and the `LSODE` integrator from the first assignment to advance the time-dependent part of the equation). Which is to say, for every grid point in the spatial domain, we generate a first order ODE to be solved with `LSODE`. The ODEs for all the grid points are coupled, and therefore, are advanced simultaneously by `LSODE`. For smaller grids (`level` $\approx$ 6), this scheme produces results with errors perhaps $\frac{1}{3}$ the size of the errors seen in the fully-discrete solutions with no appreciable difference in computing time. As the grid grows larger, however, the semi-discrete solution takes *far* longer to produce output (which is natural, since it is solving such a large system of coupled ODEs).

6.4 Code Tests and Comparisons

We performed a few basic tests of the code to insure that it was working properly. Simply comparing the results produced by different implementations (particularly comparing the fully-discrete *leapfrog* solutions with the semi-discrete *method-of-lines* solutions) is very informative, since, while the two integration schemes may both be in error, it is very unlikely that they would produce the same errors. These comparisons are seen graphically in figures (3) and (4).

Since the C and RNPL versions are implementing the same explicit finite-difference scheme on the same grids, they quite naturally have almost exactly the same output. The differences between

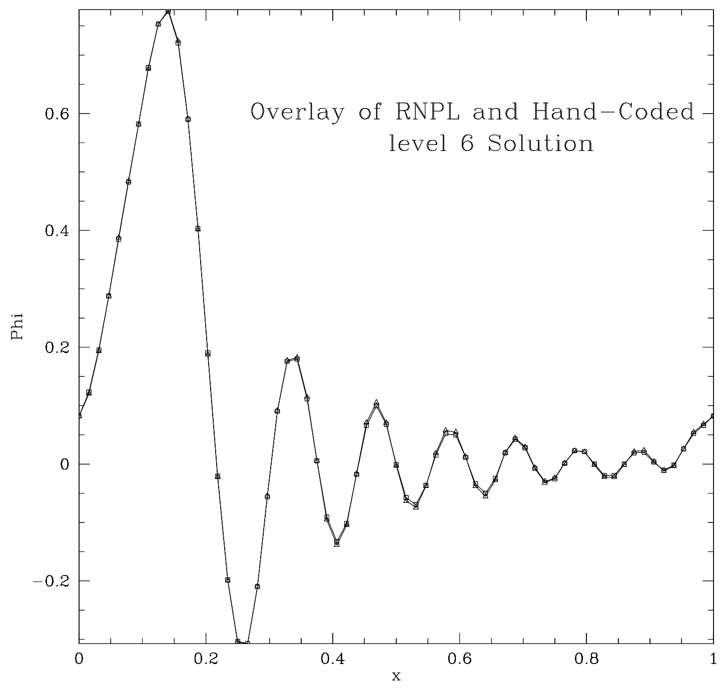

Figure 4: Comparison between handcoded and RNPL solutions.

them are almost exclusively due to the way the initial conditions are set up (specifically, data for the first time step, which is required to get the *leapfrog* scheme started, but which is not computed by the *leapfrog* method). In the C program, the first time step is computed exactly from our analytic solution. In the RNPL, the first timestep is generated by iteratively applying a *leapfrog* scheme on a half-sized grid.

We also performed a simple convergence test by running the programs for three consecutive values of the `level` parameter. Then, following the logic suggested by sections 4 and 5, we subtract the solutions on the finer grids from the solutions on the coarser grids *i.e.* forming $u^{2h} - u^h$ and $u^{4h} - u^{2h}$. To show quadratic convergence, we plot $4(u^{4h} - u^{2h})$ and $u^{2h} - u^h$ on the same graph and note that they are approximately equal. The convergence factor graphs are figures (5) and (6).

Additionally, for the hand-coded C version of advect1d, we plot the absolute error and the expected leading order error on the same graph. For coarse grids, these can differ markedly, but as the grid size is increased and the leading order error term comes to dominate the the total error, these graphs line up nicely. For `level` $= 10$ the two graphs are almost exactly coincident.

6.5 Directory Structure

The files for the C version of the advect1d program are located in:

`/d/feynman/usr/people/p387g2/hw2/p1/advect1d`

The files for the RNPL version are located in:

`/d/feynman/usr/people/p387g2/hw2/p1/rnpl`

The files for the Semi-discrete solution in Fortran are located in:

`/d/feynman/usr/people/p387g2/hw2/p1/semi-discrete`

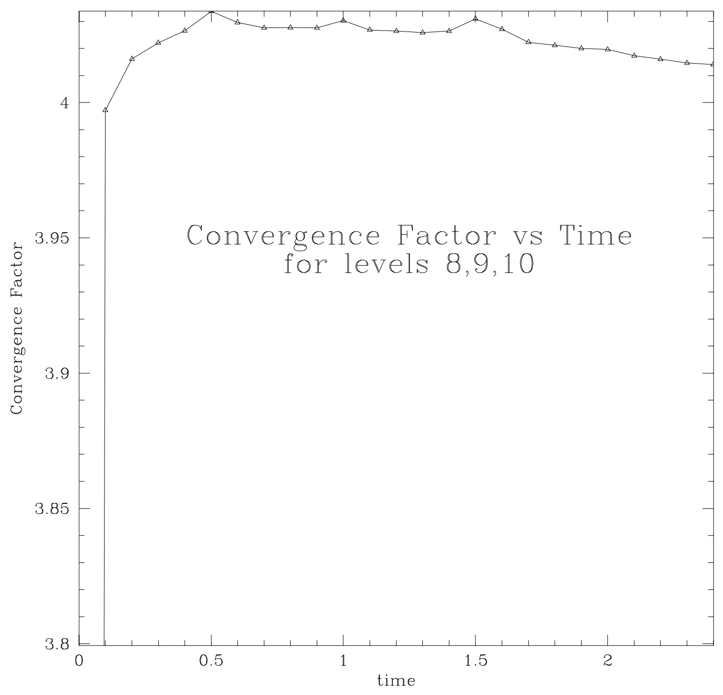

Figure 5: Convergence for the handcoded solution.

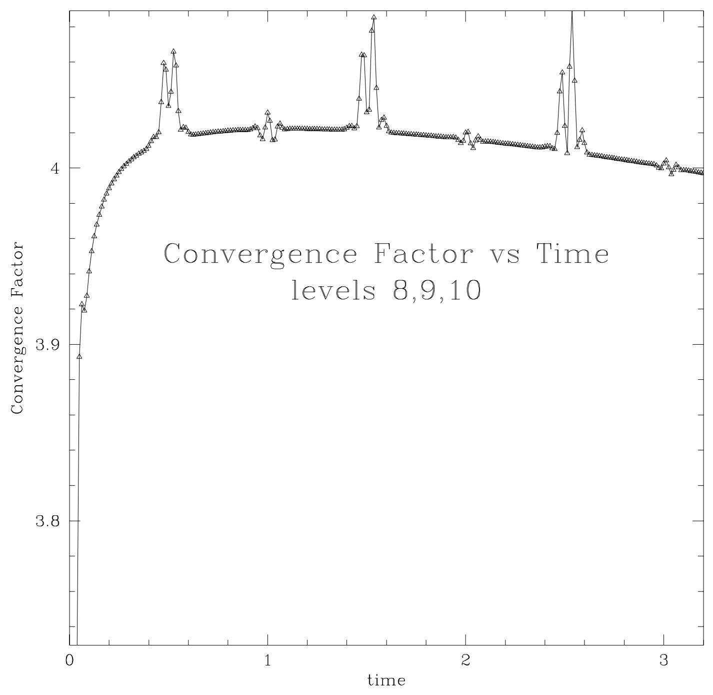

Figure 6: Convergence for the RNPL solution.

The files for the latex documentation are in

`/d/feynman/usr/people/p387g2/hw2/p1/latex`

Our web page, containing more information about the solution to these problems, with MPEG sequences and GIF images showing additional data is at:

`http://wwwrel.ph.utexas.edu/ p387g2`