1+1 Wave Equation

Abhishek Joshi

October 2020

1 Introduction

This is a write-up on the 1+1 wave equation problem, a simple example to demonstrate how to evolve a hyperbolic partial differential equation. Parts of the write-up include Guzman's article (Introduction to Numerical Relativity [2]) In this case, we consider the scalar wave equation.

$$\Box\Phi = \frac{1}{\sqrt{-g}}\partial_\mu\left[\sqrt{-g}g^{\mu\nu}\partial_\nu\Phi\right] = 0 \tag{1}$$

Here we assume flat 1D space. Without imposing any temporal or spatial gauge conditions, the metric is:

$$g_{\mu\nu} = \begin{pmatrix} (-\alpha^2+\beta^2) & \beta \\ \beta & 1 \end{pmatrix} \tag{2}$$

2 Formulation as a Time Evolution Problem

A common technique in numerical simulations is to convert a second order differential equation to two coupled first order PDEs.

Plugging in the metric in the scalar wave equation and simplifying a little, we get

$$\Box\Phi = \frac{1}{\alpha}\partial_t\left[-\frac{1}{\alpha}\partial_t\Phi + \frac{\beta}{\alpha}\partial_x\Phi\right] + \frac{1}{\alpha}\partial_x\left[\frac{\beta}{\alpha}\partial_t\Phi + \alpha\left(1-\frac{\beta^2}{\alpha^2}\right)\partial_x\Phi\right] = 0 \tag{3}$$

For simplicity, let us first start with $\alpha = 1$ and $\beta = 0$. Now defining our fields,

$$\Pi = (\partial_t\Phi) \tag{4}$$

Then we have,

$$\partial_t\Pi = \partial_x\partial_x(\Phi) \tag{5}$$

We now have two first order PDEs that we can evolve to solve for Φ

3 Numerical Formulation

Finite Difference: For simplicity, let us first start with $\alpha = 1$ and $\beta = 0$. To compute the second order spatial derivatives, we use a second or fourth order finite difference stencil. Finite difference schemes are basically Taylor expanding a function about a point to the desired order and solving for an approximate value of the derivative at that point. For a uniformly spaced spatial grid with resolution h,

$$\text{Second order finite difference: } f''(x) = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} + O(h^2) \tag{6}$$

$$\text{Fourth order: } f''(x) = \frac{-f(x+2h) + 16f(x+h) - 30f(x) + 16f(x-h) - f(x-2h)}{12h^2} + O(h^4) \tag{7}$$

Method of Lines: As for the time integration we use the method of lines. [3] is a nice pedagogical article on the method of lines (MoL) as well as the basics of numerical integration of differential equations. The basic concept of MoL is to approximate the derivatives in every dimension but one in the differential equations algebraically (Taylor expansion using finite difference, for example). Then the last dimension (typically time) can be integrated using any ODE integration technique.

Runge-Kutta Scheme: The Runge-Kutta methods are a family of iterative numerical methods to solve an ODE (usually temporal discretization) to a certain order. The basic principle is to divide a time step in half, and iteratively compute better and better slope estimates of the function at the half step to predict what the value of the function is at the full time step ahead. The number of times this iterative procedure is done is dependent on the desired order of accuracy of the integration. The most common fourth order Runge-Katta (rk4) is given as (taken from [5]):

For an initial value problem,

$$\frac{dy}{dt} = f(t, y) \qquad\qquad y(t_0) = y_0$$

The rk4 algorithm is given as (for time-step h):

$$y_{n+1} = y_n + \frac{1}{6}h\,(k_1 + 2k_2 + 2k_3 + k_4) \tag{8}$$

$$t_{n+1} = t_n + h \tag{9}$$

Where,

$$k_1 = f(t_n, y_n) \tag{10}$$

$$k_2 = f(t_n + \frac{h}{2}, y_n + h\frac{k_1}{2}) \tag{11}$$

$$k_3 = f(t_n + \frac{h}{2}, y_n + h\frac{k_2}{2}) \tag{12}$$

$$k_4 = f(t_n + h, y_n + hk_3) \tag{13}$$

One important note here is that this is the definition for a non-coupled ODE. In our case, since Π, Ψ and Φ are coupled, they need to be solved together. For example, computing k_3 in the Π evolution equation will require one to first compute k_2 in the Ψ equation as $k_3^\Pi = \partial_x \left(\Psi_n + hk_2^\Psi/2\right)$.

4 Numerical Implementation

First defining some additional details about the problem (might vary from the actual code).

- Set the propagation speed c=1
- The initial condition is $\Phi_0 \propto \exp\left[-(x-0.5)^2\right]$ and $\partial_t\Phi(t=0)=0$
- Periodic boundary conditions with x=-2 to 2 with $\Delta x = 0.01$
- Integrating out from t=0 to t=10 with $\Delta t = 0.0025$

The code is given in [4]. It is a simple python code that runs reasonably fast (for python) as the array operations are numpy array operations for the most part. Some results are shown in fig. 1. A video is also uploaded in the repository.

For stability and accuracy, the choice of resolution in space and time are important. It can be shown that for our explicit time integration scheme,

$$\Phi_{n+1} \propto \Phi_n c \frac{\Delta t}{\Delta x}$$

Where c is the speed of the wave. Clearly, if $\frac{\Delta t}{\Delta x} > 1$, the solution will grow and become unstable. This value $c\frac{\Delta t}{\Delta x}$ is called the Courant-Freidrichs-Lewy (CFL) condition. The smaller the CFL, the better the simulation, however this is a necessary but not sufficient condition. Care should be taken that the individual resolutions of each dimension are not too small as that would propagate errors and cause instability even if the CFL is low.

5 Convergence Analysis

Convergence in general is a well known term. In this case, we consider the convergence of errors. As implied before, the error in the simulation (compared to the analytic true solution) occurs due to discretization of space and time. It is useful to know how quickly these errors reduce with an increase in resolution. For example, if a fourth order finite difference method and an rk4 integrator was used, the error can be expected to converge as $O(h^4)$.

The following work is adapted from [1]. If we interpret our numerical solution as a continuum function expanded as a power series about the resolution parameter (Δ) as:

$$\Phi_\Delta(t,x) = \Phi(t,x) + \Delta e_1(t,x) + \Delta^2 e_2(t,x) + ... \tag{14}$$

where $e_n(t,x)$ represents the unscaled error of the approximation at the nth order. Therefore, for an nth order scheme,

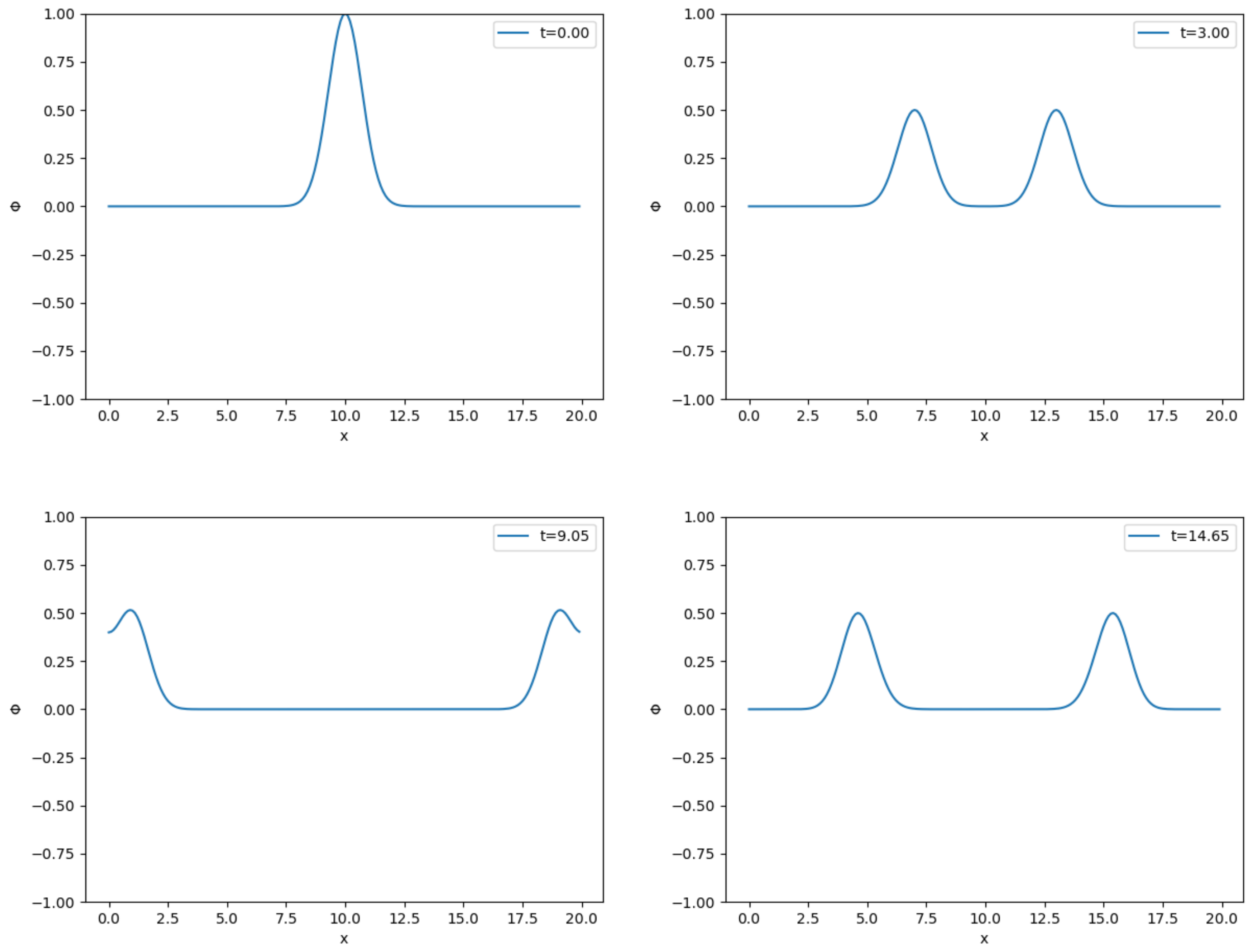

Figure 1: Numerical simulation of the wave equation at different points in time. 2nd order finite difference with CFL=0.5.

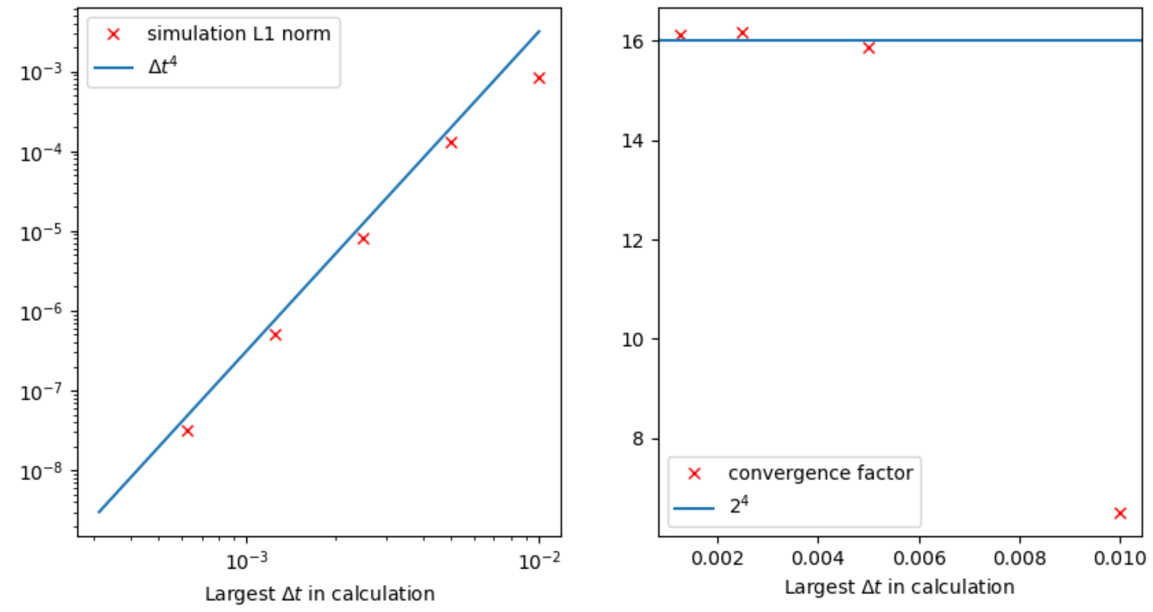

Figure 2: Temporal resolution convergence plot of the 1+1 wave equation simulation using an RK4 integrator.

$$\Phi(t,x) - \Phi_\Delta(t,x) \propto \Delta^n \tag{15}$$

and e_1=...=$e_{n-1} = 0$.

Since in most cases of numerical computation, the analytic solution is unknown, we run the simulation for at least 3 different resolutions $\Delta_1 = \frac{\Delta_2}{r} = \frac{\Delta_3}{r^2}$ where r is an arbitrary number (chosen as 2 here). Using the L1 norm to compute difference between two resolutions, we define the convergence factor as:

$$c(t) := \frac{||\Phi_{\Delta_1} - \Phi_{\Delta_2}||}{||\Phi_{\Delta_2} - \Phi_{\Delta_3}||} \tag{16}$$

Now in the continuum limit ($\Delta \to 0$), the convergence factor becomes:

$$\lim_{\Delta \to 0} c(t) = \frac{\Delta_1^n - \Delta_2^n}{\Delta_2^n - \Delta_3^n} = r^n \tag{17}$$

Coming back to our example, plotting out the L1 norm differences as well as the convergence factors for both time (Fig. 2) and space (fig. 3) shows that our methods are converging as expected, as RK4 is a 4th order scheme and we chose a 2nd order finite difference method. Note, care should be taken when checking the convergence in x as increasing the resolution in x will increase the CFL condition.

For two resolutions, $\Delta_1 = \frac{\Delta_2}{r}$, we can estimate the error in the higher resolution. Plugging in the approximations from equation 14,

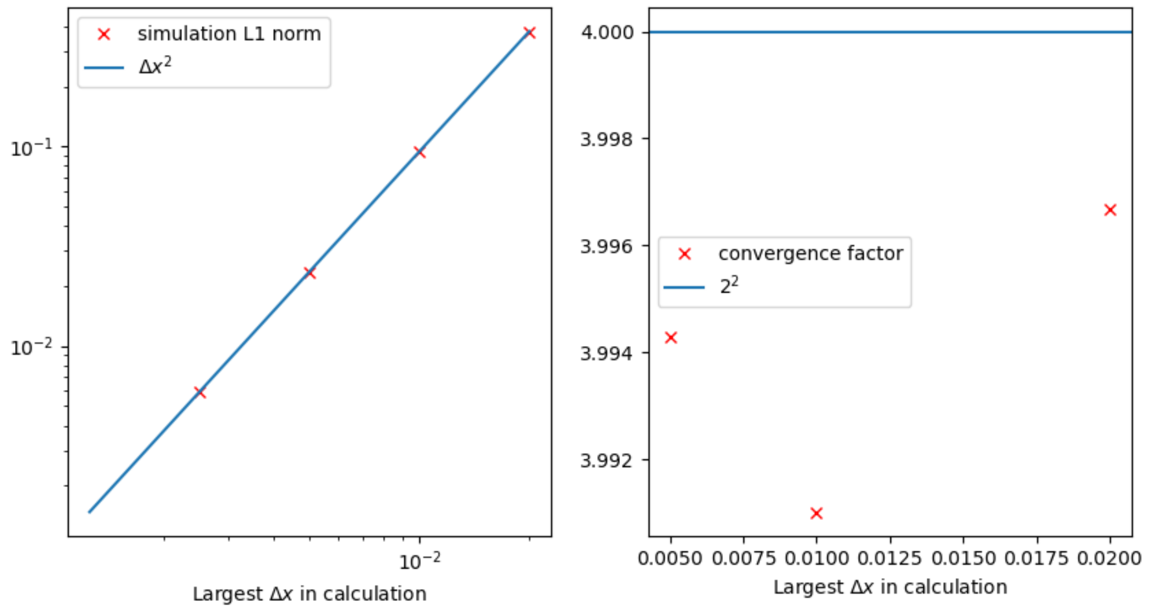

Figure 3: Spatial resolution convergence plot of the 1+1 wave equation simulation using a 2nd order finite difference scheme. Temporal resolution was taken to be small enough such that the CFL condition is not large for any of the spatial resolutions.

$$\Phi_{\Delta_1} - \Phi_{\Delta_2} = e_n\left(\Delta_1^n - \Delta_2^n\right) + O\left(\Delta^{n+1}\right) = e_n \Delta_2^n (r^n - 1)$$

Therefore, the total error for the higher resolution:

$$\epsilon_n \Delta_2^n \approx \frac{1}{r^n - 1}\left(\Phi_{\Delta_1} - \Phi_{\Delta_2}\right) \tag{18}$$

And the mean relative error is (for temporal convergence, using $\Delta_1 = 6.25 \times 10^{-4}$ and $\Delta_2 = 3.125 \times 10^{-4}$),

$$\mathbb{E}\left[\frac{\epsilon_n \Delta_2^n}{\Phi_{\Delta_2}}\right] = \mathbb{E}\left[\frac{\left(\Phi_{\Delta_1} - \Phi_{\Delta_2}\right)}{\left(r^n - 1\right)\Phi_{\Delta_2}}\right] \approx 9.16 \times 10^{-15} \tag{19}$$

Which is very close to double precision.

References

[1] Miguel Alcubierre. *Introduction to 3+1 Numerical Relativity.* 2008.

[2] F. S. Guzmán. Introduction to numerical relativity through examples. *Revista Mexicana de Fisica Supplement*, 53(4):78–93, September 2007.

[3] S. Hamdi, W. E. Schiesser, and G. W Griffiths. Method of lines. *Scholarpedia*, 2(7):2859, 2007. revision #124335.

[4] A. V. Joshi. 1+1 wave equation. `https://github.com/avjoshi21/1p1-WaveEqn/tree/master`, 2020.

[5] Wikipedia contributors. Runge–kutta methods — Wikipedia, the free encyclopedia, 2020. [Online; accessed 8-October-2020].