Finite Difference Methods for the Hyperbolic Wave Partial Differential Equations

Shuonan Dong
18.086 Spring

1 Introduction

The purpose of this discussion is to look at the different finite difference schemes for first and second order wave equations in 1-D and 2-D. Since we have covered much of the first order 1-D wave equation material in the 18.086 course, I will not dwel too deeply on derivations of stability and order of accuracy for each finite difference scheme, but rather focus more on the general performance of the schemes. Second order wave equations, as it turns out, generally do not have as many finite difference schemes associated with them, unless they are split into several first order equations. Thus each of these schemes will be discussed in a bit more detail. All schemes are implemented in MATLAB 7.0.4, and can be found online at http://mit.edu/dongs/Public/18.086/Project1.

2 First order one-way wave equation

The first order wave equation in one-dimensional space is as follows:

$$u_t = cu_x \tag{1}$$

where c is a positive constant, and $u(x,t)$ is subject to the initial condition

$$u(x,0) = f(x), \quad -\infty < x < \infty. \tag{2}$$

The solution for $t \geq 0$ and all x is a family of characteristics, which are straight lines shifted to the left in the x,t-plane, inclined to the x-axis at an angle

$$\theta = \tan^{-1}\left(\tfrac{1}{c}\right). \tag{3}$$

The explicit solution is

$$u(x,t) = f(x+ct). \tag{4}$$

2.1 Finite difference schemes

There are several finite difference schemes that I implemented for the first order 1-D wave equation. These include the explicit schemes: Forward Euler, Upwind, Lax-Friedrichs, Lax-Wendroff, Leapfrog, and Fourth-order Leapfrog; and the implicit schemes: Backward Euler, Crank-Nicolson, and Box. These schemes were found in [Threfethen 1994], and are outlined in Table 1 and 2. In all cases, $r = c\dfrac{\Delta t}{\Delta x}$. It is also

useful to keep in mind the stability and order of accuracy for each of these schemes. I include in Table 3 the same stability and accuracy information found in Chapter 4 of [Threfethen 1994]. Many of these values are verified by experimentation in a later section.

Table 1. Explicit finite difference schemes for first order 1-D wave equation

FD Scheme	Matrix Representation
Forward Euler (FEU) $u_j^{n+1} = u_j^n + \frac{1}{2} r \left(u_{j+1}^n - u_{j-1}^n \right)$	$\mathbf{u}^{\mathbf{n+1}} = \begin{bmatrix} 1 & \frac{1}{2}r & & & \\ -\frac{1}{2}r & \ddots & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & & \\ & & & & \end{bmatrix} \mathbf{u}^{\mathbf{n}} + \begin{bmatrix} -\frac{1}{2}r \\ 0 \\ \vdots \\ 0 \\ \frac{1}{2}r \end{bmatrix}$
Upwind (UPW) $u_j^{n+1} = u_j^n + r\left(u_{j+1}^n - u_j^n\right)$	$\mathbf{u}^{\mathbf{n+1}} = \begin{bmatrix} 1-r & r & & & \\ & \ddots & \ddots & & \\ & & \ddots & \ddots & \\ & & & & \end{bmatrix} \mathbf{u}^{\mathbf{n}} + \begin{bmatrix} 0 \\ \\ \vdots \\ 0 \\ r \end{bmatrix}$
Lax-Friedrichs (LXF) $u_j^{n+1} = \frac{1}{2}\left(u_{j+1}^n - u_{j-1}^n\right) + \frac{1}{2} r \left(u_{j+1}^n - u_{j-1}^n\right)$	$\mathbf{u}^{\mathbf{n+1}} = \begin{bmatrix} 0 & \frac{1+r}{2} & & & \\ \frac{1-r}{2} & \ddots & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & & \\ & & & & \end{bmatrix} \mathbf{u}^{\mathbf{n}} + \begin{bmatrix} \frac{1-r}{2} \\ 0 \\ \vdots \\ 0 \\ \frac{1+r}{2} \end{bmatrix}$
Lax-Wendroff (LXW) $u_j^{n+1} = u_j^n + \frac{1}{2} r \left(u_{j+1}^n - u_{j-1}^n\right) + \frac{1}{2} r^2 \left(u_{j+1}^n - 2u_j^n + u_{j-1}^n\right)$	$\mathbf{u}^{\mathbf{n+1}} = \begin{bmatrix} 1-r^2 & \frac{r(r+1)}{2} & & & \\ \frac{r(r-1)}{2} & \ddots & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & & \\ & & & & \end{bmatrix} \mathbf{u}^{\mathbf{n}} + \begin{bmatrix} \frac{r(r-1)}{2} \\ 0 \\ \vdots \\ 0 \\ \frac{r(r+1)}{2} \end{bmatrix}$
Leapfrog (LFG) $u_j^{n+1} = u_j^{n-1} + r\left(u_{j+1}^n - u_{j-1}^n\right)$	$\mathbf{u}^{\mathbf{n+1}} = \mathbf{u}^{\mathbf{n-1}} + \begin{bmatrix} 0 & r & & & \\ -r & \ddots & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & & \\ & & & & \end{bmatrix} \mathbf{u}^{\mathbf{n}} + \begin{bmatrix} -r \\ 0 \\ \vdots \\ 0 \\ r \end{bmatrix}$
Fourth-order Leapfrog (LF4) $u_j^{n+1} = u_j^{n-1} + \frac{4}{3} r \left(u_{j+1}^n - u_{j-1}^n\right) - \frac{1}{6} r \left(u_{j+2}^n - u_{j-2}^n\right)$	$\mathbf{u}^{\mathbf{n+1}} = \mathbf{u}^{\mathbf{n-1}} + \begin{bmatrix} 0 & \frac{4}{3}r & -\frac{1}{6}r & & \\ -\frac{4}{3}r & \ddots & \ddots & \ddots & \\ \frac{1}{6}r & \ddots & \ddots & \ddots & \ddots \\ & \ddots & \ddots & & \\ & & \ddots & & \end{bmatrix} \mathbf{u}^{\mathbf{n}} + \begin{bmatrix} -\frac{4}{3}r \\ \frac{1}{6} \\ 0 \vdots \\ -\frac{1}{6}r \\ \frac{4}{3}r \end{bmatrix}$

Table 2. Implicit finite difference schemes for first order 1-D wave equation

Backward Euler (BEU) $u_j^{n+1} = u_j^n + \frac{1}{2} r\left(u_{j+1}^{n+1} - u_{j-1}^{n+1}\right)$	$\mathbf{u}^{\mathbf{n+1}} = \begin{bmatrix} 1 & -\frac{1}{2}r & & \\ \frac{1}{2}r & \ddots & \ddots & \\ & \ddots & \ddots & \ddots \\ & & \ddots & \\ & & & \end{bmatrix}^{-1} \left(\mathbf{u}^{\mathbf{n}} + \begin{bmatrix} -\frac{1}{2}r \\ 0 \\ \vdots \\ 0 \\ \frac{1}{2}r \end{bmatrix} \right)$
Crank-Nicolson (CNS) $u_j^{n+1} = u_j^n + \frac{1}{4} r\left(u_{j+1}^{n} - u_{j-1}^{n}\right) + \frac{1}{4} r\left(u_{j+1}^{n+1} - u_{j-1}^{n+1}\right)$	$\mathbf{A1} = \begin{bmatrix} 1 & \frac{1}{4}r & & \\ -\frac{1}{4}r & \ddots & \ddots & \\ & \ddots & \ddots & \ddots \\ & & \ddots & \\ & & & \end{bmatrix}$ $\mathbf{A} = \begin{bmatrix} 1 & -\frac{1}{4}r & & \\ \frac{1}{4}r & \ddots & \ddots & \\ & \ddots & \ddots & \ddots \\ & & \ddots & \\ & & & \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} -\frac{1}{2}r \\ 0 \\ \vdots \\ 0 \\ \frac{1}{2}r \end{bmatrix}$ $\mathbf{u}^{\mathbf{n+1}} = \mathbf{A1}^{-1}\left(\mathbf{Au}^{\mathbf{n}} + \mathbf{b}\right)$
Box (BOX) $(1-r)u_j^{n+1} + (1-r)u_{j+1}^{n+1} = (1-r)u_j^n + (1+r)u_{j+1}^n$	$\mathbf{A1} = \begin{bmatrix} 1+r & 1-r & & \\ & \ddots & \ddots & \\ & & \ddots & \ddots \\ & & & \\ & & & \end{bmatrix}$ $\mathbf{A} = \begin{bmatrix} 1-r & 1+r & & \\ & \ddots & \ddots & \\ & & \ddots & \ddots \\ & & & \\ & & & \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 0 \\ \\ \vdots \\ 0 \\ 2r \end{bmatrix}$ $\mathbf{u}^{\mathbf{n+1}} = \mathbf{A1}^{-1}\left(\mathbf{Au}^{\mathbf{n}} + \mathbf{b}\right)$

Table 3. Stability and order of accuracy for finite difference schemes

FD Scheme	Order of accuracy	CFL stability restriction	Exact stability restriction
Forward Euler	1	$r \le 1$	Unstable
Upwind	1	$r \le 1$	$r \le 1$
Lax-Friedrichs	1	$r \le 1$	$r \le 1$
Lax-Wendroff	2	$r \le 1$	$r \le 1$
Leapfrog	2	$r \le 1$	$r \le 1$
4th-order Leapfrog	2	$r \le 2$	$r \le 0.728...$
Backward Euler	1	None	None
Crank-Nicolson	2	None	None
Box	2	None	None

2.2 Performance of finite difference schemes

2.2.1 Setup

Let's see how each finite difference scheme performs on some sample problems. The three sample initial conditions that I will use for experimentation include a step function, a discrete approximation of the delta function, and a smooth sinusoidal function. The initial conditions are written out in equations 5-7. Although they are quite arbitrarily chosen, I hope they will produce some interesting results.

$$u_1(x,0) = \begin{cases} 0 & x < 0 \\ 1 & x \ge 0 \end{cases} \tag{5}$$

$$u_2(x,0) = \begin{Bmatrix} 0 & x < 0 \\ \frac{1}{\Delta x} & x = 0 \\ 0 & x > 0 \end{Bmatrix} \approx \delta(x) \tag{6}$$

$$u_3(x,0) = \sin(2x) \tag{7}$$

The other input factors include values for c, Δt, and Δx. I have chosen some representative combinations for experimentation, as shown in Table 4.

Table 4. Representative trials for c, Δt, and Δx

Trial	c	Δx	Δt	r
1	0.5	0.04	0.02	0.25
2	0.5	0.02	0.02	0.5
3	0.5	0.0137	0.02	0.728
4	0.5	0.0101	0.02	0.99
5	0.5	0.0099	0.02	1.11
6	0.5	0.02	0.01	0.25
7	0.5	0.02	0.04	1

2.2.2 Results

Using the step function initial condition and Trial 1 input values from Table 4 produces the results for each finite difference scheme shown in Figures 1 and 2 below.

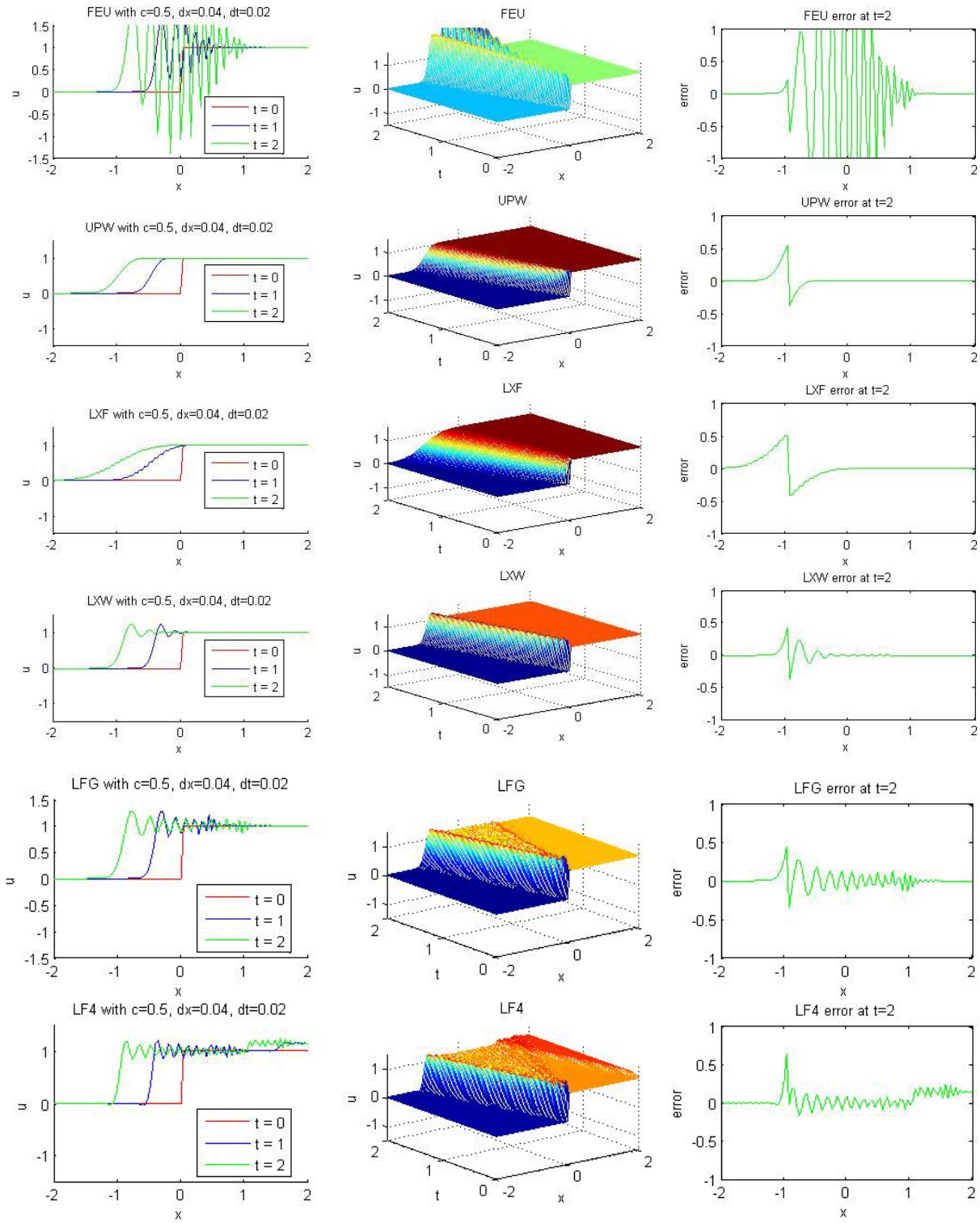


Figure 1. Results of explicit methods using step initial condition and Trial 1 inputs.

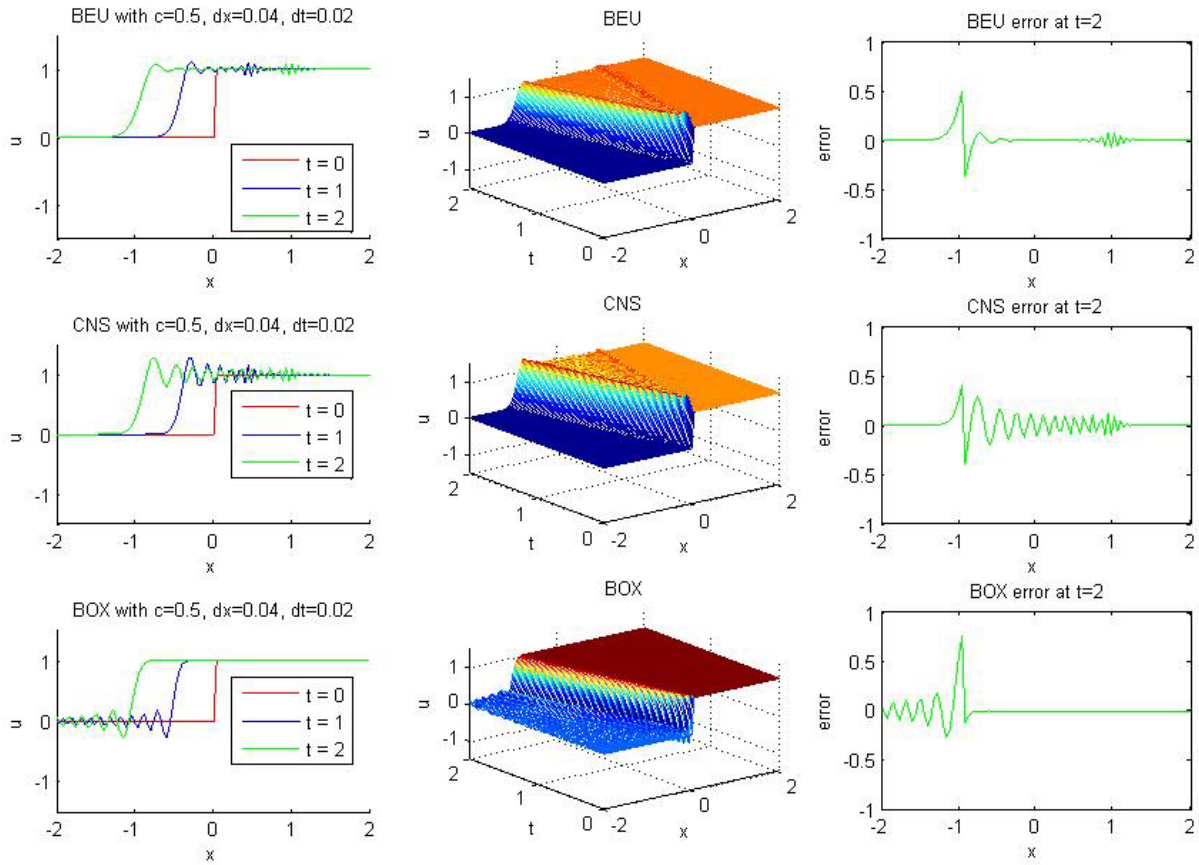


Figure 2. Results of implicit methods using step initial condition and Trial 1 inputs.

The inputs provide an *r* value of 0.25, which is quite small and should produce fairly stable results. From Figure 1, it is apparent that Forward Euler is not very stable, even for such a small *r*. Upwind and Lax-Friedrichs both produce nice inosculating results, but have much longer shock widths (i.e. take much longer to make the step) than most of the other schemes. Several schemes including the Leapfrogs, Backward Euler, and Crank-Nicolson produce oscillations between $x_0 - ct$ and $x_0 + ct$, where x_0 is the initial position of the step. The sudden discontinuity of the step function causes the ripple in the Leapfrog and implicit algorithms. The Box method anticipates a smooth and quick step by creating oscillations before the step. Lax-Wendroff initially overshoots the step, but then quickly dampens out the oscillations.

From the initial glance, each finite difference scheme has different merits and limitations, save Forward Euler, which simply leaves much to be desired.

Now let's try the delta function using the same input values as before. The results are shown in Figures 3 and 4.

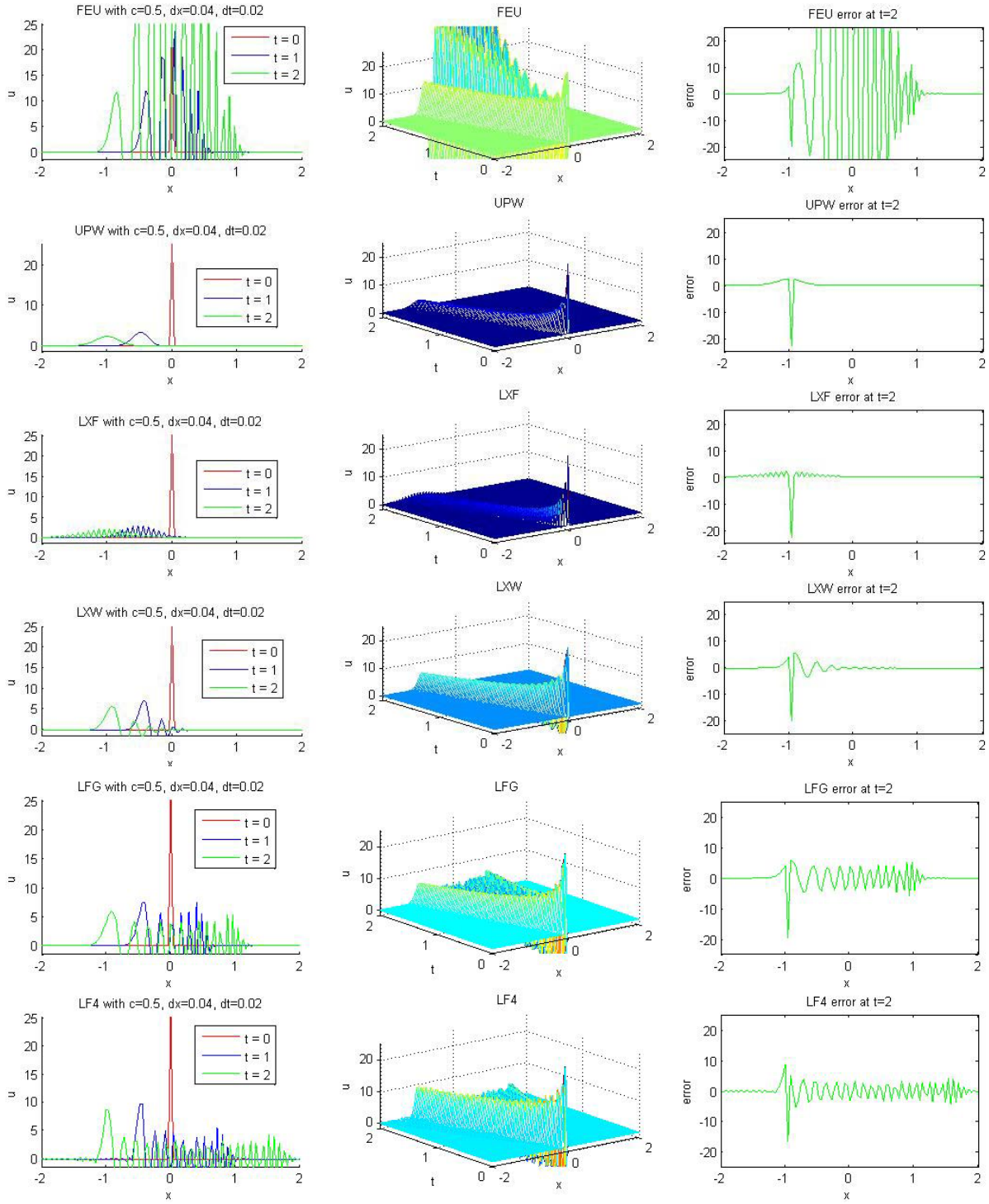


Figure 3. Results of explicit methods using delta initial condition and Trial 1 inputs.

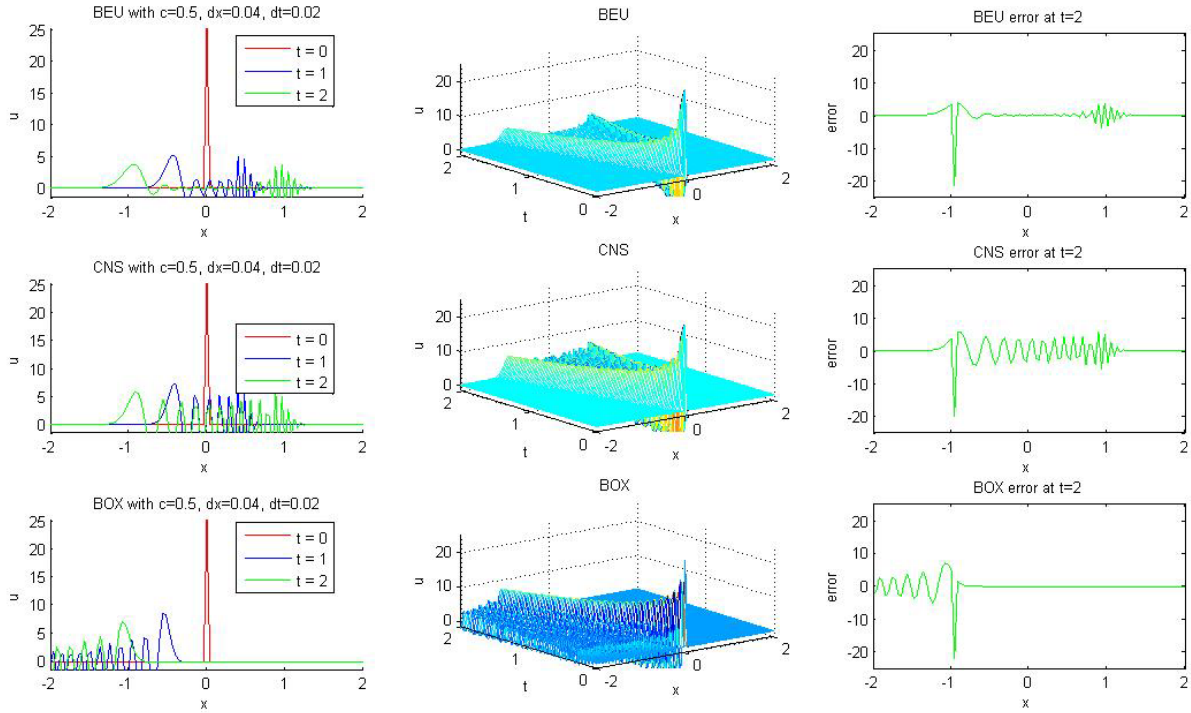


Figure 4. Results of implicit methods using delta initial condition and Trial 1 inputs.

The implementation of the delta function is a discrete approximation of the real delta function. One can also think of this approximation as a very thin triangle wave. The most difficult thing for each finite difference scheme to maintain is the height of the peak as it travels through time. The peak height was initialized to $\frac{1}{\Delta x}$ or 25. When $t = 2$, most of the finite difference schemes have reduced the peak to around 5. An interesting thing to note is that although Forward Euler is very unstable, it does maintain the highest peak of any of the finite difference schemes.

Now let's try the sinusoidal initial condition with the same input values as before. Results are shown in Figures 5 and 6.

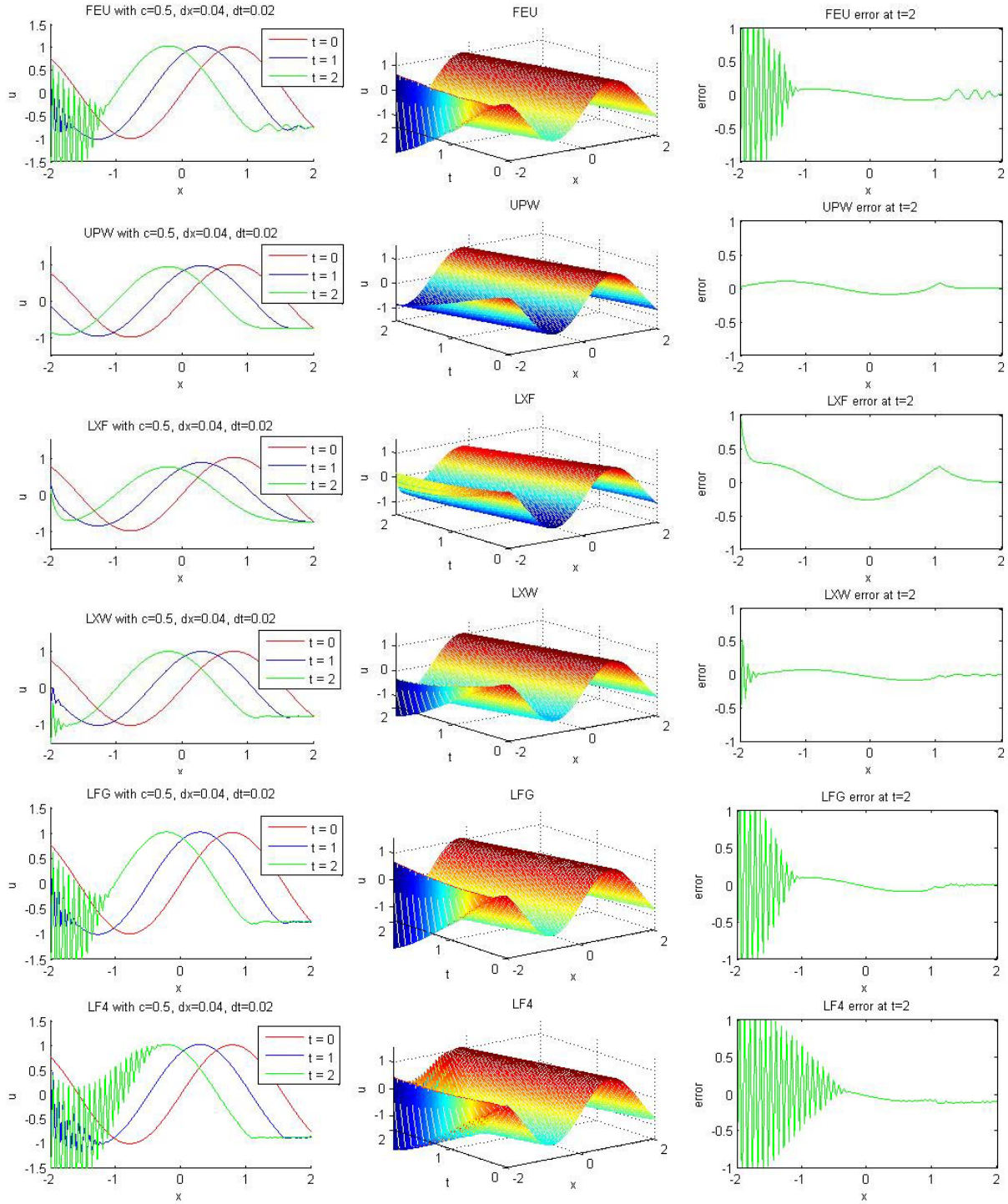


Figure 5. Results of explicit methods using sinusoidal initial condition and Trial 1 inputs.

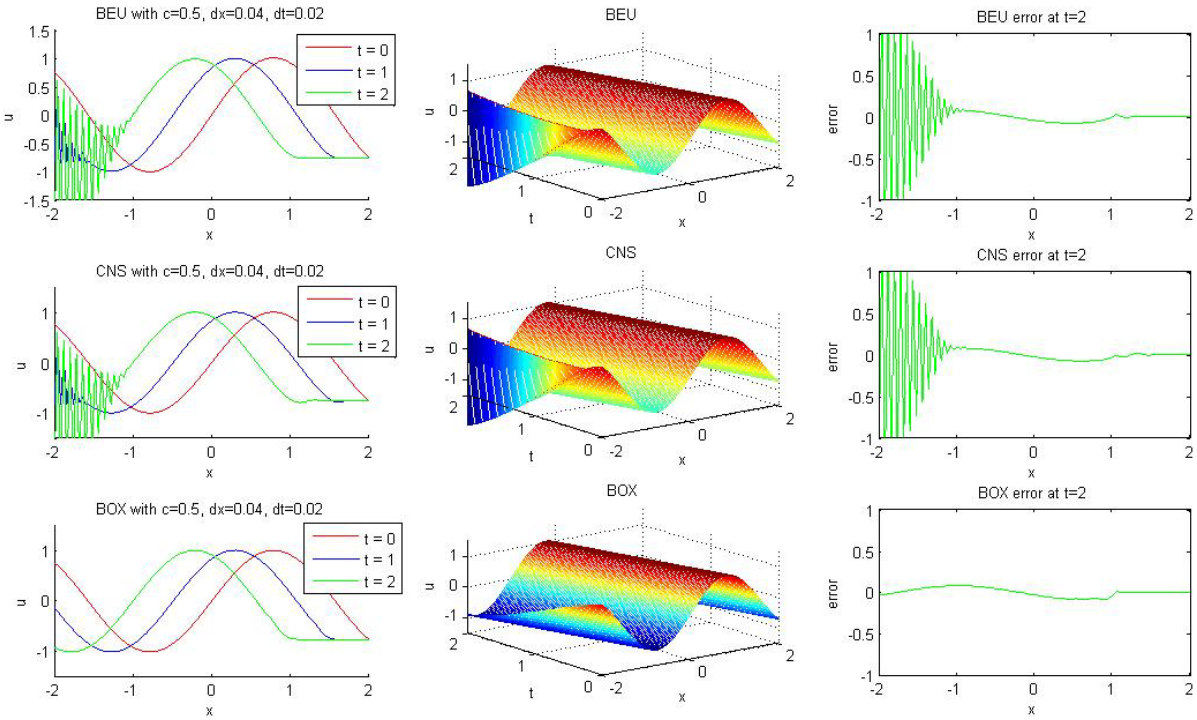
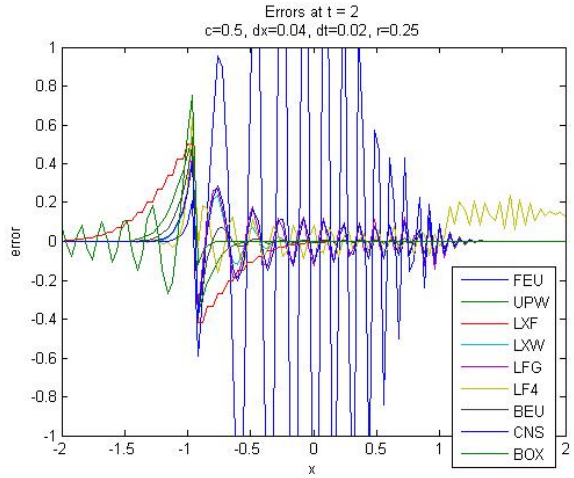


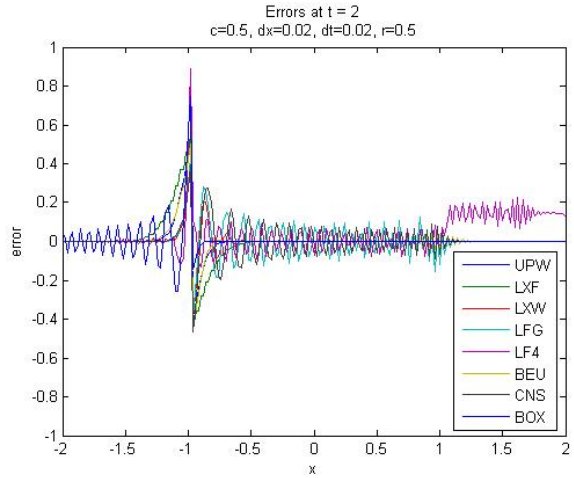
Figure 6. Results of implicit methods using sinusoidal initial condition and Trial 1 inputs.

The sinusoidal initial condition causes some interesting oscillations on the left boundary in many of the finite difference schemes. This is likely a boundary setup issue. If we assume that we are dealing with a boundless problem, we can ignore the boundary oscillations. The smoothness of the sinusoid gives the finite difference methods an easier time in modeling – most of the finite difference schemes return errors that are very close to zero. Lax-Friedrichs tended to shrink or smooth out the size of the wave, so it produced slightly more error than the other schemes. Pretty much all of the other finite difference methods produced fairly good results on the sinusoidal initial condition.

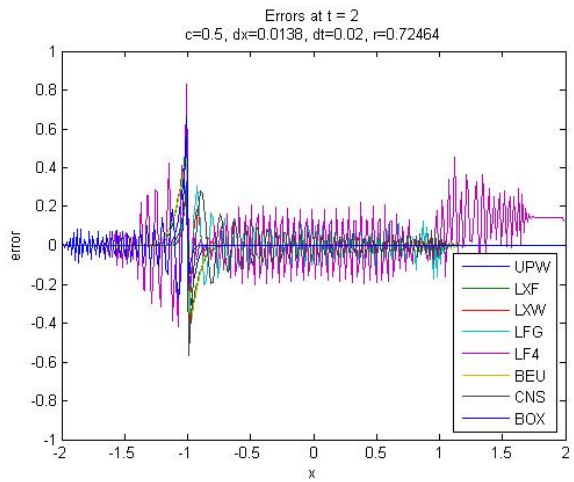
Now that we have looked at the performance of the 9 finite difference schemes using three different initial conditions, we can also look at different input values for c, Δt, and Δx. I will use the step function initial condition and follow Table 4 for the input values. Instead of showing all of the results as I have done in the past three cases, I will just show the errors for each trial to make it easier for comparison. See Figure 7 for the details.



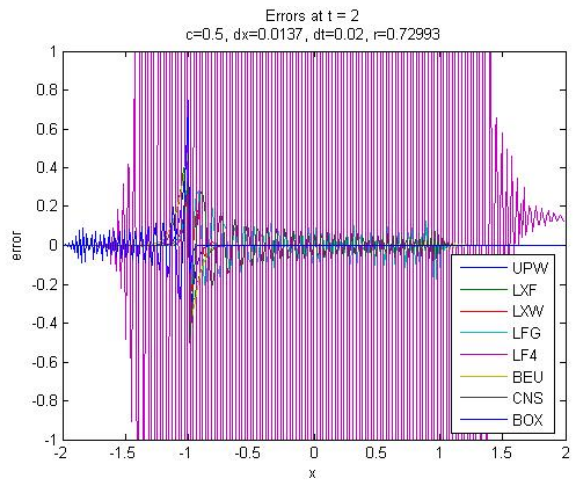
(a) $r = 0.25$



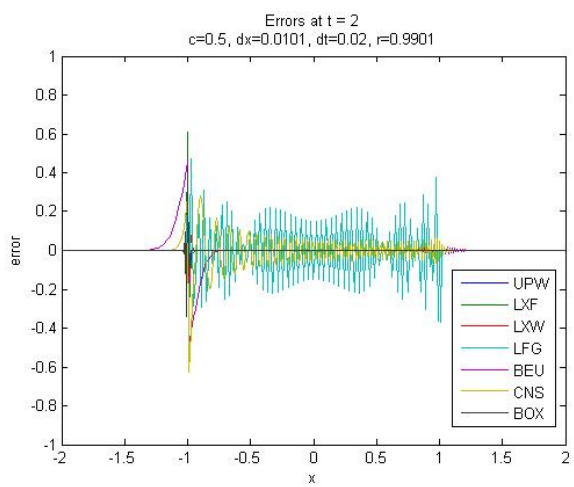
(b) $r = 0.5$



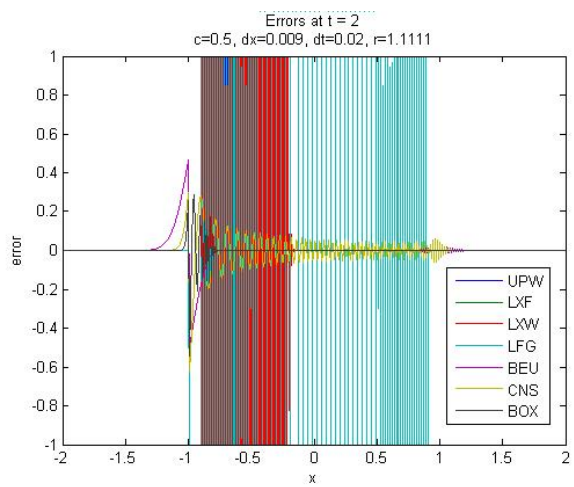
(c) $r = 0.725$



(d) $r = 0.73$



(e) $r = 0.99$



(f) $r = 1.11$

Figure 7(a-f). Procession of errors as *r* increases. Once a scheme is unstable, it is not shown.

Figure 7 experimentally verifies the exact stability restrictions in Table 3. We can see that from the beginning, Forward Euler is unstable, even for very small *r*. Fourth-order Leapfrog goes unstable at some point between $r = 0.725$ and $r = 0.73$. It is described as unstable for $r > 0.728$ in Table 3. At $r = 0.99$, the explicit schemes that have not gone unstable already are preparing to; and we see that all the explicit schemes go unstable at $r = 1.11$.

If we look very closely at the error plots, we can verify that the orders of accuracies for the different schemes are as stated in Table 3.

3 Second order 1-D wave equation

The second order wave equation in one-dimensional space is as follows:

$$u_{tt} = c^2 u_{xx} \tag{8}$$

where *c* is a positive constant, and $u(x,t)$ is subject to the initial conditions

$$\begin{aligned} u(x,0) &= f(x), \quad -\infty < x < \infty \text{ and} \\ u_t(x,0) &= g(x), \quad -\infty < x < \infty . \end{aligned} \tag{9}$$

The analytical solution as given by d'Alembert as found in [Hilderbrand 1968] is

$$u(x,t) = \frac{1}{2}\left[f(x-ct) + f(x+ct)\right] + \frac{1}{2c}\int_{x-ct}^{x+ct} g(\xi)d\xi . \tag{10}$$

For the purposes of demonstrating the effectiveness of the finite difference methods, I will assume $g(x) = 0$.

3.1 Explicit difference method (EXP)

The standard and most natural difference method for the second order wave equation is

$$\frac{u_j^{n+1} - 2u_j^n + u_j^{n-1}}{(\Delta t)^2} = c^2 \frac{u_{j+1}^n - 2u_j^n + u_{j-1}^n}{(\Delta x)^2} \tag{11}$$

which can be rewritten as

$$u_j^{n+1} = 2\left(1 - r^2\right)u_j^n + r^2\left(u_{j+1}^n + u_{j-1}^n\right) - u_j^{n-1} \tag{12}$$

or

$$u_j^{n+1} = 2u_j^n - u_j^{n-1} + r^2\left(u_{j+1}^n - 2u_j^n + u_{j-1}^n\right). \tag{13}$$

If we let

$$\mathbf{K} = \begin{bmatrix} 2 & -1 & & \\ -1 & \ddots & \ddots & \\ & \ddots & \ddots & \ddots \\ & & \ddots & \\ & & & \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} -1 \\ 0 \\ \vdots \\ 0 \\ -1 \end{bmatrix},$$

then the explicit finite difference scheme can be written in matrix form as

$$\mathbf{u}^{\mathbf{n+1}} = \left(2\mathbf{I} - r^2\mathbf{K}\right)\mathbf{u}^{\mathbf{n}} - \mathbf{u}^{\mathbf{n-1}} - r^2\mathbf{b}\,. \tag{14}$$

The von Neumann necessary condition for stability holds for the explicit difference equation if $r \le 1$. The derivation can be found in [Mitchell 1980], p198-199. This is shown experimentally in a later section.

3.2 Implicit difference method (IMP)

An implicit difference approximation to the second order wave equation is given by [Mitchell 1980] as

$$\begin{aligned} &\left(u_{j+1}^{n+1} - 2u_j^{n+1} + u_{j-1}^{n+1}\right) + 2\left(u_{j+1}^{n} - 2u_j^{n} + u_{j-1}^{n}\right) + \left(u_{j+1}^{n-1} - 2u_j^{n-1} + u_{j-1}^{n-1}\right) \\ &\qquad = \frac{4}{r^2}\left(u_j^{n+1} - 2u_j^{n} + u_j^{n-1}\right) \end{aligned} \tag{15}$$

which can be rewritten as

$$\begin{aligned} &-u_{j+1}^{n+1} + 2\left(1 + \tfrac{2}{r^2}\right)u_j^{n+1} - u_{j-1}^{n+1} \\ &= 2\left(u_{j+1}^{n} - 2\left(1 - \tfrac{2}{r^2}\right)u_j^{n} + u_{j-1}^{n}\right) + \left(u_{j+1}^{n-1} - 2\left(1 + \tfrac{2}{r^2}\right)u_j^{n-1} + u_{j-1}^{n-1}\right). \end{aligned} \tag{16}$$

If we let

$$\mathbf{K} = \begin{bmatrix} 2 & -1 & & \\ -1 & \ddots & \ddots & \\ & \ddots & \ddots & \ddots \\ & & \ddots & \\ & & & \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} -1 \\ 0 \\ \vdots \\ 0 \\ -1 \end{bmatrix},$$

then the implicit finite difference scheme can be written in matrix form as

$$\left(\tfrac{4}{r^2}\mathbf{I} + \mathbf{K}\right)\mathbf{u}^{\mathbf{n+1}} + \mathbf{b} = 2\left(\tfrac{4}{r^2}\mathbf{I} - \mathbf{K}\right)\mathbf{u}^{\mathbf{n}} - 2\mathbf{b} - \left(\tfrac{4}{r^2}\mathbf{I} + \mathbf{K}\right)\mathbf{u}^{\mathbf{n-1}} - \mathbf{b}\,. \tag{17}$$

or

$$\mathbf{u}^{\mathbf{n+1}} = \left(\tfrac{4}{r^2}\mathbf{I} + \mathbf{K}\right)^{-1}\left[2\left(\tfrac{4}{r^2}\mathbf{I} - \mathbf{K}\right)\mathbf{u}^{\mathbf{n}} - \left(\tfrac{4}{r^2}\mathbf{I} + \mathbf{K}\right)\mathbf{u}^{\mathbf{n-1}} - 4\mathbf{b}\right] \tag{18}$$

3.3 Performance of finite difference methods

I will now try the explicit and implicit finite difference methods for the same initial conditions as in equations 5-7. Figure 8 shows the results of the two methods on the delta function initial condition.

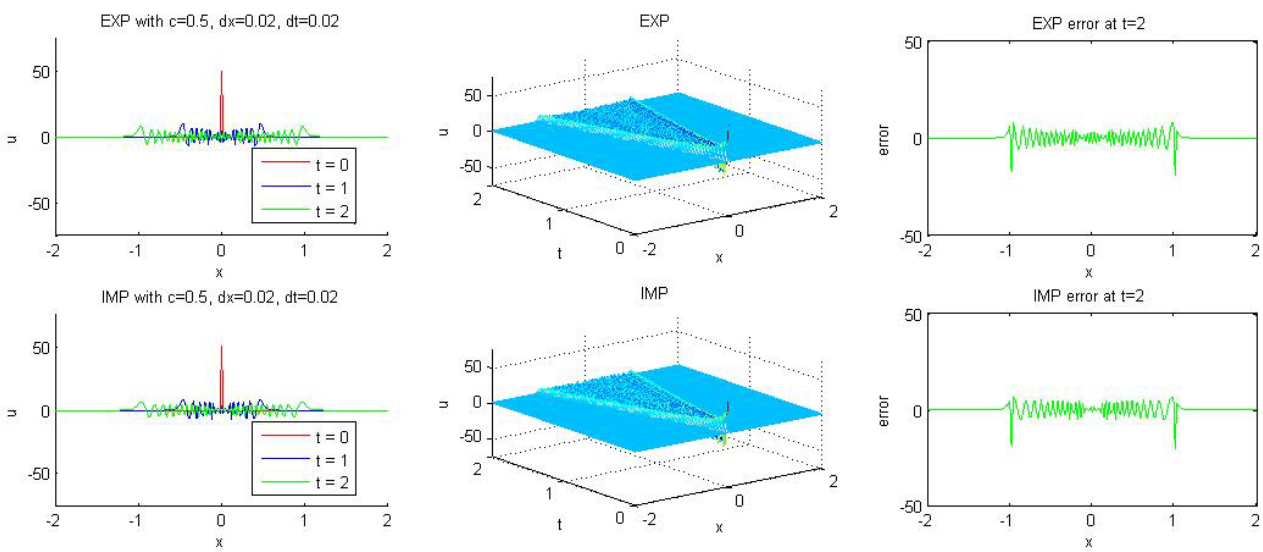


Figure 8. Explicit and implicit finite difference solutions with the delta initial condition

The general performance of both methods is generally about the same; that is, both methods produce similar errors. Now in Figure 9, I will produce the results for the sinusoidal initial condition.

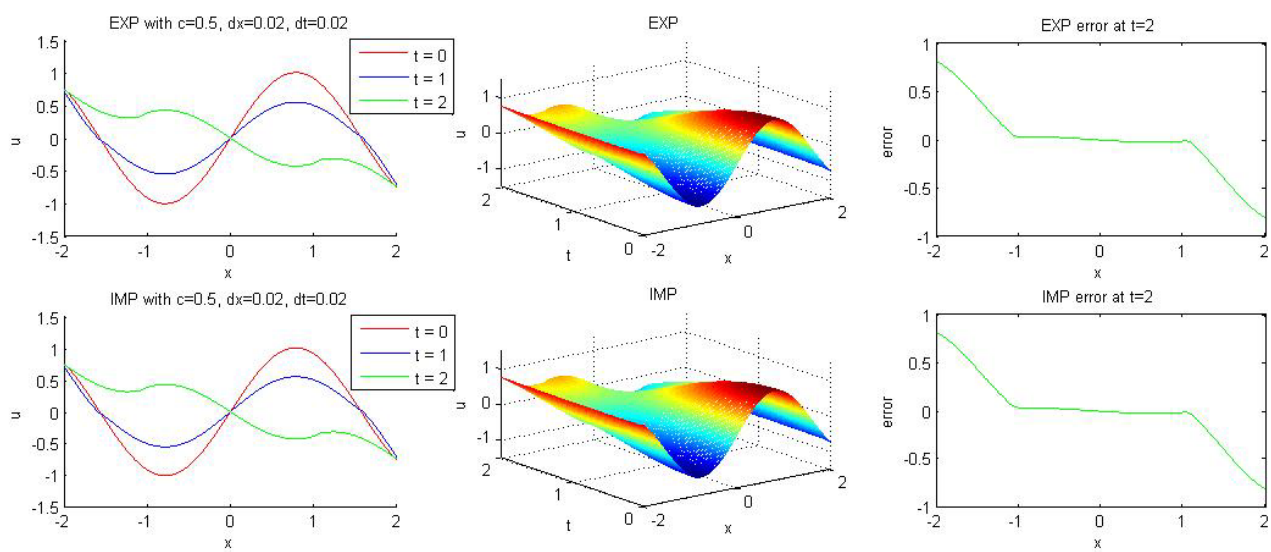


Figure 9. Explicit and implicit finite difference solutions with the sinusoidal initial condition

The finite difference schemes are designed to maintain the same boundary conditions as in the initial condition. The real solution was constructed without maintaining boundary conditions, so that is why there are seemingly large errors near the bounds. Ignoring these errors, the performance of the explicit and implicit finite difference schemes on the smooth initial condition produces very accurate results.

In Figures 10-12, I will use the step function initial condition with varying Δx.

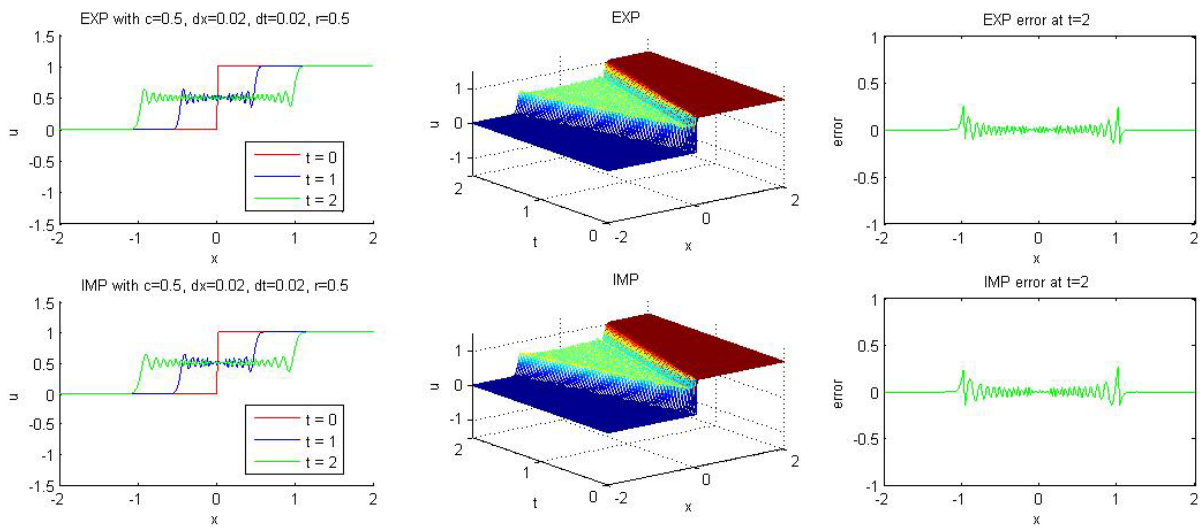


Figure 10. Finite difference solutions with the step initial condition and $r = 0.5$

At small values of *r*, the performances of the explicit and implicit finite difference solutions are very similar. Both are fairly accurate and have similar behaviors.

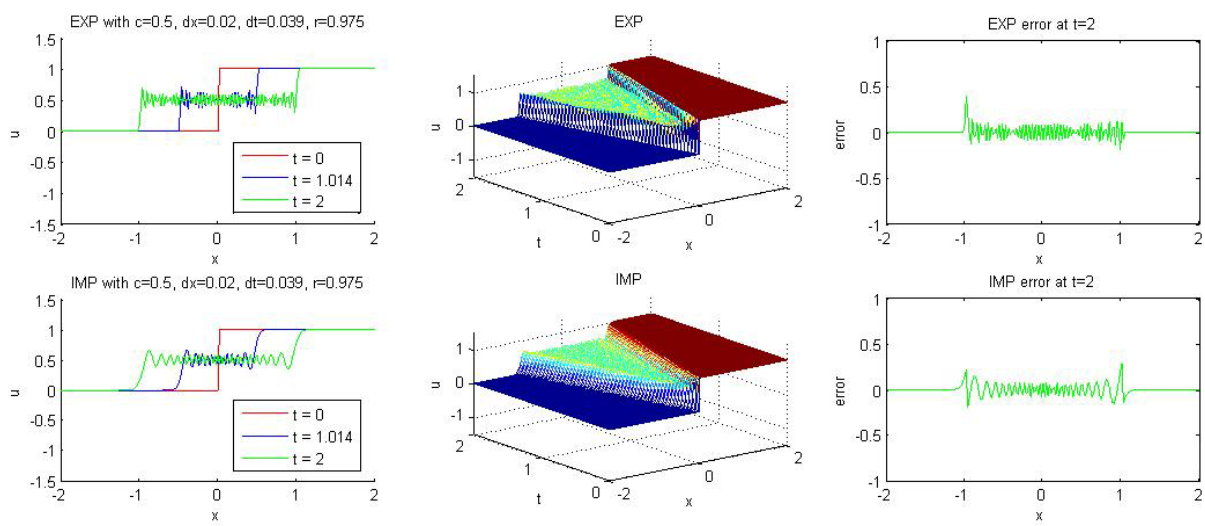


Figure 11. Finite difference solutions with the step initial condition and $r = 0.975$

When *r* approaches 1, the explicit method behaves more vigorously, as it anticipates the instability boarder. The implicit method performs about the same as before.

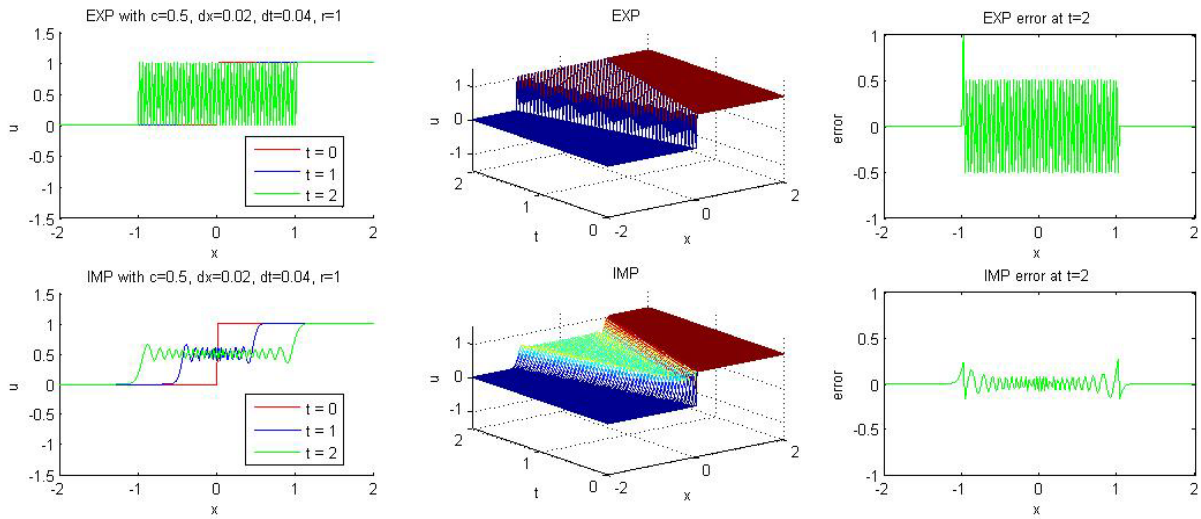


Figure 11. Finite difference solutions with the step initial condition and $r = 1$

When $r = 1$, the explicit solution reaches the instability boarder. Although it does not go unbounded until $r > 1$, it oscillates violently between the two values of the step function. Meanwhile, the implicit method is not affected by the *r* value.

4 Second order 2-D wave equation

The second order wave equation in two-dimensional space is as follows:

$$u_{tt} = c^2 \left(u_{xx} + u_{yy} \right) \tag{19}$$

where *c* is a positive constant, and $u(x, y, t)$ is subject to the initial conditions

$$\begin{aligned} u(x, y, 0) &= f(x, y), \quad -\infty < x, y < \infty \text{ and} \\ u_t(x, y, 0) &= g(x, y), \quad -\infty < x, y < \infty . \end{aligned} \tag{20}$$

For the purposes of demonstrating the effectiveness of the finite difference methods, I will assume $g(x, y) = 0$.

4.1 Explicit difference method

As in the 1-D case, the standard and most natural difference method for the second order wave equation is

$$\frac{u_{i,j}^{n+1} - 2u_{i,j}^{n} + u_{i,j}^{n-1}}{(\Delta t)^2} = c^2 \frac{u_{i+1,j}^{n} - 2u_{i,j}^{n} + u_{i-1,j}^{n}}{(\Delta x)^2} + c^2 \frac{u_{i,j+1}^{n} - 2u_{i,j}^{n} + u_{i,j-1}^{n}}{(\Delta y)^2} . \tag{21}$$

If we let $r_x = c\dfrac{\Delta t}{\Delta x}$ and $r_y = c\dfrac{\Delta t}{\Delta y}$, equation (21) can be rewritten as

$$u_j^{n+1} = \left[2 - 2\left(r_x{}^2 + r_y{}^2\right)\right]u_{i,j}^n + r_x{}^2\left(u_{i+1,j}^n + u_{i-1,j}^n\right) + r_y{}^2\left(u_{i,j+1}^n + u_{i,j-1}^n\right) - u_{i,j}^{n-1}. \tag{22}$$

If we let

$$\mathbf{K} = \begin{bmatrix} 2 & -1 & & \\ -1 & \ddots & \ddots & \\ & \ddots & \ddots & \ddots \\ & & \ddots & \\ & & & \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} -r_x^2 \\ 0 \\ \vdots \\ 0 \\ -r_y^2 \end{bmatrix}, \quad \mathbf{b2D} = \begin{bmatrix} \mathbf{b} \\ 0 \\ \vdots \\ 0 \\ flip(\mathbf{b}) \end{bmatrix}, \tag{23}$$

where the *flip* function simply reverses the order of elements in a vector, and let

$$\mathbf{K2D} = kron\left(r_x^2\mathbf{I}, \mathbf{K}\right) + kron\left(\mathbf{K}, r_y^2\mathbf{I}\right), \tag{24}$$

where *kron* is the Kronecker product, then the explicit finite difference scheme can be written in matrix form as

$$\mathbf{u}^{\mathbf{n+1}} = \left(2\mathbf{I} - \mathbf{K2D}\right)\mathbf{u}^{\mathbf{n}} - \mathbf{u}^{\mathbf{n-1}} - \mathbf{b2D}. \tag{25}$$

The stability of the explicit finite difference method for second order 2-D wave is dictated by the CFL condition

$$r \le \frac{1}{\sqrt{s}}, \tag{26}$$

where s is the space dimensions of the wave equation, and $r = r_x = r_y$. In the 2-D case, the explicit scheme is stable for $r \le \frac{1}{\sqrt{2}}$, or 0.7071.

4.2 Performance of the explicit difference method

There are several different initial conditions which I adapted from [Wakefield 2003], including a cube-like protrusion, a sinusoidal hump, a cylindrical protrusion, a dome, and a step-like protrusion. Since these initial conditions are for general demonstration, I will not go into the details of each of them. Instead, I will just focus on the sinusoid, since it is easy to derive a true solution with which to compare the finite difference solution. The initial condition is

$$u(x, y, 0) = 2\sin\left(\tfrac{\pi}{4}x\right)\sin\left(\tfrac{\pi}{4}y\right), \tag{27}$$

so the analytical solution is

$$u(x, y, t) = 2\sin\left(\tfrac{\pi}{4}x\right)\sin\left(\tfrac{\pi}{4}y\right)\cos\left(\tfrac{\pi}{2}c^2 t\right). \tag{28}$$

The following Figure 11 shows the progression of the explicit difference method, the true solution, and the error over a period of time, sampling at every second. Apologies for having Figure 11 span over multiple pages, but I want to be somewhat complete. The MATLAB animation gives a much better idea of what happens as the scheme is run.

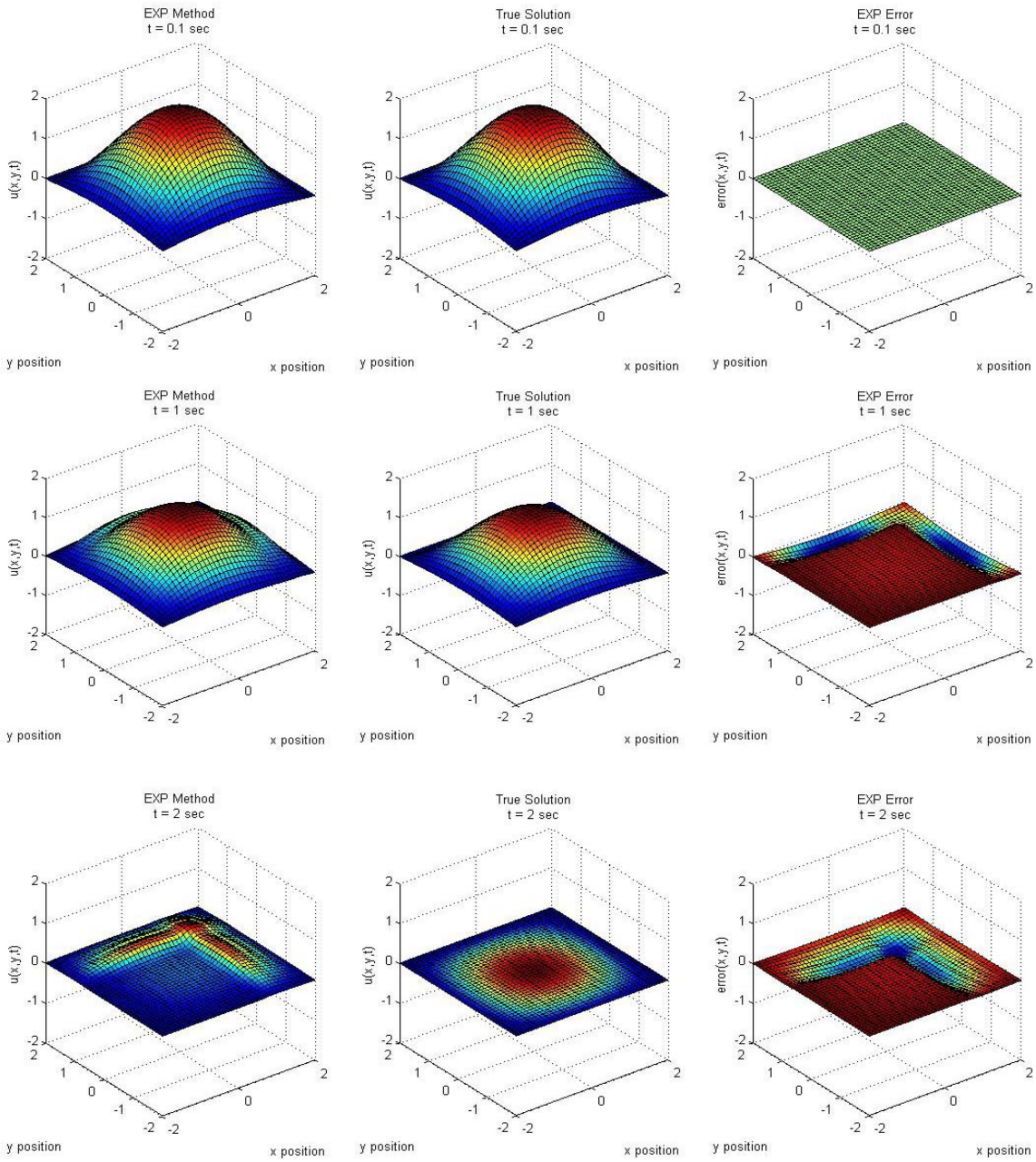


Figure 11 (continued on next page)

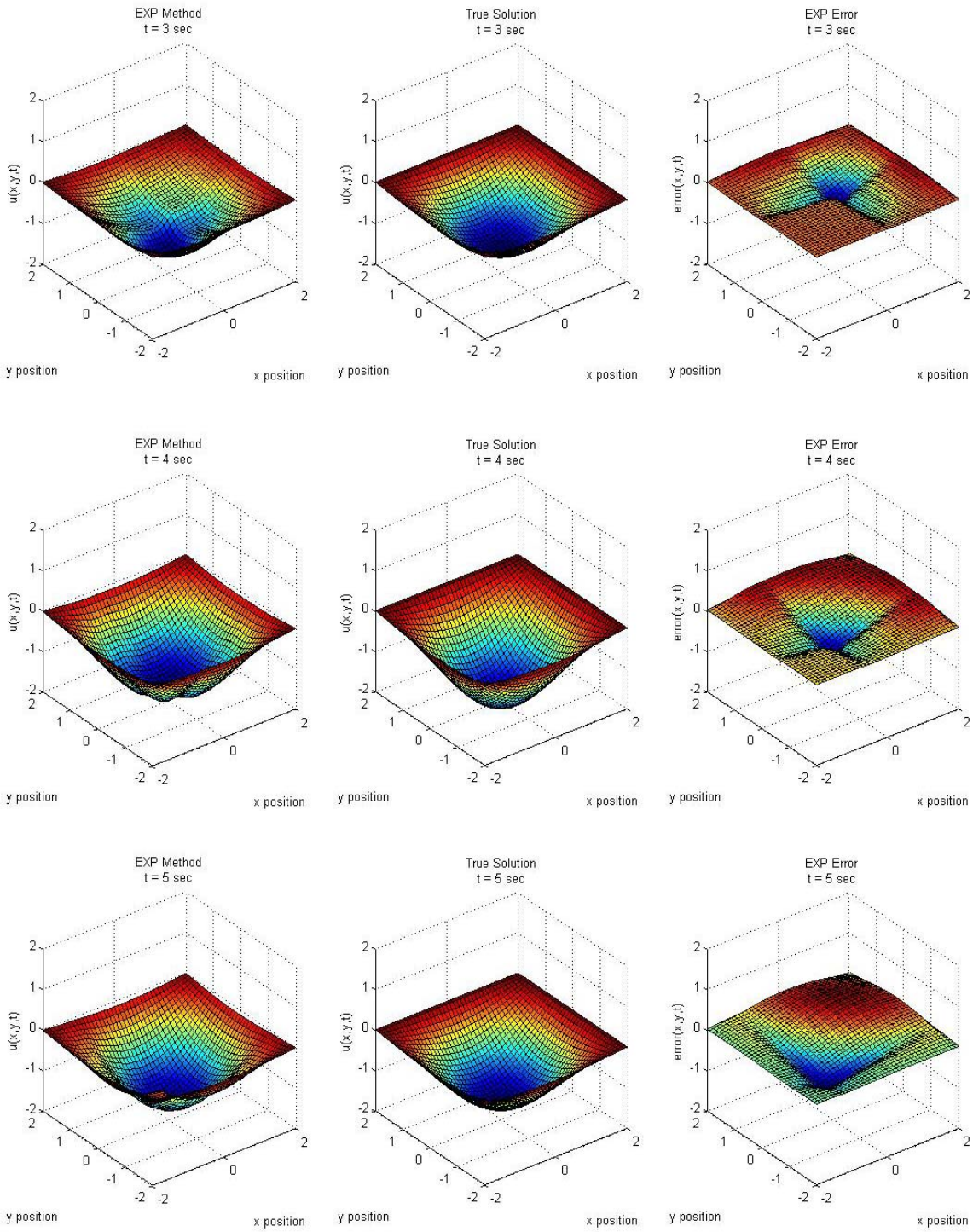


Figure 11 (continued on next page)

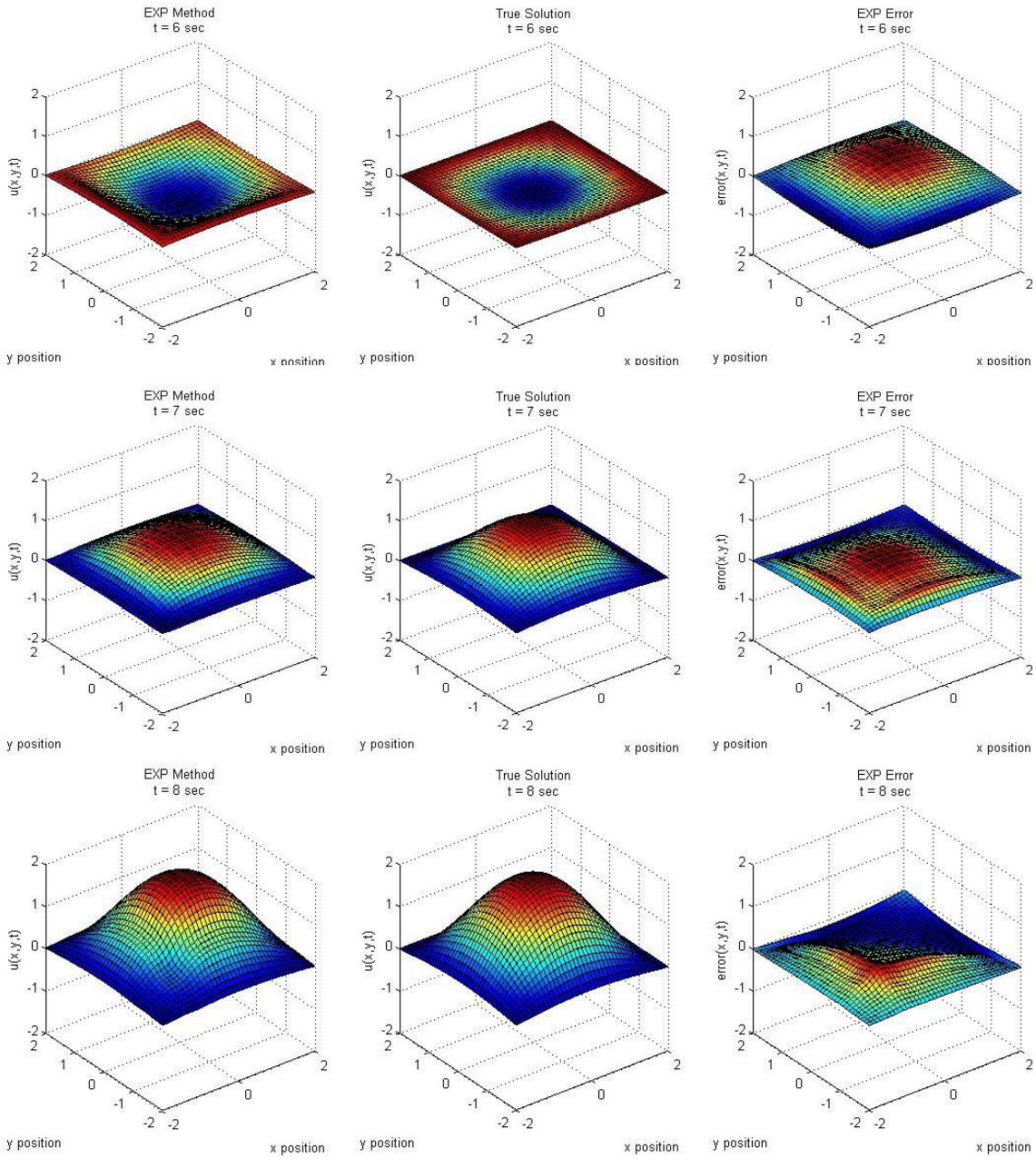


Figure 11. Results with slightly offset sinusoidal initial condition,
$c = 0.7$, $\Delta x = \Delta y = 0.1$, $\Delta t = 0.1$

The explicit scheme performs fairly well, with small error. There is a ripple that flows diagonally across the explicit difference solution, which is the result of the set boundary condition that the explicit difference solution is required to adhere to, but the true solution does not. The reason why the ripple flows diagonally is that the initial condition was diagonally offset by Δx and Δy. Without the offset, there are reflective waves that bounce off of the boundaries on all sides, as exemplified in Figure 12.

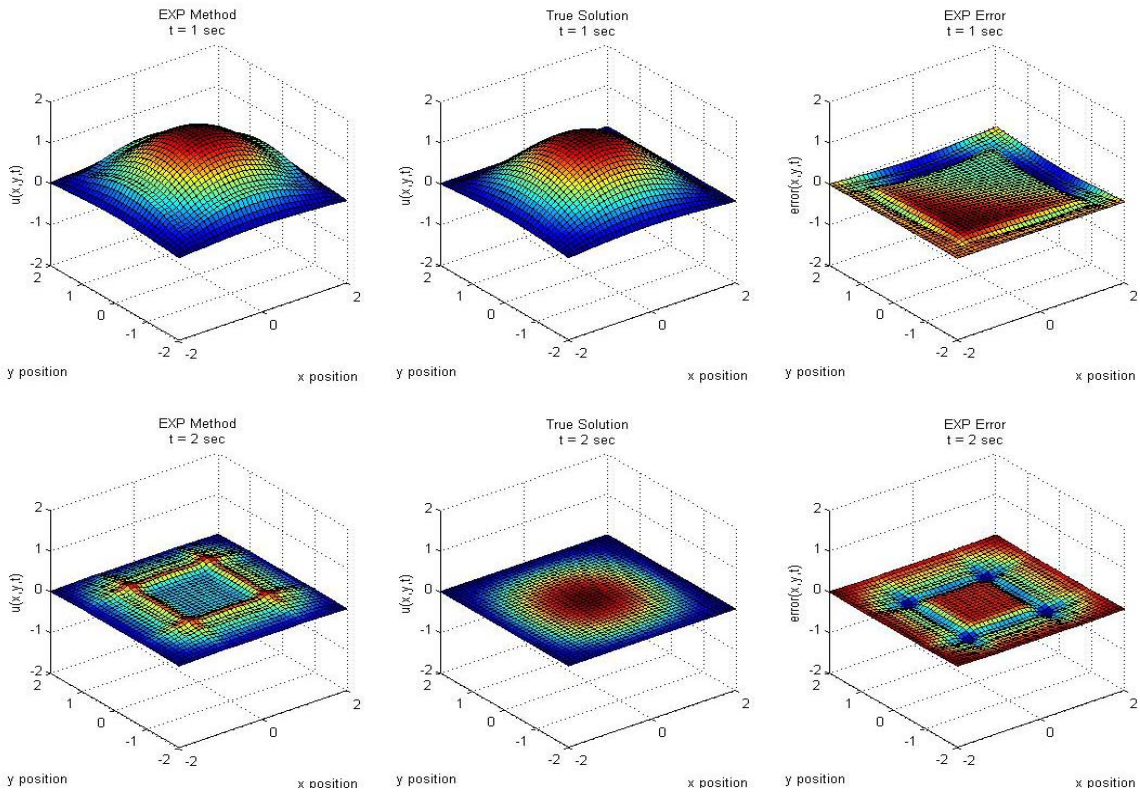


Figure 12. Sample results with centered sinusoidal initial condition,
$c = 0.7$, $\Delta x = \Delta y = 0.1$, $\Delta t = 0.1$

The boundary conditions are set by the **b** vector in the matrix form of the finite difference equations. I have currently set the bounds to match the boundaries on the initial condition. Thus whenever the wave reaches a boundary, there will be a small higher order reflective wave that is returned. This boundary condition was an arbitrary choice and may be representative of certain physical situations but not others. For a good discussion of boundary conditions, see [Threfethen 1994], Ch. 6.

Let's verify that the stability of the explicit finite difference method really is defined by

$$r \le \frac{1}{\sqrt{2}}, \text{ or } 0.7071,$$

as mentioned earlier. Figures 11 and 12 show example results for $r = 0.7$. Now I will try $r = 0.71$, which is outside the stability boundary. Figure 13 shows the results.

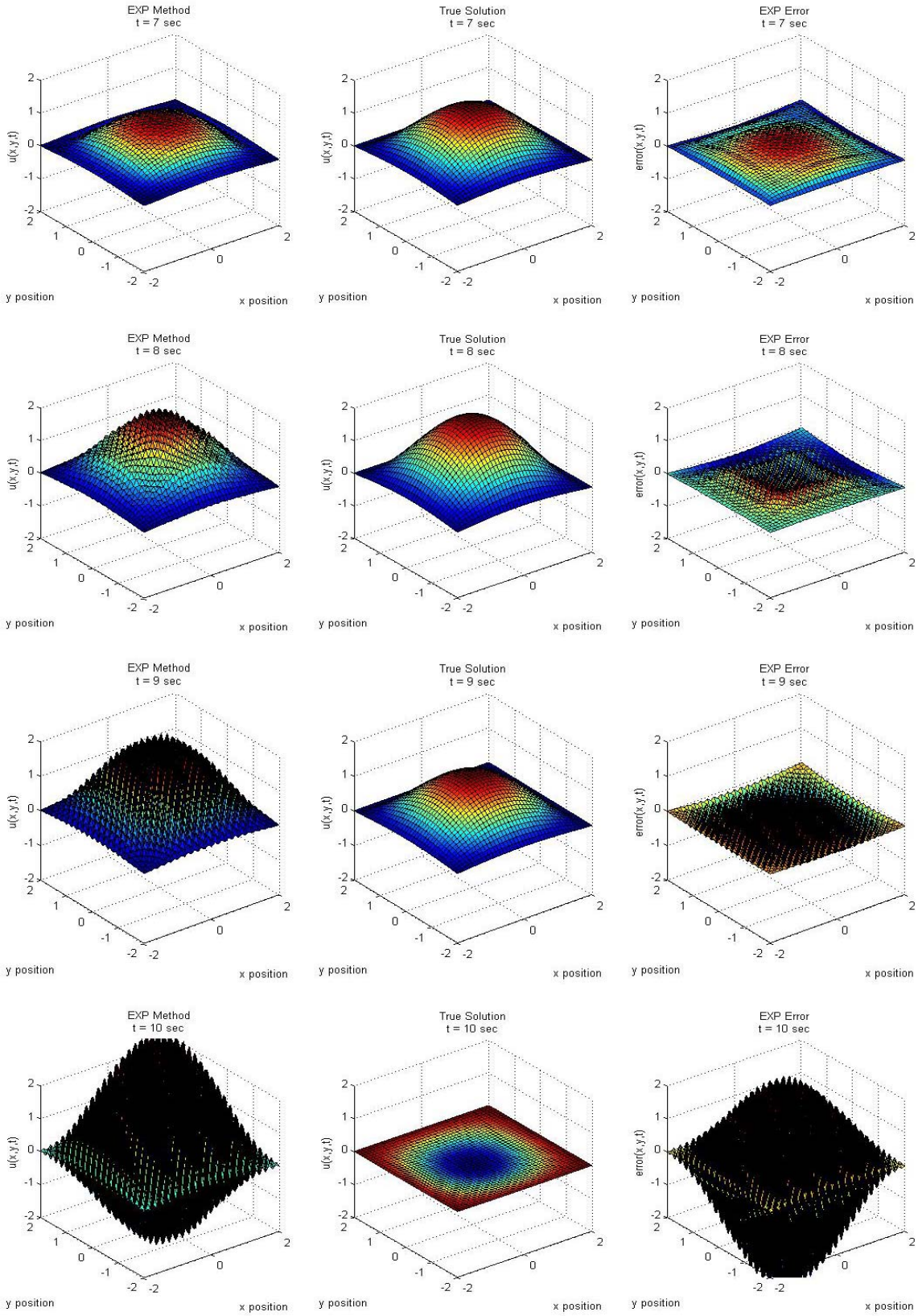


Figure 13. Sample unstable results with $c = 0.71$, $\Delta x = \Delta y = 0.1$, $\Delta t = 0.1$

Sure enough, for $r > \frac{1}{\sqrt{2}}$, the explicit difference method goes unstable. Since stability is limited by *s*, or space dimension, the explicit difference scheme could be problematic for higher spatial dimensions, as the operable range for *r* is reduced more and more.

Now let's see what would happen if I double Δx and Δy while keeping *c* at 0.7. Results are shown in Figure 14.

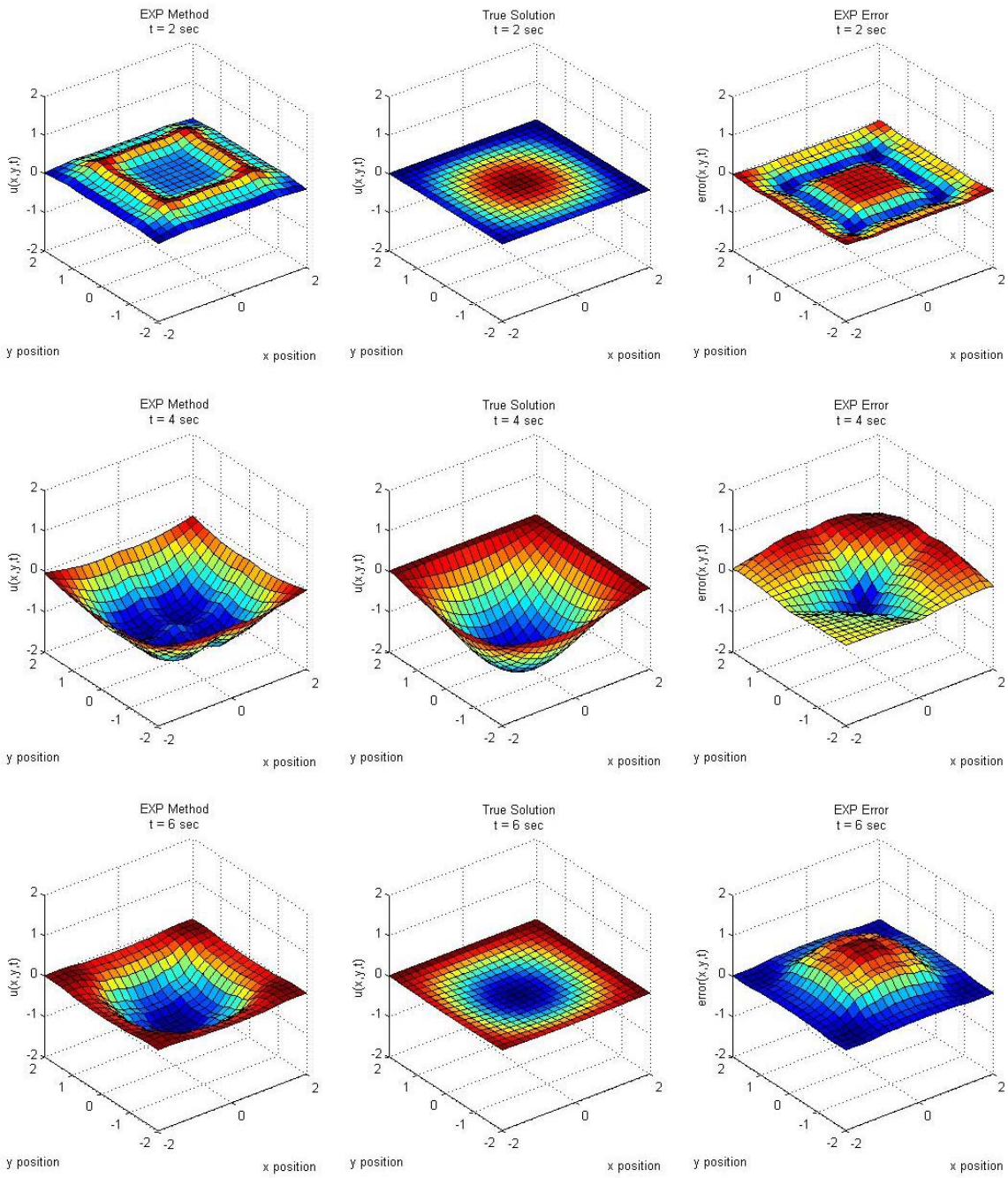


Figure 14. Results with sinusoidal initial condition, $c = 0.7$, $\Delta x = \Delta y = 0.2$, $\Delta t = 0.1$

Doubling Δx and Δy has led to the increase of the error by a factor of about four, which can be verified by looking at the actual error values. Now we can safely believe that the explicit difference algorithm indeed has second order accuracy, as many sources (such as [Buchner]) have mentioned.

Now let's just have some fun with different initial conditions. I have included here one trial with the cube-shaped protrusion initial condition and one with the dome initial condition. Figures 15 and 16 show the results of the explicit difference method on these two initial conditions.

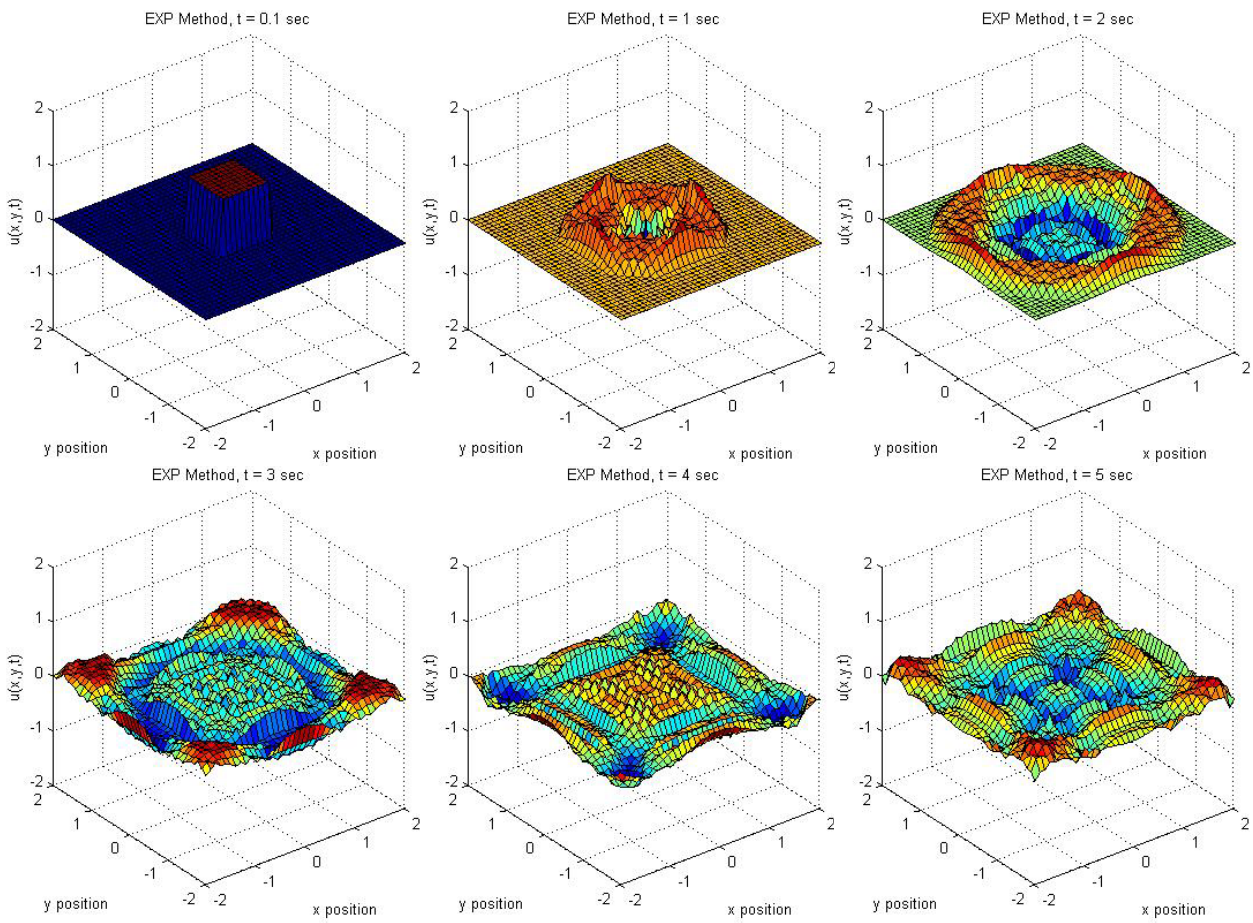


Figure 15. Cube protrusion input condition, $c = 0.7$, $\Delta x = \Delta y = 0.1$, $\Delta t = 0.1$

It is interesting to note how the square shape causes fluctuations in the wave. In Figure 15 at $t = 1$, the corners of the square have been reduced before the sides of the square. Then at $t = 4$, the square shape reforms with reinforcement from the boundary reflections.

Now let's look at the results of a dome initial condition.

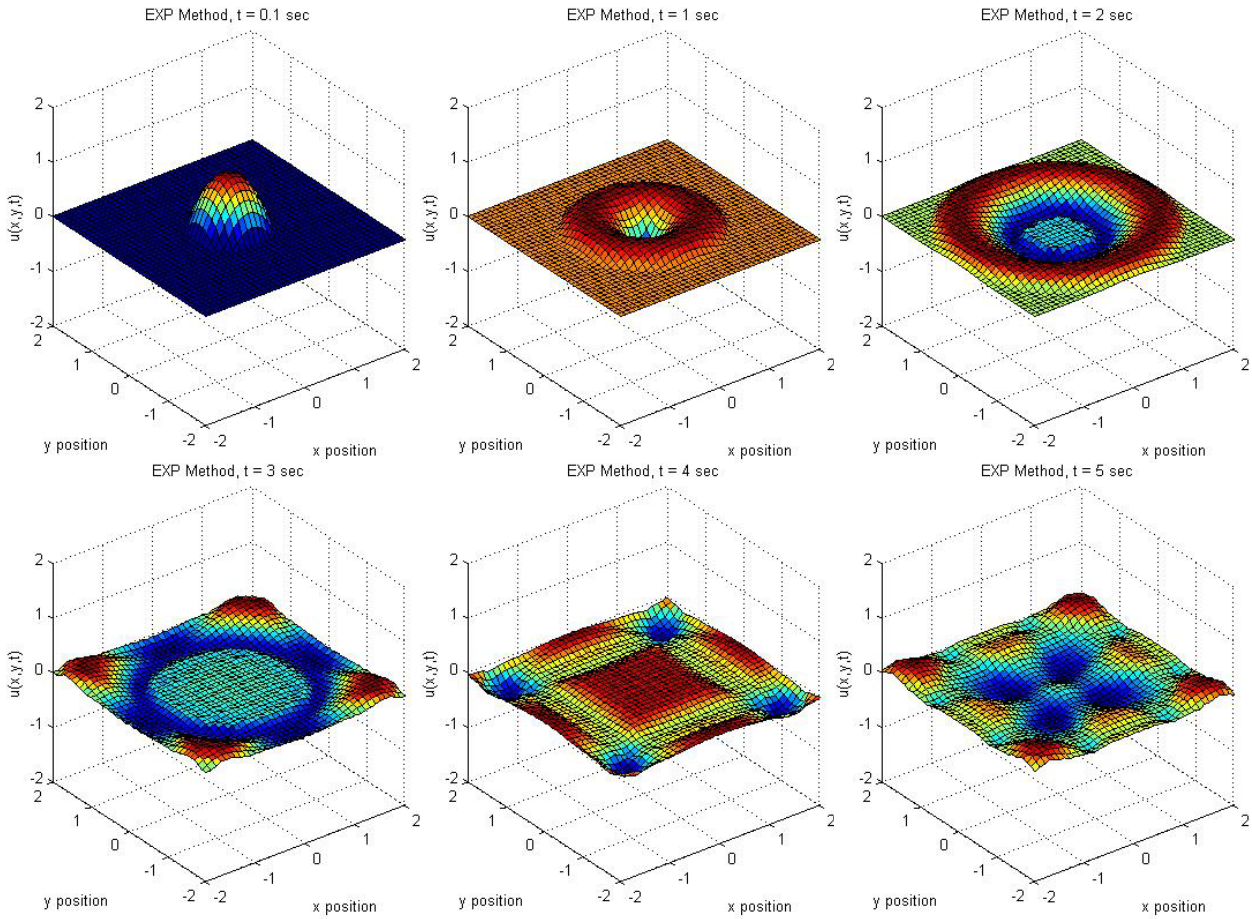


Figure 16. Dome protrusion input condition, $c = 0.7$, $\Delta x = \Delta y = 0.1$, $\Delta t = 0.1$

The first thing to note is that the dome shape is much smoother than the cube protrusion, and thus the finite difference result is also much smoother. The small dome causes a donut ripple outwards, like a water wave. At $t = 3$, the wave is still moving outward in a circular shape, but by $t = 4$, the wave has reached the boundary and builds up a square shaped reflection.

The overall performance of the explicit difference method is quite good for stable values of r and small Δx, Δy, and Δt, with second order accuracy. The stability could be improved by going to an implicit difference method, but often implicit schemes tend to be too complicated to use in a simple manner.

4.3 Implicit difference method

An implicit difference formula is given by [Mitchell 1980] in the form

$$\begin{aligned}&\left[1 + a\left(\delta_x^2 + \delta_y^2\right) + d\delta_x^2\delta_y^2\right]u_{i,j}^{n+1}\\ &= \left[2 - b\left(\delta_x^2 + \delta_y^2\right) - e\delta_x^2\delta_y^2\right]u_{i,j}^{n} - \left[1 + c\left(\delta_x^2 + \delta_y^2\right) + f\delta_x^2\delta_y^2\right]u_{i,j}^{n-1}\end{aligned} \tag{29}$$

where δ^2 is the second order centered difference formula, and *a*, *b*, *c*, *d*, *e*, *f* are coefficients dependent on *r*. There are different ways to set the coefficients. Fairweather and Mitchell suggested the following which produces a 4th order accurate scheme:

$$a = c = \tfrac{1}{12}\left(1 - r^2\right), \quad b = -\tfrac{1}{6}\left(1 + 5r^2\right), \quad e = -\tfrac{1}{72}\left(1 + 10r^2 + r^4\right), \quad d = f = a^2. \qquad (30)$$

Due to the symmetries, equation 29 can be rewritten as

$$\begin{aligned} &\left(1 + a\delta_x^2\right)\left(1 + a\delta_y^2\right)u_{i,j}^{n+1} \\ &= \left[2 - b\left(\delta_x^2 + \delta_y^2\right) - e\delta_x^2\delta_y^2\right]u_{i,j}^{n} - \left(1 + a\delta_x^2\right)\left(1 + a\delta_y^2\right)u_{i,j}^{n-1}. \end{aligned} \qquad (31)$$

The non-linearity in *u* makes this a very difficult scheme to implement. Due to time constraints, I will not be implementing this method, but only mention it here for those interested in the setup of the implicit finite difference method.

5 Conclusion and future work

In this discussion, I have covered first order 1-D, second order 1-D, and second order 2-D wave equations. I compared 9 different finite difference schemes for first order equations, and looked at explicit and implicit schemes for second order equations. In the literature, I have come across discussions of higher order finite difference schemes for second order hyperbolic equations, which might be an interesting expansion for future work. Another area which I have not touched upon is splitting second order equations into multiple simultaneous first order equations. Each of these topics merits more discussion than I have the temporal resources to provide presently, but some interesting work has been done by [Wakefield 2003] on the implementation of splitting methods.

6 References

Buchner, Jorg. http://www.mps.mpg.de/homes/buechner/vorlesungen/Skripte /AccuracyConvergConsist.pdf

Buhler, Oliver and Barnett, Alex. NYU V63.0394 Mathematical Wave Dynamics. 2004. http://www.cims.nyu.edu/vigrenew/waves/

Carter, Larry. UCSD CSE 260 - Introduction to Parallel Computation. 2-D Wave Equation. 10/9/01. http://www-cse.ucsd.edu/users/carter/260/260proj.pdf

Demmel, Jim. UC-Berkeley CS267, 2/27/1996. Sources of Parallelism and Locality in Simulation. http://www.cs.berkeley.edu/~demmel/cs267/lecture17/lecture17.html

Hildebrand, Francis. *Finite-Difference Equations and Simulations*. Prentice-Hall, Inc. Englewood Cliffs, NJ: 1968.

Hood, Alan. 2D Vibrations of a Membrane. 11/06/2000. http://www-solar.mcs.st-and.ac.uk/~alan/MT3601/Fundamentals/node60.html

Mitchell, A.R. and Griffiths, D.F. *The Finite Difference Method in Partial Differential Equations*. John Wiley & Sons, Inc. Chichester: 1980.

Sewell, Granville. *The Numerical Solution of Ordinary and Partial Differential Equations*. John Wiley & Sons, Inc. Hoboken, NJ: 2005.

Shih, Chiang. FAMU-FSU EML 3050 Analytical Tools in Mechanical Engineering. http://www.eng.fsu.edu/~shih/eml3050/lecture%20notes/lecture_notes.htm

Smith, G. D. *Numerical Solution of Partial Differential Equations Finite Difference Methods*. Oxford University Press. New York: 1985.

Smith, Julius O. "Physical Audio Signal Processing: for Virtual Musical Instruments and Digital Audio Effects, December 2005 Edition",2D Mesh and the Wave Equation. http://ccrma.stanford.edu/~jos/pasp/D_Mesh_Wave.html

Strang, Gilbert. *Introduction to Applied Mathematics*. Wellesley-Cambridge Press. Wellesley, MA: 1986.

Trefethen, Lloyd N. Finite Difference and Spectral Methods for Ordinary and Partial Differential Equations. 1994. http://web.comlab.ox.ac.uk/oucl/work/nick.trefethen/pdetext.html

Wakefield, M.A.. Wave2dmovie: 2D Wave Equation - Finite Difference. 4/24/2003. http://www.maths.reading.ac.uk/student/ug_course_support/movies/wave2dmovie.pdf

Zumbusch, Gerhard. Finite Difference Schemes on Sparse Grids for Time Dependent Problems. http://wissrech.iam.uni-bonn.de/research/projects/zumbusch/fd.html