THE UNIVERSITY OF BRITISH COLUMBIA

# MECH 510

# PROGRAMMING ASSIGNMENT 2

Prepared for:

Prof. Carl Ollivier-Gooch

CEME 2050, UBC


Prepared by:

Nicholas Earle

UBC # 21943600


Date:

November 5th, 2018

## OBJECTIVE

The objective of this assignment was to write a program that solved the wave equation numerically. This was to be done initially using a two-stage Runge Kutta (RK2) time advance with a second order upwind flux evaluation. This was first solved using the given initial conditions, for several mesh sizes to compare the error that was produced. The program was then used to experimentally check the stability of this combination of schemes, comparing it with what was calculated analytically in class. Finally, the code was modified such that it could also run the simulation using an Explicit Euler time advance, using either the first or second order upwind schemes. Combining that with the Runge Kutta scheme there was now four combinations total to be compared.

## FORMULATION

First, the equation to be solved was the wave equation in one dimension with the conditions as follows:

$$\frac{\partial T}{\partial t} + u\,\frac{\partial T}{\partial x} = 0 \tag{1}$$

$$0 \le x \le x_{max}; 0 \le t; u = 2$$

This was then solved using $x_{max} = 1$, with the following initial and boundary conditions:

$$T(x,0) = -\sin(2\pi x)$$
$$T(0,t) = \sin(4\pi t) \tag{2}$$

Given these conditions, the exact solution of the wave equation is given equation 3 below. The numerical solution was compared to the exact solution to see the error associated with each of the scheme combinations.

$$T(x,t) = \sin(2\pi(2t - x)) \tag{3}$$

Now, to actually solve this equation numerically, as mentioned above, a two stage Runge Kutta time advancing scheme was used as shown below:

$$w^{n+1} = w^n + \lambda \Delta t w^{(1)}$$
$$w^{(1)} = w^n + \frac{\lambda}{2}\Delta t w^n \tag{4}$$

This was paired with a second order upwind flux scheme taking the difference of the flux through upwind and downwind boundaries of each cell:

$$T^n_{i+\frac{1}{2}} = \frac{3\bar{T}^n_i - \bar{T}^n_{i-1}}{2} \qquad T^n_{i-\frac{1}{2}} = \frac{3\bar{T}^n_{i-1} - \bar{T}^n_{i-2}}{2}$$

$$FI_i^n = -u\frac{3\bar{T}_i^n - 4\bar{T}_{i-1}^n + \bar{T}_{i-2}^n}{2\Delta x} \tag{5}$$

However, as can be easily seen, this causes problems near the boundary cells. Using ghost cells to account for the boundary effects, accommodations need to be made for updating cells 0 and 1. For cell 1, taking the difference of the fluxes, we get that the upwind flux is the boundary, and the downwind flux is taken from the second order scheme for cells 0 and 1:

$$T_{\frac{3}{2}}^n = \frac{3\bar{T}_1^n - \bar{T}_0^n}{2} \qquad T_{\frac{1}{2}}^n = \sin(4\pi t) \tag{6}$$

Using the new value of cell 1, the new ghost cell value can be calculated using the flux at the boundary being the average of the ghost cell and first cell:

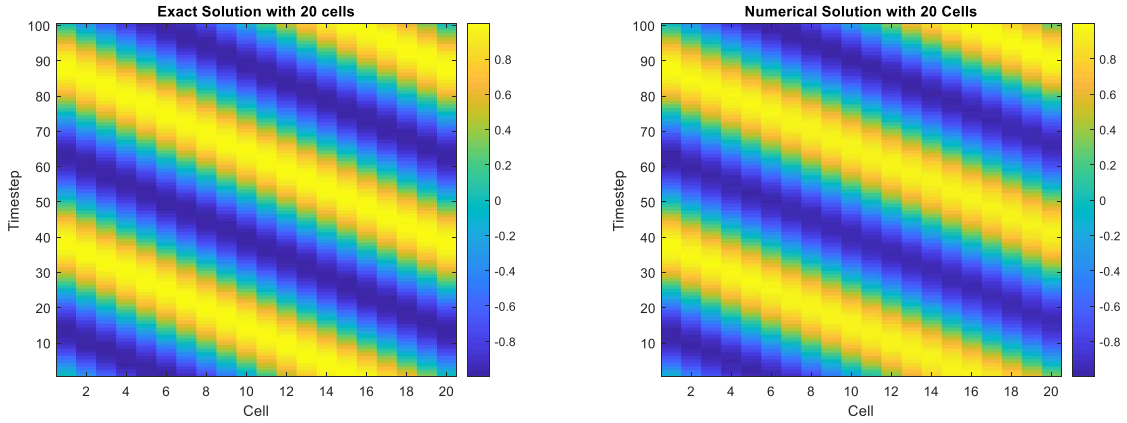
$$T_0^{n+1} = 2\sin\big(4\pi(t+dt)\big) - T_1^{n+1} \tag{7}$$

For calculating the intermediate step of RK2, each equation is only advanced by half a time step instead of the full timestep.

# RESULTS

## PART 1: VERIFICATION

For the first part the solution was calculated to a final time of $t = 1$, using a CFL $\left(\frac{u\Delta t}{\Delta x}\right)$ number of 0.4, for both a mesh of 20 cells and 40 cells. From the exact solution, the expected result is to have a sine wave that is propagated across the domain twice over the time. Figure 1, below, shows images of the solution from the initial condition at the bottom to the final time at the top.



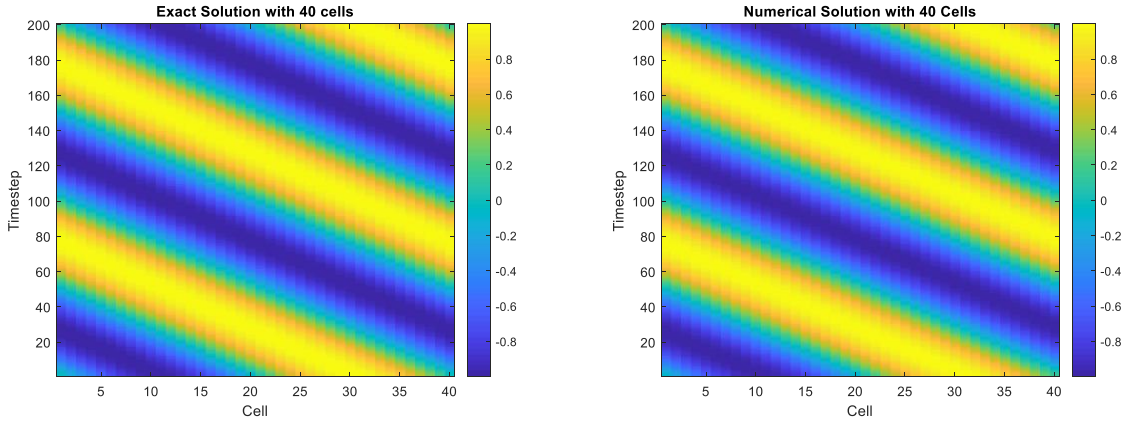



**FIGURE 1: COMPARISON OF EXACT SOLUTION AND NUMERICAL SOLUTION**

While it is difficult to see any discrepancy between the two sets of images above it does show how the solution changes over time. Figure 2, below, shows the exact solution compared to the numerical solutions as well as the error between the numerical solution and the exact solution for both instances. As expected, with more cells the error decreases, and being a second order system, it should decrease by the square of the increase in cells.

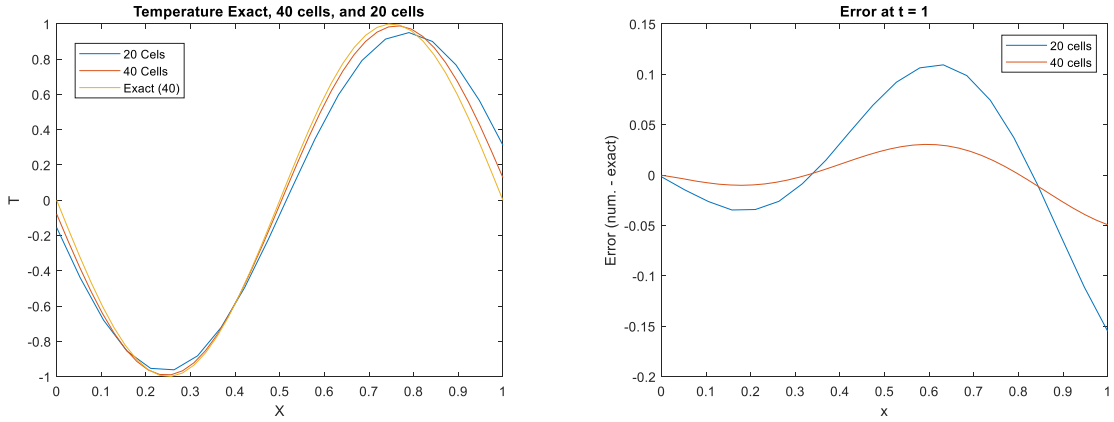



**FIGURE 2: TEMPERATURE AND ERROR AT T = 1**

The L2 Norm was used to quantify the total error at any given timestep. The L2 norms for both 20 and 40 cells can be found in Table 1, below.

**TABLE 1: L2 NORM**

| # of Cells | L2 Norm |
|---|---|
| 20 | 0.0704658 |
| 40 | 0.0207985 |

Looking at both of those values, it is clear that by doubling the number of cells, the error decreases by about a factor of 4 (~3.5). This is to be expected of a second order system, but just to be sure, the number of cells needed to decrease the L2 norm even further to $10^{-3}$ and $10^{-4}$ was then found. Given an L2 Norm of 0.0208 for 40 cells, achieving a L2 Norm of $10^{-3}$, a decrease of the L2 Norm

for 40 cells by a factor of about 21 would be needed. So, taking the square root of 21 and multiplying that by the number of cells should be a good starting point to find exactly how many cells we need. That being said, the search will start with 183 cells. Table 2, below, shows the results of the search.

**TABLE 2: L2 NORM FOR $10^{-3}$**

| # of Cells | L2 Norm |
|---|---|
| 40 | 0.0207985 |
| 183 | 0.00108167 |
| 190 | 0.00100423 |
| 191 | 0.000996928 |

So, to decrease the L2 Norm to less than $10^{-3}$, 191 cells must be used, which is not far off the initial guess of 183. Repeating this exercise to find where the L2 Norm reaches $10^{-4}$, starting with 191 cells, this must now be reduced 10 times so the next guess is the square root of 10 multiplied by 192, that is 604 cells.

**TABLE 3: L2 NORM FOR $10^{-4}$**

| # of Cells | L2 Norm |
|---|---|
| 191 | 9.94421e-04 |
| 604 | 1.00699e-04 |
| 605 | 1.00368e-04 |
| 606 | 1.00038e-04 |
| 607 | 9.98056e-05 |

So again, to decrease the L2 norm to less than $10^{-4}$, 607 cells must be used, which again is quite close to the initial guess of 604, found simply by multiplying the square root of the amount to be decreased. Given this information, it is indeed safe to say that this scheme is second order accurate.

## PART 2: STABILITY

The second part is to test the program for stability. This was done setting $x_{max} = 20$ and using the following initial conditions:

$$T(x,0) = \begin{cases} -x & 0 \le x \le 1 \\ 0 & elsewhere \end{cases} \tag{8}$$

The solution was then calculated using 500 cells up to a final time of $t = 8$. Next instead of setting a constant CFL number, the timestep was altered to find exactly where the scheme becomes unstable. For the RK2 scheme using a second order upwind flux integral, the maximum CFL number based on a stability analysis is 0.5, given $u = 2$ and $\Delta x = 0.04$, that gives the theoretical maximum timestep as $\Delta t = 0.01$, this is a good starting point. Figure 3, below, shows the solution for a number of different timesteps.

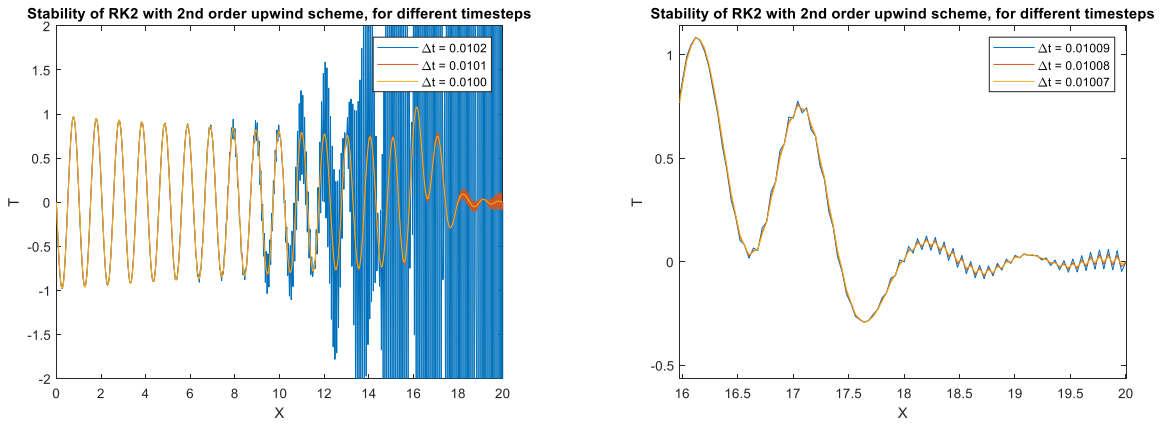



**FIGURE 3: STABILITY FOR DIFFERENT TIMESTEPS**

The graph on the left shows the solution for $\Delta t = 0.01$, 0.0101, and 0.0102. Using the simplest definition of stability as whether or not the solution "blows up," it is clear that the theoretical maximum timestep is stable, while for $\Delta t = 0.0101$ the solution is just on the edge of blowing up and for $\Delta t = 0.0102$, the solution has blown up. To get even closer to where exactly the solution starts to go awry, the graph on the right shows the solution for $\Delta t = 0.01007$, 0.01008, 0.01009, zoomed in to see the discrepancy just at the end. As can be seen the solution is still stable for $\Delta t = 0.01007$ but is on the verge of blowing up at $\Delta t = 0.01009$. Table 4, below, shows the CFL numbers for each and whether they are stable or not.

**TABLE 4: CFL NUMBER AND STABILITY FOR DIFFERENT TIMESTEPS**

| $\Delta t$ | CFL | Stable/Unstable |
|---|---|---|
| 0.01 | 0.5 | Stable |
| 0.01007 | 0.5035 | Stable |
| 0.01008 | 0.504 | Stable |
| 0.01009 | 0.5045 | Edge of Stability |
| 0.0101 | 0.505 | Edge of Stability |
| 0.0102 | 0.51 | Definitely Unstable |

## PART 3: MORE SCHEME COMBINATIONS

Returning to the initial conditions and boundary conditions of part 1, the code was now modified such that it can now run both a 2 stage Runge Kutta time advance as well as an Explicit Euler time advance, shown in Equation 9, below, with both of a first order (below) and second order space discretization.

$$w^{n+1} = w^n + \lambda \Delta t w^n \tag{9}$$

$$T_i^{n+1} = \frac{\bar{T}_i^n - \bar{T}_{i-1}^n}{\Delta x} \tag{10}$$

To compare the accuracy of each combination of schemes, they were run using 80 control volumes, from $t = 0$ to $t = 1$, using 80% of the maximum CFL number for that scheme. For the schemes

using first order approximation that is $CFL_{max} = 1$, and for second order $CFL_{max} = \frac{1}{2}$. The derivation for these can be found in Appendix 1. Figure 4, below, shows the $L_1$ norm and the $L_\infty$ norm for each of the schemes.

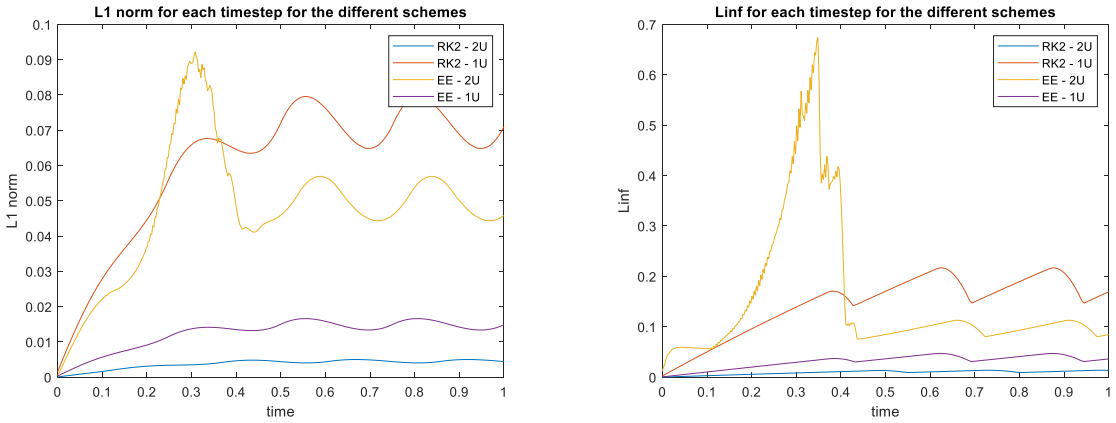



**FIGURE 4: L1 AND LINF NORMS**

Looking at the plots, it becomes quite clear just how much more accurate the pure second order scheme is compared to any of the first order schemes. But more interestingly, using a space or time scheme that doesn't match the order of the other seems to increase the error by quite a lot. Looking at the error for the Explicit Euler scheme using the second order space discretization, it seems that there is a bug present in the code, however running past a certain point it does level itself out to a solution much more consistent with the other schemes. Next, to see the actual order of accuracy of each of the schemes, Figure 5, below, shows the convergence of the schemes by running each on finer meshes.

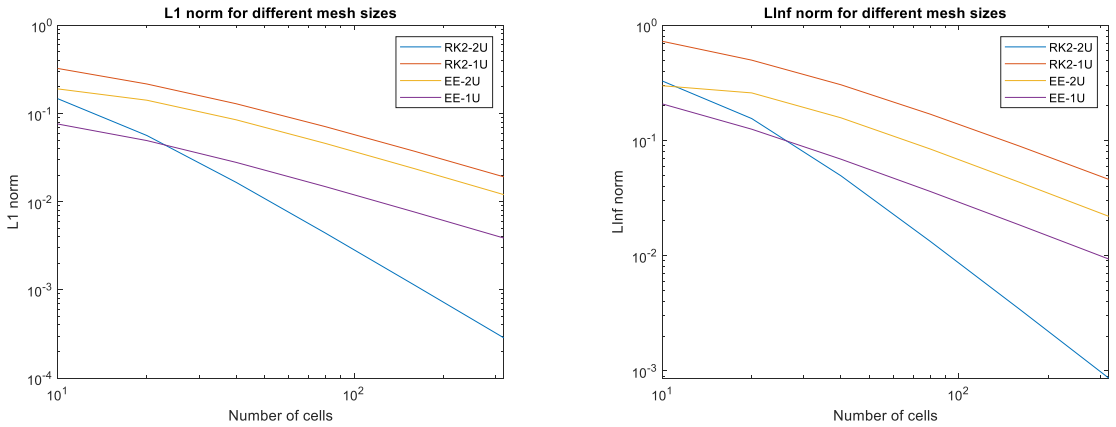



**FIGURE 5: L1 AND LINF NORMS FOR INCREASING MESH FINENESS**

Looking at the plots above, this confirms the suspicion that any combination of space and time schemes take the total order of whichever scheme is lower. That is, it doesn't matter if an Explicit Euler scheme is used with a first or second order space discretization, they will both be first order accurate as shown by the slope of the lines on the log plots above. As expected, the Explicit Euler schemes and the Runge Kutta scheme with the first order space discretization have a slope of -1,

while the pure two stage Runge Kutta scheme has a slope of -2. Looking at the overall error of the schemes above, it can be said however, that if a first order scheme is to be used, an Explicit Euler scheme with a first order space discretization produces lower overall error than any combination of higher and lower order schemes.

## CONCLUSION

To conclude, this assignment was an excellent way to see how a two stage Runge Kutta time advance scheme is implemented and compare it to the first order Explicit Euler method. As expected RK2, was much more accurate than EE1. It was also shown that using any combination of space and time discretization schemes, the total order of the method is that of the lowest order of the space or time scheme. Furthermore, by using different ordered schemes, the error actually increases compared to a scheme of the same orders. To take this assignment one step further, it would be interesting to see if a three and four stage Runge Kutta follow the same patterns. Unfortunately, that could not be accomplished on time, but perhaps can be completed some time in the future.

# APPENDIX