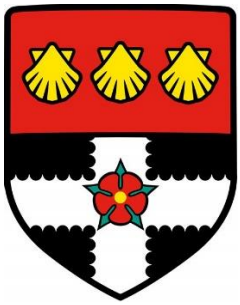
A comparison of Implicit Time-stepping and Semi-Lagrangian for Long Time-Step Advection

UNIVERSITY OF READING

Department of Meteorology



Vitalina Yevgenia Yuriyivna Lopatyuk

Submitted in partial fulfilment of the requirements of the degree of Batchelor of science, Meteorology at the University of Reading.

February 2020

Abstract

Advection is a major process to consider when modelling the evolution of the atmosphere through Numerical Weather Prediction. To solve the advection equation, there are three main types of numerical schemes used: Method of lines, Semi-Lagrangian and flux-form Semi-Lagrangian. To solve the advection equation, there are certain properties which are desirable; such as conservation, stability, accuracy and computational cost. Each of the numerical schemes have their own properties, therefore some perform better than others when solving for advection.

The most popular advection scheme currently used in Numerical Weather Prediction is the Semi-Lagrangian scheme, and even though it is a stable and accurate scheme, it is non conservative. This scheme was compared to two Eulerian schemes: Implicit Crank-Nicolson Centred-in-Space and an Explicit Centred-in-Time Centred-in-Space, with an objective to see whether the Eulerian schemes could be used instead of the Semi-Lagrangian scheme, to ensure conservation in the atmospheric models while keeping computational costs low. When using large time steps, the CTCS scheme had a restriction: it is stable only for Courant numbers less than or equal to one, however in comparison to the CNCS scheme, it had smaller errors when the courant number approached to one.

Comparing the Semi-Lagrangian scheme to the conservative method of lines Centred-in-Time Centred-in-Space and Crank-Nicolson Centred-in-Space schemes with large time-steps found, that when using a fine resolution, the method of lines schemes are as accurate as the Semi-Lagrangian scheme with a coarse resolution. To conclude whether the CTCS scheme can be used instead of the Semi-Lagrangian scheme further research has to be made.

1. Introduction

1.1 Scientific Background and Motivation

The Navier-Stokes equations for a compressible, rotating atmosphere, describe the evolution of the velocity of a fluid. Navier-Stokes equations together with mass continuity equations, ideal gas law and the first law of thermodynamics, are prognostic equations used to represent atmospheric dynamics. Together with the initial weather observations, the Navier Stokes equations are used in the modelling of the evolution of the atmosphere - this is called Numerical Weather Prediction (NWP). (Bauer *et al*, 2015)

Advection is a term included in the Navier-Stokes equation, and is defined by Park *et al* (2017) as "The transfer of properties (such as heat or moisture) or substances (such as air or water pollutants) by the flow of a current of water or air". Advection in the atmosphere is a major process (Ullrich. P. A. and M. R. Norman, 2014) and is a large term in the fluid dynamics of the atmosphere when scaling the prognostic equations (Hoskins *et al*, 2014). The following differential equation is used to describe advection with no sources or sinks in its simplest form, in one dimension:

$$\frac{\partial \varphi}{\partial t} = -u \frac{\partial \varphi}{\partial x} \tag{1.1}$$

Here φ is a scalar quantity such as temperature or a chemical component, and u is the velocity at which the scalar quantity φ is being advected. On the left-hand side of (1.1) $\frac{\partial \varphi}{\partial t}$ is the rate of change with respect to time at a given position of the scalar quantity φ. The right-hand side of (1.1) is the advection term, and $\frac{\partial \varphi}{\partial x}$ is the gradient of the scalar quantity in the x-direction at a given time.

Numerical schemes are used to solve time-dependant differential equations, such as the advection equation shown in (1.1), which is a simple wave equation (Durran, 2010). This dissertation will focus on the analysis and testing numerical schemes using grid-point methods. The three most used categories of numerical schemes are: flux-form Semi-Lagrangian, Semi-Lagrangian and method of lines Eulerian schemes.

The current atmospheric models used by world leading forecast centres, the UK Met Office and ECMWF are based on Semi-Lagrangian advection (Davies *et al*, 2005). Although Semi-

Lagrangian advection is stable and accurate for long time steps, the major limitation of this scheme is that it does not enforce conservation of advected quantities; Davies *et al* (2005) talk about how the conservation of mass in the absence of sources and sinks is critical for climate simulations. Therefore, a comparison of other advection schemes with Semi-Lagrangian is required, to identify whether another advection scheme would be better suited for atmospheric modelling.

1.2 Analysis of advection schemes

To asses whether a numerical scheme is useful, there are certain properties which are looked at, as listed by Chen *et al* (2017):

1. Inherent local conservation of the advected quantity.
2. Stability in the presence of large Courant numbers.
3. Accuracy in the presence of large Courant numbers.
4. High-order accuracy.
5. Low computational cost, good parallel scaling and multitracer efficiency.
6. Low phase and dispersion errors.
7. Low diffusion errors.
8. Boundedness, monotonicity, positivity and maintaining correlations between multiple advected tracers.

Conservation- Thuburn (2011), talks about how a numerical scheme is said to be conservative when the flux of φ out of the first grid box is equal to the flux of φ into the next grid box. The advection schemes tested in this dissertation are solving advection with no sources or sinks, shown in (1.1), therefore the schemes tested should conserve mass. When the advection scheme isn't mass conservative in numerical weather prediction, problems may arise; Johnson (1997) wrote about how suspicious sources of entropy in numerical models are linked with the conservation of potential temperature. However, historically conservation of a scheme has been sacrificed for its efficiency, example of this is the widely used (non-conservative) Semi-Lagrangian advection scheme (Thuburn, 2011).

Computational Cost – Timestep restrictions of Eulerian finite difference schemes are based on the Courant number:

$$c = \frac{u\Delta t}{\Delta x} \tag{1.2}$$

Therefore, from equation (1.3) the Courant number depends on the size of the time stepping. The Fredrichs-Lewy condition (CFL) mentioned before, requires the domain of dependence to include the domain of dependence of the Original PDE (Durran 1999); this condition depends on the Courant number of the scheme. Therefore, when testing the schemes with large time steps it is important that the CFL condition is still met so the scheme will still be stable. The computational cost of the numerical scheme is directly related to the size of the time step (Donea *et al*, 2000), as a result, using as large of a timestep as possible will reduce the computational cost of a scheme. For example, using a time step of an hour will require a smaller computational cost than using a time step of 30 minutes to make a weather forecast for the next 24 hours. It is important to keep computational efficiency high to keep computational costs low. It is therefore essential for the scheme to be stable and accurate in the presence of a large Courant number, if the cost is to be low.

Accuracy and the rate of convergence - Error norms are used to measure the accuracy of the numerical schemes/ the rate of convergence to the analytical solution. Diffusion and dispersion error can affect the accuracy of the numerical scheme.

- Dispersions error (also referred to as phase error) results from the difference in the numerical and analytical solution and can cause oscillations in the numerical solutions.
- Diffusion affects the smaller details of the numerical solution because of the dampening of the numerical solution. The higher the diffusion, the less detail is seen.

Boundedness - A numerical solution is said to be bounded where the maximum value of φ and the minimum value of φ of the numerical scheme do not lie outside the maximum and minimum values in the analytical solution. Durran (2010) explores the importance of boundedness in numerical schemes used for numerical weather production and talk about how a potential spike in the calculated water vapour can exceed the saturation mixing ratio, resulting in prediction of surplus cloud. Boundedness is also linked to Godunov's theorem, where unwanted oscillations will occur when a scheme's solutions has high gradients or contains discontinuities (Hirsch 2007).

1.3 Types of advection Schemes

There have been many years of research in developing and testing different numerical methods to solve the advection equation (also referred to as the transport equation in Durran 2010 and conservation law in LeVeque, R. J., 2005). As mentioned previously, there are three main types of advection schemes: method of lines, Semi-Lagrangian and flux-form Semi-Lagrangian. This paper will predominantly focus on the analysis of method of lines schemes and semi-Lagrangian schemes.

Method of lines schemes use the advection equation in its Eulerian form (1.1), and separately discretise the space and time derivatives, leading to either an explicit or an implicit method of lines scheme.

- Implicit method of lines schemes calculates φ using φ in the next time step, therefore, there are multiple unknowns in the calculation. To work out the multiple unknowns a matrix equation is set up, this means that it requires more computational power than its respective explicit methods (Weller *et al*., 2013), however the implicit methods allow larger time-steps compared to the explicit methods. Another drawback of the implicit methods was mentioned by Thuburn and Cotter (2012)- how Implicit methods have large phase errors.

- Explicit methods calculate φ entirely from φ at previous time steps. The main drawback of explicit Eulerian methods is that not all time steps used in the scheme satisfy the Courant-Fredrichs-Lewy condition (CFL); Durran (1999) describes the CFL as "the numerical domain of dependence of a finite- difference scheme includes the domain of dependence of the associated partial differential equation". Not satisfying the CFL condition means that the scheme isn't stable for all Courant Numbers (talked about in the next section).

Semi-Lagrangian schemes discretise the advection equation in its Lagrangian form, which is on the left-hand side of the equation bellow:

$$\frac{D\varphi}{Dt} = \frac{\partial \varphi}{\partial t} + u\frac{\partial \varphi}{\partial x} = 0 \qquad (1.3)$$

Semi-Lagrangian schemes focus on following the movement of the flow's specific air parcel along its trajectory; following air parcels as discretization units, tracking them along their trajectory (Dong *et al*, 2014). Therefore φ at the current point is calculated using its departure

point. φ's departure point doesn't always align with the grid point, interpolation is used. Larger polynomial interpolation results in a higher order of accuracy of the Semi-Lagrangian schemes than using linear interpolation (Durran, 1999). Leonard (2002) mentions how this method of calculating φ ensures that the Semi-Lagrangian methods don't have Courant number restrictions, however the use of interpolation results in the scheme being non-conservative. Compared to explicit method of lines schemes, Semi-Lagrangian schemes allow longer time steps because of the unrestricted Courant number, using large time steps means that the computational time is reduced when used in numerical forecasting. Furthermore, Semi-Lagrangian offers minimal phase error and handles sharp discontinuities (Ramachandran and Machenhauer, 2001).

Finally, flux-form Semi-Lagrangian (FFSL) methods are finite volume methods (not all finite volume methods are flux-form Semi-Lagrangian), this is where φ is calculated using influx and outflux of each grid box (Durran, 1999). An example of a FFSL scheme is the PPM scheme which uses piecewise parabolic interpolation (Xiaocong. W. *et al*, 2013). Xiaocong *et al* (2013) concluded that the FFSL methods have less dissipation and dispersion compared to other schemes, is conservative, is monotonicity preserving and has a high order of accuracy. However, the FFSL scheme isn't efficient when applied to spherical geometry (Lin and Rood 1996).

1.4 Objective

Limited research has been done comparing implicit time-stepping with Semi- Lagrangian advection. Chen *et al* (2017) compared a multi-dimensional implicit scheme with semi-Lagrangian advection. However, the comparison was made between schemes with different spatial discretisation, making it difficult to fairly compare the conservation, stability, order of accuracy, boundedness and computational cost of the schemes.

The objectives in this dissertation are on comparing implicit and explicit Eulerian schemes with the same spatial discretisation, with Semi-Lagrangian advection. The use of Semi-Lagrangian advection with cubic interpolation will improve the accuracy of the scheme. The same order of accuracy should be used for the Eulerian schemes for a fair comparison. When comparing the advection schemes, the interest lies in analysing the schemes using long time steps, as this will keep the computational costs low.

2. Methods

2.1 Centred-in-time Centred-in-space (CTCS) scheme

Second-order accurate centred-in-time centred-in-space (CTCS) advection scheme is an example of a method of lines schemes. It works by solving the advection equation (1.1), through using centred time and space differences. To solve the finite differences, time steps, Δt, and space steps, Δx,are used. These steps are used as discrete amounts, where $t_n = n\Delta t$ and $x_j = j\Delta x$, note that n is referred to as the n'th time step. The time derivative, $\frac{\partial \varphi}{\partial t}$, is approximated by a centred difference at x_j, t_n, using $\varphi_j^{(n-1)}$and $\varphi_j^{(n+1)}$(Weller 2019):

$$\frac{\partial \varphi_j^{(n)}}{\partial t} = \frac{\varphi_j^{(n+1)} - \varphi_j^{(n-1)}}{2\Delta t} \tag{2.01}$$

The space derivative, $\frac{\partial \varphi}{\partial x}$, is also approximated using a centred difference at x_j, t_n:

$$\frac{\partial \varphi_j^{(n)}}{\partial x} = \frac{\varphi_{j+1}^{(n)} - \varphi_{j-1}^{(n)}}{2\Delta x} \tag{2.02}$$

(2.01) and (2.02) are then substituted into the advection equation (1.1) at x_j, t_n, the equation is then rearranged to get $\varphi_j^{(n+1)}$ on the left-hand-side, and the Courant number (1.3) is substituted in to get the equation used in the CTCS advection scheme:

$$\varphi_j^{(n+1)} = \varphi_j^{(n-1)} - c\left(\varphi_{j+1}^{(n)} - \varphi_{j-1}^{(n)}\right) \tag{2.03}$$

The CTCS equation in (2.03) uses three-time-levels: φ at t_{n-1}, t_n and t_{n+1}. Only $\varphi(x, t_0)$ is available, to initiate the simulation $\varphi(x, t_1)$ is required. $\varphi(x, t_1)$ is obtained using the forward-in time centred-in-space (FTCS) scheme:

$$\varphi_j^{(n+1)} = \varphi_j^{(n)} - \frac{c}{2}\left(\varphi_{j+1}^{(n)} - \varphi_{j-1}^{(n)}\right) \tag{2.04}$$

The second-order accurate CTCS scheme is said to have large dispersive errors by Hofmann *et al* (2006), which are hard to overcome when applying to coarse resolution models such as the OGCMs- Ocean General Circulation Models. However, through using the CSTC scheme has negligible numerical diffusion and is conceptually simple (Hofmann *et al* 2006).
Applying the Von-Neumann stability analysis on the CTCS scheme, where $\varphi_j^{(n)} = A^n e^{ik\, j\Delta x}$

is substituted into (2.03) to find the amplification factor A, it is shown that the amplification factor is:

$$A = -ic \sin k\Delta x \pm \sqrt{1 - c^2 \sin^2 k\Delta x} \tag{2.05}$$

From the equation above, it can be concluded that when $|c| \leq 1$, the amplification factor $|A|^2 = 1$, so the scheme is stable and not dampening. But when $|c| > 1$, the amplification factor is $|A|^2 = \left(c \sin k\Delta x \pm \sqrt{c^2 \sin^2 k\Delta x - 1}\right)^2$, therefore $|A| > 1$ for one of the roots, resulting in the scheme to be unstable when the Courant number is larger than one (Weller 2019).

2.2 Crank-Nicolson Centred-in-space (CNCS) scheme

Crank-Nicolson method is an implicit time-stepping scheme and second-order method in time, based on the trapezoidal rule- which combines the forward and backward Euler methods.

Crank-Nicolson method transforms the time derivative, $\frac{\partial \varphi}{\partial t}$, into:

$$\frac{\partial \varphi_j^{\left(n+\frac{1}{1}\right)}}{\partial t} = \frac{\varphi_j^{(n+1)} - \varphi_j^{(n)}}{\Delta t} \tag{2.06}$$

And the centred space derivative is averaged in time:

$$\frac{\partial \varphi_j^{\left(n+\frac{1}{2}\right)}}{\partial x} = \frac{1}{2}\left(\frac{\varphi_{j+1}^{(n+1)} - \varphi_{j-1}^{(n+1)}}{2\Delta x} + \frac{\varphi_{j+1}^{(n)} - \varphi_{j-1}^{(n)}}{2\Delta x}\right) \tag{2.07}$$

(2.05) and (2.06) are then substituted into the advection equation (1.1) at x_j, t_n. To get $\varphi_j^{(n+1)}$ on the left-hand-side, (2.07) is rearranged and the Courant number (1.2) is substituted in to get the equation used in the CNCS advection scheme:

$$\varphi_j^{(n+1)} = \varphi_j^{(n)} - \frac{c}{2}\left(\frac{\varphi_{j+1}^{(n+1)} - \varphi_{j-1}^{(n+1)}}{2} + \frac{\varphi_{j+1}^{(n)} - \varphi_{j-1}^{(n)}}{2}\right) \tag{2.08}$$

Equation in (2.08) has known values $\varphi^{(n)}$, and unknown values $\varphi^{(n+1)}$. To solve (2.08), the know values are collected on the right-hand-side and the unknown values are collected on the right-hand-side, (2.09) forms a matrix, and the matrix is then solved.

$$\varphi_j^{(n+1)} + \frac{c}{2}\left(\frac{\varphi_{j+1}^{(n+1)} - \varphi_{j-1}^{(n+1)}}{2}\right) = \varphi_j^{(n)} - \frac{c}{2}\left(\frac{\varphi_{j+1}^{(n)} - \varphi_{j-1}^{(n)}}{2}\right) \tag{2.09}$$

Applying the Von-Neumann stability analysis on the CNCS scheme shows that the amplification factor is:

$$A = \frac{1 - i\left(\frac{c}{2}\right)\sin k\Delta x}{1 + i\left(\frac{c}{2}\right)\sin k\Delta x} \tag{2.10}$$

It is seen that $|A| = 1$ irrespective of the Courant number. Therefore, the CNCS scheme is always stable and not dampening, even for large time steps. Dehghan (2004) also showed that the Crank-Nicolson scheme has no numerical diffusion. Dehghan (2004) compared the Crank-Nicolson scheme to other fully implicit and explicit schemes and found that the Crank-Nicolson scheme is one of the more efficient implicit finite difference schemes. He also adds that although the one-dimensional advection solution is efficient, when solving for three-dimensional advection, the matrix width increases- decreasing computational efficiency, therefore, increasing the computational cost.

The Crank-Nicolson implicit finite difference method is used in this paper because as mentioned above, it is one of the most efficient implicit finite difference schemes and is easy to implement for solving one-dimensional advection. It is also a good comparison with the CTCS scheme because both are of the same order of accuracy.

2.3 Semi-Lagrangian advection with cubic interpolation

Semi-Lagrangian schemes with linear interpolation generates large diffusion, therefore, Semi-Lagrangian with higher order interpolation will be used. Semi-Lagrangian with cubic interpolation is of higher accuracy (3rd order accurate) compared to Semi-Lagrangian with linear interpolation with is 1st order accurate (Durran 1999).

The advection equation in (1.1) is the Eulerian form of the advection equation, for Semi-Lagrangian advection the Lagrangian form with no sources of sinks is used.

The following derivation of the Semi-Lagrangian with cubic interpolation scheme is taken from Durran (1999). (1.3) can be rewritten using the Semi-Lagrangian approximation:

$$\frac{\varphi\left(x_j, t^{n+1}\right) - \varphi\left(\tilde{x}_j^n, t^n\right)}{\Delta t} = 0 \tag{2.11}$$

In (2.10) $\varphi\left(\tilde{x}_j^n, t^n\right)$ is the departure point and time of the trajectory, and $\varphi\left(x_j, t^{n+1}\right)$ is the point and time of arrival of the trajectory. Because in this project advection with constant wind speed u is assumed, computation of the backward trajectory is trivial, and is given by:

$$\tilde{x}_j^n = x_j - u\Delta t$$

Supposing that $u \geq 0$, and $\tilde{x}_j^n$lies between $x_{j-p-1} \leq \tilde{x}_j^n \leq x_{j-p}$ β can be defined as:

$$\beta = \frac{x_{j-p} - \tilde{x}_j^n}{\Delta x}$$

Approximating $\varphi\left(\tilde{x}_j^n, t^n\right)$ with cubic interpolation where the departure point has four closest grid points φ_{j-p-1}^n, φ_{j-p+1}^n, φ_{j-p}^n and φ_{j-p+2}^n, (2.11) becomes:

$$\varphi\left(\tilde{x}_j^n, t^n\right) = -\frac{1}{6}\beta(1-\beta)(2-\beta)\varphi_{j-p-1}^n + \frac{1}{2}(1+\beta)(1-\beta)(2-\beta)\varphi_{j-p}^n + \frac{1}{2}(1+\beta) + \beta(2-\beta)\varphi_{j-p+1}^n - \frac{1}{6}(1+\beta)\beta(1-\beta)\varphi_{j-p+2}^n \tag{2.12}$$

Because the Semi-Lagrangian scheme has backward trajectory calculations within it, the scheme requires more computational time than Eulerian methods (Durran 1999). The Semi-Lagrangian method is also non conservative, although, when looking at one-dimensional advection, conservation properties aren't apparent because the flow has uniform velocity (Durran 1999). On the other hand, the Semi-Lagrangian scheme permits longer time-steps than the Eulerian methods (Durran 1999). A Von-Neumann stability analysis on the Semi-Lagrangian advection scheme performed by Durran (1999), where it showed from the amplification factor that the scheme is always stable for all Courant numbers.

The Semi-Lagrangian with cubic interpolation scheme is used in this dissertation because it has a higher order of accuracy than the linear interpolation scheme, however, it isn't too high to be compared with lower order of accuracy Eulerian schemes.

3. Results

Periodic boundaries are implemented to test the advection scheme, using initial conditions of a bell curve with a constant velocity u.

The ℓ_2 error norm is calculated using the equation:

$$\ell_2(\varphi) = \frac{\sqrt{\sum_j \Delta x \left(\varphi_j - f(x_j)\right)^2}}{\sqrt{\sum_j \Delta x f(x_j)^2}} \tag{3.01}$$

The order of accuracy is calculated using the following equations:

$$O(\Delta t) = \frac{ln(\ell_{2_1}) - ln(\ell_{2_2})}{ln(\Delta t_1) - ln(\Delta t_2)} \tag{3.02}$$

$$O(\Delta x) = \frac{ln(\ell_{2_1}) - ln(\ell_{2_2})}{ln(\Delta x_1) - ln(\Delta x_2)} \tag{3.03}$$

In (3.02) Δt_1 is the size of the first set of time steps used and Δt_2 is the second set. In (3.03) Δx_1 is the size of the first set of space steps used and x is the second set. ℓ_2 are the error norms.

3.1. One-dimensional advection using Centred-in-time Centred-in-space scheme (CTCS)

Figure 3.01c) shows the profile of φ being advected around the periodic domain, it is seen that the scheme becomes unstable once the Courant number is larger than one. In Figure 3.01b) there is evidence of how the scheme is starting to become unstable before becoming chaotic. The error norm of the scheme shown in Table 3.01 decreases as the time step increases while the spatial resolution remains constant, however the error rapidly increases when the scheme becomes unstable (shown by Table 3.01); this happens when the time step doesn't satisfy the CFL condition.

The scheme is also unbounded as shown by Table 3.01, where the max (φ) is larger than one when the Courant number is 1.0 and min (φ) is smaller than zero for all the Courant numbers tested

Table 3.01: Set up to test the one-dimensional CTCS scheme, model run taking 3 seconds, using nx = 100, Δx = 0.01, u = 1.0, where c is the Courant number, Δt is the time step, nt is the number of time steps taken, max/min (φ) is the maximum and minimum φ values used to check the boundedness of the scheme, ℓ_2 is the error norm and O(Δx) is the order of accuracy.

c	Δt	nt	max (φ)	min (φ)	ℓ_2	O(Δt)
0.2	0.002	1500	0.9999	-0.0528	0.0667	-0.24
0.5	0.005	600	0.9971	-0.0429	0.0535	-0.05
1.0	0.01	300	1.0199	-0.0238	0.0389	25.92
1.2	0.012	250	2.3669	-2.3669	4.3872	

It appears from the negative order of accuracy and the error norms shown in Table 3.01, that as the time steps are being increased the advection scheme becomes more accurate, until the scheme passes its stability limit (when the Courant number is larger than 1). It is clear that the scheme passes its stability limit once the Courant number is larger than 1 because the order of convergence shown in Table 3.01 peaks to 25.92.

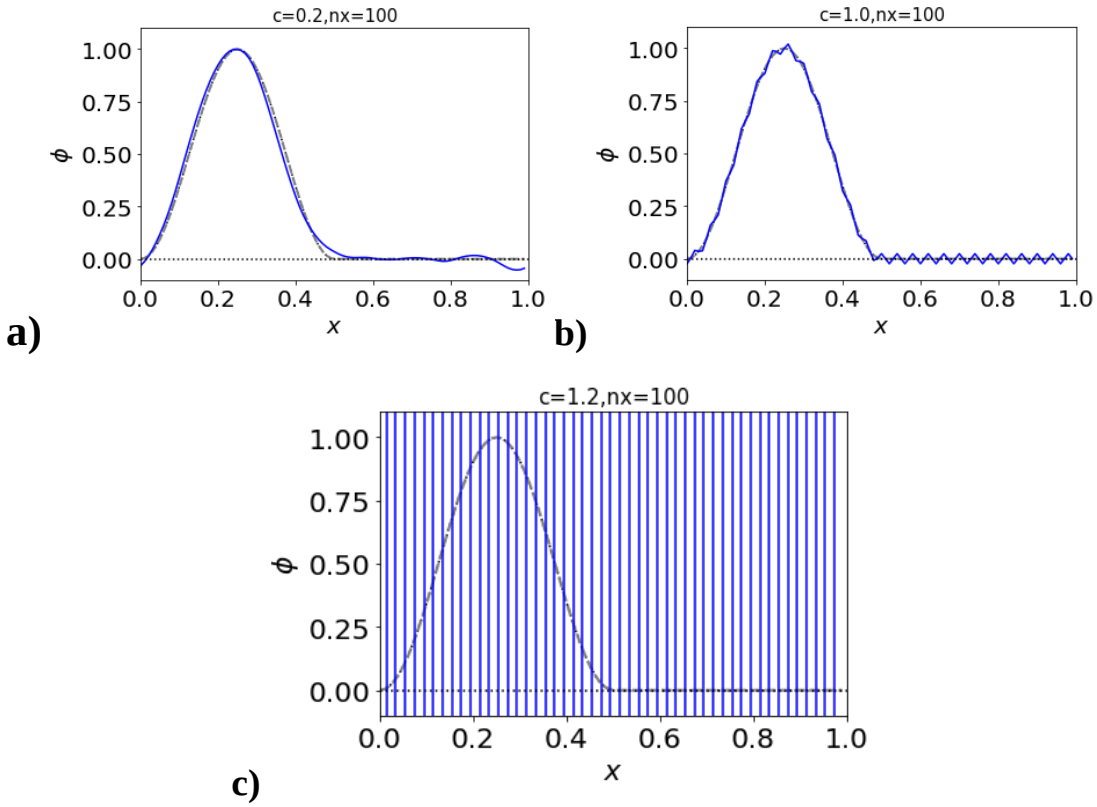


Figure 3.01: Blue line is the CTCS solution, grey dashed line is the exact solution. Graphs showing the profile of φ advected around the periodic domain. a) Numerical solution of the CTCS scheme, corresponding to c = 0.2 in table 3.1.1, b) Numerical solution of the CTCS scheme, corresponding to c = 1.0 in table 3.1.1 and c) Numerical solution of the CTCS scheme, corresponding to c = 1.2 in table 3.01.

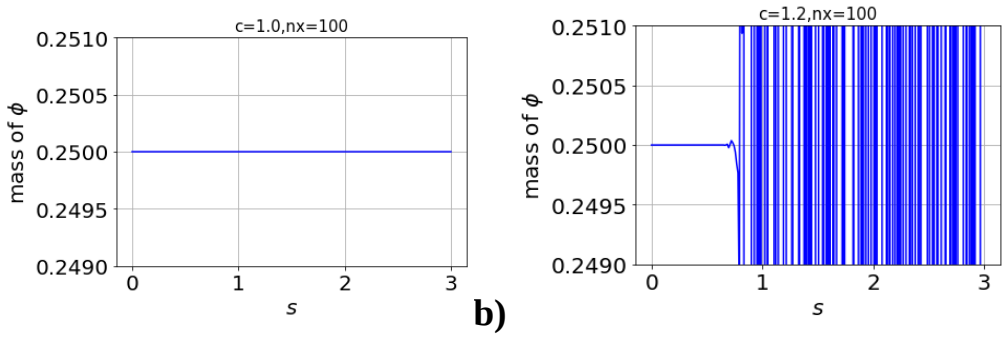


Figure 3.02: Graphs showing the conservation of the one-dimensional CTCS scheme, looking at how the mass of φ changes over the time period the model is run for. Setup used to retreve these results is described in Table 3.01. a) shows results using Courant number c = 1.0 and b) shows results using Courant number c = 1.2.

Figure 3.02a) shows how the scheme remains conservative for Courant numbers less than or equal to one then become non-conservative when the Courant number is larger than one. The scheme loses its conservation when it becomes unstable.

3.2 One-dimensional advection using Crank-Nicolson Centred-in-space scheme (CNCS)

From the set up used in Table 3.02 to test the CNCS scheme, the scheme proved to be stable for any Courant number size, as shown by Figure 3.03 where φ is advected around the periodic domain. This result confirms the Von Neumann stability analysis which concluded that the CNCS scheme is stable for any Courant number, and therefore satisfies the CFL condition. The scheme also proved to be unbounded, with max (φ) and min (φ) exceeding the bounds of one and zero, and the error norm of the CNCS scheme increases with the increasing time steps.

Table 3.02: Set up to test the one-dimensional CNCS scheme, model run taking 3 seconds, using nx = 100, Δx = 0.01, u = 1.0, where c is the Courant number, Δt is the time step, nt is the number of time steps taken, max/min (φ) is the maximum and minimum φ values used to check the boundedness of the scheme, ℓ_2 is the error norm and O(Δx) is the order of accuracy.

c	Δt	nt	max (φ)	min (φ)	ℓ_2	O(Δt)
0.2	0.002	1500	1.0020	-0.0528	0.0703	0.09
0.5	0.005	600	1.0062	-0.0589	0.0766	0.37
1.0	0.01	300	1.0114	-0.0669	0.0990	0.68
1.2	0.012	250	1.0135	-0.0739	0.1121	

Analysing the Order of accuracy and the error norms, it is apparent that the scheme becomes less accurate as the time step is being increased. The reason that the order of accuracy is so low is due to the error being dominated by spatial discretisation errors.

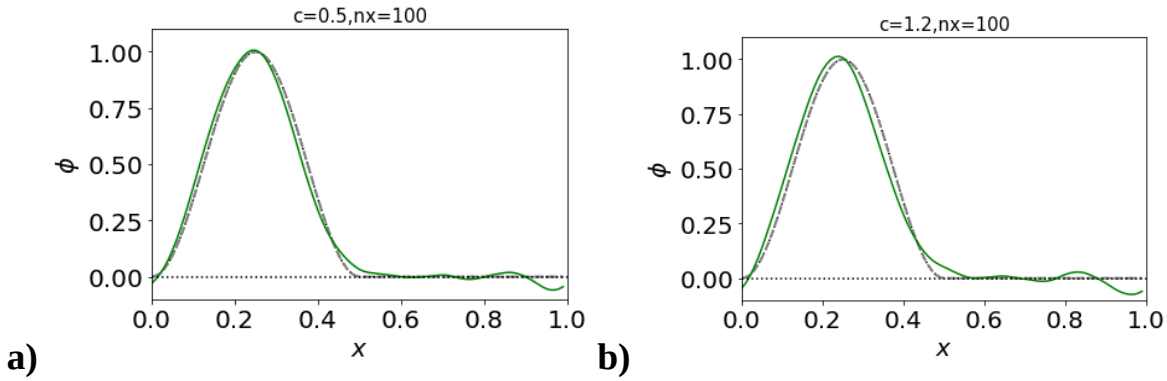


Figure 3.03: Green line is the CNCS solution, grey dashed line is the exact solution Graphs showing the profile of φ advected around the periodic domain. a) Numerical solution of the CNCS scheme, corresponding to c = 0.5 in table 3.02 and b) Numerical solution of the CNCS scheme, corresponding to c = 1.2 in table 3.2.1.

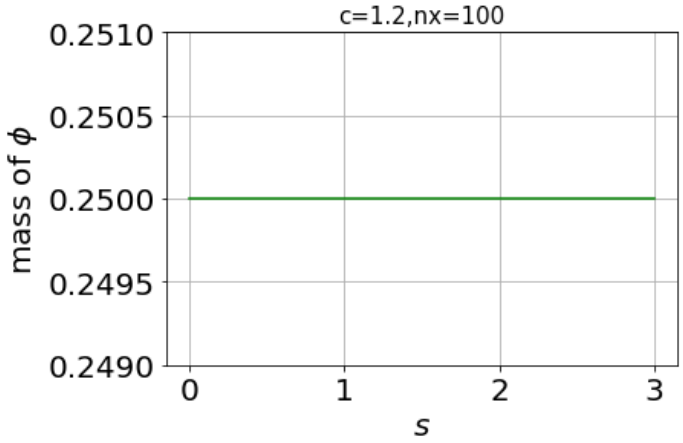


Figure 3.04: Graphs showing the conservation of the one-dimensional CNCS scheme with the Courant number c = 1.2, looking at how the mass of φ changes over the time period the model is run for. Setup used to retreve these results is described in more detail in Table 3.02.

When using the set up shown in table 3.02 to test the CNCS scheme, the scheme was conservative for all Courant numbers, just like shown in Figure 3.04.

3.3 One-dimensional advection using Semi-Lagrangian with cubic interpolation scheme

Using the set up in Table 3.03, Figure 3.05 shows how φ is advected around the periodic domain is stable for any Courant number size. The Von-Neumann stability analysis showed that the Semi-Lagrangian scheme is stable for any Courant number size, and this agrees with the results, therefore it can be used with large time steps.

The max and min φ exceeded the bounds of 0 and 1 of the real solution to the advection equation, therefore, the Semi-Lagrangian scheme is always unbounded. When looking at how the error norm varies with an increasing time step, Table 3.03 shows that the error norm decreases as the time step increases when the spatial resolution remains constant.

Table 3.03: Set up to test the one-dimensional Semi-Lagrangian scheme with cubic interpolation, model run taking 3 seconds, using nx = 100, Δx = 0.01, u = 1.0, where c is the Courant number, Δt is the time step, nt is the number of time steps taken, max/min (φ) is the maximum and minimum φ values used to check the boundedness of the scheme, ℓ_2 is the error norm and O(Δx) is the order of accuracy.

c	Δt	nt	max (φ)	min (φ)	ℓ_2	O(Δt)
0.2	0.002	1500	0.9973	-0.0067	0.0076	-0.35
0.5	0.005	600	0.9982	-0.0053	0.0055	-1.10
1.2	0.012	250	0.9996	-0.0027	0.0021	

Analysis of table 3.03 shows a negative order of accuracy, this together with the error norms decreasing, shows that the scheme becomes more accurate as the time step is being increased. This is something that wasn't expected.

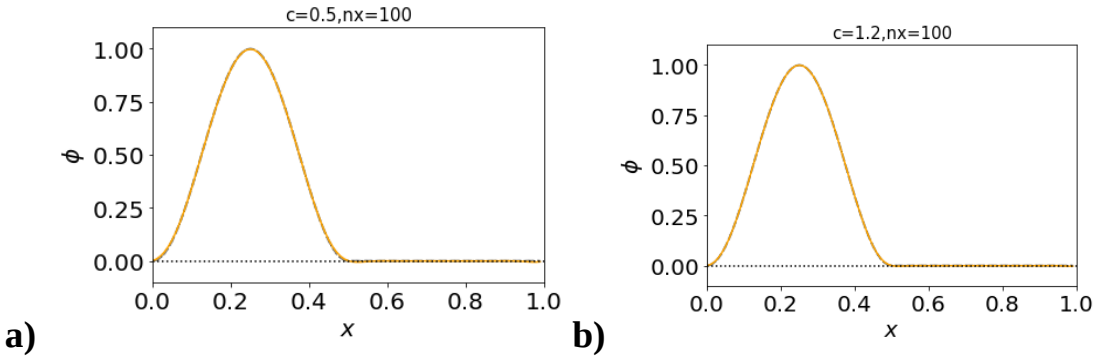


Figure 3.05: Yellow line is the Semi-Lagrangian solution with cubic interpolation, grey dashed line is the exact solution Graphs showing the profile of φ advected around the periodic domain. a) Numerical solution of the Semi-Lagrangian scheme with cubic interpolation, corresponding to c = 0.5 in table 3.03 and b) Numerical solution of the Semi-Lagrangian scheme with cubic interpolation, corresponding to c = 12.0 in table 3.03.

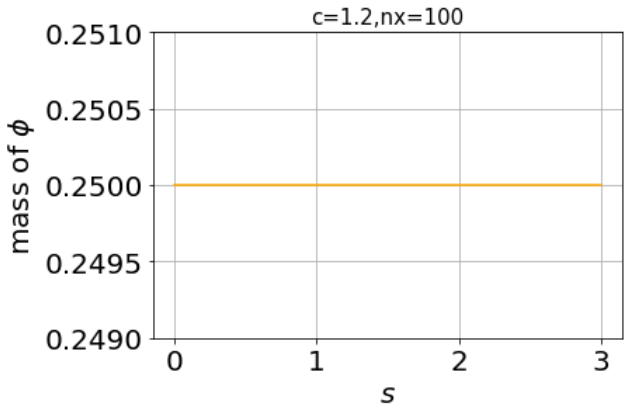


Figure 3.06: Graph showing the conservation of the one-dimensional Semi-Lagrangian with cubic interpolation scheme, with the Courant number c = 1.2, looking at how the mass of φ changes over the time period the model is run for. Setup used to retreve these results is described in more detail in Table 3.03.

The Semi-Lagrangian scheme tested is shown to be conservative by Figure 3.06, as expected when being tested with one dimentional advection.

3.4 Comparison of the schemes

The three schemes were investigated in more detail to see how their ℓ_2 error norms change when the courant number is varied in Table 3.04 and Figure 3.07 and the resolution (Δx) varies in Table 3.05 and Figure 3.09.

Table 3.04: Set up to analyse ℓ_2 error norm of the one-dimensional CTCS, CNCS and Sem-Lagrangian with cubic interpolation schemes. Model is run for 3 seconds with u = 1.0, keeping Δx constant at 0.01 and nx = 100. c is the Courant number, Δt is the length of the time step and nt is the number of time steps.

c	Δt	nt	CTCS ℓ_2	CNCS ℓ_2	Semi-Lagrangian ℓ_2
0.2	0.002	1500	0.0666	0.0703	0.0076
0.4	0.004	750	0.0592	0.0740	0.0063
0.5	0.005	600	0.0535	0.0766	0.0055
0.6	0.006	500	0.0464	0.0799	0.0047
0.8	0.008	375	0.0276	0.0882	0.0028

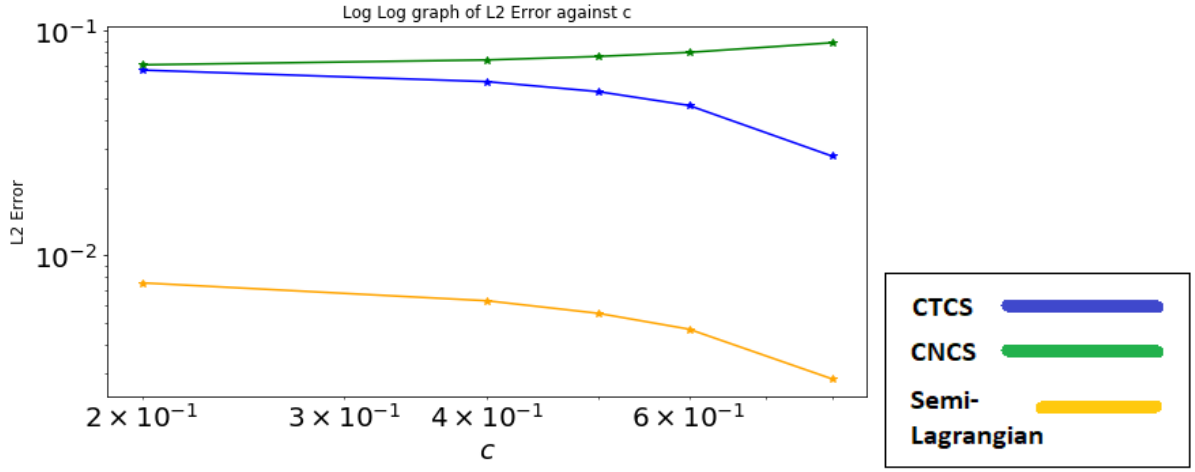


Figure 3.07: Graph showing the variation in ℓ_2 error norm of CTCS in blue, CNCS in green and Semi-Lagrangian with cubic interpolation in orange schemes, with the change in courant number (corresponding to set up in table 3.4.1) as the spatial resolution and number of space steps remains constant.

The Semi-Lagrangian and the CTCS scheme gets more accurate as the courant number approaches one, whereas the CNCS scheme's accuracy decreases as the courant number approaches to one. Figure 3.07 shows this with how the ℓ_2 error norm changes.

Table 3.07: Set up to analyse ℓ_2 error norm of the one-dimensional CTCS, CNCS and Sem-Lagrangian with cubic interpolation schemes. Model is run for 3 seconds with u = 1.0, keeping the courant number constant at c = 0.5. Δx is the size on the spatial resolution, nx is the number of space steps, Δt is the length of the time step and nt is the number of time steps taken.

Δx	nx	Δt	nt	CTCS ℓ_2	CNCS ℓ_2	Semi-Lagrangian ℓ_2
0.002	500	0.001	3000	0.0029	0.0042	0.0002
0.004	250	0.002	1500	0.0101	0.0146	0.0009
0.0060	166	0.0030	996	0.0212	0.0305	0.0019
0.008	125	0.004	750	0.0356	0.0511	0.0035
0.01	100	0.005	600	0.0535	0.0766	0.0055
0.02	50	0.01	300	0.1874	0.2627	0.0261
0.0294	34	0.0147	204	0.3696	0.4738	0.0637
0.04	25	0.02	150	0.5446	0.6481	0.1244

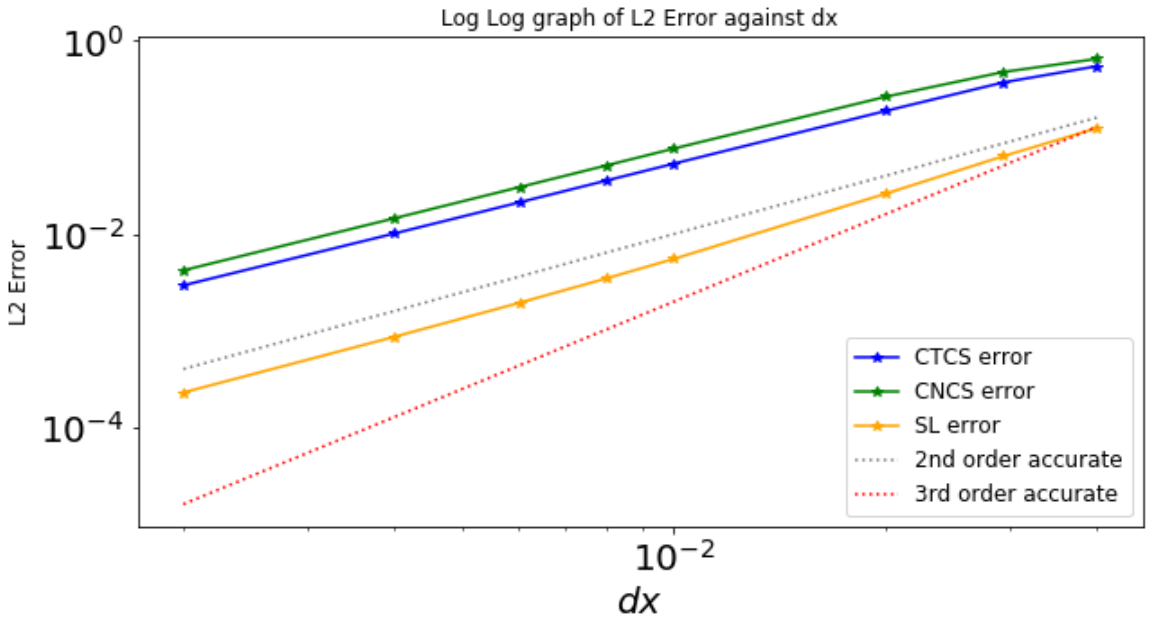


Figure 3.09: Graph showing the variation in ℓ_2 error norm of the CTCS in blue, CNCS in green and Semi-Lagrangian with cubic interpolation in orange schemes, with the change in Δx (corresponding to the set up in table 3.07) as the Courant number remains constant at c = 0.5. The grey and pink dashed lines correspond to the theoretical 2nd and 3rd orders of accuracy.

Orders of accuracy of the schemes can be estimated using Figure 3.09; the schemes error lines are compared to the 2nd and 3rd orders of accuracy lines. Figure 3.09 shows that all the schemes are the closest to the 2nd order accuracy line, meaning that they are 2nd order accurate. Semi-Lagrangian with cubic interpolation, however, is expected to be 3rd order accurate, which isn't what is shown in Figure 3.09. The 2nd order of accuracy for the Semi-Lagrangian scheme stems from the discontinuity in the initial conditions which aren't smooth in all derivatives (Holdaway *et al* 2008) when a bell curve is used. However, even though the semi-Lagrangian scheme is shown to have the same order of accuracy as the other schemes, it appears to be lower down on figure 3.09, meaning that it is more accurate overall. The CTCS scheme is lower than the CNCS scheme, therefore is more accurate of the two Eulerian schemes tested.

4. Conclusion

This dissertation looked at comparing Eulerian Schemes to the non-conservative Semi-Lagrangian with cubic interpolation scheme. Error norms showed that the Semi-Lagrangian with cubic interpolation scheme is more accurate compared to the CTCS and CNCS schemes, however the CTCS scheme was more accurate than the CNCS scheme. All the schemes were stable when the courant number was less than or equal to one, however the CTCS scheme became unstable the Courant number was larger than one. Even though the CNCS and the Semi-Lagrangian schemes were stable for Courant numbers larger than one, the Semi-Lagrangian was more accurate in presence of large Courant Numbers. For one dimensional advection all the schemes showed to be conservative, yet none were bounded.

One of the objectives of this dissertation was aimed at figuring out which scheme is best to use with larger time steps when considering the computational cost to run the schemes. From the tests that were able to be done in this dissertation, the Semi-Lagrangian showed the most accurate of the three schemes tested, and even with a large resolution; for the CTCS scheme to achieve a similar ℓ_2 error norm it needs a small resolution, meaning a larger computational cost. Therefore, the Semi-Lagrangian seems to be the most accurate and most efficient, but further research needs to be done to make definite conclusions.

5. Future Work

Due to time restrictions, this dissertation didn't cover all the objectives listed. For a fair comparison, the Semi-Lagrangian scheme must be compared to a Eulerian scheme which is of a similar order of accuracy (3^{rd} order accurate); this can be achieved by using the QUICK scheme, described by Versteeg and Malalasekera (2007). The schemes should also be tested in their two- and three-dimensional forms and incorporated into a more complete model to compare their performance with large time steps. More computational cost comparisons must also be made. The results should help to conclude whether the Semi-Lagrangian scheme can be replaced by a Eulerian scheme to cut down on computational cost when large time steps are being used. To avoid issues with initial conditions not being smooth in all derivatives, initial conditions with a sine curve should be used to test the schemes.

References

Arakawa, A., 1997: Computational Design for Long-Term Numerical Integration of the Equations of Fluid Motion: Two-Dimensional Incompressible Flow. Part I. *J. Comput. Phys.*, **135**, 2, 103-114, https://doi.org/10.1006/jcph.1997.5697.

Chen, Y., Weller, H., Pring, S., Shaw, J., 2017: Comparison of dimensionally split and multi-dimensional atmospheric transport schemes for long time steps. *Q. J. R. Meteorol. Soc.*, **143**, 2764-2779, https://doi.org/10.1002/qj.3125

Chock,P., Dunker, M., 1983: A comparison of Numerical Methods for Solving the Advection Equation. *Atmos. Environ.*, **17**, 1, 11-24, https://doi.org/10.1016/0004-6981(83)90003-3

Colella, P., P. P. Woodward, 1984: The Piecewise Parabolic Method (PPM) for gas-dynamical simulations. *J. Comput. Phys.*, **54**, 1, 174-201, https://doi.org/10.1016/0021-9991(84)90143-8

Davies, T., Cullen, M. J. P., Malcolm, A. J., Mawson, M. H., Staniforth, A., White, A. A., Wood, N., 2005: A new dynamical core for the Met Office's global and regional modelling of the atmosphere. *Q. J. R. Meteorol. Soc.*, **131**, 1759-1782, https://doi.org/10.1256/qj.04.101

Deghan, M., 2004: Numerical solution of the three-dimensional advection-diffusion equation. *Applied Mathematics and Computation*, **150**, 5-19, https://doi.org/10.1016/S0096-3003(03)00193-0.

Donea, J., B. Roig, and A. Huerta, 2000: High-order accurate time-stepping schemes for convection-diffusion problems. *Computer Methods in Applied Mechanics and Engineering*, **182**, 3-4, 249-275, https://doi.org/10.1016/S0045-7825(99)00193-0.

Dong, L., B. Wang, and L. Liu, 2014: A Lagrangian advection scheme with shape matrix (LASM) for solving advection problems. *Geosci. Model Dev.*, **7**, 2951- 2968, https:// doi.org/10.5194/gmd-7-2951-2014.

Durran, D. R., 1999: *Numerical Methods for Wave Equations in Geophysical Fluid Dynamics*. Springer-Verlag New York, Inc., 465pp.

Harris, L. M., P. H. Lauritzen, and R. Mittal, 2011: A Flux-form version of the conservative semi- Lagrangian multi-tracer transport scheme (CSLAM) on the cubed sphere grid. *J. Comput. Phys.*, **4**, 230, 1215-1237, https://doi.org/10.1016/j.jcp.2010.11.001.

Hirsch, C., 2007: *Numerical Computation of Internal and External Flows*. Butterworth-Heinemann, 674pp.

Hofmann, M. and M. A. M. Maqueda, 2006: Performance of a second-order moments advection scheme in an Ocean General Circulation Model. *J. Geophys. Res.*, **111**, https://doi.org/10.1029/2005JC003279

Holdaway, D., J. Thuburn and N. Wood, 2008: On the relation between order of accuracy, convergence rate and spectral slope for linear numerical methods applied to multiscale problems. *Int. J. Numer. Methods Fluids*, **56**, 1297-1303, https://doi.org/ 10.1002/fld.1644.

Hoskins, B. J., and I. N. James, 2014: *Fluid Dynamics of the Midlatitude Atmosphere.* John Wiley & Sons, Ltd, 408pp.

Lauritzen, P. H., Skamarock, W. C., Pather, M. J., Taylor, M.A., 2012: A standard test case suite for two-dimensional linear transport on the sphere. *Geosci. Model Dev.,* **5,** 887-901, doi:10.5194/gmd-5-887-2012

LeVeque, R. J., 2005: *Numerical Methods for Conservation Laws. Lectures in Mathematics.* Birkhäuser; 2nd edition, 232pp.

Lin, S., and R. B. Rood, 1996: Multidimensional Flux-Form Semi-Lagrangian Transport Schemes. *Mon. Wea. Rev.*, **124**, 2046-2070, https://doi.org/10.1175/1520-0493(1996)124<2046:MFFSLT>2.0.CO;2.

Nair, R. D., and B. Machenhauer, 2001: The Mass-Conservative Cell-Integrated Semi-Lagrangian Advection Scheme on the Sphere. *Mon. Wea. Rev.*, **130**, 3, 649-667, https://doi.org/10.1175/1520-0493(2002)130<0649:TMCCIS>2.0.CO;2.

Park, C., and M. Allaby, 2017: *A Dictionary of Environment and Conservation (3 ed.).* Oxford University Press.

Simmons, A. J., and D. M. Burridge, 1981: An Energy and Angular-Momentum Conserving Vertical Finite-Difference Scheme and Hybrid Vertical Coordinates. *Mon. Wea. Rev.*, **109**, 4, 758-766, https://doi.org/10.1175/1520-0493(1981)109<0758:AEAAMC>2.0.CO;2.

Thuburn, J., 2011: Conservation in Dynamical Cores: What, How and Why? *Numerical Techniques for Global Atmospheric models*, P. H. Lauritzen *et al*. (eds.), Springer-Verlag Berlin Heidelberg, 345-355.

Thuburn, J., and C. Cotter, 2012: Gung-Ho Dynamics: Survey of Transport Schemes. Tech. rep., UK Met Office, http://collab.metoffice.gov.uk/twiki/bin/view/Project/NGWCP _Pase1, 1-19pp.

Ullrich, P. A., and M. R. Norman, 2014: The Flux-Form Semi-Lagrangian Spectral Element (FF-SLE) method for tracer transport. *Q. J. Roy. Meteor. Soc.*, **140**, 1069-1085, https://doi.org/10.1002/qj.2184.

Versteeg, H. K., and W. Malalasekera, 2007: *An introduction to Computational Fluid Dynamics, The finite Volume Method*. Pearsons Education Limited, 503pp

Weller, H., 2019: *MTMW12: Introduction to Numerical Methods for Atmosphere and Ocean Models.* Lecture notes, University of Reading, 77pp, http://www.met.reading.ac.uk/~sws02hs/teaching/MTMW12/MTMW12_2_lec.pdf

Weller, H., S. J. Lock, and N. Wood, 2013: Runge-Kutta IMEX schemes for the Horizontally Explicit/Vertically Implicit (HEVI) solution of wave equations. *J. Comput. Phys.*, **252**, 365-381.

Xiaocong, W., L. Yiumin, W. Guoxiong, L. Shian-Jiann, and B. Qing, 2013: The Application of Flux-Form Semi-Lagrangian Transport Scheme in a Spectral Atmosphere Model. *Adv. Atmos. Sci.*, **30**, 89-100, https://doi.org/10.1007/s00376-012-2039-2.

Appendix

The code written to implement the schemes and the functions used are bellow. Code written by me is highlighted.

https://github.com/vlopatyuk279/Numerical-schemes-to-model-advection.git

Initial conditions function:

```
def initialBell(x):
    "Normal distribution of phi as initial condition"
    return np.where(x%1. < 0.5, np.power (np.sin(2*x*np.pi), 2), 0)
```

Advection scheme Functions:

```
def CTCS(phiOld, c, nt, dx):
    "Advection of profile in phiOld using CTCS non-dimentional Courant"
    "number, c"

    nx=len(phiOld)

    # New time-step array for phi
    phi = phiOld.copy()
    phiNew = phiOld.copy()

    totalMass = np.zeros(nt+1)
    totalMass[0] = mass(phi, dx)

    #FTCS for the first time-step, looping over space
    for j in range(nx):
        phi[j]=phiOld[j] - 0.5*c*(phiOld[(j+1)%nx] - phiOld[(j-1)%nx])

    totalMass[1] = mass(phi, dx)

    # CTCS for all time steps
    for n in range (1,nt):
        # Code for CTCS at each time-step
        # Loop over space
        for j in range(nx):
            phiNew[j] = phiOld[j] - c*(phi[(j+1)%nx] - phi[(j-1)%nx])
        # Update phi for the next time-step
        phiOld = phi.copy()
        phi = phiNew.copy()
        totalMass[n+1] = mass(phi, dx)

    return phi, totalMass

def CNCS (phi, c, nt, dx):
    "Advection of profile in phiOld using CTCS non-dimentional Courant"
    "number, c"

    #Array for the RHS of the matrix equation
    RHS = phi.copy()

    nx = len(phi)

    totalMass = np.zeros(nt+1)
```

```
    totalMass[0] =mass(phi, dx)

    M = np.zeros([nx,nx])
    for j in range (nx):
        M[j,j] = 1
        M[(j-1)%nx,j] = 0.25*c
        M[(j+1)%nx,j] = -0.25*c

    #Solution for nt time steps
    for it in range(nt):
        for j in range (nx):
            RHS[j] = phi[j] - 0.25*c*(phi[(j+1)%nx] - phi[(j-1)%nx])

        phi = np.linalg.solve(M, RHS)
        totalMass[it+1] = mass(phi, dx)

    return phi, totalMass

def Semi_Lagrangian (phi, c, nt, dx):
    "Advection of profile in phiOld using Semi-Lagrangia non-dimensional
Courant number, c"

    nx = len(phi)

    # New timestep array for phi
    phiNew = phi.copy()

    totalMass = np.zeros(nt+1)
    totalMass[0] = mass(phi, dx)

    for n in range (nt):

        for j in range(nx):
            k = m.floor(j-c)
            beta = j-k-c
            phiNew[j] = -((1/6)*beta*(1-beta)*(2-beta)*phi[(k-1)%nx]) \
                      + (0.5*(1+beta)*(1-beta)*(2-beta)*phi[k%nx])\
                      + (0.5*(1+beta)*beta*(2-beta)*phi[(k+1)%nx])\
                      - ((1/6)*(1+beta)*beta*(1-beta)*phi[(k+2)%nx])
        phi = phiNew.copy()
        totalMass[n+1] = mass(phi, dx)

    return phi, totalMass
```

Main application of the advection scheme function:

```
def main(nx, c, t):
    "Advect a bell curve on a domain between x = xmin and x = xmax over nx
"
    "spatial steps with advection coefficient c, time step dt for nt time
steps"

    # Parameters
    u = 1.0

    #derived parameters
    dx = 1.0/nx
    dt = c*dx/u
    nt = int(t/dt)
```

```
    print ("Courant Number =" , c)
    print ("dx =" , dx, "dt =", dt, "nt = ",nt)
    print ("end time =" ,t)

    # Spatial variable going from zero to one inclusive
    x = np.arange(0,1,dx)
    print ('x=', x)

    time=np.arange(0,t+dt,dt)

    # Three time levels of the dependant variable, phi
    phiOld = initialBell(x)

    #Advection using CTCS, CNCS and Semi-Langrangian
    [phiCTCS, CTCSmass] = CTCS(phiOld.copy(), c, nt, dx)
    [phiCNCS, CNCSmass] = CNCS(phiOld.copy(), c, nt, dx)
    [phiSL, SLmass] = Semi_Lagrangian(phiOld.copy(), c, nt, dx)


    #phiExact
    phiExact = initialBell(x - u*t)


    #Calculate and print out error norms
    CTCS_L2_error = L2ErrorNorm(phiCTCS, phiExact)
    print ('L2 Error Norm for CTCS:',CTCS_L2_error)
    CNCS_L2_error = L2ErrorNorm(phiCNCS, phiExact)
    print ('L2 Error Norm for CNCS:', CNCS_L2_error)
    SL_L2_error = L2ErrorNorm(phiSL, phiExact)
    print ('L2 Error Norm for Semi-Lagrangian:' , SL_L2_error)

    #Boundedness of the schemes
    print('Exact solution phi max=',max(phiExact))
    print('Exact solution phi min=',min(phiExact))
    print('CTCS phi max=',max(phiCTCS))
    print('CTCS phi min=', min(phiCTCS))
    print('CNCS phi max=',max(phiCNCS))
    print('CNCS phi min=', min(phiCNCS))
    print('SL phi max=',max(phiSL))
    print('SL phi min=', min(phiSL))

    #Plotting all advection scheme
    plt.figure()
    font = {'size' :20}
    plt.rc('font', **font)
    #Plotting starting phi
    plt.plot(x, phiOld, label= 'Initial', color= 'black', linestyle= ':')
    #Plotting exact solution
    plt.plot(x, phiExact, label='Analytic' , color='gray', linestyle= '--',
linewidth=2)
    #plotting advection schemes
    plt.plot (x, phiCTCS, label='CTCS', color='blue')
    plt.plot (x, phiCNCS, label='CNCS', color= 'green')
    plt.plot(x, phiSL, label='Semi Lagrangian', color='orange')
    plt.ylim(-0.1,1.1)
    plt.xlim(0,1)
    plt.title(f'c={c},nx={nx}', fontsize=15)
    plt.xlabel ('$x$')
    plt.ylabel('$\phi$')
    plt.show
```

```
    #plotting CTCS scheme
    plt.figure()
    font = {'size' :20}
    plt.rc('font', **font)
    #Plotting starting phi
    plt.plot(x, phiOld, label= 'Initial', color= 'black', linestyle= ':')
    #Plotting exact solution
    plt.plot(x, phiExact, label='Analytic' , color='gray', linestyle= '--',
linewidth=2)
    #Plotting scheme
    plt.plot (x, phiCTCS, label='CTCS', color='blue')
    plt.axhline(0, linestyle=':', color='black')
    plt.ylim(-0.1,1.1)
    plt.xlim(0,1)
    plt.title(f'c={c},nx={nx}', fontsize=15)
    plt.xlabel ('$x$')
    plt.ylabel('$\phi$')
    plt.show


    #Plotting Crank-Nicolson scheme
    plt.figure()
    font = {'size' :20}
    plt.rc('font', **font)
    #Plotting starting phi
    plt.plot(x, phiOld, label= 'Initial', color= 'black', linestyle= ':')
    #Plotting exact solution
    plt.plot(x, phiExact, label='Analytic' , color='gray', linestyle= '--',
linewidth=2)
    #Plotting scheme
    plt.plot (x, phiCNCS, label='CNCS', color= 'green')
    plt.axhline(0, linestyle=':', color='black')
    plt.ylim(-0.1,1.1)
    plt.xlim(0,1)
    plt.title(f'c={c},nx={nx}', fontsize=15)
    plt.xlabel ('$x$')
    plt.ylabel('$\phi$')
    plt.show


    #Plotting Semi-LAgrangian scheme with cubic interpolation
    plt.figure()
    font = {'size' :20}
    plt.rc('font', **font)
    #Plotting starting phi
    plt.plot(x, phiOld, label= 'Initial', color= 'black', linestyle= ':')
    #Plotting exact solution
    plt.plot(x, phiExact, label='Analytic' , color='gray', linestyle= '--',
linewidth=2)
    #Plotting scheme
    plt.plot(x, phiSL, label='Semi Lagrangian', color='orange')
    plt.axhline(0, linestyle=':', color='black')
    plt.ylim(-0.1,1.1)
    plt.xlim(0,1)
    plt.title(f'c={c},nx={nx}', fontsize=15)
    plt.xlabel ('$x$')
    plt.ylabel('$\phi$')
    plt.show


    #Plotting conservation of phi for all advection schemes used
```

```
    plt.figure()
    plt.plot(time, CNCSmass, label='CNCS mass of $\phi$', color='green')
    plt.plot(time, CTCSmass, label='CTCS mass of $\phi$', color='blue')
    plt.plot(time, SLmass, label='SL mass of $\phi$', color='orange')
    plt.grid(b=None, which='major', axis='both')
    plt.title(f'c={c},nx={nx}', fontsize=15)
    plt.ylim(0.249,0.251)
    plt.xlabel('$s$')
    plt.ylabel('mass of $\phi$')
    plt.show

    #Plotting conservation of phi for CTCS
    plt.figure()
    plt.plot(time, CTCSmass, label='CTCS mass of $\phi$', color='blue')
    plt.axhline(0, linestyle=':', color='black')
    plt.grid(b=None, which='major', axis='both')
    plt.title(f'c={c},nx={nx}', fontsize=15)
    plt.ylim(0.249,0.251)
    plt.xlabel('$s$')
    plt.ylabel('mass of $\phi$')
    plt.show


    #Plotting conservation of phi for CNCS
    plt.figure()
    plt.plot(time, CNCSmass, label='CNCS mass of $\phi$', color='green')
    plt.axhline(0, linestyle=':', color='black')
    plt.grid(b=None, which='major', axis='both')
    plt.title(f'c={c},nx={nx}', fontsize=15)
    plt.ylim(0.249,0.251)
    plt.xlabel('$s$')
    plt.ylabel('mass of $\phi$')
    plt.show


    #Plotting conservation of phi for Semi Lagrangian
    plt.figure()
    plt.plot(time, SLmass, label='SL mass of $\phi$', color='orange')
    plt.axhline(0, linestyle=':', color='black')
    plt.grid(b=None, which='major', axis='both')
    plt.title(f'c={c},nx={nx}', fontsize=15)
    plt.ylim(0.249,0.251)
    plt.xlabel('$s$')
    plt.ylabel('mass of $\phi$')
    plt.show
```

Diagnostics:

```
def L2ErrorNorm(phi, phiExact):
    """Calculates the L2 error norm (RMS error) of phi in comparison to
    phiExact, ignoring the boundaries"""

    # calculate the error and the error norms
    phiError = phi - phiExact
    L2 = np.sqrt(sum(phiError**2)/sum(phiExact**2))
    return L2

def mass(phi, dx):
    """Clculates the total mass of phi"""
    return np.sum(phi) * dx
```

```
def L2_E_N(nx, c, t):
    """Function to calculate the L2 error norm of CTCS, CNCS and
Semilagrangian
    advection schemes. Function converg loops over different values of nt
to
    show how the l2 error norm changes as a function of dx and dt."""

    #Parameters
    u = 1.0
    #Setting empty arrays for dx,dt and nt
    dx = np.zeros(len(nx))
    dt = np.zeros(len(nx))
    nt = np.zeros(len(nx), dtype='int')
    #Setting empty arrays for the error norms for all schemes being tested
    L2ErrorNormCTCS = np.zeros(len(nx))
    L2ErrorNormCNCS = np.zeros(len(nx))
    L2ErrorNormSemiLag = np.zeros(len(nx))
        #Loops over all of the space steps
        for i in range (len(nx)):
        dx[i] = 1.0/ (nx[i])
        dt[i] = c[i]*dx[i]/u
        nt[i] = t/dt[i]

        # Spatial variable going from zero to one exclusive
        x = np.arange(0, 1, dx[i])

        phiOld = initialBell(x)

        #Exact solution of the advection
        phiExact = initialBell(x - u*t)

        #Advection using CTCS
        [phiCTCS, CTCSmass] = CTCS(phiOld.copy(), c[i], nt[i], dx[i])

        L2ErrorNormCTCS[i] = L2ErrorNorm(phiCTCS, phiExact)


        #Advection using CNCS
        [phiCNCS, CNCSmass] = CNCS(phiOld.copy(), c[i], nt[i], dx[i])

        L2ErrorNormCNCS[i] = L2ErrorNorm(phiCNCS, phiExact)


        #Advection using Semi-Lagrangian
        [phiSemiLagrangian, SLmass] = Semi_Lagrangian(phiOld.copy(), c[i],
nt[i], dx[i])

        L2ErrorNormSemiLag[i] = L2ErrorNorm(phiSemiLagrangian, phiExact)


    #Plotting Error norms of advection schemes
    plt.figure()
    plt.plot(dx, L2ErrorNormCTCS, label='CTCS error', color='blue',
marker='*')
    plt.plot(dx, L2ErrorNormCNCS, label='CNCS error', color='green',
marker='*')
    plt.plot(dx, L2ErrorNormSemiLag, label='SL error', color='orange',
marker='*')
    plt.axhline(0, linestyle=':', color='black')
    #plt.legend(bbox_to_anchor=(1.1,1))
```

```
    #plt.title('Graph of L2 Error against the Courant number', fontsize=15)
    plt.xlabel('dx')
    plt.ylabel('$l_{2}$')
    plt.show

    #Plotting error norms for the Semi-Lagrangian scheme
    plt.figure()
    plt.plot(dx, L2ErrorNormSemiLag, label='SL error', color='orange',
marker='*')
    plt.axhline(0, linestyle=':', color='black')
    plt.xlabel('dx')
    plt.ylabel('$l_{2}$')
    plt.show

    #Plotting error norms for the CTCS scheme
    plt.figure()
    plt.plot(dx, L2ErrorNormCTCS, label='CTCS error', color='blue',
marker='*')
    plt.axhline(0, linestyle=':', color='black')
    plt.xlabel('dx')
    plt.ylabel('$l_{2}$')
    plt.show

    #Order of accuracy with dx
    plt.figure(figsize=(10,5))
    plt.loglog(dx, L2ErrorNormCTCS, label='CTCS error', color='blue',
marker='*')
    plt.loglog(dx, L2ErrorNormCNCS, label='CNCS error', color='green',
marker='*')
    plt.loglog(dx, L2ErrorNormSemiLag, label='SL error', color='orange',
marker='*')
    slopeCTCS, interseptCTCS = np.polyfit(np.log(dx),
np.log(L2ErrorNormCTCS), 1)
    slopeCNCS, interseptCNCS = np.polyfit(np.log(dx),
np.log(L2ErrorNormCNCS), 1)
    slopeSL, interseptSL = np.polyfit(np.log(dx),
np.log(L2ErrorNormSemiLag), 1)
    print('CTCS--Slope of graph(order of convergence)=',slopeCTCS)
    print('CNCS--Slope of graph(order of convergence)=',slopeCNCS)
    print('SL--Slope of graph(order of convergence)=',slopeSL)
    plt.axhline(0, linestyle=':', color='black')
    plt.title('Log Log graph of L2 Error against dx', fontsize=12)
    plt.xlabel('$dx$')
    plt.ylabel('L2 Error', fontsize=12)

    #Order of accuracy with c
    plt.figure(figsize=(10,5))
    plt.loglog(c, L2ErrorNormCTCS, label='CTCS error', color='blue',
marker='*')
    plt.loglog(c, L2ErrorNormCNCS, label='CNCS error', color='green',
marker='*')
    plt.loglog(c, L2ErrorNormSemiLag, label='SL error', color='orange',
marker='*')
    slopeCTCS, interseptCTCS = np.polyfit(np.log(c),
np.log(L2ErrorNormCTCS), 1)
    slopeCNCS, interseptCNCS = np.polyfit(np.log(c),
np.log(L2ErrorNormCNCS), 1)
    slopeSL, interseptSL = np.polyfit(np.log(c),
np.log(L2ErrorNormSemiLag), 1)
    print('CTCS--Slope of graph(order of convergence)=',slopeCTCS)
    print('CNCS--Slope of graph(order of convergence)=',slopeCNCS)
```

```
    print('SL--Slope of graph(order of convergence)=',slopeSL)
    plt.axhline(0, linestyle=':', color='black')
    plt.title('Log Log graph of L2 Error against c', fontsize=12)
    plt.xlabel('$c$')
    plt.ylabel('L2 Error', fontsize=12)
```