An Exploration of Semi-Lagrangian Methods for Advection

Final Project for the Course: Computational Methods for Classical PDEs in the Physical Sciences

Ryan Shìjié Dù

September 5, 2022

Abstract

We explore using Semi-Lagrangian numerical methods to solve linear and nonlinear advection problem. The main benefit of Semi-Lagrangian methods is that they are stable for large timestep. In particular it is stable for timestep size where the Courant–Friedrichs–Lewy (CFL) number defined in the traditional finite difference sense is larger than one. But for a Semi-Lagrangian scheme to be accurate, we need to solve for the departure points and then interpolate the advected field onto the departure points, both to high accuracy. We explore several Semi-Lagrangian schemes and reach fourth order space-time accuracy for linear advection where the advection velocity is known a priori. And for nonlinear advection, where the velocity is obtained from the advected field, we have third order. We also explore spectral interpolation and reach the same order of accuracy but without the need for spatial resolution refinement.

0 About this version

This note is an updated version of the final project submission for the numerics class by Alex Donev in Fall of 2021. This version incorporates some comments by Alex Donev on the original submitted version. There is no major revisions and some mistakes were acknowledged but uncorrected. This note should act as a sample of final project for this class, instead of a polished note for the public.

1 Introduction

We want to solve the linear non-diffusive advection of a passive tracer[1]. Assume $\nabla \cdot \boldsymbol{u} = 0$, we have the Partial Differential Equation (PDE):

$$\frac{\mathrm{D}}{\mathrm{D}t} c(\boldsymbol{X}(t), t) = \frac{\partial}{\partial t} c(\boldsymbol{x}, t) + \boldsymbol{u}(\boldsymbol{x}, t) \cdot \nabla c(\boldsymbol{x}, t) = 0 \tag{1}$$

where c is the concentration of the passive tracer of interests, $\boldsymbol{X}(t)$ is the Lagrangian trajectories of a fluid parcel (capital $\boldsymbol{X}$ denotes a marker for a particular fluid parcel), and $\boldsymbol{u}$ is the velocity of the underlying flow. We have the relation between $\boldsymbol{X}(t)$ and $\boldsymbol{u}$ is the ODE

$$\frac{\mathrm{d}}{\mathrm{d}t} \boldsymbol{X}(t) = \boldsymbol{u}(\boldsymbol{X}(t), t). \tag{2}$$

[1] It is possible to add diffusion in the Semi-Lagrangianframework, see [SK06].

(1) says that for such a passive tracer without diffusion, the concentration c is constant along the Lagrangian trajectory (i.e.: along $\boldsymbol{X}(t)$).

From this fact, one might design a Lagrangian approach of solving (1), where we solve the ODE (2) for a set of particles in time. Then the advection field c at final time T can be obtained from the knowing the final position of the particles since

$$c(\boldsymbol{X}(0), 0) = c(\boldsymbol{X}(T), T).$$

However, this approach degrades in accuracy in time since the initial well-spaced particles might evolve to irregular shapes and leaving big gaps where no information about c is available.

Semi-Lagrangian combines the Eulerian approach and the Lagrangian approach. In a nutshell, at each timestep we know c at time t on a regular *Eulerian* grid $\boldsymbol{x}_i$ and we want to we calculate the information about c at $t + \Delta t$ on the same regular *Eulerian* grid. To do this, we use

$$c(\boldsymbol{X}_i(t + \Delta t), t + \Delta t) = c(\boldsymbol{X}_i(t), t).$$

$\boldsymbol{X}_i(t + \Delta t) = \boldsymbol{x}_i$ since it is on the Eulerian grid, and we also use the notation $\hat{\boldsymbol{x}}_i = \boldsymbol{X}_i(t)$ is the departure point for the fluid parcel marked by $\boldsymbol{X}_i(t + \Delta t)$ such that a *Lagrangian* fluid trajectory goes through $\hat{\boldsymbol{x}}_i$ (resp. $\boldsymbol{x}_i$) at time t (resp. $t + \Delta t$). The Lagrangian treatment of advection in each timestep allows for bigger time steps to be taken, and we still maintain the grid structure in Eulerian space.

In general the departure points $\hat{\boldsymbol{x}}_i$ will not be on a grid point, therefore we need to interpolate to obtain an estimate. Except in trivial flows, we do not know the departure point $\hat{\boldsymbol{x}}_i$ exactly, so we need to numerically solve for an approximation. The main question to answer for Semi-Lagrangian numerical method is how do we calculate the departure points $\hat{\boldsymbol{x}}_i$ and how do we interpolate, and what are their properties.

For the rest of the report we will explore several time integrating and space interpolation schemes of various order. We will use them to solve the linear advection problem (1), but also nonlinear advection problem like the 2D inviscid and incompressible Navier-Stokes equation (6). The report will be organized as such: in Section 2 we will test many interpolation schemes by using them to solve the 1D linear advection scheme with constant velocity. Section 3 contains the test of Semi-Lagrangian Method on 2D linear advection problem (with constant and variable, but known a priori, velocity field). Section 4 studies Semi-Lagrangian Method for 2D nonlinear advection problem (2D Navier-Stokes). And finally in Section 5 we solve linear advection problems where the velocity is obtained numerically.

The code used in this report is available here. To run them, you will also need to setup some tools available here.

2 Interpolation schemes, tested in 1D constant advection

We first investigate the property of some interpolation scheme when used in Semi-Lagrangian methods. We solve 1D constant velocity advection problem:

$$\frac{\mathrm{D}}{\mathrm{D}t} c(X(t), t) = \frac{\partial}{\partial t} c(x, t) + u \frac{\partial}{\partial x} c(x, t) = 0.$$

This problem is enough to show the properties of various interpolation schemes, and it is easy to visualize so that we can determine the nature of the error produced. Variable velocity case would not help with our investigation, since timestep error would pollute the interpolation error.

We set the domain to be $[0, 1)$ periodic, and solve for $t \in [0, 1]$. $u = 1$ constant so that at $T = 1$, the tracer should be advection around the domain to its original position. This provide us with a test of accuracy of the solution. Since u is constant, we know the departure points a priori thus interpolation is the only source of errors. There is no error in time, we fix the timestep constant as 7 (this is an odd number, and Nx is always even, to ensure that we are not landing on a grid point). To investigate the order of convergence for the interpolation method, we vary the spacial resolution ($Nx = 8 : 1024$). Note that our CFL number is larger than one and reaching about 150 when the spatial resolution is the most dense. This corroborates the fact that the Semi-Lagrangian method is unconditionally stable. We solve the problem with a power of trigonometric function initial condition:

$$c(x, 0) = [\sin(\pi x)]^p \qquad \text{with } p = 40.$$

We investigate a handful of interpolation methods implemented in MATLAB's `interp1` function as well as spectral interpolation method based on Flatiron Institute Nonuniform Fast Fourier Transform (FINUFFT) [BMaK19, Bar21]. Note that for MATLAB's `interp1` function, we feed it padded data where we extend the function on the left and right by five points using periodicity. For all query point, we mod it to make sure that it stays within $[0, L)$. We start with `linear`, which is Lagrange linear interpolation. Fig. 1 shows the result: the left is comparison of the numerical solutions to the truth, and the right is the plot for order of convergence. We see that the error is mainly diffusive, and the order of the interpolation is second. For constant velocity, Semi-Lagrangian method is equivalent to the upwind method when the CFL number is less then one. Therefore it is reassuring to see that the nature of the error is the same (diffusive).

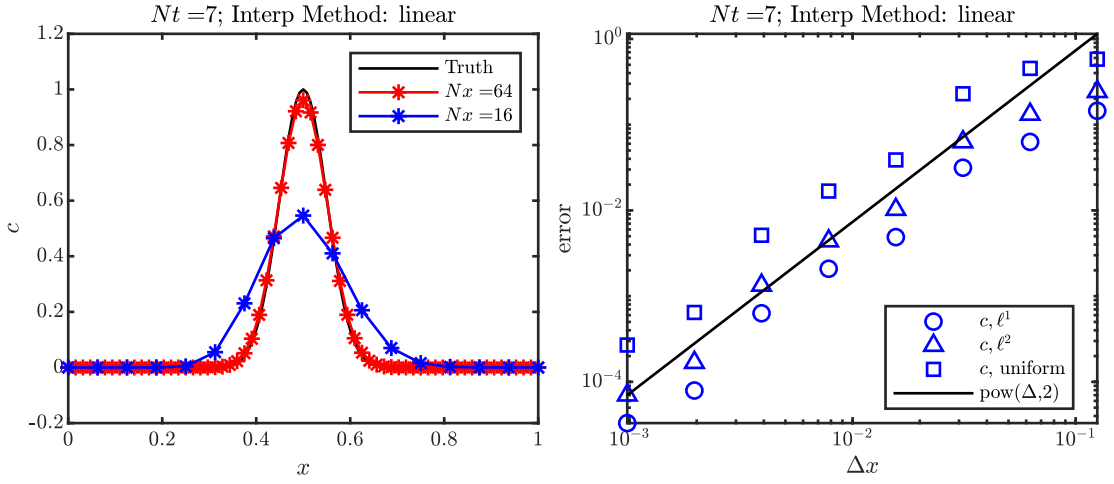

Figure 1:

Now we look at `cubic`, which is not cubic Lagrangian, but instead the cubic interpolation in [Key81] and `spline`, cubic spline[2]. Fig. 2 and Fig. 3 shows the result for `cubic` and `spline` respectively.

[2]The cubic spline in this note is not technically correct. In the periodic case, we need to use periodic spline if we wish the first derivative of the two end points match (`csape` in MATLAB).

For both methods, the error is both diffusive and dispersive. The convergence order of `cubic` is third, this corroborate that this is not cubic Lagrangian, since it should be of fourth order. Cubic `spline` is of fourth order, as we expected.

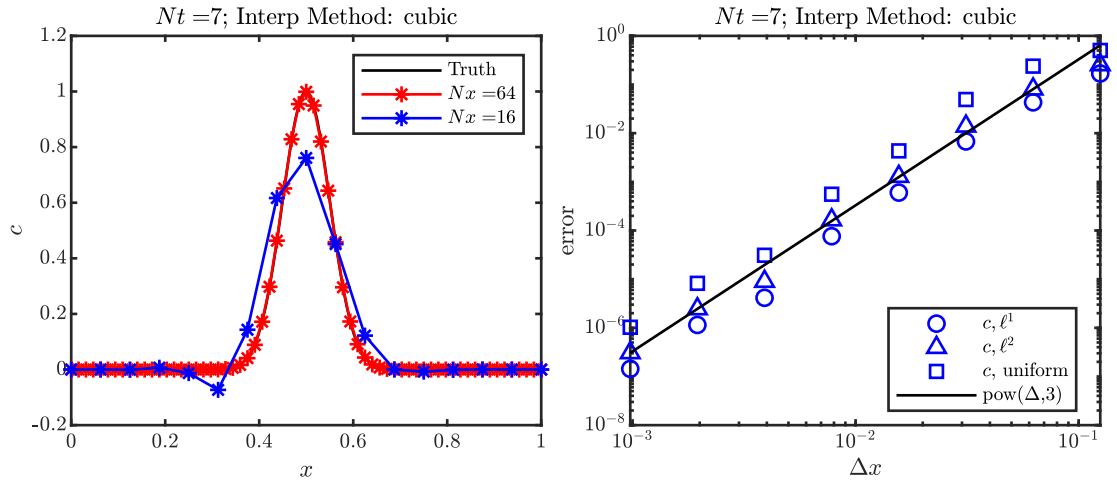

Figure 2:

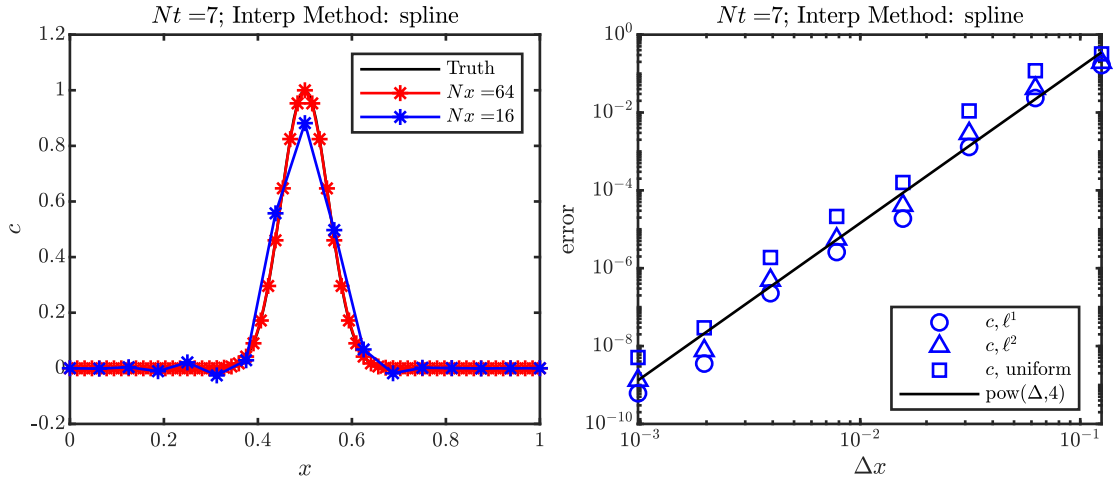

Figure 3:

Next, FINUFFT. Since our data is periodic and on a regular grid, we can calculate its Fourier representation using regular `fft`. Then to evaluate the Fourier series on an nonuniform mesh (for non-constant velocity), we use FINUFFT type 2 [BMaK19]. This means that to interpolate N non-uniform points from N uniform known data, the cost is $O(N \log N)$ instead of $O(N^2)$. In comparison, all other interpolation method presented here is of cost $O(N)$. Spectral interpolation is more expensive, but not prohibitively so. Fig. 4 shows the result. We see that the interpolation using a coarse space grid is still accurate, though there is a small magnitude oscillation on the two side where the truth is constant. The dense space grid result is perfect since for the dense grid we have enough points to fully represent the initial conditions Fourier spectrum. Also note that the order of convergence is indeed spectral.

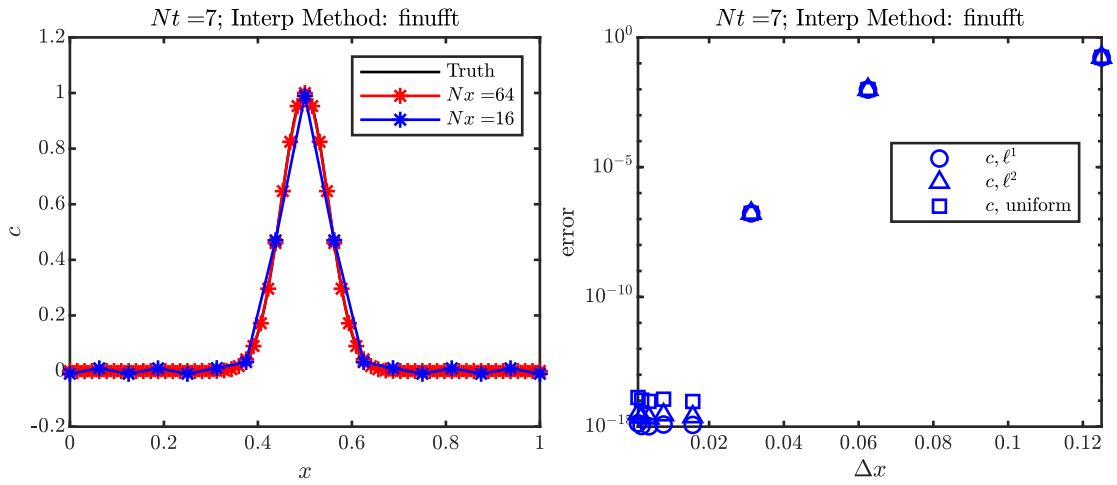

Figure 4:

Finally, we study the Piecewise Cubic Hermite Interpolating Polynomial (`pchip`) [FC80] and Modified Akima Interpolation (`makima`) [Aki70, Aki74]. They are similar in that they are both piecewise polynomial of order the most three, and they are designed to prevent overshoot. Fig. 5 (resp. Fig. 6) shows the result for `pchip` (resp. `makima`). The error is diffusive, it is also dispersive since the peak is asymmetric. But there is no undershoot on the left of the peak compared to `cubic` and `spline` in Fig. 2 and Fig. 3. This is a good feature since a negative tracer concentration c is nonphysical (cf. limiters). The order of convergence of these two methods is not the same as the above methods. The uniform norm convergence order is one less than the convergence order in other norms. This is probably due to the corrections at extrema.

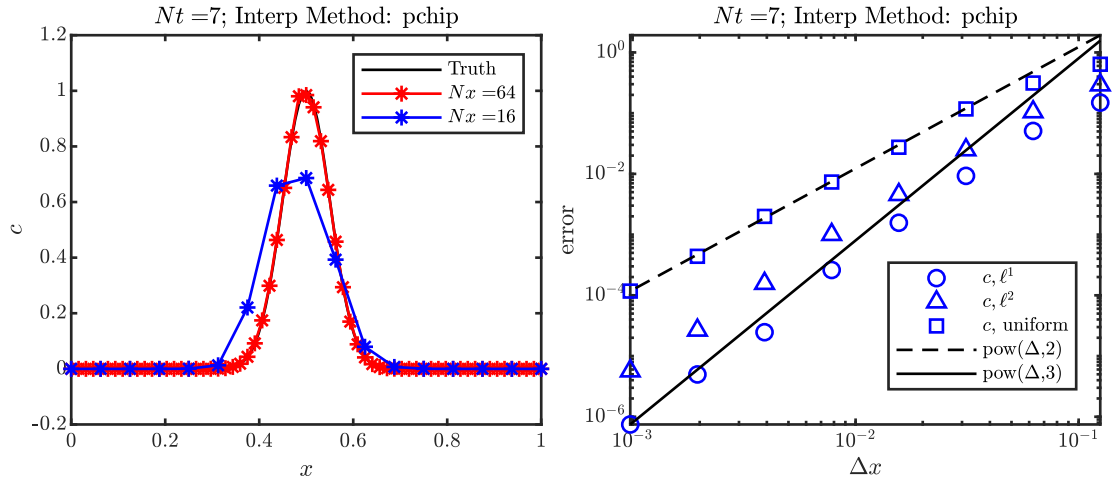

Figure 5:

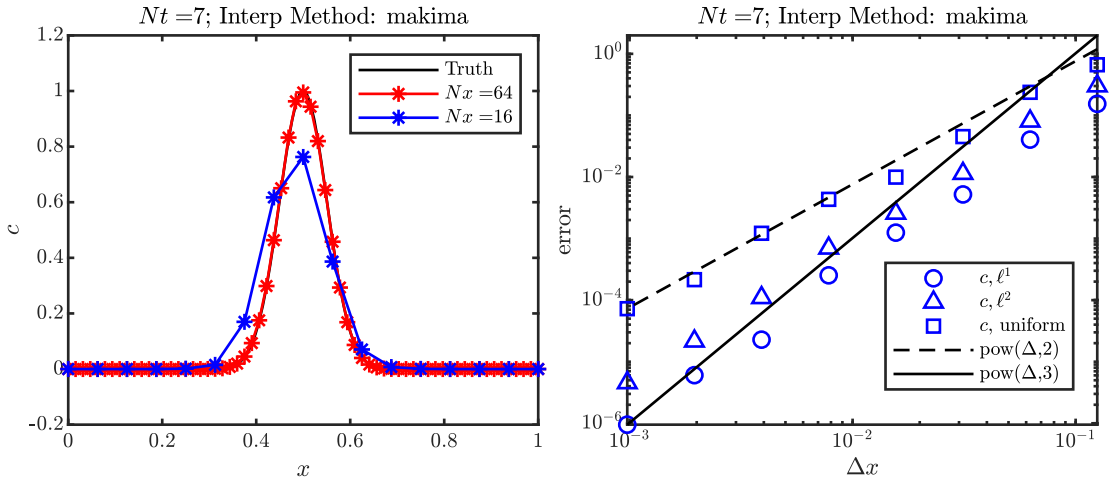

Figure 6:

The truth strength of `phip` and `makima` shows when we try to use this method to advect a step function, and that is what we will experiment on now. We set the initial condition to be a step function:

$$c(x, 0) = \chi_{(1/4,1/2)}.$$

We first show the result for `linear` in Fig. 7. We see that the error is still diffusive, but the order of convergence is no longer one. This is because the convergence theory is based on Taylor series, which clearly does not apply for a step function.

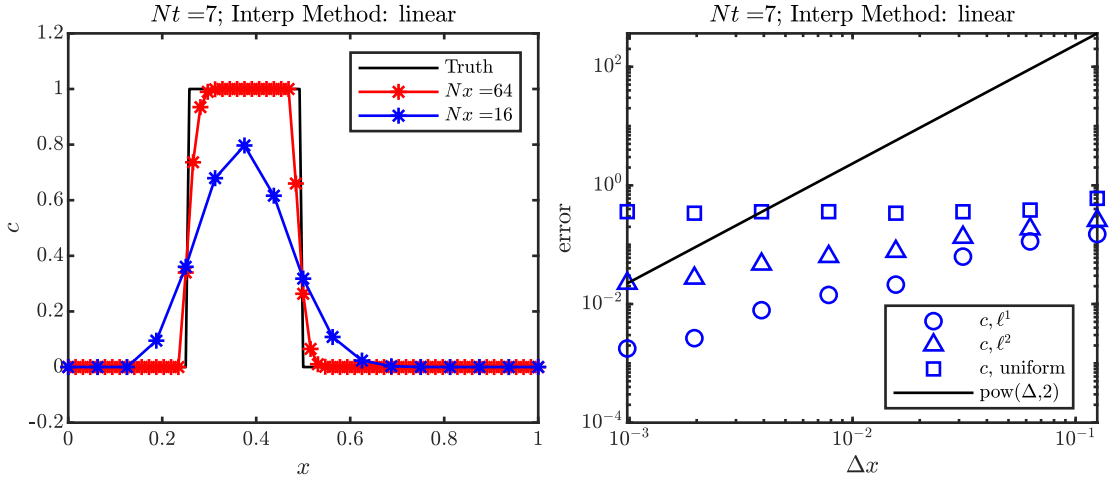

Figure 7:

The order of convergence result does not look too different from the right plot of Fig. 7 for `cubic`, `spline`, `pchip`, and `makima`, thus we will not show them. But the qualitative nature of the error are very different and it is worth looking at and analyzing. `cubic` and `spline` has large oscillation near the discontinuity accompanied with overshoot and undershoot, as shown in Fig. 8.

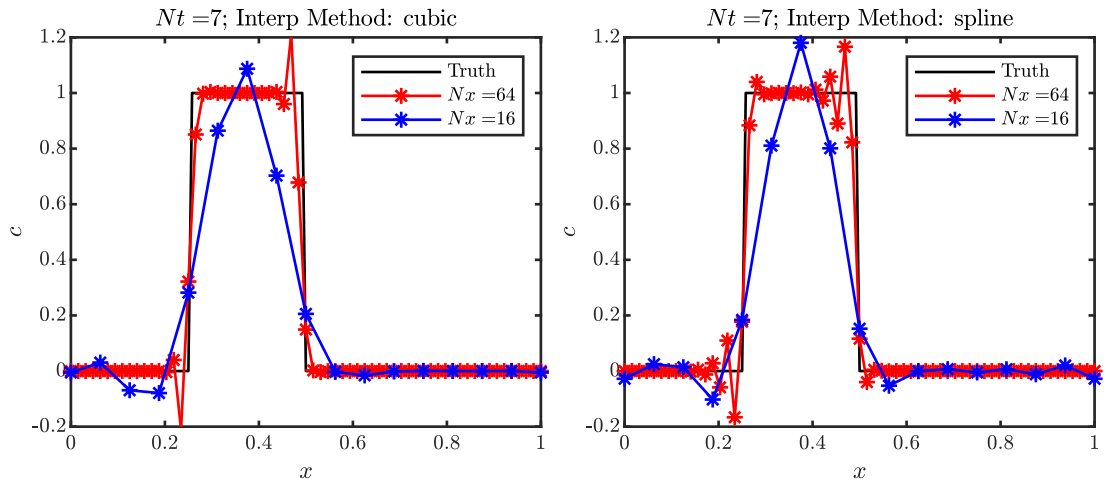

Figure 8:

In comparison `pchip` and `makima` does not have such issue, as shown in Fig. 9. This suggests that these two interpolation schemes are good for solving problems with discontinuities.

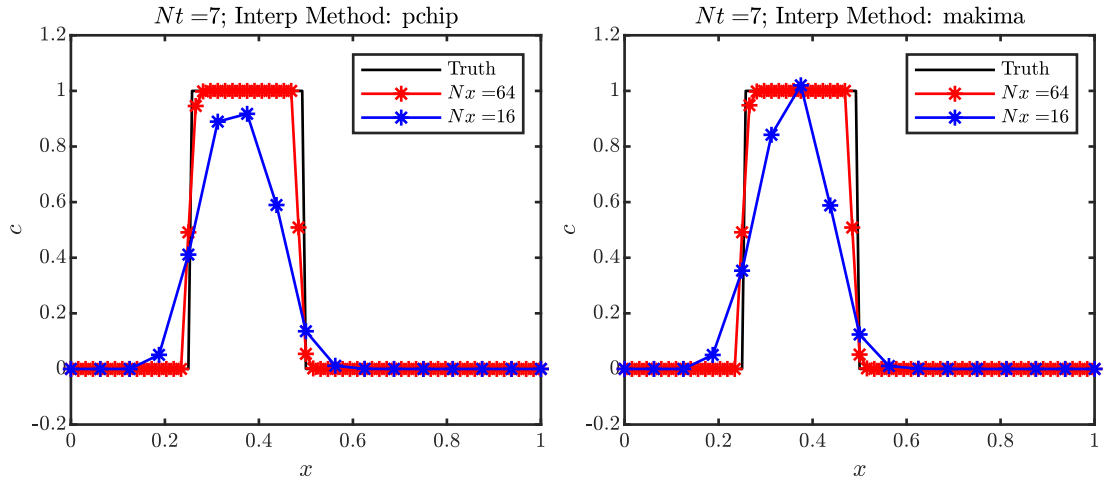

Figure 9:

Finally, we show the result from FINUFFT in Fig. 10. There are oscillations at the smallest scale of the resolution. This is likely due to the fact that for a discontinuous function, the Fourier coefficient decays very slowly. This fact is also represented in the error graph, now in `loglog` form, we can see that the error only converges in first order.

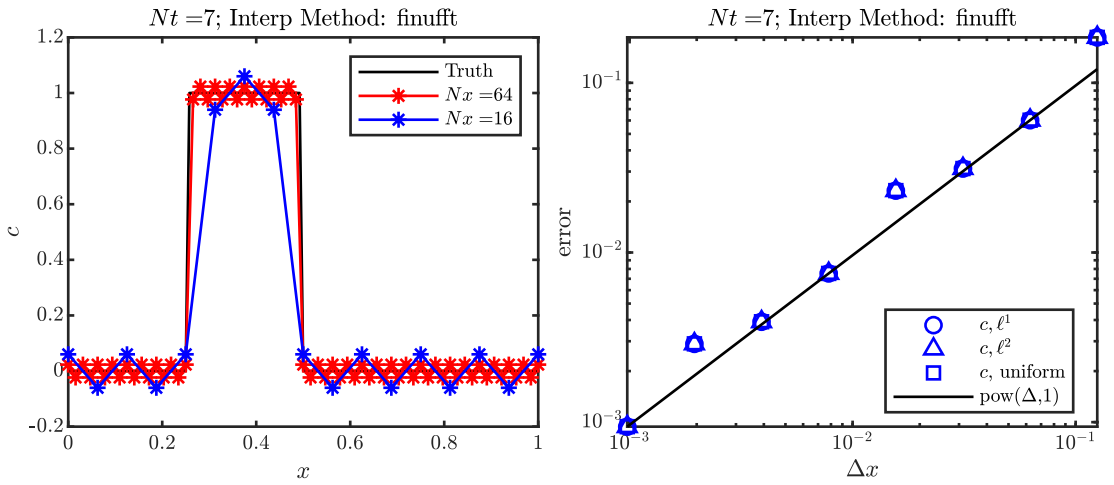

Figure 10:

Wrapping up this section, we have experimentally investigated the nature of the error and the order of convergence of various interpolation methods implemented in MATLAB's `interp1` function as well as a spectral interpolation method. We will use `linear`, `cubic`, `spline`, and FINUFFT for the future sections. We will not use `pchip` and `makima` since they are only significantly different when the solution is discontinuous, but all our future example will be continuous. Also, they will make the order of convergence study of the full Semi-Lagrangian schemes more confusing. But they should be considered when we run into future problems when there are discontinues in the solutions.

3 Semi-Lagrangian method for 2D linear advection

We verify the lesson we learn from Section 2 in 2D. We solve again the constant velocity advection problem with $[0, 1) \times [0, 1)$ with $\boldsymbol{u} = (1, 1)$. We solve to $T = 2$ and compare the solution to the initial condition. We use a 2D trigonometric polynomial as initial condition:

$$c(x, y, 0) = [\sin(\pi x) \sin(\pi y)]^p \qquad \text{with } p = 2. \tag{3}$$

We use $Nt = 15$ and $Nx = Ny = 8 : 256$ (CLF number $= 1.2 : 34$). We show the resultant convergence order estimate in Fig. 11. We see that the order of convergence for `linear`, `cubic`, and `spline` are the same with the result in Section 2. And FINUFFT is accurate for all spatial resolution since our initial condition only needs a few Fourier mode to be represented accurately.

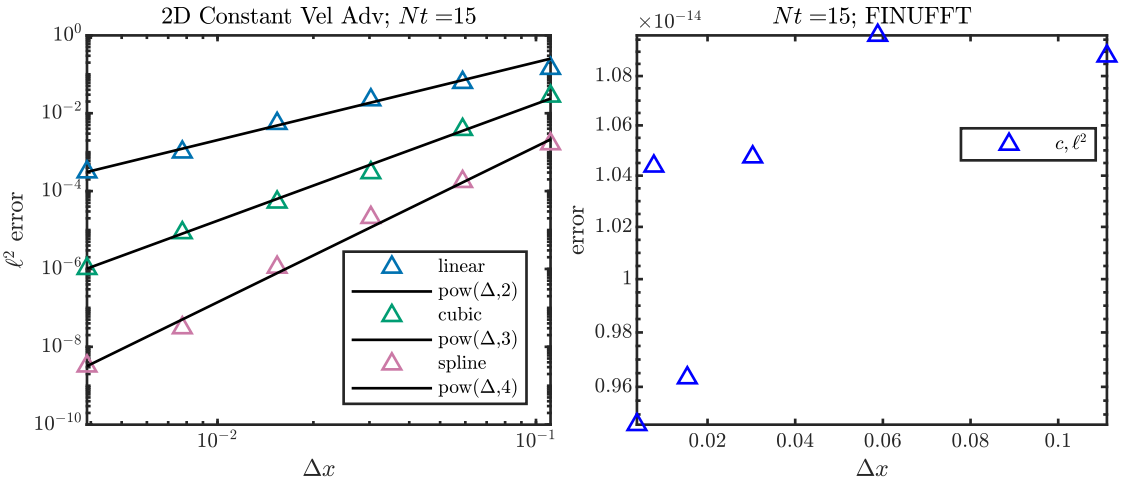

Figure 11:

Now we are familiar with the interpolation step of Semi-Lagrangian methods, we deal with the time integration step. We first focus on solving the linear advection problem where we assume the velocity $\boldsymbol{u}$ is known at all time on the spatial grid. In particular, for the timestep from t^n to t^{n+1}, we know for all x_i on the spatial grid:

$$\begin{aligned} \boldsymbol{u}_i^n &= \boldsymbol{u}(\boldsymbol{x}_i, t^n); \\ \boldsymbol{u}_i^{n+1/2} &= \boldsymbol{u}(\boldsymbol{x}_i, t^{+1/2}); \\ \boldsymbol{u}_i^{n+1} &= \boldsymbol{u}(\boldsymbol{x}_i, t^{n+1}). \end{aligned}$$

With these information, and the "initial" condition that $\boldsymbol{X}(t^{n+1}) = \boldsymbol{x}_i$ a grid point, we need to solve the ODE (2) backward in time to obtain $\hat{\boldsymbol{x}}_i = \boldsymbol{X}(t^n)$. The first method is to use an Euler step:

$$\begin{aligned} & \frac{\boldsymbol{X}(t^{n+1}) - \boldsymbol{X}(t^n)}{\Delta t} = \boldsymbol{u}(\boldsymbol{X}(t^{n+1}), t^{n+1}); \\ \iff \quad & \frac{\boldsymbol{x}_i - \hat{\boldsymbol{x}}_i}{\Delta t} = \boldsymbol{u}(\boldsymbol{x}_i, t^{n+1}). \end{aligned}$$

The error in the departure point estimate $\hat{\boldsymbol{x}}_i^{(1)}$ is $\left\| \hat{\boldsymbol{x}}_i^{(1)} - \hat{\boldsymbol{x}}_i \right\| = O(\Delta t^2)$ (cf. local truncation error for Euler). To match the time error, a second order linear interpolation of c onto $\hat{\boldsymbol{x}}_i^{(1)}$ would suffice. Together the space-time local truncation error $O(\Delta t^2 + \Delta x^2)$. Indeed, name the interpolated tracer to be $\bar{c}$, we have by triangular inequality

$$\begin{aligned} \left\| \bar{c}(\hat{\boldsymbol{x}}_i^{(1)}, t^n) - c(\hat{\boldsymbol{x}}_i, t^n) \right\| &\le \left\| \bar{c}(\hat{\boldsymbol{x}}_i^{(1)}, t^n) - c(\hat{\boldsymbol{x}}_i^{(1)}, t^n) \right\| + \left\| c(\hat{\boldsymbol{x}}_i^{(1)}, t^n) - c(\hat{\boldsymbol{x}}_i, t^n) \right\| \\ &\le O(\Delta x^2) + \text{Const} \cdot \left\| \hat{\boldsymbol{x}}_i^{(1)} - \hat{\boldsymbol{x}}_i \right\| = O(\Delta x^2 + \Delta t^2). \end{aligned}$$

Now global error of this method is one less than the local truncation error (multiply by $O(\Delta t^{-1})$), thus we have a first order space-time accurate method.

We test the order of convergence on a sample problem. We use the inviscid Taylor vortex as our

underlying velocity ($v_0 = 1, L = 2\pi$):

$$u = v_0 - \cos\left(\frac{2\pi(x - v_0 t)}{L}\right)\sin\left(\frac{2\pi(y - v_0 t)}{L}\right)$$
$$v = v_0 + \sin\left(\frac{2\pi(x - v_0 t)}{L}\right)\cos\left(\frac{2\pi(y - v_0 t)}{L}\right). \tag{4}$$

And the initial tracer distribution consists of three Gaussian peaks:

$$c(x, y, 0) = \exp\left(-5\left((x-\pi)^2 + (y - 3\pi/4)^2\right)\right) + \exp\left(-5\left((x-\pi)^2 + (y - 5\pi/4)^2\right)\right)$$
$$- \frac{1}{2}\exp\left(-\frac{5}{2}\left((x - 5\pi/4)^2 + (y - 5\pi/4)^2\right)\right). \tag{5}$$

We solve the problem numerically till $T = 0.5$ and empirically estimate the error. We can also compare the solution to the "truth" solution obtained by using the Pseudospectral IF-RK4 method from HW3. The convergence order plots look the same and we will only show the empirical estimates. First, Fig. 12 shows the initial distribution of the tracer and a "truth" solution from Pseudospectral IF-RK4:

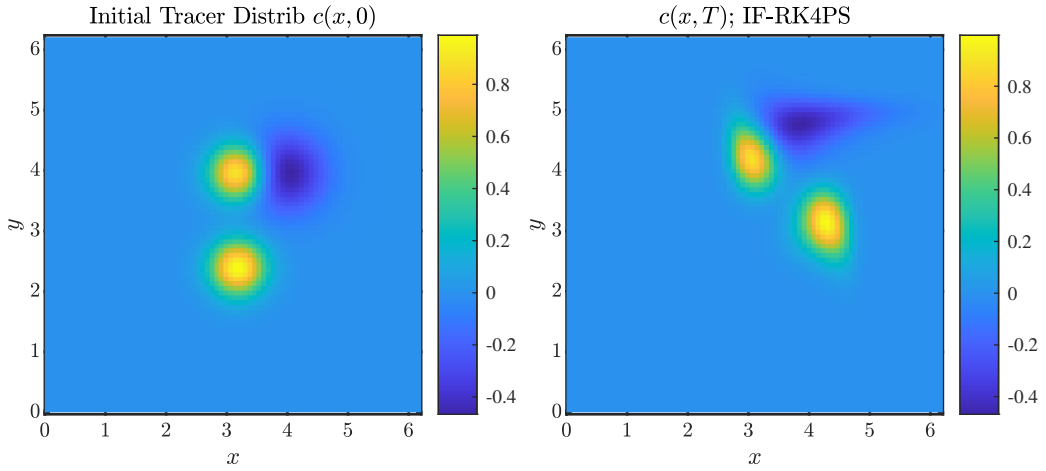

Figure 12:

Then Fig. 13 shows on the left the tracer concentration that is solved by Euler Method at the finest space-time resolution and on the right the order of convergence from the empirically estimated error. Here we use $Nx = Ny = [2^5 : 2^{10}]$ but large timestep such that the CFL number $v_0\Delta t/\Delta x \approx 1.5$. This means $Nt = 3 : 55$, not many timesteps!

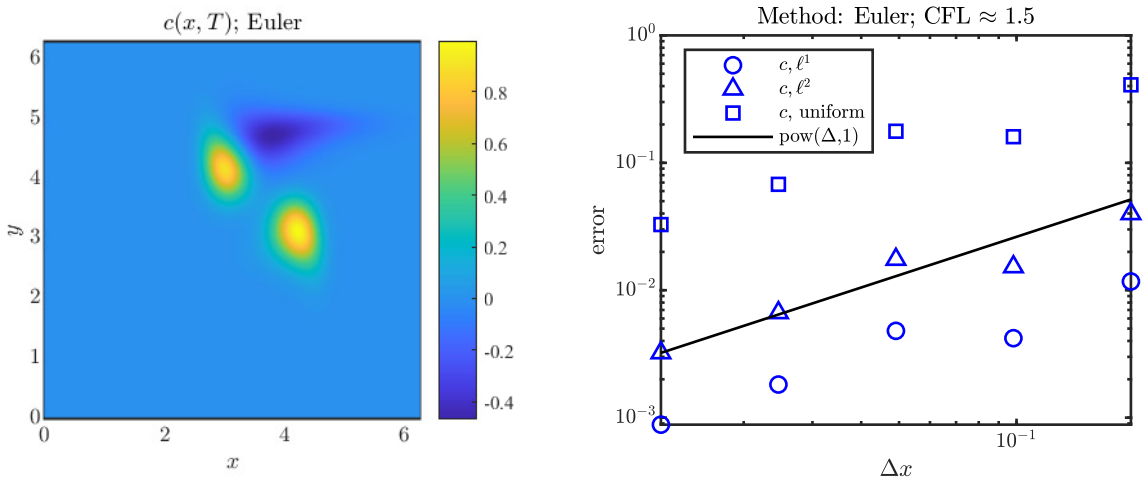

Figure 13:

A higher order method for solving ODE is the trapezoidal method. To generalize it to Semi-Lagrangian method, we need to estimate the velocity at the departure point at t^n: $\boldsymbol{u}(\hat{\boldsymbol{x}}_i, t^n)$. We write the trapezoidal rule for the ODE (2):

$$\begin{aligned} & \boldsymbol{X}(t^{n+1}) - \boldsymbol{X}(t^n) = \frac{\Delta t}{2}\left[\boldsymbol{u}(X(t^n), t^n) + \boldsymbol{u}(X(t^{n+1}), t^{n+1})\right] \\ \iff \quad & \boldsymbol{x}_i - \hat{\boldsymbol{x}}_i = \frac{\Delta t}{2}\left[\boldsymbol{u}(\hat{\boldsymbol{x}}_i, t^n) + \boldsymbol{u}(\boldsymbol{x}_i, t^{n+1})\right]. \end{aligned}$$

We have in our hand for each i a nonlinear equation, we can solve this via fix-point iterations:

$$\hat{\boldsymbol{x}}_i^{(k+1)} = \boldsymbol{x}_i - \frac{\Delta t}{2}\left[\boldsymbol{u}(\hat{\boldsymbol{x}}_i^{(k)}, t^n) + \boldsymbol{u}(\boldsymbol{x}_i, t^{n+1})\right]$$

and we start the iteration by estimating $\hat{\boldsymbol{x}}_i^{(0)} = \hat{\boldsymbol{x}}_i$. As it turns out, to reach global second order accuracy and local third order accuracy, the highest that trapezoidal rule could reach, we only need to do one fixed point iterations and obtain $\hat{\boldsymbol{x}}_i^{(1)}$. Then we have the tracer field at time t^{n+1} (remember that over-line means interpolated result)[3]

$$\begin{aligned} \hat{\boldsymbol{x}}_i^{(2)} &= \boldsymbol{x}_i - \frac{\Delta t}{2}\left[\boldsymbol{u}(\hat{\boldsymbol{x}}_i^{(1)}, t^n) + \boldsymbol{u}(\boldsymbol{x}_i, t^{n+1})\right] \\ c(\boldsymbol{x}_i, t^{n+1}) &= \overline{c}(\hat{\boldsymbol{x}}_i^{(2)}, t^n). \end{aligned}$$

A similar bound using triangular inequality is also possible for this method and it shows that we need a third order interpolation for interpolating the tracer field, thus we use `cubic`. We also need to interpolate to get the velocity field at $\hat{\boldsymbol{x}}_i^{(1)}$, as it turns out we only need second accuracy for that, thus we use `linear`. This fact is mentioned in [SC91]:

> "For step (i), the order of the interpolation is much less important. Theoretically, McDonald (1987) has shown that one should use an interpolation of order one less than for step (ii)."

[3]This is just two fixed point iterations, but separating out the last evaluation from the fixed point iteration makes the concept of iteration number more consistent with the next MCD86 method.

Here McDonald (1987) refers to [McD87][4]. It is important to use the cheapest interpolation scheme possible since interpolation is the main cost of Semi-Lagrangian methods. Now we show result, Fig. 14 shows the final tracer field on the left and the order of convergence result on the right. We see that the tracer fields look the same with the "truth" and the order of convergence is as expect, second order.

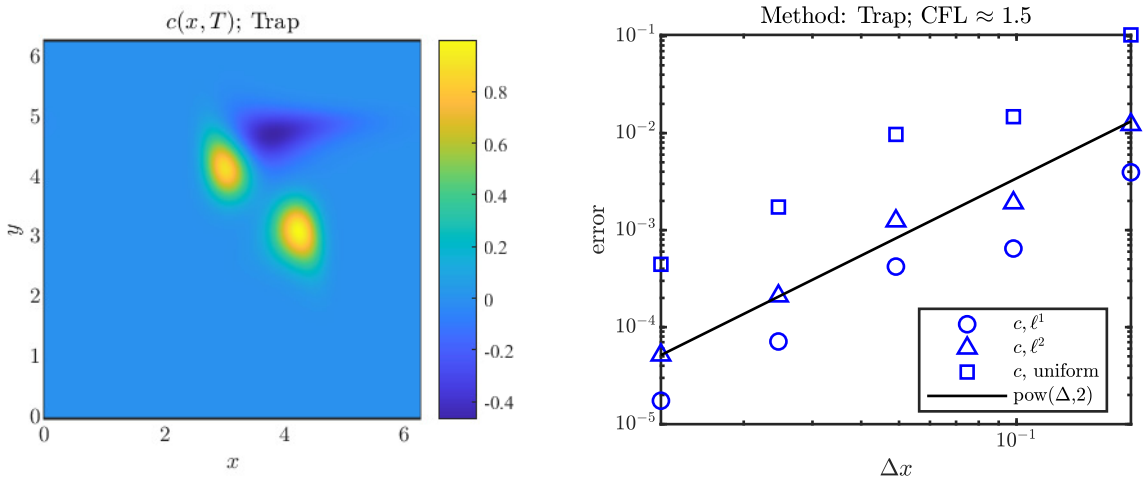

Figure 14:

Note that there are many other global space-time second order method, for example ones based on mid-point rule. Some examples are given in [McD99] and in there the author provide a general formula for second order accurate schemes. I am not sure what are the differences between all of them and I only investigate this trapezoidal method because I thought it is the most conceptually simplistic.

[McD87] presents a global space-time third order accurate method. We first solve two nonlinear equations:

$$\begin{aligned}
\boldsymbol{X}(t^{n+1}) - \boldsymbol{X}(t^{n+1/2}) &= \frac{\Delta t}{2}\left[\boldsymbol{u}(X(t^{n+1/2}), t^{n+1/2})\right]; \\
\boldsymbol{X}(t^{n+1}) - \boldsymbol{X}(t^{n}) &= \Delta t\left[\boldsymbol{u}(X(t^{n}), t^{n})\right].
\end{aligned}$$

It turns out we need to do two iterations, then we evaluate the midpoint rule and get two departure points at two time levels:

$$\begin{aligned}
\boldsymbol{p}^n &= \boldsymbol{X}(t^{n+1}) - \Delta t \cdot \boldsymbol{u}\left(\boldsymbol{X}(t^{n+1/2})^{(2)}\right); \\
\boldsymbol{q}^{n-1} &= \boldsymbol{X}(t^{n+1}) - 2\Delta t \cdot \boldsymbol{u}\left(\boldsymbol{X}(t^{n})^{(2)}\right).
\end{aligned}$$

Note that we know $\boldsymbol{X}(t^{n+1}) = \boldsymbol{x}_i$. We interpolate to have the tracer field that those position

$$c^n_{\boldsymbol{p}} = c(\boldsymbol{p}, t^n); \qquad\qquad c^{n-1}_{\boldsymbol{q}} = c(\boldsymbol{q}, t^{n-1}).$$

Finally, we obtain the tracer field at t^{n+1} via

$$c^{n+1} = \frac{1}{7}\left(8c^n_{\boldsymbol{p}} - c^{n-1}_{\boldsymbol{q}}\right).$$

[4]Note that [McD87] = MCD86. The date on the paper is 1986 but the meta data gives the year 1987. I have coded the method in this paper calling it MCD86, so we will use MCD86 in the report to refer to this paper as well.

This method reaches global third order due to a cancellation of error from the two separate midpoint calculations (see details of the calculation in [McD87]). But this cancellation is not transparent unless we write out the error estimates for both. We again need to specify what order of interpolation is needed. For the final interpolation of the tracer field to obtain $c_{\boldsymbol{p}}^n$ and $c_{\boldsymbol{q}}^{n-1}$, we need fourth order, thus `spline` is used. We also need to interpolate $\boldsymbol{u}$ to $\boldsymbol{X}^{(1)}$ and $\boldsymbol{X}^{(2)}$. We need second and third order accuracy respectively. Thus we use `linear` and `cubic`. All together, we have global third order space-time convergence. And indeed, Fig. 15 confirms this.

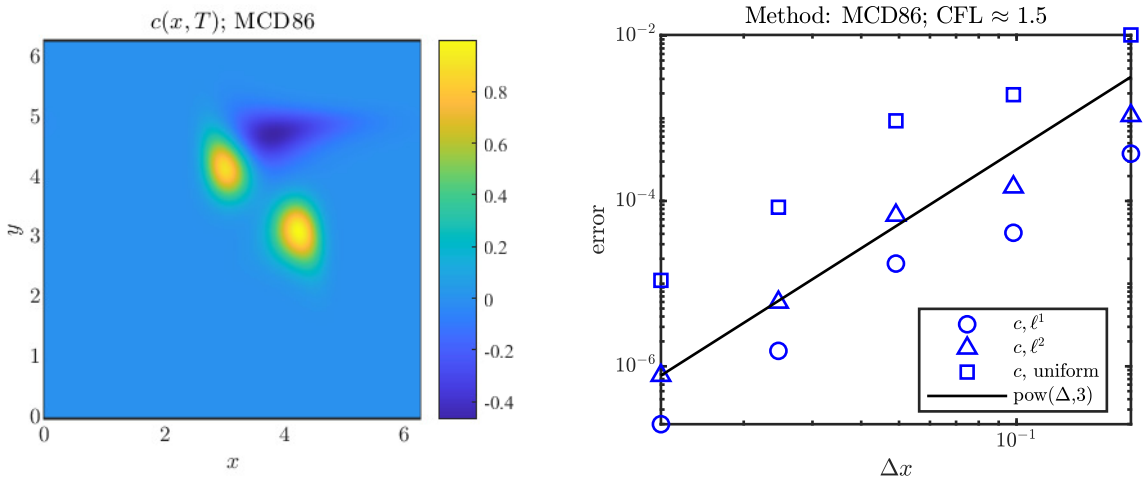

Figure 15:

The MCD86 method reaches high order convergence but it uses an opaque cancellation of error. High order accuracy can be reach for the linear advection problem more easily by using readily available ODE solver on (2). For example, we could use RK4 method to reach 4th order accuracy. We have RK4 applied to (2):

$$\begin{aligned}
\boldsymbol{x}^1 &= \boldsymbol{x}_i; & \boldsymbol{u}^1 &= \boldsymbol{u}(x^1, t^{n+1}) \\
\boldsymbol{x}^2 &= \boldsymbol{x}_i - \frac{\Delta t}{2}\boldsymbol{u}^1; & \boldsymbol{u}^2 &= \boldsymbol{u}(x^2, t^{n+1/2}) \\
\boldsymbol{x}^3 &= \boldsymbol{x}_i - \frac{\Delta t}{2}\boldsymbol{u}^2; & \boldsymbol{u}^3 &= \boldsymbol{u}(x^2, t^{n+1/2}) \\
\boldsymbol{x}^4 &= \boldsymbol{x}_i - \Delta t \boldsymbol{u}^3; & \boldsymbol{u}^4 &= \boldsymbol{u}(x^4, t^n)
\end{aligned}$$

then

$$\boldsymbol{u}^s = (\boldsymbol{u}^1 + 2\boldsymbol{u}^2 + 2\boldsymbol{u}^3 + \boldsymbol{u}^4)/6.$$

and

$$c(\boldsymbol{x}_i, t^{n+1}) = c(\boldsymbol{x}_i - \boldsymbol{u}^s \Delta t, t^n).$$

Now that the departure point is solve to fifth order locally, we need a fifth order interpolation as well. We instead just use FINUFFT which is spectral. For all other interpolations within the RK4 steps, we could use one less order, thus `spline` is used. Fig. 16 shows good convergence result.

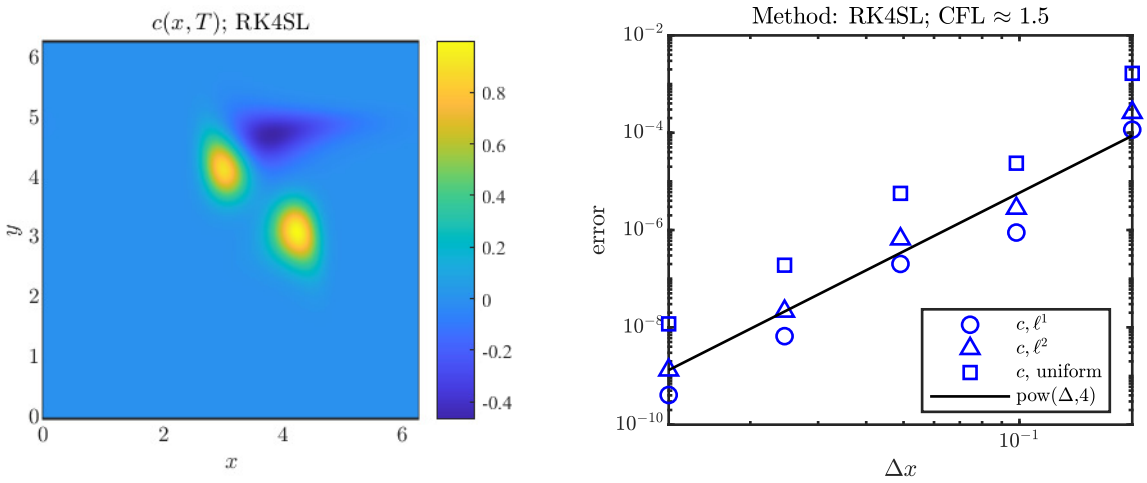

Figure 16:

It is curious why Semi-Lagrangian method literature does not use RK method to solve for the departure point. We check the cost by counting the number of interpolations and their requires order. We compare RK3, which should be third order, with MCD86: RK3 needs three third and a forth order interpolation, while MCD86 needs two second, two third, and two forth order interpolation. It seems MCD86 is more expensive. We comment that the second and third RK4 step looks like a fixed point solver, and it might be that the RK4 method and classic Semi-Lagrangian methods are more similar than the form suggests. We will continue the discussion in Section 4 for nonlinear advection.

We could replace all the interpolation with FINUFFT so that they are spectrally accurate. We give a large enough $Nx = Ny$ so that there is no spatial error (for 3 Gaussian, $Nx = Ny = 81$). Then we can fix the spatial resolution and improve accuracy by only refining the timestep. This is a good way to isolate the error produced by the timestepping. But this is also viable numerical method. In a sense, the method is pseudospectral but with with Semi-Lagrangian timestepping. This has real advantage compared to the regular pseudospectral method for advection since it could take a lot bigger a timestep while still being stable. For example, to solve the problem above, Semi-Lagrangian Pseudospectral method /w FINUFFT is stable for CFL number up to 3 while the regular IF-RK4 Pseudospectral method is only stable when CFL number is less than 0.5. Fig. 17 shows the convergence order result for Semi-Lagrangian Pseudospectral and IF-RK4 Pseudospectral methods. We see that all the convergence order of Semi-Lagrangian method stay the same. In particular, the error for the most accurate result for Semi-Lagrangian RK4 Pseudospectral (RK4SLPs) method is $O(10^{-9})$, very small! And that is achieved with 54 timestep (i.e.: $\Delta t = 0.0093$). IF-RK4 Pseudospectral method does not each the same level of accuracy with same number of timestep (the x-axis of the IF-RK4 figure is large than the rest). To truly compare the two methods, we also need to know the cost per timestep. Assuming that FINUFFT cost as much as a standard uniform Fast Fourier Transform (FFT), we approximate the cost by counting the number of `fft/ifft` pairs per timestep. For IF-RK4, assume that the velocity is given in Fourier space and we solve for the tracer in Fourier space, we only need four pairs of `fft/ifft` per timestep. The RK4SLPs method needs four u, v interpolation and one tracer field interpolation, and that is $4 \times 2 + 1 = 9$ `fft/ifft` pairs. Therefore we could say a RK4SLPs step is as expensive as two IF-RK4 steps. But compare the stability and accuracy of the two methods shown in Fig. 17, we see that IF-RK4 needs more than two times the timestep to reach the same accuracy. Therefore

we conclude that RK4SLPs is cheaper as a solver for linear advection[5].

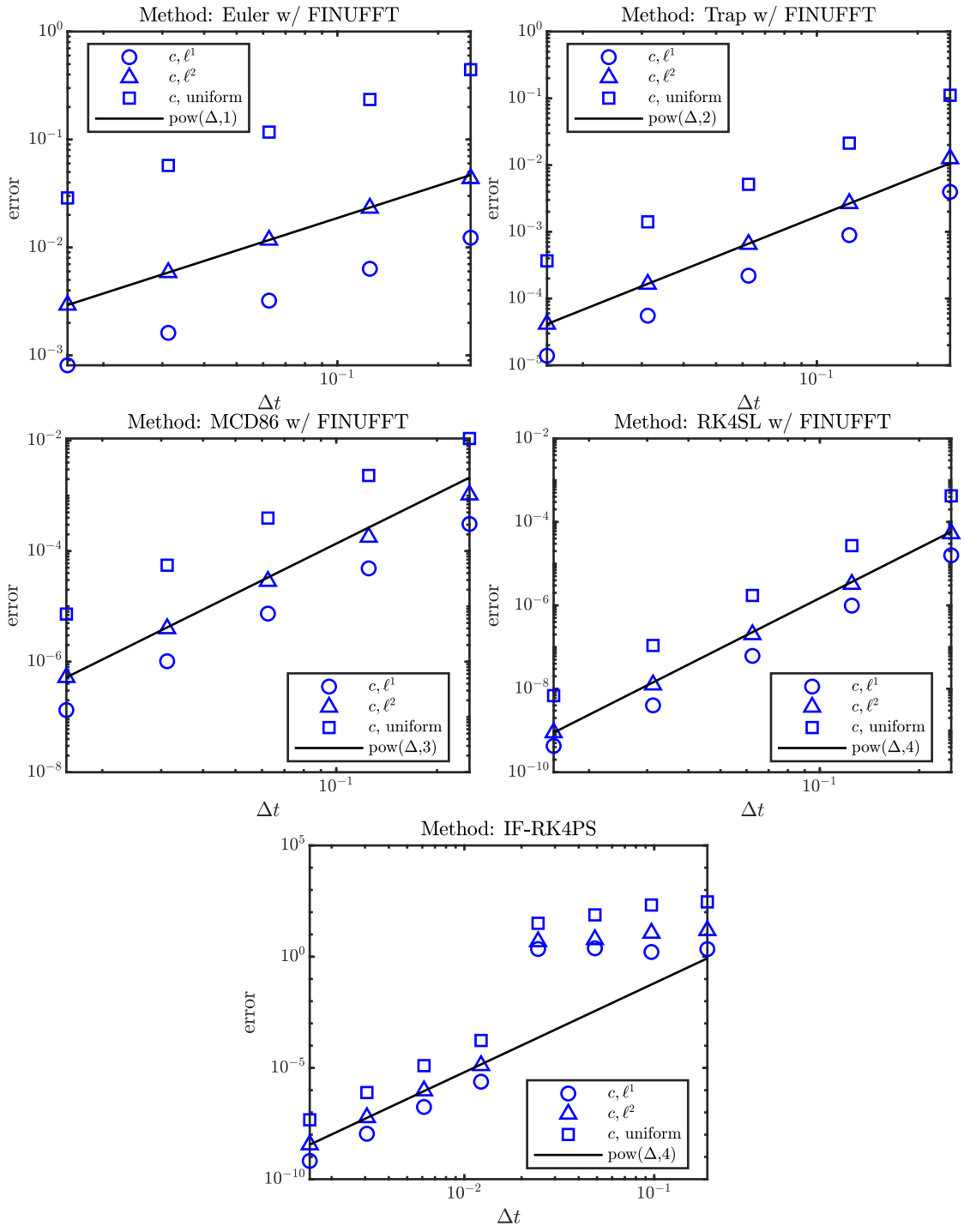

Figure 17:

[5]Comment from Alex Donev: Pseudospectral methods could be combined with spectral deferred correction (SDC) to achieve much better accuracy in Δt. It is also worth noting that NUFFT only works for periodic functions on periodic domains. These factors should also be considered when we make the comparison

4 Semi-Lagrangian method for 2D Navier Stokes

We use Semi-Lagrangian methods to numerically solve the incompressible inviscid Navier-Stokes equation:

$$\begin{cases} \partial_t \boldsymbol{u} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} = -\nabla p \\ \nabla \cdot \boldsymbol{u} = 0. \end{cases} \tag{6}$$

This is possible because we could reformulate the equation in the vorticity-stream formulation, which is an advection equation with vorticity as the tracer. First, due to incompressibility, we could define the streamfunction ψ and the vorticity ω (u is the x-component of $\boldsymbol{u}$ and v is the y-component of $\boldsymbol{u}$):

$$\begin{aligned} -\partial_y \psi = u, \quad \partial_x \psi = v; \\ \omega = -\partial_y u + \partial_x v = \nabla^2 \psi. \end{aligned} \tag{7}$$

Then the vorticity-stream formulation is:

$$\begin{aligned} & \partial_t \omega + \boldsymbol{u} \cdot \nabla \omega = 0 \\ \iff \quad & \partial_t \omega + J(\psi, \omega) = 0 \end{aligned} \tag{8}$$

and this equation is closed since $\boldsymbol{u}, \psi$ can be solved from ω via (7). This is a nonlinear advection problem since the velocity that advect tracer field needs to be solved from the tracer field ω. Therefore for each timestep we only know the velocity information at time t^n.

To test our Semi-Lagrangian schemes, we use them to solve the 2D Navier-Stokes problem with two different initial condition. First, we initialize the ω field to be the one associated with the Taylor vortex (4). We use $L = 1$ and $v_0 = 1$ and solve till time $T = 0.25$. To obtain the error of the solver, we compare the error with the truth ω given by the formula. The second initial condition is the three Gaussians (now three vortices) given by (5). For this we use $L = 2\pi$ and $v_0 = 1$ and solve till time $T = 1.5$. We estimate the error empirically for this case.

Now we show the result for the Semi-Lagrangian schemes. Fig. 18 shows the result for Euler timestepping. For Euler, only velocity information at t^n is needed and no change is needed from our previous description of the method. The first row are results for the Taylor initial condition while the second row is for the three vortices initialization. Left column is the classic Semi-Lagrangian methods where we refine space and time together while keeping the CFL number about 1.5. The right column we use FINUFFT for the interpolation, thus we keep the space resolution constant ($Nx = Ny = 9$ for Taylor and $Nx = Ny = 33$ for three vortices) and refine only time. All four plots shows the expected first order convergence.

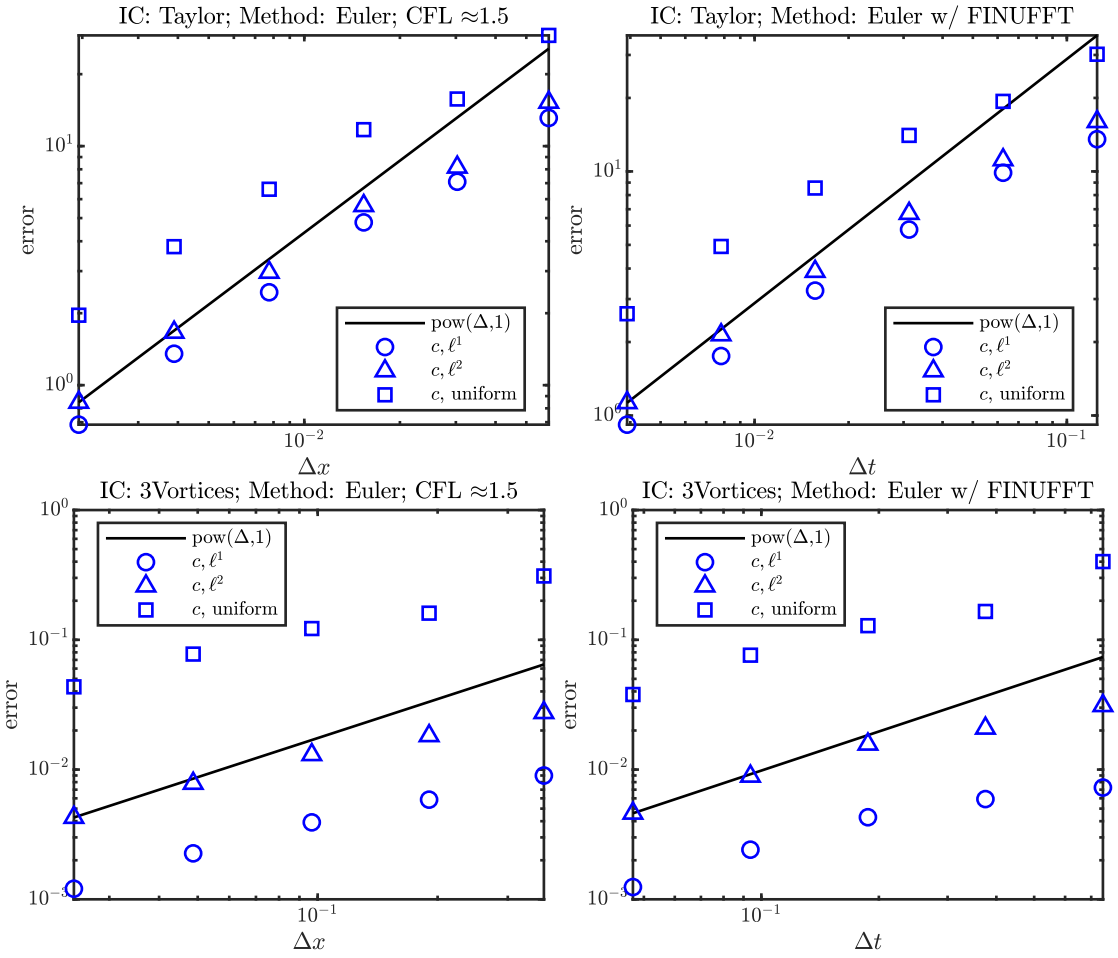

Figure 18:

Now for the trapezoidal method, we need an estimate of the velocity at t^{n+1}. To do this we extrapolate it from the velocity at t^n and t^{n-1} via

$$\boldsymbol{u}(t^{n+1}) = 2\boldsymbol{u}(t^n) - \boldsymbol{u}(t^{n-1}).$$

Then we have $\boldsymbol{u}$ available on the grid at time t^{n+1}. But now we need the information for the velocity at t^{n-1}, so for the first step, we call the Euler method instead. Now we can proceed as we have stated in Section 3. Fig. 19 shows convincing second order convergence.

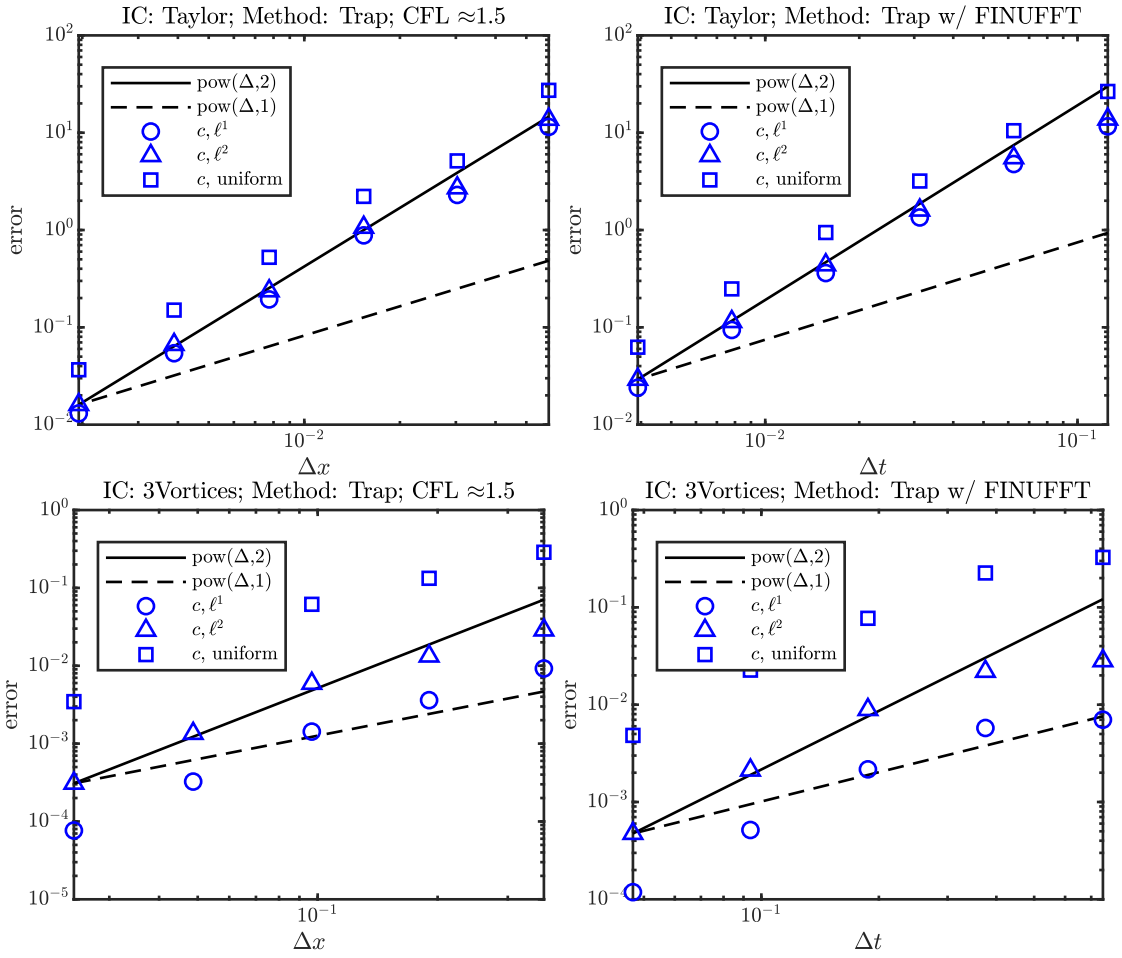

Figure 19:

The MCD86 method needs the estimate of the velocity at the half timestep $t^{n+1/2}$. Since we want global third order accuracy, the order of the extrapolation needs to be improved as well compared to one used in the trapezoidal method. For this [McD87, (19)] uses an extrapolation stated in [TS87], but this is just the standard polynomial extrapolation

$$\boldsymbol{u}(t^{n+1/2}) = \frac{1}{8}\left(15\boldsymbol{u}(t^n) - 10\boldsymbol{u}(t^{n-1}) + 3\boldsymbol{u}(t^{n-2})\right).$$

For the first two steps where we do not have the velocity information at t^{n-1} and t^{n-2}, we call the trapezoidal method. With this estimate, we could proceed with the method as stated in Section 3. Fig. 20 shows good third order convergence result.

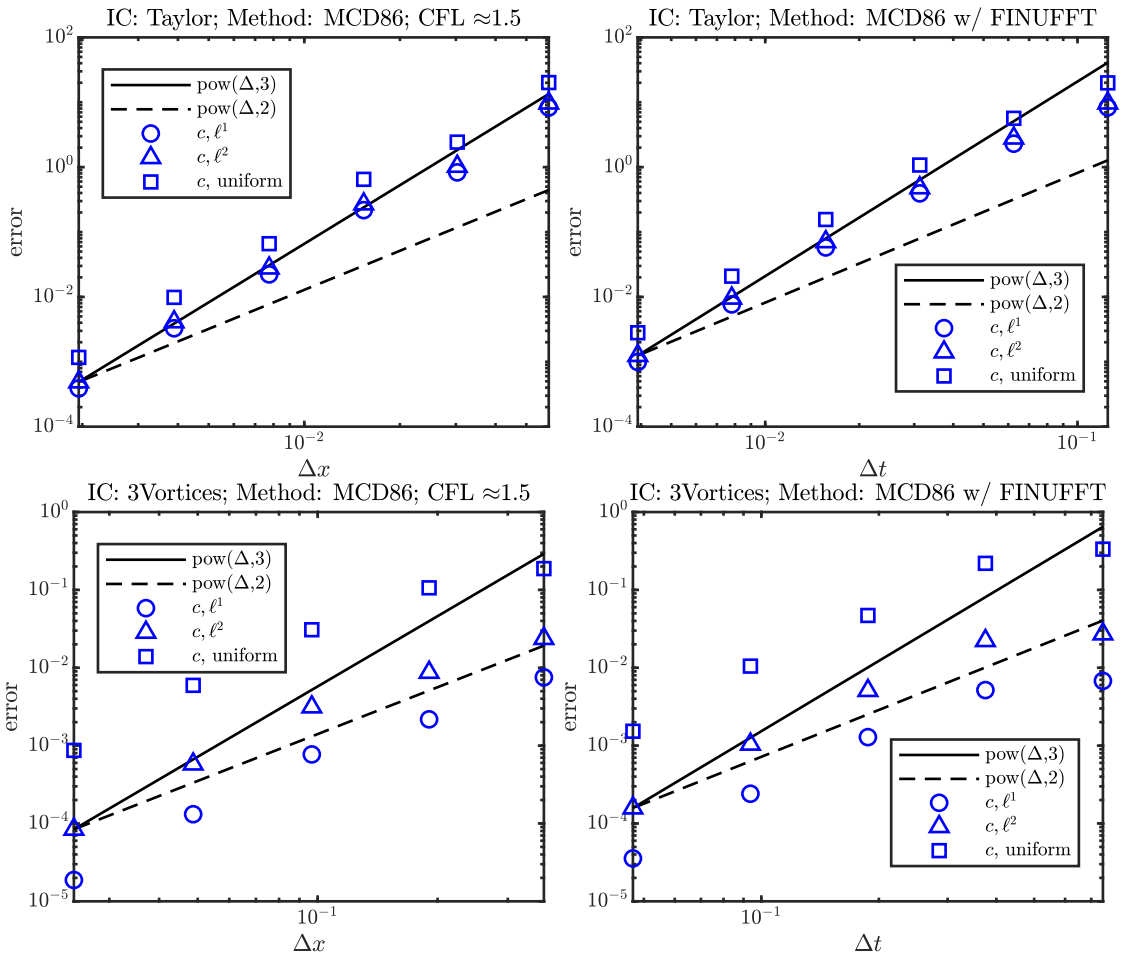

Figure 20:

The story for RK4 method is less straightforward. We could not reach fourth order convergence. This is because a fourth order extrapolation using the information at t^n to t^{n-3} does not appear to be stable, in the sense that with the fourth order extrapolation for $\boldsymbol{u}(t^{n+1/2}), \boldsymbol{u}(t^{n+1})$, the solution does not converge. So instead we use the same third order polynomial extrapolation as the one used in MCD86 method:

$$\boldsymbol{u}(t^{n+1/2}) = \frac{1}{8}\left(15\boldsymbol{u}(t^n) - 10\boldsymbol{u}(t^{n-1}) + 3\boldsymbol{u}(t^{n-2})\right);$$
$$\boldsymbol{u}(t^{n+1}) = 3\boldsymbol{u}(t^n) - 3\boldsymbol{u}(t^{n-1}) + \boldsymbol{u}(t^{n-2}).$$

Initialize the first few steps with MCD86, then we can proceed with the description of RK4 from Section 3. But the result is only third order as shown in Fig. 21.

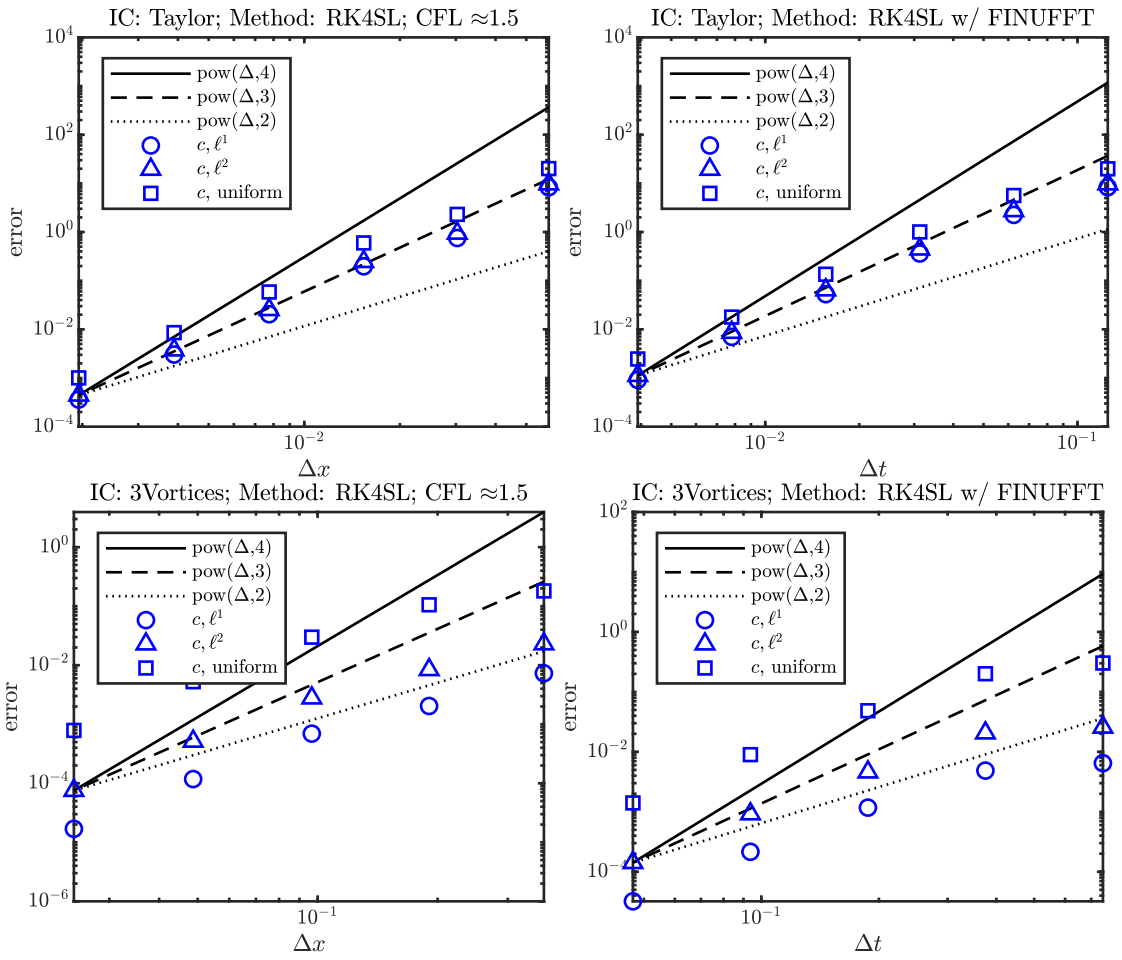

Figure 21:

We continue the discussion of using RK method in Semi-Lagrangian setting. The above result is a discouragement, but we still argue that RK3 is an strong alternative to MCD86. Only needing third order extrapolation, RK3 could reach third order convergence as RK4 above. And we have determined that RK3 is cheaper than MCD86 per timestep.

In general, for nonlinear advection problem, extrapolation to get an estimate of the velocity field seems sketchy to me even when it seems that it works for low order extrapolation with moderate timestep (but still CFL number larger than one). And it is not a good idea to push the order higher, as shown in the RK4 example. I am interested in if there is an alternative high (larger than three) order Semi-Lagrangian method for nonlinear advection.

5 Semi-Lagrangian method for 2D linear advection, cont.

We finish the note by solving linear advection problems where the velocity is obtained numerically. Given the disappointing result of Semi-Lagrangian method when applied to nonlinear advection, we choose to stick with IF-RK4 method to solve the velocity. But for advection of the tracer, we use RK4SLPs to solve. This work well with the pseudospectral IF-RK4 since we can keep space

resolution fixed while improve the accuracy by refining time. Since Semi-Lagrangian method allows for larger timestep than IF-RK4, we use four steps of IF-RK4 to solve velocity within each single step of tracer solve using RK4SLPs. Note here that we record the velocity at middle time $t^{n+1/2}$ and end time t^{n+1} for use in the tracer solve.

First we solve the problem we had in Section 3 but now with the velocity numerically solved. We initialize the ω field with the Taylor vortices (4) and the tracer field as three Gaussians (5). We again have $L = 2\pi, v_0 = 1$, and we solve still $T = 0.5$. We fix $Nx = 81$ and refine $Nt = 2 : 32$. We save the data from a high fidelity (estimated error less than 10^{-14}) result from the IF-RK4 method in Section 3 as the "truth" to calculate the error. Fig. 22 left shows the final tracer field for the solute with the finest time resolution, and the right shows the convergence result. We see that the final tracer field is similar to the ones we have seen in Section 3, this is reassuring. And we see fourth order convergence once the timestep is small enough for the IF-RK4 velocity solver. We note here that a potential fourth order Semi-Lagrangian nonlinear advection solver should remove this barrier.

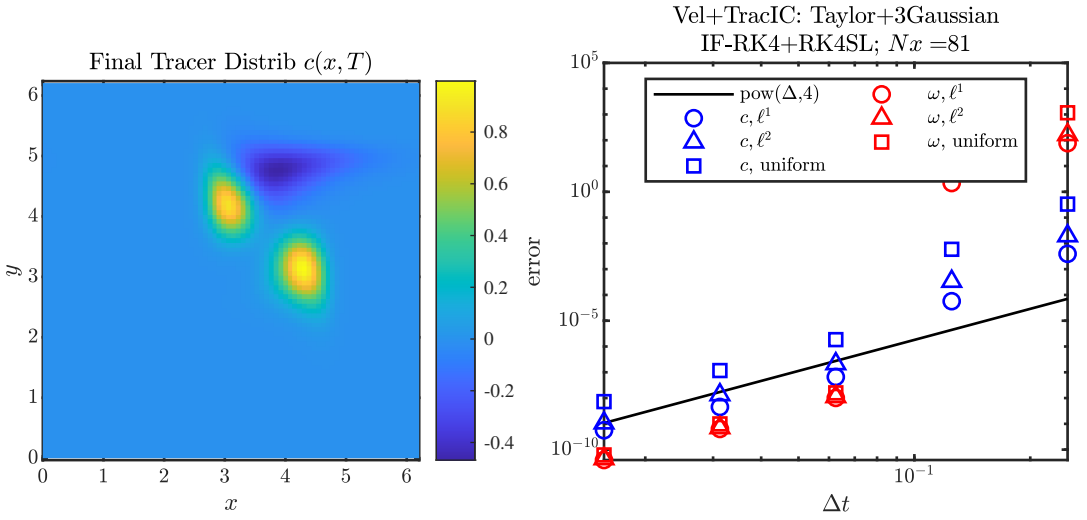

Figure 22:

Now we solve the problem where the velocity is initialize with three vortices (5), and the tracer has initial condition sinp (3). Here we use empirical error estimate. We set $L = 2\pi, v_0 = 1$ but now solve till $T = 4$. We fix $Nx = 49$. Fig. 23 shows the result. Essentially the same sized Δt is needed to keep IF-RK4 stable. Then we have convincing fourth order convergence.

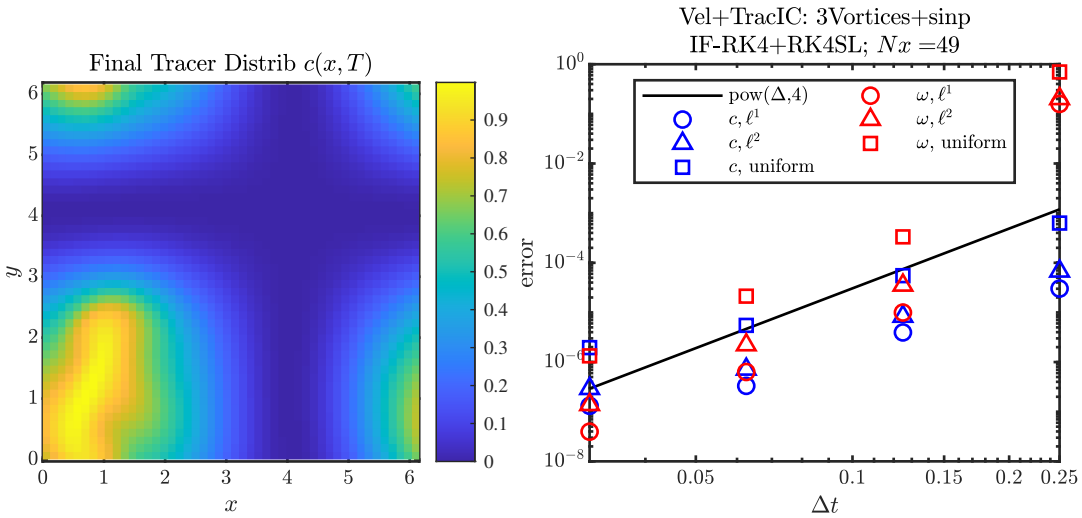
Final Tracer Distrib $c(x,T)$
x
y
Vel+TracIC: 3Vortices+sinp
IF-RK4+RK4SL; Nx =49
pow(Δ,4)
c,ℓ^1
c,ℓ^2
c, uniform
ω,ℓ^1
ω,ℓ^2
ω, uniform
error
Δt

Figure 23:

References

[Aki70] Hiroshi Akima. A New Method of Interpolation and Smooth Curve Fitting Based on Local Procedures. *Journal of the ACM*, 17(4):589–602, October 1970.

[Aki74] Hiroshi Akima. A method of bivariate interpolation and smooth surface fitting based on local procedures. *Communications of the ACM*, 17(1):18–20, January 1974.

[Bar21] Alex H. Barnett. Aliasing error of the $\exp(\beta\sqrt{1-z^2})$ kernel in the nonuniform fast Fourier transform. *Applied and Computational Harmonic Analysis*, 51:1–16, March 2021.

[BMaK19] Alexander H. Barnett, Jeremy Magland, and Ludvig af Klinteberg. A Parallel Nonuniform Fast Fourier Transform Library Based on an "Exponential of Semicircle" Kernel. *SIAM Journal on Scientific Computing*, 41(5):C479–C504, January 2019.

[FC80] F. N. Fritsch and R. E. Carlson. Monotone Piecewise Cubic Interpolation. *SIAM Journal on Numerical Analysis*, 17(2):238–246, April 1980.

[Key81] R. Keys. Cubic convolution interpolation for digital image processing. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 29(6):1153–1160, December 1981. Conference Name: IEEE Transactions on Acoustics, Speech, and Signal Processing.

[McD87] A. McDonald. Accuracy of Multiply-Upstream Semi-Lagrangian Advective Schemes II. *Monthly Weather Review*, 115(7):1446–1450, July 1987. Publisher: American Meteorological Society Section: Monthly Weather Review.

[McD99] A. McDonald. An Examination of Alternative Extrapolations to Find the Departure Point Position in a "Two-Time-Level" Semi-Lagrangian Integration. *Monthly Weather Review*, 127(9):1985–1993, September 1999. Publisher: American Meteorological Society Section: Monthly Weather Review.

[SC91] Andrew Staniforth and Jean Côté. Semi-Lagrangian Integration Schemes for Atmospheric Models—A Review. *Monthly Weather Review*, 119(9):2206–2223, September 1991. Number: 9 Publisher: American Meteorological Society Section: Monthly Weather Review.

[SK06] Marc Spiegelman and Richard F. Katz. A semi-Lagrangian Crank-Nicolson algorithm for the numerical solution of advection-diffusion problems. *Geochemistry, Geophysics, Geosystems*, 7(4), 2006.

[TS87] Clive Temperton and Andrew Staniforth. An Efficient Two-Time-Level Semi-Lagrangian Semi-Implicit Integration Scheme. *Quarterly Journal of the Royal Meteorological Society*, 113(477):1025–1039, 1987.