Parallel High-Order Time Integrators

Bradley Gadd-Speakman

Durham University

1. Introduction

Time integration methods are numerical techniques designed to solve differential equations which model the evolution of a dynamical system over time. They are used to model a wide variety of complex systems including structures, fluids, economical and biological systems. Specifically to this project, the revisionist integral deferred correction (RIDC) method is investigated as a slight alteration of its predecessor, integral deferred correction (IDC). This alteration allows the method to retain its high order of accuracy while also being parallelisable.

The IDC method splits the time domain into equally sized time intervals containing K nodes each and there are said to be N equidistant time steps across the whole domain. IDC, being a predictor-corrector method, relies on using an initial time integration technique **with the order of accuracy P** to 'predict' the approximate values of the solution to an IVP. It then finds all values in each of the **M correction levels** sequentially using an **update formula of order P** derived from the following correction formula:

$$\frac{\partial \boldsymbol{u}^{(m+1)}}{\partial t} = \boldsymbol{f}(\boldsymbol{u}^{(m+1)}, t) - \boldsymbol{f}(\boldsymbol{u}^{(m)}, t) + \frac{1}{h}\int_t^{t+h} \boldsymbol{f}(\boldsymbol{u}^{(m)}, s)ds. \tag{1}$$

The integral in the correction formula must be approximated with a quadrature which uses P(M+1) integration points and **it can be shown that the order of the IDC and RIDC methods have an order of accuracy given by P(M+1)** [2].

An example of an update formula for IDC with P=2 is given by:

$$\boldsymbol{u}_{n+1}^{(m+1)} = \boldsymbol{u}_n^{(m+1)} + \frac{\boldsymbol{K}_1}{2} + \frac{\boldsymbol{K}_2}{2} + \int_{t_n}^{t_{n+1}} \boldsymbol{f}(\boldsymbol{u}^{(m)}, t)dt,$$

where

$$\boldsymbol{K}_1 = h(\boldsymbol{f}(\boldsymbol{u}_n^{(m+1)}, t_n) - \boldsymbol{f}(\boldsymbol{u}_n^{(m)}, t_n))$$

$$\boldsymbol{K}_2 = h(\boldsymbol{f}(\boldsymbol{u}_n^{(m+1)} + \boldsymbol{K}_1 + \int_{t_n}^{t_{n+1}} \boldsymbol{f}(\boldsymbol{u}^{(m)}, t)dt, t_n) - \boldsymbol{f}(\boldsymbol{u}_{n+1}^{(m)}, t_{n+1}))$$

The sequential nature of IDC and time integration techniques in general means that many of these techniques are impossible to parallelise. However, a small adaptation to the order of calculations performed in IDC allows the RIDC method to be parallelisable.

In order to demonstrate these desired properties of RIDC, the pseudo-spectral method was also investigated as a way of dealing with the spatial aspect of the 2D incompressible Navier-Stokes equation, which is the problem of choice for this poster.

2. Pseudo-Spectral Method

The pseudo-spectral method begins by assuming the solution to a differential equation takes the form of a linear combination of predetermined basis functions. It then attempts to find the solution by choosing the coefficients which cause the approximation to best satisfy the differential equation.

The 2D incompressible Navier-Stokes equation is numerically approximated using the pseudo-spectral method in space which turns the PDE into an ODE in time. Starting from the vorticity equation, the lines below show the derivation of this method, using sinusoidal basis functions, for this problem:

Using the fact that the flow is incompressible and only in two spatial dimensions, the vorticity equation simplifies to:

$$\frac{\partial w}{\partial t} = \nu\nabla^2 w - (\boldsymbol{u}\cdot\nabla)w + \nabla\times\boldsymbol{f}.$$

Using incompressibility and the definition of vorticity reveals the following equalities for the components of the Fourier transformed velocity:

$$\tilde{u}_x = \frac{ik_y}{k_x^2+k_y^2}\tilde{w}, \qquad \tilde{u}_y = -\frac{ik_x}{k_x^2+k_y^2}\tilde{w}.$$

After taking the spatial Fourier transform we are left with an ODE without any spatial derivatives:

$$\begin{aligned}\frac{\partial \tilde{w}}{\partial t} &= -\nu(k_x^2+k_y^2)\tilde{w} + ik_x\tilde{f}_y - ik_y\tilde{f}_x \\ &\quad - \mathcal{F}_{\boldsymbol{x}}\left[\mathcal{F}_{\boldsymbol{x}}^{-1}[\tilde{u}_x]\mathcal{F}_{\boldsymbol{x}}^{-1}[ik_x\tilde{w}] + \mathcal{F}_{\boldsymbol{x}}^{-1}[\tilde{u}_y]\mathcal{F}_{\boldsymbol{x}}^{-1}[ik_y\tilde{w}]\right].\end{aligned} \tag{2}$$

It's common to find the spatial part of the solution using a finite number of discrete wavenumbers and this allows the use of the fast Fourier transform in place of the standard Fourier transform algorithm. This algorithm has the advantage of having computational complexity of $O(NlogN)$ rather than $O(N^2)$ for the standard discrete method, where N is the size of the set of frequencies. It can also be shown, but is beyond the scope of this project, that the order for the pseudo-spectral method is proportional to $(1/n)^n$ where n is the largest frequency considered in the Fourier series [3].

3. Revisionist Integral Deferred Correction

Unlike the IDC method mentioned in the first section, RIDC doesn't sequentially complete the prediction and then correction levels for the whole interval before moving on to the next. Instead, it first sequentially computes only as many values as are needed for each correction level such that the final correction has P(M+1)-1 calculated values. Future values for the prediction and correction levels can then be calculated in parallel. Ideally with proper parallel implementation, where each core of a CPU is solely responsible for calculating values in their designated correction level, **this process could achieve a speedup of up to M+1 times with a slight overhead compared to IDC**.

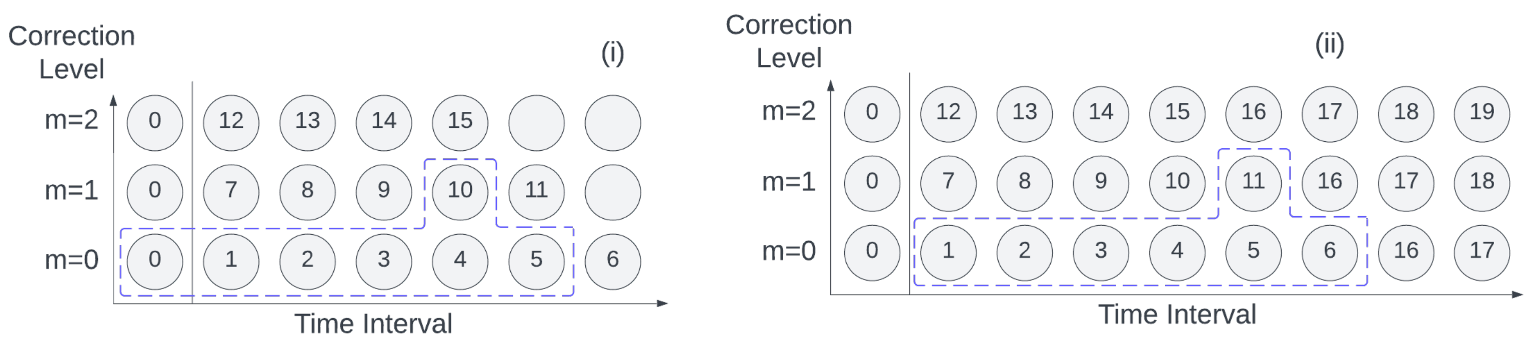

Figure 1: (i) Stencil diagram showing initial setup of RIDC method with $P=2, M=2$. Node number corresponds with order of calculation. The dotted line captures the values required to calculate node 11. (ii) Stencil diagram showing the parallel portion of the same RIDC method. The dotted line captures the values required to calculate node 16 in the correction level $m=1$.

Pros	Cons
- High order	- High order polynomial is used for qaudrature which can introduce error from Runge phenomenon
- Parallelisable	- More function evaluations required due to the many correction levels
- Equidistant nodes means no recalculation of the quadrature weights matrix is required	
- Final correction values are dropped down to prediction level at the end of every time interval	

4. Numerical Solution

To demonstrate the power of the RIDC method, the 2D incompressible Navier-Stokes equation was solved using a combination of RIDC and the spatial pseudo-spectral method. The RIDC method approximately solves the ODE in time given by equation (2) with the following parameters and initial condition:

$$\nu = 0.001,\ \boldsymbol{f} = \boldsymbol{0},\ t\in[0,80],\ \boldsymbol{x}\in[0,2\pi]^2 \qquad \begin{aligned} w(x,y,0) &= e^{-5((x-0.2\pi)^2+(y-1.1\pi)^2)} \\ &\quad - e^{-5((x-0.2\pi)^2+(y-0.9\pi)^2)}.\end{aligned}$$

An RIDC method using a prediction and a correction level both of order two incorporated 12000 time nodes and 12 time intervals with 1000 nodes each, while the pseudo-spectral method used a 100x100 spatial grid. With this setup, the following time evolution of the vorticity was obtained:

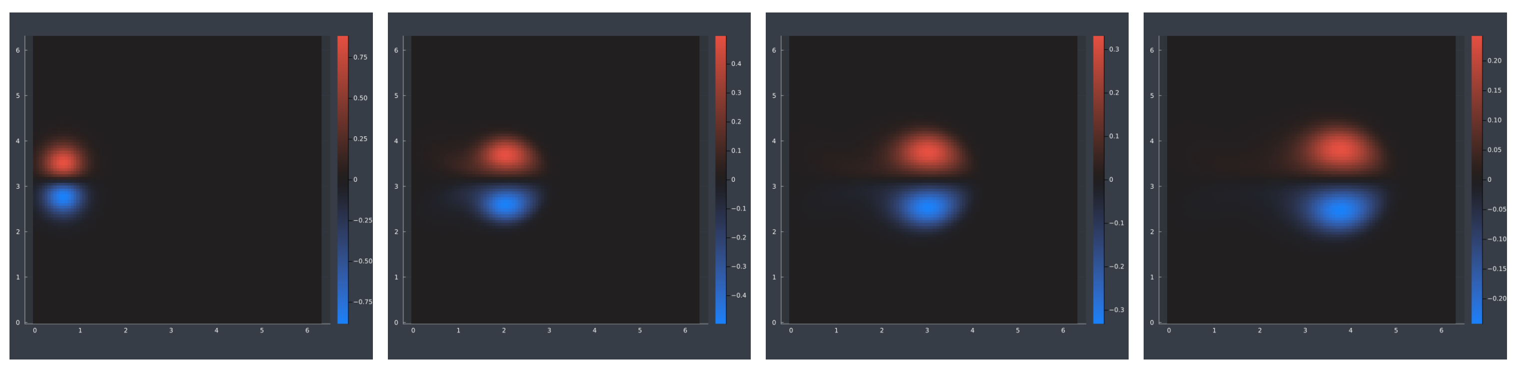

Figure 2: Time evolution of the approximated solution at time intervals; $t = 0, \frac{1}{3}\cdot 80, \frac{2}{3}\cdot 80, 80$

The RIDC method using these parameters is expected to produce a solution with an order of accuracy in time of 2(1+1)=4. My implementation of RIDC for this problem without parallelisation took around 300 seconds of computation time. It is therefore expected that, with optimal parallelisation, the computation time could be halved to 150 seconds.

5. Future Ideas

The following is a short list of my own ideas that I think could be employed to potentially improve certain aspects of the RIDC method.

- Use a neural network which would be trained to approximate the integral in equation (1). My thinking is that this could prevent numerical errors created during the numerical integration caused by the Runge phenomenon. The network could also, with enough training, hopefully approximate the integral better than quadrature and may allow a higher order of accuracy.
- Use adaptive time discretisation to try to obtain a better rate of convergence. This instantly creates the issue that the weights matrix used in the quadrature when performing corrections needs to be recalculated. It could however be worthwhile for applications where accuracy is considered more valuable over computation time.

6. References

[1] John Charles Butcher. *Numerical methods for ordinary differential equations*. John Wiley & Sons, 2016.

[2] Andrew J. Christlieb, Colin B. Macdonald, and Benjamin W. Ong. "Parallel High-Order Integrators". In: *SIAM Journal on Scientific Computing* (2010). DOI: 10.1137/09075740X.

[3] Lloyd N Trefethen. *Spectral methods in MATLAB*. SIAM, 2000.