Assessment of numerical integration schemes and boundary conditions for Lagrangian tracking of marine debris

Joel Kronborg

Master's thesis
2016:E36

Faculty of Engineering
Centre for Mathematical Sciences
Numerical Analysis

CENTRUM SCIENTIARUM MATHEMATICARUM

Master's Theses in Mathematical Sciences 2016:E36
ISSN 1404-6342

LUTFNA-3037-2016

Numerical Analysis
Centre for Mathematical Sciences
Lund University
Box 118, SE-221 00 Lund, Sweden

http://www.maths.lth.se/

Abstract

A growing environmental issue today is the amount plastic pollution in the ocean. To investigate how it might affect marine life, an important aspect is to study how plastics are transported by ocean currents and where it ends up. This can be done by use of Lagrangian simulation of plastic particles in computer ocean models. To improve the quality of the output from such simulations the number of simulated particles is continuously increased, and grid resolutions are refined. This puts increasing pressure on the simulation programs to perform the computations as efficiently as possible.

In the main part of this project the numerical time stepping schemes used in a Lagrangian ocean simulator, and their influence on the efficiency and accuracy of the simulations, were studied. The currently used 4th order Runge–Kutta method (RK4) was compared to the Runge–Kutta–Fehlberg method (RKF45) to assess the potential performance gain by switching to a more advanced scheme. The Explicit Euler method was included in initial tests to illustrate the benefit of using a higher order method. Error testing was performed on an idealized test case based on the Stommel equations, and time testing was performed on model data from the Agulhas region off the coast of South Africa. RKF45 was found to produce less accurate results than RK4 in very similar computation time. However, it used less function evaluations, which means it might still be useful in the future if more advanced interpolation schemes are introduced.

Another numerical aspect was targeted in the secondary part of the project, namely the effect of the boundary conditions between land and sea on the particle trajectories. Simple conditions, such as the currently used no–normal flow partial slip condition, include slowing particles as they approach land to prevent them from accidentally beaching. To allow them to flow freely even along shorelines, a free–slip condition was introduced and evaluated. The two different methods were compared using the Agulhas data. With partial slip, many particles beached along the shorelines, but with free–slip not a single one of the 300 particles used got stuck.

Popular scientific summary

The plastic pollution in the ocean is a growing issue for the marine environment. To determine how and where the pollution affects marine life, an important step is to map the spread of this pollution. This project concerns the numerical aspects of a computer program developed to simulate the distribution of plastics in the ocean.

Each year millions of tons of plastics enter our oceans around the world. With time the larger plastic debris is broken down into smaller pieces by the sunlight and grinding movements of the waves. These smaller pieces can easily get ingested by marine animals, and thus harm them. To evaluate the extent of this threat to marine life, it is important to map the spread of plastics in the ocean. This can be done with the help of computer simulations of the ocean currents.

One method of simulating the flow of objects in the ocean is by tracking individual particles, in this case pieces of plastic. In an ocean model, the ocean currents are given as velocity fields that define the currents at discrete points in space. The velocity of a particle is calculated by interpolating the velocities at its nearest grid points, and then the particle is transported by taking discrete steps of a predefined time length. In practice, millions of particles are tracked simultaneously. This way of using particles to simulate transportation in a fluid is known as Lagrangian simulation.

PARCELS is a computer program that is currently being developed to not only track plastics in the oceans, but also other particles such as oil droplets or tuna. In the main part of this project, different methods for stepping a particle forward in time are compared, to evaluate whether PARCELS would benefit from using a more advanced scheme than it currently does. The current method is known as the 4th order Runge–Kutta method (RK4). To take a step of length dt in time, it uses the velocity at not only the starting point, but also at three intermediate points. This produces a 4th order approximation, i.e. it reproduces polynomials up to the 4th degree exactly.

The method that I implemented in this project is known as the Runge–Kutta–Fehlberg method (RKF45). In addition to the 4th order step, it also calculates a 5th order step, and takes the difference between these two as an approximation of the local error. By putting a tolerance on this error the step size can then be varied to take longer steps where the tracked path is nice, and shorter steps only where necessary.

In the secondary part of the project, the boundary conditions that determine how a particle behaves close to land were considered. The currently used conditions are no–normal flow partial slip. This effectively means that the velocity perpendicular to land goes to 0 as the particle approaches the boundary, and velocities along a beach are slowed, but not quite to 0. To allow particles to flow near a beach without getting slowed, I implemented a free–slip condition. By

mirroring the closest ocean velocities onto land when the particle is close, the particle can be transported without getting slowed even close to boundaries.

To compare the methods used for time stepping, two different test cases were used. The first one used the Stommel equations to simulate a periodic trajectory in a rectangular region. The velocities along the western edge of this region were very large compared to the rest. This test case was used to get a correlation between the number of steps used and the error produced by RK4 and RKF45.

The second test case used real simulated currents from the Agulhas region, off the coast of South Africa. Here the correlation found in the Stommel test was used in an attempt to produce runs with RK4 and RKF45 that gave rise to similar errors, and compare the computation time of these runs.

The boundary conditions were also evaluated using the Agulhas data. 300 particles were released along the coastline, using first the partial slip condition, and then the free–slip condition. The number of particles getting stuck along the shore in both runs were counted.

In the Stommel test, RKF45 was found to produce more accurate results in far fewer steps than RK4. However, the correlation found did not translate well to the Agulhas region. The timed runs took 934.4 s in RK4 and 935.2 s in RKF45, i.e. almost exactly the same. However, the errors produced by the RKF45 run in the Agulhas region were found to be larger than those produced by RK4, meaning RK4 produced more accurate results in the same computation time.

The part of the time stepping that was expected to be most computationally expensive was the function evaluations, i.e. interpolating the grid points to calculate the velocity of the particle. RKF45 was indeed found to use fewer function evaluations than RK4, but was still slower, which implies that other overhead computation was relatively more expensive in RKF45. However, the current method for calculating velocities is by using bilinear interpolation of the grid points. This method is cheap, but might not be very accurate. Thus as the development of PARCELS continues, it might be switched to a more advanced method which would make function evaluations more expensive, and possibly also make RKF45 perform better than RK4.

The free–slip condition proved to be very successful in the testing in the Agulhas region. With partial slip, 95 particles, i.e. more than a third, got stuck along the shoreline. With free–slip, however, not a single particle got stuck. Theoretically, there are still ways for particles to get stuck even with the free–slip conditions, but this is highly unlikely and will probably not cause problems in simulations that use millions of particles.

To conclude, even though RKF45 used less function evaluations than RK4, they ran at very similar computation times. RKF45 produced larger errors in this test, but if function evaluations are made more advanced and computationally expensive in the future, it might still prove useful. The new free–slip boundary conditions prevented particles from getting stuck along the boundaries between

land and ocean, and were thus very successful.

Acknowledgements

For making this thesis possible, I would like to thank:

- my supervisor Dr. Erik van Sebille at Imperial College London, for guidance and for his dedication my project and solving the issue of plastic pollution in the ocean.
- my supervisor Prof. Eskil Hansen at Lund University, for contributing with theoretical knowledge and insightful counseling.
- Dr. Michael Lange at Imperial College London, lead programmer of PARCELS, for enthusiastic assistance with all my programming related issues.
- all the wonderful people at the Grantham Institute, for the important work they do for our planet, and for hosting me during the duration of this project.

Contents

1 Introduction

A growing issue for the world's oceans is the amount of plastic pollution in them. Several attempts have been made to estimate both how much plastics is currently in the ocean, and also how much enters the oceans every year. According to a recent report an estimated 150 million metric tons of plastics are currently in the ocean [1]. Another study by Jambeck et al. [2] estimates that between 4.8 and 12.7 million metric tons of plastics entered the ocean in 2010, which is a significant annual increase. This number is also expected to continue to grow rapidly. Most of the plastics used today and in the past are not biodegradable, which means that it will not disappear from the ocean anytime soon if nothing is done about it.

Most plastics that enter the ocean eventually break down into smaller pieces. This is an effect of the sunlight that it is exposed to, combined with the motions of waves, which eventually grinds it down [1]. The smaller plastic pieces can easily get ingested by marine animals. To study more closely what effects this has on marine life, and where the harm is done, an important aspect is to map the spread of the plastics. This includes questions like where does it come from, how does it spread and where does it end up. This information can then be used to, among other things, compare plastic distribution in the ocean to information about where marine life is most prevalent.

For decades studies have been carried out to answer where our plastics end up [3, 4]. Many observations have been made on the currents of the ocean using a range of different methods, including tracking drifter buoys and using satellite observations. To expand on information obtained from these observations, computer programs have been developed to simulate how plastics spread throughout the sea. One such piece of software called PARCELS is currently being developed at Imperial College London, and the development of this program is the subject of this thesis.

PARCELS will be a framework to numerically simulate the trajectories of not only plastics, but a range of different materia in the ocean. These include icebergs, pumice, jellyfish, tuna, oil droplets etc. The various objects are differentiated by giving them distinct properties that define them and their distribution throughout the ocean. Plastics can e.g. be given the ability to sink or fragment into smaller pieces. Also certain properties of the ocean water can be simulated, such as temperature or salinity distribution. [5]

This thesis targets two numerical aspects of the simulations that are carried out in PARCELS, namely the numerical algorithm used for tracking the particles, and the boundary conditions between land and ocean. The objective of the first and major part is to evaluate the potential performance increase by switching to a more advanced time stepping scheme. In the secondary part, a set of boundary conditions that allow particles to flow freely along a beach without getting stuck is to be implemented and evaluated.

2 Theory

2.1 Lagrangian ocean analysis

A powerful tool for analyzing fluid motion is to consider the movements of fluid parcels, with the accumulation of parcel motion representing the fluid motion [6]. A fluid parcel is defined as an infinitesimal part of the fluid being analyzed [5]. It has constant mass but can be distorted with time and change its volume. When analyzing a fluid this way parcels can be followed and used as the frame of reference, i.e. each individual parcel is tracked in space and time. Describing a fluid as individual parcels like this is know as the *Lagrangian* description. A motivation for this can be to think of the fluid as a collection of individual molecules, and solving the equations of motion for each molecule [7]. It is also a very intuitive way to simulate how objects are advected in the fluid, since the path of each individual object can be tracked. Many observations of flows are also made in a Lagrangian way, e.g. observing winds by tracking balloons or ocean flows by tracking drifter buoys [6].

In practice, we approximate fluid parcels discretely as points within the fluid, and analyze the fluid by solving the equations of motion for a limited number of points [7]. These points are known as particles. A particle can be a discretization of a fluid parcel, but it can also be an object being advected by the fluid, such as a piece of plastic or pumice [5].

To simulate different particles they can be given certain properties to describe each specific type in detail. A plastic particle can be given properties such as density and volume, as well as the ability to e.g. fragment into smaller pieces or get ingested by marine animals. It could also be given certain properties that affect how it interacts with beaches or the bottom of the ocean, e.g. probabilities to beach or get stuck at the bottom. These parameters can also be varied to examine their individual impacts on simulations. The types of particles that can be simulated cover a wide range, including plastics, pumice, jellyfish, oil droplets, icebergs, tuna etc. all with their unique properties [5].

Another way to simulate a fluid is by using an *Eulerian* description. Here, instead of following individual particles, a reference frame fixed in space is used [7]. Essentially the model domain is divided in grid cells, and at each point in time the flow of fluid through each individual grid cell is calculated. A common analogy to explain the difference between the Lagrangian and Eulerian frames of reference is by considering the flow of a river. The Lagrangian way to study it is by sitting in a canoe in the river and tracking your path. The Eulerian way is standing on the river bank observing the river flowing past you.

In PARCELS, and many other ocean simulators, a combination of these frames of reference is used [5]. Grids are used to give the full description of the domain. Flow velocities are given as fields with values given at the discrete grid points, but also properties like temperature, pressure or density can be given as fields,

and used to compute the path of the particles. The individual fields are combined to form the grid. Particles are released in the domain at specific times and positions. They are then advected within the domain. Each particle's velocity at a given time point is calculated by interpolating the velocities at the nearest grid points. Thus, the velocities are originally given at fixed points (Eulerian), but the velocities used are the ones interpolated at the positions of the moving particles (Lagrangian).

2.2 PARCELS

There is already a number of programs available for Lagrangian ocean analysis and particle simulation [5]. Several of them have been around for quite a few years. To improve the quality of the simulations, grid sizes are pushed to finer resolutions and the number of particles used is increased. Thus, as the amount of data being analyzed keeps increasing at rapid speeds, this puts increasing pressure on the efficiency and scalability of these programs. There is a growing need for Lagrangian ocean simulators that are able to handle this evolution of the field. The aim of PARCELS is to be able to do exactly this. The framework and interface of the program is written in Python, while the actual computation kernels are generated and compiled in C code during runtime using Just–In–Time (JIT) computing. It is completely open–source and can be found at https://github.com/OceanPARCELS/.

Another aspect that PARCELS targets is that many current programs were developed with specific goals in mind. They might use a specific model with specific solvers to solve a specific particle tracking problem. In contrast to this, PARCELS is being implemented as a much more generalized program with as few restrictions as possible. It is aimed toward oceanographers who want to simulate their own systems without having to write their own code from scratch. They should be able to specify their own particle types, grids, boundary conditions etc. and use PARCELS to perform the simulations efficiently.

The data is supplied by the user in NetCDF format directly as velocity grids, or other grids that describe parameters like sea surface heights or water density, depending on the problem the user wants to solve. This way PARCELS can interpolate the grids to get the velocities or values needed for each particle and use this to track them by numerical time stepping.

The data is expected to be mainly from two distinctive sources (or a combination of the two), namely mathematical Ocean General Circulation Models (OGCM), or observational data. An example of a widely used ocean modelling software is NEMO (Nucleus for European Modelling of the Ocean) [8]. NEMO comprises several different engines and includes both pre– and post–processing tools. It can be used for a number of different modelling applications, such as simulating salinity or temperature gradients in the ocean. The main model is based on the Navier–Stokes equations [8], which are commonly used for fluid

dynamics applications. It also includes an extra nonlinear equation to couple the temperature and salinity of the ocean to fluid velocities, along with several approximations and assumptions to simplify its mathematical description of the ocean.

GlobCurrent [9], on the other hand, is an example of a system that supplies oceanographic data based on satellite observations. This is complemented by a number of other observational techniques using e.g. drifter buoys, ships or gliders, all with their own specific strengths and limitations.

The data from NEMO, GlobCurrent or other sources that is to be used by PARCELS is given as grids. Thus the equations that PARCELS solve are not directly the complex equation systems that NEMO or other models are based upon. Instead, consider an initial value problem given by

$$\dot{\mathbf{y}}(t) = f(t, \mathbf{y}(t)), \quad \mathbf{y}(t_0) = \mathbf{y}_0 \tag{1}$$

where the particle position at a given time t_n is given by $\mathbf{y}(t_n) = (x_n, y_n, z_n)$. In the case of velocity fields, the function f is the velocity $\mathbf{v}$ of the particle, given as a vector where each element is the velocity in one dimension. The particle is tracked by integrating the expression in time [5]:

$$\mathbf{y}(\mathbf{y}_0, t + \Delta t) = \mathbf{y}(\mathbf{y}_0, t) + \int_t^{t+\Delta t} \mathbf{v}(\mathbf{y}, \tau) d\tau. \tag{2}$$

The solution to this equation is approximated in PARCELS by numerical time stepping. In addition to the equation given here there may also be other terms that should be included, depending on the problem. Say, for example, that the user wants to study the transportation of passive particles from a certain starting position, and study how they get distributed. Then an element of randomness in the simulation might be useful, to make particles take slightly different paths that may deviate over time. This randomness could be introduced by adding a stochastic term to equation (2) that perturbs the particle position slightly in each step.

Other particles, such as tuna, are not only advected by currents, but also swim consciously in certain directions. As particles, they may be given a desire to stay close to other fish to form schools, or stay close to favorable habitats. To include this, expressions that simulate such behavior have to be added to the equation.

2.3 Time stepping

PARCELS computes particle trajectories by using numerical time stepping. Given the initial value problem in equation (1), an approximation of f at a

given time t_n with corresponding position $\mathbf{y}(t_n) = (x_n, y_n, z_n)$ can be used to approximate the position of the particle at time $t_{n+1} = t_n + h$, where h is the time step size [10]. To take such a time step there are a range of different methods available. In this project, three different methods are used and compared. The first, and perhaps most intuitive one, is the *explicit Euler method* (EE). In this method the finite difference approximation of the derivative,

$$f(t, \mathbf{y}_n) = \dot{\mathbf{y}}_n \approx \frac{\mathbf{y}_{n+1} - \mathbf{y}_n}{h}, \tag{3}$$

is used. From this we get the approximate solution of $\mathbf{y}$ at time $t + h$ given by

$$\mathbf{y}_{n+1} = \mathbf{y}_n + hf(t_n, \mathbf{y}_n). \tag{4}$$

It can be shown that this method is accurate to the first order, i.e. it reconstructs a polynomial of the first degree exactly, but when applied to a polynomial of higher degree an error will be introduced [10]. For most applications, first order accuracy is not very satisfactory, as they tend produce relatively large local errors. In an attempt to improve accuracy, one could also use an approximation of the derivative at an intermediate point, say at $t_{n+1/2}$. This would give

$$\mathbf{y}_{n+1} = \mathbf{y}_n + hf(t_{n+1/2}, \mathbf{y}_{n+1/2}). \tag{5}$$

To approximate $\mathbf{y}_{n+1/2}$ in the expression above we use an explicit Euler step and get

$$\begin{aligned} \mathbf{k}_1 &= f(t_n, \mathbf{y}_n) \\ \mathbf{k}_2 &= f(t_n + \frac{h}{2}, \mathbf{y}_n + \frac{h}{2}\mathbf{k}_1) \\ \mathbf{y}_{n+1} &= \mathbf{y}_n + h\mathbf{k}_2. \end{aligned} \tag{6}$$

This can be further generalized by using more intermediate steps of different lengths to give the s–*stage explicit Runge–Kutta* (ERK) method [10]:

$$\begin{aligned} \mathbf{k}_1 &= f(t_n, \mathbf{y}_n) \\ \mathbf{k}_2 &= f(t_n + c_2 h, \mathbf{y}_n + h a_{21}\mathbf{k}_1) \\ \mathbf{k}_3 &= f(t_n + c_3 h, \mathbf{y}_n + h(a_{31}\mathbf{k}_1 + a_{32}\mathbf{k}_2)) \\ &\vdots \\ \mathbf{k}_s &= f(t_n + c_s h, \mathbf{y}_n + h(a_{s1}\mathbf{k}_1 + \cdots + a_{s,s-1}\mathbf{k}_{s-1})) \\ \mathbf{y}_{n+1} &= \mathbf{y}_n + h(b_1\mathbf{k}_1 + \cdots + b_s\mathbf{k}_s). \end{aligned} \tag{7}$$

An ERK method is often represented by its *Butcher tableau* [10], in the s–stage case:

$$
\begin{array}{c|ccccc}
0 & & & & & \\
c_2 & a_{21} & & & & \\
c_3 & a_{31} & a_{32} & & & \\
\vdots & \vdots & \vdots & \ddots & & \\
c_s & a_{s1} & a_{s2} & \cdots & a_{s,s-1} & \\
\hline
 & b_1 & b_2 & \cdots & b_{s-1} & b_s
\end{array}
$$

In the Butcher tableau, a_{ij} scales $h\mathbf{k}_j$ in the expression to compute $\mathbf{k}_i$ in equation (7), $c_i h$ is the size of the intermediate step i in time, and b_i scales $\mathbf{k}_i$ in the expression to calculate $\mathbf{y}_{n+1}$.

The explicit 4th order Runge–Kutta method (RK4), also known as "the classical Runge–Kutta method" or simply "the Runge–Kutta method", is widely used in particle tracking OGCM's today, and is already implemented in PARCELS as well. It is given by the Butcher tableau

$$
\begin{array}{c|cccc}
0 & & & & \\
\frac{1}{2} & \frac{1}{2} & & & \\
\frac{1}{2} & 0 & \frac{1}{2} & & \\
1 & 0 & 0 & 1 & \\
\hline
 & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6}
\end{array}
$$

In the RK4 method the time step size h is fixed. However, in many applications the velocities may vary a lot in magnitude, or the velocity field may curve sharply in some areas while being smoother in other. To produce accurate results, i.e. with small global errors, h must then be small enough to approximate the sharp changes accurately. This comes at the cost of longer computation time, as the steps must be the same size where the solution is nice as well. This motivates the use of methods with variable step sizes, such as the Runge–Kutta–Fehlberg method (RKF45) [10]. This method is based on a 4th order Runge–Kutta method, although it uses 5 stages for it. In addition to the 4th order solution however, it also calculates a 5th order 6–stage Runge–Kutta step. Theoretically this comes at the cost of just one more function evaluation, as $\mathbf{k}_1 - \mathbf{k}_5$ for this step are the same as the ones used in the 4th order solution. The reason why the method uses 5– and 6–stage solutions is that even though it is possible to make a 4–stage 4th order solution, a 5th order solution needs at least 6 stages. The proof for this is quite complicated [11], and is thus excluded here.

The difference κ between the RK5 and the RK4 step,

$$
\kappa = \|\mathbf{y}_{n+1}^{5th} - \mathbf{y}_{n+1}^{4th}\|, \tag{8}
$$

is then taken as an approximation of the local error at the point $n + 1$. The local error is defined as the error introduced by the last step taken to reach the current position. The global error on the other hand is defined as the difference between the current position and the exact solution at the same point in time.

Depending on the magnitude of the estimated local error κ the step size can then be adjusted as needed. If it is below a certain error tolerance T the step is accepted, and if $\kappa < 10T$, say, the step size h is doubled before the next step is taken. The factor 10 can be changed to suit different applications, but here 10 is used simply because it is the value given in Hairer et al. [10]. If $\kappa > T$, on the other hand, the step is rejected, h is halved, and a new step with the new step size is calculated. This is repeated until $\kappa < T$ and the step is accepted. The reason why the tolerance for doubling the step size is so much larger than for halving a step is that the risk of having to reduce the step size soon after doubling it should be low. When the step size is reduced the attempted step is rejected and has to be recalculated, which takes time and shouldn't be done unnecessarily often.

Schemes that use this way of calculating two different solutions to estimate the error are called *embedded Runge–Kutta methods*. The Butcher tableau for the RKF45 method, with the two bottom rows representing the 5th (top) and 4th (bottom) order solutions respectively, is

0						
$\frac{1}{4}$	$\frac{1}{4}$					
$\frac{3}{8}$	$\frac{3}{32}$	$\frac{9}{32}$				
$\frac{12}{13}$	$\frac{1932}{2197}$	$-\frac{7200}{2197}$	$\frac{7296}{2197}$			
1	$\frac{439}{216}$	-8	$\frac{3680}{513}$	$-\frac{845}{4104}$		
$\frac{1}{2}$	$-\frac{8}{27}$	2	$-\frac{3544}{2565}$	$\frac{1859}{4104}$	$-\frac{11}{40}$	
	$\frac{16}{135}$	0	$\frac{6656}{12825}$	$\frac{28561}{56430}$	$-\frac{9}{50}$	$\frac{2}{55}$
	$\frac{25}{216}$	0	$\frac{1408}{2565}$	$\frac{2197}{4104}$	$-\frac{1}{5}$	0

2.4 Interpolation

To calculate the velocity (or other grid data; henceforth only velocities will be considered, but most methodology translates directly to other field types too) of a particle at a certain point in space the velocities at the closest grid points are interpolated. The grids may also change in time in discrete steps, so the velocities are also interpolated (linearly for now) in time. At this stage of development the grid cells are assumed to be two–dimensional and rectangular. The interpolation currently used is *bilinear interpolation* [12], which is an ex-

tension of linear interpolation to two dimensions. Let the velocities be denoted by $\mathbf{v} = (u, v)$, where u and v are velocities in the zonal (x) and meridional (y) directions respectively. To calculate an approximation of $v(x, y)$, say, where $x \in (x_i, x_{i+1})$, $y \in (y_j, y_{j+1})$ (see figure 1) we first perform linear interpolation in the y–direction:

$$\begin{aligned} v(x, y_j) &\approx \frac{v(x_i, y_j)(x_{i+1} - x) + v(x_{i+1}, y_j)(x - x_i)}{x_{i+1} - x_i} \\ v(x, y_{j+1}) &\approx \frac{v(x_i, y_{j+1})(x_{i+1} - x) + v(x_{i+1}, y_{j+1})(x - x_i)}{x_{i+1} - x_i} \end{aligned} \tag{9}$$

Then these acquired values are interpolated in the y–direction:

$$v(x, y) \approx \frac{v(x, y_j)(y_{j+1} - y) + v(x, y_{j+1})(y - y_j)}{y_{j+1} - y_j} \tag{10}$$

In a three–dimensional grid this is easily generalized by interpolating linearly in the z–direction. The interpolation is performed separately in each dimension, i.e. for the u and v velocities.

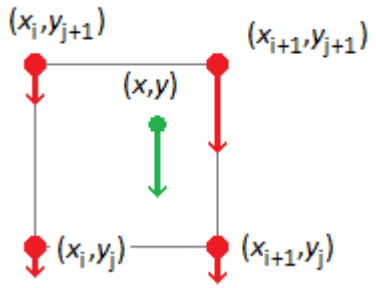

Figure 1: Bilinear interpolation performed on meridional velocities.

There are several different options available for the choice of grid type. In oceanography, it is common to use a staggered grid, i.e. a grid where the different data fields are not all defined on the same positions. Advantages of this include simplified capturing of smaller scale eddies, i.e. vortices, but details on this will not be further described here. For more information, please refer to Griffies [13].

Figure 2 shows the five Arakawa grids [14] that are commonly used for oceanographic and meteorological applications. Although there are many different arguments for choosing one grid type over another, the most commonly used

ones in OGCM's are either the staggered C– or B–grids [13]. In open water all grid types can be interpolated in similar fashion, since each field is interpolated separately. To implement the boundary conditions, however, the grids may have to be treated differently.

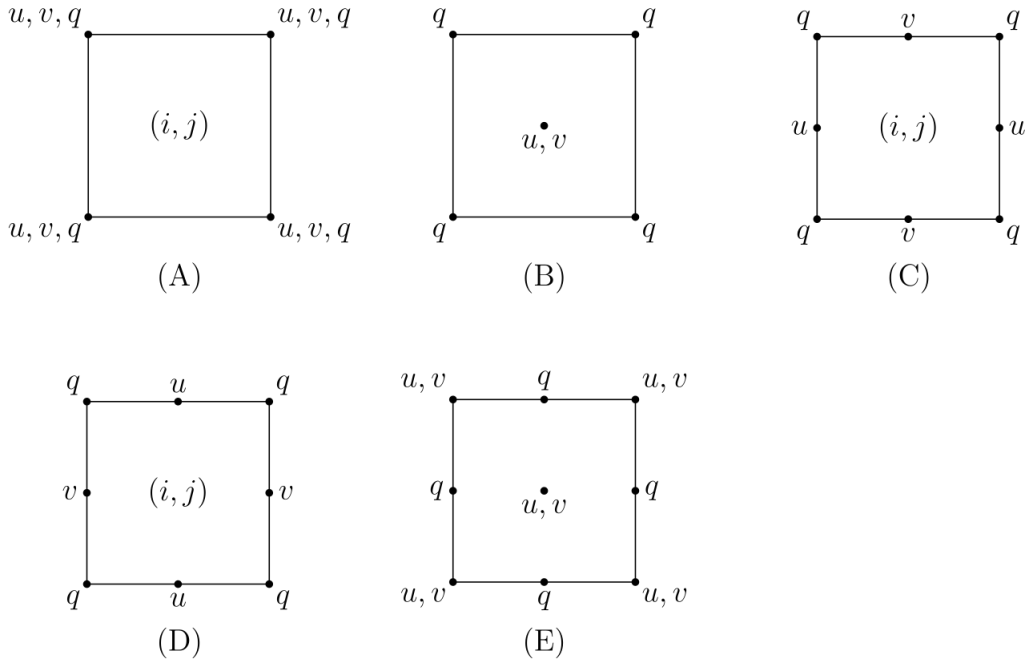

Figure 2: The five Arakawa grids in two dimensions [14]. u and v denote zonal and meridional velocities respectively, and q denotes other grid parameters such as e.g. temperature or surface height.

2.5 Boundary conditions

When carrying out simulations of particles in the ocean there may be a number of different boundaries that have to be considered. In a 2–dimensional case there may be boundaries between land and sea, and also at the edge of the simulation domain. When including a third dimension (depth) there are also boundaries between sea and bottom, and sea and surface. There may be a great number of different conditions applicable to these boundaries, depending on the experiments the user wants to conduct. For example different particles should probably behave differently on the edge towards land. A plastic particle may be allowed to be washed ashore and beach, i.e. get stuck, but this should never happen when the particle considered is a fish. In this thesis we only consider boundaries between land and ocean, and restrict ourselves to the 2–dimensional case.

While the user should be able to define and implement their own boundary conditions based on their own application, this part of the thesis is mainly concerned with implementing a specific condition between sea and land. This is to have an example to use for other testing and development of PARCELS, and hopefully to use for some real application. It is for now restricted to two dimensions, but the same conditions could be applicable to the boundary between sea and bottom. The grid is assumed to be an Arakawa C–grid [14] with velocities for each grid cell defined on the east and north edge of each cell as per NEMO convention [8]. In NEMO the grid is also masked by setting the v (u) velocities on edge cells with land to the north (east) to 0, which is the value that is given to all land grid points. The purpose of this is to always have north/south (east/west) edges defined by v (u) points. This is also assumed to have been done in this application. In PARCELS, however, the border and land grid points are instead given the value NaN.

The objective of the conditions is to make particles flow freely along edges without risking beaching, i.e. getting stuck. A simple way to prevent particles from being advected onto land grid cells is to simply let all land points have the value 0, as NEMO does [8]. On an Arakawa C–grid, this effectively results in *no–normal flow* combined with a *partial slip* condition [15], see figure 3 (a). The no–normal flow condition means, as the name implies, that a particle's velocity perpendicular to the boundary goes to 0 as the particle approaches the edge. This holds both for velocities toward the boundary and away from it, meaning that it does not only slow particles approaching the beach, but also makes it harder for them to drift back to open water.

The partial slip condition slows the particle velocities parallel to the boundary when they are close to it, but the velocity doesn't go to 0. (If the velocity goes to 0 it is known as a *no–slip* condition.) In many fluid dynamics applications no–slip and partial slip conditions are widely used, as it can be shown that the velocity of a fluid reduces to 0 near boundaries at very small scales [16]. However, these scales are far too small to be accurately represented in today's OGCM's [15]. This might motivate the use of another type of condition, such as the *free–slip* condition [15]. The free–slip condition is defined as letting the gradient of the velocity perpendicular to the boundary go to 0 when the particle is close. This condition is also used in combination with a no–normal flow condition, but does not slow velocities along a boundary, effectively reducing the risk for unwanted deceleration of particles.

In practice there are a few different ways to implement a free–slip condition. One of these is by making use of *ghost points* [15]. A ghost point is essentially a land grid point which is temporarily given the same value as a neighboring sea grid point. Consider the v velocities of a particle at (x, y) flowing along a western boundary (figure 3 (b), 1 for notation). The values on the land points in PARCELS are originally $v(x_i, y_j) = v(x_i, y_{j+1}) = \text{NaN}$, but for the interpolation to calculate $v(x, y)$ the sea velocities $v(x_{i+1}, y_j)$ and $v(x_{i+1}, y_{j+1})$ are mirrored onto the corresponding land points, i.e. $v(x_i, y_j)$ and $v(x_i, y_{j+1})$ respectively.

In figure 3 these ghost point velocities are light red. This effectively collapses the interpolation equations 9 and 10 to

$$v(x,y) = \frac{v(x_{i+1},y_j)(y_{j+1}-y) + v(x_{i+1},y_{j+1})(y-y_j)}{y_{j+1}-y_j}. \tag{11}$$

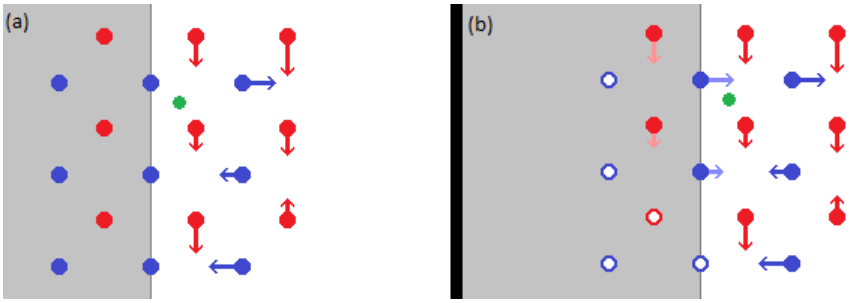

Figure 3: Illustration of the original partial slip condition (a) and the newly implemented free–slip condition (b) on an Arakawa C–grid. In (a), land grid points simply have the value 0, resulting in slower velocities for nearby particles. In (b), land grid point values are NaN, until a particle (green) is adjacent to them. At this stage the nearest ocean velocities are mirrored onto the land points to force the particle away from land and also allow it to have an unslowed velocity parallel the beach.

The no–normal flow condition considered thus far does not only slows particles' velocities toward boundaries, but also away from them. A possible remedy for this could be to only have the normal velocities on the boundaries be 0 when the closest sea velocities are directed toward the boundary, and otherwise mirror the sea velocities. To force particles away from the coastlines even more strongly, and thus reduce the risk of unwanted beaching, we instead use ghost pointing for perpendicular velocities too by mirroring the closest velocities and adjusting the sign to make the ghost point velocity point away from the boundary. These are also slightly scaled to make velocities away from a boundary larger than toward them.

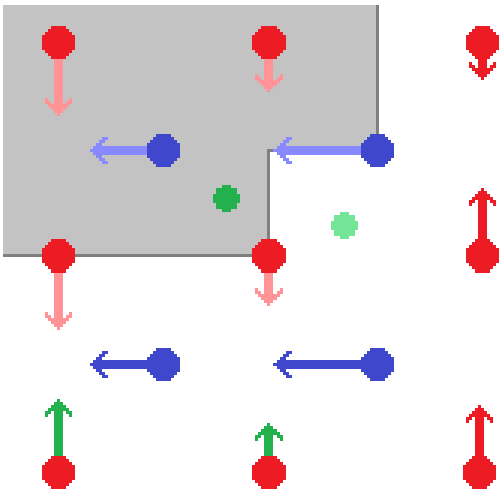

Figure 4: Sketch showing how a particle with the free–slip boundary conditions might end up on "land", i.e. all its closest grid points in either the u or the v velocity field are NaN. The light green particle is advected west since both of the u grid points north of it can be used for ghost pointing velocities along the boundary, and thus no beaching threat is detected. If the particle does get advected onto land, i.e. where the dark green particle is, the velocities marked with green are ghost pointed onto all four v points surrounding it, pointing southward and scaled to force the particle back to sea.

With these boundary conditions in place there is still a way for particles to reach areas where all the nearest grid points of one field are NaN. This is illustrated in figure 4. Consider a corner with land to the north–west and velocities in the surrounding sea points pointing west (u) and slightly north (v). As the light green particle in figure 4 moves westward it will not detect a western boundary hindering it since both its southern u field points will always be sea points, which may be mirrored onto the land points to the north of them. Thus if the light green particle keeps moving westward it may end up at the position marked with dark green in figure 4, where all its four closest v grid points are land points. It is a (fairly) safe assumption[1] that the particle will still be closer to its southern v grid points in this case. Thus if a particle is on land in the v field and closer to its southern v points than its northern ones, the grid points

[1]The exception would be if a particle moves out of bounds in both the x and y directions in the same time step, resulting in it being surrounded by only NaN points in both u and v, which is quite unlikely. In case this actually does occur the particle simply gets stuck, at least at this stage of development, which is expected to be a minimal loss considering the great number of particles used in most simulations.

another step to the south of it (green velocities in figure 4) are mirrored onto the points surrounding the particle, all pointing south to force the particle back to sea. These mirrored velocities are scaled to speed up the returning of the particle to sea.

2.6 Summary of aims and restrictions

The main part of the thesis concerns the efficiency of the numerical time stepping scheme used in the computations. Rather than determining the ultimate method, it should be viewed as an evaluation of the improvements that can be made by switching to another scheme than the current RK4. If the improvements when using RKF45 prove to be significant, the benefit of putting more effort into finding a better method might be worthwhile, but if not it might be better to prioritize efforts in other aspects of the program. EE will also be included in initial testing, hopefully displaying benefits of using a higher order method. The project is restricted to using these three methods only.

It is difficult to say beforehand what improvements can be made by switching to another numerical scheme. At the very least RK4 is expected to perform better than a EE because of its higher order. An embedded Runge–Kutta method should be able to improve performance even further by taking larger steps where the solution is nice, even though each individual step it calculated at a greater computational cost.

In the secondary part of the project the boundary conditions between land and ocean are considered. While PARCELS should be a very general program which allows the user to specify their own boundary conditions, the objective of this part was to implement a specific type of condition, to be used for early applications and further development. With these conditions, particles should be able to travel close to beaches without getting slowed in any direction, and without getting stuck.

To simplify the formulation of these conditions some additional constraints are assumed. The grid type may play a significant part in how the boundary conditions are implemented, so for the time being, the commonly used Arakawa C–grid [14] is assumed. This grid is assumed to be 2–dimensional. As the boundary conditions are implemented by considering a number of different cases that may occur, i.e. the particle may be in a grid cell with 0–4 land points placed in different ways, the interpolation algorithm used is restricted to bilinear interpolation [12].

3 Methodology

In the main part of the thesis, the ultimate goal of the testing was to run a time test on a case where RK4 and RKF45 produced similar results in terms of accuracy. To do this two different test cases were used. The first one is based on the Stommel equation, which describes an idealized periodic current along a western boundary. This was used to get a correlation between the number of steps taken and the error of the solution. For the actual time testing a data set from NEMO was used. The data set approximates currents around the coast of South Africa. This data set was also used for the evaluation of the boundary conditions.

3.1 The Stommel test case

To compare the numerical methods they were initially applied to the (periodic) steady–state solution of the Stommel equation, as described by Fabbroni [17]. The equation describes a periodic velocity field in a rectangular region, with large magnitudes along the western boundary. Further background details are given in Pedlosky [18].

The analytical steady–state solution for the stream function ψ is given by

$$\psi = (1 - e^{-x/\epsilon_s} - x)\pi \sin(\pi y) \tag{12}$$

where ϵ_s, the thickness of the western boundary layer, is defined as $\epsilon_s = r/(\beta a)$. Here r is the inverse time scaling caused by bottom friction, β is the latitudinal variation of the Coriolis parameter f, i.e. $\beta = df/dy$, and a is related to the width of the field. The parameter values used in this implementation are the same as the ones used by Fabbroni, except for a scaling of a by 100 to widen the field:

$$\begin{aligned} a &= 2000000, \\ \beta &= 2 \cdot 10^{-11}\ (\mathrm{m\,s})^{-1}, \\ r &= \frac{1}{11.6 \cdot 86400}\ \mathrm{s}^{-1}. \end{aligned} \tag{13}$$

The velocity field is given by

$$\begin{aligned} u &= -\frac{\partial \psi}{\partial y} = -(1 - e^{-x/\epsilon_s} - x)\pi^2 \cos(\pi y), \\ v &= \frac{\partial \psi}{\partial x} = \left(\frac{e^{-x/\epsilon_s}}{\epsilon_s} - 1 \right) \pi \sin(\pi y). \end{aligned} \tag{14}$$

Figure 5 shows a quiver plot of the velocity field. To test the numerical methods only the trajectory of one particle was studied, with starting position $(x, y) = (10, 50)$. The simulation time was 2387334 s, which allowed the particle to make approximately one full lap. To implement the velocity field a 1000×1000 A–grid was used with bilinear interpolation. While a more exact solution could in this case probably be achieved by using the analytical solution to equation (14), instead of interpolating a discretized vector field, a grid is used here to make the test case more similar to real applications.

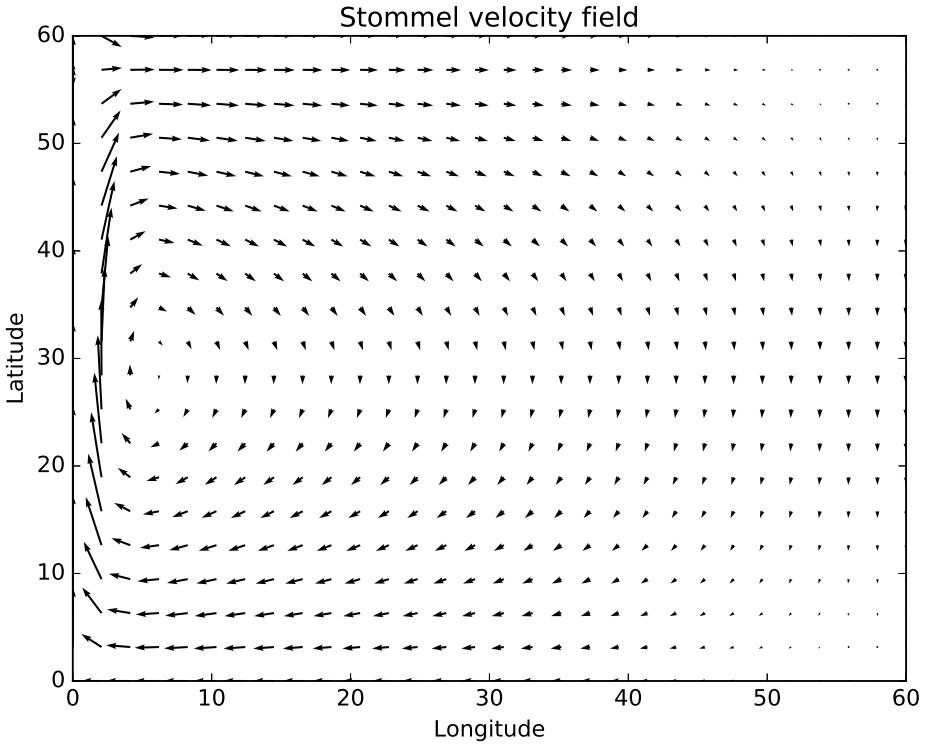

Figure 5: Quiver plot showing the Stommel velocity field. A particle placed in the field will have a periodic trajectory, i.e. it will return to its starting point after making a full lap.

As ground truth an RK4 run with 100000 time steps was used. The error examined was defined as the distance between the end point of the simulation and the end point of the "ground truth", taken with an accuracy of 10^{-6}. The unit of distance and the error here is a little bit abstract, since positions are given in degrees, but the grid cells are uniform squares, i.e. no correction for the deformation of the grid closer to the poles has been used. Thus the unit of

positions and error should rather be considered general length units.

The Stommel test case was used to get a correlation between the number of time steps used and the resulting error for the different numerical schemes. To do this, first a set of runs using RKF45 with tolerances κ ranging from 10^{-10} to $10^{-8.5}$ in 200 steps evenly spaced on a logarithmic scale was performed. To allow RKF45 to set its starting time step size implicitly, the step size was initially given as the full execution time, and was then scaled down in the time stepping algorithm by rejecting steps that did not fulfil the tolerance. The number of steps used by these runs were saved, and two additional sets of runs, one with RK4 and one with EE, with these same numbers of steps were performed. The errors for all runs were examined to get a correlation between the number of steps and the error for RK4 and RKF45. This correlation was then used in the Agulhas test case to perform the time measurements. Since the purpose of including EE was to show the benefits of using a higher order method in terms of accuracy, it was omitted from the time testing.

3.2 The Agulhas current

The test case used to perform the actual time tests was based on NEMO data on currents in the Agulhas region, off the coast of South Africa. The region used here spans approximately from 4°E 44°S to 38°E 24°S. Figure 6 shows a map of the velocity fields for one time point in the region. The region is interesting from an oceanographic perspective because of the different currents that converge and cause turbulence here [19]. The narrow and swift Agulhas Current flows southwestward along the southeast coast of Africa, transporting warm water from the Indian ocean. In the southern part of the region the Antarctic Circumpolar Current, flowing eastward around Antarctica, causes the Agulhas current to retroflect, resulting in an eastward flow back toward the Indian Ocean. In the western part of the region the colder Benguela current transports water north along the southwest coast of Africa. There is also a limited leakage of warm water from the Agulhas current to the western part of the region.

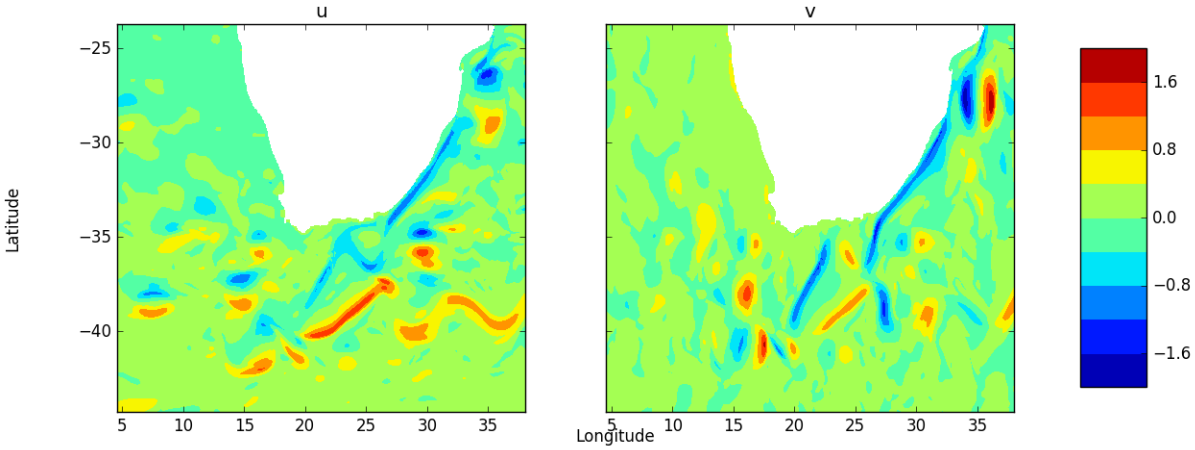

Figure 6: The zonal (u, left) and meridional (v, right) velocity fields at the starting time of the Agulhas data.

The data is given as zonal and meridional velocities on a C–grid with dimensions 402×302, varying in time in discrete steps. The time span is from 2012–01–05 to 2012–02–29, with data given every 5 days, i.e. at 12 time points. To perform the tests of the numerical schemes, 200 particles were released evenly spaced along a straight line from 10°E 40°S to 28°E 40°S, to flow freely for 55 days, i.e. the full time span of the data set. Particles reaching the outer edge of the region within the run time were discarded, since no boundary conditions were defined for this yet.

Before running the time tests an approximate calculation of the errors was performed, to assess whether the correlation found in the Stommel test would hold even for this less idealized test case. The error was once again defined as the deviation of a solution from a more exact "ground truth" run, this time using RK4 with 10000 time steps. Since the Agulhas case is based on real currents, however, it is of a more chaotic nature than the Stommel test. Thus a small deviation early in a particle trajectory might end up making it follow a significantly different trajectory, making the size of the errors a less precise metric. This is the motivation for including the Stommel test to study errors.

To examine the error, an RKF45 run with a tolerance of $1.06 \cdot 10^{-9}$ was performed. The same starting positions as in the time test were used. The number of steps used in each particle trajectory was saved, scaled according to the correlation found in the Stommel test, and then an RK4 run with the corresponding number of steps for each particle was performed. To assess whether there was any significant difference between the errors, a two–sample Kolmogorov–Smirnov (KS) test [20] was performed. The two–sample KS test uses the the empirical distribution functions of two test samples to determine if they are

from the same distributions. The KS statistic $D_{n,m}$ is defined as

$$D_{n,m} = \sup_x |F_{1,n}(x) - F_{2,m}(x)|, \tag{15}$$

where $F_{1,n}(x)$ and $F_{2,m}(x)$ are the empirical distribution functions of the two test samples respectively. The two samples are determined to not be from the same distribution with a certainty level of α if

$$D_{n,m} > c(\alpha)\sqrt{\frac{n+m}{nm}}, \tag{16}$$

where n and m are the sizes of the two samples respectively. For $\alpha = 0.05$, which is the level used here, $c(\alpha) = 1.36$.

To perform the time tests, 50000 particles were released at each starting position, and the simulation was carried out for 55 days. RK4 was used with 1000 time steps.

To test the boundary conditions, the Agulhas test case was used again. The evaluation was done by releasing 300 particles in grid cells adjacent to land all along the coastline, and counting the number of particles that got stuck. A particle was defined as stuck if, by the end of the run time, it was in a grid cell on or just next to land, and it didn't travel more than 10^{-3} degrees in the last 5 time steps (i.e. 0.275 days). The run time used was again 55 days. Particles that reached the edge of the region within the run time were not discarded from this test. Instead their velocities were set to 0 for the remainder of the test run. These were not counted as stuck, since they were not in grid cells adjacent to land. For the tests included in this report only RK4 was used, but the boundary conditions were confirmed to work with RKF45 as well.

4 Results

4.1 Performance of the numerical methods

The global error for each run at the end point of the Stommel test is plotted in figure 7 against the number of steps used in each run. The EE error appears to have a more or less steady logarithmic decline with increasing number of steps, although the error is in the order of 10^1–10^2 for almost all step sizes used here, which is well above any useful level for practical applications. The error also appears to follow a slightly wavy pattern for lower number of steps.

For RK4, the error also appears to follow a logarithmic decline with increased number of steps, but with a wavy pattern along the "mean" decline. This wavy pattern, most clearly visible around 10^2 steps, is however dampened with increased number of steps.

The errors for the RKF45 runs appear to be on almost constant levels for many runs with different numbers of steps, with some jumps between these levels. Increasing the number of steps by a large enough amount leads to a decrease in the error, although there is no clear correlation between the error and the number of steps used.

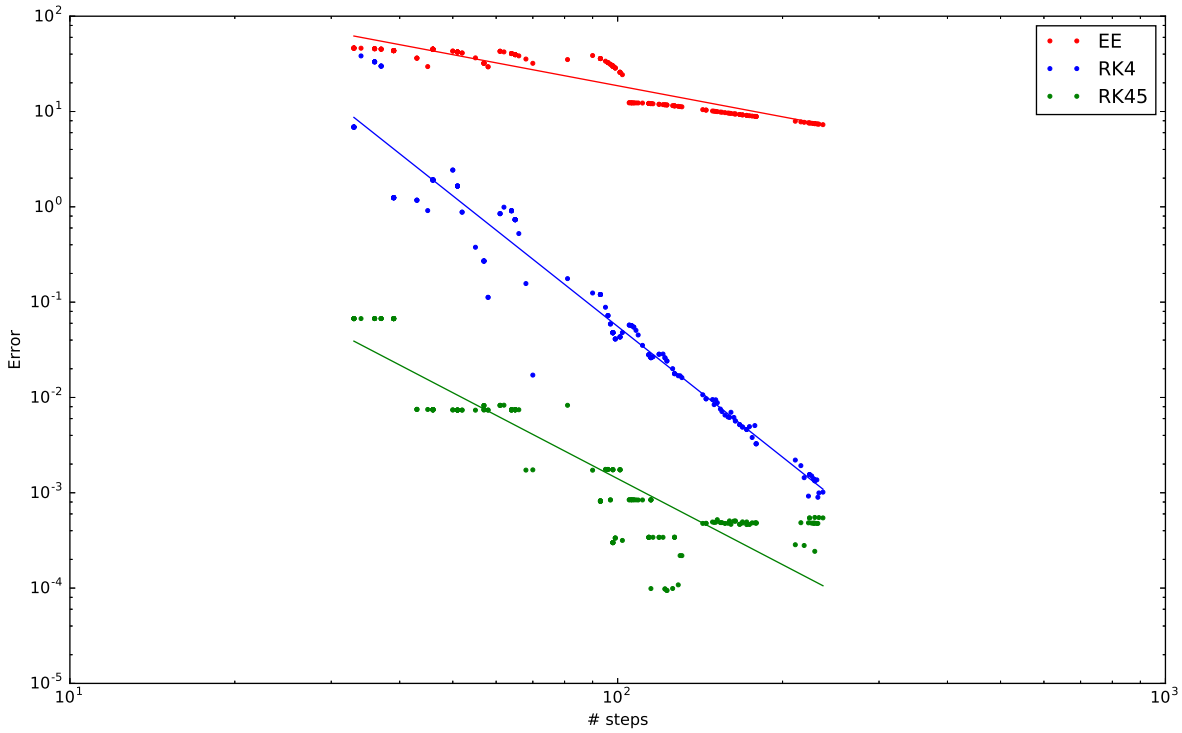

Figure 7: The global error at the end point of RKF45, RK4 and EE, all using bilinear interpolation, for 200 different runs on the Stommel test using different numbers of steps. The numbers of steps used were picked by running RKF45 first with tolerances evenly spaced on a logarithmic scale from $10^{-8.5}$ to 10^{-10}, and then using the number of steps used in these runs for RK4 and EE. A line is fit to each data set using the least squares method, purely for visualization purposes.

To scale the numbers of steps to be used in the Agulhas test case one of the RKF45 runs that produced an error of $7.4 \cdot 10^{-3}$ was used. This choice was rather arbitrary, only motivated by the fact that the wavy pattern of the RK4 error looks to have stabilized at this point. No further investigation of the sensitivity to this value on the end results has been made.

Specifically, the tolerance used was $1.06 \cdot 10^{-9}$, which produced a trajectory with 65 steps in the Stommel test. Several RK4 runs achieved errors very close to this, the closest one using 156 steps. Thus the scaling parameter used to make RK4 and RKF45 achieve similar error levels in the Agulhas test was

$\gamma = 156/65 = 2.4$.

The ground truth trajectories of the particles in the Agulhas test case are displayed in figure 8, with the u field at the starting time (2012–01–05) as the background. Figure 9 displays the empirical distribution functions for the errors of the RK4 and RKF45 methods when applied to the Agulhas case. The Kolmogorov–Smirnov test yielded a statistic of $D_{n,m} = 0.225$. 54 particles were discarded since they reached the outer boundary of the simulation region, leaving 146 particles. This means that the null hypothesis that the two error sets were from the same distribution is rejected on a level $\alpha = 0.05$, since

$$D_{n,m} = 0.225 > c(0.05)\sqrt{\frac{146 + 146}{146 \cdot 146}} = 0.159. \tag{17}$$

In other words, the errors produced by RKF45 and RK4 in the Agulhas test were from different distributions. Figure 9 shows that the RK4 curve lies above the RKF45 one, at least for lower error levels, meaning that the results from RK4 are more accurate than those from RKF45.

The results from the time measurements in the Agulhas test with 50000 particles released at each starting position are displayed in table 1 below.

Table 1: Computation time used by RK4 and RKF45 in the time testing in the Agulhas region.

RK4 (s)	RKF45 (s)
934.4	935.2

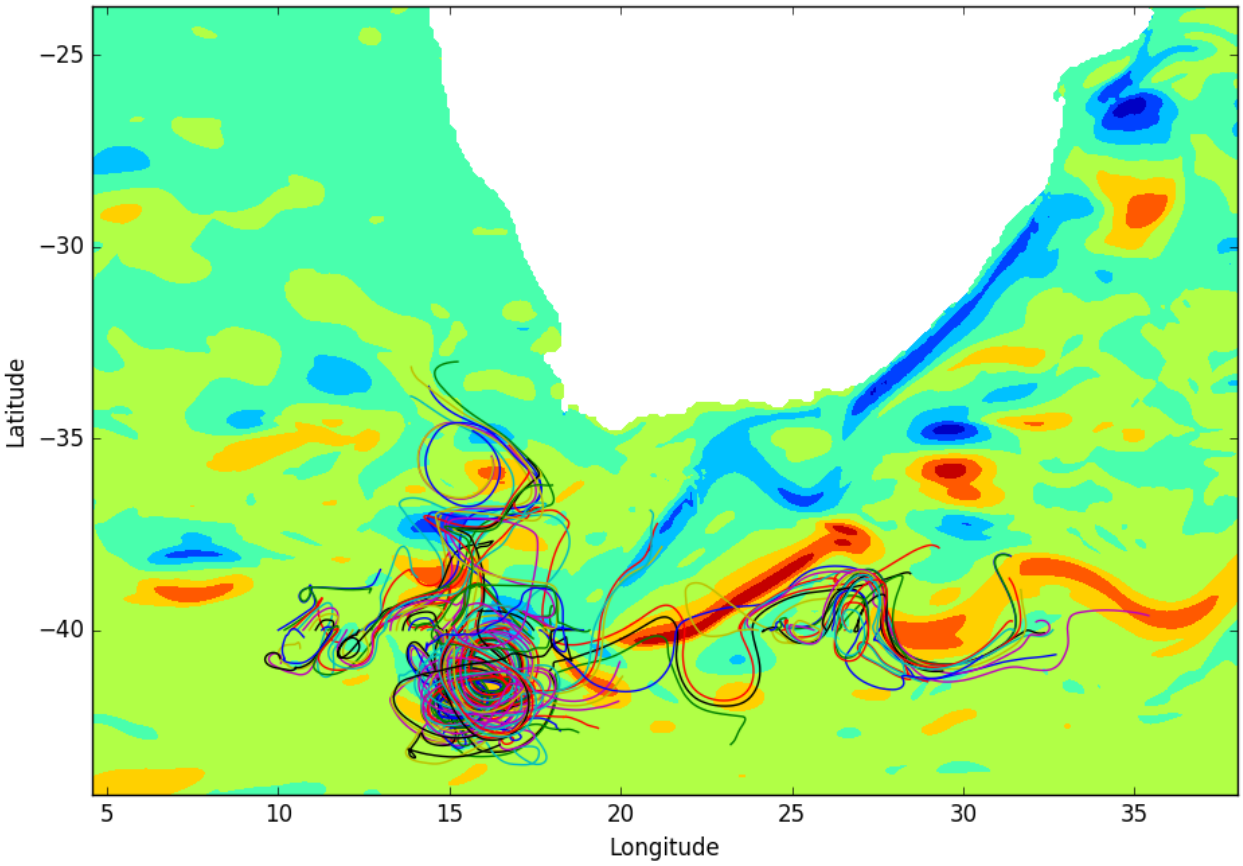

Figure 8: Ground truth trajectories used for testing the numerical integration schemes in the Agulhas region. 200 particles were released evenly spaced along a line from 10°E 40°S to 28°E 40°S. Particles that reached the outer boundary of the region were discarded from tests and are not shown here. The background shows the u velocity field at the starting time. Colors of the trajectories are only to distinguish separate paths, and have no further meaning.

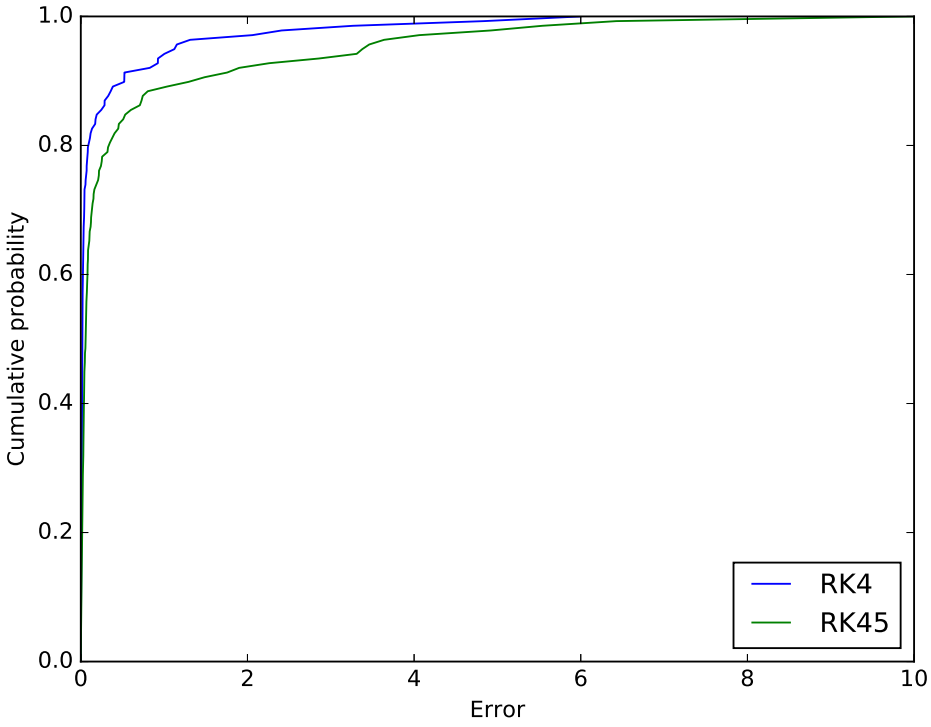

Figure 9: Empirical distribution functions of the errors, defined as the normed distance in degrees from the end points of the ground truth trajectories, for RK4 and RKF45 when applied to the Agulhas test.

4.2 Boundary conditions

Figure 10 shows the trajectories of particles released along the coastline, both with simple no–normal flow partial slip condition and with the new free–slip implementation. There is a visible increase in particles being advected away from the coastlines when free–slip is used. More particles released along the east coast actually get caught in the Agulhas Current and follow it for a considerable distance, even through the retroflection in the south and back toward the Indian ocean. In the north eastern corner of the region, in the Maputo Bay area, very few particles actually leave the coastline when partial slip is used. With free–slip, however, many of them eventually get caught in eddying motions off the coast.

Along the west coast the difference is not quite as visually obvious, but there

appears to be some increase in the number of particles leaving the coastline and being caught in the Benguela Current. There also appears to be a slight increase in the distance covered by some of them, implying that the partial slip condition slows them down at the beginning of their journey.

To quantify the difference, the number of stuck particles and mean distance covered by the particles of the two different runs are displayed in table 2 below.

Table 2: Number of stuck particles and mean distance covered by particles released along the coastline in the Agulhas region when using no–normal flow with partial slip and free–slip respectively.

	Partial slip	Free–slip
# particles stuck	95	0
Mean distance traveled [deg.]	5.11	9.55

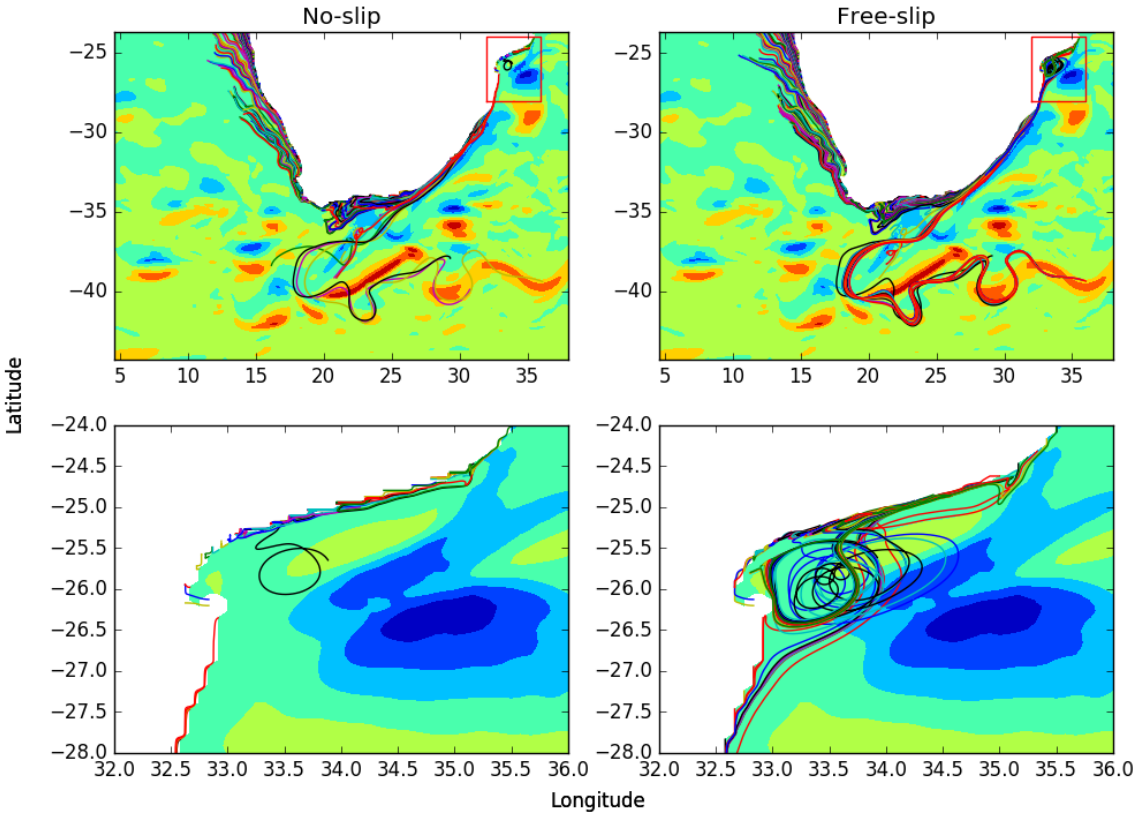

Figure 10: Trajectories for particles released off the coast in the Agulhas region using the partial slip (left) and free–slip (right) boundary conditions. The two bottom plots show detailed views of the section marked with red squares in the top figures. The background shows the u velocity field at the starting time. Some particles appear to be on land in the detailed views. This is because all four grid points of a cell need to have a strictly non–NaN value to color the cell correctly, but with the use of ghost points the particles can still flow through cells with one or more NaN grid points.

5 Discussion

5.1 Performance of the numerical methods

The errors produced in the Stommel test, displayed in figure 7, show many interesting traits. Not surprisingly, the errors produced by EE are significantly larger than the ones produced by the other methods, and its slope appears smaller. In magnitude the errors are quite large, only just reaching below 10 degrees when more than 100 steps are used. A quick look at an EE trajectory using 39 steps, displayed in figure 11, shows that the large errors are because the trajectory goes out of bounds at the western edge. This makes the test results rather unfair to EE, as it would probably be able to produce smaller errors if the test region were larger. However, even before going out of bounds the error is visibly quite large.

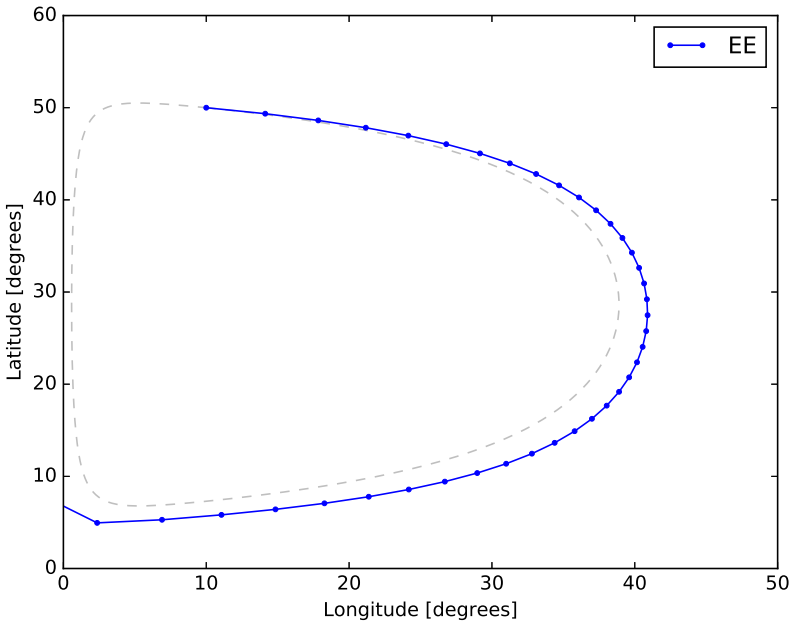

Figure 11: Trajectory using 39 steps produced with the EE method in the Stommel test. The dashed grey line shows the ground truth trajectory.

The EE errors also display a strange, almost wave shaped pattern. The RK4 errors also appear to vary in a similar fashion. These seem to follow a quite clear logarithmic slope with decreasing time step sizes, but especially for larger step

sizes the wavy pattern appears along the slope. Figure 12 shows a more detailed view of this, with more closely spaced data points. To examine the cause of the wavy pattern, the western part of three trajectories with 63, 66 and 69 steps respectively are shown in detail in figure 13. Their produced errors are marked with colored dots in figure 12. In figure 13, we see that all three trajectories seem to track similar curves until they get close to the western edge, where velocities are much larger than in the rest of the region. The large velocities appear quite problematic, as a visible error is introduced in both the green (66 steps) and blue (63 steps) trajectories. The red one, however, happens to take steps that don't deviate much from the ground truth, for no obvious reason. The wavy error pattern in figure 7 seems to depend on where the steps along the western edge are placed. However, as the number of steps increase, so does the amplitude of this pattern and the error in general, which is of bigger importance.

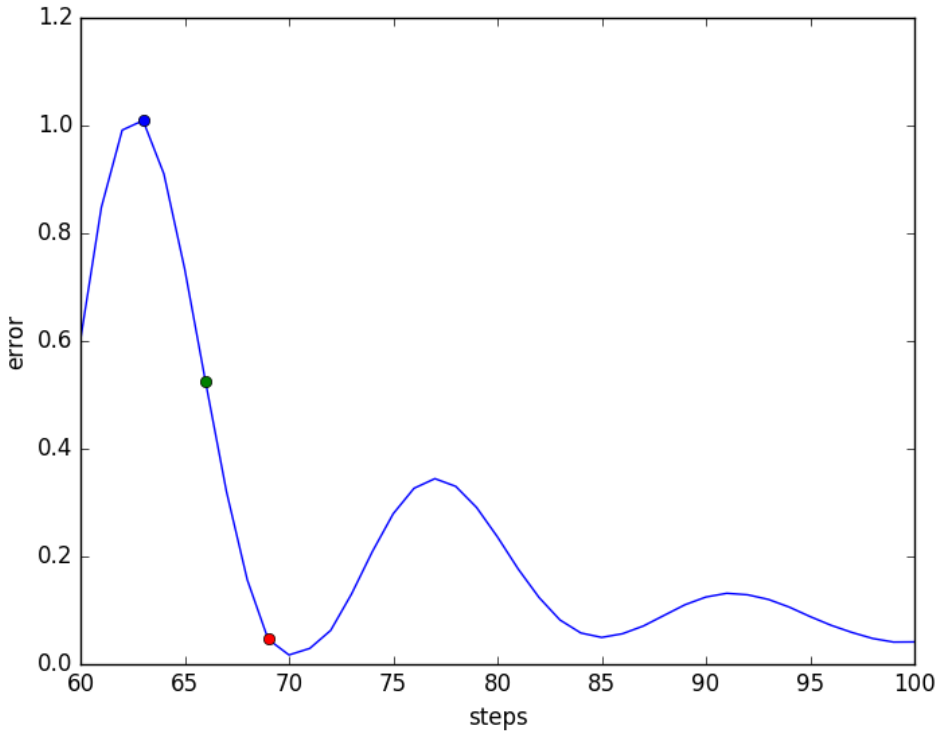

Figure 12: A detailed view of the errors produced by RK4 on the Stommel test with number of steps ranging from 60 to 100. Here both axes are linear, as opposed to the logarithmic axes in figure 7. The colored dots correspond to the colors of the trajectories in figure 13.

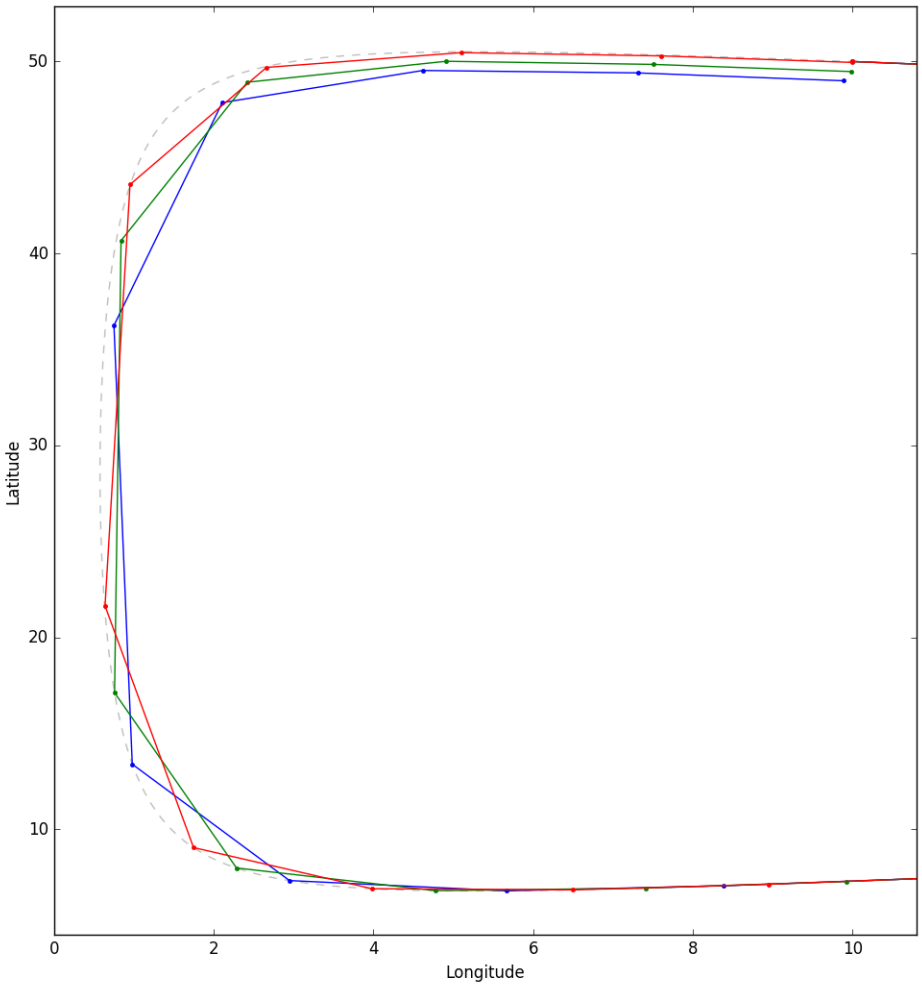

Figure 13: Detailed view of three trajectories produced by RK4 runs using 63 (blue), 66 (green) and 69 (red) steps respectively in the Stommel test. All trajectories trace a similar curve until they reach close to the western edge, whereupon different errors are introduced by the large velocity shift. The ground truth trajectory is shown as a dashed grey line.

Not surprisingly, the errors of the RKF45 method prove to be significantly lower than for the other methods in most cases. More surprising, however, is the pattern it yields with almost constant error levels for several different numbers of steps, followed by a jump to another almost constant level. A closer look at the trajectories and the step sizes of two runs using 42 and 65 steps respectively is shown in figure 14. In figure 7 these runs can be seen to have very similar global errors at the end point, namely $7.498 \cdot 10^{-3}$ and $7.428 \cdot 10^{-3}$ respectively. As can be seen in figure 14, the step sizes prove to be exactly the same from the first through the ninth step, which is equivalent to approximately 71% of the total trajectory distance. Hence it appears that differences between the two trajectories introduced after this point are too late and too small to lead to a large difference in the global error at the end point.

Looking instead at the RKF45 trajectories and step sizes from two runs with very similar number of steps but different errors (figure 15) reveals an important difference. The studied runs are one taking 65 steps and one taking 67, producing global errors of $7.428 \cdot 10^{-3}$ and $1.729 \cdot 10^{-3}$ respectively at the end point. Figure 15 reveals that the first step in the trajectory with 67 steps is half as long as the first step of the 65 step trajectory. This introduces a difference of $1.00 \cdot 10^{-3}$ between the two solutions already at the second point of the 65 step trajectory and the third point of the 67 step trajectory. This difference then propagates throughout the solutions and eventually grows to a significant global error difference, despite the overall similarity of the step size distributions.

The size of the first step in these experiments is not explicitly set. Instead, it is picked by setting the step size equal to the entire run time, and letting the RKF45 try to take steps and reject them, effectively halving the step size until a step is accepted. The purpose of this technique is to not explicitly affect the efficiency of the different runs by giving them step sizes so small that they have to double them repeatedly to reach a more stable level. Since the first step appears to affect the global error at the end point so significantly though, it might be a good idea to set the starting step size to something moderately small in practice. Depending on the length of the run it might not play as big a role, but delaying the introduction of errors could certainly be impactful since local numerical errors accumulate and can lead to large deviations of the trajectories. It also comes at a low computational cost, unless the seeded starting step is set incredibly small. One way of seeding the starting step size could be to put a stricter tolerance for accepting the starting step, say $T/5$, thus making sure that it is relatively small while still not forcing repeated doubling of it for the first few steps of a simulation.

Another visible difference between the trajectories in figure 15 are the positions of the steps along the eastern edge. Although they appear to trace out roughly the same curve, the data points are spaced out on different positions. This could possibly have an impact on the end result, depending on how fine the grid is and how it is interpolated, which would further increase the difference between the end results of the two solutions.

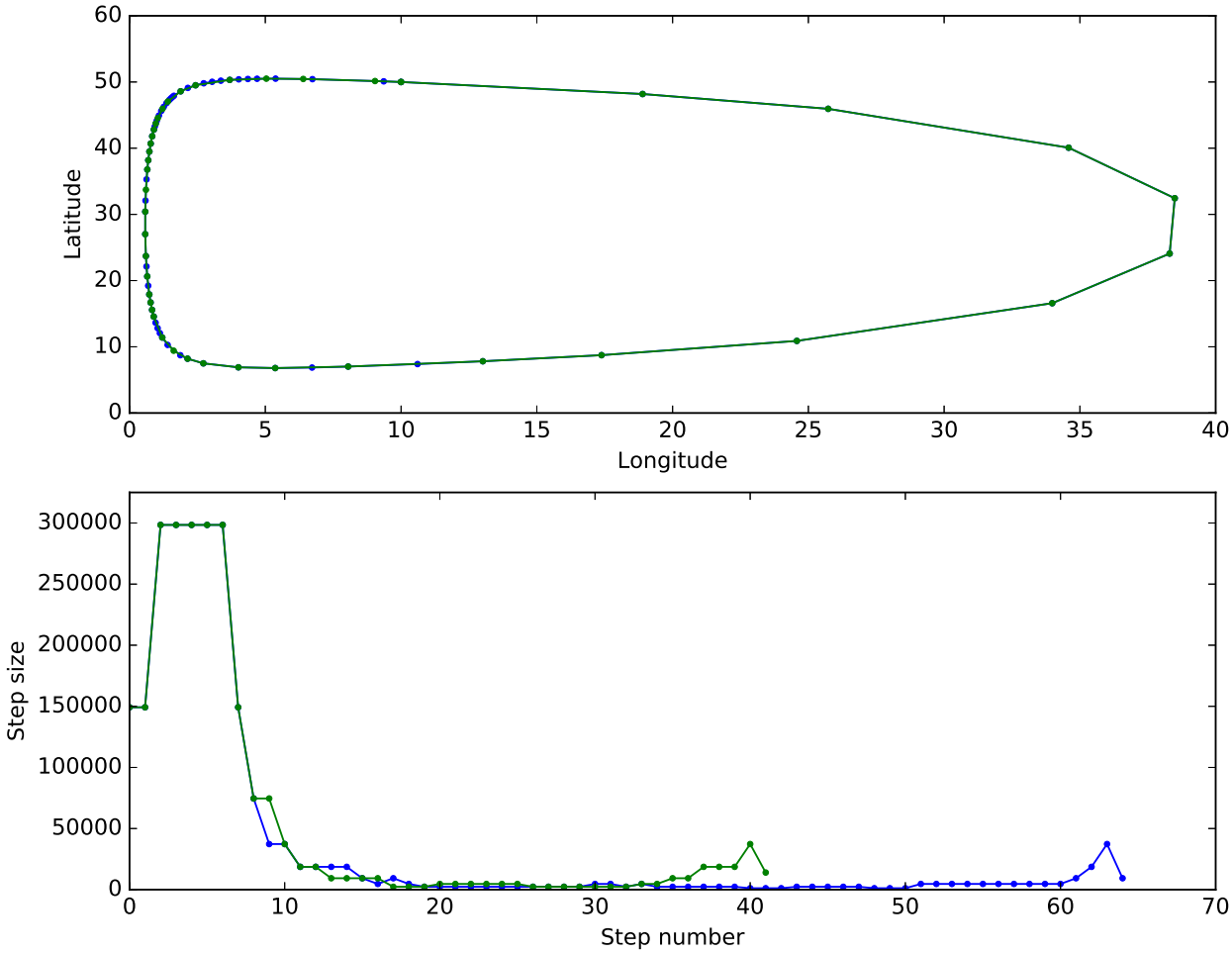

Figure 14: Trajectories (top) and step size distributions (bottom) for two runs on the Stommel test, producing very similar end results while using significantly different numbers of steps. The green trajectory uses 42 steps and produces a global error of $7.498 \cdot 10^{-3}$ at the end point, whereas the blue trajectory uses 65 steps and produces an error of $7.428 \cdot 10^{-3}$.

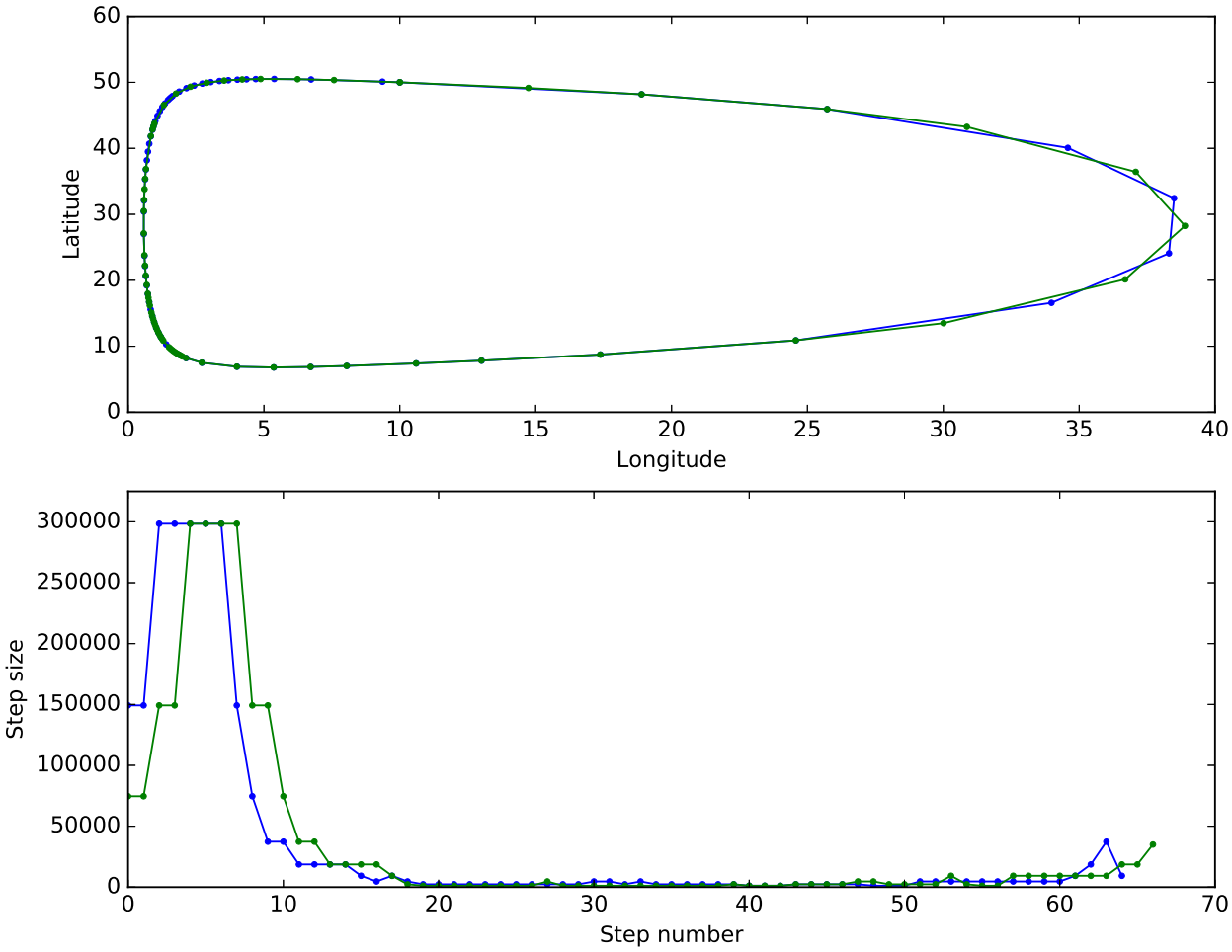

Figure 15: Trajectories (top) and step size distributions (bottom) for two runs on the Stommel test, producing very different end results while using very similar numbers of steps. The green trajectory uses 67 steps and produces a global error of $1.729 \cdot 10^{-3}$ at the end point, whereas the blue trajectory uses 65 steps and produces an error of $7.428 \cdot 10^{-3}$.

Bilinear interpolation was used in the Stommel test to compute the velocities of the particles. This is the method currently used by PARCELS to interpolate grids in JIT mode, but to examine the impact of this interpolation on the errors and see if it could explain some of the strange patterns in figure 7, another test was conducted on the Stommel case. This time around, the particle velocities were computed analytically by solving equation (14).

Using the same tolerances as described above, the errors displayed in figure 16 were achieved. Similar patterns to those found in figure 7 appear again, but the difference in number of steps used by RKF45 for the same tolerances is remarkable, as is its steeper slope. When using bilinear interpolation more than 100 steps were needed to achieve errors below 10^{-4}, and only a few runs did this. Using the analytical velocities, however, this level of accuracy was reached in 54 steps. This shows that RKF45 could potentially benefit greatly from a more accurate interpolation scheme than the currently used bilinear one.

Running the test again with tolerances ranging from $10^{-8.5}$ to 10^{-12} instead, displayed in figure 17, shows a stabilization in the slope of the errors for both RKF45 and RK4 (as well as EE) with smaller time steps. The performance of RK4 is similar to the results in figure 7, but RKF45 again shows significant improvements even for smaller time steps. The slope for both of them are about -4, which is the expected slope from a 4th order method.

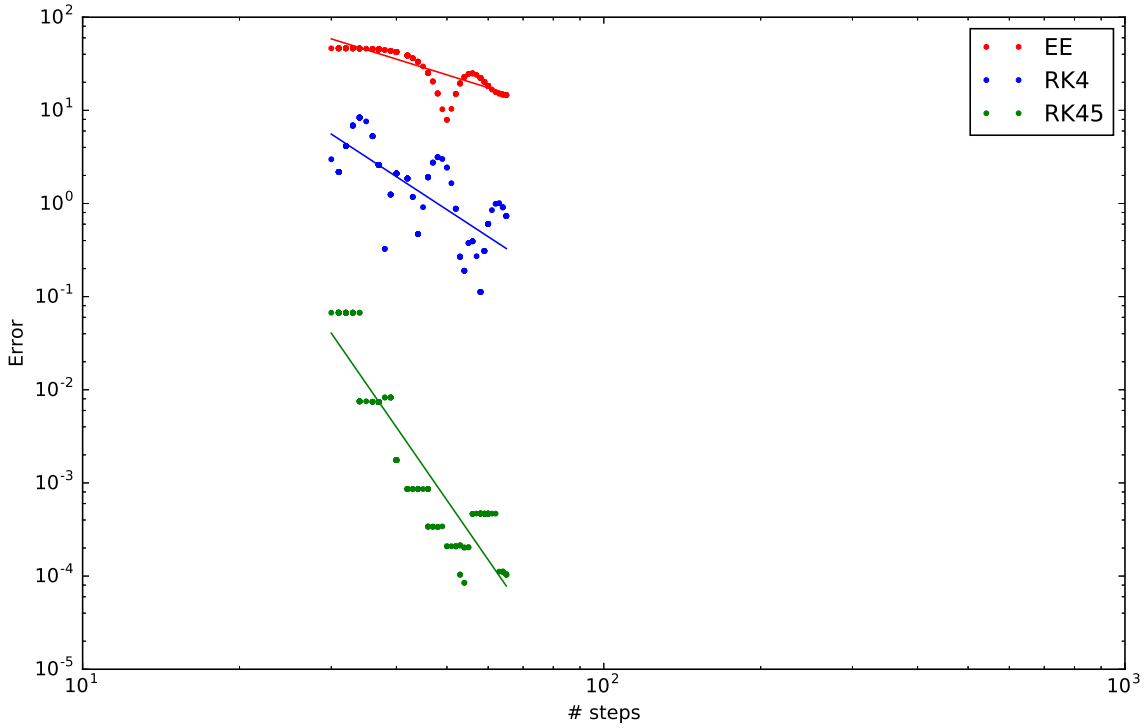

Figure 16: Global error at the end point of RKF45, RK4 and EE using analytically calculated velocities in the Stommel test. 200 runs were performed, with the number of steps used picked by running RKF45 first with tolerances evenly spaced on a logarithmic scale from $10^{-8.5}$ to 10^{-10}, and then using the number of steps used in these runs for RK4 and EE.

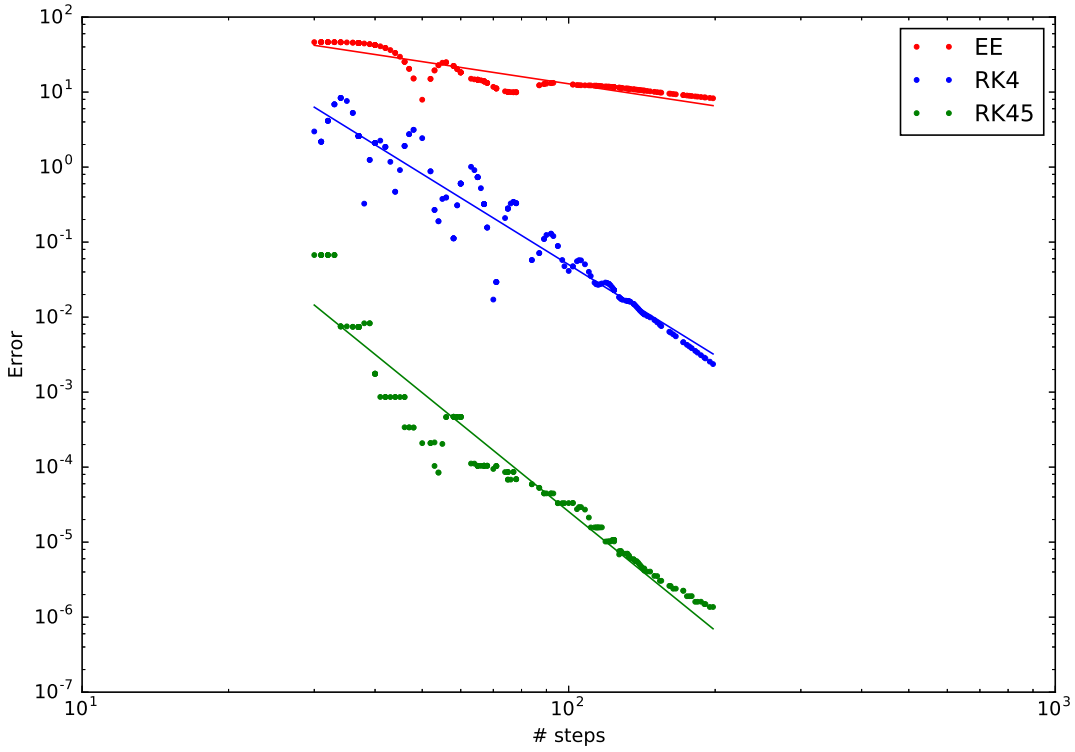

Figure 17: Global error at the end point of RKF45, RK4 and EE using analytically calculated velocities in the Stommel test. 200 runs were performed, with the number of steps used picked by running RKF45 first with tolerances evenly spaced on a logarithmic scale from $10^{-8.5}$ to 10^{-12}, and then using the number of steps used in these runs for RK4 and EE.

Another aspect of the design of the Stommel tests used here that could affect the results is the choice of starting position. In the above experiments, the trajectories start by tracing the slower parts of the region and reach the troublesome western part towards the end. Starting instead at $(x, y) = (10, 7.296)$, in the southwestern part of the region, could benefit RKF45 by restricting the time step size in the beginning of the trajectory, thus limiting the local error introduced early. It might also be beneficial for RK4 and EE, by minimizing the accumulated global error at the start of the problematic western region.

Figure 18 displays the error achieved when the starting position is switched to $(x, y) = (10, 7.296)$, using bilinear interpolation to compute velocities. The errors produced by EE are generally smaller here than in figure 7. This is likely because it goes out of bounds closer to the starting position, which means that it still fails quite badly.

The RK4 errors don't follow an obvious pattern for the longer time steps anymore, but at around 120 steps it reaches a fairly consistent slope similar to the one in figure 7. The magnitude is a bit larger here, however, implying that RK4 benefits from starting with the nicer parts of the trajectory.

The RKF45 errors in figure 18 do not recreate the level pattern from figure 7. Instead the errors appear more randomly scattered, but still with a slight slope as step sizes decrease. The error magnitudes and number of steps used are still quite similar to the ones achieved in figure 7, implying that the overall results for RKF45 are not greatly affected by switching starting position.

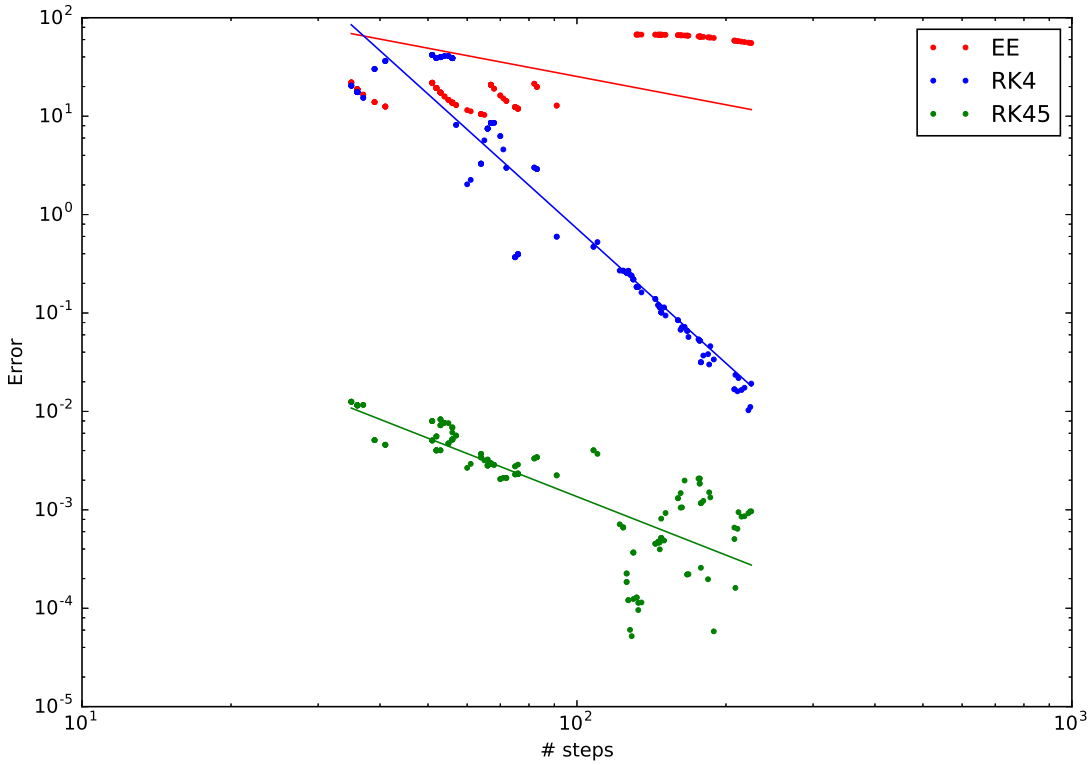

Figure 18: Global error at the end point of RKF45, RK4 and EE using bilinearly interpolated velocities in the Stommel test, with starting point at $(x, y) = (10, 7.296)$. 200 runs were performed, with the number of steps used picked by running RKF45 first with tolerances evenly spaced on a logarithmic scale from $10^{-8.5}$ to 10^{-10}, and then using the number of steps used in these runs for RK4 and EE.

Even though the RKF45 and RK4 runs in the Stommel test yield similar errors when the tolerance for RKF45 is $1.06 \cdot 10^{-9}$ and the number of steps used by RK4 is $\gamma = 2.4$ times more than with RKF45, this is not true in the Agulhas test. The two–sample KS test shows that the errors yielded by RK4 and RKF45 in the Agulhas test are from different distributions, and looking at figure 9 shows that the errors from the RK4 run are smaller than those from the RKF45 run.

The reason why RKF45 gives a more accurate solution in so much fewer steps in the Stommel test is likely because it is a typical example of a function where RK4 would be expected to perform poorly compared to RKF45. The velocity in

the Stommel test along the ground truth trajectory ranges from 0.039 to 9.006, whereas in the Agulhas test it ranges from $7.325 \cdot 10^{-5}$ to 2.385. From the earlier discussion about figure 14 we know that the RKF45 run using 45 steps in the Stommel test covers the first 71% of its entire path in just 9 steps, leaving 36 steps for the last 29% of the path. RK4, on the other hand, uses 37 steps to cover the first 71% of the path, leaving just 8 steps for the last 29%. In the Agulhas test the smaller range and nicer distribution of the velocities spaces the RK4 steps more evenly, and thus doesn't "waste" as many steps on the nicer parts of the solution only to perform poorly and inaccurately at other parts.

Not only do the RK4 runs in the Agulhas test produce smaller errors than the RKF45 runs, they do it in very similar computation times. Theoretically, the computation times are expected to be strongly linked to the number of function evaluations used, since this is the computationally "expensive" part of the time steps. RKF45 is a 6–stage method, which means that in every step 12 function evaluations are performed, i.e. 6 in each dimension. RK4 is a 4–stage method, and thus uses 8 function evaluations per step. This means that an RKF45 step is expected to take roughly 1.5 times as long as an RK4 step. In addition to this, an RKF45 step has a chance to be rejected, thus having to be recalculated. In the current implementation the step is recalculated from scratch, i.e. no velocities from the rejected step are saved and used in the recalculation. Since RK4 uses approximately 2.4 times as many (accepted) steps as RKF45, assuming RK4 and RKF45 take the same time (see table 1) and time is directly proportional to number of function evaluations used, RKF45 would have to reject every $2.4/1.5 = 1.6$ steps, which is unlikely.

Running a similar test, but with all calculations performed directly in Python instead of using JIT compilation, yields the results displayed in table 3. For this particular test only three different particle trajectories were used. It is not enough to make any statements about the actual ratio between function evaluations used by Python and JIT, but table 3 does show that in Python the ratio between computation time and number of function evaluations for RK4 and RKF45 is about the same. This implies a correlation between the two. In JIT, however, the computation times are again quite similar for both methods, with RK4 even being slightly faster than RKF45. This adds credibility to the hypothesis that there should be a strong correlation between the number of function evaluations and computation time, but that the function evaluations in JIT are relatively cheap, thus making overhead computation in RKF45 relatively more expensive.

Table 3: Comparison between Python and JIT time results, using three different particle trajectories in the Agulhas region. The ratio column shows the RK4 result divided by the RKF45 result, to illustrate the connection between computation time and number of function evaluations. The ratio between number of steps used is excluded since it is irrelevant.

	RK4 (s)	RKF45 (s)	ratio
Python time (300 part.)	36.5	30.8	1.19
JIT time (300000 part.)	42.1	44.7	0.94
# steps computed	405	225	-
# function evaluations	3240	2700	1.20

The function evaluations might not continue to be this cheap as the development of PARCELS continues, however. The argument for the current usage of bilinear interpolation of the grids is mainly simplicity. Bilinear interpolation only requires values from the four closest grid points, but to increase the degree of the interpolation, it also needs to consider grid points further away. This effectively makes the interpolation algorithm more advanced and time consuming in itself, and also requires a more advanced strategy for the boundary condition implementation. It may however provide a more accurate solution than the current bilinear interpolation. Thus, if each function evaluation is more expensive, RKF45 could benefit more from using less function evaluations than RK4. There might also be ways to make RKF45 more efficient by making a more in–depth analysis of its overhead computation time. Also, nifty tricks such as saving the velocities from the starting point of a step, thus not having to recalculate them if the step is rejected, could be utilized to increase its speed slightly. However, the current results don't imply any great performance improvements gained by switching from RK4, so efforts to improve the performance of PARCELS might be better applied in other areas.

5.2 Boundary conditions

The objective for the boundary conditions was to prevent particles from getting stuck along the coastlines, and a quick look at table 2 confirms that the new conditions are indeed very successful in the Agulhas test case.

As mentioned, there is still a way, albeit unlikely, for particles to get stuck if they cross onto land in two directions simultaneously, e.g. if the dark green particle in figure 4 ends up north of the two blue land grid points. From the results in the Agulhas region this is not expected to happen very frequently, and it is thus considered an acceptable loss. In most real simulations millions of particles will be used, and losing a tiny fraction of them will almost certainly not affect results much. However, it is possible to prevent even this beaching risk by changing the conditions slightly. Consider again the light green particle in figure

4. The function to interpolate a velocity field can only be called on one field at a time, i.e. information about the u field cannot be used in the interpolation of the v field. However, the light green particle in figure 4 has two NaN u field points just north of it (although with ghost point velocities displayed in figure 4). A northern boundary will always be marked by NaN v grid points. Thus, by checking the u grid points just east (not NaN) and west (NaN) of the NaN u points just north of the light green particle, it can be determined that the northwestern quadrant of the "cell" spanned by the four blue points in figure 4 is land. Using this information, instead of simply ghost pointing the two southern u points as shown in figure 4, they can be mirrored to point east away from the boundary, effectively preventing the light green particle from ending up at the dark green spot.

There are two obvious arguments for using this method instead of the one I have implemented: the boundary between land and sea becomes more well–defined (i.e. no particles have to be pushed back into the sea like the dark green particle in figure 4), and the slight risk of particles beaching is diminished. The two main arguments for my conditions, however, are that the conditions that I have implemented already prevents the majority of particles from beaching, and that they are simpler. The latter of these arguments is actually quite important. This is because of the early state that the implementation is currently in. The conditions I have implemented assume a 2D Arakawa C–grid, and is applicable to the very specific case that the user simply wants particles to flow freely and not get stuck along coastlines. There is a lot more work to be done on the implementation of boundary conditions in general in PARCELS, and this work will not be done by me. Thus there is a great benefit in keeping the conditions relatively simple to make it easier for another programmer to pick up the work where I have finished.

There is also work left to be done on the way these conditions are implemented. Ideally, the time stepping should be divided into two separate cases: the "common" case, where the particle is in open water away from the shore, and the "special" case, where the particle is close to a boundary. The point of keeping these cases separate is that a large majority of particles are expected to be well away from the boundaries during most of the simulation time, so this case should ideally be made as fast as possible. Only when a particle is close to a boundary, special functions are called to handle this. In the current implementation, this check is done by explicitly determining that none of the particle's nearest grid points are NaN. To speed it up, one could instead e.g. utilise a predefined map that quickly determines when the particle is in the "common" open water case, and when boundary conditions have to be considered. This separation is also something that will not be done by me, which further motivates the desire to keep the boundary conditions simpler and comprehensible.

6 Conclusions

In the idealized Stommel test case, RKF45 needs significantly fewer steps than RK4 to produce results with similar accuracy for most tolerances. In the Agulhas test case however, the correlation found in the Stommel test doesn't hold. This is probably because of the extreme variations in velocity in the Stommel test, which favors a method with adaptive step size.

Despite the higher number of steps used by RK4 in the time test, both methods run at similar computation times, meaning that RK4 outperforms RKF45 since it produces a smaller error. However, the number of function evaluations, which is expected to be closesly related to the computation time, used by RKF45 seems to be smaller. This implies that other overhead calculations in RKF45 is what causes the delay compared to RK4. The method currently used for interpolating the grids is linear interpolation, which is a cheap but perhaps not very accurate method. As the development of PARCELS continues, maybe a more advanced and accurate interpolation method will be implemented, making each function evaluation more computationally expensive. This could in turn mean that RKF45 might be able to compete with RK4, especially since it performs so much better in the Stommel test when an analytical solution is used. In any case, the potential benefit of switching from RK4 to an embedded RK method currently seems to be quite limited, and will thus probably not be a major focus in the further development of PARCELS.

The new free–slip boundary conditions, on the other hand, were more successful. They successfully prevented particles from getting stuck along the shorelines in the Agulhas region. There is still a very slight risk of particles getting stuck, but this risk is accepted to keep the boundary conditions simple and comprehensible.

References

[1] O. Conservancy, M. C. for Business, and Environment, "Stemming the tide: Land-based strategies for a plastic-free ocean," McKinsey Center for Business and Environment, Tech. Rep., 2015.

[2] J. R. Jambeck, R. Geyer, C. Wilcox, T. R. Siegler, M. Perryman, A. Andrady, R. Narayan, and K. L. Law, "Plastic waste inputs from land into the ocean," *Science*, 2015.

[3] L.-M. Lebreton, S. Greer, and J. Borrero, "Numerical modelling of floating debris in the world's oceans," *Marine Pollution Bulletin*, 2012.

[4] E. van Sebille, C. Wilcox, L. Lebreton, N. Maximenko, B. D. Hardesty, J. A. van Franeker, M. Eriksen, D. Siegel, F. Galgani, and K. L. Law, "A global inventory of small floating plastic debris," *Environmental Research Letters*, 2015.

[5] E. van Sebille et al., "Lagrangian ocean analysis: fundamentals and practices," unpublished.

[6] A. Bennett, *Lagrangian Fluid Dynamics*. Cambridge University Press, 2006.

[7] R. Salmon, *Lectures on Geophysical Fluid Dynamics*. Oxford University Press, 1998.

[8] G. Madec and the NEMO team, "Nemo ocean engine," Institut Pierre-Simon Laplace, Tech. Rep., 2016.

[9] Consortium, "Globcurrent user requirements document," NERSC, Tech. Rep., 2014.

[10] E. Hairer, S. P. Nørsett, and G. Wanner, *Solving Ordinary Differential Equations I - Nonstiff Problems*. Springer-Verlag, Berlin, 1987.

[11] A. Iserles, *A First Course in the Numerical Analysis of Differential Equations*. Cambridge University Press, 2008.

[12] F. P. Miller, A. F. Vandome, and J. McBrewster, *Bilinear Interpolation*. VDM Publishing, 2010.

[13] S. M. Griffies, C. Böning, F. O. Bryan, E. P. Chassignet, R. Gerdes, H. Hasumi, A. Hirst, A.-M. Treguier, and D. Webb, "Developments in ocean climate modelling," *Ocean Modelling*, 2000.

[14] A. Arakawa and V. R. Lamb, "Computational design of the basic dynamical processes of the ucla general circulation model," *New York, Academic Press*, 1977.

[15] S. A. Moore, "Lateral boundary conditions in numerical ocean models," Ph.D. dissertation, University of Reading, 2004.

[16] X. L. Qui and K. Q. Xia, "Spatial structure of the viscous boundary layer in turbulent convection," *Physical Review E*, 1998.

[17] N. Fabbroni, "Numerical simulations of passive tracers dispersion in the sea," Ph.D. dissertation, University of Bologna, 2009.

[18] J. Pedlosky, *Geophysical Fluid Dynamics.* Springer-Verlag, 1987.

[19] L. M. Beal, W. P. M. D. Ruijter, A. Biastoch, R. Zahn, and S. W. G. 136, "On the role of the agulhas system in ocean circulation and climate," *Nature*, 2011.

[20] W. Conover, *Practical Nonparametric Statistics.* Wiley India Pvt. Limited, 2006.