Numerical Integration and Interpolation in Marine Pollutant Transport Modelling

Tor Nordam*, Ute Brönner, Jørgen Skancke, Raymond Nepstad,
Petter Rønningen and Morten O. Alver
SINTEF Ocean
Trondheim, Norway
*tor.nordam@sintef.no

Abstract

Transport modelling of oil and other chemicals in the ocean relies on information on current velocity as a function of position and time, $\mathbf{v}(\mathbf{x},\ t)$. These data will in many cases be supplied from an ocean circulation model, which produces a velocity field with vectors given at discrete points on a four-dimensional grid. If we consider for simplicity a passive tracer that moves with the water velocity at its position, then calculating the trajectory of the tracer, $\mathbf{x}(t)$, starting from a position $\mathbf{x}_0$ when $t = 0$, can be done by numerically integrating the differential equation $\dot{\mathbf{x}} = \mathbf{v}(\mathbf{x},\ t)$, with the initial condition $\mathbf{x}(t = 0) = \mathbf{x}_0$.

This immediately raises three questions that every transport modeller must answer: Which integrator should one use, what timestep to use, and how does one evaluate the current velocity field at locations between grid points? In this paper, we investigate some of these issues by extensive numerical experiments. The most basic choice in marine pollutant transport models is the combination of the Forward Euler integrator ($\mathbf{x}(t + \Delta t) = \mathbf{x}(t) + \mathbf{v}(\mathbf{x},\ t)\Delta t$) with nearest-neighbour interpolation (constant values within each grid cell). This is compared to the more advanced options of 2nd-, 3rd- and 4th-order Runge-Kutta methods, combined with spline interpolation of first, second and third order in both space and time. As the velocity field, we have used both an analytically expressed double gyre field, and three nested datasets from the ROMS model, at 800 m, 4 km and 20 km horizontal resolution.

The main goal of using more advanced numerical integration schemes is to save computational time by allowing the use of a longer timestep, while still obtaining the same accuracy as from a simpler method with a shorter timestep. Hence, the choice of timestep is investigated in detail. Our conclusions are that there is no benefit from using a high-order integrator, unless accompanied by a high-order interpolator, and that in practice, the combination of linear interpolation and a 2nd-order integrator delivers the best balance between accuracy and computational effort. Finally, we find that using the same transport timestep as that of the underlying current dataset is sufficient in the cases we have investigated.

1 Introduction

Trajectory modelling in the marine environment is used to track the position of "objects of interest" over time. Practical applications include floating objects trajectories for Search and Rescue operations (SaR, see e.g. De Dominicis et al. (2013)), particle transport (see e.g. Döös et al. (2016)), oil spill modelling (see e.g. French-McCay (2003); Reed et al. (2000); Zelenke et al. (2012)) or modelling of chemical discharges (see e.g. Reed and Hetland (2002)). A common modelling approach is Lagrangian modelling, where "Lagrangian elements" represent the modelled objects and are transported by the forces of the marine environment produced by wind, currents, waves, turbulence, etc. Among the advantages of Lagrangian modelling is that the many properties and characteristics of the transported object or substance, e.g. oil, can be straightforwardly represented and tracked over time, in addition to merely predicting a spatial position. Lagrangian transport modelling furthermore benefits from good numerical stability, and performance will usually degrade gracefully with reduced resolution (longer timesteps, fewer Lagrangian elements).

Nordam, T., U. Brönner,J. Skancke, R. Nepstad, P. Rønningen, and M.O. Alver, Numerical Integration and Interpolation in Marine Pollutant Transport Modelling, Proceedings of the Fortieth AMOP Technical Seminar, Environment and Climate Change Canada, Ottawa, ON, pp. 586-609, 2017.

For each Lagrangian element, the transport model will have to solve the transport equation, which in its simplest form means numerically integrating the differential equation $\dot{\mathbf{x}} = \mathbf{v}(\mathbf{x}, t)$ for advection. Trajectory modelling of oil and other chemicals in the ocean relies on information on, e.g., current velocity as a function of time and location, $\mathbf{v}(\mathbf{x}, t)$. These data will in many cases be modelled data themselves, supplied from an ocean circulation model. These models produce a velocity field with vectors given at discrete points, often on a four-dimensional grid with a given resolution in space and time. For the modeller this poses three challenges: (1) choice of an integration method to compute advection for a timestep, (2) the choice of this timestep (its length) and (3) the choice of an interpolation method to determine the value for $\mathbf{v}(\mathbf{x}, t)$ in the advection equation. These choices have to be made under the constraints of computational costs (time) versus the cost of accuracy:

- Short timesteps can be assumed to give good accuracy (almost) independent of the chosen integration method, but come with high computational costs.
- Higher order methods for integration and interpolation increase accuracy, but will also increase the computational cost at each step.

Different modelling applications have their own requirements of spatial and temporal resolution, and may have different criteria for balancing computational cost against accuracy. For example, oil spill simulations covering large areas can operate at timesteps of 1 hour or more, where the oil might be transported 800 m on average per timestep. For simulation of oil well drilling discharges, however, the majority of the released mass may be deposited within 1 km of the release point, and consequently higher resolution in both time and space is required.

Newer applications of transport models for operational monitoring in (near) real time, with the purpose of direct feedback and decision making, creates higher expectations of accuracy (see e.g. Brönner et al. (2013); Hodges et al. (2015)). Furthermore, oil spill models are used for hindcast modelling of important events, like the Deepwater Horizon incident (see e.g. Barker (2011); North et al. (2011)). In these cases, the availability of measurements and observations for comparison against model results can also lead to an increased focus on accuracy in the modelling. Advances in computational power in recent years allow the production of oceanographic data at increasingly high resolution. In order to take advantage of the increased resolution in the velocity data, care must be taken in the selection of integrator, interpolator and timestep in the transport model.

1.1 Integration and Interpolation

It is common knowledge that the 4th-order Runge-Kutta integrator gives better stability and accuracy than the Forward Euler method, and there are several papers referring its use in oil spill transport modelling or other marine transport problems (see e.g. Mariano et al. (2011); Price et al. (2004)). However, these papers do not discuss interpolation of the velocity field at arbitrary time and location, which is a requirement for any numerical integration scheme, and which will be shown here to have significant effect on the results. There is also work referring to interpolation in time, suggesting it should be fourth order, to match the fourth order integrator, but with no discussion of interpolation in space (García-Martínez and Flores-Tovar, 1999). A more general paper exists, doing a more theoretical analysis of the effect of interpolation on particle path calculations in 3D, but again discussing interpolation in time only (Darmofal and Haimes, 1996). Another general study considered the effect of various interpolation schemes in space and time, with the variable-timestep integrator Runge-Kutta-Fehlberg, but using an analytical expression for a flow field that was then gridded before interpolation (Mancho et al., 2005). In this paper, we seek to analyse the interaction of nu-

Nordam, T., U. Brönner,J. Skancke, R. Nepstad, P. Rønningen, and M.O. Alver, Numerical Integration and Interpolation in Marine Pollutant Transport Modelling, Proceedings of the Fortieth AMOP Technical Seminar, Environment and Climate Change Canada, Ottawa, ON, pp. 586-609, 2017.

merical integrators with interpolation in both space and time, using realistic modelled ocean current data of different resolutions.

The aim of the current paper is to assess the numerical error in transport modelling results given by the combination of the choices for

- interpolation method
- integration method
- timestep
- spatial resolution of current data

while keeping in mind the practical consideration of "How accurate is 'accurate enough' for the purpose at hand?"

To limit the scope of the study, we consider only transport by currents in the horizontal plane. We ignore wind, buoyancy and the presence of boundaries such as the coast or the sea bed. Sub-grid turbulent mixing is also ignored. Hence, we are considering the pure advection problem in two dimensions.

In our simulations, we represent the advected matter as a field of 10000 particles, or "Lagrangian elements", which are passively transported by a velocity field. We consider one case where the velocity field is expressed in analytical form, and one case using real ocean current data sets. The ocean data are provided on regular quadratic grids, in three different horizontal resolutions (800 m, 4 km and 20 km, all with timestep 1 hour). We calculate the global error in the trajectory of each particle, and discuss the average and mean errors found with different integrators, interpolators and timesteps.

We have chosen to limit the study to the Runge-Kutta family of methods, considering methods of order 1 to 4 with a constant timestep of integration. For interpolation, we have used nearest neighbour, linear, quadratic spline and cubic spline interpolation both in the horizontal plane and time, using the same order of interpolation in all directions.

2 Theory

2.1 Transport Equation

The advection-diffusion-reaction equation for a concentration, c, of a substance reads

$$\frac{\partial c}{\partial t} = \nabla \cdot (D \nabla c) - \nabla \cdot (\mathbf{v} c) + R, \tag{1}$$

where $\mathbf{v}$ is the advection velocity, D is the diffusivity and R is the reaction term, where $\mathbf{v}$ and D are in general functions of $\mathbf{x}$ and t, while R can in principle be a function of $\mathbf{x}$, t, and c, as well as the concentration of other substances. All numerical oil spill models essentially solve this equation, in one way or another. If we ignore reactions ($R = 0$), and let D be a constant in space ($\nabla D = 0$), the equation simplifies to

$$\frac{\partial c}{\partial t} = D \nabla^2 c - \nabla \cdot (\mathbf{v} c), \tag{2}$$

which can be shown to be equivalent to an infinite ensemble of particles (each representing a mass m_i, such that $\sum m_i = \int c \, \mathrm{d}V$) with positions governed by the equation

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \mathbf{v}\delta t + \boldsymbol{\xi}_i \sqrt{2nD\delta t}, \tag{3}$$

for sufficiently small δt (Visser, 2008). Here, n is the number of spatial dimensions (in which the diffusion takes place), and $\boldsymbol{\xi}_i$ is a random vector whose length is of zero mean and unity variance, and whose direction is uniformly distributed.

Nordam, T., U. Brönner,J. Skancke, R. Nepstad, P. Rønningen, and M.O. Alver, Numerical Integration and Interpolation in Marine Pollutant Transport Modelling, Proceedings of the Fortieth AMOP Technical Seminar, Environment and Climate Change Canada, Ottawa, ON, pp. 586-609, 2017.

The topic of the current paper is to study the numerical calculation of advection by gridded velocity fields. Hence, we will further simplify our system, by setting $D = 0$. In this case, the trajectory of a particle being advected passively through a velocity field is defined by the ordinary differential equation (ODE)

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, t), \tag{4}$$

where $\mathbf{f}(\mathbf{x}, t) = \mathbf{v}(\mathbf{x}, t)$ is the velocity at time and position $(\mathbf{x}, t)$, along with an initial condition, $\mathbf{x}(t = t_0) = \mathbf{x}_0$.

2.2 Numerical Integration

Finding the solution of an initial value problem by numerical means is known as "numerical integration" of the differential equation. A large body of literature exists on the topic of numerical integration, and a range of different techniques exist, including many special purpose methods (see e.g. Hairer and Wanner (1996); Hairer et al. (1993, 2006)).

In this paper, we will consider four different methods from the Runge-Kutta family of methods, including the frequently used Forward Euler method, and 4th-order Runge-Kutta (commonly known as *the* Runge-Kutta method). Common to all of these of methods is that we deal with time as a discrete variable, with a fixed timestep, h. For convenience, we introduce the notation

$$t_i = t_0 + ih, \tag{5}$$

and we let $\mathbf{x}_i$ denote the numerically obtained solution at time t_i. Furthermore, we let $\mathbf{x}(t_i)$ be the true solution at time t_i (although note that $\mathbf{x}(t_i)$ is usually not known).

2.3 Error Estimates

Since numerical integration is most commonly used in situations where the exact solution is unknown, it becomes necessary to estimate the error by purely numerical means. In general, the idea is that a smaller timestep, h, gives a more accurate solution, and as $h \to 0$, the numerically obtained solution converges to the true solution. The rate of convergence depends on the chosen integration method. There are two important measures of the error: The local error and the global error. The local error is the error made in a single step. Assume there is no error in the position at time t_{k-1}, i.e., $\mathbf{x}(t_{k-1}) = \mathbf{x}_{k-1}$. Then, the local error in step k is given by

$$e(h) = \mathbf{x}(t_k) - \mathbf{x}_k. \tag{6}$$

The global error, on the other hand, is the error at the end of the computation, at time t_N:

$$E(h) = \mathbf{x}(t_N) - \mathbf{x}_N. \tag{7}$$

If the local error is, e.g., $\mathcal{O}(h^2)$, then the global error will be $\mathcal{O}(h)$, since the number of steps, N, is $\sim 1/h$. In this case, when the global error is proportional to h, the method is said to be first order, while a method where $E(h) \sim h^2$ is said to be second order, etc.

It can be shown that for a Runge-Kutta method of order p, and for an ODE given by $\dot{\mathbf{x}} = f(\mathbf{x}, t)$, where all partial derivatives of $f(\mathbf{x}, t)$ up to order p exist and are continuous, the local error is bounded by

$$|\mathbf{x}(t_0 + h) - \mathbf{x}_1| \leq Ch^{p+1}, \tag{8}$$

where C is some constant, which depends on the method, and on the partial derivatives of $f(\mathbf{x}, t)$ (Hairer et al., 1993).

To obtain a numerical estimate of the local error for a given method and problem, it is possible to use a technique first described by Richardson (1910) (see also Hairer et al. (1993)). With a Runge-Kutta method of order p, numerically integrating the ODE $\dot{\mathbf{x}} = f(\mathbf{x}, t)$, starting from an initial value $(\mathbf{x}_0, t_0)$, we take first two steps, of size h, which gives the numerical results $\mathbf{x}_1$ and $\mathbf{x}_2$. We then again start from the same initial value, taking one large step, of length $2h$, obtaining the numerical result $\mathbf{w}$. Then it can be shown that the difference between the true solution, $\mathbf{x}(t_0 + 2h)$, and $\mathbf{x}_2$, is given by

$$\mathbf{x}(t_0 + 2h) - \mathbf{x}_2 = \frac{\mathbf{x}_2 - \mathbf{w}}{2^p - 1} + \mathcal{O}(h^{p+2}). \tag{9}$$

An important point for our purposes is that these error estimates are only guaranteed to hold if all partial derivatives up to order p of the velocity field, $\mathbf{v}(\mathbf{x}, t)$, exist and are continuous. This has implications for how we should treat the gridded velocity field used in a particle transport simulation. For example, if one chooses to use nearest-neighbour interpolation, i.e., constant values in a grid cell, the first partial derivatives of the interpolated velocity field are 0 almost everywhere, and discontinuous at cell boundaries. Using linear interpolation, the first partial derivatives will be constant inside a cell, but again discontinuous at cell boundaries. Only by advancing to second order interpolation (for example quadratic splines) are we guaranteed to have continuous first partial derivatives.

2.4 Runge-Kutta Methods

The simplest method from the Runge-Kutta family is commonly known as the Forward Euler method. Given a position at a time t_i, $\mathbf{x}_i$, and the velocity $\mathbf{v}(\mathbf{x}_i, t_i)$ at that time and position, it calculates the position a time h later:

$$\begin{aligned} \mathbf{k}_1 &= \mathbf{v}(\mathbf{x}_i, t_i) \\ \mathbf{x}_{i+1} &= \mathbf{x}_i + \mathbf{k}_1 h. \end{aligned} \tag{10}$$

This is clearly an approximation, as one in practice treats the velocity field as constant over the time and distance of each step. The error can be reduced by making the timestep, h, smaller, but since the number of steps increases when the timestep becomes shorter, this increases the total amount of work done. This method has a local error that scales as $e(h) \sim \mathcal{O}(h^2)$, and a global error $E(h) \sim \mathcal{O}(h)$. Hence, this is said to be a first-order method.

More advanced methods generally use two or more evaluations of the velocity field, calculating a weighted average, using this to calculate the next position. A common example is the method known as the Improved Euler method, or Heun's method, which is one of several examples of a two-stage Runge-Kutta method of order 2.

$$\begin{aligned} \mathbf{k}_1 &= \mathbf{v}(\mathbf{x}_i, t) \\ \mathbf{k}_2 &= \mathbf{v}(\mathbf{x}_i + \mathbf{k}_1 h, t + h) \\ \mathbf{x}_{i+1} &= \mathbf{x}_i + \left(\frac{1}{2}\mathbf{k}_1 + \frac{1}{2}\mathbf{k}_2 \right) h. \end{aligned} \tag{11}$$

Note that this 2nd-order Runge-Kutta method only evaluates the velocity field at times $t_i = t_0 + ih$, where i is an integer. Moving on to order 3, we have chosen to consider a scheme known as Kutta's method, which uses three evaluations of $\mathbf{v}(\mathbf{x}, t)$, including at half-timestep

offsets.

$$\begin{aligned} \mathbf{k}_1 &= \mathbf{v}(\mathbf{x}_i, t) \\ \mathbf{k}_2 &= \mathbf{v}(\mathbf{x}_i + \mathbf{k}_1 h/2, t + h/2) \\ \mathbf{k}_3 &= \mathbf{v}(\mathbf{x}_i - \mathbf{k}_1 h + 2\mathbf{k}_2 h, t + h) \\ \mathbf{x}_{i+1} &= \mathbf{x}_i + \left(\frac{1}{6}\mathbf{k}_1 + \frac{4}{6}\mathbf{k}_2 + \frac{1}{6}\mathbf{k}_3\right) h. \end{aligned} \tag{12}$$

Finally, we consider perhaps the most common example of a higher order method, which is the 4th-order Runge-Kutta method (commonly known simply as "the Runge-Kutta method"). It uses four evaluations of the velocity field to calculate the next position, and again considers half-timestep offsets.

$$\begin{aligned} \mathbf{k}_1 &= \mathbf{v}(\mathbf{x}_i, t) \\ \mathbf{k}_2 &= \mathbf{v}(\mathbf{x}_i + \mathbf{k}_1 h/2, t + h/2) \\ \mathbf{k}_3 &= \mathbf{v}(\mathbf{x}_i + \mathbf{k}_2 h/2, t + h/2) \\ \mathbf{k}_4 &= \mathbf{v}(\mathbf{x}_i + \mathbf{k}_3 h, t + h) \\ \mathbf{x}_{i+1} &= \mathbf{x}_i + \left(\frac{1}{6}\mathbf{k}_1 + \frac{2}{6}\mathbf{k}_2 + \frac{2}{6}\mathbf{k}_3 + \frac{1}{6}\mathbf{k}_4\right) h. \end{aligned} \tag{13}$$

2.5 Interpolation

The ocean current velocity data we have chosen for this study are provided on regular grids of discrete points, (x_i, y_j, z_k), as well as discrete times t_i. For simplicity, we have chosen to consider motion in the xy plane only, *i.e.*, constant depth z. The available data then represents a 2D vector field of two-component vectors, on a regular quadratic grid.

In order to calculate the trajectory of a particle that moves in the velocity field defined by these data, we will have to evaluate the vector field at arbitrary locations, and possibly (depending on the choice of timestep) arbitrary times. In order to evaluate the velocity at a location (x, y), one possible option is to identify which cell that location is in, and use the vector defined at the center of that cell. This is sometimes known as nearest-neighbour interpolation. The advantages of this approach include simple implementation, and the fact that it makes a certain intuitive sense, as a vector defines the average velocity inside a cell, over an interval in time. The major disadvantage is that the vector field is then discontinuous at the boundaries of cells, which is not realistic. Furthermore, we have seen that the error bounds of the numerical integration methods depend on the existence and continuity of the partial derivatives of the velocity field. For nearest neighbour interpolation, the partial derivatives do not exist at cell boundaries.

In this study, we have chosen to consider four different interpolation schemes, using the same order of interpolation in both space and time:

- Zero-order: Nearest neighbour
- First-order: Linear interpolation
- Second-order: Quadratic spline interpolation
- Third-order: Cubic spline interpolation

For a description of how spline interpolation was implemented, see Section 3, and for further details of the different kinds of interpolation, see e.g. Press et al. (2007).

Once an interpolation scheme has been chosen, one has effectively replaced the gridded data by an analytical expression, by specifying a way in which to evaluate the velocity

field at any point $(\mathbf{x}, t)$. Hence, under this assumption, there exists a *true solution*, and as $h \to 0$, all numerical integration schemes should converge towards this solution.

Note that the purpose of interpolation is *not* to approximate the true velocity field at sub-grid scales, but rather to provide a consistent recipe for evaluating, at arbitrary points, the averaged velocity field represented by the data. Note also that interpolated velocity fields of different order are not identical, and will not produce identical trajectories.

3 Method

To allow easy testing of different combinations of interpolators and integrators, a simple particle transport code was written in Python, making use of the SciPy stack. The library `netCDF4` was used to read gridded current data, spatial interpolation was done by applying the class `RectBivariateSpline` from `scipy.interpolate`, and time interpolation was done with `interp1d`, also from `scipy.interpolate`. Both of these methods support linear interpolation, as well as quadratic and cubic splines. Nearest neighbour interpolation was handled as a special case, by finding the cell indices for the nearest grid point, and looking up the value of that cell directly. Using `numpy` arrays, both interpolation and the transport step can be done as operations carried out on arrays of particle positions, thereby avoiding much of the overhead of interpreted Python code.

For computational efficiency, spatial interpolation was done on a subset of the velocity field (separately for the x and y components of the vectors), where the subset was selected to cover all particles, plus a halo of 5 cells in both directions. Note that in order to determine the weights of a higher order spline, more than two data points are needed. Hence, to evaluate the velocity at position $\mathbf{x}$ and time t, first spatial interpolation was used to obtain $\mathbf{v}(\mathbf{x}, T_n)$, for several consecutive timesteps T_n of the available data. Then, interpolation in time was used to obtain the value $\mathbf{v}(\mathbf{x}, t)$. For the results reported here, 2 points in time were used for linear interpolation, 6 points for quadratic splines, and 8 points for cubic splines, selected in such a way that t was found in the middle interval.

The datasets used in this study are provided as NetCDF files with vectors defined in the cell centers (de-staggered grids). The dimensions of the datasets are x, y, z and t, with the xy plane defined in a polar stereographic projection. The current velocity field is provided as vectors on the xy basis. In our simulations, we used the same xy coordinate system as the data files, converting to longitude and latitude only at the end, for presentation purposes. All errors are calculated from distances in the xy plane, measured in meters.

4 Results - Case 1: Double Gyre

We begin by studying a simplified test system, where the velocity field is known analytically as a function of $\mathbf{x}$ and t. The chosen system is called a double gyre, and consists of two counter-rotating vortices, where the line separating the vortices will oscillate left and right as time passes. The velocity field represents the flow of an incompressible fluid, and satisfies the conservation of mass ($\nabla \cdot \mathbf{v} = 0$), but is otherwise not meant to be an actual solution of the Navier-Stokes equations. The equations for the double gyre field are taken from Shadden et al. (2005).

The velocity field is defined in the xy-plane, in the region $x \in [0, 2]$, $y \in [0, 1]$, and is given by:

$$\begin{aligned} v_x &= -\pi A \sin(\pi f(x,t)) \cos(\pi y), \\ v_y &= \pi A \cos(\pi f(x,t)) \sin(\pi y) \frac{\partial f(x,t)}{\partial x}, \end{aligned} \tag{14a}$$

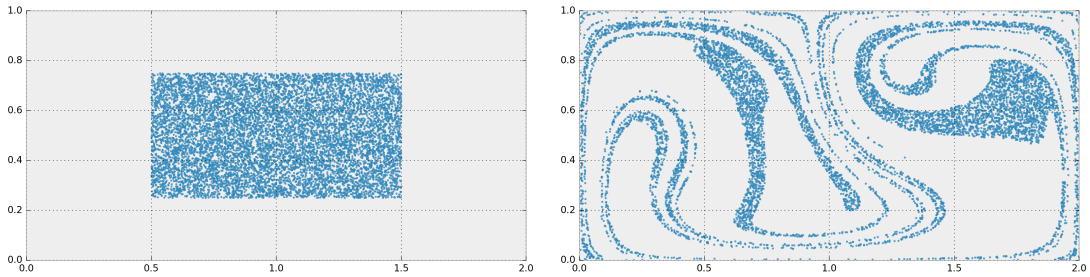

Figure 1 Distribution of particles in the double gyre velocity field (see Eq. (14)), shown at $t = 0$ and $t = 10$ days.

where

$$\begin{aligned} f(x,t) &= a(t)x^2 + b(t)x, \\ a(t) &= \epsilon \sin(\omega t), \\ b(t) &= 1 - 2\epsilon \sin(\omega t), \end{aligned} \tag{14b}$$

where the parameters A, ϵ and ω are chosen to adjust the properties of the system. Here, the values $A = 0.1$, $\epsilon = 0.25$ and $\omega = 1$ were used. In applying this field as a test case, we have scaled the time variable and the velocity field by a factor $\frac{1}{12 \cdot 3600}$, in order to obtain a more meaningful comparison of timesteps with the ocean current case. Figure 1 shows the initial distribution of particles used in this study, and their positions after 10 days.

4.1 Error Analysis

In Section 2.3, we introduced the concept of the global error of a trajectory, which is the distance from the end position of the numerically calculated trajectory, to the exact solution. In the case of the double gyre, we do not know the exact solution, but since the velocity field can be evaluated at arbitrary positions and times, we can calculate the trajectories at any accuracy desired, limited only by the precision of the floating point calculations. Hence, we expect the numerically calculated trajectory to converge to the exact answer when the timestep becomes sufficiently short. The purpose of introducing the analytically known double gyre field is thus to establish a baseline for the scaling of the numerical error with timestep. Later, we will compare the results obtained for the double gyre, to those obtained from the gridded current velocity fields.

To estimate the error for a given integrator, and at a timestep, h, we first calculate a reference solution using the 4th-order Runge-Kutta integrator and a very short timestep, h_0, found by trial and error. Then, for a given timestep h, we estimate the global error in the trajectory of particle i, $E_i(h)$, as the distance from the end point of the trajectory, to the end point of the reference trajectory calculated with the short timestep, h_0:

$$E_i(h) = \left| \mathbf{x}_i^{h_0}(t_N) - \mathbf{x}_i^{h}(t_N) \right| \tag{15}$$

where t_N is the time at the end of trajectory.

Using $N_p = 10000$ particles, with initial positions distributed as shown in Fig. 1 (left panel), we have calculated the trajectories using 42 different timesteps, spanning from 30 seconds to 28800 seconds (8 hours). Using trajectories calculated with $h_0 = 10$ s and the 4th-order Runge-Kutta integrator as a reference, we can then calculate the global error for each particle, for each different timestep. In Fig. 2, we show, on log-log scale, the average error

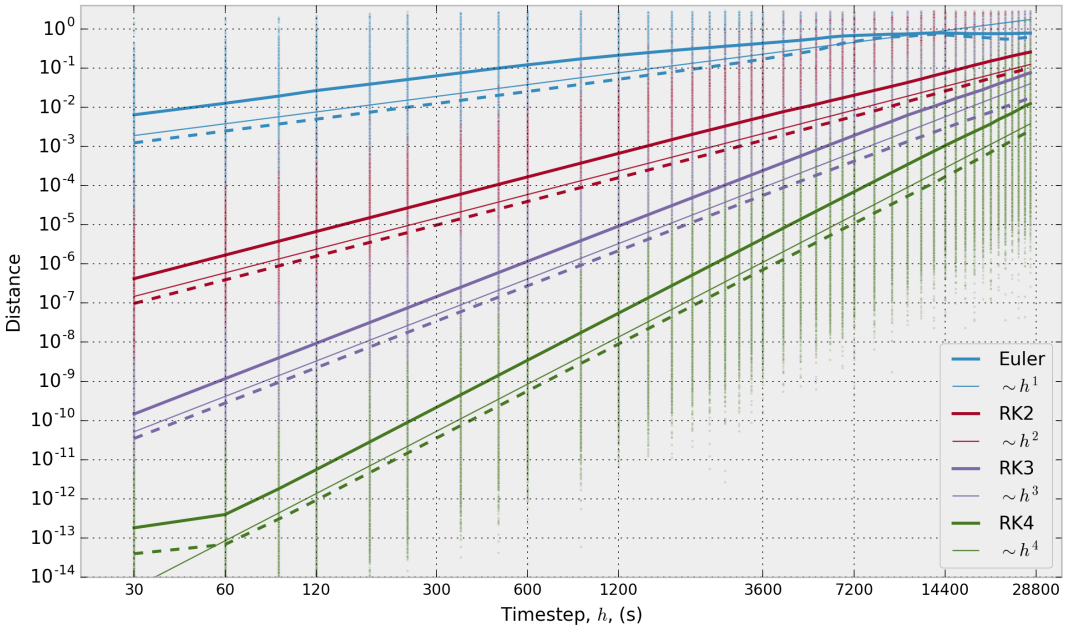

Figure 2 Scaling of error with timestep for the double gyre system, shown for Runge-Kutta methods of order 1 to 4. The continuous lines show the average global error, taken over 10000 trajectories, and the dashed line show the median global error. The thin lines are added to indicate the slope.

(continuous line) and the median error (dashed line) for Runge-Kutta methods of orders 1 to 4. The individual dots show the error for each particle, and are included to give a sense of the distribution. For each integrator, a thin, continuous line has been added to indicate the slope. Note that the smallest obtainable local error using double precision floating point numbers is in practice around 10^{-15}, and the largest error is limited by the size of the domain, which is $\sqrt{5}$ length units from corner to opposite corner.

The main purpose of presenting the results in Fig. 2 is, as mentioned above, to establish a baseline for comparison of the behaviour of the different integrators. Since the function that describes the double gyre field is infinitely differentiable, we know from Eq. (8) that the local error is bounded by $e(h) \leq Ch^{p+1}$, where p is the order of the numerical integration scheme, and consequently that the global error is $\sim h^p$. This is indeed what can be seen for each of the integrators presented in Fig. 2. The flattening of the curve for the Forward Euler integrator is caused by the fact that as the timestep becomes longer, more and more trajectories have an error comparable to the size of the system, after which it cannot grow further.

5 Results - Case 2: Gridded Current Data

We have selected to use three different datasets, with horizontal resolutions of 800 m, 4 km and 20 km (Fig. 3). These are all publicly available, provided by the Norwegian Meteorological Institute*, and are known as NorKyst800m, Nordic4km, and Arctic20km respectively. In all three cases, the ROMS ocean model has been used as the simulation engine, and the 4 km model has been forced with the 20 km model as boundary conditions, and similarly the 800 m model has been forced with the 4 km model. All three datasets are provided at 1 hour temporal resolution.

For each of the three datasets, we have considered $N_p = 10000$ particles, in each

*see `http://thredds.met.no/thredds/fou-hi/fou-hi.html`

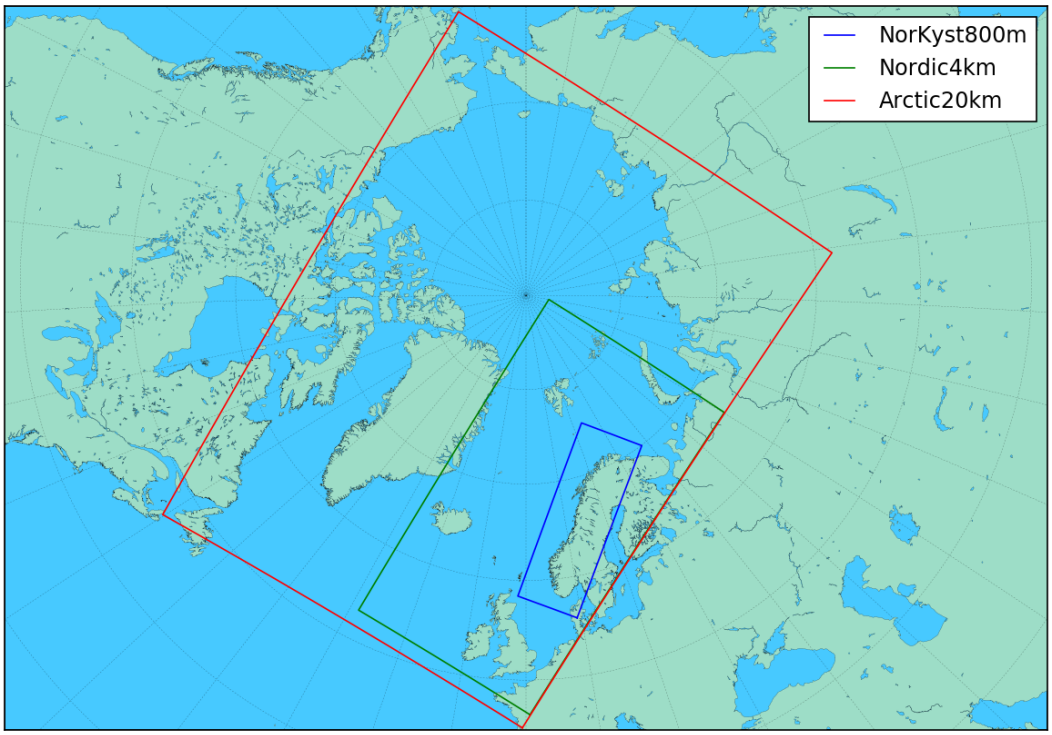

Figure 3 Geographical extent of the three different ocean current datasets used in this work.

case starting out from the same initial conditions, at the same time t_0 = February 1, 2017, 12:00 GMT, and being transported for a period of 10 days. The initial positions where selected at random within a 40 km $\times$ 100 km rectangle, and the same initial positions were used for all simulations. The initial positions, and the particle field after transport for each of the three datasets, are shown in Fig. 4. Note that while the center of mass of the particle cloud appears in approximately the same location of all three datasets, the particles moved with the higher resolution datasets display more mixing due to eddies which are unresolved by the coarser data.

The area chosen for the initial distribution of particles is outside the coast of Norway in the northernmost part of the North Sea. The area is outside the influence of the northward flowing Norwegian Coastal Current. The Norwegian Current, branching from the North Atlantic Current, carries Atlantic water towards Norway and northward along the coast. The main inflow of Atlantic water is to the north of the initial distribution area, and a branch of the inflow turns southwards, flowing outside of the Coastal Current (see e.g. Sætre, R. (2005).)

Due to the choice of initial area, particles tend to have a slightly northward net movement initially, while those particles closest to the Norwegian coast are eventually carried southward by the branch of the Atlantic inflow. The eddy activity between the main currents leads to increased dispersal of particles, and some may eventually start moving northwards along the Coastal Current.

5.1 Error Analysis

A fundamental assumption in numerical integration of ODEs must be that the result will converge when the timestep, h, is reduced. In a case such as ours, where the derivative is only known with limited resolution, ignoring sub-grid scale fluctuations, the numerical solution will not converge to the "correct" answer, in the sense of "the true trajectory of a passive

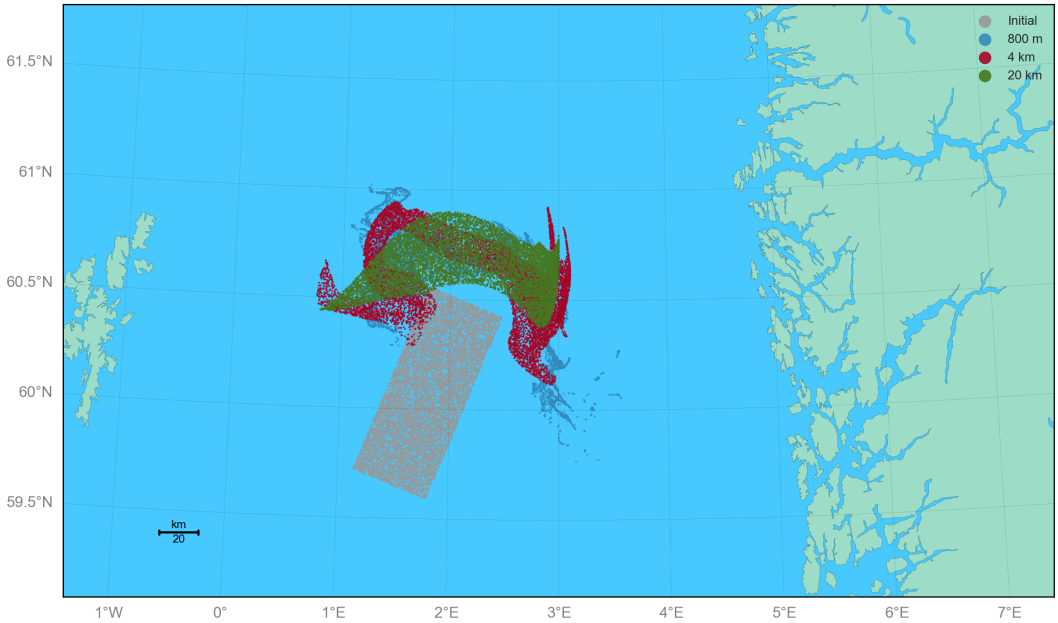

Figure 4 10000 particles, starting from the same initial positions (marked by grey dots) at t_0 = February 1, 2017, 12:00 GMT, transported for 10 days with current data of three different horizontal resolutions.

tracer released into the ocean at the given time and location". However, for the numerical approach to make sense, we must at least expect that the solution will settle towards some definite value when the timestep is reduced. At some point, no further useful information can be extracted from the velocity field by reducing the timestep.

The situation is illustrated in Fig. 5. It shows five trajectories, calculated with the same initial position, using the 800 m dataset, and timesteps of 900, 1800, 3600, 7200, and 14400 s (the timestep of the current data is 3600 s). The 4th-order Runge-Kutta integrator, combined with quadratic spline interpolation, was used to calculate the trajectories. The trajectories for h = 900 s and h = 1800 s are practically identical, while the trajectory for h = 3600 s can be seen to deviate somewhat from the two others. For h = 7200 s and h = 14400 s, on the other hand, the difference is much greater, and the endpoints of the trajectories (after being transported for 10 days) are separated by several kilometers.

To investigate the scaling of the global error with timestep, for different combinations of interpolation and integration methods, we will conduct the same type of analysis as presented for the analytically known double gyre field. For each of the $N_p = 10000$ initial positions, and for each degree of interpolation, we calculate a reference trajectory, using the 4th-order Runge-Kutta integrator, and a very short timestep, $h_0 = 30$ s. The value of h_0 was chosen partially out of practical considerations, as the simulation time for the higher order methods was approaching 12 hours for a single ensemble of 10000 particles at $h_0 = 30$ s. However, the value of h_0 also fulfills three other criteria: It is much, much shorter than the temporal resolution of the data, the typical steplength at h_0 (~ 15 m) is much, much shorter than the horizontal resolution of the datasets, and it evenly divides the temporal resolution of the data, meaning all intervals in the data files are weighted equally.

For all longer timesteps, we then proceed to estimate the global error for each particle trajectory to be the distance from the end of the trajectory, to the end of the reference trajectory (Eq. (15)). Note that all trajectories are compared to a reference trajectory calculated with the same level of interpolation. The reason for this is that a gridded dataset, interpolated with,

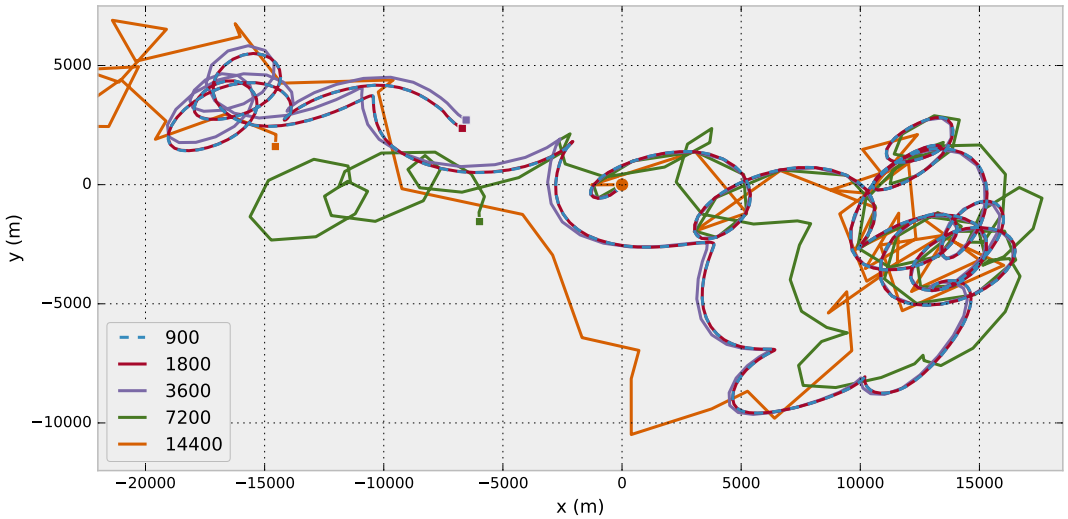

Figure 5 Trajectories for the same initial position, calculated with 5 different timesteps. The trajectories for h = 900 s and h = 1800 s are essentially identical at the scale shown here (the dashed blue line lies on top of the continuous red line).

e.g., linear interpolation, is not identical to the same dataset interpolated with cubic splines, and trajectories calculated at different levels of interpolation should not in general be expected to be identical, even as $h \to 0$. This is illustrated in Fig. 6, where trajectories from two initial positions have been calculated with $h = 2$ s, and the 4th-order Runge-Kutta integrator, using interpolated current fields of different degrees. Both the initial conditions and the duration (3 days) of these trajectories were chosen to illustrate the fact that different interpolation can give different trajectories. The simulation was repeated using timesteps of 5 s, 10 s, and 30 s, producing visually identical results (the changes in the end points of the trajectories by going from $h = 5$ s to $h = 2$ s were in all cases less than 0.2 m).

In Fig. 7, the global error, as a function of timestep, for each particle $E_i(h)$ is shown as individual points, with the error calculated according to Eq. (15). The average error is shown as a continuous line, and the median error is shown as a dashed line. The trajectories have been calculated with the Forward Euler integrator, using the 800 m dataset, and four different interpolation schemes: Nearest neighbour (upper left), linear (upper right), quadratic splines (lower left), and cubic splines (lower right). The graphs are plotted on a log-log scale, and a thin red line has been added to indicate the slope of the scaling of the error with timestep.

In Figures 8, 9, and 10, the same type of results are shown, for the 2nd-, 3rd-, and 4th-order Runge-Kutta methods, again for the 800 m dataset, and for the four different interpolation schemes. For each combination of interpolator and integrator, the trajectories calculated with the same interpolator and the 4th-order Runge-Kutta integrator, at timestep $h_0 = 30$ s, has been used as a reference when calculating the global error.

In Figures 11 - 14, the same type of results are shown, but for the 4 km dataset, and in Figs 15 - 18, the same type of results are again shown, now for the 20 km dataset.

Finally, we have produced tables of the average and median global error for selected timesteps, for each combination of dataset, interpolator and integrator. These were calculated in the same way as the results shown in Figs. 7 to 18, using the results obtained with the 4th-order Runge-Kutta integrator, at a timestep $h_0 = 30$ s, as the reference. In Table 1, we show the average, for timesteps 1800 s, 3600 s and 7200 s, and in Table 2, the median global error is shown for the same timesteps. Note that the temporal resolution of all three datasets is 3600 s.

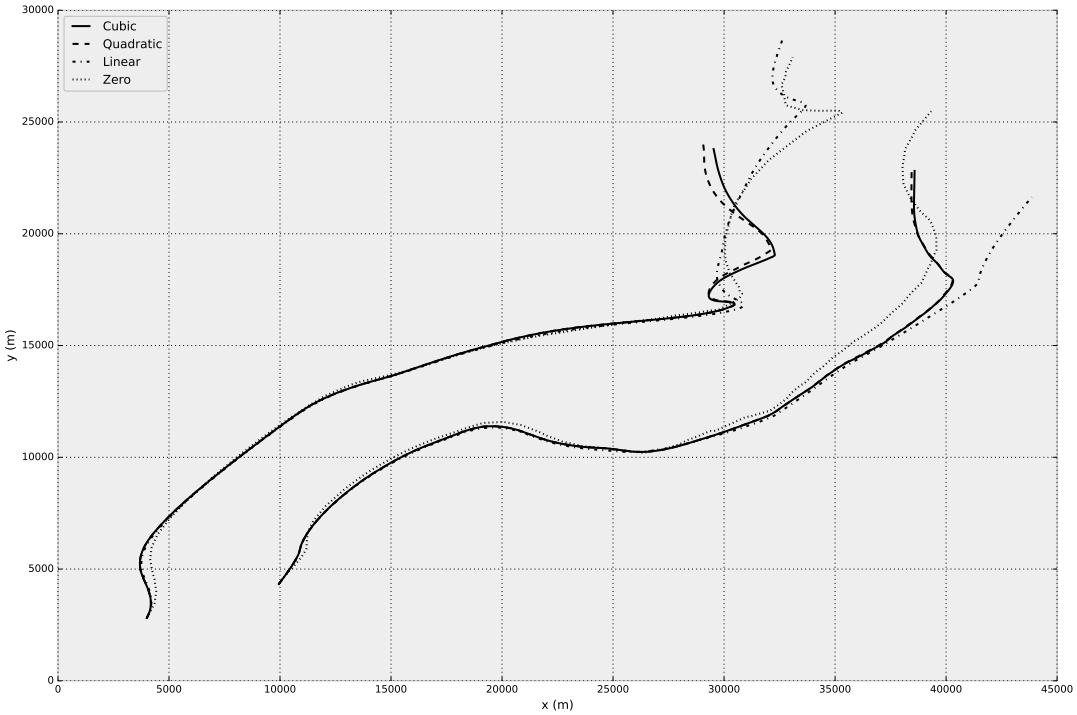

Figure 6 Three-day trajectories for two initial positions, calculated with different levels of interpolation, using $h = 2$ s, and the 4th-order Runge-Kutta integrator.

Table 1 Average global error (in meters) for combinations of interpolator and integrator for timesteps 1800 s, 3600 s, and 7200 s.

		800 m				4 km				20 km			
		Euler	RK2	RK3	RK4	Euler	RK2	RK3	RK4	Euler	RK2	RK3	RK4
1800	Zero	784	829	344	340	382	352	138	137	229	265	102	97
	Linear	821	62	7.61	5.81	326	7.90	0.28	0.20	387	2.63	0.10	0.05
	Quad	887	58	4.09	2.76	329	7.33	0.19	0.01	399	3.43	0.08	0.01
	Cubic	878	45	4.79	2.33	329	8.09	0.19	0.01	401	3.77	0.08	0.00
3600	Zero	1497	1437	618	614	742	688	270	268	456	524	203	192
	Linear	1633	208	44	26	652	32	1.82	0.79	775	11	0.67	0.19
	Quad	1706	225	42	33	654	29	1.46	0.16	794	15	0.65	0.03
	Cubic	1709	222	45	35	654	32	1.47	0.12	798	16	0.61	0.02
7200	Zero	3270	2805	1745	1719	4033	4233	1171	1151	2746	4146	999	991
	Linear	3467	2440	868	858	3504	3680	1254	1253	1988	3504	1183	1182
	Quad	3533	2425	1019	988	3514	3679	1274	1272	1942	3508	1174	1173
	Cubic	3533	2443	1013	984	3510	3676	1277	1276	1939	3506	1178	1177

Table 2 Median global error (in meters) for combinations of interpolator and integrator for timesteps 1800 s, 3600 s, and 7200 s.

		800 m				4 km				20 km			
		Euler	RK2	RK3	RK4	Euler	RK2	RK3	RK4	Euler	RK2	RK3	RK4
1800	Zero	323	319	118	120	205	283	96	97	142	241	85	80
	Linear	384	15	1.83	1.52	270	5.37	0.20	0.12	380	1.42	0.07	0.04
	Quad	400	20	0.94	0.38	273	6.00	0.14	0.01	390	3.50	0.06	0.00
	Cubic	399	19	0.84	0.14	274	6.57	0.14	0.00	391	3.97	0.06	0.00
3600	Zero	635	604	233	230	420	566	197	197	287	482	170	158
	Linear	776	66	11	6.62	541	21	1.33	0.48	760	5.39	0.48	0.14
	Quad	787	88	11	6.42	549	24	1.05	0.11	775	15	0.50	0.03
	Cubic	790	83	11	6.19	550	27	1.07	0.08	781	17	0.46	0.02
7200	Zero	1929	1640	822	815	3165	3505	779	767	2262	3724	677	663
	Linear	2024	1511	509	513	2595	2884	963	967	1844	3336	1109	1107
	Quad	2048	1479	563	563	2556	2869	979	983	1800	3313	1100	1098
	Cubic	2036	1480	560	560	2558	2867	985	986	1792	3318	1104	1101

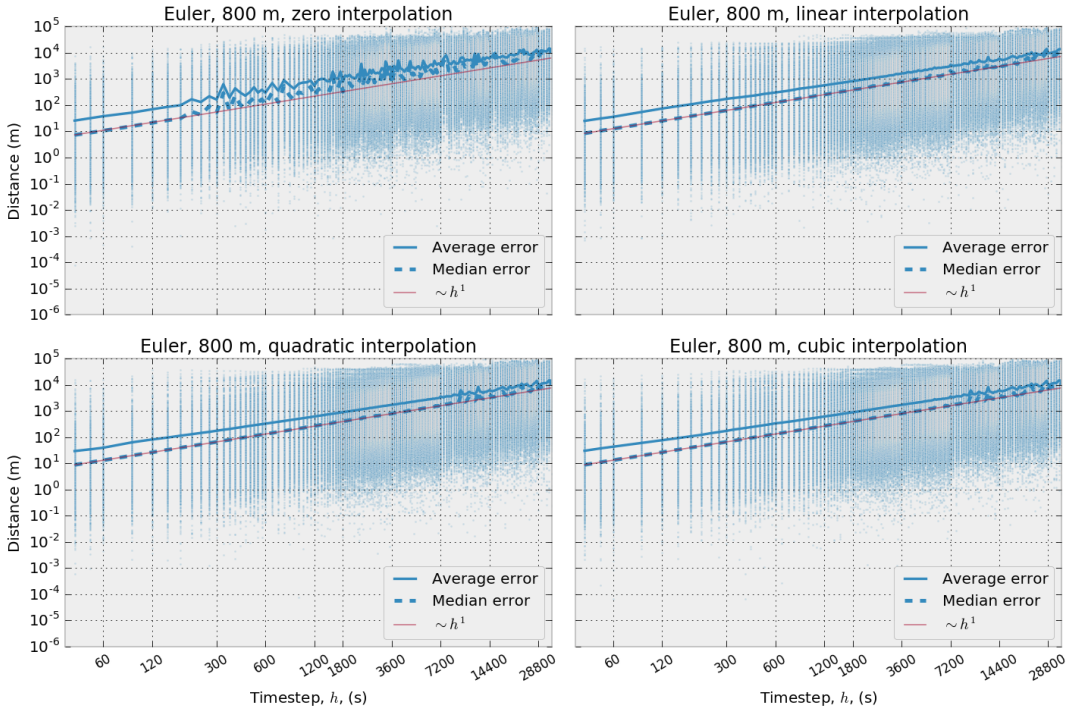

Figure 7 Global error for 10000 particles (see Eq. (15)), after 10 days of transport, for the Forward Euler integrator, using the 800 m dataset, and four different interpolation schemes: Upper left: Nearest neighbour, upper right: linear, lower left: quadratic splines, lower right: cubic splines. The red lines are included to indicate slope.

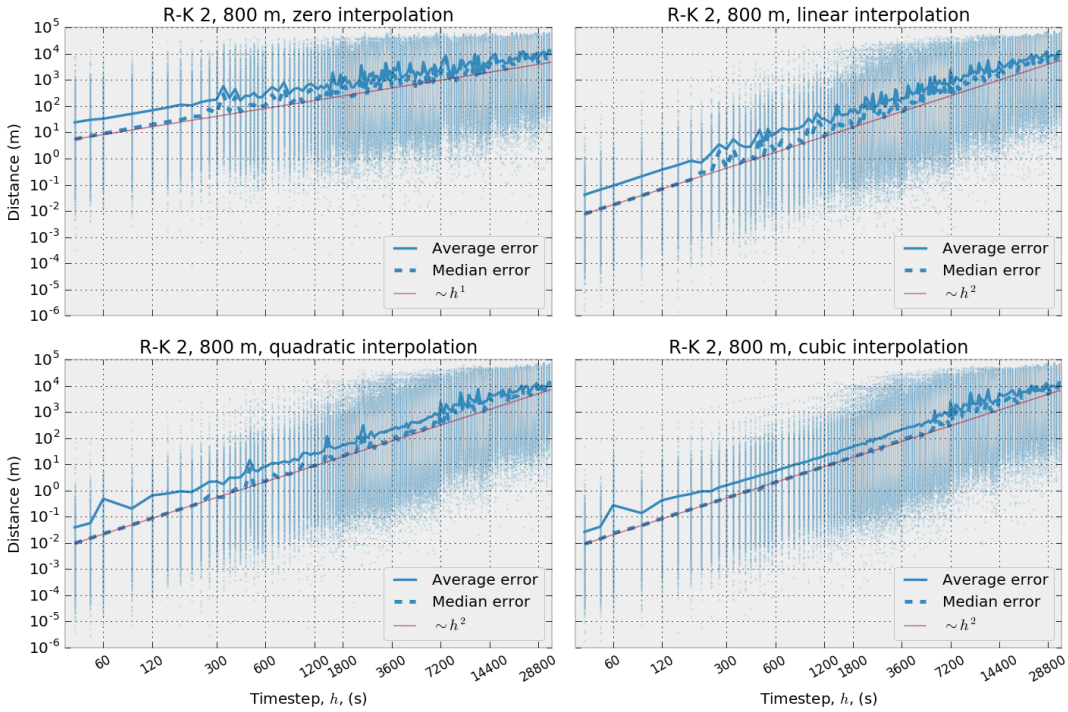

Figure 8 The same as for Fig. 7, but for the 2nd-order Runge-Kutta integrator.

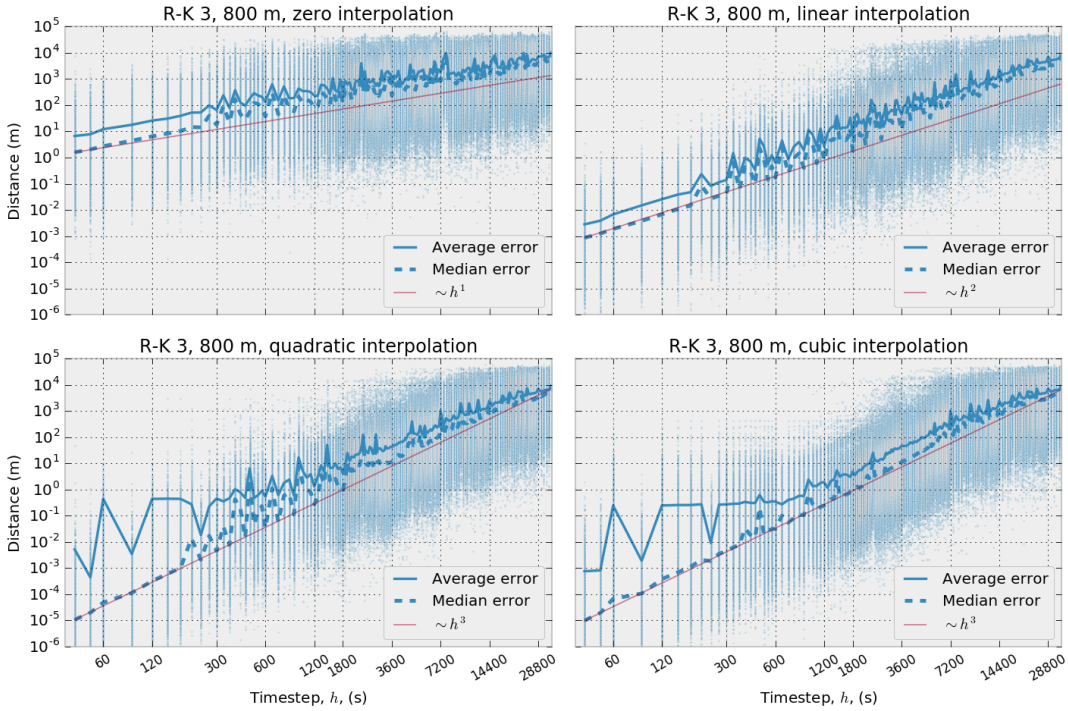

Figure 9 The same as for Fig. 7, but for the 3rd-order Runge-Kutta integrator.

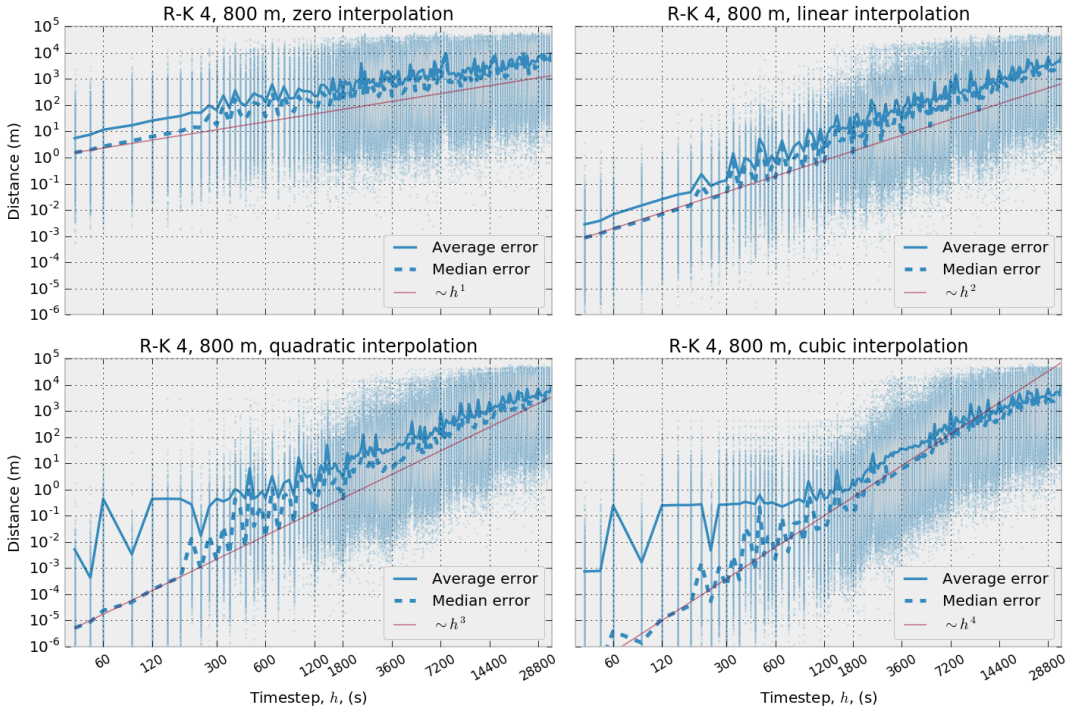

Figure 10 The same as for Fig. 7, but for the 4th-order Runge-Kutta integrator.

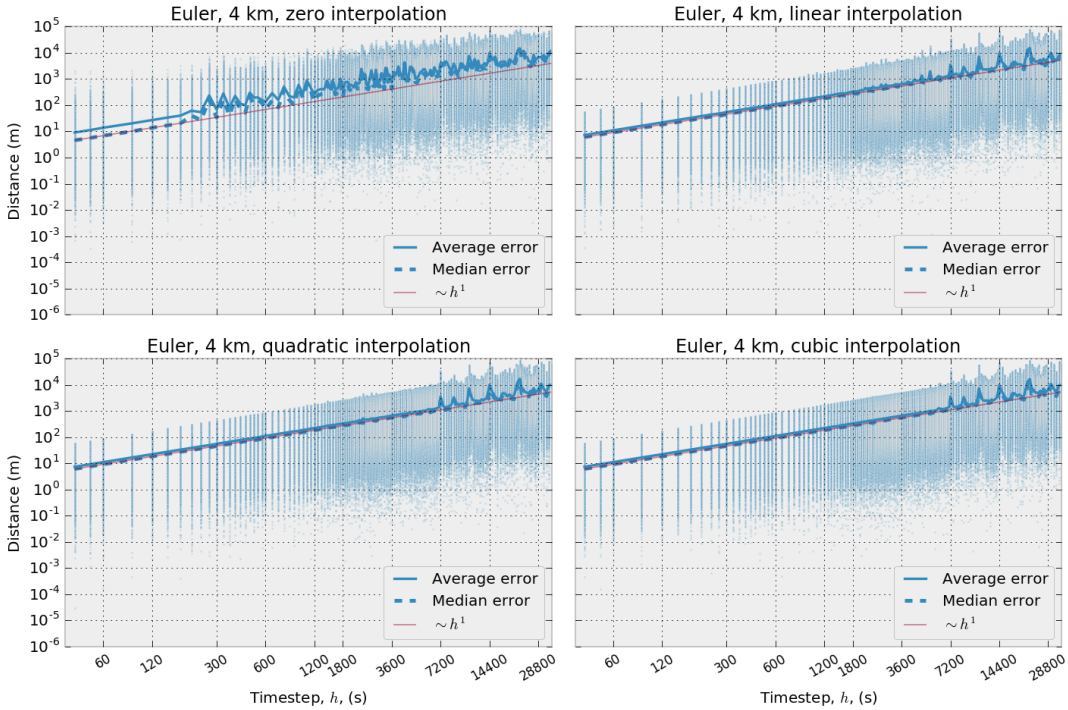

Figure 11 Global error for 10000 particles (Eq. (15)), after 10 days of transport, for the Forward Euler integrator, using the 4 km dataset, and four different interpolation schemes: Upper left: Nearest neighbour, upper right: linear, lower left: quadratic splines, lower right: cubic splines. The red lines are included to indicate slope.

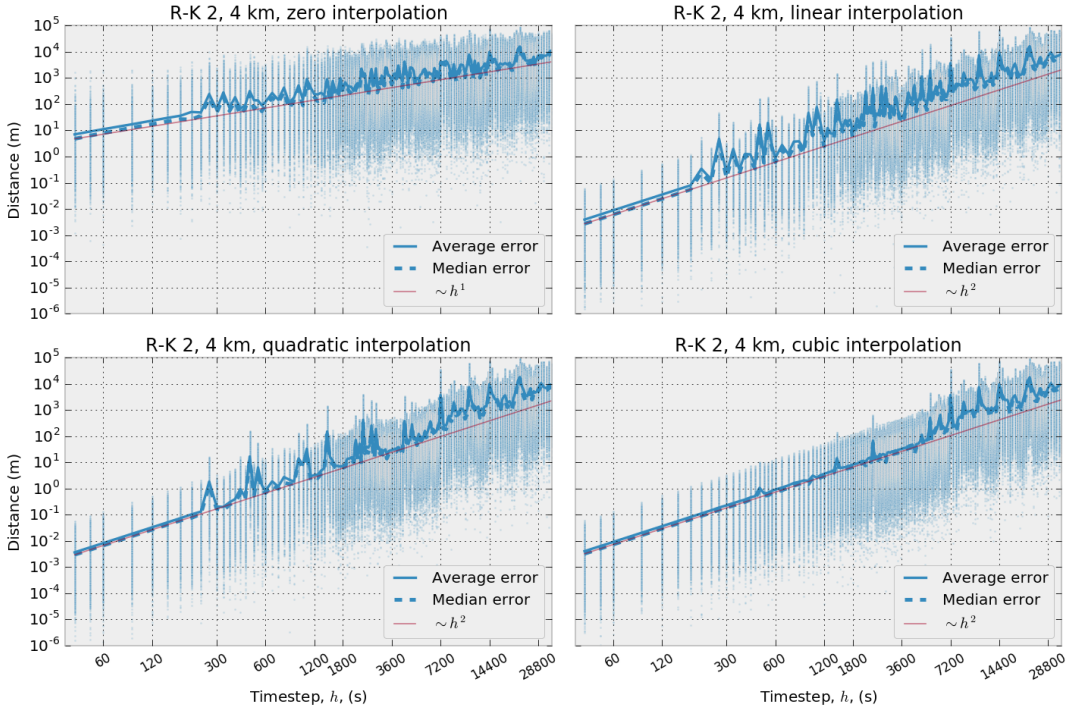

Figure 12 The same as for Fig. 11, but for the 2nd-order Runge-Kutta integrator.

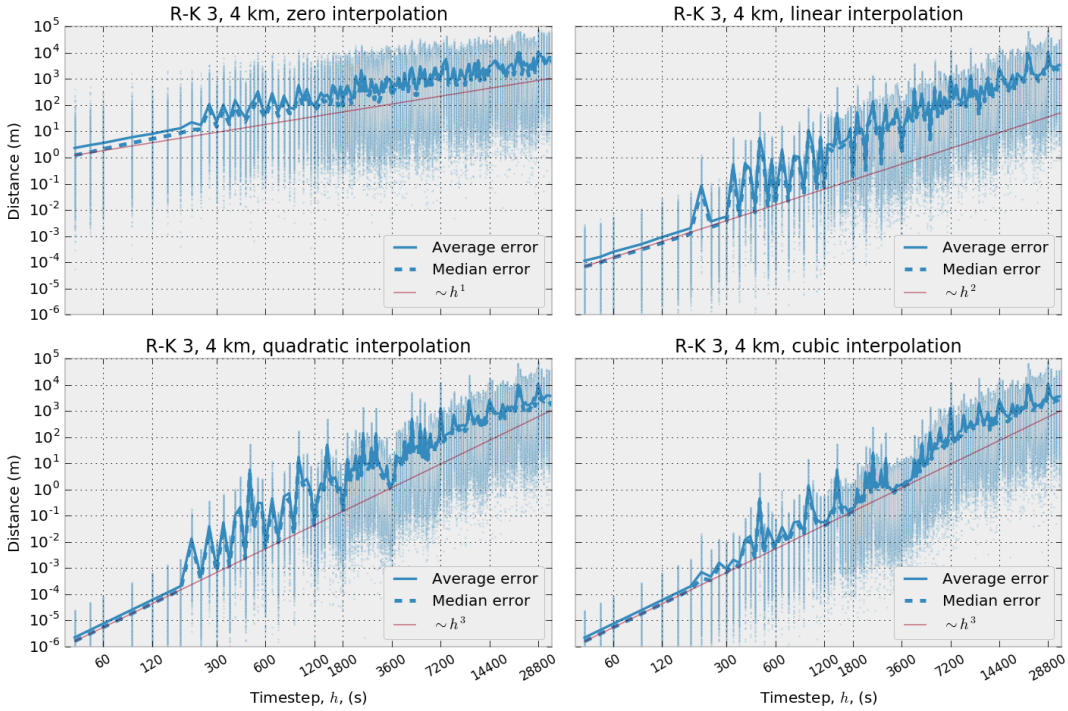

Figure 13 The same as for Fig. 11, but for the 3rd-order Runge-Kutta integrator.

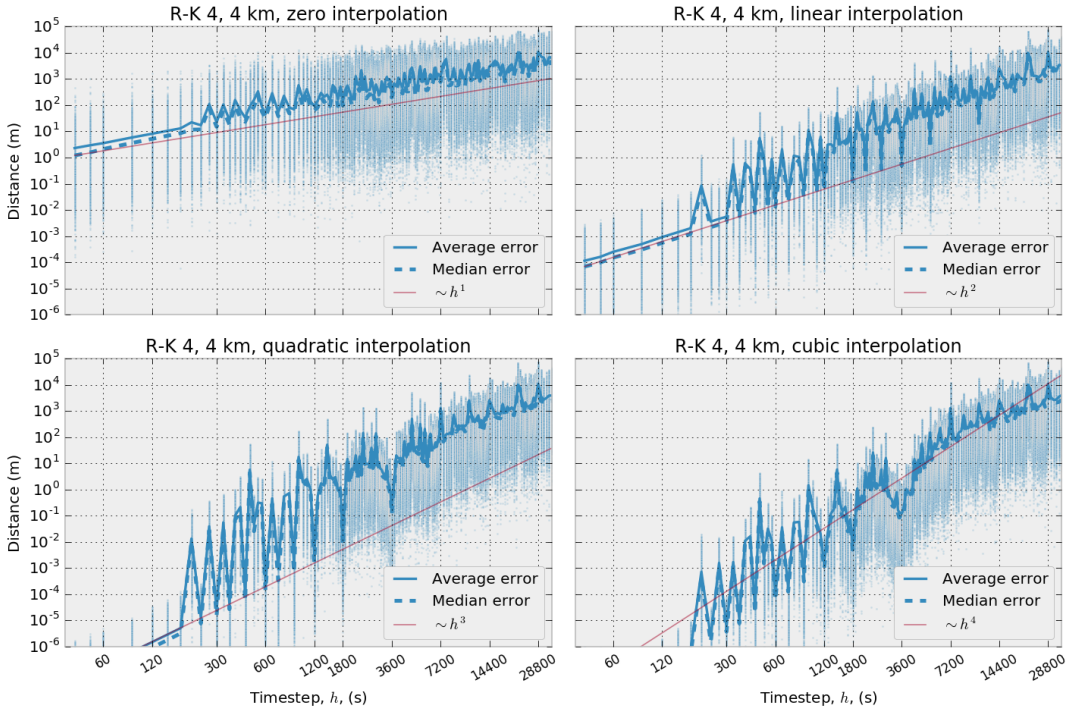

Figure 14 The same as for Fig. 11, but for the 4th-order Runge-Kutta integrator.

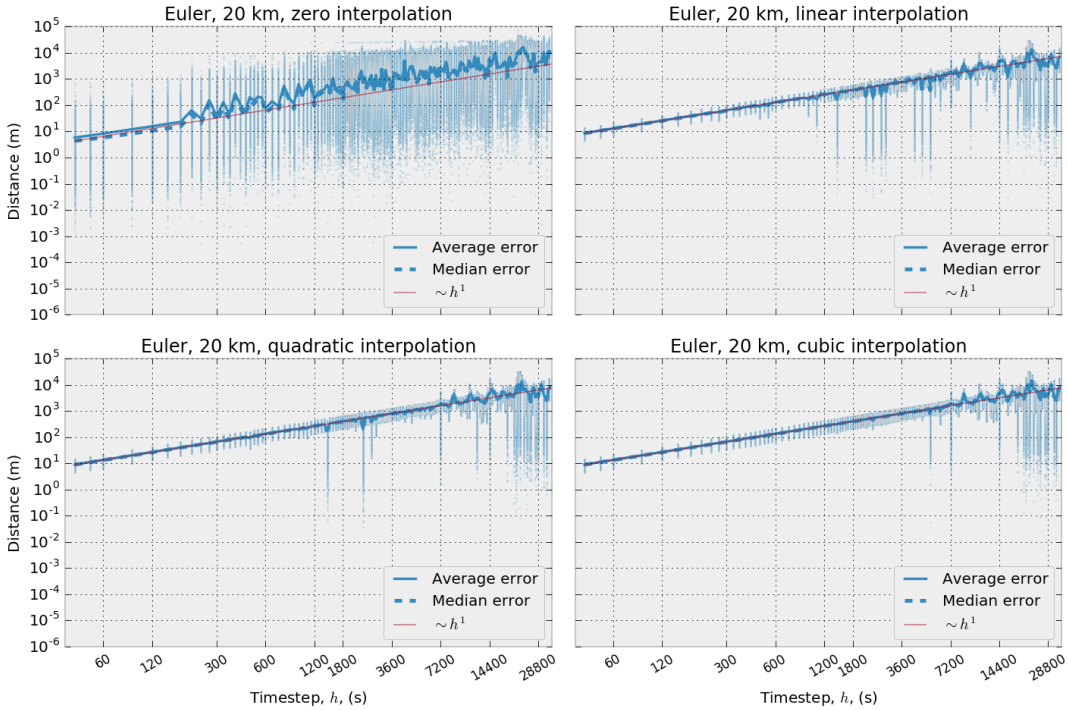

Figure 15 Global error for 10000 particles (Eq. (15)), after 10 days of transport, for the Forward Euler integrator, using the 20 km dataset, and four different interpolation schemes: Upper left: Nearest neighbour, upper right: linear, lower left: quadratic splines, lower right: cubic splines. The red lines are included to indicate slope.

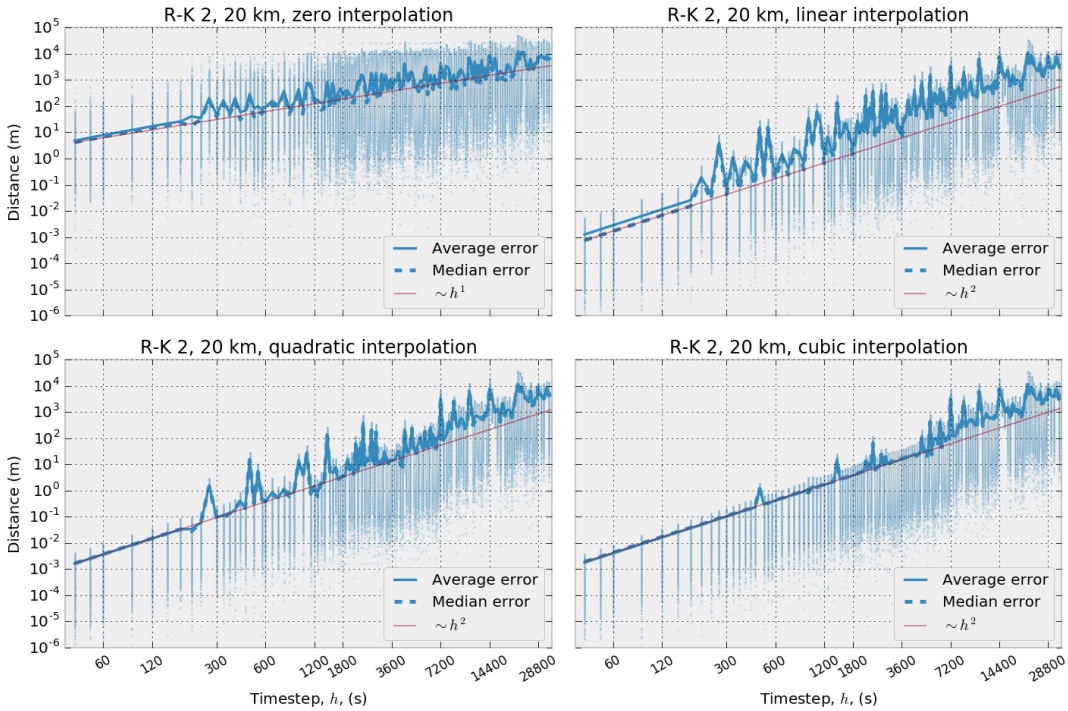

Figure 16 The same as for Fig. 15, but for the 2nd-order Runge-Kutta integrator.

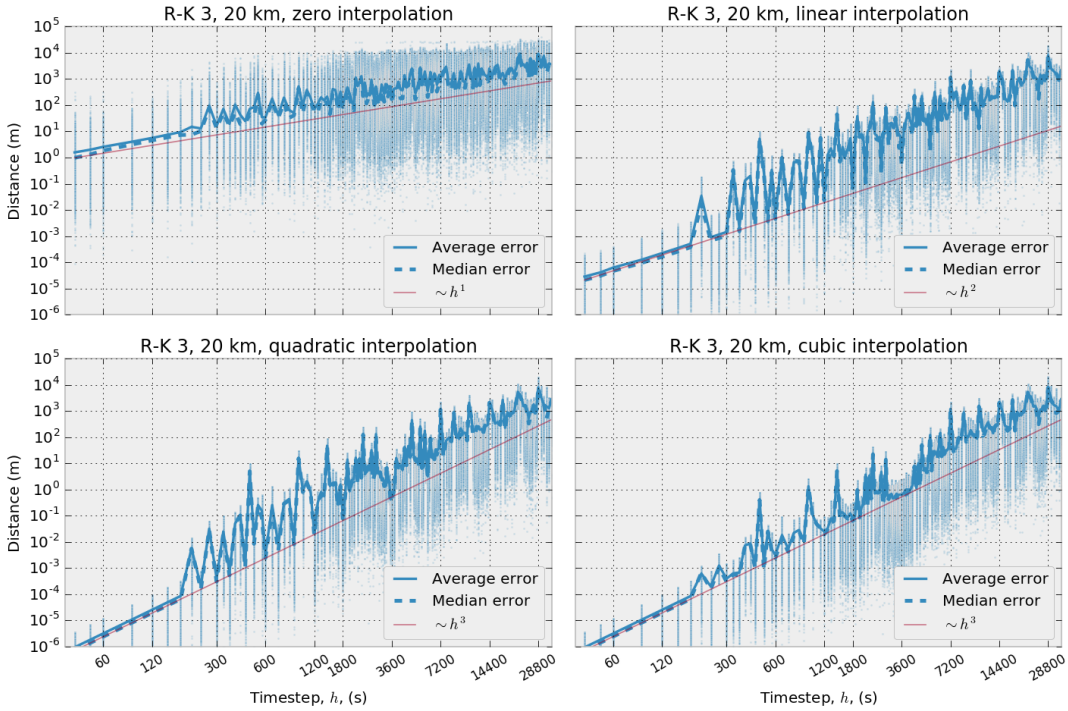

Figure 17 The same as for Fig. 15, but for the 3rd-order Runge-Kutta integrator.

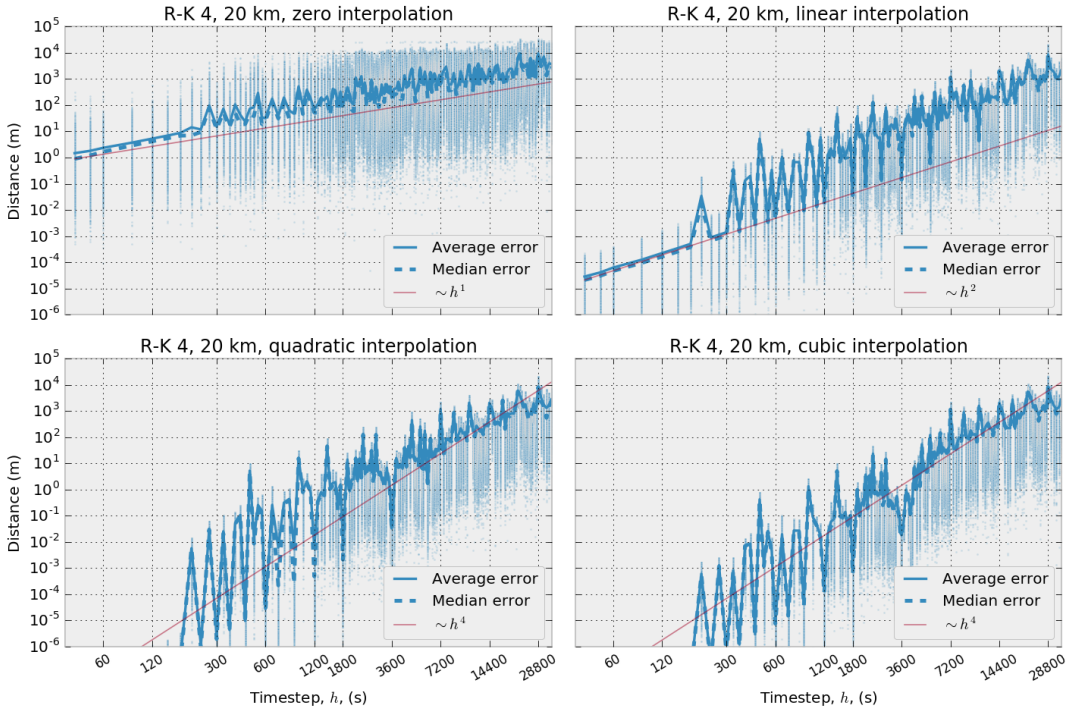

Figure 18 The same as for Fig. 15, but for the 4th-order Runge-Kutta integrator.

6 Discussion

From the error analysis of the double gyre system, shown in Section 4.1, one might expect the different integrators to display the same type of behaviour for the ocean current data. However, we see from the results of Section 5.1 that this is not the case. In this section, we will discuss some of the observations one can make from the results presented above, as well as the way these results apply to practical situations.

6.1 Error Analysis - 800 m Dataset

Considering first the results for the Forward Euler integrator, shown in Fig. 7, we see that with nearest neighbour interpolation, the error does indeed scale approximately linearly, but with considerable variation between timesteps. Once interpolation of first order or higher is applied, however, the scaling becomes very smooth, and matches the line of slope h almost perfectly.

Moving on to the 2nd-order Runge-Kutta method, we see that with nearest neighbour interpolation, the error scales linearly, even for a second order method. With linear interpolation, the error scales as h^2, as expected for a second-order method, although again with considerable variation among timesteps. When higher order interpolation is applied, the convergence becomes smoother, but still no faster than h^2.

Recall that for a cubic spline, the function itself, and its first and second derivatives exist, and are continuous (while the third derivative is not in general continuous). Hence, for cubic spline interpolation and a second order integrator, the criterion that the derivatives exist and be continuous up to the order of the integrator is fulfilled. This explains why the error plot for this case shows smoother and more idealised scaling, similar to the results for the double gyre case (see Fig. 2).

For the 3rd-order Runge-Kutta method, the results are shown in Fig. 9. With no interpolation, the error scales as h, with linear interpolation, the error scales as h^2. With quadratic spline interpolation, the convergence of the median error approaches h^3, however the average error seems to flatten out at a much higher level as $h \to 0$, indicating the presence of large outliers inflating the mean. From the individual points shown in Fig 9, which shows the global error for each trajectory, it can be confirmed that the distribution in errors becomes quite broad at the lower timesteps.

Finally, for the 4th-order Runge-Kutta method, the picture is quite similar to that of the 3rd-order method. The median error scales as h, h^2, h^3, and finally h^4, for increasingly high orders of interpolation, and again, as $h \to 0$ the average error flattens out at a larger values than the median error for the two highest orders of interpolation.

Note that for the 3rd- and 4th-order methods, we begin to see the error reaching a threshold at very large timesteps ($h > 7200$ s). This is caused by the fact that the largest errors are limited by the typical speed of the ocean currents, placing limitations on how far it is possible for two trajectories to separate. We can see from the individual points in Figs. 9 and 10 that the largest errors are on the order of 50 km. For comparison, the average traveled distance for a particle in these simulations is about 70 km (from start to end, not along the trajectory). The same limitations are of course found also for the first and second order integrators, but the effects are less visible in Figs. 7 and 8, due to the overall smaller slope of the error graphs in those cases.

6.2 Error Analysis - 4 km and 20 km Datasets

For the two coarser resolution datasets, the overall picture is much the same as for the 800 m data. The convergence of the integrator is limited by the order of the interpolation

scheme used. For example, in order to achieve a global error that scales with h^3 for the 3rd-order Runge-Kutta integrator, it is necessary to apply quadratic spline interpolation, and even then, the error varies quite strongly between timesteps.

The most noticeable differences from the 800 m case are first that for the 4 km case, and even more so for the 20 km case, the average and the median error are much more similar, indicating a narrower distribution of errors. Secondly, there is a very large difference in the error between those timesteps that evenly divide 3600 s, such as $h = 300$ s and $h = 360$ s, compared to those that do not, such as $h = 270$ s and $h = 330$ s. This is especially noticeable in the case of the 4th-order integrator and quadratic spline interpolation, where the difference is more than 3 orders of magnitude for the 4 km dataset, and more than 4 orders of magnitude for the 20 km dataset.

6.3 Practical Considerations

From what we have seen so far, we can conclude that there is no benefit from applying a high-order integration scheme, without also applying at least linear interpolation. However, if linear interpolation is applied, then substantial reductions in error can be achieved by switching from first to second order integration. For example, from Table 1, we see that for the 800 m dataset, and a timestep of 1 hour, the average global error is reduced from 1497 m to 208 meters by switching from Forward Euler with nearest neighbour, to 2nd-order Runge-Kutta with linear interpolation. By switching to higher order integration methods, it is possible to reduce the error further, whereas higher order interpolation does not appear to have a significant effect. Note, however, that the error is very sensitive to the choice of timestep when the order of interpolation is lower than the order of integration. Choosing a timestep that does not evenly divide 1 hour can lead to 1 - 4 orders of magnitude increase in the average error (see in particular Figs. 10, 14, and 18).

From a practical perspective, it is also reasonable to ask what kind of accuracy should be expected. For example, using a timestep of $h = 1800$ s, cubic spline interpolation and the 4th-order Runge-Kutta integrator, with the 4 km dataset, one can achieve an average global error of 1 cm, after an average displacement of 70 km. However, there is absolutely no reason to expect the current data to be accurate to this level.

6.4 Choosing an Optimal Combination of Interpolation Method, Integration Method, and Timestep Based on Acceptable Error Thresholds

An ocean current model has inherent uncertainty in its calculation of the velocity field. Therefore, one must be be willing to accept some uncertainty in particle transport based on the ocean current model. Given that some error is accepted, the question becomes what combination of interpolation method, integration method, and timestep requires the least computational time while remaining within the acceptable error. To answer this, we focused on simulations run for timesteps of 900 s, 1800 s, 3600 s, and 7200 s. These timesteps were selected since they are likely candidates to be chosen in particle simulations for ocean current data with a timestep of 3600 s. Further, we had observed that very low errors could be obtained for all three model resolutions and all three interpolation methods (Figures 7-18). Therefore, we further focused the analysis on an optimal method in terms of walltime, considering the simulations with linear interpolation in space and time, using the 4 km dataset.

To quantify thresholds for acceptable error, we compared the errors in these simulations with the average displacement of particles from the beginning to the end of the simulation. Displacement was calculated as the average distance between start position and end position of all particles for the simulation with the shortest timestep. We set error thresholds at 5 %, 1 %, 0.1 % of the average displacement associated with the shortest timestep.

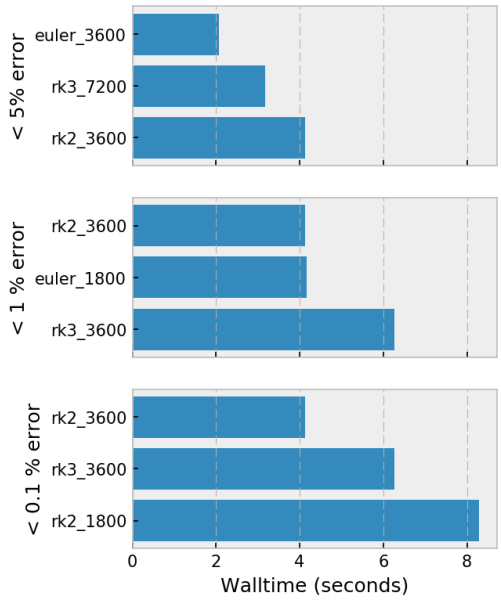

Figure 19 Top-three methods, ranked by walltime of calculation, for error bounds of 0.1%, 1% and 5%. Results shown for linear interpolation in space and time, and the 4 km dataset.

For each threshold, we selected the 3 combinations of methods and timestep that had the shortest simulation walltime while transporting particles within the error threshold (see Fig. 19). For all thresholds, we found that a timestep of 3600 s, corresponding to the timestep of the velocity field data, was the most cost-effective. When a 5 % error was acceptable the Euler integrator was the most effective method, while the 2nd order Runge-Kutta method was most effective when 1 % and 0.1 % error was acceptable.

In the simulations used to evaluate the optimal method, the walltime did not include constructing the interpolator. Since the simulations here were relatively short, the walltime required for constructing of the interpolator would effectively cancel the difference in computation time for each method. Thus, for a generic particle tracking simulation, which generally includes time-demanding computational steps beyond particle transport, the choice between the top 3 interpolators for each threshold in Figure 19 is relevant only as long as particle transport is a significant computational cost in the simulation. Finally, it should be considered that the integration and interpolation methods in this work were implemented in Python, while most particle transport models may be expected to be written in Fortran or C++, in which numerical efficiency may be different between the methods. At the same time, the code used in this work relies on the Scipy and Numpy libraries for array handling, integration, and interpolation, which means that most of the computational time also in this work is spent in complied Fortran or C code in the underlying libraries used by Scipy and Numpy.

7 Summary and Conclusion

In this study, we have performed an extensive comparison of timesteps and interpolation schemes, used with Runge-Kutta integrators of order 1 to 4. In general, higher order integration and interpolation methods decreased the error at a given timestep, but increased the required computational time in our particle transport simulations. Therefore, the choice of which method to use becomes a question of choosing a fast method that has an acceptable error.

We evaluated the run-time efficiency within different error thresholds for 4 commonly

used timesteps for a 3600 s ocean current field. For the case considered here, we found that the 2nd-order Runge-Kutta is the most effective integrator if an error of 1 % or 0.1 % is acceptable, while the Euler integrator was most effective if a larger error of 5% is acceptable (Figure 19).

Since the optimal methods all had the same timestep of the integration as the temporal resolution of the dataset, no interpolation in time would have been necessary at all for the methods that only evaluate the velocity field at integer multiples of h, leading to further potential savings in computational time. Note that spatial interpolation is always required, as velocities must be evaluated at arbitrary positions.

These results further show that the 4th-order Runge-Kutta integrator, which is perhaps the most commonly used integrator of the Runge-Kutta family, was not cost-effective even when 0.1 % displacement error was required (see Fig. 19), demonstrating that the most popular choice should not be taken for granted. If even higher accuracy is desired, the 4th-order Runge-Kutta method may be required, but from a practical point of view, there is no reason to expect the current data to be accurate to the level that this integrator can deliver.

8 Acknowledgements

The authors would like to thank their colleagues for many an interesting discussion in the SINTEF CoffeeLab. This work used the ARCHER UK National Supercomputing Service (`www.archer.ac.uk`), with computer time awarded through the "ARCHER Driving Test".

9 References

Barker, C. "A Statistical Outlook for the Deepwater Horizon Oil Spill" In *Monitoring and Modeling the Deepwater Horizon Oil Spill: A Record Breaking Enterprise*, pp 237–244. 2011.

Brönner, U., R. Nepstad, G. Eidnes, P. Rønningen, H. Rye, M. Alver, and D. Slagstad, "A Real-Time Discharge Modelling and Environmental Monitoring System for Drilling Operations" In *European HSE Conference and Exhibition, 16-18 April, London, United Kingdom*. Society of Petroleum Engineers, 2013.

Darmofal, D. and R. Haimes, "An Analysis of 3D Particle Path Integration Algorithms", *Journal of Computational Physics*, vol. 123, pp 182 – 195, 1996.

De Dominicis, M., N. Pinardi, G. Zodiatis, and R. Lardner, "MEDSLIK-II, a Lagrangian marine surface oil spill model for short-term forecasting – Part 1: Theory", *Geoscientific Model Development*, vol. 6, pp. 1851–1869, 2013.

Döös, K., B. Jönsson, and J. Kjellsson, "Evaluation of oceanic and atmospheric trajectory schemes in the TRACMASS trajectory model v6.0". *Geoscientific Model Development Discussions*, 2016:1–26, 2016.

French-McCay, D. P. "Oil spill impact modelling". *Environmental toxicology and chemistry*, 23(10), 2003.

García-Martínez, R. and H. Flores-Tovar, "Computer Modeling of Oil Spill Trajectories With a High Accuracy Method". *Spill Science & Technology Bulletin*, 5(5–6):323 – 330, 1999.

Hairer, E. and G. Wanner, *Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems*. Springer-Verlag Berlin Heidelberg, 1996.

Hairer, E., S. P. Nørsett, and G. Wanner, *Solving Ordinary Differential Equations I: Nonstiff Problems*. Springer-Verlag Berlin Heidelberg, 2nd edition edition, 1993.

Hairer, E., G. Wanner, and C. Lubich, *Geometric Numerical Integration: Structure-Preserving Algorithms for Ordinary Differential Equations*. Springer-Verlag Berlin Heidelberg, 2006.

Hodges, B., A. Orfila, J. Soyol, and X. Hou, Operational oil spill modeling: from science to engineering applications in the presence of uncertainty. In Ehrhardt, M., editor, *Mathematical Modelling and Numerical Simulation of Oil Pollution Problems.*, chapter 5, pages 99–126. Springer International Publishing, Switzerland, 2015.

Mancho, A. M., D. Small, and S. Wiggins, "A comparison of methods for interpolating chaotic flows from discrete velocity data". *computers & fluids*, 35:416 – 428, 2005.

Mariano, A., V. Kourafalou, A. Srinivasan, H. Kang, G. Halliwell, E. Ryan, and M. Roffer, "On the modeling of the 2010 Gulf of Mexico Oil Spill". *Dynamics of Atmospheres and Oceans*, 52(1–2):322 – 340, 2011.

North, E. W., E. E.Adams, Z. Schlag, C. R. Sherwood, R. He, K. H. Hyun, and S. A. Socolofsky, Simulating Oil Droplet Dispersal From the Deepwater Horizon Spill With a Lagrangian Approach. In *Monitoring and Modeling the Deepwater Horizon Oil Spill: A Record Breaking Enterprise*, pages 217–226. 2011.

Press, W. H., S. Teukolsky, W. Vetterling, and B. Flannery, *Numerical Recipes*. Cambridge University Press, 2007.

Price, J. M., W. R. Johnson, Z.-G. Ji, C. F. Marshall, and G. B. Rainey, "Sensitivity testing for improved efficiency of a statistical oil-spill risk analysis model". *Environmental Modelling & Software*, 19(7–8):671 – 679, 2004.

Reed, M. and B. Hetland, DREAM: a Dose-Related Exposure Assessment Model Technical Description of Physical-Chemical Fates Components. In *SPE International Conference on Health, Safety and Environment in Oil and Gas Exploration and Production, 20-22 March, Kuala Lumpur, Malaysia*. Society of Petroleum Engineers, mar 2002.

Reed, M., P. S. Daling, O. G. Brakstad, I. Singsaas, L. G. Faksness, B. Hetland, and N. Ekrol, OSCAR2000: a multi-component 3-dimensional Oil Spill Contingency and Response model. In *Proceedings of the 23rd AMOP Technical seminar*, pages 663–680, 2000.

Richardson, L. "The approximate solution of physical problems involving differential equations using finite differences, with an application to the stress in a masonry dam". *Phil. Trans. Roy. Soc. London, Ser. A*, 210:307, 1910.

Sætre, R. (ed.). *The Norwegian Coastal Current: Oceanography and Climate*. Tapir Akademisk Forlag, Trondheim, 2005.

Shadden, S. C., F. Lekien, and J. E. Marsden, "Definition and properties of Lagrangian coherent structures from finite-time Lyapunov exponents in two-dimensional aperiodic flows". *Physica D: Nonlinear Phenomena*, 212(3–4):271 – 304, 2005.

Visser, A. W. "Lagrangian modelling of plankton motion: From deceptively simple random walks to Fokker-Planck and back again". *Journal of Marine Systems*, 70(3-4):287–299, 2008.

Zelenke, B., C. O'Connor, C. Barker, CJ Beegle-Krause, and L. E. Eclipse, General NOAA Operational Modeling Environment (GNOME), Technical Documentation, NOAA Technical Memorandum NOS OR&R 40, 2012.