Universiteit Utrecht

Faculteit Bètawetenschappen

Opleiding Natuur- en Sterrenkunde

Pushing stuck particles used in Lagrangian ocean analysis, from land back to the ocean using PARCELS

Bachelor Thesis

Raymond Edwards

Supervisors:

Dr. Erik van Sebille

Institute for Marine and Atmospheric Research Utrecht

Dr. Philippe Delandmeter

Institute for Marine and Atmospheric Research Utrecht

June 2018

Abstract

One of the larger global problems is plastic in the ocean. Every year millions of tonnes of plastic waste enters the ocean and it it not clear what happens to the plastics and where it all ends up. A few models have been built to use Lagrangian ocean analysis to predict pathways of particles, like plastic, in the ocean. One example is PARCELS, which is a open source framework in Python. It uses numerical integration to calculate trajectories of particles in hydrodynamic models. But at some locations, with a large timestep or when using random motion, it is possible that the particles end up on land unintentionally, outside of the domain, and get stuck. In this thesis, we develop a method to prevent particles getting stuck on the land by pushing the particles back into the ocean. It is found that it is possible to get less or even no particles getting stuck. Using the hydrodynamic fields from GlobCurrent, a push with a velocity of 10^{-2} meters per second halves the number of particles that get on land and a velocity in the order of 1 meters per second reduces that number to zero.

Contents

1 Introduction

Every year, many tonnes of plastic waste end up in the ocean. For example, in Jambeck et al. [1], it is estimated that in 2010, between 4.8 and 12.7 million metric tonnes of plastic waste entered the ocean only from coastal regions. Other possible sources of plastic debris, for example ships, fishers and inland regions, are not included, so the total number of tonnes per year would be larger. Some important questions are: how much plastic is in the ocean and what happens to the plastics that are in the oceans?

We already do know some of the problems caused by these plastics. From observations it became clear, that they are bad for the environment. Especially marine life suffers a lot from plastic, according to Laist [2] and Gall and Thompson [3]. We see animals trapped by plastics. We also find animals that have died as a result of many plastics in their stomach. So it is important to remove the waste from the ocean. Questions here are then: what are methods to remove the plastics from the ocean and what are the best locations to start? Examples of methods currently used are cleaning up waste from beaches where part of the plastics end up and fishing plastics out of the water. But currently that is not good enough, as this is just a small part of all plastics in the oceans.

From observations it is known that a part of all plastics in the ocean accumulates in so called garbage patches (Lebreton et al. [4]). They are located in ocean gyres. These are large systems of circulating ocean currents, formed as a result of global wind patterns, Earth's rotation and Earth's continents. While most plastics are found in these garbage patches, these are still large enough to be observes. We also know that a portion of plastics may become of microscopic sizes. This can happen due to degradation, like mechanical forces andor photochemical processes, of (larger) plastics. This also means that plastics have a long live time, so they stay for a long time in the oceans, as larger items or even as microscopic pieces.

Due to the long live time and interactions with the environment, it is important to know how much plastics is in the ocean and where these are. By only using observational data, it is impossible to solve these points. Not only are the found items large, they are also buoyant and on the surface or already on the coast. So we are not getting much data on smaller plastics and pieces that are deeper in the oceans.

1.1 Numerical model simulations

A way to get more information on the distribution and pathways of smaller (and large) plastics, numerical model simulations can be used (Hardesty et al. [5]). The first set of models use data from drifting buoys. Here buoys that move in the ocean are tracked. The information about the (reported) locations of the buoys can be used to create a statistical model of surface pathways in the ocean (van Sebille et al. [6]). This model can then be used to get an idea of what happens to (virtual) particles, like plastics, if they are released somewhere in the ocean. In the end, the model gives a prediction of the pathways of these (virtual) particles as they move along the surface pathways in the ocean. The important point of this kind of model is the amount of data from buoys. To get a high resolution model for all oceans, many releases of such drifting buoys are needed.

1.1.1 Lagrangian ocean analysis

An other set of models make use of Lagrangian ocean analysis. Here virtual particles, like parcels of seawater, plastics or icebergs, are tracked within flow fields of hydrodynamic models (for example Lange and van Sebille [7]). These hydrodynamic models can be outputs of ocean general current models (OGCMs) or they can be observational data. The OGCMs describe physical and thermodynamical processes in ocean. Examples of OGCMs are NEMO (NEMO European Consortium [8]) and MITgcm (Massachusetts Institute of Technology [9]). The models

use different equations, that describe geophysical flows, to calculate flow velocities for points on a predetermined grid, placed on over the Earth's surface. Usable observational data is for example GlobCurrent (the European Space Agency [10]). For the use for Lagrangian ocean analysis, the velocity fields are the most important.

The basis of particle tracking in Lagrangian ocean analysis is equation (1) from Lange and van Sebille [7], which is given by

$$\vec{X}(t+\Delta t) = \vec{X}(t) + \int_t^{t+\Delta t} \vec{v}(\vec{x},\tau)\,\mathrm{d}\tau + \Delta\vec{X}_b(t). \tag{1}$$

Here, $\vec{X}(t)$ is the position of the particle at time t, $\vec{X}(t+\Delta t)$ is the position after a timestep Δt, $\vec{v}(\vec{x},t)$ is the ocean velocity field and $\Delta\vec{X}_b(t)$ is the change of position due to some kind of effect, like a random motion. For simple cases, the random effect $\Delta\vec{X}_b(t)$ can be zero. To determine the new position of a particle, the integral has to be calculated. This can be done by numerical integration. In most cases, the fourth order Runge-Kutta method or the Forward Euler method is used. The result of the fourth order Runge-Kutta is more accurate than that of the Forward Euler, but it needs four extra steps to calculate the result, so it is a bit slower. Because a particle can be everywhere on the grid, but the velocity field is only defined on specific gridpoints, the value of $\vec{v}(\vec{x},t)$ has to be approximated by interpolation. In the case of nearest-neighbour interpolation, the value of $\vec{v}(\vec{x},t)$ is the value at the closest gridpoint. For linear interpolation, the value depends on the distance to the closest gridpoints. In the case of a function $f(x)$ at given points x_0 and x_1, we can find the value at an x_* between the two points by drawing a straight line between $f(x_0)$ and $f(x_1)$. Then, the value at x_* is the value at the drawn line at x_*. If there are more dimensions, then the interpolation has to be done for all of them.

There are a number of models that use Lagrangian ocean analysis for tracking virtual particles in OGCMs. One is PARCELS by OceanParcels [11], which is discussed in Lange and van Sebille [7]. PARCELS is an open-source framework in Python. Important points of the framework is that it is modular and flexible. By default it uses the fourth order Runge-Kutta to calculate new positions of particles using equation (1), but other integration methods, like Forward Euler, can be used. It can be customized to also include a random motion $\Delta\vec{X}_b(t)$.

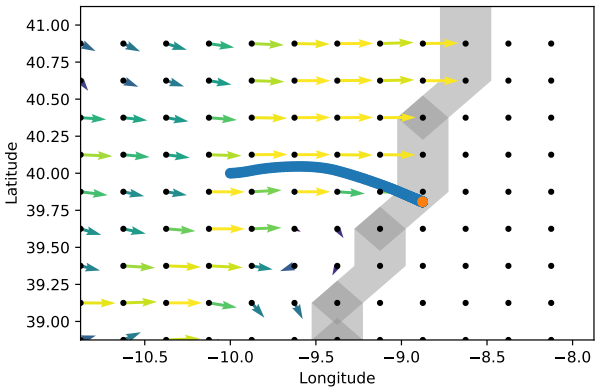


Figure 1: Trajectories of a particle released at $(10, 40\mathrm{S}, a\mathrm{E})$nd advected for 50 days. Shown in orange and blue are the last and previous locations of the particles. Difference in time between two locations is 6 hours. Furthermore the velocity field is shown at the black gridpoints. A yellow arrow means a higher absolute velocity. Green and blue arrows correspond to lower velocities. The gray areas show the coastal gridpoints. For the plots, a single non changing velocity field is used. Here this is data from GlobCurrent at date $2018-02-25$ at $12:00$.

When using large timesteps or large random motions, it is possible that a particle gets to places

where the velocity field is not defined by the used OGCM, like land. But also when the velocity field is pointing towards land, it is possible that a virtual particle gets stuck. Because the velocities are not defined on land, i.e. they are equal to zero, the particles can not return to the ocean. An example is shown in figure 1. In the plot, we see an orange particle with a blue trajectory getting stuck on the coast, which is shown as black dots on a gray background. Here it is caused by the fact that the field is pointing towards the land. In the case of a numerical simulation, we want to keep particles in our domain where the velocities are defined, which is in the ocean. A way to keep particles in our domain, we can try to push particles back in the ocean. In this thesis, this method will be tested and discussed.

In the following sections, the method will be explained in more detail. In section 3 two tests for the method are described. Section 4 contains the results of the tests and starting in section **??**, the method will be discussed.

2 Method

2.1 Flow fields

In this thesis we mainly use the data from GlobCurrent from the European Space Agency [10]. The data contains flow velocities at the surface, which are derived using different sources of data. These sources include satellite data, like altimetry and optical and passive microwaves, but also measurements from buoys and ship observations. GlobCurrent provides data for different dates and times from 2002 to 2014 with for every 3 hours a snapshot, in other words, the flow velocities changes with time.

All the sources of data resulted in ocean current fields for dates from 2002 to 2014 with for every 3 hours a different flow field. The current field here is the sum of the Ekman and the geostrophic components. In the datafiles the zonal and meridional velocities, U and V, are separated. The zonal velocity U is the velocity in the east-west direction and the meridional velocity V is the velocity in the north-south direction. A positive value for U means a velocity from west to east and a positive value for V corresponds to a velocity from south to north. The values of both velocity components are defined on an Arakawa A-grid with a resolution of $1/4°$, i.e. the space between two gridpoints is $0.25°$. The minimum and maximum longitudes are -179.875 and $+179.875$, which results in 1440 points in the east-west direction. For the latitudes, the minimum and maximum are -79.875 and $+79.875$, so there are 640 points in the north-south direction. In total there are 921600 gridpoints, where U and V are defined. Values for the velocities between gridpoints are approximated by linear interpolation.

2.1.1 Arakawa grids

In [14], Arakawa and Lamb have introduced five so called Arakawa Grids. These are labelled with A, B, C, D and E and are shown in figure 2 from [12]. In this section, we use ideas from [14], [12] and [15].

The A-grid is a so called unstaggered grid. The main idea of this grid is that all variables are defined on the same points on the grid. This can be the center or a corner of a gridcell. So in the case of the components U and V, these are both defined on the given gridpoints. In other words, the size of the data of U is the same as the size of the data of V, i.e. they have the same amount of values. This amount is also the same as the total number of gridpoints. An other advantage is that discretization is often easier on an A-grid than on a different grid, although this depends on the problem. In this thesis, all used grids are A-grids.

An Arakawa B-grid is a staggered grid, where the quantities are separated. So all velocity components can be located on the center of the gridcells, while all other quantities are defined

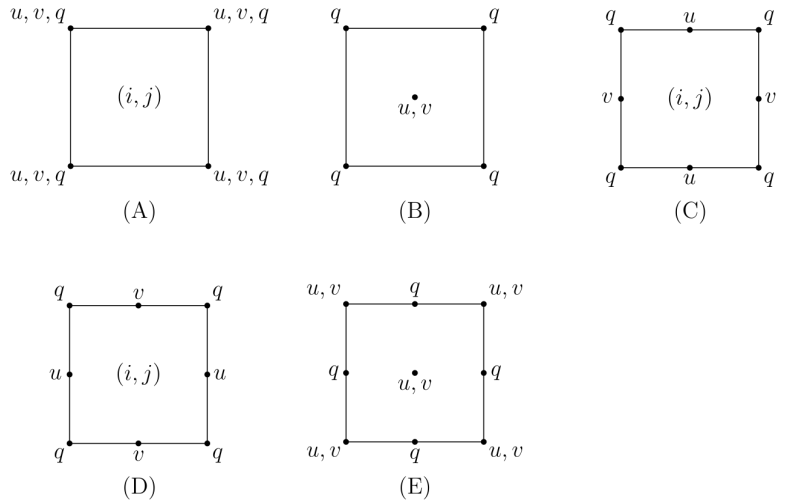


Figure 2: Different types of Arakawa grids. From left to right: The unstaggered A-grid, where all velocity components and other quantities, like pressure and temperature, are defined on the same points. The staggered B-grid, in which the velocities are separated from the other quantities. In the staggered C-grid, all velocity components are separated from each and from the other quantities, while non-velocity variables are still grouped. The D-grid is a variant of the C-grid. Finally the E-grid is a staggered grid where the velocity components are separated from the other quantities and are defined between the points where the other quantities are defined. Note that the dots do not necessarily mean that they are gridpoints, they are positions where the given variables are defined. The exact gridpoint in a cell can be chosen everywhere, but it is useful to have a gridpoint at a position where the most variables are defined. Source: [12], [13]

on the corners. It is also possible to have the velocity components on the corners and the other quantities in the center. This kind of grid can be useful when a discretizised equation needs quantities halfway two gridpoints.

The C-grid is one that is widely used in modelling of atmosphere and oceans. In this staggered grid, the velocity components U and V (and W for three dimensions) are separated. These are now placed at the centers of the cellborders, assuming the that the gridpoint is the center of the gridcell. So a two-dimensional example is that U is defined at the upper and lower grid borders, and that V is defined at the left and right grid borders. In the figure the other quantities are located on the corners of the gridcell.

A variant of the C-grid is the D-grid. A D-grid is obtained by rotating the C-grid by 90 degrees. Here U is defined at the left and right grid borders and V on the upper and lower borders.

The E-grid is also a staggered grid. The difference with the other grids is that it is rotated by 45 degrees. The result is a grid where all quantities are defined along the borders of a gridcell. A disadvantage is that the gridpoint density is larger than the other grids. This can be seen by comparing it to the A-grid.

In general, the choice of using a specific grid, is determined by the problem. For example, if the used equations or model uses quantities on locations that are also gridpoints, then an A-grid is useful. The use of a different grid-type, also leads to different discretization errors.

As noted before, in this thesis data from [10] is used. All velocity components are defined on an Arakawa A-grid.

2.2 Parcels

To track virtual particles, we use PARCELS [11]. The source code can be found on [16] and is described in [7]. This is a framework in Python which allows to load output from hydrodynamic fields and use them to calculate pathways of virtual particles, using equation (1).

The inputted data, in this case data from GlobCurrent, is converted to a set of fields (`FieldSet`). Here there are two `field`s, one contains the zonal velocity U and the other contains the meridional velocities V. Other data sources may contain more `field`s, for other quantities like pressure or temperature. Then we can create a `ParticleSet` which contains the virtual particles, each placed on given locations. These particles can then be advected to determine new locations and thus pathways.

For the calculation of new locations, equation (1) is used. Rhe time-integration can be done by using the fourth order Runge-Kutta method (RK4). A reason to choose this method over other methods is due to stability and accuracy. The Forward Euler (FE) method in section 1 is less accurate for larger timesteps than RK4. Also has RK4 a larger stability region than FE, which means that errors in calculations have a lower chance to grow exponentially. In general, this chance depends on the type of differential equation or integral and on the values in the equation. The method RK4 is described in, for example, [17] and is given by

$$
\begin{aligned}
Y_1 &= y_i, \\
Y_2 &= y_i + \frac{h}{2} f\left(Y_1, t_i\right), \\
Y_3 &= y_i + \frac{h}{2} f\left(Y_2, t_{i+1/2}\right), \\
Y_4 &= y_i + h f\left(Y_3, t_{i+1/2}\right),
\end{aligned}
$$

$$
y_{i+1} = y_i + \frac{h}{6}\left(f\left(Y_1, t_i\right) + 2f\left(Y_2, t_{i+1/2}\right) + 2f\left(Y_3, t_{i+1/2}\right) + f\left(Y_4, t_{i+1}\right)\right).
$$

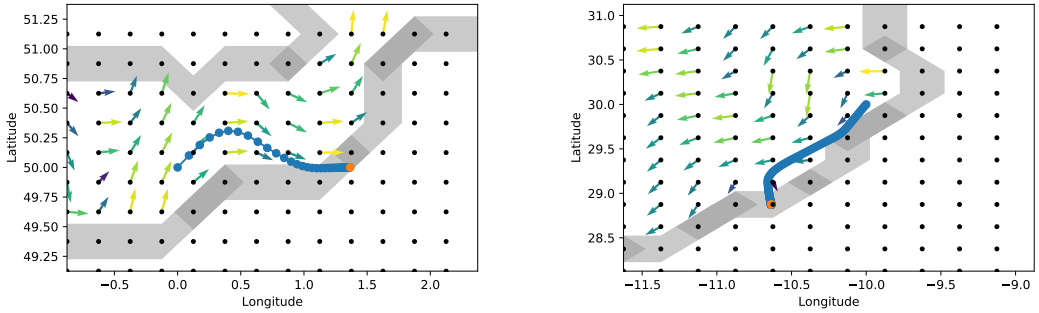
To compare it with (1), we have that the function $f(x,t)$ corresponds to $\vec{v}(\vec{x},\tau)$, step size h to Δt, x to $\vec{x}$ and t to τ. Also note that y_i corresponds to $\vec{X}(t)$ and y_{i+1} to $\vec{X}(t+\Delta t)$. The accuracy is $\mathcal{O}(h^4)$. This can be compared to the simpler Forward Euler (FE) (see for example [17]), which is given by

$$
yi + 1 = y_i + h * f\left(y_i\right)
$$

and has an accuracy of $\mathcal{O}(h)$. Because RK4 needs more calculations to find a result than FE, it is slower. But on the other hand, RK has a better accuracy, so we can use larger steps (h) than with FE, without getting too much numerical errors.

2.3 Coasts

Although tracking virtual particles on a hydrodynamic field using PARCELS works for particles in the ocean, there are some issues with particles are close enough to land to get stuck on land. Examples are shown in figures 3a and 3b. Here two particles are released at locations $(0, 50)$ and $(-10, 30)$ and are advected for 10 hours and 50 hours with a timestep of $dt = 10$ minutes. The starting time is the date $2014 - 02 - 15$ at $12 : 00$. Plotted are the vector field, where the vectors are calculated from the U and V components in the `FieldSet`. Also shown are the coastlines as gray areas, which will be discussed later in section 2.3.1. The blue lines and dots are historical positions of the particle with a constant time between two dots, while the orange dot is the location at the end of the simulation. Here we see that the two particles follow the flow and in the end, the stop on the land. This is where both velocities U and V are 0, shown in the plot by black dots without arrows.



(a) Trajectory of a particle that is released at $(0\text{N}, 50\text{E})$ and was advected for 10 days.

(b) Trajectory of a particle that is released at $(10\text{S}, 30\text{E})$ and was advected for 50 days.

Figure 3: Trajectories of particles. Shown in orange and blue are the last and previous locations of the particles. Difference in time between two locations is 6 hours. Furthermore the velocity field is shown at the black gridpoints. A yellow arrow means a higher absolute velocity. Green and blue arrows correspond to lower velocities. The gray areas show the coastal gridpoints. For the plots, a single non changing velocity field is used. Here this is data from GlobCurrent at date $2018 - 02 - 25$ at $12:00$.

The problem here is that the vector field points in the direction of gridpoints where the velocities are not defined, i.e. gridpoints on land. One could argue that this corresponds to beaching of a particle, but this is not always true. The data used for particle tracking only contains ocean (surface) velocity fields. These surface velocities are not the only part that contributes to whether a particle beaches or not. But then we would need more data and data at a higher resolution. So for now, we want to keep the particles in our domain, the ocean where the velocity fields are defined.

2.3.1 Finding coasts

Here we develop a possible solution, where particles that get on land will be pushed back into the ocean. The idea is to push a particle back if it gets on a gridpoint on the coast. This way, the particle will get back into the domain.

First we need a new field, or a mask, that says ´here starts land´ or ´this is a coast´. To create such a field, we have to determine where these coasts are. As explained, the only data we have contains only information about the velocities U and V on all gridpoints. Because we know, from figures in 3, that the velocity on land is equal to 0, we can search for points where we have that one of the neighbouring gridpoints has a absolute velocity larger than zero, so $|U| > 0$ and $|V| > 0$, while the other point has also velocities of 0. More precise, we search for three points, where two points have no velocity. It is also possible with only two points, where one has no velocities, but then it is possible to flag small islands, containing one or two gridpoints, as coasts. Although these are coasts, they are not places where particles would get stuck, due to linear interpolation using neighbouring gridpoints.

Because we have two velocity components, U and V, we can split the problem in a north-south direction and a east-west direction. For the north-south direction, we only look for three vertical located points and the meridional velocities V. The sequences that are interesting, are $[v, 0, 0]$ and $[0, 0, v]$, with a v that is non-zero. Here we can assume we have one point in the ocean and two points on the land. If these are found, then it can be said that the gridpoint in the center is a coastal gridpoint. The same can be done for the east-west direction, by replacing V with U and instead of three vertical located points, the three points are located horizontal. By doing this search for all gridpoints, we can find the coasts in a given `FieldSet`. This has only to be

done for one snapshot, because we assume that the location of the land, or the gridpoints where the velocities are not defined, in the data for different snapshots is always the same.

Furthermore we save the locations of the coasts in four different `Field`s and we add these to the used `FieldSet`. The reason for using four fields is to be able to distinguish the four different coast orientations. For example a coastal gridpoint can have a gridpoint in the ocean in the north, while a different point can have an ocean in the south and west. So this way, we can know which points are on the coast and where the ocean is, without having to look again at the U and V fields.

2.3.2 Kernels

First we want to track how long a particle was near a coastal gridpoint, so we define a custom particle class and a custom kernel. A custom particle class is used to track different variables which have some connection with the coasts. Each particle has the same variables but with different values. The first variables are `´total_time_coast´`, `´current_time_coast´`, `´total_time_ocean´` and `´current_time_ocean´`. These are straightforward, the ´current time´ variables are the times a particle was on the coast or in the ocean since the last time it moved from one to an other. Here the definition of on the coast is the same as close to a coastal gridpoint, and in the ocean is the same as away from a coastal gridpoint. The ´total time´ variables are the total time a particle was on the coast or in the ocean, since the start of the simulation. If everything works as intended, the sum of these totals is equal to the run time. Other variables are `´number_on_coast´` and `´number_in_ocean´`. These are the number of times a particle switched from a gridcell in the ocean to a gridcell on the coast or vice versa. These can be used to calculate the average times on coasts and in the ocean.

To change the values of these variables, we use a custom kernel. In this kernel, the location of each particle is being examined at each iteration. If a particle is in a gridcell on the coast, the corresponding variables will be changed. So for the total and current times, we add the timestep. The same is done for the variables for the ocean. In addition, the current times will be set to zero if the opposing times increase. So if a particle moves from a gridcell in the ocean to the coast, then the current time for the ocean will be set to zero. This way one can tell where a particle was at the end of the simulation.

2.3.3 Pushing particles back

A second custom kernel is used to push the particle back from the coast to the ocean with a pre-defined magnitude. This magnitude is a constant saved in a `FieldSet`, using the method `FieldSet.add_constant`. The kernel itself manipulates the position of a particle by moving the particle a distance, conversion-factor $*$ magnitude $*$ dt, so the magnitude is considered as a velocity. The conversion-factor is used to go from meters (or meters per second) to degrees (or degrees per second). This means that the push is dependent on the length of the timestep dt. The direction of the push is determined by the orientation of the coast. So if the ocean is in the south of a coastal gridpoint, the particle will be pushed to the south, by adding $-$conversion-factor $*$ magnitude $*$ dt to the latitude (note that southward and westward need negative signs, while northward and eastward need positive signs).

With a small push-magnitude, it is possible that a particle on the coast stays on the coast, while a large magnitude is able to move a particle in one timestep to the ocean. The idea is then to find a value that is not too large to have too much effect and that is not too small to have no effect.

3 Experiment

3.1 Test-case: Peninsula

First, the effects of the method are being examined. We do this by using a theoretical model, a steady-state flow around a peninsula, as described by Ådlandsvik et al. [18] in section 2.2.2. For this thesis we use the implementation from Lange and van Sebille [7], which is optimized for use with PARCELS. In this implementation, the zonal and meridional velocity components, as well as the pressure are calculated for an Arakawa A grid with longitudes between 0° and 1° and latitudes between 0° and 0.5°.

In this thesis, the number of gridpoints in the longitudinal direction (east-west) and in the latitudinal direction (north-south) is 50. This resolution can be converted to a distance between gridpoints, by dividing the lengths of the domain by the number of points in each direction. We can use this distance and the maximum flow velocities to get a measure for the maximum usable timestep. In most cases, it is useful that particles do not skip one or more gridcells, because it is possible that the velocities of two neighbouring gridcells are very different. A simple example is a situation with two neighbouring gridcells, one with only a meridional velocity and the other with only a zonal velocity. If one of these two are skipped by a moving particle, it would not follow the streamlines. In the test case there are 50 grid points in each direction. With 50 points in the longitudinal direction, the distance is 0.018°. For the latitudinal direction, the distance between two neighbouring points is 0.009°. The maximum absolute value of U is 1.93 meters per second and for V the maximum absolute value is 0.94 meters per second. This leads to a minimum time, for a particle to move a whole gridcell, of 17.3 minutes for the longitudinal direction and 17.6 minutes for the latitudinal direction. So if the timestep is smaller than 17.3, particles do not skip any gridcells. Note that more gridpoints and thus smaller distances between them leads to a lower maximum timestep.

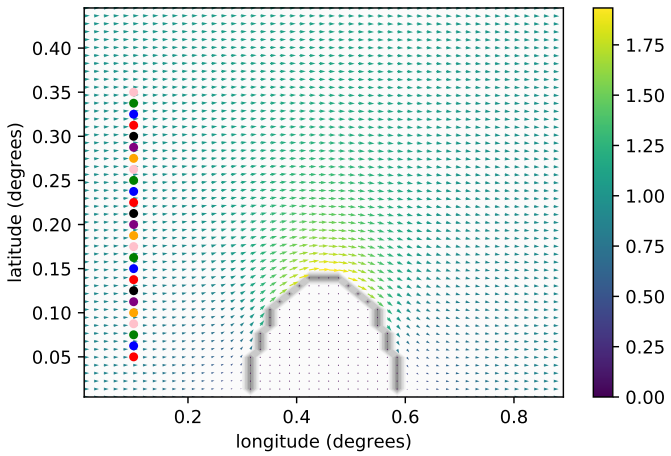


Figure 4

To test the effect of the method, we use the configuration as shown in figure 4. In the figure we see the peninsula in the middle at the bottom of the domain. The grey shade around the peninsula corresponds to the location of the coasts, found by the method in the sections starting from 2.3.1. The particles are shown on the left side. They all start at a longitude of 0.1° and a latitude between 0.05° and 0.35°. 25 particles are equally placed on this line. Also plotted is the velocity as vectors. For every point on the grid, the magnitude and the direction of the flow is shown. At this position the flow is laminar.

First the errors due to integration, using the fourth order Runge-Kutta method, are being

calculated. This will be done by using the pressure values. Because particles should follow streamlines, the pressure along the trajectory should not change. The particles will be advected for one timestep, for different timesteps between a few seconds and 17 minutes. The latter is chosen because it is close to the minimum time a particle needs to move from one border of a gridcell to the other border. After the single timestep, the change in pressure is calculated for all particles.

The next step is to check what happens when particles are given an artificial push. For multiple combinations of timesteps and push-magnitudes, the change in pressure is calculated. Push-magnitudes are in meters per second, so the result could be dependent on the timestep. The maximum push tested is about 10 meters per second while the minimum push magnitude is 0.01 meters per second. For the test, the artificial pushes are before the time integration. The direction of the pushes will be to the south. This means that the pushes are perpendicular to the laminar flow in the starting region. The result is a larger change in pressure, than when the particles are pushed along the streamlines.

For completeness, a single push without a time integration will also be tested. In other words, we only push the particle. these are between 0.01 and 1 meters. This will give more information about the effect of a push and can be compared to the first test with only a single integration.

3.2 Global scale

The final test is to use the method on a larger and more realistic situation. Now, the zonal and meridional velocities from GlobCurrent from the European Space Agency [10] and these are defined on a global scale. The grid is an Arakawa A grid with a resolution of $1/4°$ and is defined for longitudes between -179.875 and 179.875 degrees and for latitudes between -79.875 and 79.875 degrees. From the data and the given latitudes and longitudes, the minimum time a particle needs to move more than one gridcell is about 90 minutes. The first snapshot contains data from the date $2014-02-15$ at $12:00$.

The coasts are determined using the algorithm described in section 2.3.1. We determine the coasts only for the first snapshot.

Particles are placed at every 5 degrees between longitudes -175 and 175. The same goes for the latitudes, at every 5 degrees of latitude between -65 and 65 degrees the particles are placed. This results in 1917 particles in all oceans (and on land), each within a square of 5 by 5 degrees. The particles that start on land are removed. For these particles the initial velocities in all directions is equal to 0, so they can be removed easily. The number of particles that are being simulated is 1318.

First the effect of the timestep on the number of particles that get on the coast will be tested, by doing simulations with different timesteps, but without pushing the particles back in the ocean. The timesteps are between a few seconds to 20 minutes. Between 5 and 60 seconds there are 12 timesteps (spacing of 5 seconds) and between 1 and 20 minutes there are 20 timesteps (spacing of 1 minute). Although there are many timesteps, the focus here is on the timesteps 5, 10 and 15 minutes.

After looking at reference calculations, simulations will be done where particles are pushed when they reach a coastal gridpoint. The direction of the pushes is perpendicular to and away from the coast. The order of kernels is the following: first is the Runge-Kutta-4 advection kernel, then the particles are pushed when they get close to the coast and then the parameters of the particles, like time on coast gridpoints, are being examined and saved. For the experiment, different values for the magnitude of the pushes are used. These are 25 numbers, evenly spaced on a logarithmic scale between 10^{-5} and 10 meters per second. Note that the maximum absolute zonal and meridional velocities are a bit lower than 6 meters per second, they are about 5.8 and 5.3 meters per second for each component. So we are testing push magnitudes below (and a

fraction above) the maximum absolute flow velocity. The timesteps used in this test are 5, 10 and 15 minutes, so they are the same as for the test case in section 3.1.

4 Results

4.1 Test-case: Peninsula

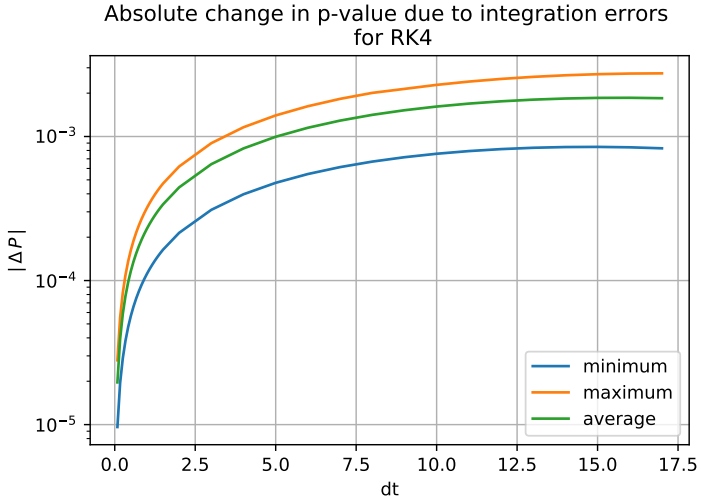


Figure 5: The absolute change of the pressure value due to a single integration for different timesteps.

In figure 5, the absolute change in pressure as result of integration errors is plotted against the timestep. Here we did only one integration by using a runtime that is equal to the timestep. In the figure, the maximum, minimum and average change are plotted. All increase as dt becomes larger, as expected because the error in position is bigger if the timestep is larger.

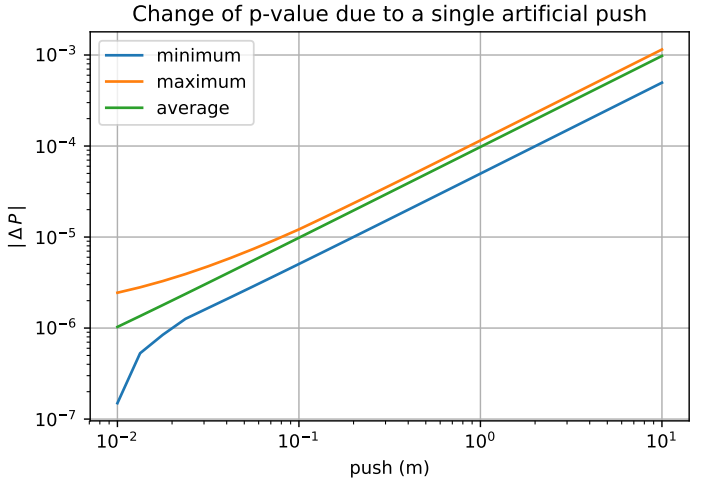


Figure 6: The absolute change of the pressure due to a single artificial push for different magnitudes.

The effects of a single artificial push, so without a time integration, can be seen in figure 6.

Here we see that if the push is multiplied by 10, then the change of the pressure value is also multiplied by 10. It is also possible to compare the change due to the artificial displacement to the change due to integration errors in figure 5. For a timestep of 5 minutes, the change in pressure is in the order of 10^{-3}. To get the same effect with only a push, this change of position has to be larger than 10 meters. This corresponds to a velocity of about $1.1 \cdot 10^{-2}$ for $dt = 5$ minutes and $3.3 \cdot 10^{-2}$ for $dt = 15$ minutes. If the artificial pushes are larger, then the effect of these pushes on the change in pressure is becoming larger than the effect of integration errors.

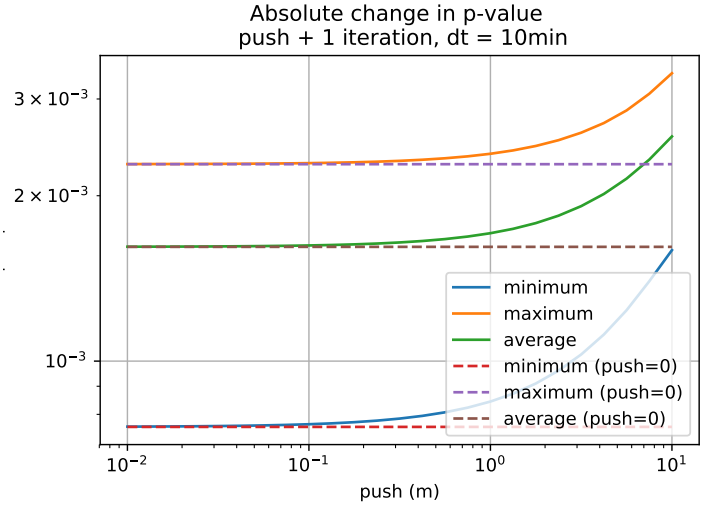


Figure 7: The absolute change of the pressure after a single integration with timestep $dt =$ 10 minutes and a single artificial push. Also note that the unit in the plot is wrong, this should be meters per second.

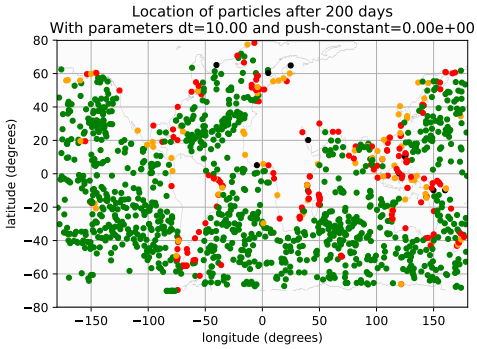
Figure 7 shows the absolute change after a push and one integration of respectively 10 minutes for different push-velocities. The results for the timesteps of 5 and 15 minutes are shown in figures 12 and 13 in appendix A. In all these plots, the changes in pressure when there is no push are plotted, these are drawn as dashed horizontal lines. Note that the pushes are in meters per second and therefore they depend on the used timestep dt. Also note that both axes are using logarithmic scales.

4.2 Global scale

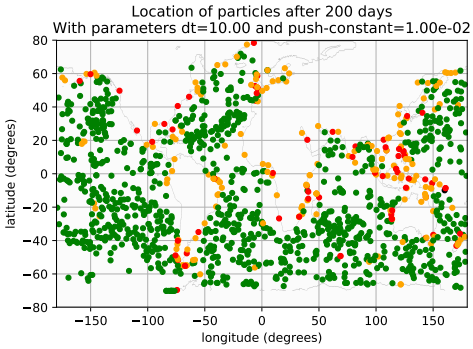
In this section, the results, for the test of the method of pushing particles back if they get close to the coast, are shown. Here GlobCurrent is used. The first snapshot was from $2014 - 02 - 15$ at $12 : 00$ and the runtime of the simulation was 200 days.

In figure 8 the final location of all simulated particles is shown. Green particles are particles that were always in the ocean, so they did not come close to a coastal gridpoint. Yellow dots are particles that do have reached such a gridpoint, but after some time they went back to the ocean. Red dots are particles that were on the coast at the moment the simulation ended. And finally the black dots are particles that were on the coast for the whole simulation. Also shown are the coasts itself as thin grey lines. Here the parameters are $dt = 10$ minutes and a push-magnitude of 0.0, 0.01, 0.1 and 1 meters per second for figures 8a, 8b, 8c and 8d. Results for timesteps $dt = 5$ and $dt = 15$ minutes are shown in appendix A in figures 14 and 15

If the plots in figure 8 are compared, we see that the number of red particles decreases with an increasing push-magnitude. The same is true for the number of yellow particles. Also visible is the location of these coloured dots. Most notable are the North Sea, the north-eastern part

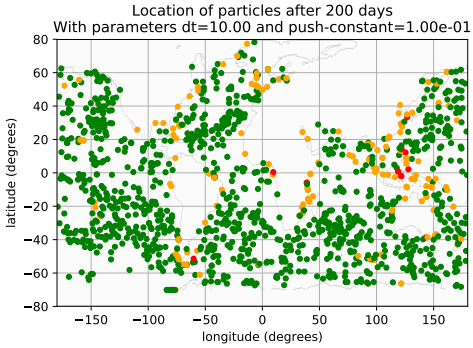


(a) Final locations for timestep $dt = 10$ minutes and a push-magnitude of 0 m/ s.

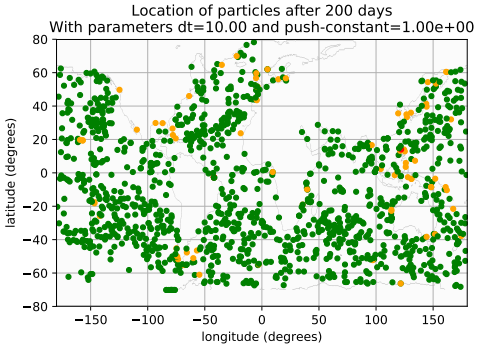


(b) Final locations for timestep $dt = 10$ minutes and a push-magnitude of 0.01 m/ s.

Figure 8



(c) Final locations for timestep $dt = 10$ minutes and a push-magnitude of 0.1 m/ s.



(d) Final locations for timestep $dt = 10$ minutes and a push-magnitude of 1 m/ s.

Figure 8: The locations of all particles at the end of the simulation. The start date is $2014 - 02 - 15$ at $12 : 00$ and the simulation time is 200 days. The green dots are particles that were always in the ocean. Yellow dots are particles that have been on the coast at least once. Red dots are particles that ended on the coast and black dots are particles that were on the coast for the whole simulation. Furthermore the coasts are plotted as gray lines.

of the atlantic and the coasts around South-East Asia. The issue there is that the currents are pointing towards land. So there the flows perpendicular and in the direction of the coast are larger than the flows parallel to the coast. This means that particles can easily get stuck on the coasts.

Figure 9 shows the number of particles that get on the coast at least once for different timesteps. In this case, there was no artificial push. In the figure we see that a higher timestep leads to a slightly higher amount of particles that get on the coast. But the difference is small if compared to the total number of particles, which is 1318. For timesteps 5, 10 and 15 minutes, the number of particles that get close to coast gridpoints is respectively 168, 172 and 177, which is about 12 to 14% of the total.

In figure 10 the number of unique particles that get on a coastal gridpoint is plotted for three different timesteps. On the x-axis is the magnitude of the artificial push in meters per second. The red line is a reference, which is the number for a timestep of 5 minutes and a push-magnitude of 0.0 meters/ seconds, i.e. there is no artificial push. What can be seen is that for larger magnitudes, the number decreases and eventually this number reaches to 0. Also note that for these timesteps, the curve is almost the same. Sometimes the number for $dt = 5$ is higher and at other points the number for $dt = 15$ is higher. But in the end, the number of particles that

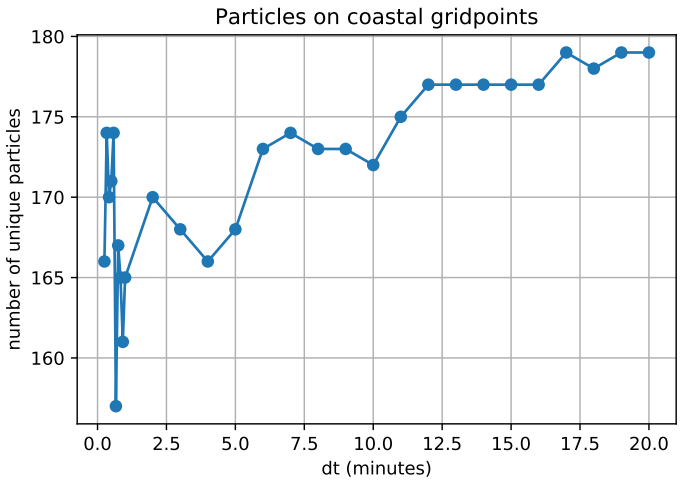


Figure 9: Number of particles that move on a coastal gridpoint at least once for different timesteps. Note that this is without artificial pushes.

reaches coastal gridpoints for a specific push-magnitude is for each timestep approximately the same.

The average time close to coastal gridpoints is plotted in figure 11. On the x-axis again the magnitude of the push and on the y-axis now the average time. Here only the particles in figure 10 are taken into account. So the sum of the total time on the coast is divided by the unique number of particles that contribute to the sum. Also the average time is set to 0 if the counted particles is 0. The plot has the same form as in figure 10. With again no result if the magnitude of the push is lower than 10^{-4} meters per second and low values, or maximum result, if the magnitude is larger than $1 (= 10^0)$.

5 Conclusion

In section 4.1 we see the effects of artificial pushes on the pressure value in an idealized case, here a peninsula. Also shown are the changes due to integration errors. Important to note is that the change in pressure is not larger than $3 \cdot 10^{-3}$ for the tested timesteps between a few seconds and 17 minutes. To get the same change in pressure by only using an artificial push, we would need a push of about 10 meters. This can be calculated back to velocities. So for a timestep of 5 minutes, the 10 meters corresponds to 0.03 meters per second. For a timestep of 15 minutes, this value of 10 meters corresponds to 0.01 meters per second.

If we look at the results for the more realistic global scale in section 4.2, we see that the artificial pushes actually helps with reducing the number of particles that get on the coast and with reducing the time a particle is on the coast. Because the push magnitudes are in meters per second, the effects are about the same for the tested timesteps 5, 10 and 15 minutes.

The number of particles that get on the coast halves when the push is in the order of 10^{-2} meters per second (see figure 10. While a push in the order of 1 meters per second will reduce the number of particles that get stuck on the coast to almost zero. Higher magnitudes are not necessary. The same can be said about the average time on the coast (see figure 11). Here we see that the average time halves when the magnitude of the push is in the order of 10^{-2} to 10^{-1} meters per second and it reduces to zero when the magnitude is larger than 1.

If we now combine the results of the idealized case and the global scale test, we can say that pushing particles from the coast to the ocean, is a plausible way to prevent many particles from

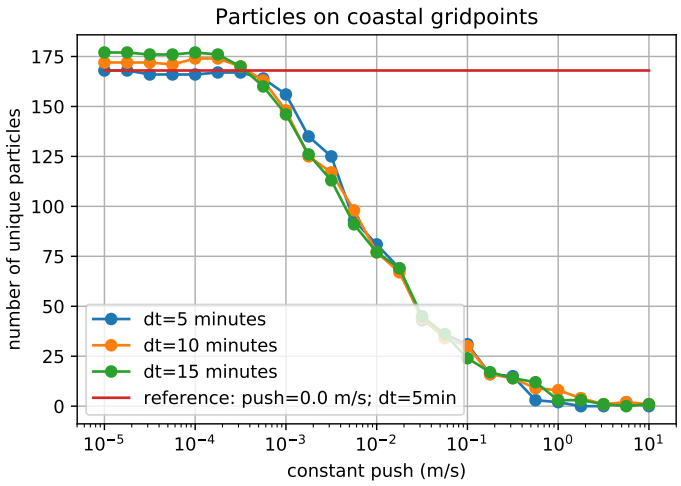


Figure 10: Number of particles that move on a coastal gridpoint at least once for different push-magnitudes.

getting stuck on the land. While the larger effects on the global scale were acquired with push magnitudes in the order of the maximum flow velocity in the used dataset (GlobCurrent), it is still useful to use a slightly smaller magnitude due to introducing artificial errors by moving particles away from their initial path.

6 Discussion

Although the problem of particles getting stuck on land can be solved by pushing them back into the ocean, it is not very physical. As already said, particles in the ocean follow streamlines. But here in our method, we push them away from the streamlines they were following. This means we change a lot of variables that (can) affect the particle, like pressure and temperature. This is not really examined in depth, because the used data did not provide data on pressure and temperature. This is why the test case with peninsula was used. Here it was possible to check what happens to the pressure when a particle is moved by a artificial push.

In the peninsula test case, a problem was that, in general, particles did not get stuck by advection. The only way to get particles on the land was by using very large timesteps, by altering the velocity fields in a specific way or by adding a (large) random motion. The first one is not useful, as using large timesteps will let particles move more than one gridcell. This leads to larger errors in position, which means that particles change to other streamlines. The second one is in general not physically correct. To get stuck particles, one could change the velocity fields such that particles move to the coast. But then the fields do not necessarily satisfy the equations of motion for fluids. The last one with random motion is not easy as there is a random displacement. In the case of a random displacement, the particle is able to move to other streamlines, which is what we try to prevent.

A different way to test with peninsula is to place the particles close to the coast. The problems with this method is that on the coasts and land, the velocity fields and pressure fields are not defined. We could get some numbers from this method, but they will never be as large as the largest numerical error one could make by pushing particles away. The problem here is the (linear) interpolation between gridpoints. While the velocity and pressure is defined on the gridpoint on the ocean, they are not defined (or zero) on the coastal gridpoint. This means that close to the coast, the values of the velocities and pressure are small (close to zero).

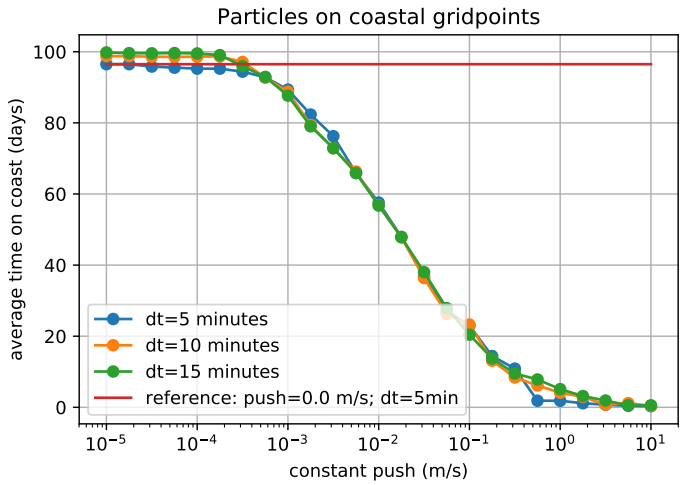


Figure 11: Average time on the coast for particles that get on the coast, plotted for different push-magnitudes.

If we now look at using an artificial push at a global scale (results in section 4.2), we can see the last locations of the particles at the end of the simulations in figures 8, 14 and 15. A possible effect of pushing a particle back if it comes close to the coast, is that it can affect the distribution of particles.

We can assume that the particles in the ocean are not affected by this push, as they do not come close to the coasts. These are the green particles, which should be on the same location for different push magnitudes. But there could be cases where particles are pushed back, without being noted that they were on the coast. This is due to the order of the used kernels. Here advection comes first, then the artificial push and then we check where a particle is.

In the figures we can see that the number of red and yellow particles decreases when increasing the push magnitude. This means that less particles are stuck at the end or that less particles get stuck in general. Although this is what we wanted, it is not necessarily correct. Depending on the magnitude of the push, it is possible to push particles in a different current with a different direction andor speed. To prevent this, it is recommended to use smaller magnitudes.

An important point is the time varying flow. A flow can change with every snapshot. So a different starting time will lead to a different distribution of particles at the end of a simulation. The figures in the introduction are therefore made using a non-changing flow field. It makes it easier to reproduce and to visualize the flow. The figures in the results are created with use of a time varying flow, with for all simulations the same start date and runtime. But with a different start time, it is certain that one would get different results.

The time varying flow can also be an issue when pushing particles back. The time it takes for a particle to get back in the ocean depends on the flow, the timestep and the magnitude. If this time is short, the particle would get back in the same flow as before. But if the time it takes is long enough, it is possible that a particle would get in a different flow. If this happens for all particles, it would alter the distribution of particles.

Lastly, some remarks on the method. In our method (see section 2.3.1) the check for a coast involves three gridpoints (one in the ocean, two on land), while it is possible to do it with only two (one in the ocean, one on land). The difference would be that small islands, that consists of only one or a few gridpoints, will be classified as coasts. A result of using two points would be that particles that get close to a small island will be pushed away, while it was not getting stuck due to (linear) interpolation. With two neighbouring land points, a particle would get stuck in between them, but with only one land point, a particle would not get stuck because of the land.

This is why the three point method is more useful than a two point method.

For the tests we used a specific order of kernels. As noted before, the order of kernels is as follows: advection, artificial push and then updating other variables in the particles. There is no particular reason to change this order, as advection is more important than the push, but it is possible that it is useful to do first a push towards the ocean and then do the advection, because then the velocities would be larger. The differences would not be that large that the distribution of particles changes much.

Concluding, there are many small details that have to be examined. Whether to use a large push magnitude or a small one. But on the larger scale and in the bigger picture, the method does decrease the number of particles on land and it does decrease the average time on land.

A Additional figures

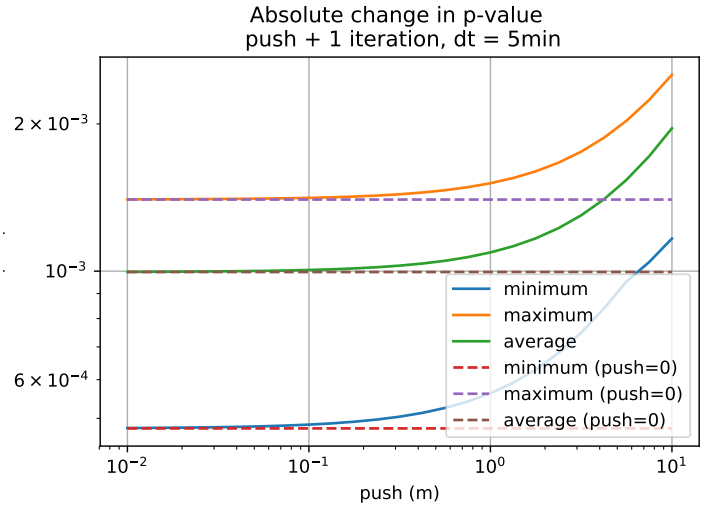


Figure 12: The absolute change of the pressure after a single integration with timestep dt = 5 minutes and a single artificial push. Also note that the unit in the plot is wrong, this should be meters per second.

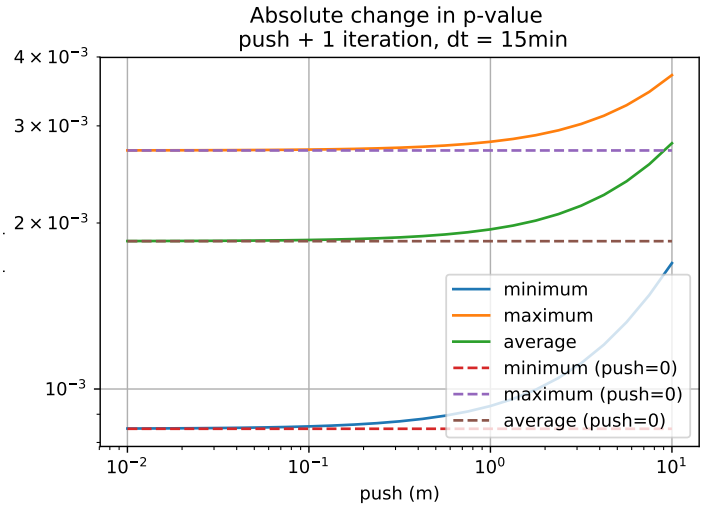
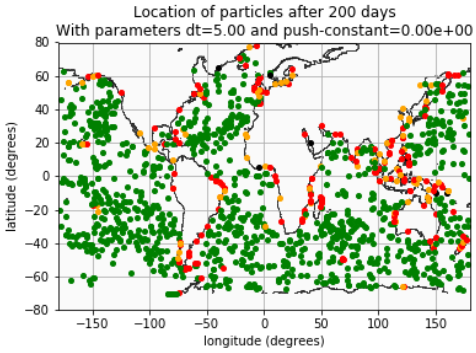
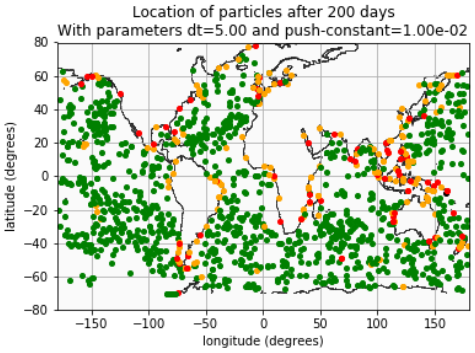


Figure 13: The absolute change of the pressure after a single integration with timestep dt = 15 minutes and a single artificial push. Also note that the unit in the plot is wrong, this should be meters per second.

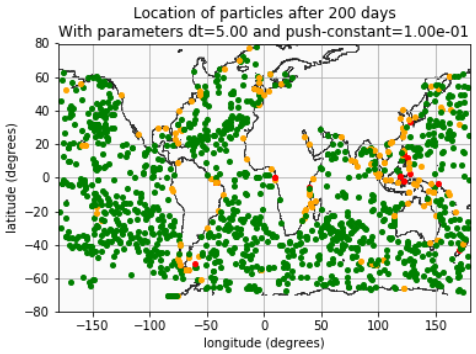


(a) Final locations for timestep $dt = 5$ minutes and a push-magnitude of 0 m/ s.

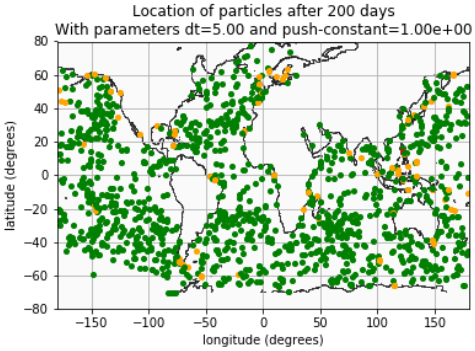


(b) Final locations for timestep $dt = 5$ minutes and a push-magnitude of 0.01 m/ s.

Figure 14



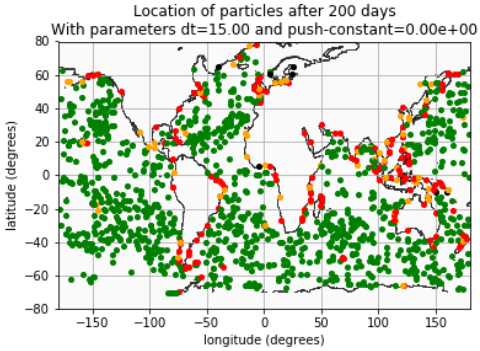
(c) Final locations for timestep $dt = 5$ minutes and a push-magnitude of 0.1 m/ s.



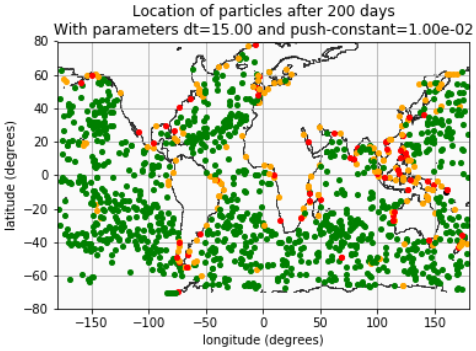
(d) Final locations for timestep $dt = 5$ minutes and a push-magnitude of 1 m/ s.

(e)

Figure 14: The locations of all particles at the end of the simulation. The start date is $2014 - 02 - 15$ at $12:00$ and the simulation time is 200 days. The green dots are particles that were always in the ocean. Yellow dots are particles that have been on the coast at least once. Red dots are particles that ended on the coast and black dots are particles that were on the coast for the whole simulation. Furthermore the coasts are plotted as gray lines.

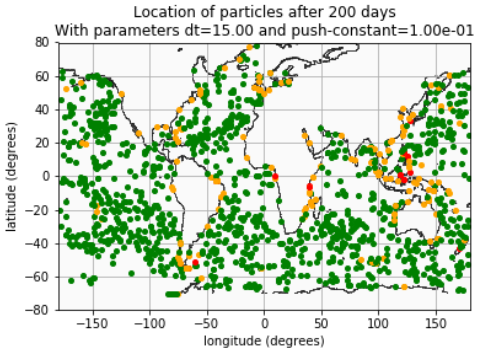


(a) Final locations for timestep $dt = 15$ minutes and a push-magnitude of 0 m/ s.

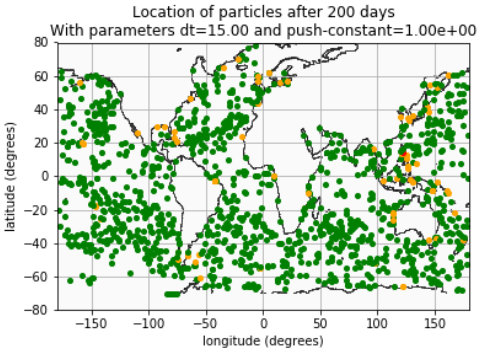


(b) Final locations for timestep $dt = 15$ minutes and a push-magnitude of 0.01 m/ s.

Figure 15



(c) Final locations for timestep $dt = 15$ minutes and a push-magnitude of 0.1 m/ s.



(d) Final locations for timestep $dt = 15$ minutes and a push-magnitude of 1 m/ s.

Figure 15: The locations of all particles at the end of the simulation. The start date is $2014-02-15$ at $12:00$ and the simulation time is 200 days. The green dots are particles that were always in the ocean. Yellow dots are particles that have been on the coast at least once. Red dots are particles that ended on the coast and black dots are particles that were on the coast for the whole simulation. Furthermore the coasts are plotted as gray lines.

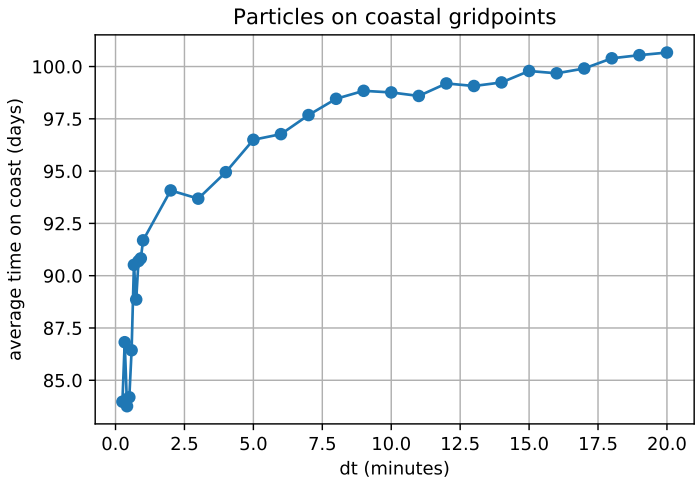


Figure 16: The average time a particle is on the coast for different timesteps. Simulated time was 200 days started at $2014-02-15$ at $12:00$. Here there were no artificial pushes.

References

[1] J. R. Jambeck, R. Geyer, C. Wilcox, T. R. Siegler, M. Perryman, A. Andrady, R. Narayan, and K. L. Law, Science **347**, 768 (2015), ISSN 0036-8075, http://science.sciencemag.org/content/347/6223/768.full.pdf, URL http://science.sciencemag.org/content/347/6223/768.

[2] D. W. Laist, Marine Pollution Bulletin **18**, 319 (1987), ISSN 0025-326X, URL http://www.sciencedirect.com/science/article/pii/S0025326X8780019X.

[3] S. Gall and R. Thompson, Marine Pollution Bulletin **92**, 170 (2015), ISSN 0025-326X, URL http://www.sciencedirect.com/science/article/pii/S0025326X14008571.

[4] L. Lebreton, B. Slat, F. Ferrari, B. Sainte-Rose, J. Aitken, R. Marthouse, S. Hajbane, S. Cunsolo, A. Schwarz, A. Levivier, et al., Scientific Reports **8** (2018), ISSN 2045-2322, URL https://doi.org/10.1038/s41598-018-22939-w.

[5] B. D. Hardesty, J. Harari, A. Isoba, L. Lebreton, N. Maximenko, J. Potemra, E. van Sebille, A. D. Vethaak, and C. Wilcox, Frontiers in Marine Science **4** (2017).

[6] E. van Sebille, M. H. England, and G. Froyland, Environmental Research Letters **7**, 044040 (2012).

[7] M. Lange and E. van Sebille, Geoscientific Model Development **10**, 4175 (2017).

[8] NEMO European Consortium, *Nucleus for european modelling of the ocean*, https://www.nemo-ocean.eu/, last accessed: 2018-06-13.

[9] Massachusetts Institute of Technology, *Mit general circulation model*, http://www.mitgcm.org/, last accessed: 2018-06-13.

[10] the European Space Agency, *Globcurrent*, http://www.globcurrent.org/, last accessed: 2018-06-13.

[11] OceanParcels, *Oceanparcels*, http://www.oceanparcels.org/, last accessed: 2018-05-30.

[12] W. contributors, *Arakawa grids*, https://en.wikipedia.org/wiki/Arakawa_grids, last accessed: 2018-05-30.

[13] R. ocn, *Different arkawa grids*, https://upload.wikimedia.org/wikipedia/commons/8/8a/Discretization_for_the_different_Arakawa_grids_correct.svg, last accessed: 2018-05-30.

[14] A. Arakawa and V. Lamb, Methods of Computational Physics **17**, 173 (1977).

[15] B. Cushman-Roisin and J. Beckers, *Introduction to Geophysical Fluid Dynamics: Physical and Numerical Aspects*, International Geophysics (Elsevier Science, 2011), ISBN 9780080916781, URL https://books.google.nl/books?id=4g-6iX6EQ8YC.

[16] OceanParcels, *Main code for parcels*, https://github.com/OceanParcels/parcels/, last accessed: 2018-05-30.

[17] U. M. Asher and C. Greif, *A First Course in Numerical Methods* (Society for Industrial and Applied Mathematics, 2011), ISBN 9780898719970.

[18] B. Ådlandsvik, J. Bartsch, D. Brickman, H. I. Browman, K. Edwards, O. Fiksen, A. Gallego, A. J. Hermann, S. Hinckley, E. Houde, et al., ICES Cooperative Research Report (2009).