UPPSALA
UNIVERSITET

Numerical methods for studying the vortex patch problem

Mattias Bertolino, Jonas Melander, Lina Viklund

Project in Computational Science: Report

February 2018

PROJECT REPORT

Institutionen för informationsteknologi

Abstract

In 2005 Córdoba et al. used a numerical method based on interpolation with cubic splines to solve the α-patches problem, which describes an interpolation between the 2D Euler equation and the surface quasi-geostrophic equation. This paper presents a comparison between their method and a pseudo-spectral method. The properties of both methods are described, along with details about their implementations. Comparisons of convergence rate, accuracy and computational speed, show that the pseudo-spectral method is superior in these regards.

Acknowledgements

The continuous discussions with Jordi-Lluis Figueras, our supervisor in this project, significantly helped us in forming and finishing this project. We specially thank him for his support, his guidance and his interest in what we achieved. We would also like to thank Maya Neytcheva, our project coordinator, for the provided structure and expertise. All of your inputs have contributed considerably to this project.

Keywords

Vortex patch problem, α-patches problem, discontinuous vorticity equations, 2D Euler equations, Surface Quasi-Geostrophic equations, Pseudo-spectral methods.

Populärvetenskaplig sammanfattning

En stor fråga inom vetenskapen är om de så kallade Navier-Stokes ekvationerna är stabila, det vill säga om de vid beräkningar med realistiska värden också ger realistiska svar. Navier-Stokes ekvationer är ett matematiskt problem som har tillämpningar i väldigt många områden inom vetenskapen idag. Stora myndigheter som till exempel NASA, ESA och SMHI, men även företag inom datorspelsvärlden lägger pengar och beräkningskraft på att lösa detta problem. Navier-Stokes ekvationer används till exempel för att beskriva luftflöden kring vingar eller runt raketer, men de används även till att beskriva hur rök rör sig i datorspel.

Det här arbetet riktar in sig på en grupp relaterade problem, som är enklare att behandla men har en nära relation, och liknar således Navier-Stokes ekvationer. Det enklare problemet kallas för alpha-patches problemet. Ekvationerna i alpha-patches problemet kan användas för att beskriva hur ett område med en konstant rotation beter sig i tiden när det är omgivet av ett annat område som inte roterar, som till exempel en virvel eller en orkan. Här undersöker vi tidsutvecklingen för två sådana områden bredvid varandra och kan se hur de efter ett tag närmar sig varandra och till slut i princip kolliderar. Mellan dessa två områden finns så kallade skarpa fronter, som till exempel kan jämföras med hög- och lågtrycksfronter i meteorologin. I och med att sådana fronter existerar kan områdena kollidera men inte slås ihop. Fronterna är intressanta att undersöka, för att det är här instabiliteterna kan uppkomma.

Dessa problem har studerats i flera år, och vad många av studierna har gemensamt är att metoden som hittills använts för att lösa dem lätt blir långsam vid noggranna beräkningar. En snabbare och mer noggrann metod behövs alltså. I denna rapport beskrivs hur vi har implementerat den gamla metoden, men också hur vi utvecklat en ny, snabbare metod. Vi jämför sedan för- och nackdelar med de båda metoderna. Resultatet visar hur den ny-utvecklade metoden är både snabbare och mer noggrann. Dock finnes även nackdelar hos den nya metoden, såsom hur små fel som introduceras under beräkningsprocessen inte behandlas i den kod som skrivits. Däremot finns det möjligheter att kombinera metoderna och på så sätt begränsa nackdelarna hos dem båda.

Contents

1 Introduction

It is still an open question in mathematical fluid dynamics whether or not the Navier-Stokes equations develop singularities in finite time. This problem was stated by Charles Fefferman as one of the Millenium Prize Problems [1]. These equations model how the velocity, viscosity, pressure, temperature, and density of a moving fluid are related. They can be used to model a wide variety of physical phenomena, and because of this they are of great interest to physicists, mathematicians and engineers. With the help of high-speed computers and numerical methods such as finite differences, finite volumes and spectral methods, it has been possible to compute approximations to the solution. However, the questions regarding the model's well-posedness remains. To guide the analytical studies in the search for the existence of singularities, numerical simulations provide a great tool to sort the pile of possible occurrences of such. While numerical methods can provide rigorous proofs, these are more often used to give indication of where to focus analytical studies. In a 2014 paper by Scott and Dritschel [6], evidence of a self similar collapse is given. An ellipsoidal patch is evolved until it collapses on itself, possibly forming singularities.

If the effects of viscosity in the fluid are neglected, the Navier-Stokes equations reduce to the Euler equations. The 2D Euler Equations (2DEE) have been rigorously proved to have global existence of regular solutions by Chemin [4], and Bertozzi and Constantin [5]. However, Studies [2, 7, 8] have shown indications of the existence of such singularities on the boundaries for a family of related contour dynamics equations. By linearly altering the stream function in the 2D Euler equations one obtains the Surface Quasi-Geostrophic Equations (SQGE), a system of equations with strong physical analogies to the 3D Euler Equations (3DEE), in which global existence of regular solution has not yet been proven. This means that we have two systems of equations, both with similarities to the 3D Euler Equations where one exhibits singularities and the other does not. Further, we have a possibility to linearly interpolate between them. They are also both two-dimensional, yielding a manageable size of the problem from a numerical perspective. It is thus of interest to study an interpolation between the 2D Euler equations and the surface quasi-geostrophic equation in search for singularities. This interpolated problem is known as the α-patches problem.

Córdoba et al. [2] and Mancho [3] provide numerical evidence of such singularities using an algorithm based on work done by Dritschel [12] which uses series expansion and an adaptive Runge-Kutta scheme, Runge-Kutta 4-5, to evaluate the derivative of the contour boundaries. The implemented method interpolates between nodes to represent the curve and uses the interpolation in a semi-discretised model of the problem to evolve the physical system. The focus of this study is to implement the method used in [2, 3] and compare it in terms of accuracy, speed and convergence rate to a pseudo-spectral method. This pseduo-spectral method uses the Fast Fourier Transform and Riemann sums to evaluate the velocity field of the contour boundaries in each time step. Our initial hypothesis is that this method would be both several times faster and more accurate.

Due to the connection between the α-patches problem and other vorticity equations, numerical evidence of singularities in the α-patches problem may be of interest to study further. This to give an indication of possible singularities in the related problems. To facilitate further numerical studies of this problem, faster and more accurate methods are needed, especially when the dimensionality of the problem increases.

Outline. In Section 2 the vortex patch problem and its connection to similar discontinuous vorticity problems is thoroughly explained as well as a derivation of a normal projected setting later used. In Section 3 a description of both implemented methods is given. The description includes how the vector field is evaluated, how node handling is done as well as some method-specific issues, such as precision tuning in the interpolation method and aliasing in the pseudo-spectral method. In Section 4 a description of the planning of the numerical study is given as well as a description of the technical setup used to perform simulations. Further, in Section 5 results of the comparison of the methods in terms of accuracy and speed is presented, alongside with a simulation showing numerical evidence of singularities in the problem setting. Finally, a discussion of the results and conclusions are given in Section 6 and Section 7 together with suggestions on further research and notes on the importance of the findings.

Contributions. All three authors have contributed significantly to the presented work. The work consisted among other things of developing theory, implementing the discussed ideas in C-code, running simulations to interpret and finally to summarize the work in this paper. Giving credits to one individual author for any specific part of the work would not describe the work flow in a fair manner. The discussions held along the project were of distinct importance for the development of this work. The C-code and supplementary material is available upon request.

2 The vortex patch problem

In many physical problems, the sought solution is not a solution whose dynamics can be described by smooth functions and models. Instead, the dynamics are described by sharp fronts and transients which appear locally. The vortex patch problem is such a problem, and what it intends to address is a situation where a bounded region exhibits a constant and intense vorticity whereas its surroundings do not. When seeking non-smooth solutions to the Euler equations one finds that this relates to a hurricane, where the change in vorticity is stepwise when entering or exiting the hurricane [23]. Another example from meteorology is when a moving temperature front enters a region where the potential temperature drastically changes from one level to another, as modeled by SQGE. A generalization of the vortex patch problem, the α-patches problem, seeks non-smooth solutions to an interpolation of the 2DEE and SQGE. In this section, we present the theoretical setting of the vortex patch problem and the transition to α-patches problem, but also its analogies to 3DEE. This is to give an understanding of one of the main issues with the α-patches problem, namely that on one end of the interpolation the 2DEE have regular solutions, but on the other end SQGE do not.

We begin with a definition of the discontinuous but bounded vorticity, or potential temperature in the context of the surface quasi-geostrophic equations, used to model the appearance of sharp fronts at the boundaries of the vortex patches.

Definition 2.1 (Discontinuous vorticity)**.** Let the vorticity θ be such that

$$\theta = \begin{cases} \theta_1, & \vec{x} \in D(t) \\ \theta_2, & \vec{x} \in \mathbb{R}^2 \backslash D(t) \end{cases},$$

where the patch is bounded inside $D(t) \subset \mathbb{R}^2$, with $\theta_1 \neq \theta_2$ and $\theta_1, \theta_2 \leq \mathbb{R}$.

A vortex patch consists of a simply bounded and connected inviscid, two-dimensional, region $D(t)$ with constant vorticity over the region defined as in Definition 2.1. Vortex patches are weak solutions to the Euler equations, a broad class of solutions which also include non-smooth solutions. A weak formulation of the vorticity-stream form of the Euler equations is used, since we are in need of a mathematical model other than partial differential equations to describe non-smooth behaviour of a physical system. Such weak solutions are defined in Section 2.2.

2.1 Analogies to other vorticity equations

In order to highlight the importance of studying the α-patches problem, and hence the importance of having proper numerical tools for studying it, we here show its analogies to other vorticity problems. One important example is the 3D Navier-Stokes system of equations

$$\frac{\partial u}{\partial t} + (u \cdot \nabla)u = \nu \Delta u - \frac{\nabla p}{\rho} + f,$$
$$\nabla \cdot u = 0,$$

where $u = (u_1, u_2, u_3)$ is a flow velocity vector satisfying $\nabla \cdot u = 0$, ν the kinematic viscosity, p the pressure, ρ the density of the fluid and f a given body load. Neglecting the effects of viscosity, the Navier-Stokes equations reduce to the Euler equations

$$\frac{\partial u}{\partial t} + (u \cdot \nabla)u = -\nabla p + f,$$
$$\nabla \cdot u = 0.$$

In vorticity form, this reads

$$\frac{D\omega}{Dt} = (\nabla u)\omega, \tag{1}$$

where the convective derivative is defined as

$$\frac{D\varphi}{Dt} := \frac{\partial \varphi}{\partial t} + u(\vec{x}) \cdot \nabla \varphi,$$

for any arbitrary function $\varphi = \varphi(\vec{x}, t)$ and with the vorticity ω satisfying $\omega = \nabla \times u$. We remark that the right hand side of Equation (1) does not appear in the 2DEE. The connection between 2DEE and 3DEE are clear, and therefore we only show the connections between SQGE and 3DEE. The SQGE are given by the system

$$\frac{D\theta}{Dt} = \frac{\partial \theta}{\partial t} + u \cdot \nabla \theta = 0, \tag{2}$$

where the velocity u is related to the stream function ψ as $u = (u_1, u_2) = \nabla^\perp \psi$ which in turn is related to the potential temperature θ as $-\Delta^{1/2}\psi = \theta$ in accordance with Zabusky et al. [13]. The notation $\nabla^\perp$ is used to denote the operator $(-\partial/\partial y, \partial/\partial x)$. By differentiating Equation (2), we obtain

$$\frac{D(\nabla^\perp \theta)}{Dt} = (\nabla u)\nabla^\perp \theta.$$

In the resulting system of equations, the term $\nabla^\perp \theta$ is analogous to the vorticity, ω, in the Euler equations. Both the Euler equations and SQGE are energy conservative and have integral curves of ω and $\nabla^\perp \theta$ that follow the fluid. More analogies are given by Rodrigo [21]. In Figure 1, the relation between the α-patches problem and 3DEE is clarified by a schema.

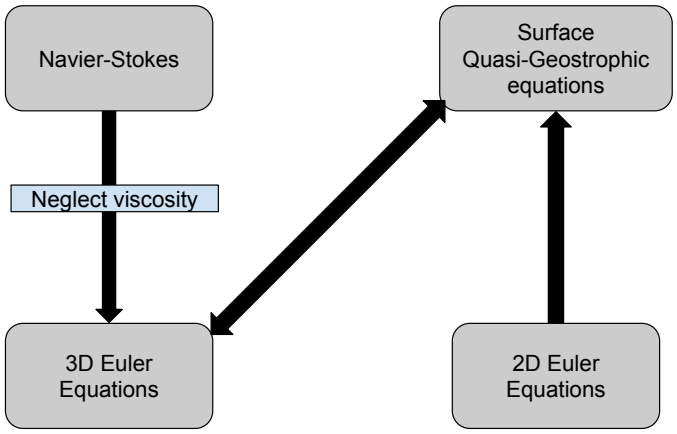

Figure 1: Schema depicting the relation between the α-patches problem and other discontinuous vorticity problems.

With the connection between SQGE or 2DEE and 3DEE established, it remains to define the weak solutions that exist for the 2DEE. Such a definition, as given by Majda and Bertozzi [8] and clarified by Kelliher [18], is shown in Section 2.2. The proof of existence of solutions in $\mathcal{L}^1(\mathbb{R}^2) \cap \mathcal{L}^\infty(\mathbb{R}^2)$ by Yudovich [19] and extended by Majda [20] is not given. In the following sections we adopt the notation of SQGE and use θ to denote the vorticity.

2.2 Weak solutions to the 2D Euler equations

While the connection between the Navier-Stokes equations and the Euler equations is clear, the relationship between the α-patches problem and the Euler equations is not as obvious. To highlight the similarities between the α-patches problem and 3DEE we show that a vortex patch with stream function ψ satisfying $-\Delta\psi = \theta$ is a weak solution to the Euler equations. Consider the vorticity-stream formulation of the Euler equations

$$\frac{D\theta}{Dt} = 0,$$
$$\theta|_{t=0} = \theta_0,$$

where the velocity $u(\vec{x})$ is determined by the vorticity θ as the convolution $u(\vec{x}) = (K * \theta)(\vec{x})$, that is

$$u(\vec{x}) = \int_{\mathbb{R}^2} K(\vec{\chi})\theta(\vec{x} - \vec{\chi})d\vec{\chi},$$

with the kernel

$$K(\vec{x}) = \frac{\vec{x}^{\perp}}{2\pi|\vec{x}|},$$

in accordance with Biot-Savart law. To cope with non-smooth vorticities of the type $\theta \in \mathcal{L}^1(\mathbb{R}^2) \cap \mathcal{L}^\infty(\mathbb{R}^2)$, we make use of an equivalent form of the vorticity-stream formulation, namely

$$\frac{d}{dt}\int_{\mathbb{R}^2} \varphi\theta d\vec{x} = \int_{\mathbb{R}^2} \frac{D\varphi}{Dt}\theta d\vec{x} + \int_{\mathbb{R}^2} \varphi\frac{D\theta}{Dt}d\vec{x},$$

where φ and θ are smooth and vanishing as $|\vec{x}|$ approaches infinity. It is important to mention that this formulation is possible since vorticity is constant along particle trajectories. Integrating in time, we obtain

$$\left[\int_{\mathbb{R}^2} \varphi(\vec{x},\tau)\ \theta(\vec{x},\tau)d\vec{x}\right]_{\tau=0}^{\tau=T} = \int_0^T \int_{\mathbb{R}^2} \frac{D\varphi}{Dt}\theta d\vec{x}d\tau + \int_0^T \int_{\mathbb{R}^2} \varphi\frac{D\theta}{Dt}d\vec{x}d\tau.$$

If we further assume that the vorticity θ is smooth, the velocity u is divergence free and hence the convective derivative is equal to zero and we get

$$\left[\int_{\mathbb{R}^2} \varphi(\vec{x},\tau)\ \theta(\vec{x},\tau)d\vec{x}\right]_{\tau=0}^{\tau=T} = \int_0^T \int_{\mathbb{R}^2} \frac{D\varphi}{Dt}\theta d\vec{x}d\tau,$$

for all test function $\varphi \in \mathcal{C}^1\left([0,T] \times \mathbb{R}^2\right)$ with compact support supp $\varphi(\cdot,t) \subset \{\vec{x} : |\vec{x}| < R\}$. We can conclude this derivation with the following definition.

Definition 2.2 (Weak solution 2DEE)**.** Let $\theta_0(\vec{x}) \in \mathcal{L}^1(\mathbb{R}^2) \cap \mathcal{L}^\infty(\mathbb{R}^2)$ be the initial vorticity. The pair (u, θ) is a weak solution to the vorticity-stream formulation of 2DEE if the following three conditions are fulfilled.

1. $\theta(\vec{x}) \in \mathcal{L}^\infty\{[0,T];\ \mathcal{L}^1(\mathbb{R}^2) \cap \mathcal{L}^\infty(\mathbb{R}^2),$
2. $u = K * \theta$ and $\theta = \nabla \times v$,
3. For all $\varphi \in \mathcal{C}^1\{[0,T];\ \mathcal{C}_0^1(\mathbb{R}^2)\}$ there holds

$$\left[\int_{\mathbb{R}^2} \varphi(\vec{x},\tau)\ \theta(\vec{x},\tau)d\vec{x}\right]_{\tau=0}^{\tau=T} = \int_0^T \int_{\mathbb{R}^2} \frac{D\varphi}{Dt}\theta\ d\vec{x}d\tau.$$

It is worth noting that the weak solutions have well-defined particle trajectories and are area-conservative although the length and curvature of the contour are not preserved. To show that a vortex patch is indeed a weak solution to the 2DEE, let $D_0 \in \mathbb{R}^2$ be a bounded and simply connected smooth region, a patch, with boundary ∂D_0. We seek a weak solution as $\theta(\cdot,t) \in \mathcal{L}^1(\mathbb{R}^2) \cap \mathcal{L}^\infty(\mathbb{R}^2)$ with vorticity

$$\begin{aligned}\theta(\vec{x},t) &= \theta_0, \quad \vec{x} \in D(t),\\ \theta(\vec{x},t) &= 0, \quad \vec{x} \notin D(t),\end{aligned}$$

and constant initial vorticity $\theta_0(x) = \theta_0$. The velocity is obtained with the Biot-Savart law, that is

$$u(\vec{x},t) = \frac{\theta_0}{2\pi}\int_{D(t)} \frac{(\vec{x}(\gamma',t) - \vec{x}(\gamma,t))^{\perp}}{|\vec{x}(\gamma,t) - \vec{x}(\gamma',t)|}d\gamma',$$

where γ and γ' are different parameterizations of the boundary $\partial D(t)$ and $\vec{x} \in \mathbb{R}^2$. Using Green's formula we get

$$u(\vec{x}(\gamma,t),t) = \frac{\theta_0}{2\pi}\int_0^{2\pi} \ln|\vec{x}(\gamma,t) - \vec{x}(\gamma',t)|\frac{\partial\vec{x}}{\partial\gamma}(\gamma',t)d\gamma',$$

which describes how a vortex patch evolves in time.

2.3 The α-patches problem

Consider a sharp stepwise vorticity in accordance with Definition 2.1 with zero vorticity outside the patch. In more general terms, however, the stream function can be expressed as any linear operator $L(\psi)$. In the 2DEE, the linear operator is defined as $L(\psi) = -\Delta\psi$ and in SQGE the linear operator is defined as $L(\psi) = -\Delta^{1/2}\psi$.

The α-patches problem generalizes the vortex patch problem by interpolating between the two underlying operators of the stream function. Let the interpolated linear operator be defined as

$$L(\psi) = -\Delta^{1-\alpha/2}\psi,$$

for $\alpha \in [0, 1)$. When α approaches 1, one obtains the linear operator in SQGE. When α approaches 0, the linear operator in the 2DEE is obtained. Using this linear operator, the velocity of a point $\vec{x}(\gamma, t)$ on a curve parametrized by γ in a problem with P patches, is given by

$$\frac{\partial \vec{x}}{\partial t}(\gamma, t) = \sum_{k=1}^{P} \frac{\theta_k}{2\pi} \oint_{C_k(t)} \frac{\frac{\partial \vec{x}_k}{\partial \gamma'}(\gamma', t)}{|\vec{x}(\gamma, t) - \vec{x}_k(\gamma', t)|^\alpha} d\gamma', \ \vec{x} \in \mathbb{R}^2, \tag{3}$$

where γ' is another parametrization of the kth contour.

2.4 Normal contribution of the vector field

According to the divergence theorem, the tangential component of the vector field over the contour will not act in deforming the curve, hence when evolving the curves, only the normal projection is of interest. By projecting the vector field in Equation (3), we get

$$\frac{\partial \vec{x}}{\partial t}(\gamma, t)\bigg|_n = \left(\frac{\partial \vec{x}}{\partial t}(\gamma, t), \ \vec{n}(\gamma, t)\right) \cdot \vec{n}(\gamma, t),$$

where the inner product of the vector field and the normal, $\vec{n}(\gamma, t)$, is given by

$$\left(\frac{\partial \vec{x}}{\partial t}(\gamma, t), \ \vec{n}(\gamma, t)\right) = \sum_{k=1}^{P} \frac{\theta_k}{2\pi} \oint_{C_k(t)} \frac{\frac{\partial \vec{x}_k}{\partial \gamma'}(\gamma', t) \cdot \vec{n}(\gamma, t)}{|\vec{x}(\gamma, t) - \vec{x}_k(\gamma', t)|^\alpha} d\gamma', \ \vec{x} \in \mathbb{R}^2.$$

Further, the normal is

$$\vec{n}(\gamma, t) = \frac{\partial_\gamma x(\gamma, t)^\perp}{||\partial\gamma x(\gamma, t)||},$$

where the path-derivative is denoted by

$$\partial_\gamma x(\gamma, t) = \frac{\partial \vec{x}}{\partial \gamma}(\gamma, t),$$

to ease the notation. Rather than projecting the vector field on the normal, it is equivalent to subtract the tangential component. Let the functional $F(\vec{x}(\gamma, t))$ denote the integral in Equation (3).

$$F(\vec{x}(\gamma, t)) \cdot \vec{n} = \left(F(\vec{x}(\gamma, t)) + a(\gamma, \gamma')\frac{\partial \vec{x}}{\partial \gamma}(\gamma, t)\right) \cdot \vec{n},$$

for any function $a(\cdot, \cdot)$ along the tangent. If we choose $a(\gamma, \gamma') = -1$, we end up with

$$\frac{\partial \vec{x}}{\partial t}(\gamma, t)\bigg|_n = \sum_{k=1}^{P} \frac{\theta_k}{2\pi} \oint_{C_k(t)} \frac{\frac{\partial \vec{x}_k}{\partial \gamma'}(\gamma', t) - \frac{\partial \vec{x}_k}{\partial \gamma}(\gamma, t)}{|\vec{x}(\gamma, t) - \vec{x}_k(\gamma', t)|^\alpha} d\gamma', \ \vec{x} \in \mathbb{R}^2. \tag{4}$$

Equation (4) has the advantage over Equation (3) that it is zero in the limit

$$\lim_{\epsilon \to 0} \int_{-\epsilon}^{\epsilon} \frac{\frac{\partial \vec{x}_k}{\partial \gamma'}(\gamma', t) - \frac{\partial \vec{x}_k}{\partial \gamma}(\gamma, t)}{|\vec{x}(\gamma, t) - \vec{x}_k(\gamma', t)|^\alpha} d\gamma' = 0,$$

whereas Equation (3) is not. In Section 3, we describe implementations for both expressions.

3 Implementation

In this section, we provide a description of the two studied methods and the assumptions that lie behind their implementation. The α-patches problem has some intrinsic difficulties to consider when implementing any method. Both implemented methods struggle with the occurrence of a locally divergent integral which forces the methods to either use a slower scheme or in some way modify the computations in order to avoid it. Focus is on the evaluation of the contour integrals, since it is the most involved step of both methods and is essentially where the two differ.

In each time step the velocity field at a point is calculated by evaluating the integral in Equation (5). The interpolation method, implemented by Córdoba et al. [2], approximates the integral by series expansion for three out of four cases to ease the evaluation, and uses an adaptive Runge-Kutta scheme for the fourth. The adaptive Runge-Kutta is both the most frequently used and time-consuming case of the four, and is the main motivation for the search of an alternative computational engine to evaluate the integrals. The pseudo-spectral method approximates the time-derivative as a Riemann sum over coefficients retrieved by the FFT, instead of subdividing the contour into different cases.

The two methods require the contour to be discretised in different ways. In the interpolation method the number of points are completely determined by an adaptive scheme, which also relocates points along an interpolation based on local curvature. The pseudo-spectral method on the other hand does not relocate or change the number of nodes. It works with a fixed number of nodes determined in the beginning of the simulation. Further, the number of nodes on the curve is fixed to a power of two, since our choice of FFT algorithm works most efficiently this way.

3.1 The interpolation method

Parts of the method used by Córdoba et al. [2] and Mancho [3] to describe the evolution in Equation (3) were first presented by Dritschel [12], and represent the contour with cubic splines as defined in Definition 3.1. This ensures that integration between the nodes is possible, and consistency is guaranteed since the fitting error decreases as the number of nodes increases.

Definition 3.1 (Cubic Spline Interpolation)**.** Let a cubic spline be defined as

$$\vec{x}_j(p) = \vec{x}_j + p\vec{t_j} + \eta_j(p)\vec{n}_j,$$

with $0 \leq p \leq 1$ such that $\vec{x}_j(0) = \vec{x}_j$ and $\vec{(x)}_j(1) = \vec{x}_{j+1}$. Further, let

$$\begin{aligned}
\vec{t_j} = (a_j, b_j) = \vec{x}_{j+1} - \vec{x}_j, \vec{t_j} \quad &\in \mathbb{R}^2, \\
\vec{n}_j = (-b_j, a_j), \vec{n}_j \quad &\in \mathbb{R}^2, \\
\eta_j(p) = \mu_j p + \beta_j p^2 + \gamma_j p^3, \eta_j \quad &\in \mathbb{R}.
\end{aligned}$$

The interpolation coefficients in η are given by

$$\begin{aligned}
\mu_j &= -\frac{1}{3} d_j \kappa_j - \frac{1}{6} d_j \kappa_{j+1}, \\
\beta_j &= \frac{1}{2} d_j \kappa_j, \\
\gamma_j &= \frac{1}{6} d_j (\kappa_{j+1} - \kappa_j),
\end{aligned}$$

with the distance between two successive nodes $d_j = |\vec{x}_{j+1} - \vec{x}_j|$ and local curvature $\kappa_j = 2\frac{a_{j-1}b_j - b_{j-1}a_j}{|d_{j-1}^2 \vec{t}_j + d_j^2 \vec{t}_{j-1}|}$.

By Definition 3.1 the semi-discrete formulation of Equation (3) becomes

$$\frac{d\vec{x_j}}{dt}(t) = \sum_{k=1}^{2} \frac{\theta_k}{2\pi} \sum_{i=1}^{N_k} \int_0^1 \frac{\frac{\partial \vec{x}_{i,k}}{\partial p}(p,t)}{|\vec{x}_j(t) - \vec{x}_{i,k}(p,t)|^\alpha} dp, \tag{5}$$

with p being the interpolation parameter.

The system in Equation (5) is thus given by $M = 2 \times (N_1 + N_2)$ autonomous functional differential equations, and in each time step the contour segments are evaluated using one of the four methods described in Section 3.1.2. Substituting the interpolation from Definition 3.1 into the equation, the integral to evaluate for each curve segment is given by

$$\int_0^1 \frac{\left((\vec{t}_i + \mu_i \vec{n}_i) + (2\beta_i p + 3\gamma_i p^2)\vec{n}_i\right)}{|\vec{x}_j - \vec{x}_i - p\vec{t}_i - \eta_i(p)\vec{n}_i|^\alpha} dp = \left(\vec{t}_i + \mu_i \vec{n}_i\right) \int_0^1 \frac{dp}{|\vec{x}_j - \vec{x}_i - p\vec{t}_i - \eta_i(p)\vec{n}_i|^\alpha} + \vec{n}_i \int_0^1 \frac{(2\beta_i p + 3\gamma_i p^2)}{|\vec{x}_j - \vec{x}_i - p\vec{t}_i - \eta_i(p)\vec{n}_i|^\alpha} dp, \tag{6}$$

where the contour index, k, is omitted for the sake of simplicity.

3.1.1 Relocation of nodes

When the patches are evolved through time, the length and curvature of the contour is not conserved. This causes problems with the spacing of the computational nodes, especially where the curvature is increasing. Sections of the contour where the curvature becomes acute are where singularities are thought to appear. Further, as the contour evolves, small numerically introduced errors may result in node clustering. To combat these issues and to keep an accurate representation of the curve, the nodes are globally adjusted along the contour.

The relocation of nodes is implemented in accordance with [12] and uses the local density of the nodes ρ_i to adjust the number of points on each segment. The local density of nodes is given by

$$\rho_i = \frac{\hat{\kappa}_i}{1 + \epsilon \hat{\kappa}_i / \sqrt{2}},$$

where the mean of the local curvature $\hat{\kappa}_i$ is given by

$$\hat{\kappa}_i = \frac{\tilde{\kappa}_i + \tilde{\kappa}_{i+1}}{2},$$

which in turn is given by

$$\tilde{\kappa}_i = (\nu L)^{-1} (\breve{\kappa}_i L)^a + \sqrt{2}\breve{\kappa}_i.$$

The user-chosen parameters ν, L and a control the accuracy of the representation by determining the number of nodes, and are chosen in accordance with [3]. The weighted sum of the curvature values at all nodes $\breve{\kappa}_i$ incorporates non-local effects and is given by

$$\breve{\kappa}_i = \sum_j \frac{d_j |\bar{\kappa}_j|}{h_{ij}^2} \left(\sum_j \frac{d_j}{h_{ij}^2} \right)^{-1},$$

where the distance d_j is given by Definition 3.1, $h_{ij} = |\vec{x}_i - (\vec{x}_{j+1} + \vec{x}_j)/2|$ is the distance between the node and the midpoint of the interpolation, $\bar{\kappa}_j = (\kappa_j + \kappa_{j+1})/2$ is the average local curvature, and κ_j is given by Definition 3.1. Once the local density ρ_i is obtained, the quantity $\sigma_i = \rho_i d_i$ is computed. This gives the fractional number of nodes that is to be placed between nodes i and $i+1$. To find the new total number of nodes to represent the contour, $\tilde{N}_k$, the quantity q is computed as

$$q = \sum_{i=1}^{N_k} \sigma_i,$$

and $\tilde{N}_k = [q] + 2$, i.e. the nearest integer to q plus two. One node on a contour k is held fixed during the relocation process. Thus $N_k - 1$ old nodes will be replaced with $\tilde{N}_k - 1$ new nodes, which will be placed along the contour segments connecting the old nodes. To find the positions of the new nodes $j = 2 \ldots \tilde{N}_k$, we let $\sigma_i' = \sigma_i \tilde{N}_k / q$ such that $\sum_{i=1}^{N_k} = \tilde{N}_k$, and successively seek i and p such that

$$\sum_{l=1}^{i-1} \sigma_l' + \sigma_i' p = j - 1.$$

Each new node j is then placed between the old nodes i and $i+1$ at the position $\vec{x}_i(p)$, given by the equation defined in Definition 3.1.

3.1.2 Evaluation of the contour integrals

The velocity field at x_j can be classified into four cases [3] and evaluated based on its relative position to the node at x_i. Runge-Kutta 4-5, which is used in Case 4, could be used for all cases, but in order to speed up calculations the integrals are classified as presented below.

1. $\vec{x}_j = \vec{x}_i$.
   Although the integrand of the first integral in Equation (6) goes to infinity at $p = 0$, its principal value is defined for $\alpha \in [0, 1)$. Rewriting the integral as follows, it can be evaluated using series expansion as

$$\begin{aligned}\int_0^1 \frac{dp}{|p\vec{t}_i + \eta_i(p)\vec{n}_i|^\alpha} &= \frac{1}{|\vec{t}_i|^\alpha}\frac{1}{(1+\mu_i^2)^{\alpha/2}}\int_0^1 \frac{p^{-\alpha}dp}{\left(1+\frac{\beta_i^2p^2+\gamma_i^2p^4+2\mu_i\beta_ip+2\mu_i\gamma_ip^2+2\beta_i\gamma_ip^3}{1+\mu_i^2}\right)^{\alpha/2}} \\ &= \frac{1}{|\vec{t}_i|^\alpha}\frac{1}{(1+\mu_i^2)^{\alpha/2}}\int_0^1 p^{-\alpha(c_0+c_1p+c_2p^2)+\mathcal{O}(p^3)} \\ &\sim \frac{1}{|\vec{t}_i|^\alpha}\frac{1}{(1+\mu_i^2)^{\alpha/2}}\sum_{n=0}^{10}\frac{c_n}{n-\alpha+1},\end{aligned}$$

   with the coefficients c_n retrieved using automatic differentiation as described in Appendix C. For the second integral in Equation (6), we use the series expansion as

$$\frac{1}{|\vec{t}_i|^\alpha}\frac{1}{(1+\mu_i^2)^{\alpha/2}}\sum_{n=0}^{10}c_n\left(\frac{2\beta_i}{n-\alpha+2}+\frac{3\gamma_i}{n-\alpha+3}\right),$$

   where the coefficients c_n are identical to the ones in the first integral. Since the coefficients decay rapidly, the first ten coefficients are expected to give a sufficiently accurate result.

2. $\vec{x}_j = \vec{x}_{i+1}$.
   Since the first integral in Equation (6) becomes divergent as $\vec{x}_i(p \mapsto 1) = \vec{x}_j$, the results in Case 1 are also applicable in this case, if written in terms of $\tilde{p} = 1 - p$, and using the corresponding parameters $\tilde{\mu}_i = \mu_i + 2\beta_i + 3\gamma_i$, $\tilde{\beta}_i = -\beta_i - 3\gamma_i$ and $\tilde{\gamma}_i = \gamma_i$.

3. $d_x = |\vec{x}_j - \vec{x}_i| > f|\vec{x}_{i+1} - \vec{x}_i|$, with $f = 1/\sqrt{Q}$ and $Q \ll 1$

   The first integral in Equation (6) can not be approximated by the coefficients in Case 1 or 2. Instead, automatic differentiation yields another set of coefficients as follows

$$\int_0^1 \frac{dp}{|\vec{x}_i - \vec{x}_j + p\vec{t}_i + \eta_i(p)\vec{n}_i|^\alpha} = \frac{1}{d_x^\alpha}\int_0^1 \frac{dp}{\left(1+\frac{p^2|\vec{t}_i|^2+\eta_i(p)^2|\vec{n}_i|^2+d_tp+d_n\eta_i(p)}{d_x^2}\right)^{\alpha/2}} \sim \frac{1}{d_x^\alpha}\sum_{n=0}^{10}g_n,$$

   where $d_n = -(\vec{x}_j - \vec{x}_i)\cdot\vec{n}$ and $d_t = -(\vec{x}_j - \vec{x}_i)\cdot\vec{t}$. As in Case 1 and 2, the second integral of Equation (2.2) is approached in a similar manner but with the coefficients g_n.

4. $|\vec{x}_j - \vec{x}_i| \sim 0$.
   The integrand of the first integral in Equation (6) varies steeply for high values of α. The integrals are therefore instead evaluated using Runge-Kutta 4-5, by first rewriting the integrals on differential form,

$$\int_0^1 w(p)dp \to \frac{dY}{dp} = w(p),$$

   with initial condition $Y(p = 0) = 0$. The convergence criteria for Runge-Kutta is set to 10^{-10}.

3.1.3 Normal contribution of the vector field

In accordance with Section 2.4, only the velocity field projected on the normal of the contour curves is of interest since the tangential component does not contribute to the deformation of the curve. In order to compare the convergence speed and accuracy of the interpolation method with the

pseudo-spectral method explained in Section 3.2, in each time step we have removed the tangential component of the velocity field at each node, as given in Equation (4). This is equivalent in terms of curve deformation, and is used for the interpolation method in order to compare the integral evaluation capabilities of the two methods.

Starting from the formulation in Equation (4), where the tangential component has been removed from the vector field, we substitute the interpolation defined in Definition 3.1 to arrive at

$$\frac{d\vec{x}_j}{dt}(t)\bigg|_n = \int_0^1 \frac{(\vec{t}_i + \mu_i\vec{n}_i) + (2\beta_i p + 3\gamma_i p^2)\vec{n}_i - (\vec{t}_j + \mu_j\vec{n}_j)}{|\vec{x}_i - \vec{x}_j + p\vec{t}_i + \eta_i(p)\vec{n}_i|^\alpha} dp,$$

where the tangential component is evaluated at p equals zero. This is simplified to the expression

$$\begin{aligned}\frac{d\vec{x}_j}{dt}(t)\bigg|_n &= \left((\vec{t}_i + \mu_i\vec{n}_i) - (\vec{t}_j + \mu_j\vec{n}_j)\right)\int_0^1 \frac{dp}{|\vec{x}_i - \vec{x}_j + p\vec{t}_i + \eta_i(p)\vec{n}_i|^\alpha} \\ &\quad + \vec{n}_i \frac{(2\beta_i p + 3\gamma_i p^2)dp}{|\vec{x}_i - \vec{x}_j + p\vec{t}_i + \eta_i(p)\vec{n}_i|^\alpha},\end{aligned}$$

which is what has been implemented.

3.2 The pseudo-spectral method

The pseudo-spectral method calculates the velocity field at a node x_j by taking the Riemann sum over coefficients retrieved with the FFT, divided by the magnitude of the difference between the parametrisations to the power of α.

Consider the semi-discrete model where a Riemann sum is used to evaluate the integral in Equation (3),

$$\frac{d\vec{x}_j}{dt}(\gamma, t) = \frac{\theta_k}{2\pi}\sum_{k=1}^{2}\frac{2\pi}{N_k}\sum_{n=1}^{N_k}\frac{\frac{\partial\vec{x}_{n,k}}{\partial\gamma'}(\gamma', t)}{|\vec{x}_{n,k}(\gamma', t) - \vec{x}_j(\gamma, t)|^\alpha}, \tag{7}$$

for each discrete point $\vec{x}_j(\gamma, t)$ on the contour curves. Expanding the path-derivative in the numerator in a basis via the FFT allows for easy differentiation. However, as the curvature of the contour increases, one would need more densely spaced nodes to properly represent the curve. This is equivalent to needing a higher sampling rate for a signal with a higher frequency, according to the Nyquist-Shannon theorem [16]. In order to compute the path-derivative in Equation (7), we need a discrete differentiation operator. The operator chosen was defined by Hou and Li [17] and mitigates the aliasing that occurs in the context of FFT.

Definition 3.2 (First order path-derivative)**.** Consider a differentiation operator of the form

$$\widehat{(D_h\vec{x})}_k = ik\rho(k/N)\hat{\vec{x}}, \quad k = -N/2 + 1, ..., N/2,$$

where N is the number of points, k is the wave number and i is the imaginary identity, denote the Fourier transform of the first order path-derivative of a point x. The anti-aliasing filter, $\rho(k/N)$, is given by

$$\rho(k/N) = e^{-a(|k|/N)^m}.$$

The constants a and m are chosen depending on given cut-off desires. In short notation, the path-derivative is given by

$$\frac{\partial\vec{x}}{\partial\gamma} = \mathcal{F}^{-1}\left\{\widehat{D_h\vec{x}}\right\}.$$

The use of an anti-aliasing filter when computing the path-derivative is important since for any regular function, as the wave number increases, its magnitude decreases. High frequency modes do not contribute significantly to the evolution of the contour, but cause unwanted aliasing that give rise to spatial wave-like clustering of points on the curve. This is also know as a moiré pattern. However, simply removing the high frequency modes abruptly with a rectangular filter will give rise to Gibbs phenomenon and the loss of $\mathcal{L}^2$ energy of the solution. Gibbs phenomenon will cause oscillations, or ringing artifacts in the notion of signal processing, along the path which completely

alters the evolution of it. To avoid aliasing, and to ensure a smooth filter, an anti-aliasing filter has thus been designed according to the filter ρ in Definition 3.2. The filter constants have been set to $a = 36$ and $m = 19$ and are determined by two criteria. First, the cut-off should reach machine epsilon ($\sim 10^{-16}$) for $|k| \to N$. Second, the filter should stay close to 1 for $|k| < \frac{4N}{5}$ and decay rapidly but smoothly. In Figure 2, the chosen anti-aliasing filter is shown for a filter length of $2N + 1 = 1024$ bins.

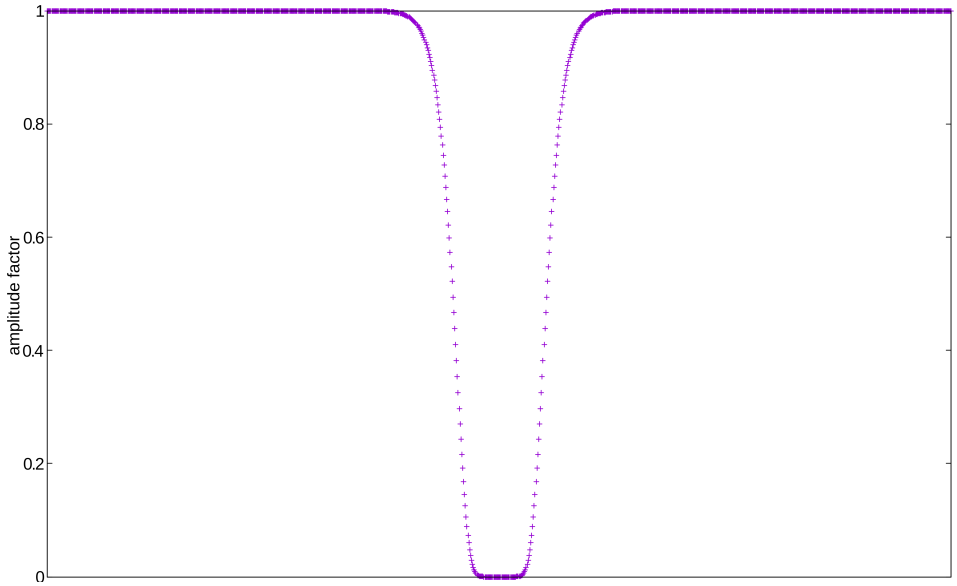

Figure 2: Anti-aliasing filter $\rho(k/N)$ for filter coefficients $a = 36$ and $m = 19$ for a filter length of $2N + 1 = 1024$ bins.

With the differentiation operator as in Definition 3.2, taking aliasing into account, the velocity field at x_j as given by Equation (3) is then approximated by

$$\frac{\partial \vec{x}_j}{\partial t} \approx \sum_{k=1}^{2} \frac{\theta_k}{N_k} \sum_{\substack{n=1 \\ n \neq j}}^{N_k} \frac{\mathcal{F}^{-1}\left\{\widehat{D_h \vec{x}_n}\right\}}{|\vec{x}_j - \vec{x}_n|^\alpha}. \tag{8}$$

Note that for the case $n = j$ the integral approximated by the Riemann sum is divergent, so the sum is changed to skip that case.

3.2.1 Removing the tangential component

The evaluation of the velocity field in Equation (8) is not only suffering from the case where $n = j$, but also from the cases where $n \approx j$ since their contributions are too large to ensure precision. To avoid this, we use the formulation in Section 2.4, where the tangential component is removed. This results in equal deformation of the curve, but the nodes will move differently along the contour. Consequently, the velocity field evaluated at $\vec{x}_j$ is

$$\frac{\partial \vec{x}_j}{\partial t} = \sum_{k=1}^{2} \frac{\theta_k}{N_k} \sum_{n=1}^{N_k} \frac{\mathcal{F}^{-1}\left\{\widehat{D_h \vec{x}_n}\right\} - \mathcal{F}^{-1}\left\{\widehat{D_h \vec{x}_j}\right\}}{|\vec{x}_j - \vec{x}_n|^\alpha}.$$

The singular case only occurs when $\vec{x}_j$ is on the same contour as $\vec{x}_n$, so there is no need to handle the case otherwise. Hence, the velocity field evaluated at $\vec{x}_j$ is in each time step calculated as

$$\frac{\partial \vec{x}_j}{\partial t} = \frac{\theta_1}{N_1} \sum_{n=1}^{N_1} \frac{\mathcal{F}^{-1}\left\{\widehat{D_h \vec{x}_n}\right\} - \mathcal{F}^{-1}\left\{\widehat{D_h \vec{x}_j}\right\}}{|\vec{x}_j - \vec{x}_n|^\alpha} + \frac{\theta_2}{N_2} \sum_{n=1}^{N_2} \frac{\mathcal{F}^{-1}\left\{\widehat{D_h \vec{x}_n}\right\}}{|\vec{x}_j - \vec{x}_n|^\alpha},$$

for $\vec{x}_j$ located on the first patch, and vice-versa if $\vec{x}_j$ is located on the second patch.

3.3 Algorithmic view

This section presents a more technical algorithmic description of how the methods are implemented. The algorithms presented below do only regard the vector field evaluation performed in each step. This is because the time integration used for both methods is the same adaptive Runge-Kutta scheme. However, note that the two different methods may have different overhead costs.

In Algorithm 1 the interpolation technique is given in an algorithmic way to highlight how the velocity vector field is evaluated by the method in each time step. Not described in the algorithm is the relocation of nodes along the contours. This however is performed after each time step based on a new cubic interpolation.

Algorithm 1 Interpolation method

```
1:  function EvaluateVectorFieldInterpolation
2:      Cubic interpolation of the nodes
3:      Determine precision parameter Q
4:      for j from 1 to {number of nodes} do
5:          for i from 1 to {number of nodes} do
6:              if x_i equals x_j then
7:                  Retrieve Taylor coefficients using Automatic Differentiation
8:                  dx_j/dt ← {sum of Taylor coefficients}
9:              else if i equals j − 1 then
10:                 Retrieve Taylor coefficients using Automatic Differentiation
11:                 dx_j/dt ← {sum of Taylor coefficients}
12:             else if Q < tolerance then
13:                 Retrieve Taylor coefficients using Automatic Differentiation
14:                 dx_j/dt ← {sum of Taylor coefficients}
15:             else
16:                 dx_j/dt ← {adaptive Runge-Kutta}
17:         Optional: Project dx_j/dt on the normal
```

To highlight the technicalities of the pseudo-spectral method, Algorithm 2 describes in a similar algorithmic way how the method evaluates the velocity vector field in each time step.

Algorithm 2 Pseudo-spectral method

```
1: function EvaluateVectorFieldPseudoSpectral
2:     for j from 1 to {number of nodes} do
3:         Expand the numerator in a finite set of basis functions (FFT)
4:         Retrieve coefficients using the differentiation operator defined in Definition 3.2
5:         Multiply the retrieved coefficients with the denominator coefficients
6:         dx_j/dt ← {Riemann sum}
7:         Optional: Project dx_j/dt on the normal
```

4 Measurements

Here we describe the setup regarding both software and hardware, as well as how the numerical study was planned and setup.

The computations performed in this study demand high level of rigor as small perturbations quickly and powerfully distort the calculations. To evaluate whether the methods calculates the velocity field correctly up to a certain tolerance, and whether they converge with increasing the number of nodes, we compare the velocity field in the point $(1, 0)$ using a single patch for 2^n nodes each with a lower and upper bound retrieved using validated numerics in accordance with Tucker [22] and $n = 3, ..., 15$. This is done before and after any projection on the normal is done, by using Equation (3) and Equation (4) respectively.

We opted to use two values of α in this study, 0.5 and 0.7. The value 0.5 was chosen since it is exactly in between the 2DEE and SQGE and is hence an interesting value to study. The value 0.7 was chosen as larger values of α put higher demands on the numerical scheme since the local fronts, possibly with singularities or not, appear quicker. For the interpolation method, the precision parameter Q was set so that the adaptive Runge-Kutta scheme was always used instead of the approximative Taylor series expansion in Case 3. This pushes the method to use its most accurate calculations which were compared to the pseudo-spectral method.

Our timings include the cost of setting up the actual interpolation in the interpolation method, since that is considered an integral part of the algorithm. This computational cost of the interpolation is of $\mathcal{O}(N)$ complexity and thus its contribution to the overall cost of the algorithm is minimal. In the pseudo-spectral method the transformation of the sampled contour, which may be viewed as a form of interpolation, is inherent in the scheme via the FFT.

The implementation of the numerical study has been written in C using C99 standard and all simulations have been performed using compiler flags *-O3* and *-march=native*. The choice of using the flag *-march=native* was based on the reason that the executables were run on the same computer as it was compiled. Therefore, it was chosen to use the available instructions on each computer. The implemented methods have been written from scratch along with sub-methods such as the adaptive Runge-Kutta scheme (both in time and space). The only exception is the Fourier transform in the pseudo-spectral method, which was included from the subroutine library FFTW [1] (Fastest Fourier Transform in the West). The program was written to run serially on the computers and no effort has been put in comparing the parallelization properties of the two different methods.

4.1 Hardware specifications

For the numerical simulations performed, two computers have been used with the following specifications and instruction sets:
Computer 1: Intel Core i7-4702MQ 2.20GHz, 16GB RAM, AVX2, Ubuntu 17.10 64-bit, gcc 7.2.0
Computer 2: Intel Core i5-M540 2.53GHz, 8GB RAM, SSE 4.2, Arch Linux 4.13.8-1-ARCH 64-bit, gcc 7.2.0

[1] The subroutine library can be found at https://www.fftw.org.

5 Numerical results

In Section 5.1 and 5.2 we present convergence tests for the two methods evaluating the velocity field with and without the tangential component respectively. In Section 5.3 we present a comparison of computational speed. Section 5.4 shows the evolution of the patches for both methods, to highlight the numerical evidence of singularities on the boundaries.

5.1 Convergence in evaluation of velocity field

In Table 1 we present the lower and upper bounds of the validated numerics of the velocity field in point $(1, 0)$ with tangential component. We also give the intervals for both x- and y-directions, and for $\alpha = \{0.5, 0.7\}$. The exact value of the velocity field can be found inside the interval. Table 2 shows the corresponding results obtained for point $(1, 0)$ with both methods.

Table 1: Lower and upper bounds on the exact value of the velocity field at $\vec{x} = (1, 0)$ for the case with tangential component.

Shape	$\underline{x'}$	$\overline{x'}$	$\underline{y'}$	$\overline{y'}$	α
Circle	0	0	0.39344686629	0.39344686638	0.5
Ellipse	0	0	0.43615367378	0.43615367393	0.5
Other	−0.10069870762	−0.10069870779	−0.23679377190	−0.23679377217	0.5
Circle	0	0	0.84000655004	0.84000655016	0.7
Ellipse	0	0	0.88298932219	0.88298932233	0.7
Other	−0.38512950018	−0.38512950125	−0.44645590600	−0.44645590665	0.7

Table 2: Accuracy results obtained for both methods using 32768 nodes for the case with tangential component.

Shape	Interpolation x	Pseudo x	Interpolation y	Pseudo y	α
Circle	1.238000e-08	-1.745565e-17	-3.934469e-01	-3.419521e-01	0.5
Ellipse	8.737540e-08	7.358480e-16	-4.361537e-01	-3.846590e-01	0.5
Other	-1.006994e-01	-2.686229e-02	-2.367935e-01	-2.039766e-01	0.5
Circle	2.987891e-08	3.911801e-16	-8.400066e-01	-6.037871e-01	0.7
Ellipse	1.979592e-07	8.385220e-16	-8.829893e-01	-6.467701e-01	0.7
Other	-3.851305e-01	-1.022770e-01	-4.464553e-01	-3.207403e-01	0.7

Figures 3-4 show the absolute error of the vector field in point $(1, 0)$ for both methods as a function of the number of computational nodes on the high-curvature shape. This is presented separately for the x- and y-direction.

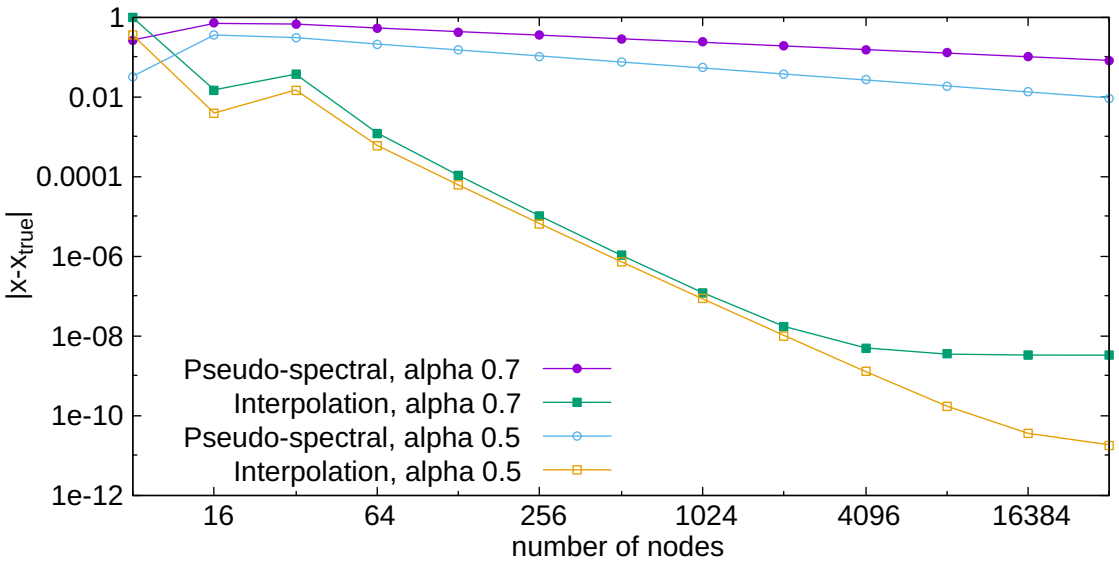

Figure 3: the absolute error in the x-direction as a function of the number of computational nodes in log-log scale. The values for both methods are shown with α 0.7 and 0.5.

As can be seen in both figures, the pseudo-spectral method does not converge to the true solution a timely manner. This is due to the local divergence being removed, instead of being corrected for in some other manner, from the Riemann sum in Equation (8).

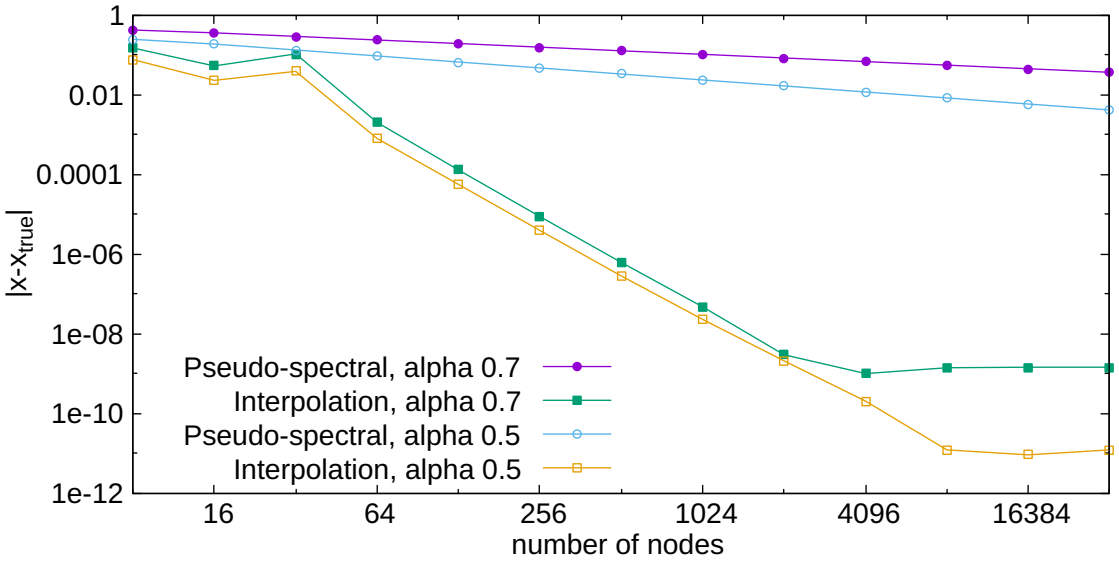

Figure 4: the absolute error in the y-direction as a function of the number of computational nodes in log-log scale. The values for both methods are shown with α 0.7 and 0.5.

5.2 Convergence in evaluation of the normal direction of the velocity field

In Table 3 we present the lower and upper bounds of the interval calculations of the velocity field in point $(1, 0)$ without tangential component. We also give the intervals for both x- and y-directions, and for $\alpha = \{0.5, 0.7\}$. The exact value of the velocity field can be found inside the interval. Table 4 shows the corresponding results obtained for point $(1, 0)$ with both methods.

Table 3: Lower and upper bounds on the exact value of the velocity field at $\vec{x} = (1, 0)$ for the case without tangential component.

Shape	$\underline{x'}$	$\overline{x'}$	$\underline{y'}$	$\overline{y'}$	α
Circle	-0.00000020655	-0.00000020651	0.78689373383	0.78689373381	0.5
Ellipse	-0.00000041297	-0.00000041293	0.59945671983	0.59945671981	0.5
Other	2.28459107523	2.28459107310	0.82333502068	0.82333501942	0.5
Circle	-0.00000061270	-0.00000061265	-0.72000561714	-0.72000561711	0.7
Ellipse	-0.00000122526	-0.00000122522	-0.49694858029	-0.49694858027	0.7
Other	2.38435914865	2.38435913175	0.78442793846	0.78442792963	0.7

Table 4: Accuracy results obtained for both methods using 32768 nodes for the case without tangential component.

Shape	Interpolation x	Pseudo x	Interpolation y	Pseudo y	α
Circle	0.000000000000	0.000000000002	0.786893736288	0.786893732674	0.5
Ellipse	-0.000000000014	0.000000000004	0.599456748240	0.599456718675	0.5
Other	2.284404080870	2.284591073579	0.823251911832	0.823335019673	0.5
Circle	-0.000000000000	0.000000000003	0.720005619058	0.720005614357	0.7
Ellipse	-0.000000000018	0.000000000005	0.496948613464	0.496948577510	0.7
Other	2.384142053755	2.384359137909	0.784331451285	0.784427933298	0.7

Figure 5-6 shows the vector field evaluated in point $(1, 0)$ for both the interpolation method and the pseudo-spectral method as a function of the number of nodes to represent the contour in the high-curvature shape. This is presented separately for the x- and y-direction for the normal-component of the velocity field only.

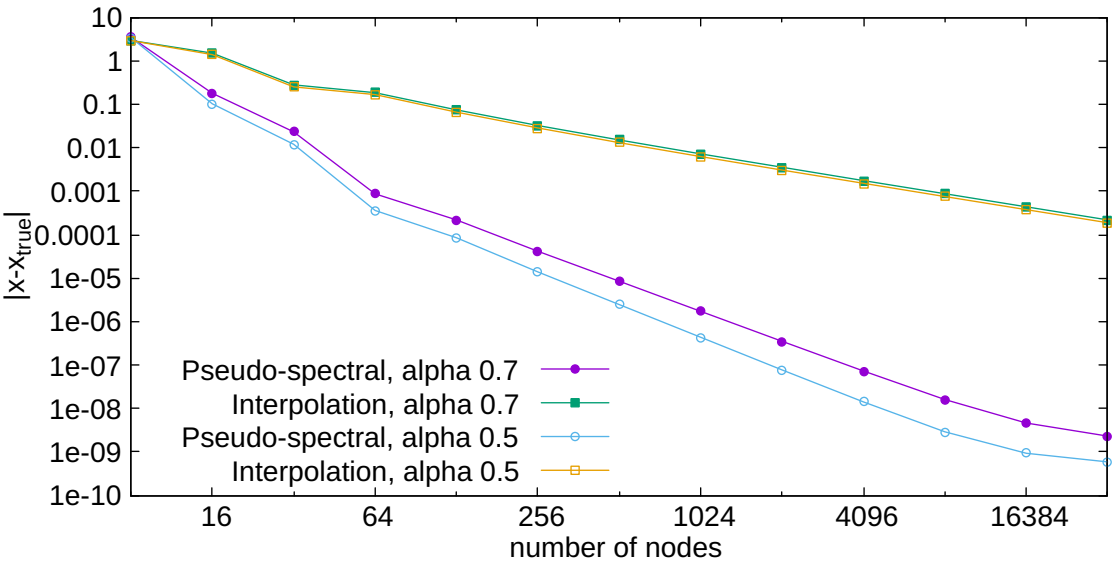

Figure 5: On the high-curvature shape, the normal projected velocity of point $(1, 0)$ in the x-direction as a function of the number of nodes for both methods, with $\alpha = 0.7$. The right figure is a close up, where the true value is enclosed by the solid lines.

As can be seen, the evaluations of the vector field seem to converge to the same value for both methods, although the interpolation method converges more slowly. This is in part due to the design of the test, because of the near analytical nature that the differentiation in the pseudo-spectral method offers. That is, for the pseudo-spectral method the spatial configuration of the nodes does not need to resemble the shape of the curve to have correct derivatives. This is not the case for the interpolation method.

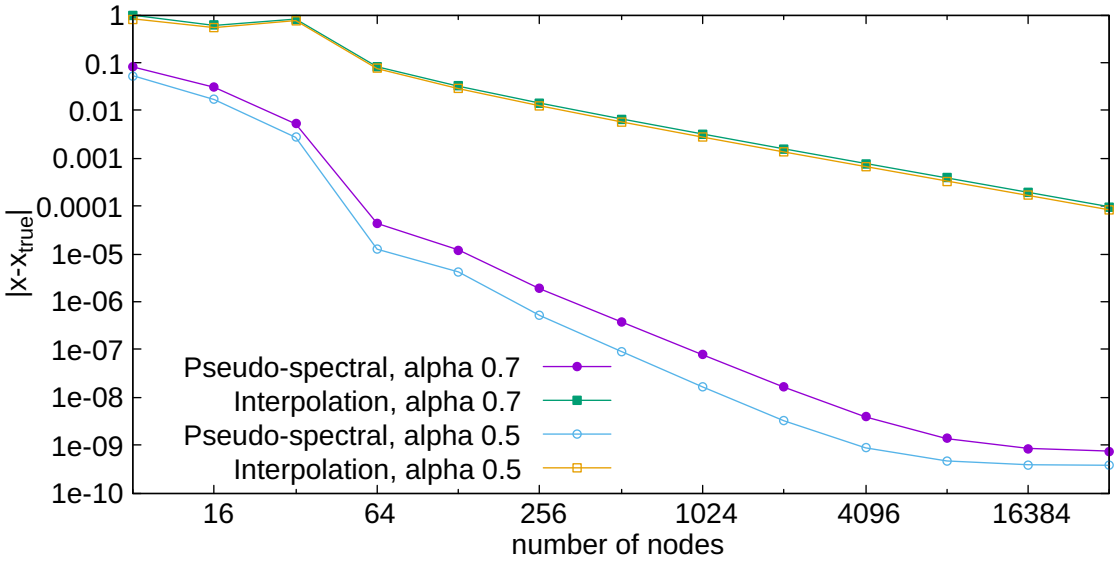

Figure 6: On the high-curvature patch, the normal projected velocity of point $(1, 0)$ in the y-direction as a function of the number of nodes for both methods, with $\alpha = 0.7$. The right figure is a close up, where the true value is enclosed by the solid lines.

5.3 Compuational speed test

In Figure 7 a comparison of the time to evaluate the velocity field acting on the unit circle as a function of the number of nodes is presented in logarithmic scale for both methods.

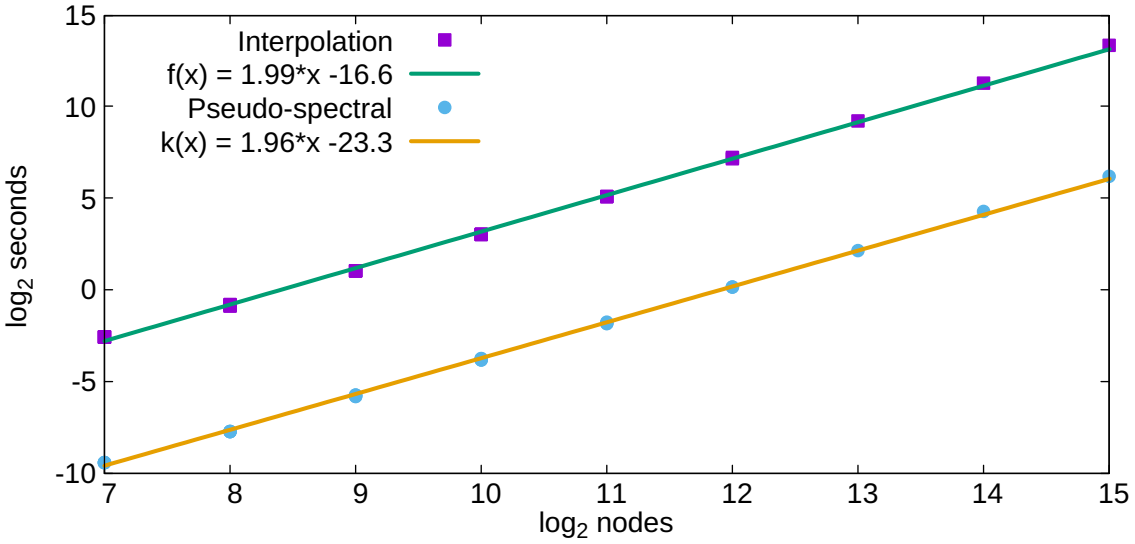

Figure 7: Computational time for evaluating the velocity field on the unit circle versus number of nodes, for both methods in loglog-scale, using $\alpha = 0.7$

In Figure 7 one may see that for 2^{15} number of nodes the time to evaluate the vector field once differs significantly. This is true for both $\alpha = 0.7$ and $\alpha = 0.5$, although only $\alpha = 0.7$ is shown. The pseudo-spectral method computes the integral in $7.25 \cdot 10^1$ seconds while the interpolation method takes $1.05 \cdot 10^4$ seconds.

5.4 Numerical evidence of singularities

Figure 8 illustrates the evolution of two patches with constant vorticity, framed at 0.4 and 0.5 seconds for $\alpha = 0.7$ using the pseudo-spectral method with 1024 points on each patch's contour. This is to compare with Figure 9 where the interpolation method has been used to evolve the patches. In both figures, the vector field is projected on the normal of the contour.

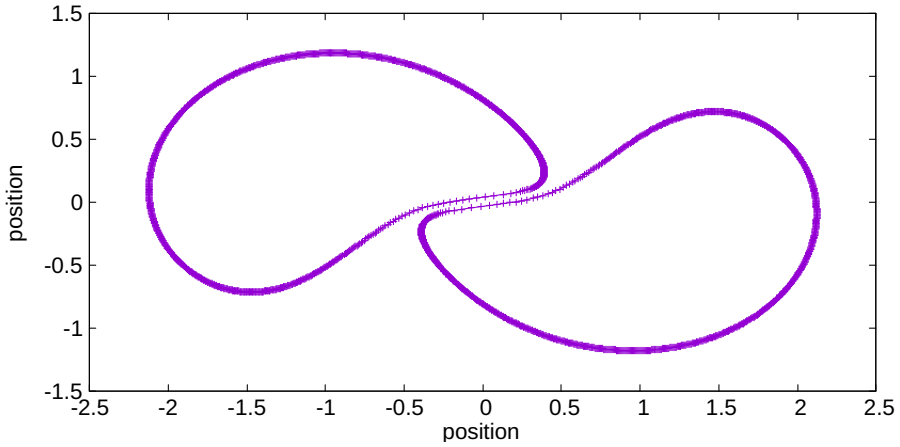

Figure 8: The evolution of two vortex patches with constant vorticity $\theta = -1$ framed at 0.4 seconds using the pseudo-spectral method.

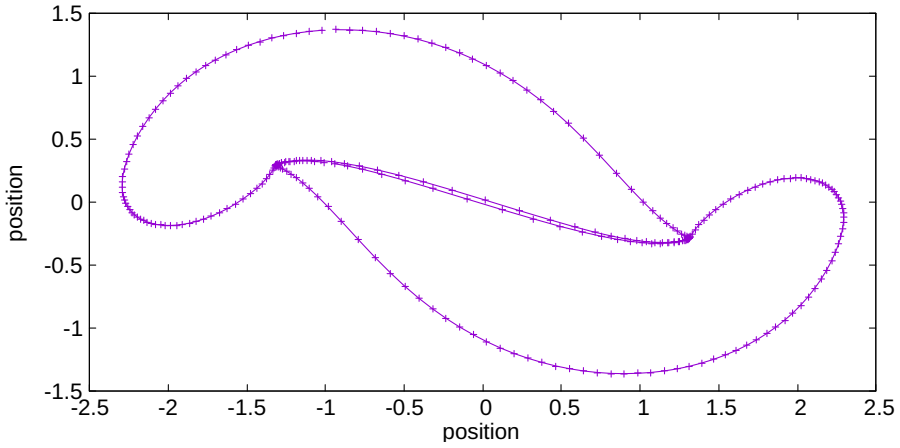

Figure 9: The evolution of two vortex patches with constant vorticity $\theta = -1$ framed at 0.4 and 0.5 seconds using the interpolation method.

The figures presented in [2] could be reproduced with both methods. However, the simulation using the pseudo-spectral method broke down earlier than the simulation using the interpolation method. This was not because the evaluation of the velocity field was incorrect, but instead due to insufficient node handling for the pseudo spectral method. This explains the difference in shape for figure 8 and 9.

6 Discussion

We have implemented and studied two discrete methods for solving an interpolation of 2DEE and SQGE for the vortex patch problem in the α-patches setting. The interpolation method, described in Section 3.1, has previously been used by Cordóba et al. [2] and Mancho [3] to solve the same problem, and the goal of this paper has been to compare this method to a pseudo-spectral method, according to an initial hypotheses that the pseudo-spectral method will be faster and more accurate.

The pseudo-spectral method indeed proved to be much faster. It uses neither any relocation nor interpolation scheme, completely cutting that cost from the computational time. Further, it does not in any step of the spatial integration rely on a Runge-Kutta scheme, which is one of the most costly steps in the interpolation method. In the end, the methods scale similarly but the constant in the pseudo-spectral method is a lot smaller, as can be seen in Figure 7. It should be noted however, that for the pseudo-spectral method to work efficiently, the number of points used to represent the curve is constrained to be a power of two. This is due to the nature of the FFT.

In regards to accuracy, the pseudo-spectral method proved to be better than the interpolation method. Since the patches are simply bounded and connected, they are periodic in their parameterizations. The Riemann sum will be exponentially accurate when evaluating the velocity field using the pseudo-spectral method. In the interpolation method on the other hand, the integrals in Case 1, 2 and 3, approximates as a Taylor series of tenth order, and for the RK-45 case it will be fifth order accurate. However, the interpolation scheme is of third order, yielding a total third order accurate method.

The approximation in the pseudo-spectral method that omits divergent part of the summation when calculating the velocity of a node hinders good convergence and as seen in the results in Section 5.1 the convergence rate for the velocity is not satisfactory for the case where the tangential component is kept. The divergence theorem ensures that the normal-projected vector field is the only interesting part of it in respect to curve deformation. When this was applied, the pseudo-spectral method proved better than the interpolation method, as can be seen in Table 4.

Further, a troubling aspect of the pseudo-spectral method, and one that is not easily handled, is that aliasing occurs for high frequency modes. As the curve gets acute, more frequency information is needed for a good contour representation. This is similar to a problem where a signal has high frequency components but where the sampling frequency is too low. In our case, it is troublesome since it effectively hinders a good contour representation but also distorts the evaluation of the velocity field. A finite Fourier series can not exactly represent a sharp local front or a discontinuity and Gibbs phenomenon will always be present. This is intrinsic to the method and the contours may contain important high frequency modes which need to be considered while at the same time cope with aliasing. To take care of this issue, the differentiation operator in 3.2 used an exponential anti-aliasing filter which worked to some extent. This is still a sub-optimal solution however, since it affects the true representation of the curve by possibly suppressing relevant frequencies. The anti-aliasing filter has not been tuned numerically, which should ideally be done adaptively depending on the curvature of the contour.

One area where the interpolation method surpasses the pseudo-spectral method for this problem, is node handling. As singularities start to form, more and more nodes are needed to manage the increasingly sharp corners. In the interpolation method this is handled by the relocation of nodes-part of the method. For the pseudo-spectral method however, there is no good way to handle those possible sharp corners as there is no easy way to add nodes without the interpolation step.

A possible solution to the above mentioned issues, both regarding the interpolation method and the pseudo-spectral method, could be to combine the two methods. If one could find a good way to merge the node relocation of the interpolation method with the FFT's efficiency constraint of using a power of two number of nodes, that would solve the issue of handling sharp corners, while keeping the advantage in efficiency provided by the pseudo-spectral method. This would require introducing the interpolation step to the pseudo-spectral method, but it would still remain considerably faster than the interpolation method, since the computation of the spatial integral is where

the two methods differ the most in their efficiency.

There are several reasons to choose a pseudo-spectral method in the computation of the velocity field in this problem. A patch is bounded and connected and therefore intrinsically periodic in its parametrization. The periodicity of the contour curves ensures that there are no discontinuous jumps that cause Gibbs phenomenon. It is also computationally cheap to retrieve coefficients of the derivative of the signal which appear in every evaluation of the vector field at each point's position. The properties of the FFT in terms of computational complexity are the main motivations for investigating this method.

7 Conclusions

We found that the pseudo-spectral method compared to the interpolation method indeed showed better performance in terms of speed. Normally, one would expect a trade-off between speed and accuracy, but in this case the pseudo-spectral method has a better convergence rate in evaluating the velocity field of the contour than the interpolation method when projecting the velocity field on the contour normal. As mentioned, when opting for the pseudo-spectral method, one should always use the projection on the normal to ensure better convergence. Concluding, the pseudo-spectral method outperforms the interpolation method in the vector field evaluation both regarding accuracy and speed. However, the adaptive node handling of the interpolation method is crucial to better represent the contour as singularities may appear. This means that the pseudo-spectral method could be used as a substitute of the interpolation method in the sense of using it as a mere integral evaluation tool, seeking the fruits of the convergence rate and speed properties while still comprising the adaptive node handling of the interpolation method. At this stage the methods are not implemented in such a way as to make it feasible to switch between the two while calculating.

The importance of better understanding how non-regular solutions to Navier-Stokes like equations behave can not be overstated. Many of todays improvements and advances in popular technical fields may shadow the demands of progress in fundamental research. However, it should be clear that such research broadens the map of depicting what is observed, and discontinuous settings in the field of dynamical systems is such a field. Partial differential equations are great tools for problems where the dynamics are smooth and well-behaving but the sharp local fronts in for instance the vortex patch problem requires mathematical tools properly designed for such settings. While this study does not nearly prove global existence of regular solutions in the Navier-Stokes equations, it does give insight in how similar vorticity equations behave. It would already be an interesting problem to study on its own, how an interpolation between 2DEE and SQGE behave, but the connection to other physical problems is undoubtedly the most interesting part of this problem.

A Adaptive Runge-Kutta

The time integration as well as the integrals in the fourth case of algorithm used by Mancho (3.1.2) may be solved by an adaptive Runge-Kutta scheme.

Runge-Kutta 4-5 (RK-45) is a method combining RK-4 and RK-5, with order 4 and 5 respectively. RK-45 works by calculating any given step at least twice, once by each RK-4 and once by RK-5. If the difference between the two methods becomes sufficiently small, below a given tolerance, $\varepsilon > 0$, it is considered to have converged on an approximate result and a step is taken with RK-5. If $|\text{RK-4}-\text{RK-5}| > \varepsilon$, the parameter controlling the step-size is reduced and the methods are recalculated until sufficient accuracy is achieved. A formula for RK methods of arbitrary order is defined by

Definition A.1 (Runge-Kutta (RK))**.** Let $\frac{dy}{dx} = f(x, y)$ and $y(x_0) = y_0$. Then for a given step-size, $h > 0$, the embedded coefficients are given by

$$f_\kappa = f(x + \alpha_\kappa h, y + h \sum_{l=0}^{\kappa-1} \beta_{\kappa l} f_l), \quad \kappa \in \mathbb{N} \backslash [0, p+1].$$

Then one step is given by

$$y_{n+1}(x, y; h) = \sum_{\kappa=0}^{p+1} c_\kappa f_\kappa(x, y; h),$$

where the order of accuracy is $p + 1$.

The coefficients used for the RK methods of order 4 and 5 are the coefficients given by Fehlberg [15].

If a convergence criteria for RK-45 has been achieved, one may in accordance with Bulirsch and Stoer [14], use

$$h_{next} = \Lambda h \sqrt[4]{\frac{\varepsilon}{|\hat{y}_{n+1} - y_{n+1}|}}$$

to find an appropriate next step-size. Here $\hat{y}_{n+1}$ is the fourth order method and y_{n+1} is the fifth order method, and ε is the convergence criteria. $\Lambda = 0.9$ is an adjustment factor, as given by Bulirsch and Stoer [14].

B The Fast Fourier Transform

In many physical time-dependent processes, it is of interest to describe the process in the frequency spectrum. The implementation of the fast Fourier transform-algorithm (FFT) by Cooley and Tukey[9] in 1965 (although others, starting with Gauss in 1805[], had discovered and implemented it earlier) severely reduced computation time from the previous $\mathcal{O}(N^2)$ to $\mathcal{O}(N \log_2 N)$. This broadened the possibilities of using the Fourier transform in real-time applications often used in e.g. signal processing.

Effectively, any complex, continuous time-series $x(t)$ can be described in its frequency domain by the (continuous) transformation

$$X(\theta) := \int_{-\infty}^{\infty} x(t) e^{i\theta t}, \tag{9}$$

where the angular frequency is defined as $\theta := 2\pi f$. The Fourier transform has many useful properties not mentioned in this study. In the case of discrete data, as in most practical cases, the discrete Fourier transform defined in Definition B.1 needs to be used.

Definition B.1 (Discrete Fourier Transform (DFT))**.** Let x_n be N discrete data points. The discrete Fourier transform is then given by

$$X_n := \sum_{k=0}^{N-1} W_N^{nk} x_k, \tag{10}$$

with $W_N^n := e^{2\pi i n/N}$ for $n = 0, 1, ...N-1$ and $i = \sqrt{-1}$ being the imaginary unit. W_N^n are called the Nth roots of unity since they are vertices of a regular polygon on the unit circle in the complex plane with the first vertex in $(1, 0)$ and thus $(W_N^n)^N = 1$ for all n. All powers of the roots of unity are periodic with N and therefore a multiplication with W_N^{nk} is equal to a rotation of angle $\frac{2\pi}{N}k$ in the complex plane.

The complexity of computing the discrete Fourier transform is of order $\mathcal{O}(N^2)$ since it involves a multiplication of the vector x_k by the (n, k)th power of the constant W. This can be severely reduced by applying the FFT. Starting with Definition B.1, the sum in Equation (10) can be split into two sums for the even and the odd indeces respectively

$$\begin{aligned}\sum_{k=0}^{N-1} W_N^{nk} x_k &= \sum_{\text{k even}}^{N-1} W_N^{nk} x_k + \sum_{\text{k odd}}^{N-1} W_N^{nk} x_k \\ &= \sum_{\text{m=0}}^{N/2-1} W_N^{2mn} x_{2m} + \sum_{\text{m=0}}^{N/2-1} W_N^{(2m+1)n} x_{2m+1}\end{aligned}$$

for $n = 0, ..., N-1$. Using the periodicity of DFT we notice that $W_N^2 = W_{N/2}$. By substituting this relation into Equation (10), the DFT can be expressed as

$$\begin{aligned}X_n &= \sum_{m=0}^{N/2-1} f_1(m) W_{N/2}^{nm} + W_{N/2}^n \sum_{m=0}^{N/2-1} f_2(m) W_{N/2}^{nm} \\ &= F_1(n) + W_N^n F_2(n), \quad n = 0, ..., N-1.\end{aligned}$$

By recursively repeating splitting the sums into even and odd parts, one can effectively compute the DFT with a computational complexity of $\mathcal{O}(N \log_2 N)$ instead of $\mathcal{O}(N^2)$.

C Automatic Differentiation

Automatic differentiation is a recursive procedure used to numerically compute the derivative of a function composed by sums, products, quotients and elementary functions at a given point. It differs from symbolic differentiation and finite-differences schemes; it is more efficiently implemented than symbolic differentiation and does not introduce round-off errors in the arithmetics.

Automatic differentiation can thus effectively be used to recursively retrieve the Taylor coefficients of an intricate function using already retrieved coefficients of a simpler function. The set of rules to procedure automatic differentiation was presented by Jorba and Zou [11]. Two of these rules are given but not proved in Proposition C.1 and in Example C.1, an example of computing Taylor coefficients of an involved function using automatic differentiation is given.

Proposition C.1 (Automatic Differentiation)**.** Let $\alpha \in \mathbb{R}\backslash\{0\}$, and $b, c : \mathbb{R} \to \mathbb{R}$ be functions of class $\mathcal{C}^n$. Then the following holds:

1. If $a(t) = \frac{b(t)}{c(t)}$, then
$$a^{[n]}(t) = \frac{1}{c^{[0]}(t)} \left[b^{[n]}(t) - \sum_{j=1}^{n} c^{[j]}(t) a^{[n-j]}(t) \right].$$
2. If $a(t) = b(t)^\alpha$, then
$$a^{[n]}(t) = \frac{1}{n b^{[0]}(t)} \sum_{j=0}^{n-1} \left((n\alpha - j(\alpha+1)) \right) b^{[n-j]}(t) a^{[j]}(t).$$

Example C.1 (Computation of Taylor coefficients)**.** *Let $a(t) = \frac{b(t)}{c(t)}$ with $b(t) = 1$ and $c(t) = 1 + 2t + 3t^2$ being functions of class $\mathcal{C}^n$. The Taylor coefficients to $b(t)$ and $c(t)$ are trivially given by*
$$b^{[n]}(t) = \begin{cases} 1, & \text{if } n = 0, \\ 0, & \text{if } n \neq 0 \end{cases},$$
and
$$c^{[n]}(t) = \begin{cases} n+1, & \text{if } n = 0, 1, 2, \\ 0, & \text{if } n \neq 0, 1, 2 \end{cases}.$$
We can then find the Taylor coefficients to the function $a(t)$. Using Rule 1 in Proposition C.1, the first three Taylor coefficients of $a(t)$ are given by
$$\begin{aligned} a^{[0]}(t) &= 1, \\ a^{[1]}(t) &= -2, \\ a^{[2]}(t) &= 1. \end{aligned}$$

Automatic differentiation is especially useful for integrating ordinary differential equations with not too complicated vector field. In essence, it can be used to have a very high-order solver.

References

[1] https://www.claymath.org/sites/default/files/navierstokes.pdf (Visited 2018-01-17)

[2] Córdoba D., Fontelos M.A., Mancho A.M., Rodrigo, J.L. 2005. *Evidence of singularities for a family of contour dynamics equations.* PNAS, (Vol. 102, no. 17): pp. 5949-5952.

[3] Mancho, A.M. 2015. *Numerical studies on the self-similar collapse of the α-patches problem.* Communications in Nonlinear Science and Numerical Simulation, (Vol. 26): pp. 152-166.

[4] Chemin J.Y. 1993. *Persistance de structures géométriques dans les fluides in-compressibles bidimensionnels.* Annales Scientifiques de l'Ecole Normale Superiéure. (Vol. 26, no. 4): pp. 1-16.

[5] Bertozzi A.L., Constantin P. 1993. *Global regularity for vortex patches.* Communications in Mathematical Physics (Vol. 152, no. 1): pp. 19-28.

[6] Scott R.K., Dritschel D.G. 2014. *Numerical Simulation of a Self-Similar Cascade of Filament Instabilities in the Surface Quasigeostrophic System.* Physical Review Letters. (Vol. 112, no. 144505).

[7] Constantin P. 2007. *On the Euler equation of incompressible fluids.* Bull Am Math Soc (Vol. 44, no. 4): pp. 603-21.

[8] Majda A, Bertozzi A. 2002. *Vorticity and incompressible flow.* Cambridge Texts Applied Mathematics (Vol 27).

[9] Cooley J.W., Tukey J.W. 1965. *An algorithm for the machine calculation of complex Fourier series.* Math. Comput. Mathematics of Computation (Vol. 19): pp. 297–301.

[10] http://www.cec.uchile.cl/cinetica/pcordero/MC_libros/NumericalRecipesinC.pdf (Visited: 2017-10-20)

[11] Jorba À., Zou M. 2004. *A Software Package for the Numerical Integration of ODEs by Means of High-Order Taylor Methods.* Experimental Mathematics, (Vol. 14, no. 1): pp. 99-116.

[12] Dritschel D.G. 1989. *Contour dynamics and contour surgery: Numerical algorithms for extended, high-resolution modelling of vortex dynamics in two-dimensional, inviscid, incompressible flows.* Computer Physics Reports, (Vol. 10): pp. 77-146.

[13] Zabusky N, Hughes M.H., Roberts K.V. 1979. *Contour dynamics for the Euler equations in two dimenstions.* Journal of Computational Physics (Vol. 135, no. 2): pp. 96-106.

[14] Bulirsch R, Stoer J. *Introduction to numerical analysis.*

[15] Fehlberg E. 1970. *Klassische Runge-Kutta-Formeln vierter und niedrigerer Ordnung mit Schrittweiten-Kontrolle und ihre Anwendung auf Wärmeleitungsprobleme.* Computing, (Vol. 6, no 1-2): pp. 66-71.

[16] Shannon C.E. 1949. *Communication in the presence of noise.* Proceedings of the Institute of Radio Engineers. (Vol. 37, no. 1): pp. 10–21.

[17] Hou T.Y., Li R. 2007. *Computing Nearly Singular Solutions Using Pseudo-Spectral Methods.* Journal of Computational Physics. (Vol. 226, no. 1): pp. 379-397

[18] Kelliher J. 2012, March. *Bounded vorticity, bounded velocity (Serfati) solutions to 2D Euler equations.* [online]. Carnegie Mellon University. http://www.math.cmu.edu/CNA/Seminars/Documents/KelliherJames.pdf (Retrieved: 2017-12-20).

[19] Yudovich V.I. 1963. *Non-stationary flows of an ideal incompressible fluid.* USSR Computational Mathematics and Mathematical Physics. (Vol. 3, no. 6): pp. 1407-1456.

[20] Majda A. 1986. *Vorticity and the mathematical theory of incompressible fluid flow.* Commun. Pure Appl. Math. (Vol. 39): pp. 5187–5220.

[21] Rodrigo J.L. 2005. *On the evolution of sharp fronts for the Quasi-geostrophic equation.* Commun. Pure Appl. Math. (Vol. 58, no. 6): 821-866.

[22] Tucker W. 2011 *Validated Numerics A Short Introduction to Rigorous Computations.*

[23] S. Y. Lou, M. Jia, X. Y. Tang, F. Huang. 2007. *Vortices, circumfluence, symmetry groups, and Darboux transformations of the (2+1)-dimensional Euler equation.* Phys. Rev. (Vol. 75, no. 5): pp. 056318.

Bibliography

[24] http://www.cs.cmu.edu/afs/andrew/scs/cs/15-463/2001/pub/www/notes/fourier/fourier.pdf (Visited: 2017-12-01)

[25] http://www.cmlab.csie.ntu.edu.tw/cml/dsp/training/coding/transform/fft.html (Visited: 2017-12-01)

[26] Arfken, G. 1985. *Gibbs Phenomenon. §14.5 in Mathematical Methods for Physicists, 3rd ed.* Orlando, FL: Academic Press, pp. 783-787.