A volume-preserving Characteristic Mapping Method for the 2D incompressible Euler equations

William Holman-Bissegger

Department of Mathematics and Statistics

McGill University

Montreal, Quebec, Canada

June 2024

A thesis submitted to McGill University in partial fulfillment of the requirements of the degree of Master of Science

Abstract

We consider the Characteristic Mapping Method (CMM) for the 2D incompressible Euler equations [45]. This semi-Lagrangian scheme exploits the semigroup decomposition of the inverse flow map, permitting arbitrary spatial resolution and faithfully capturing multi-scale dynamics. In this thesis we decouple the time integration in the CMM framework from space. This decoupling enables structure-preserving integration at the level of the submaps. It will also enable Characteristic Mapping methods in domains with complex boundaries. Symplectic integrators are employed for exact volume preservation. There is nevertheless a final composition step which cannot be performed exactly conservatively at this stage. Instead, this composition is performed to a very high order of conservation, in particular using Fourier upsampling. The method is applied to the 2D incompressible Euler equations on the torus. Four-modes and vortex merge experiments are performed with a GPU implementation. Tests confirm that the final composition is essentially conservative.

Abrégé

Nous considérons la méthode d'Application Caractéristique (AC) pour les équations d'Euler incompressibles en deux dimensions [45]. Cette méthode semi-lagrangienne exploite la décomposition du semi-groupe de l'application de flot inverse, permettant une résolution spatiale arbitraire et capturant fidèlement la dynamique à plusieurs échelles. Dans cette thèse, nous découplons l'intégration temporelle de celle de l'espace, dans le cadre de l'AC. Ce découplage permet une intégration préservant la structure au niveau des sous-applications; il permettra également de mettre en œuvre des méthodes d'AC dans des domaines ayant des frontières complexes. Des intégrateurs symplectiques sont employés pour la conservation exacte du volume. Il reste néanmoins une étape finale de composition ne pouvant être réalisée de manière tout à fait conservatrice à ce stade. Nous réalisons cette composition plutôt avec un ordre de conservation très élevé, en utilisant un suréchantillonnage de Fourier. Nous appliquons la méthode aux équations d'Euler incompressibles sur le tore. Des expériences de quatre modes et de fusion de vortex sont réalisées avec une implémentation pour cartes graphiques. Des tests confirment que la composition finale est essentiellement conservatrice.

Acknowledgements

I would first like to express my profound gratitude to my supervisor, Professor Jean-Christophe Nave. I have been working under his supervision for the past three and a half years, beginning in the middle of my undergraduate degree. This has been an incredibly rewarding experience. His tutelage has enabled me to explore many different areas of numerical analysis. It has also brought me tremendous growth, both professionally and personally. I deeply appreciate his patience, his generosity with his time, and his kindness.

I would also like to thank my family: my mother and father, Christopher and Alyssa. This project could not have been completed without their unconditional love and support. Finally, I would like to thank some dear friends: Sophia Howard, Mikaela Fasold, Nathaniel Giroux, Jacob Reznikov, Samuel Zeitler, Negar Matin, Maya Marsonia, Jessie Meanwell, Shereen Elaidi, and many others. I sincerely appreciate the encouragement and companionship they have brought throughout my academic journey.

This work was partially funded through the ANR and NSERC grants of Prof. Nave.

Contribution

The entire thesis was written solely by the author. Prof. Nave and some colleagues of the author provided some assistance with regards to the structure and phrasing. Sections 1.1, 1.2, 2.2 and 3.1.2, as well as Appendix A, contain a review of background material adapted from various sources. The rest of this thesis is the original work of the author.

List of Figures

List of Tables

Contents

1 Introduction

1.1 Background

The incompressible Euler equations model the motion of an ideal fluid in the absence of viscosity. In this ideal flow, fluid particles move freely, while neither crossing paths nor exhibiting friction-like interactions. For a classical derivation of the equations, see [31]. By the incompressibility condition, the resulting flow map is at all times a volume-preserving diffeomorphism ([26]). In other words, the flow traces a path through $\mathrm{SDiff}\,(M)$, the infinite-dimensional Lie group of volume-preserving diffeomorphisms ([2]) of the base manifold M. In fact, a remarkable result is that flows of the Euler equations are precisely the geodesics through the identity in $\mathrm{SDiff}\,(M)$, when the latter is equipped with the right-invariant L^2 metric (see e.g. [2], [31], [13]). This mirrors some finite-dimensional mechanics; for instance, Eulerian motions of rigid bodies in $\mathbb{R}^3$ are geodesics in the Lie group of rotations SO(3) ([2]). The incompressible Euler equations are part of the larger class of Euler-Arnold equations, which are similar geodesic flows that arise when the group of diffeomorphisms and its Riemannian metric are varied ([31]).

Closely related to the incompressible Euler equations are of course the incompressible Navier-Stokes equations. These can be seen as the incompressible Euler equations with an added diffusion term, modelling nonzero viscosity. In certain specific cases, such as on compact manifolds without boundary, it is known that solutions of the incompressible Navier-Stokes equations converge to solutions of the incompressible Euler equations, in the limit of zero viscosity ([13]). However in general, particularly in the presence of boundaries, convergence is not observed. It is known that near boundaries, the viscosity, no matter how small, creates effects that cannot be described by the incompressible Euler equations ([3]).

For a sufficiently-regular initial vorticity in L^∞, classical solutions of the 2D incompressible Euler equations exist for all time ([3]). This is in contrast to the 3-dimensional case, where the question of whether or not solutions develop singularities in finite time is open, even for very regular initial conditions ([26]). However, even in the 2-dimensional case, solutions rapidly become complex. It is known that in 2 dimensions, the L^∞ norm of the gradient of the vorticity is bounded by a double exponential in time ([47]). There are initial vorticities on the disk that are known to reach this bound ([23]). Moreover on the torus, given an arbitrary time interval, there is an initial vorticity whose gradient will grow exponentially in this time interval ([1]). One explanation of this fast-growing gradient is the energy cascade in the incompressible Euler equations. Due to the nonlinearity in the equations, energy can over time move from the large to the fine spatial scales ([15]). In the incompressible Euler equations, due to the absence of viscosity, these fine scale features do not dissipate. This is in contrast to the Navier-Stokes equations, where the viscosity can cause a diffusion of the energies at small scales ([15]). This generation of exponentially-fine scales over time causes difficulties for numerical simulations. Accurately resolving an exponentially-growing gradient requires an exponentially-large number of grid points in time.

Numerical algorithms for fluids, and related problems, are typically classified on a spectrum from Eulerian to Lagrangian, based on the nature of the discretization. In an Eulerian description, one examines fluid quantities in a fixed reference frame; in a Lagrangian description, one measures quantities as they evolve along a fluid trajectory ([35]). There are advantages and drawbacks of discretizing according to either description. In a Lagrangian formulation, one is not constrained to a Courant-Friedrichs-Lewy (CFL) condition ([39]). However, it is in general highly nontrivial to convert quantities in Lagrangian frames to their Eulerian versions, which is usually required for time-stepping ([39]). For these reasons, some schemes operate in a semi-Lagrangian manner, following the approach of the CIR method [12]. In a semi-Lagrangian method, it is typical to consider quantities in an Eulerian frame, but to perform the time-stepping in a Lagrangian manner. This is commonly implemented by integrating the characteristic ODEs backward over one time step, and then performing a spatial interpolation ([39]).

Classical Eulerian algorithms include the Maker and Cell method [20], where the velocity and pressure are discretized via finite differences on staggered grids, and spectral methods [8], where collocation or Galerkin approximations are applied to the PDE in the velocity-pressure or vorticity-stream formulations. For a comparison of these Eulerian methods in 3 dimensions, see [17]. An example of a purely Lagrangian scheme is the vortex blob method [4]. As for a semi-Lagrangian method, there is the Cauchy-Lagrangian method [34]. In all of the above schemes, since computations are carried out on fixed spatial grids, there are incurred spatial truncation errors. These truncations can create undesirable resonance phenomena; see an analysis in [36] for the Galerkin truncation case, and [34] in the case of the Cauchy-Lagrangian method. Crucially, these truncations also create artificial diffusion. Thus, traditional methods in fact simulate a small amount of viscosity. This viscosity may lead to solutions that do not qualitatively resemble ideal, inviscid Euler flow.

A unique approach to the numerical simulation of the incompressible Euler equations is that of the Characteristic Mapping Method (CMM), first introduced in [30]. The CMM is a general framework for advection-type problems. We review the CMM framework and its application to the 2D incompressible Euler equations in the upcoming Section 1.2. In brief, the CMM considers the inverse flow map. This inverse flow map is integrated in time, typically in a semi-Lagrangian fashion. The map can be decomposed into a sequence of submaps, which recovers the whole map by composition. Hence, a large deformation may be split into a sequence of smaller deformations. This enables a dynamic, arbitrarily-fine spatial resolution in the solution, while only needing to compute each submap on a fixed, coarse grid. Finally, advected quantities are computed as a pullback by the inverse flow map, and are never directly advected on a grid. This avoids the spatial truncations and artificial diffusion mentioned previously. The CMM framework thus enables totally inviscid numerical solutions.

The CMM has been applied to linear transport problems, in Euclidean spaces ([30]) and on the sphere ([41]). The incompressible Euler equations, in vorticity-stream form, can be written as a self-advecting transport problem. The CMM has thus been applied to solve the incompressible Euler equations in 2- and 3-dimensional Euclidean space (see [45] and [46], respectively), and on the sphere ([40]). In a similar vein, the CMM has been applied to solve the Vlasov-Poisson equation, a model of particles in a self-consistent electric field ([24]). This work is currently being extended to other flows of diffeomorphisms. The CMM has seen some further applications, such as investigating the formation of singularities in the 2D Euler equations with non-smooth initial data ([6]), and as a particle management framework in the context of surface evolution ([44]). One shortcoming of current Characteristic Mapping methods, however, is that they are not conservative. This is a known drawback that is typical of semi-Lagrangian schemes, due to the spatial interpolation at each time step ([39]).

Exact structure preservation has become an increasingly important property for numerical schemes. That is, designing numerical methods that exactly preserve, at the discrete level, an important structure of the continuous problem. Many standard methods are not conservative for general problems. For instance, it is known that all Runge-Kutta methods do not preserve arbitrary polynomial invariants ([10]). However, it is typically possible to construct conservative methods for specific cases of interest. For example, one can construct Runge-Kutta integrators that exactly preserve a given polynomial Hamiltonian ([10]); discrete gradient methods are available for preserving first integrals ([33]); one can construct integrators that exactly satisfy a discrete variational principle ([28]); and there are the Lie-Poisson integrators ([48]) and symplectic integrators ([18]) that preserve the Lie-Poisson and symplectic structures, respectively, of the space. There is moreover the Discrete Multiplier Method [42], with which one can construct integrators that exactly preserve a number of conserved quantities simultaneously.

We would like to design a Characteristic Mapping Method for the 2D incompressible Euler equations which is exactly incompressible. That is, where the inverse flow map is at each time step exactly volume-preserving. We will see that there is an obstruction to exact conservation (namely, a spatial projection at each time step) in the time integration method typically used in the CMM. There is also a final composition step, which we will not be able to perform exactly conservatively. The contribution of this thesis is thus the following. We reformulate the Characteristic Mapping framework so that the time integration of each submap is decoupled from space. This enables exactly conservative integration of the submaps. It moreover will enable time integration that respects spatial boundaries. Finally, we design a method in which the final composition step can be done to a very high order of conservation.

1.2 Review of the Characteristic Mapping Method for the 2D incompressible Euler equations

We first review the Characteristic Mapping framework, and the Characteristic Mapping Method for the 2D incompressible Euler equations [45]. The method we develop is based heavily on the latter.

In this thesis, the space we are working in is the flat torus $T^2 = \mathbb{R}^2/\mathbb{Z}^2$. We identify functions and vector fields on T^2 with their periodic analogues on $\mathbb{R}^2$. We also identify maps $\boldsymbol{X} : T^2 \to T^2$ that are homotopic to the identity with maps of the form $\boldsymbol{X}(\boldsymbol{x}) = \boldsymbol{x} + \boldsymbol{\eta}(\boldsymbol{x})$ for a periodic function $\boldsymbol{\eta} : T^2 \to \mathbb{R}^2$. Finally, we use the following standard notation: for functions $\boldsymbol{u}, \boldsymbol{v} : \mathbb{R}^2 \to \mathbb{R}^2$, for example, $\boldsymbol{u} \cdot \nabla \boldsymbol{v}$ denotes the space derivative

$$\boldsymbol{u} \cdot \nabla \boldsymbol{v} = \begin{pmatrix} \boldsymbol{u} \cdot \nabla v_1 \\ \boldsymbol{u} \cdot \nabla v_2 \end{pmatrix} = (\nabla \boldsymbol{v})\boldsymbol{u}.$$

1.2.1 Flows, advection and semigroup decomposition

Let $\boldsymbol{u} : T^2 \times [t_0, \infty) \to \mathbb{R}^2$ be a vector field, possibly time-dependent, which is C^1 in space and time. Throughout this thesis we assume that $\operatorname{div} \boldsymbol{u} \equiv 0$, though note that this is not necessary in the full CMM framework ([44]). We let $\boldsymbol{\Phi}_{[t_0,t]} : T^2 \to T^2$ denote the flow of $\boldsymbol{u}$ from a time t_0 to a later time t. That is, $\boldsymbol{\Phi}_{[t_0,t]}$ solves

$$\begin{cases} \partial_t \boldsymbol{\Phi}_{[t_0,t]} = \boldsymbol{u}(\cdot,t) \circ \boldsymbol{\Phi}_{[t_0,t]} \\ \boldsymbol{\Phi}_{[t_0,t_0]}(\boldsymbol{x}) = \boldsymbol{x}. \end{cases}$$

We then define the inverse flow map $\boldsymbol{X}_{[t,t_0]} := \boldsymbol{\Phi}^{-1}_{[t_0,t]}$. We refer to $\boldsymbol{\Phi}$ as the forward map, and to $\boldsymbol{X}$ as the backward map. Note the inversion of the subscripts; $\boldsymbol{\Phi}_{[t_0,t]}$ denotes the flow from t_0 up to t, and $\boldsymbol{X}_{[t,t_0]}$ denotes the flow from t back to t_0.

The backward map has the interpretation of bringing particles that have been flowing under the velocity $\boldsymbol{u}$, back to their original locations. That is, if $\boldsymbol{y}_{\boldsymbol{x}}(t)$ is an integral curve of $\boldsymbol{u}$ indexed by its initial position, i.e.

$$\frac{d}{dt}\boldsymbol{y}_{\boldsymbol{x}}(t) = \boldsymbol{u}(\boldsymbol{y}_{\boldsymbol{x}}(t),t), \qquad \boldsymbol{y}_{\boldsymbol{x}}(t_0) = \boldsymbol{x}, \tag{1.1}$$

then $\boldsymbol{X}_{[t,t_0]}(\boldsymbol{y}_{\boldsymbol{x}}(t)) = \boldsymbol{x}$ for all $t \geqslant t_0$. In a sense, $\boldsymbol{X}$ captures the deformation of the underlying space generated by the velocity $\boldsymbol{u}$. This is made more precise below.

A fundamental property of flows is that for arbitrary $t_0 < t_1 < t_2$, we have

$$\boldsymbol{\Phi}_{[t_0,t_2]} = \boldsymbol{\Phi}_{[t_1,t_2]} \circ \boldsymbol{\Phi}_{[t_0,t_1]}.$$

For the backward map, this fact becomes the following: for arbitrary $t_0 < \cdots < t_n$, we can decompose the backward map as

$$\boldsymbol{X}_{[t_n,t_0]} = \boldsymbol{X}_{[t_1,t_0]} \circ \cdots \circ \boldsymbol{X}_{[t_n,t_{n-1}]}. \tag{1.2}$$

This is referred to as the semigroup decomposition of the backward map.

Further properties of the backward map are show in Proposition 1.1.

Proposition 1.1. *The following are properties of the backward flow map. The first holds for arbitrary $\boldsymbol{u}$, while the second requires* $\operatorname{div}\boldsymbol{u} \equiv 0$.

1. *The backward map solves the advection equation*

$$\begin{cases} \partial_t \boldsymbol{X}_{[t,t_0]} + \boldsymbol{u}\cdot\nabla\boldsymbol{X}_{[t,t_0]} = 0 \\ \boldsymbol{X}_{[t_0,t_0]} = \mathrm{id}. \end{cases} \tag{1.3}$$

2. *Suppose* $\operatorname{div}\boldsymbol{u} \equiv 0$, *and let* $\omega \in C^1(T^2 \times [t_0,\infty))$ *solve the conservation law*

$$\begin{cases} \partial_t \omega + \operatorname{div}(\omega\boldsymbol{u}) = 0 \\ \omega(\cdot,t_0) = \omega_0. \end{cases} \tag{1.4}$$

Then ω is constant along trajectories of $\boldsymbol{u}$. In particular, the solution at any time t is given by the pullback of the initial condition ω_0 by the backward map:

$$\omega(\cdot, t) = \omega_0 \circ \boldsymbol{X}_{[t,t_0]}. \tag{1.5}$$

Thus, the backward map acts as a solution operator for quantities transported along the flow. In the geometric formulation of the CMM, the backward map acts as a solution operator for general Lie-advected quantities; see e.g. [40]. Note further that the pullback (1.5) allows for more general advected quantities than the conservation law (1.4), such as when ω_0 is not smooth. In particular, we can take ω_0 to be the indicator function of a set, as is done in [30]. Then $\omega_0 \circ \boldsymbol{X}_{[t,t_0]}$ tracks the transport of the set along the velocity $\boldsymbol{u}$. This makes precise the statement that $\boldsymbol{X}$ tracks the deformation of the space.

The proof of Proposition 1.1 is a direct application of the method of characteristics; hence the name, Characteristic Mapping Method.

Proof of Proposition 1.1. 1. Let $\boldsymbol{X}(\boldsymbol{x}, t)$ solve the advection equation (1.3). We apply the method of characteristics, in the formulation of Evans [14]. Consider each component X_i. We augment the usual variables with time dependence,

$$\boldsymbol{p} = \begin{pmatrix} \nabla X_i \\ \partial_t X_i \end{pmatrix}, \qquad \boldsymbol{y} = \begin{pmatrix} \boldsymbol{x} \\ t \end{pmatrix}.$$

Each component X_i solves the equation

$$F(\boldsymbol{p}, z, \boldsymbol{y}) := \boldsymbol{p} \cdot \begin{pmatrix} \boldsymbol{u}(\boldsymbol{y}) \\ 1 \end{pmatrix} = 0. \tag{1.6}$$

Let z denote the value of X_i along characteristics. Then the characteristic equations read

$$\begin{aligned} \frac{dz}{ds} &= \nabla_{\boldsymbol{p}} F \cdot \boldsymbol{p} = \begin{pmatrix} \boldsymbol{u}(\boldsymbol{y}) \\ 1 \end{pmatrix} \cdot \boldsymbol{p} = 0 \\ \frac{d\boldsymbol{y}}{ds} &= \nabla_{\boldsymbol{p}} F = \begin{pmatrix} \boldsymbol{u}(\boldsymbol{y}) \\ 1 \end{pmatrix}, \end{aligned}$$

where we used Equation (1.6) to simplify $\frac{dz}{ds}$. Thus X_i is constant along characteristics. Moreover we see that the characteristic curves are precisely integral curves of $\boldsymbol{u}$. Thus, again letting $\boldsymbol{y}_{\boldsymbol{x}}(t)$ be an integral curve of $\boldsymbol{u}$ parametrized by its initial position (defined in Equation (1.1)), we get that $\boldsymbol{X}(\boldsymbol{y}_{\boldsymbol{x}}(t), t)$ is constant in t, so

$$\boldsymbol{X}(\boldsymbol{y}_{\boldsymbol{x}}(t), t) = \boldsymbol{X}(\boldsymbol{y}_{\boldsymbol{x}}(t_0), t_0) = \boldsymbol{X}(\boldsymbol{x}, t_0) = \boldsymbol{x}.$$

Hence $\boldsymbol{X}$ is precisely the backward map.

2. Noting that $\operatorname{div} \boldsymbol{u} = 0$ implies $\operatorname{div}(\omega \boldsymbol{u}) = \boldsymbol{u} \cdot \nabla \omega$, the conservation law for ω reduces to an advection equation. By the same method of characteristics applied above, ω is constant on integral curves of $\boldsymbol{u}$. Since $(\boldsymbol{x}, t)$ and $(\boldsymbol{X}_{[t,t_0]}(\boldsymbol{x}), t_0)$ lie on the same characteristic, it follows that

$$\omega(\boldsymbol{x}, t) = \omega(\boldsymbol{X}_{[t,t_0]}(\boldsymbol{x}), t_0) = \omega_0(\boldsymbol{X}_{[t,t_0]}(\boldsymbol{x})).$$

□

The semigroup decomposition (1.2) and the facts of Proposition 1.1 are the heart of the CMM. Over some long time interval $[t_0, t_n]$, the deformation of the space generated by a velocity may be quite complex. This is especially true in the case of the Euler equations, where the velocity develops fine details over time. However, in this formulation, we can split $[t_0, t_n]$ into shorter time intervals $t_0 < t_1 < \cdots < t_n$, and consider the deformation in each subinterval $[t_{i-1}, t_i]$. These are captured by the maps $\boldsymbol{X}_{[t_i,t_{i-1}]}$, which we call the submaps. We can then represent advected quantities at t_n by combining the pullback (1.5) with the semigroup decomposition (1.2):

$$\omega(\cdot, t_n) = \omega_0 \circ \boldsymbol{X}_{[t_1,t_0]} \circ \cdots \circ \boldsymbol{X}_{[t_n,t_{n-1}]}. \tag{1.7}$$

In the CMM, we do not store ω on a grid; rather, we store all of the submaps $\boldsymbol{X}_{[t_i,t_{i-1}]}$, and use (1.7) to sample ω, using some interpolation scheme. This has a number of computational advantages. In principle, the deformation of the space in each subinterval $[t_{i-1}, t_i]$ is much smaller and more manageable than the total deformation. Thus, each submap $\boldsymbol{X}_{[t_i,t_{i-1}]}$ can be computed on a relatively coarse grid. The total deformation and its complexities are however recovered by the composition (1.2).

In the case of the incompressible Euler equations, ω will be the vorticity (defined below). It is known that for certain initial vorticities, the quantity $\nabla\omega$ can grow exponentially in time. Properly representing ω on a grid, then, requires a grid with a number of points that is also exponential in time. However, by (1.7) we have

$$\nabla\omega = \nabla\omega_0 \cdot \nabla\boldsymbol{X}_{[t_1,t_0]} \cdot \cdots \cdot \nabla\boldsymbol{X}_{[t_n,t_{n-1}]}$$

(where we have omitted the location at which each gradient is evaluated). We can thus still have $\nabla\omega$ growing exponentially in time, while only needing each $\nabla \boldsymbol{X}_{[t_i,t_{i-1}]}$ to grow linearly in time.

In practice, we are also able to dynamically pick the times $t_0 < \cdots < t_n$ at which we split the backward map. When integrating the backward map, when some criterion is reached, the current submap is saved and we initialize a new submap at the identity. This process is called remapping. The criterion is usually a spatial resolution criterion, based on how well the current submap represents the deformation on the grid on which it is computed. Thus, even if the submaps are all computed on a fixed grid, by remapping appropriately, we are able to achieve an arbitrarily-high spatial resolution.

Moreover, computing ω by (1.7), as opposed to advecting ω on a fixed grid, does not incur the usual dissipation errors that occur when undersampling or otherwise truncating in space. While there may be dissipation errors incurred in the computation of the submaps, when we compute $\omega = \omega_0 \circ \boldsymbol{X}$, the dissipation error in $\boldsymbol{X}$ shows up as evaluating ω_0 in a slightly different location. Thus, dissipative error in $\boldsymbol{X}$ is turned into advective error in ω. This is the key in computing the vorticity in the case of the incompressible Euler equations in a non-dissipative manner, which is crucial to the exactly inviscid flow.

1.2.2 2D incompressible Euler equations

The incompressible Euler equations in two dimensions ([26]) are

$$\begin{cases} \partial_t \boldsymbol{u} + \boldsymbol{u}\cdot\nabla\boldsymbol{u} = -\nabla p \\ \operatorname{div}\boldsymbol{u} = 0, \end{cases} \tag{1.8}$$

where $\boldsymbol{u} : T^2 \times [t_0,\infty) \to \mathbb{R}^2$ is the fluid velocity and $p : T^2 \times [t_0,\infty) \to \mathbb{R}$ is the pressure. We define two important quantities related to the velocity $\boldsymbol{u}$. The vorticity $\omega : T^2 \times [t_0,\infty) \to \mathbb{R}$ is defined as the 2D curl of the velocity field,

$$\omega = \operatorname{curl}\boldsymbol{u} := \partial_{x_1}\boldsymbol{u}_2 - \partial_{x_2}\boldsymbol{u}_1.$$

In this thesis, we assume that $\boldsymbol{u}$ has mean zero: $\int_{T^2} u_i = 0$, $i = 1,2$. Then, since $\operatorname{div}\boldsymbol{u} = 0$, by a Helmholtz decomposition on the torus there exists a function $\psi : T^2 \times [t_0,\infty) \to \mathbb{R}$, called the stream function, whose perpendicular gradient is $\boldsymbol{u}$:

$$\boldsymbol{u} = \nabla^\perp\psi := \begin{pmatrix} \partial_{x_2}\psi \\ -\partial_{x_1}\psi \end{pmatrix}.$$

Thus the fluid motion is always tangent to contour lines of ψ, and ψ acts as a non-autonomous Hamiltonian for $\boldsymbol{u}$. Combining these definitions, we can relate all three quantities by

$$-\Delta\psi = \operatorname{curl}\boldsymbol{u} = \omega. \tag{1.9}$$

This equality is referred to as the Biot-Savart law. Note that our convention for ψ differs from that of [26] by a sign. Moreover we always take ψ to have mean zero. Thus since we are on the torus and $\operatorname{div}\boldsymbol{u} = 0$, the quantities ψ, $\boldsymbol{u}$ and ω are all uniquely defined in terms of one another. This is referred to as the vorticity-stream formulation of the incompressible Euler equations ([26]).

Taking the curl of (1.8) gives a conservation law for the vorticity,

$$\partial_t\omega + \operatorname{div}(\omega\boldsymbol{u}) = 0,$$

i.e. ω is an advected quantity of the fluid, constant on flow lines. In particular, writing $\boldsymbol{X}$ for the backward map of the flow generated by $\boldsymbol{u}$, by Proposition 1.1 we can write ω at any time t as the pullback of the initial condition ω_0 by $\boldsymbol{X}$,

$$\omega(\cdot, t) = \omega_0 \circ \boldsymbol{X}_{[t,t_0]}.$$

Thus, computing $\boldsymbol{u}$ from the vorticity ω by inverting the Biot-Savart law (1.9), and by (1.3), we can write a "self-advecting" evolution equation for $\boldsymbol{X}$,

$$\begin{cases} \partial_t\boldsymbol{X}_{[t,t_0]} + \boldsymbol{u}\cdot\nabla\boldsymbol{X}_{[t,t_0]} = 0 \\ \boldsymbol{u} = -\nabla^{\perp}\Delta^{-1}(\omega_0 \circ \boldsymbol{X}_{[t,t_0]}). \end{cases} \tag{1.10}$$

This is referred to as the advection-vorticity coupling ([45]), and forms a full solution to the incompressible Euler equations.

The solution proposed in [45] to solve the incompressible Euler equations is to compute the backward map $\boldsymbol{X}$ by the advection-vorticity coupling (1.10). The map and the velocity are successively updated in time using the coupling. At some time t_n, the map $\boldsymbol{X}_{[t_n,t_0]}$ is updated as follows. First, the velocity is integrated backward in time to compute the backward map over one time step, $\boldsymbol{X}_{[t_{n+1},t_n]}$. Then, $\boldsymbol{X}_{[t_{n+1},t_0]}$ is computed by interpolating $\boldsymbol{X}_{[t_n,t_0]}$ in space, and using the fact

$$\boldsymbol{X}_{[t_{n+1},t_0]} = \boldsymbol{X}_{[t_n,t_0]} \circ \boldsymbol{X}_{[t_{n+1},t_n]}.$$

In previous iterations of the CMM, this has typically been implemented via the Gradient-Augmented Level Sets method ([32]). There, both the map and its gradient are advected in a coupled manner. The solution of the Euler equations described above enjoys the benefits mentioned in Section 1.2.1; namely, arbitrary spatial resolution, and dissipation-free computation of the vorticity.

1.3 Thesis outline

The goal of this work is to construct an essentially volume-preserving Characteristic Mapping Method. The main facets are to decouple the time integration from space, so as to use exactly-conservative integration whenever possible, and to use high-order spatial methods when exact conservation is not possible. As such, this thesis is organized as follows. In Section 2, we examine the current obstruction to conservation in the CMM, and reformulate the CMM to enable space-decoupled time integration. We then discuss properties and examples of symplectic integrators, the conservative integrators of interest for this project. We finally perform a finite difference test of volume preservation with an analytically-known vector field. In Section 3, we repeat the previous test but computing the Jacobian in Fourier space. We illustrate the issue of spatial resolution, and motivate using the semigroup decomposition of the backward map. We illustrate the difficulty in constructing an exactly volume-preserving interpolation, and lay out an alternate strategy for interpolation that is conservative to a very high order. Finally, we describe a specific velocity interpolant, whose construction is rather technical but which is nonetheless required for the essentially-conservative composition strategy. Sections 2 and 3 make essentially no mention of the incompressible Euler equations, and thus apply to any Characteristic Mapping Method.

In Section 4 we combine constructions from the previous two sections, and design a volume-preserving CMM specifically for the incompressible Euler equations on the torus. We describe in detail the time integration process. We derive an error estimate, and perform convergence tests. In Section 5, we perform numerical experiments using a GPU implementation of the algorithm proposed in the previous section. We perform various four-modes and vortex merge tests. Throughout, we examine the conservation properties of the scheme. Finally, in Appendix A, we review the properties and construction of periodic cardinal B-spline interpolation, which will be used for all spatial interpolation.

2 Volume-preserving time integration in the CMM

A classical result is that the flow of a divergence-free vector field is volume-preserving ([18], VI.9). In other words, let $\boldsymbol{u} : \mathbb{R}^d \times [t_0, \infty) \to \mathbb{R}^d$ be a C^1 vector field, and let $\boldsymbol{\Phi}_{[t_0,t]} : \mathbb{R}^d \to \mathbb{R}^d$ be its flow. Then

$$\operatorname{div} \boldsymbol{u} \equiv 0 \quad \forall t \geqslant t_0 \implies \det \nabla_{[t_0,t]} \boldsymbol{\Phi} \equiv 1 \quad \forall t \geqslant t_0.$$

This volume preservation of course extends to the backward map $\boldsymbol{X}$. This fact is elegantly phrased in terms of the Lie group $\mathrm{SDiff}(\mathbb{R}^d)$ of volume-preserving diffeomorphisms: the Lie algebra of $\mathrm{SDiff}(\mathbb{R}^d)$ is the algebra $\mathfrak{X}_0(\mathbb{R}^d)$ of divergence-free vector fields. This fact extends to an arbitrary Riemannian manifold, after fixing a volume form.

We would like to numerically compute the backward map $\boldsymbol{X}$ in such a way that the discretized map is always exactly volume-preserving. By this we mean that our numerical backward map is always the projection onto the grid of a volume-preserving diffeomorphism. This is a desirable property of a numerical scheme for a variety of qualitative and quantitative reasons. The condition $\det \nabla \boldsymbol{X} \equiv 1$ implies that $\boldsymbol{X}$ is exactly a diffeomorphism. Computing an advected quantity ω via the pullback

$$\omega(\cdot, t_n) = \omega_0 \circ \boldsymbol{X}_{[t_n,t_0]},$$

then, is just a rearrangement of the initial condition ω_0. Computing ω in this manner incurs no numerical dissipation, and fine details present in the flow are not lost to diffusion. Moreover, having a flow $\boldsymbol{X}$ with $\det \nabla \boldsymbol{X} \equiv 1$ is an equivalent definition of incompressible flow ([26]). Thus, while the time integration may not be exact, exact volume preservation ensures that the numerical solution is still qualitatively an incompressible flow. Finally, $\det \nabla \boldsymbol{X} \equiv 1$ ensures exact preservation of conserved integrals, by the change of variables formula.

In this section, we see that the time integration method that is typically used in the CMM is incompatible with conservation. We reformulate the time integration of the backward map to enable conservation, by decoupling the time integration from space. While we focus on volume preservation, this reformulation enables the use of an arbitrary conservative integrator. We then discuss the integrators of interest for volume preservation, which are the symplectic integrators. Finally, we illustrate the volume preservation of a symplectic flow via finite differences.

2.1 Space-time decoupled integration of the backward map

2.1.1 The backward map

The main object of interest in this thesis is the backward flow map $\boldsymbol{X}$, defined in Section 1.2.1. The picture we have of $\boldsymbol{X}$, especially when it comes to discretization, is as follows. We think of $\boldsymbol{X}_{[t_n,t_0]} : T^2 \to T^2$ as a map from the space at time t_n to the space at t_0, which assigns to trajectories of the flow at t_n their initial locations at t_0. In particular, we can have $\boldsymbol{X}_{[t_n,t_0]}$ live on a regular grid of points at t_n. The images of these regular grid points, however, following a nontrivial flow, will not land on a regular grid at t_0. This is illustrated in Figure 2.1.

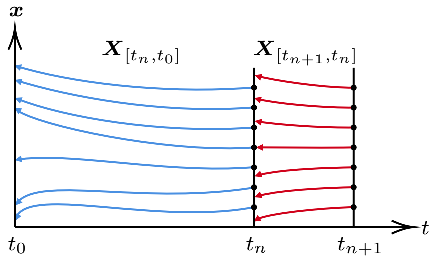

$\boldsymbol{x}$
$\boldsymbol{X}_{[t_n,t_0]}$
$\boldsymbol{X}_{[t_{n+1},t_n]}$
t
t_0
t_n
t_{n+1}

Figure 2.1: Illustration of backward maps.

The fact that $\boldsymbol{X}_{[t_n,t_0]}$ can live on a regular grid at t_n is a major advantage. While $\boldsymbol{X}$ is itself a Lagrangian quantity, it acts as a solution operator for advected Eulerian quantities, see Section 1.2.1. In particular for the incompressible Euler equations, $\boldsymbol{X}$ allows for the computation of the vorticity ω at time t_n, on the same grid as $\boldsymbol{X}$. Since this grid is regular, it is then relatively straightforward to compute the velocity $\boldsymbol{u}$ at t_n: we can solve the Poisson problem in the Biot-Savart law (1.9) with e.g. a Fourier spectral method.

In general, it is highly nontrivial to compute the velocity from Lagrangian quantities such as the forward flow. This issue is precisely due to the fact that grids become deformed, and consequently irregular, under a flow. A projection back onto some regular grid is usually required to compute the velocity, which can be quite complicated and costly. This is a common shortcoming of Lagrangian and semi-Lagrangian methods ([39]). This problem is however circumvented by considering the backward map. In a sense, $\boldsymbol{X}$ maps from a Lagrangian frame to an Eulerian frame. Moreover, the backward map represents a fundamentally Lagrangian quantity, and can be computed in an Eulerian, Lagrangian or semi-Lagrangian manner. The Lagrangian approaches present a considerable computational advantage. In particular, they circumvent the Courant-Friedrichs-Lewy (CFL) condition, a significant upper bound on the time step.

On the other hand, a disadvantage is that a backward flow is less straightforward to compute than a forward flow. To see this, note that we could compute $\boldsymbol{X}_{[t_n,t_0]}$ by setting $\boldsymbol{X}_{[t_n,t_0]}(\boldsymbol{x}) = \boldsymbol{z}_{\boldsymbol{x}}(t_n)$, where $\boldsymbol{z}_{\boldsymbol{x}}(s)$ solves

$$\frac{d}{ds}\boldsymbol{z}_{\boldsymbol{x}}(s) = -\boldsymbol{u}(\boldsymbol{z}_{\boldsymbol{x}}(s), t_0 + t_n - s), \qquad \boldsymbol{z}_{\boldsymbol{x}}(t_0) = \boldsymbol{x}. \tag{2.1}$$

However, this requires knowledge of the velocity in the entire interval $[t_0, t_n]$. In the case of the Euler equations, for instance, we simply do not know the velocity past the current time step; the later velocities depend on the backward map by the advection-vorticity coupling (1.10).

The solution to step $\boldsymbol{X}_{[t_n,t_0]}$ forward in time in the CMM [45] is to compute the backward flow over one time step $[t_n, t_{n+1}]$ using (2.1), which gives $\boldsymbol{X}_{[t_{n+1},t_n]}$, and then computing $\boldsymbol{X}_{[t_{n+1},t_0]}$ by the composition

$$\boldsymbol{X}_{[t_{n+1},t_0]} = \boldsymbol{X}_{[t_n,t_0]} \circ \boldsymbol{X}_{[t_{n+1},t_n]}. \tag{2.2}$$

At t_n we have $\boldsymbol{X}_{[t_n,t_0]}$ on regular grid points; the same is true with $\boldsymbol{X}_{[t_{n+1},t_n]}$ at t_{n+1}. The flow $\boldsymbol{X}_{[t_{n+1},t_n]}$, however, in general does not land at grid points at t_n. This is illustrated in Figure 2.1. Thus to compute (2.2), $\boldsymbol{X}_{[t_n,t_0]}$ is interpolated in space, and this interpolant is evaluated at $\boldsymbol{X}_{[t_{n+1},t_n]}$.

However, we are currently unaware of how to construct a volume-preserving interpolant, or more generally how to construct a conservative interpolant for a given conserved quantity. We go into some more details of these difficulties in Section 3.2.2. As such, when left with the usual interpolation techniques, even if $\boldsymbol{X}_{[t_{n+1},t_n]}$ were integrated in a conservative manner, the composition (2.2) would destroy conservation.

What we have here is a coupling between space and time; at each time step, a spatial projection (namely, the interpolant of $\boldsymbol{X}_{[t_n,t_0]}$) is applied to the data. This coupling is undesirable from a conservation point of view. To integrate each submap conservatively, we would ideally leave the data free of spatial projections, and leave all matters of conservation to the integrator.

2.1.2 An ODE for the backward map

To integrate conservatively, for the reasons discussed above, we would like the time integration of the backward map to be completely decoupled from space. Thus, we would like an ODE for the backward map. It happens that one is available; it is given in Proposition 2.1.

Proposition 2.1. *Let $\boldsymbol{u} : \mathbb{R}^2 \times [t_0, \infty) \to \mathbb{R}^2$ be a C^1, divergence-free vector field. Let $\boldsymbol{\Phi}_{[t_0,t]} : \mathbb{R}^2 \to \mathbb{R}^2$ be its flow, and $\boldsymbol{X}_{[t,t_0]} = \boldsymbol{\Phi}_{[t_0,t]}^{-1}$ its inverse flow. Then $\boldsymbol{X}$ solves the ODE*

$$\partial_t \boldsymbol{X}_{[t,t_0]} = \boldsymbol{v}(\cdot, t) \circ \boldsymbol{X}_{[t,t_0]},$$

with the vector field $\boldsymbol{v} : \mathbb{R}^2 \times [t_0, \infty) \to \mathbb{R}^2$ defined as

$$\boldsymbol{v} = -\operatorname{adj}(\nabla \boldsymbol{\Phi}_{[t_0,t]})(\boldsymbol{u}(\cdot, t) \circ \boldsymbol{\Phi}_{[t_0,t]}), \tag{2.3}$$

where adj *denotes the adjunct matrix. In particular, if $\boldsymbol{u} = \nabla^\perp \psi$ for some scalar function ψ, then this can be written more succinctly as*

$$\boldsymbol{v} = -\nabla^\perp(\psi(\cdot, t) \circ \boldsymbol{\Phi}_{[t_0,t]}). \tag{2.4}$$

Proof. For ease of notation, we suppress all time dependence. The backward map $\boldsymbol{X} = \boldsymbol{X}_{[t,t_0]}$ solves an advection equation (1.3), so we have

$$\partial_t \boldsymbol{X} = -\boldsymbol{u} \cdot \nabla \boldsymbol{X} = -(\nabla \boldsymbol{X})\boldsymbol{u}.$$

But $\boldsymbol{X} = \boldsymbol{\Phi}^{-1}$, and $\det \nabla \boldsymbol{X} \equiv 1$, so by the inverse function theorem we can write

$$\nabla \boldsymbol{X} = (\nabla \boldsymbol{\Phi})^{-1} \circ \boldsymbol{X} = \operatorname{adj}(\nabla \boldsymbol{\Phi}) \circ \boldsymbol{X},$$

where we also used $\det \nabla \boldsymbol{\Phi} \equiv 1$. Using $\boldsymbol{\Phi} \circ \boldsymbol{X} = \mathrm{id}$ we can write

$$\boldsymbol{u} = \boldsymbol{u} \circ \boldsymbol{\Phi} \circ \boldsymbol{X}$$

and so

$$\partial_t \boldsymbol{X} = -\left[\operatorname{adj}(\nabla \boldsymbol{\Phi})(\boldsymbol{u} \circ \boldsymbol{\Phi})\right] \circ \boldsymbol{X}, \tag{2.5}$$

which gives (2.3)

Now suppose that $\boldsymbol{u} = \nabla^\perp \psi$. Let $\boldsymbol{v}$ be defined by (2.4), then

$$\begin{aligned} \boldsymbol{v} = -\nabla^\perp(\psi \circ \boldsymbol{\Phi}) &= -J\nabla(\psi \circ \boldsymbol{\Phi}) \\ &= -J(\nabla \boldsymbol{\Phi})^T(\nabla \psi \circ \boldsymbol{\Phi}), \end{aligned}$$

where the matrix

$$J = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}.$$

A straightforward matrix computation shows $JA^T = \operatorname{adj}(A)J$ for all $A \in \mathbb{R}^{2\times 2}$, so

$$\boldsymbol{v} = -\operatorname{adj}(\nabla\boldsymbol{\Phi})J(\nabla\psi)|_{\boldsymbol{\Phi}} = -\operatorname{adj}(\nabla\boldsymbol{\Phi})\nabla^{\perp}\psi|_{\boldsymbol{\Phi}} = -\operatorname{adj}(\nabla\boldsymbol{\Phi})(\boldsymbol{u}\circ\boldsymbol{\Phi}),$$

which is the same vector field as in (2.5).

$\square$

In the rest of this thesis we will use Proposition 2.1 for the time integration of the backward map, and will denote by $\boldsymbol{v}$ its velocity. We always assume that $\boldsymbol{u} = \nabla^{\perp}\psi$ for some ψ, so we'll exclusively use the second definition (2.4) of $\boldsymbol{v}$. One can check from (2.3) that if $\boldsymbol{u}$ is divergence-free, then so is $\boldsymbol{v}$. Moreover by (2.4) if $\boldsymbol{u}$ is Hamiltonian, then so is $\boldsymbol{v}$. Thus the velocity of the backward map $\boldsymbol{v}$ has the same properties as the forward velocity (for the purpose of volume preservation).

In practice, to use the velocity $\boldsymbol{v}$, we now have to also compute the forward flow $\boldsymbol{\Phi}$ of $\boldsymbol{u}$ (to use Equation (2.4)). We see this as an acceptable overhead, however. The reason for this is that the conservative integrators we have in mind are more often than not implicit. Their computation thus involves many iterations of the velocity, which will itself be interpolated; the whole process is rather expensive. On the other hand, we will see that we are not at all interested in a conservative integration of the forward flow $\boldsymbol{\Phi}$, and so $\boldsymbol{\Phi}$ can be computed by some relatively inexpensive explicit scheme.

Finally, we note that there is an interesting duality between the forward and backward velocities. Indeed, assume that $\boldsymbol{u} = \nabla^{\perp}\psi$ and write $\Psi = -\psi(\cdot,t)\circ\boldsymbol{\Phi}_{[t_0,t]}$ so that $\boldsymbol{v} = \nabla^{\perp}\Psi$ as in Equation (2.4). Then, repeating the same process but with the backward map and backward Hamiltonian, we find that

$$-\nabla^{\perp}(\Psi(\cdot,t)\circ\boldsymbol{X}_{[t,t_0]}) = \nabla^{\perp}(\psi(\cdot,t)\circ\boldsymbol{\Phi}_{[t_0,t]}\circ\boldsymbol{X}_{[t,t_0]}) = \boldsymbol{u}$$

recovers the forward velocity. We have not investigated this duality further, however.

2.1.3 Conservative integration near boundaries

This section is an aside for a future project, in which we would consider a domain with boundary. Let Ω be a domain in $\mathbb{R}^2$ with a piecewise-smooth boundary $\partial\Omega$. For problems on such domains, the velocity $\boldsymbol{u}$ typically satisfies a boundary condition of the form

$$\boldsymbol{u}\cdot\boldsymbol{n} = 0 \quad \text{on } \partial\Omega,$$

where $\boldsymbol{n}$ is the unit outward normal to $\partial\Omega$. In the case of the incompressible Euler equations, this corresponds to the slip boundary condition. With the above condition, the true trajectories of $\boldsymbol{u}$ do not cross $\partial\Omega$. It would be desirable for a numerical scheme to also have this property, i.e. to map the interior of Ω to itself. There are a number of qualitative reasons for this. There are also technical reasons. For instance, advected quantities, interpolants, and so on, that need to be evaluated along trajectories, may only be defined in the interior of Ω.

A numerical integrator, however, in general does not respect boundaries. This is especially true in the case of a complex (e.g. non-rectangular) domain. One property of interest here for integrators is positivity preservation ([21], Sections I.7, II.4). In general, the only unconditionally positivity-preserving integrator (that is, without restriction on the time step Δt) is the backward Euler scheme. Any other Runge-Kutta integrator is only positivity-preserving, for a general velocity, when Δt is sufficiently small.

A potential solution is to project onto $\overline{\Omega}$ after each time step. This projection presents a number of issues, however. First, an orthogonal projection may not be well-defined, for instance in the case of a non-convex domain. One could instead consider a projection along solution curves, though this adds a layer of complexity. Moreover, these projections can reduce the order of the method. They will also affect the conservation properties of the scheme. Finally, they can send interior trajectories to the boundary $\partial\Omega$, a qualitatively undesirable feature of the solution.

Thus, it would be ideal if by construction, the integrator mapped Ω to itself. This is possible to achieve by exploiting properties of conservative integrators. One possibility we have in mind is the following. In an autonomous Hamiltonian system, integral curves follow level sets of the Hamiltonian. We can construct an integrator that exactly conserves the Hamiltonian, for instance with the Discrete Multiplier Method ([42]). Then if the velocity interpolant is constructed so that $\partial\Omega$ is a contour of the Hamiltonian, the integrator will by construction map the interior of Ω to itself.

In particular, all of the work in not crossing the boundary is done by the integrator. There is no need to design a spatial interpolant that maps Ω to itself, which may be a difficult task on a complex domain. Using the framework of this thesis, in particular the space-time decoupled integration of the backward map, opens the possibility for Characteristic Mapping Methods on complicated domains.

2.2 Symplectic integration for volume preservation

Using Proposition 2.1, we may now integrate the backward map conservatively. For the rest of this thesis we focus on volume preservation. The class of integrators that preserve volume, in two dimensions, is precisely the class of symplectic integrators. Here we review some properties of symplectic integrators, see some examples, and then examine their relationship to volume preservation.

2.2.1 Symplectic integrators and their properties

A map $\mathbf{\Phi}$ between subsets of $\mathbb{R}^2$ is called symplectic if it preserves the standard symplectic form $dx_1 \wedge dx_2$. This condition can be written in coordinates as

$$\nabla\mathbf{\Phi}^T J \nabla\mathbf{\Phi} = J, \qquad J := \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}.$$

These definitions generalize to even-dimensional spaces $\mathbb{R}^{2d}$ along with their standard symplectic structures, and to general manifolds equipped with a symplectic form.

A theorem of Poincaré ([18], VI.2) says that Hamiltonian vector fields on $\mathbb{R}^{2d}$ generate flows that are symplectic maps for all time. Moreover, a classical result says that any symplectic map (sufficiently regular and close to the identity) is the time-1 flow of some non-autonomous Hamiltonian ([9]). There is thus a close connection between symplectomorphisms (symplectic maps that are diffeomorphisms) and flows of Hamiltonian systems. In particular, it is natural to desire a numerical integrator which, when applied to a Hamiltonian system, gives discrete (in time) flows that are symplectomorphisms.

A numerical integrator is called symplectic if, when applied to a Hamiltonian vector field, its resulting flow maps are symplectic. Some history on the development of symplectic integrators can be found in [11] and [18]. It turns out that these integrators are precisely the volume-preserving integrators in two dimensions, and are useful for volume preservation in higher dimensions. We delay a discussion of volume preservation to Section 2.2.3. Symplectic integrators also satisfy a number of other remarkable properties. While we will not use any of these, we spend the rest of this section listing some of them.

Interpolating Hamiltonian. As previously mentioned, an arbitrary symplectic map that is sufficiently close to the identity, is exactly the time-1 flow of a non-autonomous Hamiltonian. This is a classical result when the symplectic map is analytic, known as an interpolating Hamiltonian problem. This has also been proven to be true in the case when the symplectomorphism is only C^1

([9]). In particular, the discrete symplectic flows we compute in this thesis (which are computed by some C^k interpolated velocity), are exactly Hamiltonian flows; only, the Hamiltonian in question is slightly modified.

Modified equation, quasi-interpolating autonomous Hamiltonian. On the topic of modified Hamiltonians, it is known that flows of symplectic integrators satisfy modified equations that are also Hamiltonian ([18], IX.3). These modified equations are typically infinite series, meant to be interpreted formally. One can however truncate these series such that the truncations are also Hamiltonian. Via these truncations, one can show the existence of an autonomous Hamiltonian whose flow approximates the flow given by a symplectic integrator, to a high order and for an exponentially-long time. Relatedly, there are results on the existence of quasi-interpolating autonomous Hamiltonians. That is, given a symplectomorphism, one can find an autonomous Hamiltonian whose time-1 flow approximates the symplectomorphism exponentially-well ([5]).

Quasi-conservation and stability. It is known that symplectic integrators in general do not conserve the Hamiltonian or other invariants. However, in light of the quasi-interpolation results above, symplectic integrators do tend to be quasi-conservative, in the sense that they are conservative to a very high degree and for very long times ([11], [5]). Due to this quasi-conservation, symplectic integrators have excellent long-term stability properties ([43], [11]).

Discrete variational principle, high-order integrators. Symplectic integrators can be shown to satisfy a discrete Hamilton's principle ([18], VI.6). That is, they extremize a discretized action functional, defined in terms of a discretized Lagrangian. In fact, symplectic integrators can actually be derived by starting with a discrete variational principle. Finally, one can derive symplectic integrators of arbitrarily-high order. This is typically achieved by the use of so-called generating functions for symplectic maps, see [11] or [18], Section VI.5.

2.2.2 Examples and Runge-Kutta structure

Let $\boldsymbol{u} : \mathbb{R}^2 \times [t_0, \infty) \to \mathbb{R}^2$ be a C^1 Hamiltonian vector field, i.e. $\boldsymbol{u} = \nabla^\perp \psi$ for some Hamiltonian ψ. Let $\boldsymbol{\Phi}^0(\boldsymbol{x}) = \boldsymbol{x}$ in $\mathbb{R}^2$. We will write

$$\boldsymbol{\Phi}^1(\boldsymbol{x}), \boldsymbol{\Phi}^2(\boldsymbol{x}), \ldots : \mathbb{R}^2 \to \mathbb{R}^2$$

for a sequence of flow maps (not discretized in space) given by a numerical integrator. Take a sequence of uniform time steps $t_0, t_1, \ldots$, with step size Δt.

In this thesis we consider three well-known symplectic integrators. The first is one variant of the symplectic Euler method,

$$\boldsymbol{\Phi}^{n+1} = \boldsymbol{\Phi}^n + \Delta t \ \boldsymbol{u}(\Phi_1^{n+1}, \Phi_2^n, t_n), \tag{2.6}$$

with the other variant seeing the velocity replaced with $\boldsymbol{u}(\Phi_1^n, \Phi_2^{n+1}, t_n)$. This is a first-order method. The second is the standard midpoint method,

$$\boldsymbol{\Phi}^{n+1} = \boldsymbol{\Phi}^n + \Delta t \ \boldsymbol{u}\left(\frac{\boldsymbol{\Phi}^n + \boldsymbol{\Phi}^{n+1}}{2}, t_n + \frac{\Delta t}{2}\right), \tag{2.7}$$

which is second order. The last is a fourth-order method, with two intermediate stages, referred to as a Gauss collocation method. Its Butcher tableau is given in Figure 2.2. Proofs of symplecticity of the above methods can be found in [18], Chapter VI (though we essentially replicate the proof for the symplectic Euler method in Proposition 2.3 below).

$$\begin{array}{c|cc} \frac{1}{2} - \frac{\sqrt{3}}{6} & \frac{1}{4} & \frac{1}{4} - \frac{\sqrt{3}}{6} \\ \frac{1}{2} + \frac{\sqrt{3}}{6} & \frac{1}{4} + \frac{\sqrt{3}}{6} & \frac{1}{4} \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}$$

Figure 2.2: Butcher tableau of 4th-order symplectic Gauss collocation method.

One notices that all the symplectic integrators above are implicit. This is in fact the case for all symplectic Runge-Kutta methods, when applied to arbitrary velocities, at least. The following is a theorem from [18], Section VI.7.

Theorem 2.2. *An irreducible Runge-Kutta method with Butcher tableau*

$$\begin{array}{c|ccc} c_1 & a_{11} & \cdots & a_{1s} \\ \vdots & \vdots & \ddots & \\ c_s & a_{s1} & & a_{ss} \\ \hline & b_1 & \cdots & b_s \end{array}$$

is symplectic if and only if

$$b_i a_{ij} + b_j a_{ji} = b_i b_j \tag{2.8}$$

for all i, j.

Similar necessary and sufficient conditions for the symplecticity of e.g. irreducible, partitioned Runge-Kutta methods exist, see [18]. From this we can see that a symplectic irreducible Runge-Kutta method applied to a general velocity must be implicit: assume without loss of generality that $b_i \neq 0$ for all i (otherwise, the method can be reduced). Then along diagonals, the condition (2.8) gives

$$2b_i a_{ii} = b_i^2,$$

and so $a_{ii} \neq 0$, meaning the integrator is at least diagonally implicit.

There are special cases where the symplectic integrators in fact end up being explicit, for certain forms of the velocity. For instance, if the Hamiltonian in question is separable, that is, $H(x_1, x_2) = T(x_1) + U(x_2)$, then some symplectic integrators, such as the symplectic Euler and Störmer-Verlet schemes, are explicit ([18], VI.3). There are many problems in mechanics where the Hamiltonian is separable, and so the explicit symplectic integrators are popular there due to their high stability and conservation properties. In our case, however, we will not know velocities ahead of time, and would not like to restrict to certain classes of velocities. Thus for our purposes, the structure-preserving integrators are always implicit.

Finally, we note that there exist symplectic integrators in other symplectic geometries. For instance, the so-called spherical midpoint method is a symplectic integrator on the sphere S^2 ([29]). It is however more challenging to derive symplectic integrators for general Hamiltonians in non-Euclidean spaces, due to topological considerations; see the discussion in [29].

2.2.3 Volume preservation

As we saw, a map $\mathbb{R}^2 \to \mathbb{R}^2$ is symplectic if it preserves the symplectic form $dx_1 \wedge dx_2$. But this is exactly the volume form. Thus in two dimensions, a map is symplectic if and only if it is volume- and orientation-preserving. Hence, in the case of a Hamiltonian vector field, the integrators from the previous section are exactly volume-preserving. In fact, the volume preservation of symplectic integrators only requires the vector field to be locally Hamiltonian. This reflects the fact that the flow of a vector field is symplectic if and only if the vector field is locally Hamiltonian ([18], VI.2). Thus, since in two dimensions locally Hamiltonian is equivalent to divergence-free, the integrators above are volume-preserving in 2D when $\operatorname{div} \boldsymbol{u} = 0$. We check this explicitly in the case of the symplectic Euler method, in the following proposition. Proofs of volume preservation for the other integrators from Section 2.2.2 follow similarly.

Proposition 2.3. *In $\mathbb{R}^2$, when $\operatorname{div} \boldsymbol{u} = 0$, the discrete flow $\boldsymbol{\Phi}^n(\boldsymbol{x})$ defined by the symplectic Euler method (2.6) satisfies $\det \nabla \boldsymbol{\Phi}^{n+1}(\boldsymbol{x}) = \det \nabla \boldsymbol{\Phi}^n(\boldsymbol{x})$. In particular when $\boldsymbol{\Phi}^0 = id$, each $\boldsymbol{\Phi}^n$ is a volume-preserving map.*

Proof. We can write the symplectic Euler step as $\boldsymbol{F}(\boldsymbol{x}, \boldsymbol{\Phi}^{n+1}(\boldsymbol{x})) = 0$ with

$$\boldsymbol{F}(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{\Phi}^n(\boldsymbol{x}) - \boldsymbol{y} + \Delta t \; \boldsymbol{u}(y_1, \Phi_2^n(\boldsymbol{x}), t_n).$$

Then $\boldsymbol{\Phi}^{n+1}$ is defined by the implicit function theorem. The theorem also gives an expression for $\nabla \boldsymbol{\Phi}^{n+1}$ (equivalently, by the chain rule):

$$\nabla \boldsymbol{\Phi}^{n+1} = -(\nabla_{\boldsymbol{y}} \boldsymbol{F})^{-1} \nabla_{\boldsymbol{x}} \boldsymbol{F} \qquad \Longrightarrow \qquad \det \nabla \boldsymbol{\Phi}^{n+1} = \frac{\det \nabla_{\boldsymbol{x}} \boldsymbol{F}}{\det \nabla_{\boldsymbol{y}} \boldsymbol{F}}.$$

Here, the gradients of $\boldsymbol{F}$ are evaluated at $(\boldsymbol{x}, \boldsymbol{\Phi}^{n+1}(\boldsymbol{x}))$. We compute

$$\det \nabla_{\boldsymbol{y}} \boldsymbol{F} = \det \begin{pmatrix} -1 + \Delta t \; \partial_{x_1} u_1 & 0 \\ \Delta t \; \partial_{x_1} u_1 & -1 \end{pmatrix} = 1 - \Delta t \; \partial_{x_1} u_1,$$

where $\boldsymbol{u}$ is evaluated on (y_1, Φ_2^{n+1}, t_n). Similarly, a straightforward computation gives

$$\det \nabla_{\boldsymbol{x}} \boldsymbol{F} = (1 + \Delta t \; \partial_{x_2} u_2) \det \nabla \boldsymbol{\Phi}^n.$$

Thus

$$\det \nabla \boldsymbol{\Phi}^{n+1} = \frac{1 + \Delta t \; \partial_{x_2} u_2}{1 - \Delta t \; \partial_{x_1} u_1} \det \nabla \boldsymbol{\Phi}^n.$$

But since $\operatorname{div} \boldsymbol{u} = 0$, we in fact have $\det \nabla \boldsymbol{\Phi}^{n+1} = \det \nabla \boldsymbol{\Phi}^n$. □

In higher dimensions, the situation of volume preservation is more delicate. In $\mathbb{R}^{2d}$, $d > 1$, symplectic implies volume-preserving: the wedge power of the symplectic form is the scaled volume form. The converse is not true, however; here symplectic is much stronger than volume-preserving. This reflects the fact that general divergence-free vector fields in $\mathbb{R}^{2d}$ are far from Hamiltonian. Thus a symplectic integrator might not be volume-preserving for a divergence-free vector field in higher even dimensions. And in odd dimensions, of course, there is no symplectic structure to begin with. While in certain very specific cases a symplectic integrator can be volume-preserving in $\mathbb{R}^3$ for instance ([18], VI.9), the general case remains unclear.

Moreover in [22], it is shown that non-partitioned Runge-Kutta methods in dimension $n \geqslant 3$ cannot be volume-preserving, for a general divergence-free analytic velocity. In the same article however is proposed a workaround for volume-preserving integration in arbitrary dimensions. We illustrate here idea in the case of three dimensions.

In $\mathbb{R}^3$ an arbitrary divergence-free vector field $\boldsymbol{u}$ can be written as a curl,

$$\boldsymbol{u} = \operatorname{curl} \boldsymbol{\psi} = \begin{pmatrix} \partial_{x_2} \psi_3 - \partial_{x_3} \psi_2 \\ \partial_{x_3} \psi_1 - \partial_{x_1} \psi_3 \\ \partial_{x_1} \psi_2 - \partial_{x_2} \psi_1 \end{pmatrix}.$$

This however can be rewritten as a sum of vector fields that resemble 2D Hamiltonians:

$$\boldsymbol{u} = \begin{pmatrix} 0 \\ \partial_{x_3}\psi_1 \\ -\partial_{x_2}\psi_1 \end{pmatrix} + \begin{pmatrix} -\partial_{x_3}\psi_2 \\ 0 \\ \partial_{x_1}\psi_2 \end{pmatrix} + \begin{pmatrix} \partial_{x_2}\psi_3 \\ -\partial_{x_1}\psi_3 \\ 0 \end{pmatrix}.$$

This decomposition into a sum of 2D Hamiltonians generalizes to arbitrary dimension, which is given in [22]. The proposal in [22] is then to apply a 2D symplectic integrator to each component of the decomposition above, and to combine these flows via a splitting method. The resulting scheme is volume-preserving in higher dimensions.

2.3 Finite difference volume preservation test

To get a sense of the volume preservation qualities of symplectic integrators, we perform a test. Here we compute the backward flow map of an analytically-known velocity, and approximate its Jacobian with a finite difference scheme. Of course, since the finite difference is not exact, the computed Jacobian error will not be exactly at machine precision. In the next section we will repeat the test and compute the Jacobian with Fourier spectral methods. There we will see a Jacobian error at machine precision. We will also see qualitative differences between the finite difference and spectral Jacobian errors, which is part of the reason for which we include this test.

We think of the torus as $\mathbb{R}^2/2\pi\mathbb{Z}^2$. We take the following Hamiltonian vector field,

$$\boldsymbol{u}(\boldsymbol{x}) = \nabla^\perp \left(2\sin^2\left(\frac{x_1}{2}\right)\sin^2\left(\frac{x_2}{2}\right)\right).$$

This is an autonomous version of the so-called swirl test, from [30]. We take an autonomous vector field because the velocity of the backward map is then simply

$$\boldsymbol{v} = -\boldsymbol{u}.$$

In particular, we do not need to worry about computing the forward flow and interpolating the velocity, to use the backward velocity of Proposition 2.1. We would like to test with an analytically-known vector field, to see the best Jacobian results available. We will cover the non-autonomous case with our interpolation strategy of Section 3.

Here and in the rest of this thesis, we use the notation $\boldsymbol{\chi}$ to denote a discretized backward map. Define a grid Γ of $N \times N$ points,

$$\Gamma = \{\boldsymbol{x_j} := h\boldsymbol{j} : 0 \leqslant j_1, j_2 < N\}, \tag{2.9}$$

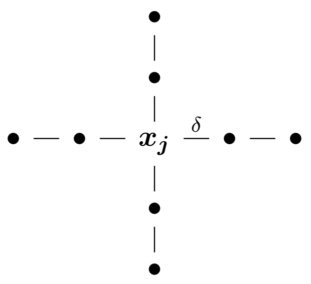


Figure 2.3: Centered finite difference stencil in each dimension around $\boldsymbol{x_j}$.

where $h = \frac{2\pi}{N}$. Around each point $\boldsymbol{x_j}$ of Γ we put a finite difference stencil. Let $\delta > 0$ denote the stencil size. For this test we choose $\delta = 10^{-3.5}$. The stencil consists of the points $\boldsymbol{x_j} + (i\delta, 0)$ and $\boldsymbol{x_j} + (0, i\delta)$ for $i \in \{\pm 1, \pm 2\}$, and is illustrated in Figure 2.3. Consider the fourth-order centered finite-difference approximation

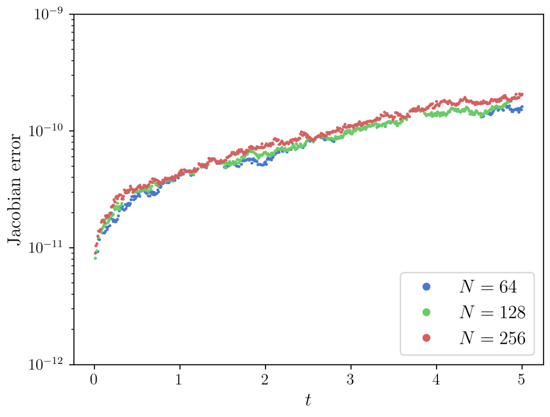
$$\frac{d}{dx} f(x_0) \approx \frac{1}{\delta} \left(\frac{1}{12} f(x_0 - 2\delta) - \frac{2}{3} f(x_0 - \delta) + \frac{2}{3} f(x_0 + \delta) - \frac{1}{12} f(x_0 + 2\delta) \right) \tag{2.10}$$

for $f : \mathbb{R} \to \mathbb{R}$. We first flow a stencil of points around each $\boldsymbol{x_j}$ by a symplectic integrator. We then use the above finite difference (2.10) to approximate the gradient of the backward map at each $\boldsymbol{x_j}$, using the stencil points. Finally, we take the maximum Jacobian error over the grid. That is, we compute $\|\det D\boldsymbol{\chi} - 1\|_{\ell^\infty(\Gamma)}$, where we have written D for the finite difference described above.

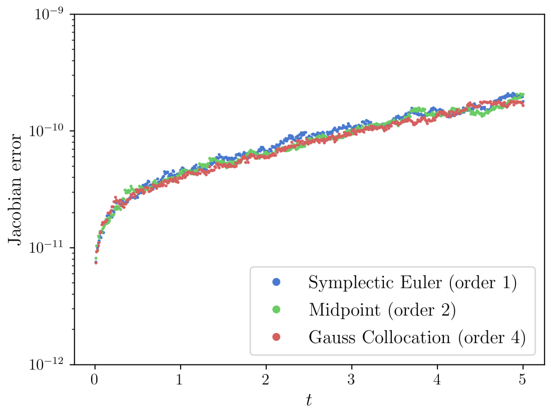
The symplectic integrators are implemented by a standard fixed-point iteration. The iteration is initialized with a forward Euler step. We iterate until the increment is within a parameter $\varepsilon_{\text{sym}} > 0$. Here we take $\varepsilon_{\text{sym}} = 10^{-13}$. We have found experimentally that we do not need to take ε_{sym} too close to machine precision for convergence of the Jacobian error. Moreover we do not want to take ε_{sym} too small, to avoid convergence issues and so as to not introduce spurious oscillations.

For the first three tests, we isolate the following parameters: the grid size N, the symplectic integrator (and in particular, the order of the integrator), and the time step Δt. The Jacobian errors over time are plotted in Figure 2.4 for $t \in [0, 5]$. Figure 2.4a compares the Jacobian errors across grid sizes. Here the midpoint integrator is used, and $\Delta t = 1/100$. Figure 2.4b compares the various symplectic schemes. Here we take $N = 128$ and $\Delta t = 1/100$. Figure 2.4c compares different values of Δt. Here we use the midpoint integrator and $N = 128$. Finally we run a fourth test, which is a long-term test. Jacobian error results are plotted in Figure 2.4d. Here we run until $t = 100$, using the midpoint integrator, $N = 128$, and $\Delta t = 1/10$. We shall return to this long-term test in the following section.

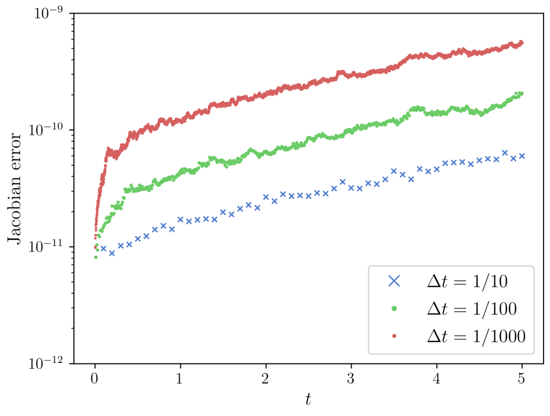
We see that there are very little Jacobian differences when varying the grid size and the symplectic scheme. There are some small differences when varying Δt; the Jacobian error is smaller for larger Δt. However, this is likely due to the simple fact that there are more computations to be carried out for smaller Δt. These tests illustrate a crucial point: symplectic integrators are, by construction, exactly volume-preserving. They are not volume-preserving up to some order, which is the case for all other integrators. Volume preservation here is an "algebraic" property of the integrators, and not an analytic one.



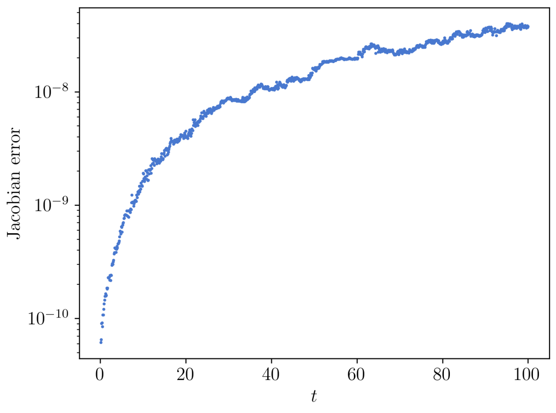
(a) Grid size comparison; midpoint scheme, $\Delta t = 1/100$.



(b) Integrator comparison; $N = 128$, $\Delta t = 1/100$.



(c) Time step comparison; midpoint scheme, $N = 128$.



(d) Long-term test; midpoint scheme, $N = 128$, $\Delta t = 1/10$.

Figure 2.4: Finite difference Jacobian errors of symplectic schemes.

3 Essentially-conservative spatial discretization

At this point we are able to integrate a backward map in a conservative manner. We will see that, for reasons of spatial resolution, using a single map is insufficient. We will instead split the backward map into submaps, using the semigroup decomposition (1.2). This however presents an additional problem: how to compose submaps in a conservative manner. At this stage, we are unaware of how to do so. Instead, we perform the composition to a high order of volume preservation. We will do so in part using Fourier-spectral upsampling. Thus, for the rest of this thesis, we switch from computing the Jacobian error with finite differences to doing so in Fourier space.

In this section, we first examine the issue of spatial resolution. We then discuss a strategy to compose the submaps to a high order of conservation. Finally we describe an interpolation scheme for the velocity, which will be required for the composition strategy to be correct.

3.1 Illustration of spatial resolution

3.1.1 Fourier-pseudospectral Jacobian test

We begin by repeating the volume preservation test of Section 2.3, this time computing the Jacobian in Fourier space. We denote by $\mathcal{F}$ the DFT/FFT, and use a hat to denote the true Fourier coefficients. We take the same grid Γ of $N \times N$ points. Let F be a function $T^2 \to \mathbb{R}$, and denote by $f \in \mathbb{R}^\Gamma$ its projection onto the grid: $f_{\boldsymbol{j}} = F(\boldsymbol{x_j})$. We compute ∂_{x_j} in frequency space as usual by the Fourier collocation derivative operator ([8]), which we denote by $\mathcal{D}_j : \mathbb{R}^\Gamma \to \mathbb{R}^\Gamma$:

$$\mathcal{D}_j f = \mathcal{F}^{-1}(D_j \cdot \mathcal{F}(f)).$$

Here the dot means componentwise multiplication, and D_j is the matrix of wavenumbers

$$(D_j)_{\boldsymbol{k}} = ik_j.$$

We write $\mathcal{D}$ for the Fourier collocation version of the gradient operator ∇.

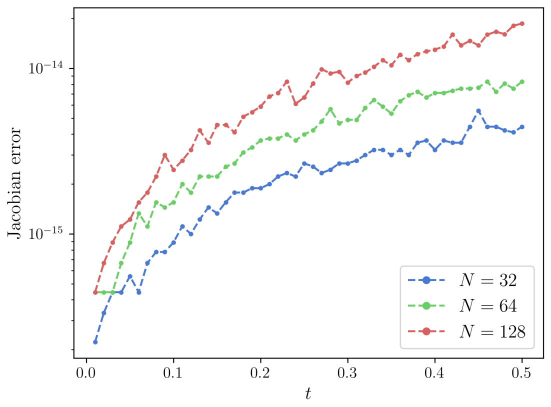
Remark 3.1. *Here we flow maps $\boldsymbol{\chi}$ that are periodic perturbations of the identity. We assume the maps take the form*

$$\boldsymbol{\chi}(\boldsymbol{x}) = \boldsymbol{x} + \boldsymbol{\eta}(\boldsymbol{x}),$$

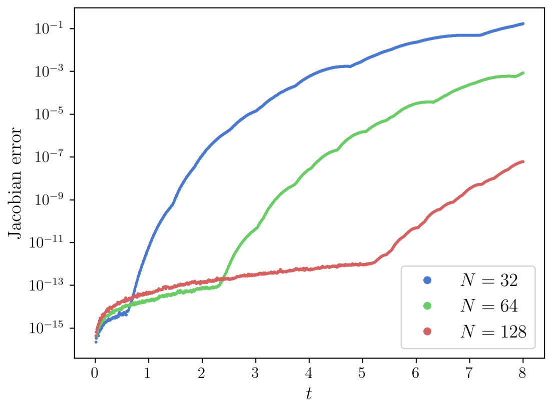
for $\boldsymbol{\eta}$ a periodic map on T^2. Operations in Fourier space should be applied to smooth, periodic data. For the sake of notation, however, we will write

$$\mathcal{D}\boldsymbol{\chi} := I + \mathcal{D}\boldsymbol{\eta}.$$

We will use similar notation for the Fourier upsampling and periodic spline interpolation operators, that we will define in the upcoming sections. Thus, while we understand that operators in Fourier space should be applied to the periodic perturbation $\boldsymbol{\eta}$ and not to the map $\boldsymbol{\chi}$, we will apply them to $\boldsymbol{\chi}$ for notational convenience.



(a) Short-term Jacobian error.



(b) Long-term Jacobian error.

Figure 3.1: Spectral Jacobian errors of the backward map in the swirl test, over different grid sizes; midpoint scheme, $\Delta t = 1/100$.

At each time step, we plot the maximum spectral Jacobian error of the map $\boldsymbol{\chi}_{[t,t_0]}$ on the grid; that is, the quantity

$$e_{\det}[\boldsymbol{\chi}_{[t_n,t_0]}] := \|\det \mathcal{D}\boldsymbol{\chi}_{[t_n,t_0]} - 1\|_{\ell^\infty}.$$

We note that we do not use the spectral gradient in any computations that flow the map. The maps are integrated exclusively by the symplectic integrators, and there are no spatial projections between time steps. We only use the spectral gradient to assess the Jacobian generated by the symplectic integrators. The test is run with the midpoint integrator, with $\Delta t = 1/100$. We repeat the test over various grid sizes N only. For the same reasons as in Section 2.3, different choices of the scheme and of Δt have very little effect on the Jacobian error. The Jacobian errors are plotted in the short term and in the long term in Figure 3.1.

In the short term, we see a Jacobian error very close to machine precision. The finer grids have a slightly larger error, but this is simply due to having to compute FFTs on a larger grid. In the long term, however, the Jacobian error has surprising behaviour. We see that the error on each grid splits into two distinct regimes. In the first regime we see slow growth. There is a point however where the error shoots upward, after which it increases at a much faster rate. After this point is reached, the data does not seem to be at all volume-preserving.

The time at which the error switches regime seems to depend on the grid size N. Moreover, while we don't include the relevant plots here, it depends only on N, and not on Δt or on the integrator used. Further, no matter the grid size, there will always eventually be a point where the error shoots upward. The spectral Jacobian error here is qualitatively very different from the finite-difference error from Section 2.3. This is especially true considering the long-term test in Figure 2.4d. There, the finite-difference error seemed to stay close to zero for a long time, at least until $t = 100$. Moreover, we know that our maps should be volume-preserving by construction, due to the symplectic integrators. There thus appears to be a contradiction. It turns out that this behaviour is not contradictory, and we explain it in the coming sections.

3.1.2 The Fourier spectrum

To explain the Jacobian behaviour of the previous section, we first see some Fourier approximation theory. We introduce some objects and facts from [8], Sections 2.1, 9.1. Here, again, $F : T^2 \to \mathbb{R}$ is a function and f is its projection onto the grid.

Define the space of trigonometric polynomials of degree $N/2$,

$$S_N = \operatorname{span}\{e^{i\boldsymbol{k}\cdot\boldsymbol{x}} : -N/2 \leqslant \boldsymbol{k} < N/2\}.$$

The inequality $-N/2 \leqslant \boldsymbol{k} < N/2$ is taken componentwise. There are two main ways of projecting functions to S_N: truncation and interpolation. The truncation operator P_N is defined as expected, by truncating higher wavenumbers:

$$P_N\left(\sum_{\boldsymbol{k}\in\mathbb{Z}^2} \hat{F}_{\boldsymbol{k}} e^{i\boldsymbol{k}\cdot\boldsymbol{x}}\right) = \sum_{-N/2\leqslant\boldsymbol{k}<N/2} \hat{F}_{\boldsymbol{k}} e^{i\boldsymbol{k}\cdot\boldsymbol{x}}.$$

We next define the interpolation operator I_N as follows: $I_N F$ is the unique element of S_N which interpolates F on the grid Γ ([8]). $I_N f$ is similarly defined for any data f on the grid. This interpolant can be constructed using the DFT. In fact, the DFT precisely computes the (scaled) Fourier coefficients of the interpolant: $\mathcal{F}$ and I_N are related by

$$(\mathcal{F}f)_{\boldsymbol{k}} = N^2(\widehat{I_N f})_{\boldsymbol{k}}.$$

It follows that the quantity computed by the Fourier collocation derivative is the smooth derivative of the trigonometric interpolant, projected back onto the grid:

$$\mathcal{D}_j f = (\partial_{x_j} I_N f)|_\Gamma. \tag{3.1}$$

Hence, understanding the operator $\mathcal{D}$ requires a study of the interpolation operator, and in particular the interpolation error.

The interpolation error $\|I_N F - F\|$ is in general quite nontrivial to quantify. Indeed, interpolating a function over a grid may miss important features of the function. As such, an analysis of sampling and aliasing is required; see [8] for discussions of these. Fortunately, we may think of the interpolation error in terms of the truncation error $\|P_N F - F\|$, on regular grids at least. The following relations between the two hold. The L^2 interpolation error is always at least the L^2 truncation error ([8]):

$$\|I_N F - F\|_{L^2} \geqslant \|P_N F - F\|_{L^2}.$$

On the other hand, the L^∞ interpolation error is never more than twice the L^∞ truncation error ([7]):

$$\|I_N F - F\|_{L^\infty} \leqslant 2\|P_N F - F\|_{L^\infty}.$$

Moreover in [8], it is shown that the L^2 interpolation error is asymptotically equal to the L^2 truncation error, as the grid size $N \to \infty$. Thus, while interpolation sees problems not present in truncation, namely aliasing, it in fact suffices to consider truncation as a model for the interpolation error. The truncation error is much more straightforward. The truncation operator zeroes all modes higher than the Nyquist frequency $N/2$, which is the highest representable frequency on the given grid. It then follows from Parseval's identity that the L^2 truncation error is given directly in terms of the removed Fourier coefficients:

$$\|P_N F - F\|_{L^2}^2 = 4\pi^2 \sum_{\boldsymbol{k} \notin [-N/2, N/2-1]^2} |\hat{F}_{\boldsymbol{k}}|^2.$$

To estimate the truncation error, we thus consider the decay in the Fourier coefficients. It is a fact that when $F \in C^m(T^2)$, the Fourier coefficients of F decay like

$$|\hat{F}_{\boldsymbol{k}}| = \mathcal{O}\left(|\boldsymbol{k}|^{-(m+1)}\right), \qquad \text{as } |\boldsymbol{k}| \to \infty,$$

where $|\boldsymbol{k}| = \left(k_1^2 + k_2^2\right)^{1/2}$. Thus, C^∞ functions typically have geometric convergence of Fourier coefficients ([7]),

$$|\hat{F}_{\boldsymbol{k}}| = \mathcal{O}\left(e^{-c|\boldsymbol{k}|}\right), \qquad c > 0.$$

The rate of decay of Fourier coefficients roughly indicates how complicated a function is. A function with more fine details will have slower decay in its spectrum. Crucially, the facts about decay also extend to the discrete Fourier coefficients. Indeed, when the grid size N is sufficiently large, we have

$$|(\mathcal{F}f)_{\boldsymbol{k}}| = \mathcal{O}\left(|\hat{F}_{\boldsymbol{k}}|\right)$$

as $N, |\boldsymbol{k}| \to \infty$ and as long as $|\boldsymbol{k}| \leqslant N/2$; see [8]. These facts, combined with the interpolation bound, give the well-known geometric convergence of pseudospectral methods (for smooth problems).

On a computer, we will almost always incur truncation errors, since we can only store finitely many coefficients (and functions of interest are almost never finite Fourier series). However, if the truncation error is not much more than $\varepsilon_{\text{mach}}$, where $\varepsilon_{\text{mach}}$ denotes machine precision, then we consider this truncation acceptable. Indeed, we will be accumulating round-off errors around $\varepsilon_{\text{mach}}$ either way. As such, when working on a fixed grid of size N, we consider a function F to be "well-represented" on the grid if the $\varepsilon_{\text{mach}}$-support in frequency space of F is contained below the Nyquist frequency. By this we mean that $|\hat{F}_{\boldsymbol{k}}| \leqslant \varepsilon_{\text{mach}}$ whenever $|\boldsymbol{k}| \geqslant N/2$. Conversely, if Fourier coefficients of F around the Nyquist frequency are above $\varepsilon_{\text{mach}}$, then the truncation error (and thus, the spectral derivative error) becomes significant compared to the finite precision round-off error.

To conclude, the Fourier collocation derivative (3.1) is computed via an interpolation. The interpolation error can be modelled as a truncation. When the $\varepsilon_{\text{mach}}$-support of the Fourier coefficients is contained below the Nyquist frequency, we expect the interpolation error to be small, and thus to have spectral (essentially machine precision) accuracy in the derivative (3.1). Otherwise, and in particular when aliasing occurs, the truncation may be large, and the spectral accuracy can be lost.

Finally, it is in general difficult to estimate the incurred truncation and interpolation errors, especially when solving for unknown functions on the grid. To this end, what is typically done is to inspect the L^2 energies below the Nyquist frequency, and to estimate the truncation above Nyquist based on these. Define

$$E_K(f) = \frac{1}{2} \sum_{K \leqslant |\boldsymbol{k}| < K+1} |(\mathcal{F}f)_{\boldsymbol{k}}|^2$$

as in [45]. Then again by Parseval, $E_K(f)$ measures the L^2 energy of f in each "shell" of wavenumbers $K \leqslant |\boldsymbol{k}| < K + 1$. We can then estimate the magnitude of truncation by considering $E_K(f)$

for K near $N/2$.

3.1.3 Jacobian error and spatial truncation

At this point we are in a position to explain the Jacobian behaviour of Section 3.1.1. $\boldsymbol{\chi} \in (\mathbb{R}^2)^\Gamma$ is the flow by a symplectic integrator on the grid. We will also adopt the following notation: $\boldsymbol{\chi}(\boldsymbol{x}) : \mathbb{R}^2 \to \mathbb{R}^2$ denotes the same flow by the symplectic integrator, without discretizing in space. Then $\boldsymbol{\chi}$ is the projection of $\boldsymbol{\chi}(\boldsymbol{x})$ onto the grid. By construction, since the integrators are symplectic, we always have $\det \nabla\boldsymbol{\chi}(\boldsymbol{x}) \equiv 1$. However, by computing $\mathcal{D}\boldsymbol{\chi}$, we are not computing $\nabla\boldsymbol{\chi}(\boldsymbol{x})$, but rather $\nabla I_N\boldsymbol{\chi}(\boldsymbol{x})$. The Jacobian error thus depends on the truncation incurred by $I_N\boldsymbol{\chi}(\boldsymbol{x})$. To estimate the truncation error, we inspect the spectra of the maps over time, in Figure 3.2.

We consider the Jacobian error on the grid of size $N = 64$. In Figure 3.2a, we have zoomed into the spectral Jacobian error, near the change in regime. Here we clearly see the two different regimes of error growth, with the switch happening near $t \approx 2.3$. In Figure 3.2b we plot the spectra of the maps at times $t = 1.0, 2.2, 3.2$. At $t = 1.0$ we see the spectrum decay nicely until it hits $\varepsilon_{\text{mach}}^2$. This indicates that the backward map is well-represented on the grid at this point. At $t = 2.2$ we see that the $\varepsilon_{\text{mach}}$-support of the spectrum has just crossed the Nyquist frequency. At $t = 3.2$ we see that the spectrum is much higher than $\varepsilon_{\text{mach}}^2$ at Nyquist, which corresponds to a much higher Jacobian error. One can imagine a series of coefficients to the right of the graph that have been truncated. In Figure 3.2c we plot the spectra at much finer times $t = 1.6, 1.8, \ldots, 2.6$. We see the spectra rise slowly in time. The spectra seem to first cross the Nyquist at some $t \in [2.0, 2.2]$, which corresponds roughly to when the spectral Jacobian error shoots upward. Finally in Figure 3.2d we compare maps across grid sizes: the spectrum of the $N = 64$ grid is overlaid on the spectrum of the $N = 128$ grid, at $t = 3.2$. We recall from Figure 3.1 that at this time, the spectral Jacobian error of the $N = 128$ grid is still in the first regime. On the other hand, the error of the $N = 64$ grid has already shot upward here. One can picture in this graph that the $N = 64$ grid is missing coefficients that have been truncated (though this is technically interpolation, and not truncation).

To conclude, we reiterate that the maps are still always volume-preserving. That is, they are projections onto the grid of exactly volume-preserving maps. The time integration is decoupled from space, i.e. the evolution is diagonal in time. The map data at each grid point is thus the same regardless of the current grid size. In other words, the data on the coarser grids is just the restriction of the data on the finer grids. The obstruction to having a small spectral Jacobian error, then, has only to do with the number of grid points, or equivalently the number of Fourier modes.

When we see a sudden growth in the spectral Jacobian error, it is because we are undersampling the map and seeing aliasing errors. We are seeing a spatial truncation that shows up in the solution as dissipation, averaging.

One may wonder why the Jacobian error is not exactly at machine precision in the first regime. Indeed, there seems to always be a growth in Jacobian error, although it is very slow at first. This is because a volume-preserving map can almost never be written in the form $\boldsymbol{\chi}(\boldsymbol{x}) = \boldsymbol{x} + \boldsymbol{\eta}(\boldsymbol{x})$ where $\boldsymbol{\eta}$ is a finite Fourier series. The reason for this can be seen roughly as follows. Suppose we have a map of the form $\boldsymbol{\chi}(\boldsymbol{x}) = \boldsymbol{x} + \boldsymbol{\eta}(\boldsymbol{x})$ with $\boldsymbol{\eta} \in S_N$. Suppose that $\boldsymbol{\chi}$ is exactly volume-preserving, i.e. $\det \nabla \boldsymbol{\chi} \equiv 1$. This can be rewritten as

$$1 = \det \nabla \boldsymbol{\chi} = \det(I + \nabla \boldsymbol{\eta}) = (1 + \partial_{x_1}\eta_1)(1 + \partial_{x_2}\eta_2) - \partial_{x_2}\eta_1 \partial_{x_1}\eta_2,$$

which becomes

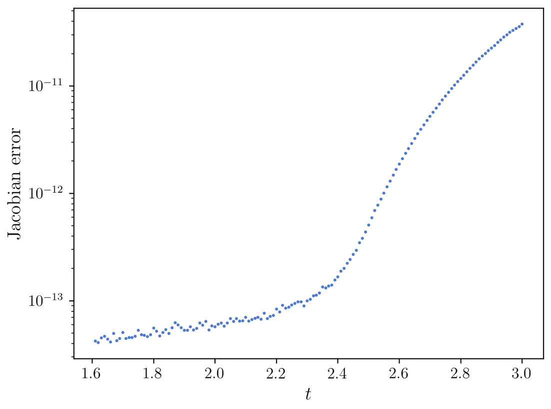
$$\operatorname{tr} \nabla \boldsymbol{\eta} + \det \nabla \boldsymbol{\eta} = 0. \tag{3.2}$$

Since $\boldsymbol{\eta} \in S_N$ we in fact have

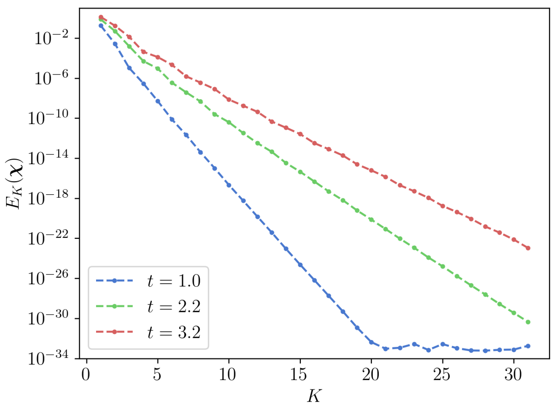
$$\operatorname{tr} \nabla \boldsymbol{\eta} = \det \nabla \boldsymbol{\eta} = 0. \tag{3.3}$$

This can be seen as follows. Assume that Equation (3.3) does not hold, so $\operatorname{tr} \nabla \boldsymbol{\eta}$ and $\det \nabla \boldsymbol{\eta}$ are both nonzero. Both of these have compact support in Fourier space, since $\boldsymbol{\eta} \in S_N$. The determinant term is similar to a convolution in Fourier space. Thus, barring some convenient cancellation, in most cases we should have that the frequency support of $\det \nabla \boldsymbol{\eta}$ is strictly larger than that of $\operatorname{tr} \nabla \boldsymbol{\eta}$. Then there is some wavenumber $\boldsymbol{k}$ such that $\mathcal{F}(\det \nabla \boldsymbol{\eta})_{\boldsymbol{k}} \neq 0$, yet $\mathcal{F}(\operatorname{tr} \nabla \boldsymbol{\eta})_{\boldsymbol{k}} = 0$. This contradicts Equation (3.2), however.

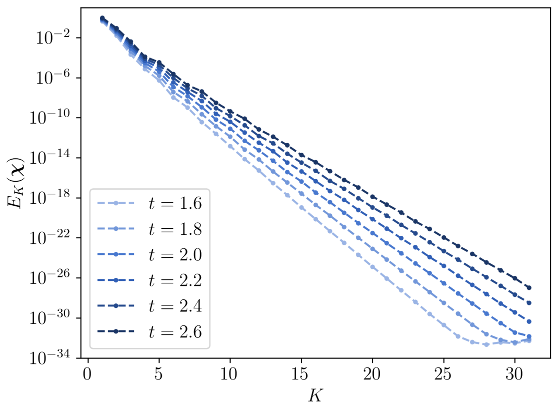
The conditions of Equation (3.3) are extremely restrictive. The identically zero Jacobian alone implies that $\nabla \eta_1$ and $\nabla \eta_2$ are everywhere colinear, and so the contour lines of η_1 and η_2 are the same. In other words, η_1 is a function of only η_2 (and vice versa). Maps obtained by integrating a velocity will almost never be of this form. It follows that a general volume-preserving map of interest cannot be represented as a finite Fourier series perturbation of the identity. This is expected, given the nonlinearity of the determinant. This means that when computing the spectral Jacobian, we will always have a nonzero truncation. Thus, even in the first regime, we will always see a slow growth in the spectral Jacobian error.



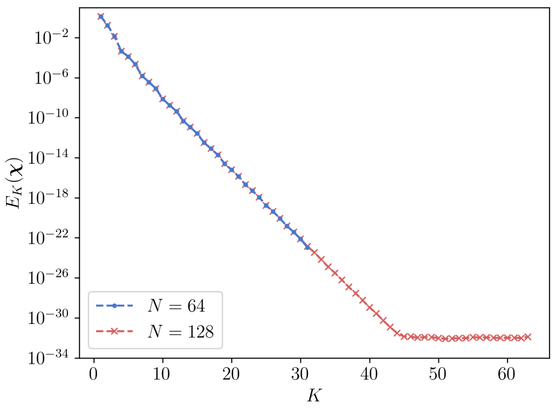
(a) Spectral Jacobian error of backward map, $N = 64$, near the change in regime.



(b) Spectra of backward map, $N = 64$, at $t = 1.0, 2.2, 3.2$.



(c) Spectra of backward map, $N = 64$, near $t = 2.2$.



(d) Spectra of backward map at $t = 3.2$, $N = 64$ vs. $N = 128$ grid.

Figure 3.2: Jacobian error and spectra of backward maps, near the change in regime of the spectral Jacobian error.

3.2 Backward map decomposition and sampling

In the previous section, we saw that on a fixed grid, we eventually lose spatial resolution in the backward map. This affects not only the Jacobian, but all important features of the solution. Recall however that we can use the semigroup decomposition (1.2) to split the backward map

$$\boldsymbol{X}_{[t_n,t_0]} = \boldsymbol{X}_{[\tau_1,\tau_0]} \circ \cdots \circ \boldsymbol{X}_{[\tau_m,\tau_{m-1}]} \tag{3.4}$$

over some time intervals $[\tau_{i-1}, \tau_i]$. The complex deformation is then split across multiple maps, each of which can be accurately represented on the fixed grid. In particular, the decomposition avoids large spatial truncation. The decomposition of the backward map has nothing to do with conservation; each submap is exactly volume-preserving. It is purely an issue of spatial resolution.

The above decomposition of the backward map is absolutely necessary in order to accurately represent the solution. However, it presents a new problem. We will eventually need to evaluate the advected quantity

$$\omega = \omega_0 \circ \boldsymbol{X}_{[\tau_1,\tau_0]} \circ \cdots \circ \boldsymbol{X}_{[\tau_m,\tau_{m-1}]}. \tag{3.5}$$

Performing this composition with grid data requires interpolating each $\boldsymbol{\chi}_{[\tau_i,\tau_{i-1}]}$. We would ideally perform this interpolation in an exactly volume-preserving manner. However, to our knowledge, there is currently no method to construct such an interpolant. Instead, we will perform this composition to a very high order of volume-preservation.

Moreover, the loss of spatial resolution does not only affect the backward map, but the advected quantity ω itself. Indeed, even if the backward maps are exactly volume-preserving, and we sample ω by Equation (3.5), if this sampling is done on too coarse a grid, we will see the same spatial truncation problem in ω. We will see this undersampling of ω in Section 5. Thus, the solution of the method we design does not consist of ω, sampled by (3.5), on a fixed grid. Rather, the solution consists of a sequence of submaps $\boldsymbol{\chi}_{[\tau_1,\tau_0]}, \ldots, \boldsymbol{\chi}_{[\tau_m,\tau_{m-1}]}$, as well as a specific interpolation scheme, to sample ω at arbitrary locations by the composition (3.5).

In this section we define the criterion by which we will split the submaps into the decomposition (3.4), called the remapping criterion. We then examine the obstruction to exactly volume-preserving composition. Finally we define our strategy for composing the submaps and sampling the final ω.

3.2.1 Jacobian remapping criterion

We refer to the process of dynamically splitting the backward map into submaps as remapping. We describe the remapping procedure now; the procedure is the same as for previous Characteristic Mapping methods (e.g. [45]). By the semigroup decomposition, we can arbitrarily decompose the backward maps into submaps. The submaps start at the identity and can be integrated independently. In practice, this means that we are able to integrate the submaps as follows. Suppose we are integrating the first submap $\boldsymbol{\chi}_{[t,\tau_0]}$ (we let $\tau_0 := t_0$). At time t_n, we have $\boldsymbol{\chi}_{[t_n,\tau_0]}$. Suppose that upon computing $\boldsymbol{\chi}_{[t_{n+1},\tau_0]}$, a spatial resolution criterion is violated. We discard the computation of $\boldsymbol{\chi}_{[t_{n+1},\tau_0]}$, set $\tau_1 := t_n$, and store the submap

$$\boldsymbol{\chi}_{[\tau_1,\tau_0]} := \boldsymbol{\chi}_{[t_n,\tau_0]}.$$

We then initialize a new submap at the identity $\boldsymbol{\chi}_{[t_n,\tau_1]} := \mathrm{id}$, and continue the iteration with this new submap. This process is what we refer to as remapping. We repeat this process with each subsequent submap until the iteration is complete. This yields a sequence of time intervals $[\tau_0,\tau_1],\ldots,[\tau_{m-1},\tau_m]$ as well as a sequence of backward maps $\boldsymbol{\chi}_{[\tau_1,\tau_0]},\ldots,\boldsymbol{\chi}_{[\tau_m,\tau_{m-1}]}$ over these intervals. This is the dynamic decomposition of the backward map.

The remapping criterion we will use is based on the spectral Jacobian error. This is the same condition as in [45]. Recall

$$e_{\det}[\boldsymbol{\chi}] = \|\det \mathcal{D}\boldsymbol{\chi} - 1\|_{\ell^\infty}.$$

We remap when $e_{\det}[\boldsymbol{\chi}] > \delta_{\det}$, for some parameter $\delta_{\det}$. Typically, $\delta_{\det}$ will be on the order of 10^{-13}. This means that each submap will satisfy $e_{\det}[\boldsymbol{\chi}_{[\tau_i,\tau_{i-1}]}] \leqslant \delta_{\det}$. Part of the motivation for this definition, from a conservation and composition point of view, is given in the upcoming Section 3.2.3.

There are however other reasons for which we believe the Jacobian remapping condition makes sense, from a spatial resolution point of view. Firstly, the Jacobian $\det \nabla\boldsymbol{\chi}$ is a difference of multiplications in physical space. Hence, its Fourier coefficients are a difference of convolutions of the Fourier coefficients of $\nabla\boldsymbol{\chi}$. For finite Fourier series, the support in frequency space of a convolution is in general the Minkowski sum of the supports of the factors. We thus expect the $\varepsilon_{\mathrm{mach}}$-support of the Jacobian to grow past the Nyquist frequency before that of the map does. Moreover, as a derivative, the Fourier coefficients of $\mathcal{D}\boldsymbol{\chi}$ are scaled by the wavenumber compared to the coefficients of $\boldsymbol{\chi}$. The spectrum of $\mathcal{D}\boldsymbol{\chi}$ thus decays more slowly than that of $\boldsymbol{\chi}$. It is thus a general practice to inspect derivatives, to determine how well data is represented on a grid. Finally, the Jacobian is a nonlinear invariant. In Section 3.1.3 we showed that is rarely well-represented by finite Fourier series. We thus expect the spectral Jacobian error to give a reasonable indication of the truncation error.

3.2.2 Interpolation in $\mathfrak{X}_0$ vs. in SDiff

We have mentioned previously that it is not clear how one would construct a structure-preserving interpolant in SDiff(T^2), the group of volume-preserving diffeomorphisms, such that moreover this interpolant is practical to construct and evaluate. We illustrate here the difficulty. By interpolant here we really mean approximation, as the constructions here are not generally interpolating.

We first contrast with the relative ease of interpolation in $\mathfrak{X}_0$, the algebra of divergence-free vector fields. Suppose we have $\boldsymbol{u} \in \mathfrak{X}_0(T^2)$. We would like to construct a polynomial approximation $\tilde{\boldsymbol{u}}$ such that $\tilde{\boldsymbol{u}} \in \mathfrak{X}_0(T^2)$ as well. A standard approach, as is done in [45], is as follows. We assume that $\boldsymbol{u} = \nabla^\perp \psi$ for some function $\psi : T^2 \to \mathbb{R}$; we interpolate ψ and call the interpolant $\tilde{\psi}$; and finally we set $\tilde{\boldsymbol{u}} = \nabla^\perp \tilde{\psi}$. Note that this last gradient is the gradient of a polynomial, which can be evaluated exactly. This gives an identically divergence-free polynomial $\tilde{\boldsymbol{u}}$. We note that we are not concerned with any properties of the interpolant $\tilde{\psi}$. All of the work in constructing a divergence-free interpolant is done by the gradient operator. To summarize, we have an operator $\nabla^\perp$ for which the following hold.

1. $\nabla^\perp$ maps from the vector space of functions $T^2 \to \mathbb{R}$, inside which interpolation is straightforward, to the space $\mathfrak{X}_0(T^2)$.
2. There exists a ψ such that $\boldsymbol{u} = \nabla^\perp \psi$.
3. We can find ψ from $\boldsymbol{u}$ by solving the Poisson problem $-\Delta \psi = \operatorname{curl} \boldsymbol{u}$. This can be done efficiently and to spectral accuracy in Fourier space, for instance.
4. When $\tilde{\psi}$ is a polynomial, $\nabla^\perp \tilde{\psi}$ can be inexpensively and exactly evaluated.

To construct an interpolant in SDiff(T^2), we could imitate the above procedure. This would consist of replacing the operator $\nabla^\perp$, which constructs elements of $\mathfrak{X}_0(T^2)$, with an operator $\mathcal{G}$, that constructs elements of SDiff(T^2). For this interpolant to be computationally feasible, it should satisfy similar properties to those listed above. We can thus extract from the points above some axioms that $\mathcal{G}$ should satisfy, for the purpose of yielding a practical interpolation.

1. $\mathcal{G}$ maps from a vector space V, inside which we can interpolate, to SDiff(T^2).

2. $\mathcal{G}$ is surjective, or sufficiently close to surjective.
3. $\mathcal{G}$ is relatively straightforward to invert, to some sufficiently high order. Of course, $\mathcal{G}$ need not be injective, and by invert we mean to find an element in each fibre.
4. $\mathcal{G}$ is inexpensively and exactly computable, on the subspace of interpolants in V.

To satisfy the first axiom, a reasonable choice for $\mathcal{G}$ could be some volume-preserving approximation of the exponential map $\exp : \mathfrak{X}_0(T^2) \to \mathrm{SDiff}(T^2)$. That is, a symplectic integrator. In a similar vein, we could take $\mathcal{G}$ as a map from the space of Hamiltonians to the symplectomorphisms, given the close connection between these two spaces mentioned in Section 2.2.1.

However, it is unclear which such $\mathcal{G}$ would satisfy all four axioms simultaneously. As an example, take $\mathcal{G}$ to be a time-1 step of the (symplectic) midpoint method, given a Hamiltonian. That is, $\mathcal{G} : C^1(T^2) \to \mathrm{SDiff}(T^2)$, $\psi \mapsto \mathbf{\Phi}$, where $\mathbf{\Phi}$ satisfies

$$\mathbf{\Phi} = \mathrm{id} + \nabla^{\perp}\psi\left(\frac{\mathrm{id} + \mathbf{\Phi}}{2}\right). \tag{3.6}$$

Then $\mathcal{G}$ satisfies axiom 1. Moreover, it is proven in [9] that any C^1 symplectomorphism, not too far from the identity, can be written as the time-1 flow of the midpoint method, for some autonomous Hamiltonian. This fact is remarkable on its own. It also establishes axiom 2, that $\mathcal{G}$ is surjective (for symplectomorphisms not too far from the identity).

On the other hand, it is not clear how to practically construct such a Hamiltonian. Thus axiom 3 does not hold. Moreover, the midpoint expression (3.6) is implicit. We typically compute this by fixed-point iteration. If the symplectomorphism is far from the identity, then the iteration may not converge, it may require a significant amount of interpolations of the Hamiltonian, and so on. Thus axiom 4 also fails.

To conclude, practical interpolation is currently infeasible in SDiff. More generally, an interpolant that conserves some given invariant must be constructed in a case-by-case basis, and may not even exist. On the other hand, interpolation in $\mathfrak{X}_0$ is relatively straightforward. We will exploit this fact in our composition and velocity interpolation schemes. In particular, we perform many operations in the algebra $\mathfrak{X}_0$, rather than attempting any in the group SDiff.

3.2.3 Map composition strategy

At this point we have a sequence of submaps $\boldsymbol{\chi}_{[\tau_1,\tau_0]}, \ldots, \boldsymbol{\chi}_{[\tau_m,\tau_{m-1}]}$ on the grid, each exactly volume-preserving and each having spectral Jacobian error $e_{\mathrm{det}}[\boldsymbol{\chi}] \leqslant \delta_{\mathrm{det}}$. Eventually we will need to evaluate the advected quantity ω. In the case of the incompressible Euler equations, ω is the vorticity, which is ultimately the quantity of interest. Recall that we can write ω as

$$\omega = \omega_0 \circ \boldsymbol{X}_{[\tau_1,\tau_0]} \circ \cdots \circ \boldsymbol{X}_{[\tau_m,\tau_{m-1}]}.$$

To sample ω at a point $\boldsymbol{x}_0$, then, we interpolate each $\boldsymbol{\chi}_{[\tau_i,\tau_{i-1}]}$ as $\tilde{\boldsymbol{\chi}}_{[\tau_i,\tau_{i-1}]}$, and compute

$$\omega(\boldsymbol{x}_0) = \left(\omega_0 \circ \tilde{\boldsymbol{\chi}}_{[\tau_1,\tau_0]} \circ \cdots \circ \tilde{\boldsymbol{\chi}}_{[\tau_m,\tau_{m-1}]}\right)(\boldsymbol{x}_0). \tag{3.7}$$

An important remark is in order. In a traditional numerical method, the solution would consist of ω at the final time, on a fixed grid. Restricting to a single grid size is undesirable due to spatial truncation, however. We saw this explicitly in the spectral Jacobian, for the case of the submaps. The same is true for ω: computing ω by (3.7) is still prone to undersampling, as we will see in Section 5.1.2. Thus, instead, the solution of our method consists of the data of Equation (3.7): a specific interpolant $\tilde{\boldsymbol{\chi}}_{[\tau_i,\tau_{i-1}]}$ for each $\boldsymbol{\chi}_{[\tau_i,\tau_{i-1}]}$ (and hence, a scheme to arbitrarily sample ω, without constraining to a single grid size).

We cannot perform the composition (3.7) in an exactly conservative manner, as illustrated in the previous section. Instead, we construct the interpolants $\tilde{\boldsymbol{\chi}}_{[\tau_i,\tau_{i-1}]}$ in such a way that the order of conservation is as high as possible. Thus while each map is exactly conservative, for lack of a better option, we essentially brute-force the final composition. Our solution for high-order conservation relies heavily on upsampling in Fourier space. We thus first define the Fourier upsampling operator.

We take a grid Γ_{up} of $N_{\text{up}} \times N_{\text{up}}$ points, where $N_{\text{up}} \geqslant N$. Then the Fourier upsampling operator $\mathcal{U}_{N_{\text{up}}} : \mathbb{R}^{\Gamma} \to \mathbb{R}^{\Gamma_{\text{up}}}$ maps data f on the coarse grid to data on the fine grid, by padding with zeros in Fourier space:

$$\mathcal{U}f = \frac{N_{\text{up}}^2}{N^2}\mathcal{F}^{-1}\begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathcal{F}f & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}.$$

Here the $\mathbf{0}$ are matrices of zeros of appropriate size. $\mathcal{F}f$ is of course centered so that

$$\mathcal{F}(\mathcal{U}f)_{\boldsymbol{k}} = \frac{N_{\text{up}}^2}{N^2}\mathcal{F}(f)_{\boldsymbol{k}}$$

whenever $\boldsymbol{k} \in \left[-\frac{N}{2}, \frac{N}{2} - 1\right]^2$. The scaling factor comes from the fact that the DFT coefficients are scaled by the grid size compared to the Fourier coefficients. Put differently, the upsampling operator interpolates f in S_N, passes through the embedding $S_N \hookrightarrow S_{N_{\text{up}}}$, and projects onto the fine grid:

$$\mathcal{U}f = (I_N f)|_{\Gamma_{\mathrm{up}}}.$$

We return to the problem of interpolation that is conservative to a high order. Consider a single map $\boldsymbol{\chi} := \boldsymbol{\chi}_{[\tau_i,\tau_{i-1}]}$ on a grid. $\boldsymbol{\chi}$ is obtained by some number of steps of a symplectic integrator on the grid. We again write $\boldsymbol{\chi}(\boldsymbol{x})$ for the exact map obtained by the symplectic integrator, without discretizing in space. That is, $\boldsymbol{\chi} = \boldsymbol{\chi}(\boldsymbol{x})|_\Gamma$, and $\boldsymbol{\chi}(\boldsymbol{x})$ is exactly volume-preserving. One possibility for interpolating $\boldsymbol{\chi}$ to a high order of conservation, is to compute $\boldsymbol{\chi}(\boldsymbol{x})$ itself to as high an order as possible. If the truncation error $\|P_N\boldsymbol{\chi}(\boldsymbol{x}) - \boldsymbol{\chi}(\boldsymbol{x})\|$ is not much more than machine precision, then we can recover $\boldsymbol{\chi}(\boldsymbol{x})$ to spectral accuracy with $I_N\boldsymbol{\chi}$. While evaluating this trigonometric interpolation at arbitrary locations is not practical, we can still obtain $I_N\boldsymbol{\chi}$ on a very fine grid by upsampling. We can then interpolate between these fine grid points by some high-degree polynomial interpolation.

We thus define the solution of the method as follows. Denote the operator that interpolates with periodic splines of degree p as $\mathrm{Sp}^{(p)}[\cdot]$. We go into more detail on the spline interpolation in the following section and in Appendix A. We upsample each backward map to a fine grid of size N_{up}, and interpolate with degree p splines: define the interpolants

$$\tilde{\boldsymbol{\chi}}_{[\tau_i,\tau_{i-1}]}(\boldsymbol{x}) = \mathrm{Sp}^{(p)}\left[\mathcal{U}_{N_{\mathrm{up}}}\left[\boldsymbol{\chi}_{[\tau_i,\tau_{i-1}]}\right]\right](\boldsymbol{x})$$

for $i \in \{1, \ldots, m\}$. We recall that the upsampling and spline construction do not apply directly to $\boldsymbol{\chi}$ but rather to the perturbation of the identity, see Remark 3.1. Typical values we will use for upsampling and interpolation degree, in our incompressible Euler implementation, are $N_{\mathrm{up}} = 2048$ and $p = 7$. The final advected quantity at time $t = \tau_m$, which is the solution of the method, is then the smooth function

$$\omega = \omega_0 \circ \tilde{\boldsymbol{\chi}}_{[\tau_1,\tau_0]} \circ \cdots \circ \tilde{\boldsymbol{\chi}}_{[\tau_m,\tau_{m-1}]}.$$

In particular, ω does not live on a fixed grid, and can be sampled arbitrarily. The burden of spatial resolution is effectively passed onto the backward maps.

We now inspect the conservation of this interpolation scheme. Suppose that $\boldsymbol{\chi}$ satisfies the remapping condition $e_{\mathrm{det}}[\boldsymbol{\chi}] = \|\det \mathcal{D}\boldsymbol{\chi} - 1\|_{\ell^\infty} \leqslant \delta_{\mathrm{det}}$. As $\mathcal{D}\boldsymbol{\chi} = (\nabla I_N\boldsymbol{\chi})|_\Gamma$, we then have

$$\|\det \nabla I_N\boldsymbol{\chi} - 1\|_{L^\infty} \lessapprox \delta_{\mathrm{det}}.$$

Since upsampling is just trigonometric interpolation and projection onto a finer grid, we thus also have

$$\|\det \nabla I_{N_{\mathrm{up}}} \mathcal{U}_{N_{\mathrm{up}}} \boldsymbol{\chi} - 1\|_{L^\infty} \lessapprox \delta_{\mathrm{det}}.$$

Thus $\mathcal{U}\boldsymbol{\chi}$ is volume-preserving to the same order as $\boldsymbol{\chi}$, in the spectral Jacobian sense. However, $\mathcal{U}\boldsymbol{\chi}$ lives on a much finer grid. If we now interpolate $\mathcal{U}\boldsymbol{\chi}$ as above and call the interpolant $\tilde{\boldsymbol{\chi}}$, we have

$$\|\det \nabla \tilde{\boldsymbol{\chi}} - 1\|_{L^\infty} \lessapprox \delta_{\mathrm{det}} + \mathcal{O}\left(N_{\mathrm{up}}^{-p}\right)$$

(see the spline interpolation error estimate, Theorem A.1). We typically take $p = 7$ and $N_{\mathrm{up}} = 2048$. Assuming N_{up} is small enough that we may replace the $\mathcal{O}$ above with a constant C (depending on $\boldsymbol{\chi}$ and its derivatives), with these parameters the bound above becomes

$$\|\det \nabla \tilde{\boldsymbol{\chi}} - 1\|_{L^\infty} \lessapprox \delta_{\mathrm{det}} + C \cdot 6.6 \cdot 10^{-24}.$$

We believe that this level conservation error in the interpolant is acceptable, especially given how close to machine precision δ_{det} is.

For this composition strategy to work, we note that we require $e_{\mathrm{det}}[\boldsymbol{\chi}] \leqslant \delta_{\mathrm{det}}$. Or more generally, that the truncation error is small. This is required for the upsampling to approximate the flow of the integrator. Thus, while the spectral Jacobian error does not reflect the volume preservation of the map, we will indeed need a small spectral Jacobian, for the purposes of composition. This also partially motivates the Jacobian remapping condition.

3.3 Velocity interpolation

The spectral Jacobian error of Section 3.1.1, in the first regime before the truncation is significant, looks ideal. In those tests, however, the velocity was known analytically. In the incompressible Euler equations, and in general for the velocity of the backward map, the velocity is only known at grid points, and thus must be interpolated. Interpolating the velocity with a standard cubic scheme does not yield a good spectral Jacobian error, as we shall see. Moreover, the interpolant must be identically divergence-free for the symplectic integrators to produce volume-preserving maps. In this section, we thus construct a velocity interpolant, that looks as much like a nice, analytic, divergence-free vector field as possible.

We emphasize that any divergence-free interpolant, no matter its construction, will yield volume-preserving maps. The result of a symplectic integrator is always exactly volume-preserving. However, here we would also like our maps to have suitable decay in their spectra, to be able to apply our composition strategy from Section 3.2.3. We thus first perform some processing steps on the velocity, so that the resulting maps have sufficiently decaying spectra. We note that we do not apply any projections to the maps themselves. We preprocess the velocity, but leave the responsibility of conservation up to the integrator. In particular, we perform computations in the algebra $\mathfrak{X}_0$, where these are straightforward, rather than in the group SDiff. We believe that the philosophy of working in an algebra rather than working in a group is a valuable one.

3.3.1 Spectra of interpolated compositions

One step of an implicit integrator, when computed by fixed-point iteration, is just a finite number of compositions with the velocity. We are thus interested in what happens to the Fourier spectrum under composition. In general, the result of a composition has unbounded spectrum, even when the functions being composed both have bounded spectrum. This can be seen already with very simple waves: by a Taylor expansion we have

$$e^{ix} \circ e^{ix} = 1 + ie^{ix} - \frac{1}{2}e^{2ix} - \frac{i}{6}e^{3ix} + \cdots ,$$

a Fourier series with unbounded spectrum. Thus, even though our velocity will be represented by finite Fourier series, we should expect the maps to have unbounded spectra after a single time step. Hence, we are interested in the rate of decay of compositions. Composing functions $g \circ \boldsymbol{f}$ known analytically will yield sufficient decay. However, if g is interpolated with a standard cubic scheme for instance, then the decay of the resulting composition is not satisfactory. In general, cubic interpolation schemes yield interpolants that are C^2. The resulting Fourier coefficients thus decay like $\mathcal{O}\left(|\boldsymbol{k}|^{-3}\right)$. This does not yield coefficients near the Nyquist frequency that are below machine precision, on any reasonable grid size. Thus in this section we look for interpolants with quickly-decaying spectra.

In this thesis, all spatial interpolation will be performed with periodic cardinal B-splines. We review the spline basis and interpolant construction for periodic bivariate splines in Appendix A. For the sake of exposition, we repeat the important points here. We denote by $\mathrm{Sp}^{(p)}$ the space of periodic bivariate splines of degree p. We also denote by $\mathrm{Sp}^{(p)}[\cdot] : \mathbb{R}^{\Gamma} \to \mathrm{Sp}^{(p)}$ the interpolation operator. For odd p, $\mathrm{Sp}^{(p)}$ is the space of functions $q \in C^{p-1}(\mathbb{R}^2/2\pi\mathbb{Z}^2)$ such that q reduces to a polynomial inside each grid cell of Γ. We can construct basis functions of arbitrary degree by the Cox-de Boor recursion formula. Moreover given data on the grid, an interpolant in $\mathrm{Sp}^{(p)}$ can be easily constructed by inverting a convolution in Fourier space. Finally, de Boor's algorithm is available for the numerically stable evaluation of the basis functions. Because of these facts, it is straightforward to construct B-spline interpolants of arbitrarily-high degree.

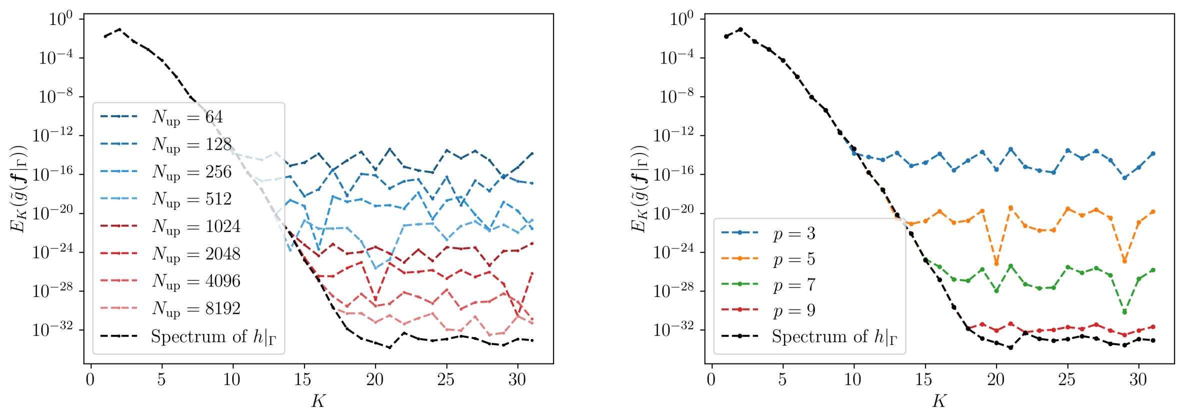
We now return to the problem of lowering the spectrum of a composition. The obvious way to lower the spectrum is to raise the regularity of the interpolant. Elements of $\mathrm{Sp}^{(p)}$, being globally C^{p-1}, have Fourier spectra decaying like $\mathcal{O}\left(|\boldsymbol{k}|^{-p}\right)$. We will thus take the interpolation degree p to be higher than 3; for our Euler implementation we typically take $p = 7$. Another way to lower the spectrum is to construct the interpolant on a finer grid. Though the interpolant is typically an unknown function on a fixed grid, we can still achieve this by upsampling the interpolant. These will be our two ways of lowering the spectrum of interpolants: upsampling and raising the interpolation degree.

To illustrate the effects of upsampling and raising the degree of interpolation, we run a composition test. Define the test functions

$$\boldsymbol{f}(\boldsymbol{x}) = (\cos x_1, \sin x_2), \qquad g(\boldsymbol{x}) = \sin(2x_1)\cos(2x_2),$$

which were chosen arbitrarily. We consider the problem of computing $h := g \circ \boldsymbol{f}$. Take a base grid Γ of $N \times N$ points, on which the interpolant g is sampled, with $N = 64$. We interpolate g by $\tilde{g} := \mathrm{Sp}^{(p)}\left[\mathcal{U}_{N_{\mathrm{up}}}[g|_\Gamma]\right]$, for parameters p and N_{up}, and compute the spectrum of $\tilde{g}(\boldsymbol{f}|_\Gamma)$. We test the effects of upsampling and of the degree separately. In the upsampling test we take $p = 3$ and let N_{up} range from 64 (no upsampling) to 8192. In the degree test we take $p \in \{3, 5, 7, 9\}$, and set $N_{\mathrm{up}} = 64$, so there is no upsampling. The resulting spectra are plotted in Figure 3.3.

We see that we can indeed lower the spectrum of the composition with upsampling and higher-degree interpolation. In our numerical experiments, we shall see that we need to use both upsampling and a high p simultaneously. In our Euler implementation, we typically take $p = 7$ and $N_{\mathrm{up}} = 2048, 4096$. We concede that it is unusual to see such high-degree interpolation in a numerical scheme for a PDE. Indeed, we do not want to take p too high, so as to avoid spurious oscillations. However, we've found that raising the degree is acceptable performance-wise (on a GPU, at least), and has a good payoff in the spectrum. Moreover, we cannot simply raise N_{up} without bound. For $N_{\mathrm{up}} \gtrsim 4096$ we start to see serious performance degradation. Indeed, with



(a) Different upsampling grid sizes, with degree $p = 3$ interpolation.

(b) Different degree p interpolation, no upsampling.

Figure 3.3: Spectra of interpolated compositions, varying upsampling and interpolation degree. In black is the (finite precision) spectrum of the exact composition.

$N_{\text{up}} = 8192$ for instance, performing the velocity upsampling requires

$$8192^2 \text{ points } \times 2 \text{ doubles/complex point } \times 8 \text{ bytes/double } \times 2 \text{ velocity components } = 2 \text{ GiB}$$

of memory, simply to store the FFTs.

3.3.2 Gradient of spline interpolants

For volume preservation, our velocity interpolant must be identically divergence-free. To construct such interpolants we follow the procedure laid out in Section 3.2.2. That is, we interpolate the Hamiltonian, and take the polynomial gradient $\nabla^\perp$, which can be evaluated exactly. This must be done with some care, however. Derivatives of B-splines are finite differences. Since we compute these on a very fine (upsampled) grid, we've found that these finite differences incur significant round-off error. We would like to avoid these numerical difficulties, as we are interested in seeing a spectral Jacobian error as close to machine precision as possible.

In this section, we construct the derivative of a B-spline as a shifted B-spline of a lower degree, and compute the spline coefficients of the derivative by a diagonal operation in Fourier space. This avoids the aforementioned numerical issues caused by the finite differences. It also reduces the number of B-spline evaluations. The latter is an important consideration, since the repeated evaluation of the velocity in the fixed point iteration of the symplectic integrators is the most expensive step of the method.

Suppose we have data f on the grid, and we interpolate by $\tilde{f} := \mathrm{Sp}^{(p)}[f]$. We illustrate here the computation of $\frac{\partial}{\partial x_1}\tilde{f}$. Following the construction in Appendix A, we compute the spline coefficients $a_{\boldsymbol{k}}$ of $\tilde{f}$ and write

$$\tilde{f}(\boldsymbol{x}) = \sum_{0 \leqslant \boldsymbol{k} < N} a_{\boldsymbol{k}} b_{p,k_1}(x_1) b_{p,k_2}(x_2).$$

Here $b_{p,k}$ is a shifted and scaled periodic cardinal B-spline, centered at the point $x_k = kh$, defined in Equation (A.2). A known fact is that the derivative of a cardinal basis spline is a difference of splines of lower degrees, see Equation (A.3). This fact extends to our basis functions $b_{p,k}$, see Equation (A.4). It follows that we can write the polynomial derivative

$$\frac{\partial}{\partial x_1}\tilde{f}(\boldsymbol{x}) = \sum_{0 \leqslant \boldsymbol{k} < N} \frac{a_{\boldsymbol{k}}}{h}\left(b_{p-1,k_1}\left(x_1 + \tfrac{h}{2}\right) - b_{p-1,k_1}\left(x_1 - \tfrac{h}{2}\right)\right) b_{p,k_2}(x_2). \tag{3.8}$$

We could simply evaluate $\nabla\tilde{f}$ by Equation (3.8). However, the terms of Equation (3.8) are finite differences, and so this direct evaluation is undesirable, for the reasons mentioned above. Thus instead of evaluating Equation (3.8) directly, we proceed as follows. Notice that by periodicity of the basis splines, we can rewrite the expression of Equation (3.8) as

$$\frac{\partial}{\partial x_1}\tilde{f}(\boldsymbol{x}) = \sum_{0 \leqslant \boldsymbol{k} < N} c_{\boldsymbol{k}} b_{p-1,k_1}\left(x_1 + \tfrac{h}{2}\right) b_{p,k_2}(x_2), \tag{3.9}$$

with coefficients $c_{\boldsymbol{k}}$ to be found. A straightforward exercise in comparing like terms in Equations (3.8) and (3.9) shows that the spline coefficients of $\tilde{f}$ and $\frac{\partial}{\partial x_1}\tilde{f}$ are related by

$$c_{\boldsymbol{k}} = \frac{1}{h}(a_{\boldsymbol{k}} - a_{k_1+1,k_2}).$$

The above expression is still a finite difference, and we've found that evaluating Equation (3.9) with these coefficients still causes issues numerically. However, by the Fourier shift property we can write a_{k_1+1,k_2} by multiplying $a_{\boldsymbol{k}}$ in frequency space by e^{-ik_1h}. We can thus write an expression for the spline derivative coefficients in Fourier space,

$$\mathcal{F}(c)_{\boldsymbol{k}} = \frac{1 - e^{-ik_1h}}{h}\mathcal{F}(a)_{\boldsymbol{k}}. \tag{3.10}$$

Computing the spline coefficients by Equation (3.10) eliminates the previously mentioned numerical issues.

3.3.3 Mollification

In the case of the incompressible Euler equations, we know that the velocity develops fine details over time. Equivalently, the Fourier spectrum of the velocity rises over time. In the time integration, these fine details are then carried over to the backward maps. This causes the spectra of the maps, and thus their spectral Jacobian error, to rise very quickly. In fact, at higher times where the velocity is highly detailed, we see a large spectral Jacobian error after even a single time step. We thus apply a smoothing to the velocity to eliminate fine details. We note that we do not mollify the maps themselves; this itself would eliminate conservation. We only mollify the velocity data that we then pass to the symplectic integrators.

We perform a simple mollification by truncating Fourier modes. Given a parameter K_{mol}, define the mollification operator $\mathcal{M} : \mathbb{R}^\Gamma \to \mathbb{R}^\Gamma$ which acts on grid data f by

$$\mathcal{F}(\mathcal{M}f)_{\boldsymbol{k}} = \begin{cases} \mathcal{F}(f)_{\boldsymbol{k}} & \text{if } |\boldsymbol{k}| \leqslant K_{\mathrm{mol}}, \\ 0 & \text{otherwise.} \end{cases}$$

This is a low-pass filter, equivalent to convolving with a sinc mollifier in physical space. In practice, we apply the mollification operator to Ψ, where $\nabla^\perp \Psi = \boldsymbol{v}$, before interpolating. However, commuting convolutions and derivatives, this is equivalent to mollifying the velocity directly.

That we are allowed the mollify the velocity requires some justification. In principle, the error incurred by the mollification should be less than the error incurred by the integrators. Moreover, the primary reason for mollifying the velocity is to ensure that fine details are not carried over to the maps. Thus, if in fact the mollification truncates important information in the velocity, we can simply increase the grid size of the maps and increase K_{mol}. Further, truncation of Fourier modes gives a smaller L^2 norm in the velocity, by Parseval's identity. That is, mollification corresponds to a decrease in energy. The result is a flow that is slightly slower in time ([45]).

Finally, we operate under the assumption that the flow is generated by the large scales in the velocity. The fine scales of the velocity contribute very little to the global deformation and can be discarded. This is the assumption made in [45]. The assumption comes from the connection between the CMM for Euler, and the Lagrangian averaged Euler-α (LAE-α) equations. The LAE-α equations are a modification of the incompressible Euler equations, where a Lagrangian averaging (depending on the parameter α) of the velocity is used. The result is that at spatial scales larger than α, solutions accurately resemble those of the regular Euler equations; at scales finer than α, features are averaged out. Since the averaging happens at the level of the velocity, the flow is still incompressible. See [27] for a derivation of the LAE-α equations. It happens that in the CMM for Euler formulation, inserting a Lagrangian averaging of the velocity into the advection-vorticity coupling (1.10) recovers the LAE-α equations ([45]).

3.3.4 Construction of symplectic velocity interpolation

We finally construct a velocity interpolant that will yield maps with suitable spectrum decay, and thus suitable spectral Jacobian. For a velocity $\boldsymbol{v} = \nabla^{\perp}\Psi$, suppose that we know the Hamiltonian Ψ on the grid. Alternatively, that we know $\boldsymbol{v}$ on the grid, but can solve for Ψ by the Poisson problem $-\Delta\Psi = \operatorname{curl}\boldsymbol{v}$. Combining the previous sections, we mollify and upsample Ψ, then apply high-degree interpolation:

$$\tilde{\Psi} := \mathrm{Sp}^{(p)}\left[\mathcal{U}_{N_{\mathrm{up}}}\mathcal{M}_{K_{\mathrm{mol}}}\Psi\right].$$

Our velocity interpolant is then defined as the polynomial gradient $\tilde{\boldsymbol{v}} = \nabla^{\perp}\tilde{\Psi}$.

This consists of four operations: mollification, upsampling, spline construction and spline gradient. These are four operations on the spline coefficients, that are "diagonal" in Fourier space. The upsampling is not technically diagonal, as it changes the grid size. Still, this can be seen as a sequence of pointwise multiplications in Fourier space. These are summarized in Algorithm 3.1. Once these spline coefficients are computed, we are ready to evaluate $\tilde{\boldsymbol{v}}$ by Equation (3.9) and de Boor's algorithm.

Algorithm 3.1 Symplectic velocity interpolant construction.

Input:

- Ψ: grid data, the Hamiltonian of the backward map
- K_{mol}: mollification wavenumber cutoff
- N_{up}: upsample grid size
- p: degree of Hamiltonian interpolant

Output: $c_{1,\boldsymbol{k}}$, $c_{2,\boldsymbol{k}}$: spline coefficients of the components of the velocity interpolant

```
1: $\hat{a} \leftarrow \mathcal{F}(\Psi)$
2: $\hat{a}_{\boldsymbol{k}} \leftarrow \begin{cases} \hat{a}_{\boldsymbol{k}} & \text{if } |\boldsymbol{k}| \leqslant K_{\text{mol}} \\ 0 & \text{otherwise} \end{cases} \quad \forall \boldsymbol{k}$        ▷ mollification
3: $\hat{a} \leftarrow \frac{N_{\text{up}}^2}{N^2} \begin{pmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \hat{a} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{pmatrix}$        ▷ upsampling
4: $\hat{a}_{\boldsymbol{k}} \leftarrow \hat{a}_{\boldsymbol{k}} / \mathcal{F}(K_p)_{\boldsymbol{k}} \quad \forall \boldsymbol{k}$
        ▷ compute spline coefficients by inverting convolution, $K_p$ defined by (A.6)
5: $\hat{c}_{1,\boldsymbol{k}} \leftarrow \frac{1-e^{-ik_2 \cdot 2\pi/N_{\text{up}}}}{2\pi/N_{\text{up}}} \hat{a}_{\boldsymbol{k}} \quad \forall \boldsymbol{k}$        ▷ coefficients of spline derivative, Equation (3.10)
6: $\hat{c}_{2,\boldsymbol{k}} \leftarrow -\frac{1-e^{-ik_1 \cdot 2\pi/N_{\text{up}}}}{2\pi/N_{\text{up}}} \hat{a}_{\boldsymbol{k}} \quad \forall \boldsymbol{k}$
7: $c_1 = \mathcal{F}^{-1}(\hat{c}_1)$, $c_2 = \mathcal{F}^{-1}(\hat{c}_2)$
```

4 Volume-preserving CMM for incompressible Euler

In this section, we derive a volume-preserving Characteristic Mapping Method for the 2D incompressible Euler equations on the torus. Combining the advection-vorticity coupling (1.10) with the backward velocity of Proposition 2.1, the incompressible Euler equations become a system of flows for the forward and backward maps. We integrate these with various explicit and symplectic integrators, using the velocity interpolation scheme of the previous section. The final vorticity can then be sampled using the composition strategy described in Section 3.2.3. We first describe a time integration procedure for the flows. We then derive an error estimate, and perform convergence tests.

4.1 Time integration

Let $\boldsymbol{u}$ be the velocity in the 2D incompressible Euler equations. Let $\boldsymbol{\Phi}$ denote the flow of $\boldsymbol{u}$, and let $\boldsymbol{X}$ be the backward map. Then, as in the advection-vorticity coupling (1.10), we have $\boldsymbol{u} = \nabla^{\perp}\psi$, where the stream function ψ solves $-\Delta\psi = \omega_0 \circ \boldsymbol{X}$, following the Biot-Savart law (1.9). From Proposition 2.1, the backward map $\boldsymbol{X}$ is then the flow of the velocity $\boldsymbol{v} = -\nabla^{\perp}(\psi \circ \boldsymbol{\Phi})$. We have arrived at a system of flows for the forward and backward maps, with coupled velocities:

$$\begin{cases} \partial_t \boldsymbol{X} = \boldsymbol{v} \circ \boldsymbol{X} \\ \partial_t \boldsymbol{\Phi} = \boldsymbol{u} \circ \boldsymbol{\Phi} \\ \psi = -\Delta^{-1}(\omega_0 \circ \boldsymbol{X}) \\ \boldsymbol{v} = -\nabla^{\perp}(\psi \circ \boldsymbol{\Phi}) \\ \boldsymbol{u} = \nabla^{\perp}\psi. \end{cases} \tag{4.1}$$

We now describe a scheme to integrate (4.1), using the symplectic integrators from Section 2.2.2 and the velocity interpolation of Section 3.3.

4.1.1 Notation

In addition to the quantities used in Equation (4.1), we define the following quantity for convenience: write $\Psi = -\psi \circ \boldsymbol{\Phi}$, so that $\boldsymbol{v} = \nabla^{\perp}\Psi$. That is, Ψ acts as a non-autonomous Hamiltonian for the backward map. All of the quantities thus far refer to the true solution.

We then denote by $\boldsymbol{\chi}$ a discretized backward map, and by $\boldsymbol{\varphi}$ a discretized forward map. We compute these on a uniform rectilinear grid $\Gamma_{\boldsymbol{\chi}}$ of $N_{\boldsymbol{\chi}} \times N_{\boldsymbol{\chi}}$ points. The vorticity ω, the stream function ψ and the forward velocity $\boldsymbol{u}$ are computed on a grid Γ_ψ of $N_\psi \times N_\psi$ points. Finally, the velocity $\boldsymbol{v}$ of the backward map, which we will obtain from Algorithm 3.1 by upsampling, lives in a grid $\Gamma_{\boldsymbol{v}}$ of size $N_{\boldsymbol{v}} \times N_{\boldsymbol{v}}$. All grids follow the same definition of Equation (2.9).

There are thus three grids at play. The grid $\Gamma_{\boldsymbol{\chi}}$ is typically the coarsest grid. We would like to keep $N_{\boldsymbol{\chi}}$ as small as possible, as it is on this grid that the bulk of computations will occur. Namely, these are the fixed point iterations in the symplectic integrators, which all require multiple evaluations of the velocity. The next grid Γ_ψ is typically finer than $\Gamma_{\boldsymbol{\chi}}$. The motivation for this is that we need to properly resolve the stream function ψ, which generates the velocities, and accumulates fine details over time. Finally, the grid $\Gamma_{\boldsymbol{v}}$ is the finest; it holds the upsampled backward velocity.

We denote by p the global order of all the integrators at play. We denote the degrees of spatial interpolation by p_{low} and p_{high}. p_{high}, which is typically 7, is used for the interpolation of the symplectic velocity, as motivated in Section 3.3.1. p_{low}, which is typically 3, is used for all other spatial interpolation, whenever we are not concerned with the decay in the spectrum of the resulting quantity. We assume uniform time steps $t_0, t_1, \ldots$, with step size Δt. Finally, we will also be interpolating in time, using Lagrange interpolation. We denote by s the degree of interpolation in time, and by $\ell_{t_i}(t)$ the Lagrange basis polynomial at the node t_i. To differentiate between interpolated quantities and quantities of the true solution, we denote interpolated quantities with a tilde.

$\mathcal{U}$ and $\mathcal{M}$ denote the upsampling and mollification operators, defined respectively in Sections 3.2.3 and 3.3.3.

4.1.2 Time-stepping procedure

Suppose that at the beginning of a time step, $t = t_n$, we have the current submap $\boldsymbol{\chi}_{[t_n,\tau_m]}$ and forward flow $\boldsymbol{\varphi}_{[\tau_m,t_n]}$, as well as the previous, completed submaps $\boldsymbol{\chi}_{[\tau_1,\tau_0]}, \ldots, \boldsymbol{\chi}_{[\tau_m,\tau_{m-1}]}$. We also have access to all the quantities we describe below, from previous time steps. We describe a method to compute $\boldsymbol{\chi}_{[t_{n+1},\tau_m]}$ and $\boldsymbol{\varphi}_{[\tau_m,t_{n+1}]}$.

We begin by approximating the vorticity ω^n at t_n as the pullback of ω_0 by the backward map, which by the semigroup decomposition (1.2) we can write as

$$\omega^n = \omega_0 \circ \boldsymbol{\chi}_{[\tau_1,\tau_0]} \circ \cdots \circ \boldsymbol{\chi}_{[\tau_m,\tau_{m-1}]} \circ \boldsymbol{\chi}_{[t_n,\tau_m]}. \tag{4.2}$$

This is computed as follows. First, $\boldsymbol{\chi}_{[t_n,\tau_m]}$ is upsampled onto the grid Γ_ψ, if necessary. Then, each

$\boldsymbol{\chi}_{[\tau_i,\tau_{i-1}]}$ is interpolated in space; we call this interpolant $\tilde{\boldsymbol{\chi}}_{[\tau_i,\tau_{i-1}]}$. Finally, the composition (4.2) is evaluated on the finer grid:

$$\omega^n := \left(\omega_0 \circ \tilde{\boldsymbol{\chi}}_{[\tau_1,\tau_0]} \circ \cdots \circ \tilde{\boldsymbol{\chi}}_{[\tau_m,\tau_{m-1}]}\right)\left(\mathcal{U}_{N_\psi}[\boldsymbol{\chi}_{[t_n,\tau_m]}]\right). \tag{4.3}$$

The interpolation degree for $\tilde{\boldsymbol{\chi}}_{[\tau_i,\tau_{i-1}]}$ is taken to be p_{low}. The lower-order interpolation may be used here since the velocities resulting from this computation will be mollified. Next, we approximate the stream function by inverting the Biot-Savart law (1.9), as well as the forward velocity:

$$\begin{aligned} \psi^n &:= -\Delta^{-1}\omega^n \\ \boldsymbol{u}^n &:= \nabla^\perp \psi^n. \end{aligned} \tag{4.4}$$

These are computed in frequency space. As usual, the Poisson problem is solved with the integral condition $\int_{T^2} \psi^n = 0$.

Flowing the forward and backward maps in the interval $[t_n, t_{n+1}]$ requires the respective velocities in this interval. However, the velocities at any $t > t_n$ are unknown at this point. We have found that it is sufficient, for the order of accuracy that we desire, to extrapolate the velocities. This is performed by interpolating the velocities over some previous time steps with Lagrange interpolation. We note that there are some alternatives to this extrapolation if a higher order is desired. For instance, one could modify an Adams-Bashforth multi-step method to estimate $\boldsymbol{X}_{[t_{n+1},\tau_m]}$ in terms of the velocities $\boldsymbol{u}^{n-s}, \ldots, \boldsymbol{u}^n$. One would then have an estimate of ω^{n+1}, which gives $\boldsymbol{u}^{n+1}$, circumventing the need for extrapolation. In this work, however, we shall just use the extrapolation method.

At this point, we may choose to flow either the forward or backward map first (or, perhaps, to flow them both simultaneously in a coupled fashion). We choose to first flow the forward map $\boldsymbol{\varphi}$, and then flow the backward map $\boldsymbol{\chi}$, using the updated backward velocity given in terms of $\boldsymbol{\varphi}_{[\tau_m,t_{n+1}]}$. To this end, we mollify the forward velocities $\boldsymbol{u}^i$, and then interpolate them by degree p_{low} splines in space and degree s Lagrange polynomials in time:

$$\tilde{\boldsymbol{u}}(\boldsymbol{x}, t) := \sum_{i=n-s}^{n} \mathrm{Sp}^{(p_{\text{low}})}[\mathcal{M}[\boldsymbol{u}^i]](\boldsymbol{x})\ell_{t_i}(t). \tag{4.5}$$

The mollification is performed in order to smooth out the fine details that $\boldsymbol{u}$ acquires over time, which by our assumption contribute very little to the overall flow. Note that by commuting convolutions and derivatives, this is equivalent to mollifying ψ^n or ω^n. We then apply some integrator of order p to $\boldsymbol{\varphi}_{[\tau_m,t_n]}$ with the velocity $\tilde{\boldsymbol{u}}$, to obtain $\boldsymbol{\varphi}_{[\tau_m,t_{n+1}]}$. This integrator is typically RK2 or RK4.

We remark that we are not concerned with any conservation properties of $\boldsymbol{\varphi}$; we compute $\boldsymbol{\varphi}$ purely to compute the velocity of $\boldsymbol{\chi}$. We can thus make some choices in the interest of performance and accuracy. For instance, we can flow $\boldsymbol{\varphi}$ by some non-conservative (and in particular, explicit) integrator, such as RK4. We are moreover not concerned with the decay in the spectrum of $\boldsymbol{\varphi}$, and so we only use the lower-order spatial interpolation. Finally, we also do not need to take care to construct $\tilde{\boldsymbol{u}}$ to be exactly divergence-free. This allows computing each $\boldsymbol{u}^i$ as a gradient in Fourier space rather than as a polynomial gradient, which saves an order of accuracy in space.

We are now ready to flow the backward map $\boldsymbol{\chi}$. To this end we first interpolate the stream function in space and time:

$$\tilde{\psi}(\boldsymbol{x}, t) := \sum_{i=n-s}^{n} \mathrm{Sp}^{(p_{\mathrm{low}})}[\psi^i](\boldsymbol{x})\ell_{t_i}(t). \tag{4.6}$$

This interpolant is used to extrapolate ψ. The lower-order spatial interpolation is used, as once again, the backward velocity computed from the quantity $\tilde{\psi}$ will be mollified. We then compute the Hamiltonian of the backward map following Proposition 2.1, at time t_{n+1} using the updated forward flow:

$$\Psi^{n+1} := -\tilde{\psi}(\boldsymbol{\varphi}_{[\tau_m, t_{n+1}]}, t_{n+1}). \tag{4.7}$$

From Ψ^{n+1} we compute the velocity used for the symplectic flow of the backward map, as described in Section 3.3. That is, we mollify Ψ^{n+1} with cutoff K_{mol}, upsample it onto the fine grid $\Gamma_{\boldsymbol{v}}$, interpolate it with the higher-order splines, and take the polynomial gradient. This amounts to the following steps:

$$\begin{aligned}
\Psi^{n+1} &\leftarrow \mathcal{M}_{K_{\mathrm{mol}}}[\Psi^{n+1}] \\
\Psi^{n+1} &\leftarrow \mathcal{U}_{N_{\boldsymbol{v}}}[\Psi^{n+1}] \\
\Psi^{n+1}(\boldsymbol{x}) &\leftarrow \mathrm{Sp}^{(p_{\mathrm{high}})}[\Psi^{n+1}](\boldsymbol{x}) \\
\boldsymbol{v}^{n+1}(\boldsymbol{x}) &:= \nabla^{\perp}\Psi^{n+1}(\boldsymbol{x}).
\end{aligned} \tag{4.8}$$

All of these steps are diagonal in frequency space, computed as in Algorithm 3.1. We then interpolate this quantity in time,

$$\tilde{\boldsymbol{v}}(\boldsymbol{x}, t) := \sum_{i=n-s+1}^{n+1} \boldsymbol{v}^i(\boldsymbol{x})\ell_{t_i}(t),$$

and apply one step of an order-p symplectic integrator with the velocity $\tilde{\boldsymbol{v}}$ to $\boldsymbol{\chi}_{[t_n, \tau_m]}$, to finally obtain $\boldsymbol{\chi}_{[t_{n+1}, \tau_m]}$.

Remark 4.1. *In this section we defined in particular the quantities ω^n and $\tilde{\boldsymbol{u}}$. We note that these do not constitute the solution of the method. Rather, these are only intermediate quantities used to flow the backward maps. The solution of the method is still defined to be the composition*

$$\omega_0 \circ \tilde{\boldsymbol{\mathcal{X}}}_{[\tau_1,\tau_0]} \circ \cdots \circ \tilde{\boldsymbol{\mathcal{X}}}_{[\tau_m,\tau_{m-1}]}$$

as defined in Section 3.2.3.

After each time step we check the remapping condition, and if it is violated, we initialize a new submap. Algorithm 4.1 summarizes the time-stepping and remapping processes.

4.1.3 Initializing the method

The above iteration process relies heavily on velocity extrapolation. However, at $t = t_0$, we only know one sample of the velocity in time, namely $\boldsymbol{u}^0 = \nabla^\perp(-\Delta^{-1}\omega_0)$. Given this single sample, we could simply lower the order of interpolation in time near t_0. However, this also lowers the global order of the method. Thus, here we describe a scheme to sample some high-order approximations of the velocity near t_0, so as to bootstrap the method.

For simplicity, we illustrate the process to obtain a second-order approximation of the forward velocity $\boldsymbol{u}$ after one time step. At $t = t_0 := 0$, the forward velocity $\boldsymbol{u}^0 = \nabla^\perp(-\Delta^{-1}\omega_0)$ is known. We then initialize $\boldsymbol{u}^1 = \boldsymbol{u}^0$, and interpolate $\boldsymbol{u}^0$ and $\boldsymbol{u}^1$ in space and time by $\tilde{\boldsymbol{u}}$. We want to approximate $\boldsymbol{X}_{[\Delta t,0]}$. In this case since the final time Δt is known (and small), we can do so by integrating $\tilde{\boldsymbol{u}}$ backward, which is equivalent to integrating

$$\tilde{\boldsymbol{v}}(\boldsymbol{x}, t) := -\tilde{\boldsymbol{u}}(\boldsymbol{x}, \Delta t - t) \tag{4.9}$$

forward, for $t \in [0, \Delta t]$. Thus we apply one step of some explicit, sufficiently-high order integrator, to $\boldsymbol{\chi}_{[0,0]} := \text{id}$ with the velocity $\tilde{\boldsymbol{v}}$ from (4.9), to obtain $\boldsymbol{\chi}_{[\Delta t,0]}$. We set $\omega^1 = \omega_0(\boldsymbol{\chi}_{[\Delta t,0]})$ and $\boldsymbol{u}^1 = \nabla^\perp(-\Delta^{-1}\omega^1)$, and then again interpolate by $\tilde{\boldsymbol{u}}$. We then reset $\boldsymbol{\chi}_{[0,0]} = \text{id}$, integrate $\boldsymbol{\chi}_{[\Delta t,0]}$ again, recompute ω^1 with the new map, and update $\boldsymbol{u}^1 = \nabla^\perp(-\Delta^{-1}\omega^1)$ with the better approximation. We iterate this process until the ω^1 computed by the one-step flow converges; then ω^1 is an at least second-order approximation to ω at $t = \Delta t$.

Once this process converges, we use ω^1 to compute all the quantities from Section 4.1.2 that are necessary for the time-stepping, we interpolate all these quantities in time, and then proceed with the above time-stepping as usual. Note that we do not save the map $\boldsymbol{\chi}_{[\Delta t,0]}$ computed here, as this map is not computed in a volume-preserving fashion.

Algorithm 4.1 One time step of volume-preserving CMM for 2D incompressible Euler.

Input:

- $\boldsymbol{\chi}_{[\tau_1,\tau_0]}, \dots, \boldsymbol{\chi}_{[\tau_m,\tau_{m-1}]}$: previous submaps
- $\boldsymbol{\chi}_{[t_n,\tau_m]}$: current submap
- $\boldsymbol{\varphi}_{[\tau_m,t_n]}$: current forward flow
- ψ^i, $\boldsymbol{u}^i$ and $\boldsymbol{v}^i$: stream functions and forward and backward velocities, respectively, from previous time steps

```
① Compute vorticity and forward velocity
1: $\omega^n \leftarrow \omega_0 \circ \boldsymbol{\chi}_{[\tau_1,\tau_0]} \circ \cdots \circ \boldsymbol{\chi}_{[\tau_m,\tau_{m-1}]} \circ \boldsymbol{\chi}_{[t_n,\tau_m]}$    ▷ computed by Equation (4.3)
2: $\psi^n \leftarrow -\Delta^{-1}\omega^n$, $\boldsymbol{u}^n \leftarrow \nabla^{\perp}\psi^n$    ▷ computed in Fourier space, Equation (4.4)

② Forward flow
3: $\tilde{\boldsymbol{u}}(\boldsymbol{x},t) \leftarrow \text{Interpolate}\,(\mathcal{M}(\boldsymbol{u}^{n-s}),\dots,\mathcal{M}(\boldsymbol{u}^n))$    ▷ degree $p_{\text{low}}$ in space, $s$ in time
4: $\boldsymbol{\varphi}_{[\tau_m,t_{n+1}]} \leftarrow \text{Explicit RK integrator step}\,(\boldsymbol{\varphi}_{[\tau_m,t_n]}, \tilde{\boldsymbol{u}}(\boldsymbol{x},t))$

③ Compute backward velocity
5: $\tilde{\psi}(\boldsymbol{x},t) \leftarrow \text{Interpolate}\,(\psi^{n-s},\dots,\psi^n)$    ▷ degree $p_{\text{low}}$ in space, $s$ in time
6: $\Psi^{n+1} \leftarrow -\tilde{\psi}(\boldsymbol{\varphi}_{[\tau_m,t_{n+1}]}, t_{n+1})$
7: $\boldsymbol{v}^{n+1}(\boldsymbol{x}) \leftarrow \text{Construct symplectic velocity}\,(\Psi^{n+1})$
       ▷ Algorithm 3.1 with parameters $K_{\text{mol}}$, $N_{\boldsymbol{v}}$, $p_{\text{high}}$

④ Backward flow
8: $\tilde{\boldsymbol{v}}(\boldsymbol{x},t) \leftarrow \text{Interpolate}\,(\boldsymbol{v}^{n-s+1}(\boldsymbol{x}),\dots,\boldsymbol{v}^{n+1}(\boldsymbol{x}))$    ▷ degree $s$ in time
9: $\boldsymbol{\chi}_{[t_{n+1},\tau_m]} \leftarrow \text{Symplectic integrator step}\ (\boldsymbol{\chi}_{[t_n,\tau_m]}, \tilde{\boldsymbol{v}}(\boldsymbol{x},t))$

⑤ Remapping
10: if $e_{\text{det}}[\boldsymbol{\chi}_{[t_{n+1},\tau_m]}] > \delta_{\text{det}}$ then
11:     $\boldsymbol{\chi}_{[\tau_{m+1},\tau_m]} \leftarrow \boldsymbol{\chi}_{[t_n,\tau_m]}$    ▷ store previous submap
12:     $m \leftarrow m+1$,  $\boldsymbol{\chi}_{[t_n,\tau_m]} \leftarrow \text{id}$,  $\boldsymbol{\varphi}_{[\tau_m,t_n]} \leftarrow \text{id}$    ▷ reset current maps
13:     go to line 4    ▷ restart current time step
14: end if
```

One can extrapolate the above process to achieve an approximation of the initial velocities to any order, by adding more samples in time and raising the order of the interpolation and integration. Thus for third order, for instance, one would also compute $\boldsymbol{u}^{1/2}$, interpolate $\{\boldsymbol{u}^0, \boldsymbol{u}^{1/2}, \boldsymbol{u}^1\}$, compute $\boldsymbol{\chi}_{[\Delta t/2,0]}$ and $\boldsymbol{\chi}_{[\Delta t,0]}$, and update the velocities accordingly.

4.2 Error analysis

4.2.1 Error estimate

In this section, we provide an asymptotic estimate of the global error of the backward map, computed from the scheme described in Section 4.1.2 . The error is given in the ℓ^∞ norm on the grid. For ease of notation we omit dependence on the grid; thus by the expression $\|\boldsymbol{\chi}_{[t_n,t_0]} - \boldsymbol{X}_{[t_n,t_0]}\|_{\ell^\infty}$, for instance, we understand that $\boldsymbol{X}_{[t_n,t_0]}$ is first projected onto the same grid as $\boldsymbol{\chi}_{[t_n,t_0]}$. For simplicity, we prove the error estimate for one single submap $\boldsymbol{\chi}_{[t_n,t_0]}$ only, with no remapping. Some error analysis has been done in the CMM with remapping and over multiple submaps; see e.g. [40].

We shall make some assumptions while deriving the error bound. First, when computing ω^n by Equation (4.2), we can write

$$\|\omega^n - \omega(t_n)\|_{\ell^\infty} = \mathcal{O}\left(\|\boldsymbol{\chi}_{[t_n,t_0]} - \boldsymbol{X}_{[t_n,t_0]}\|_{\ell^\infty}\right)$$

using a Lipschitz condition for ω_0. When computing ψ^n and $\boldsymbol{u}^n$ by Equation (4.4) however, we shall also assume that

$$\|\psi^n - \psi(t_n)\|_{\ell^\infty}, \quad \|\boldsymbol{u}^n - \boldsymbol{u}(t_n)\|_{\ell^\infty} = \mathcal{O}\left(\|\boldsymbol{\chi}_{[t_n,t_0]} - \boldsymbol{X}_{[t_n,t_0]}\|_{\ell^\infty} + \Delta x^{1/\Delta x}\right). \tag{4.10}$$

This is the assumption made in [45]. It is made rigorous in [40], where the authors derive error estimates in the appropriate Hölder norms. The end result however is the same.

Another assumption is that all the quantities in question are sampled onto sufficiently fine grids, so that there is no significant aliasing errors; and thus that any errors incurred from operations in frequency space are on the order $\mathcal{O}\left(\Delta x^{1/\Delta x}\right)$.

Moreover, we do not include the effects of any mollification in the analysis, similar to the error analysis in [45]. The justification for this is twofold. First, the primary reason for mollifying the backward velocity $\boldsymbol{v}$ is to have sufficient decay in the spectrum of the resulting map $\boldsymbol{\chi}$. However, if in the error analysis we may take the grid for $\boldsymbol{\chi}$ to be arbitrarily fine, then we are not concerned with how quickly the spectrum of $\boldsymbol{\chi}$ decays, and thus we may discount mollification. Secondly, we mainly mollify the stream function ψ. However, the grid of ψ is separate than that of $\boldsymbol{\chi}$, and so we can make Γ_ψ as fine as required to resolve all important details of the flow, independent of the other grids.

For notational convenience we take $N_{\boldsymbol{\chi}} = N_\psi = N_{\boldsymbol{v}}$, and let $\Delta x = \frac{2\pi}{N_{\boldsymbol{\chi}}}$ be the uniform spacing of all three grids. While in practice, $N_{\boldsymbol{\chi}}$ and $N_{\boldsymbol{v}}$ differ by several powers of two, we understand that the upsampling only affects the constant in front of the error bound, and not the order of the method. We also remark that the interpolated velocities $\tilde{\boldsymbol{u}}$ and $\tilde{\boldsymbol{v}}$ may be discontinuous in time at each t_i, but that this does not have an effect on the overall order of the method. This is because the Runge-Kutta integrators here only require regularity in the interior of each time interval $[t_i, t_{i+1}]$ for convergence ([45]).

Finally, we remark that the estimates here are essentially standard ODE and interpolation estimates, due to the decoupling of space and time. The estimate is exactly what one would expect given the order of each integrator and interpolant. Moreover, the powers of Δx and Δt in the estimate are separate. This is in contrast to other implementations of the CMM (e.g. [45]), where the error analysis is slightly more involved and the errors in space and time are coupled.

Theorem 4.2. *Under the assumptions above, the global error of the backward map is asymptotically*

$$\|\boldsymbol{\chi}_{[t_n,t_0]} - \boldsymbol{X}_{[t_n,t_0]}\|_{\ell^\infty} = \mathcal{O}\left(\Delta t^{\min(p,\, s+1)} + \Delta x^{\min(p_{low}+1,\, p_{high})}\right).$$

Proof. Define the error in the backward and forward maps respectively, at any time t_i, to be

$$E^i := \|\boldsymbol{\chi}_{[t_i,t_0]} - \boldsymbol{X}_{[t_i,t_0]}\|_{\ell^\infty}, \qquad E^i_{\boldsymbol{\varphi}} := \|\boldsymbol{\varphi}_{[t_0,t_i]} - \boldsymbol{\Phi}_{[t_0,t_i]}\|_{\ell^\infty}.$$

Given E^n, we derive an asymptotic bound on E^{n+1}.

We first bound the error in the forward map in terms of the error in the backward map. $\tilde{\boldsymbol{u}}$ is defined by Equation (4.5) as the interpolation of the $\boldsymbol{u}^i$, of degree p_{low} in space and s in time. Thus by (4.10) and the spline interpolation error estimate Theorem A.1, we have

$$\begin{aligned}\|\tilde{\boldsymbol{u}}(t) - \boldsymbol{u}(t)\|_{L^\infty} &= \mathcal{O}\left(E^n + \Delta t^{s+1} + \Delta x^{p_{\text{low}}+1} + \Delta x^{1/\Delta x}\right)\\ &= \mathcal{O}\left(E^n + \Delta t^{s+1} + \Delta x^{p_{\text{low}}+1}\right)\end{aligned}$$

for all t reasonably close to t_n. Then, $\boldsymbol{\varphi}_{[t_{n+1},t_0]}$ is obtained from $\boldsymbol{\varphi}_{[t_n,t_0]}$ by one step of an integrator of order p with velocity $\tilde{\boldsymbol{u}}$, so by Grönwall's lemma we can write

$$\begin{aligned}E^{n+1}_{\boldsymbol{\varphi}} &\leqslant \|\boldsymbol{\varphi}_{[t_n,t_0]} - \boldsymbol{\Phi}_{[t_n,t_0]}\|_{\ell^\infty} + \mathcal{O}\left(\Delta t^{p+1} + \Delta t \sup_{t_n \leqslant t \leqslant t_{n+1}} \|\tilde{\boldsymbol{u}}(t) - \boldsymbol{u}(t)\|_{L^\infty}\right)\\ &= E^n_{\boldsymbol{\varphi}} + \mathcal{O}\left(\Delta t^{p+1} + \Delta t E^n + \Delta t^{s+2} + \Delta t \Delta x^{p_{\text{low}}+1}\right).\end{aligned}$$

Iterating this over previous time steps yields

$$
\begin{aligned}
E_{\boldsymbol{\varphi}}^{n+1} &\leqslant \mathcal{O}\left(\Delta t \sum_{i=1}^{n} E^i + \Delta t^p + \Delta t^{s+1} + \Delta x^{p_{\text{low}}+1}\right) \\
&= \mathcal{O}\left(E^n + \Delta t^{\min(p,\, s+1)} + \Delta x^{p_{\text{low}}+1}\right). \qquad (4.11)
\end{aligned}
$$

Next we interpolate the stream function following Equation (4.6), with degrees p_{low} in space and s in time, so again by (4.10) and Theorem A.1 we have

$$\|\tilde{\psi}(t) - \psi(t)\|_{L^\infty} = \mathcal{O}\left(E^n + \Delta t^{s+1} + \Delta x^{p_{\text{low}}+1}\right)$$

for t near t_n. Then following Equation (4.7) we set $\Psi^{n+1} = -\tilde{\psi}(\boldsymbol{\varphi}_{[t_{n+1},t_0]}, t_{n+1})$, and estimate

$$
\begin{aligned}
\|\Psi^{n+1} - \Psi(t_{n+1})\|_{\ell^\infty} &= \|\tilde{\psi}(\boldsymbol{\varphi}_{[t_{n+1},t_0]}, t_{n+1}) - \psi(\boldsymbol{\Phi}_{[t_{n+1},t_0]}, t_{n+1})\|_{\ell^\infty} \\
&\leqslant \|\tilde{\psi}(\boldsymbol{\varphi}_{[t_{n+1},t_0]}, t_{n+1}) - \psi(\boldsymbol{\varphi}_{[t_{n+1},t_0]}, t_{n+1})\|_{\ell^\infty} \\
&\qquad + \|\psi(\boldsymbol{\varphi}_{[t_{n+1},t_0]}, t_{n+1}) - \psi(\boldsymbol{\Phi}_{[t_{n+1},t_0]}, t_{n+1})\|_{\ell^\infty} \\
&\leqslant \|\tilde{\psi}(t_{n+1}) - \psi(t_{n+1})\|_{L^\infty} + C\|\boldsymbol{\varphi}_{[t_{n+1},t_0]} - \boldsymbol{\Phi}_{[t_{n+1},t_0]}\|_{\ell^\infty}
\end{aligned}
$$

where C is a Lipschitz constant for $\psi(t_{n+1})$. It follows that

$$\|\Psi^{n+1} - \Psi(t_{n+1})\|_{\ell^\infty} = \mathcal{O}\left(E^n + E_{\boldsymbol{\varphi}}^{n+1} + \Delta t^{s+1} + \Delta x^{p_{\text{low}}+1}\right). \qquad (4.12)$$

Following Equation (4.8), we upsample and interpolate Ψ^i, this time with the higher-degree splines, and take $\boldsymbol{v}^i$ to be the perpendicular gradient of this interpolation. This derivative of polynomials makes us lose an order in space, by Theorem A.1, and so

$$\|\tilde{\boldsymbol{v}}(t) - \boldsymbol{v}(t)\|_{L^\infty} \leqslant \|\Psi^{n+1} - \Psi(t_{n+1})\|_{\ell^\infty} + \mathcal{O}\left(\Delta t^{s+1} + \Delta x^{p_{\text{high}}}\right) \qquad (4.13)$$

for $t \in [t_n, t_{n+1}]$. We finally apply one step of a symplectic integrator of order p to $\boldsymbol{\chi}_{[t_n,t_0]}$, with velocity $\tilde{\boldsymbol{v}}$, so once more Grönwall's lemma gives

$$E^{n+1} \leqslant E^n + \mathcal{O}\left(\Delta t^{p+1} + \Delta t \sup_{t_n \leqslant t \leqslant t_{n+1}} \|\tilde{\boldsymbol{v}}(t) - \boldsymbol{v}(t)\|_{L^\infty}\right).$$

Inserting equations (4.11), (4.12) and (4.13) and simplifying gives

$$E^{n+1} = \mathcal{O}\left(E^n + \Delta t \cdot \Delta t^{\min(p,\, s+1)} + \Delta t \Delta x^{p_{\text{low}}+1} + \Delta t \Delta x^{p_{\text{high}}}\right).$$

Finally, iterating over previous time steps yields

$$E^{n+1} = \mathcal{O}\left(\Delta t^{\min(p,\, s+1)} + \Delta x^{\min(p_{\text{low}}+1,\, p_{\text{high}})}\right).$$

□

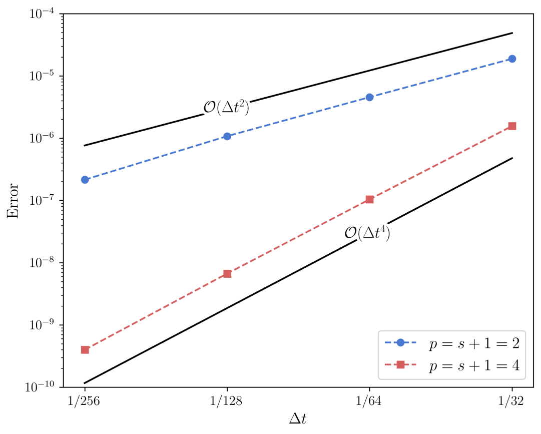
Time step Δt	1/512	Time interpolation order $s+1$	4
Map grid $N_{\boldsymbol{\chi}}$	512	Integrator order p	4
Stream function grid N_ψ	2048	Mollification cutoff K_{mol}	20
Upsampled velocity grid $N_{\boldsymbol{v}}$	2048	Fixed-point convergence $\varepsilon_{\mathrm{sym}}$	10^{-13}
Spatial interpolation degree p_{low}	5	Spatial interpolation degree p_{high}	7

Table 4.1: Parameters for the convergence test baseline solution.

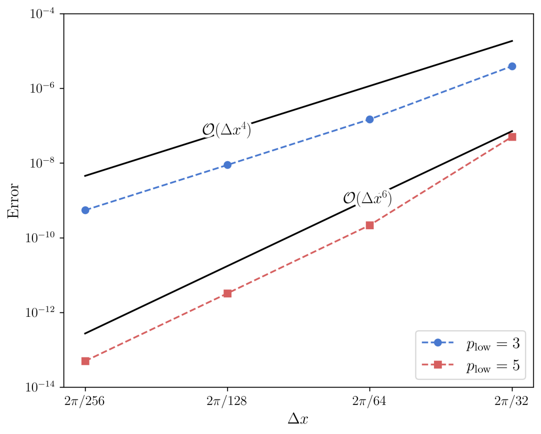
4.2.2 Convergence tests

We perform convergence tests to provide numerical evidence of the error estimate in the previous section. The initial vorticity we use is that of the four-modes test (see [45], [34]). We go into more detail about the four-modes test and the implementation in the upcoming section.

We test the convergence in space and time separately. In all tests we compute a single backward map until $t = 1$. We estimate the backward map error by comparing against a baseline numerical solution with high parameters. These parameters of the baseline solution are shown in Table 4.1. For the time convergence test, we then vary $\Delta t \in \{1/256, \ldots, 1/32\}$ and compute the map error in both second and fourth order in time configurations. The other parameters match those of the baseline solution. The results are plotted in Figure 4.1a. For the space convergence test, we vary $N_{\boldsymbol{\chi}} \in \{32, \ldots, 256\}$ and compute the error in the fourth and sixth order in space configurations. The results are plotted in Figure 4.1b. In all tests we see convergence rates matching those predicted by Theorem 4.2.



(a) Time convergence test.



(b) Space convergence test.

Figure 4.1: Convergence tests: backward map errors compared to the baseline solution at $t = 1$.

5 Numerical experiments

We have developed a GPU implementation of the algorithm proposed in Section 4. In this section we run various tests inspired by the previous iteration of the Characteristic Mapping Method for the 2D incompressible Euler equations [45]. All simulations are run on an NVIDIA GeForce RTX 4070, with 12 GB VRAM and 5888 CUDA cores. The implementation is written primarily in Python, with GPU computations enabled by the library CuPy. Certain portions, namely the interpolation and symplectic integrators, are implemented directly as CUDA kernels in C++. The proposed algorithm, being a semi-Lagrangian scheme, is naturally parallelizable. The only operations that are not pointwise in physical or frequency space are FFTs.

This is not the first GPU implementation of a Characteristic Mapping Method; see for instance [37] and [24]. This implementation was developed independently, however. We also note that efficiency is not necessarily the focal point of this thesis. Thus, while we present some simulation times, although fast, these are almost certainly not optimal.

In this section we perform four-modes tests and a vortex merger test. We first perform a four-modes test with optimal parameters for the spectral Jacobian. We inspect the generated vorticity and Jacobian error of each submap, and in particular investigate the conservation when composing submaps. We then consider a four-modes test where the parameters do not lead to a satisfactory Jacobian error. We finally run a vortex merger simulation, where we consider the Jacobian error in an unstable test and illustrate the arbitrary spatial resolution of the method.

5.1 4-modes test: optimal Jacobian configuration

In this section and the next, we run a four-modes test. The initial vorticity of this test is

$$\omega_0(\boldsymbol{x}) = \cos(x_1) + \cos(x_2) + 0.6\cos(2x_1) + 0.2\cos(3x_1).$$

The four-modes test is ideal for scrutinizing the spatial accuracy of schemes. The spatial complexity of solutions grows at a precise rate; see the radius of analyticity plots of the vorticity in [34]. This test is performed in [45] with the Characteristic Mapping Method, and in [34] with the Cauchy-Lagrangian method. The Cauchy-Lagrangian method is a scheme for the incompressible Euler equations that integrates solutions in time by expanding fluid particle trajectories as Taylor series. By its nature, the Cauchy-Lagrangian method is very high order and can get near machine precision solution and conservation error. However, it is not formulated in terms of the backward map, and thus cannot exploit the semigroup decomposition (1.2). As a result, for reasons of spatial resolution,

Time step Δt	1/32	Time interpolation order $s+1$	2, 4
Map grid $N_{\boldsymbol{\chi}}$	256	Integrator order p	2, 4
Stream function grid N_ψ	256	Mollification cutoff K_{mol}	20
Upsampled velocity grid $N_{\boldsymbol{v}}$	2048	Fixed-point convergence ε_{sym}	10^{-13}
Spatial interpolation degree p_{low}	3	Jacobian remapping condition δ_{det}	10^{-13}
Spatial interpolation degree p_{high}	7		

Table 5.1: Parameters for the four-modes test.

fine grids are required. In [34], the four-modes test is run with the Cauchy-Lagrangian method until $t = 5$, with grid sizes ranging from 1024^2 to 8192^2. In contrast, in [45] with the Characteristic Mapping Method, the test is run until $t = 8$, and the backward maps only need to be computed on a grid of size 128^2. Here, we compute the backward maps on a grid of size $N_{\boldsymbol{\chi}} = 256$, as we've found that we need slightly more resolution to properly resolve the Jacobian.

In this section we perform the test with parameters that we have found to be optimal for the four-modes initial condition. The parameters are listed in Table 5.1. In particular we use degree $p_{\text{high}} = 7$ interpolation and upsampling to $N_{\boldsymbol{v}} = 2048$ for the symplectic velocity interpolation. Moreover, we use a mollification cutoff $K_{\text{mol}} = 20$, which is approximately one sixth of the Nyquist frequency of the map grid. We repeat the test with the second and fourth order integrators.

5.1.1 Vorticity and Jacobian

We first inspect the vorticity generated by the method. A contour plot of the initial vorticity is plotted in Figure 5.1. We then plot the vorticity at times $t = 1, 2, \dots, 8$ in Figures 5.2a-5.2h. For the contour plots we use the maps generated by the fourth order scheme. Following our definition of the solution of the method in Section 3.2.3, each submap $\boldsymbol{\chi}_{[\tau_i,\tau_{i-1}]}$ is interpolated as $\tilde{\boldsymbol{\chi}}_{[\tau_i,\tau_{i-1}]}$. This interpolation uses an upsampled grid of size $N_{\text{up}} = 2048$ and degree 7 splines. To plot the vorticity, we then sample the pullback

$$\omega_0 \circ \tilde{\boldsymbol{\chi}}_{[\tau_1,\tau_0]} \circ \cdots \circ \tilde{\boldsymbol{\chi}}_{[\tau_m,\tau_{m-1}]}$$

on a grid of size $N = 1024$. The vorticity plots are visually essentially identical to those in [45] and [34] with the Characteristic Mapping and Cauchy-Lagrangian methods, respectively.

We next inspect the spectral Jacobian error of each submap. In Figures 5.3a and 5.3b we plot the spectral Jacobian errors of the submaps over time, for the integrators of order 2 and 4

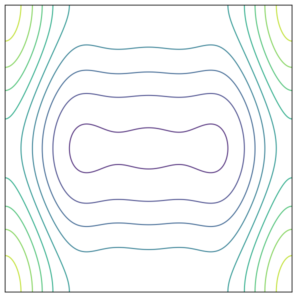
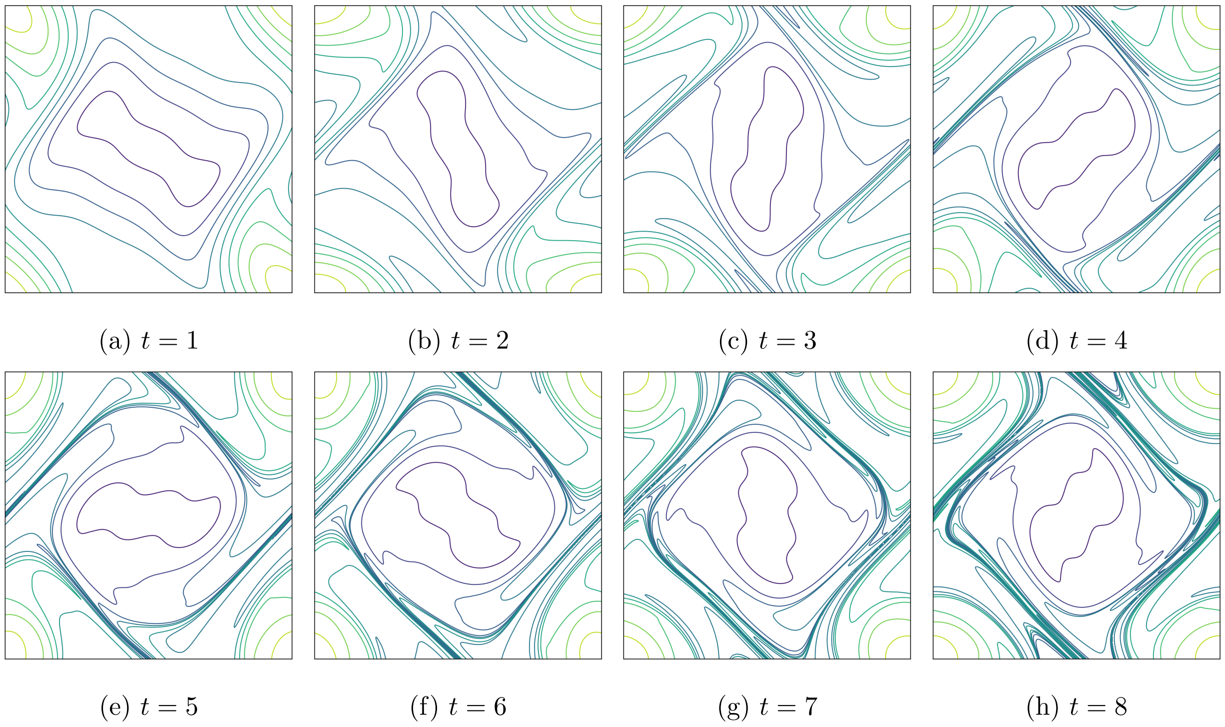


Figure 5.1: Initial vorticity in the four-modes test.



(a) $t = 1$ (b) $t = 2$ (c) $t = 3$ (d) $t = 4$

(e) $t = 5$ (f) $t = 6$ (g) $t = 7$ (h) $t = 8$

Figure 5.2: Vorticities in the four-modes test over time.

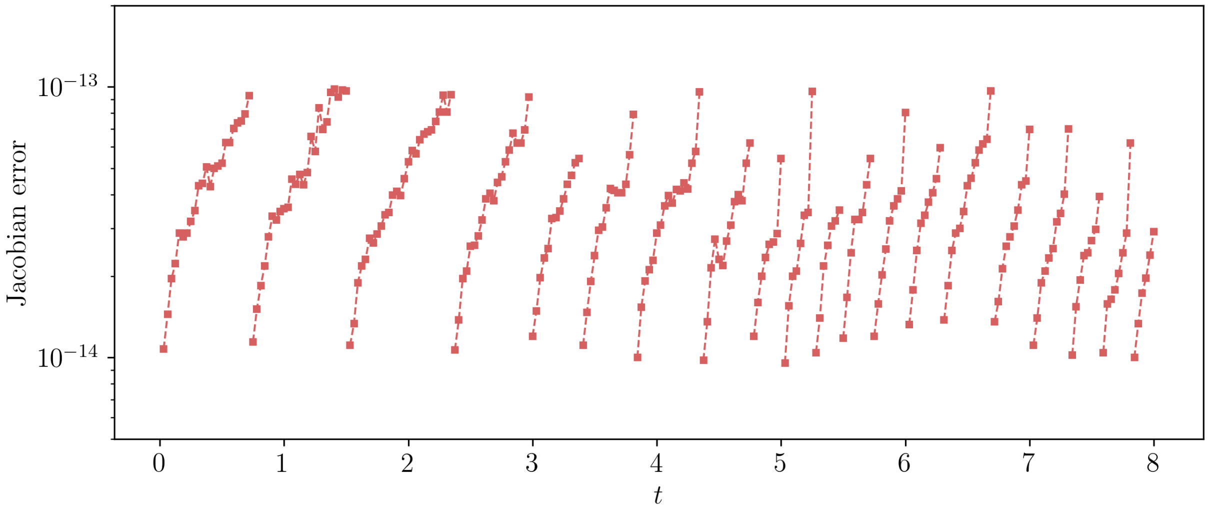
Solution time t	1	2	3	4	5	6	7	8
Runtime, order 2	1.58s	2.25s	3.40s	4.54s	5.79s	7.14s	8.56s	10.09s
Runtime, order 4	1.92s	3.69s	5.52s	7.49s	9.53s	11.75s	14.04s	16.30s
Time to sample ω	0.068s	0.13s	0.20s	0.30s	0.40s	0.53s	0.66s	0.78s

Table 5.2: Runtimes for the four-modes tests.

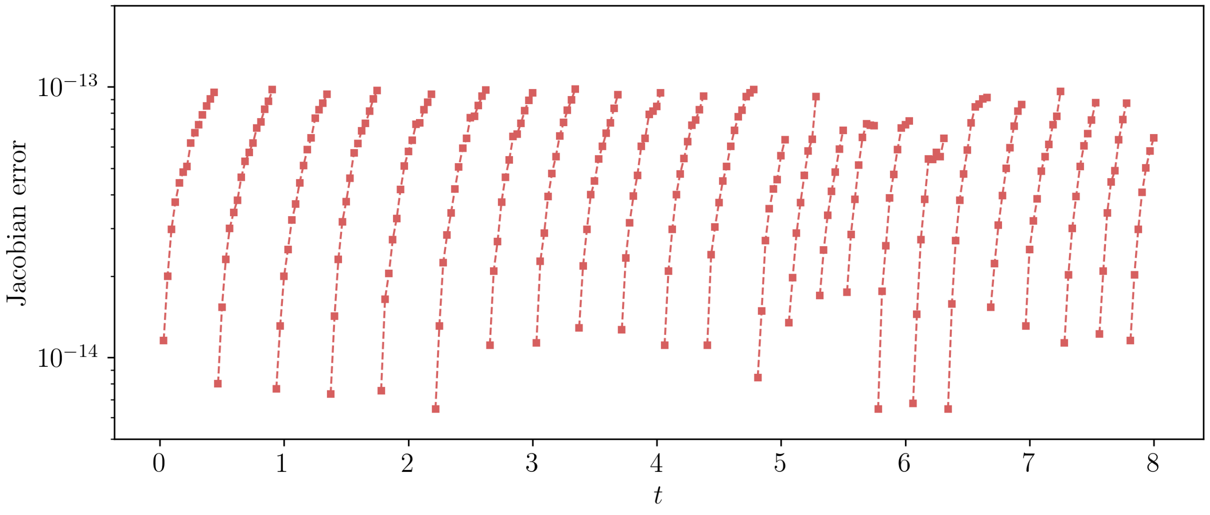
respectively. That is, at each time step, we plot the spectral Jacobian error of the current submap at that time. In these plots we clearly see the remapping process. When the spectral Jacobian error reaches $\delta_{\text{det}} = 10^{-13}$ we initialize a new submap at the identity. We thus see a sequence of "spikes", each representing the evolution of one submap. To compute the solution until $t = 8$, 20 submaps are required for the second order integration and 24 submaps are required for the fourth order configuration. This discrepancy is due to the fact that the fourth order integrator has more intermediate stages, and hence accumulates more rounding error than the second order integrator. In particular, we see again that the Jacobian error is independent of the order of the scheme.

Considering the number of submaps required, we see a serious improvement over the space-time coupled Characteristic Mapping Method. In the CMM for incompressible Euler implementation of [45], 105 submaps are required to reach the final time $t = 8$. Moreover, we see in [45] that the number of submaps required to represent the solution rises significantly at higher times. Here, the number of time steps per submap is more steady over time. Finally, we note that [45] employs a Jacobian remapping condition with $\delta_{\text{det}} = 10^{-4}$. Our condition of $\delta_{\text{det}} = 10^{-13}$ is thus a large improvement. This is not a fair comparison, however, given that [45] does not perform exactly volume-preserving integration.

Finally, we examine the time taken to run these simulations. Table 5.2 shows the total runtime required to reach each $t = 1, 2, \ldots, 8$, for both the second and fourth order configurations. Table 5.2 also shows the amount of time taken to sample the vorticities plotted above. That is, the time taken to upsample each submap to the 2048^2 grid, and evaluate each submap at 1024^2 points with degree 7 interpolation. In particular, we see that we can very quickly sample the final vorticity at arbitrary locations.



(a) Order 2 integration.



(b) Order 4 integration.

Figure 5.3: Four-modes test: spectral Jacobian error of the current submap at each time step.

5.1.2 Conservation of composed submaps and enstrophy

Each submap is exactly volume-preserving by construction. We would now like to examine the conservation properties when we compose the submaps, and when we sample the vorticity using this composition.

We first examine the volume preservation of the composed maps. We take the same submaps $\boldsymbol{\chi}_{[\tau_i,\tau_{i-1}]}$ as in the test of the previous section; that is, the exact same data that generated the above plots and table of runtimes. In particular, each $\boldsymbol{\chi}_{[\tau_i,\tau_{i-1}]}$ lives on a grid of size 256^2. Then, as usual, $\tilde{\boldsymbol{\chi}}_{[\tau_i,\tau_{i-1}]}$ is obtained from $\boldsymbol{\chi}_{[\tau_i,\tau_{i-1}]}$ by upsampling to a grid of size 2048^2 and interpolating with degree 7 splines. We then sample the composition

$$\tilde{\boldsymbol{\chi}}_{[\tau_1,\tau_0]} \circ \cdots \circ \tilde{\boldsymbol{\chi}}_{[\tau_m,\tau_{m-1}]}$$

on various choices of grid size N. We plot the spectral Jacobian error of the resulting composed map over time. To be precise, suppose that at time t_n the current submap is $\boldsymbol{\chi}_{[t_n,\tau_i]}$. Then, we upsample $\boldsymbol{\chi}_{[t_n,\tau_i]}$ to the grid size N, evaluate the composition

$$\left(\tilde{\boldsymbol{\chi}}_{[\tau_1,\tau_0]} \circ \cdots \circ \tilde{\boldsymbol{\chi}}_{[\tau_i,\tau_{i-1}]}\right) \left(\mathcal{U}_N \boldsymbol{\chi}_{[t_n,\tau_i]}\right), \tag{5.1}$$

and compute the spectral Jacobian error. The results are shown in Figure 5.4a. We see a jump in the Jacobian error near $t \approx 0.7$; this is the point at which we switch from having a single submap to having multiple submaps.

Otherwise, here we see the same behaviour as for the spectral Jacobian of a single submap. We at first see a slow growth of the spectral Jacobian error, and then see a sudden regime change where the error quickly rises. The change in regime represents a loss of spatial resolution on the current grid size, and is later for finer grids. In particular, we see that the spectral Jacobian error in the composition is again only a property of the grid, and not of the data. In practice, we would rarely be interested in the Jacobian error of the composed maps, since at later times, the Jacobian might require an extremely fine grid to be properly resolved. However, sampling the composition (5.1) at individual points remains essentially volume-preserving. We again emphasize that the composition uses the submaps on relatively coarse grids. In particular, the time to compute the submaps is the same as in Table 5.2. It is only to estimate the spectral Jacobian that the fine grids are used.

We next examine conservation in the advected vorticity ω. In general, the vorticity has infinitely many conserved integrals. For any measurable function $h : \mathbb{R} \to \mathbb{R}$, the quantity

$$\int_{T^2} h(\omega(\boldsymbol{x}, t)) d\boldsymbol{x}$$

is constant in time. This follows from the change of variables formula and the volume preservation of the backward map. Taking $h(\omega) = \omega^2$ gives a conserved quantity known as the enstrophy ([26]). We thus define the following quantity:

$$\text{Enstrophy conservation error} \ := \ \|\omega\|_{L^2}^2 - \|\omega_0\|_{L^2}^2.$$

This is computed at each time step, on various grid sizes N^2, using the same vorticity sampling strategy as above. The enstrophy is computed in Fourier space by Parseval's identity. The composition (5.1) is again sampled over time, for different grid sizes $N = 256, 512, 1024, 2048$. We then compute the vorticity on the grid by sampling ω_0. The enstrophy error is plotted over time in Figure 5.4b.

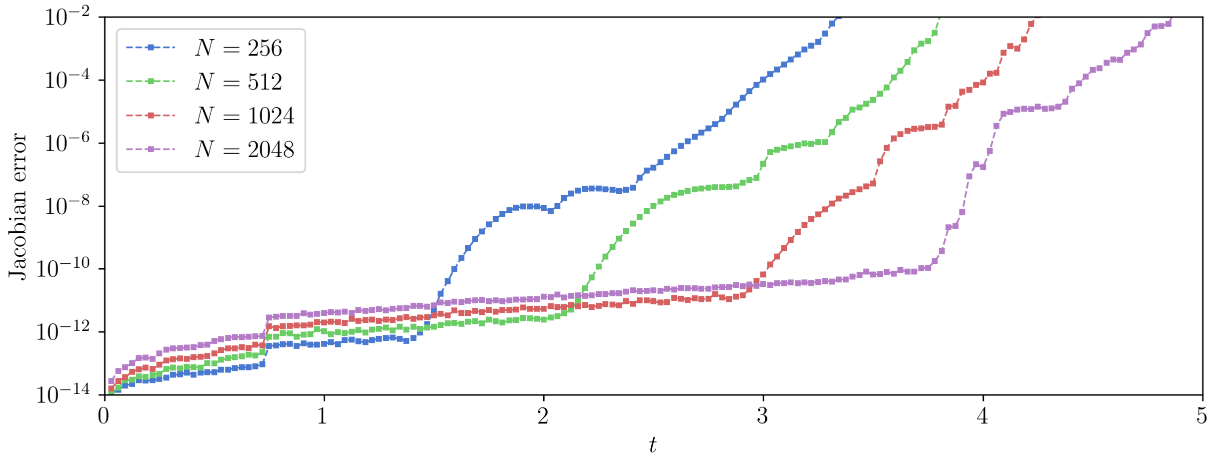
The situation in the enstrophy error is very similar to that of the composed Jacobian. On sufficiently fine grids, the error is essentially at machine precision. For any grid size however, there is a point at which ω is no longer well-represented on the grid, and the error begins to grow. In particular, enstrophy is exactly conserved, but it may take an extremely fine grid to see, numerically, that it is conserved. We see the enstrophy error remain near machine precision longer than the Jacobian error, even for the same grid sizes. This is due to the fact that the Jacobian involves taking a derivative, while the enstrophy does not. The derivative multiplies in Fourier space by the wavenumber, raising the tail of the spectrum.

An important conclusion is the following: our composition strategy is essentially volume-preserving. By extension, integrals of ω are essentially conserved. By this we mean that the sampling at individual points is essentially conservative. Seeing this conservation numerically, however, requires sampling on a full grid. To avoid aliasing errors, this grid must be sufficiently fine. Thus, one may always sample the final vorticity at arbitrary locations. However, when making statements about the final vorticity on a grid, one must be careful to make the grid sufficiently fine.

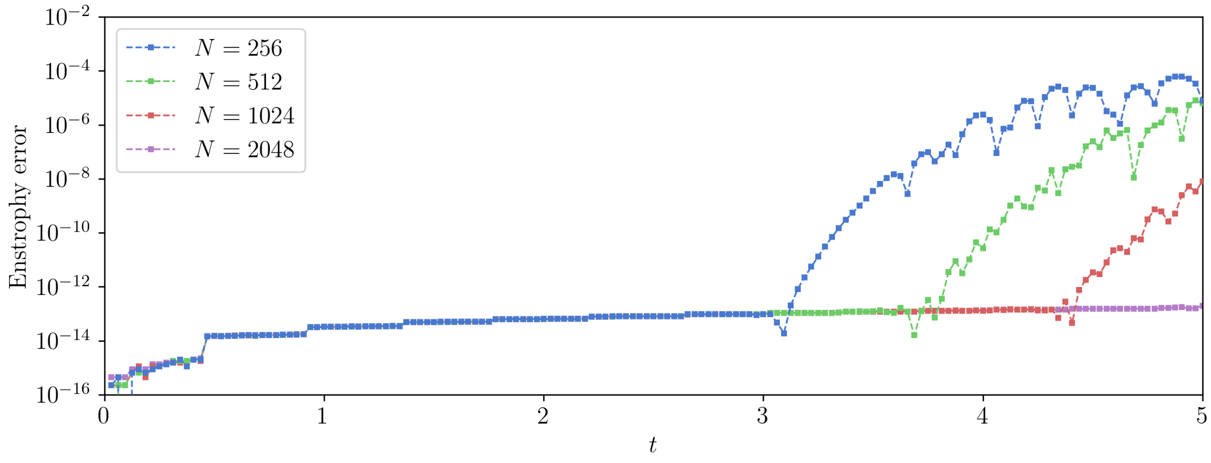
5.1.3 Fine grid comparison

Our composition strategy relies on the assumption that upsampling the submaps yields data that is spectrally close to volume-preserving. In other words, upsampling should give, up to spectral accuracy, the same data as flowing a symplectic integrator on the fine grid. We would like to validate this assumption.

We perform the following test. We take the same configuration as in the previous sections. At each time we have the current submap, on a grid of size 256^2; we denote this submap by $\boldsymbol{\chi}_{256}$. This



(a) Composed spectral Jacobian error.



(b) Composed enstrophy error.

Figure 5.4: Spectral Jacobian and enstrophy errors of composed submaps, sampled over various grid sizes, over time.

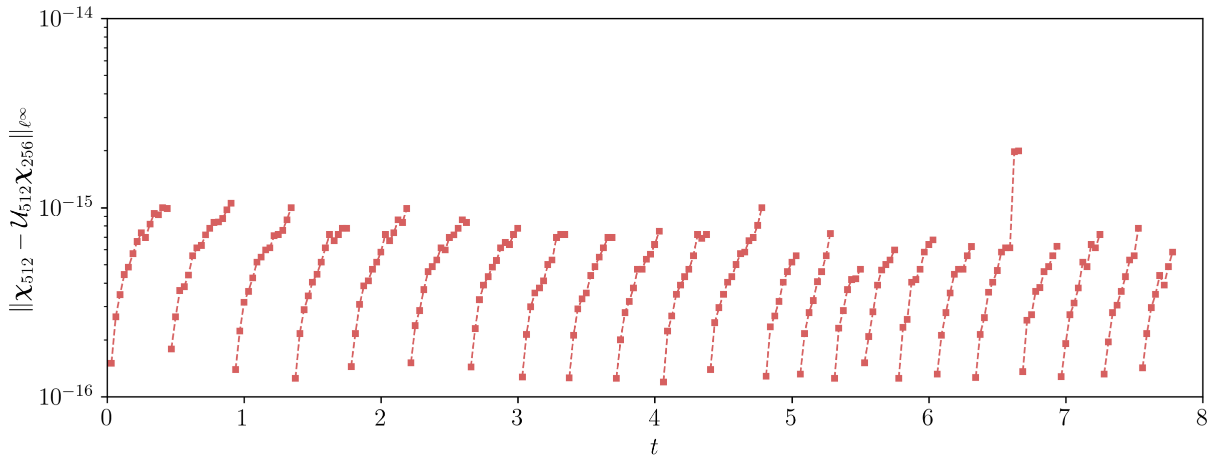


Figure 5.5: Fine grid comparison error (5.2) of the current submap over time.

map is integrated by a backward velocity, which we call $\tilde{\boldsymbol{v}}_{256}$. We then take a map $\boldsymbol{\chi}_{512}$, on a finer grid of size 512^2, and integrate it with the coarse velocity $\tilde{\boldsymbol{v}}_{256}$. We also imitate the remapping of $\boldsymbol{\chi}_{256}$; that is, when the coarse grid remaps, the fine grid remaps as well. Thus, we are computing the same flow as $\boldsymbol{\chi}_{256}$, but on a finer grid. At each time step, we compute the following quantity:

$$\text{Fine grid comparison error} \;:=\; \|\boldsymbol{\chi}_{512} - \mathcal{U}_{512}\boldsymbol{\chi}_{256}\|_{\ell^\infty}. \tag{5.2}$$

That is, we inspect how well upsampling the coarse grid data recovers the fine grid data, and thus how well it recovers the volume-preserving flow as a whole.

The quantity (5.2) is plotted over time in Figure 5.5. We see that upsampling the coarse grid data recovers the fine grid data to incredible accuracy. In fact, the difference is always less than 10^{-13}, which is the iteration tolerance of the fixed-point iteration for the symplectic integrators. Thus, up to machine precision, upsampling can indeed recover the flow of the symplectic integrator.

We note finally that this upsampling comparison could serve as an alternate remapping criterion. Instead of the Jacobian remapping condition, one could also integrate a selection of points on a finer grid. In this example, we would integrate $\boldsymbol{\chi}_{256}$, as well as a small set of points in $\Gamma_{512} \setminus \Gamma_{256}$. We would then remap based on the difference between the offgrid points and $\mathcal{U}_{512}\boldsymbol{\chi}_{256}$. This would be of interest especially for the case where we would conserve some other quantity than the Jacobian.

5.2 4-modes test: suboptimal Jacobian configuration

In this section, we illustrate that our choices of parameters in the previous section are in fact necessary to achieve a satisfactory spectral Jacobian error. We run the four modes test. Unless specified otherwise, the parameters used are the same as in the previous section; that is, those from Table 5.1. Here we replace the Jacobian remapping condition with a fixed remapping strategy. We remap every 8 time steps, regardless of the Jacobian error. We again will plot at each time step the spectral Jacobian error of the current submap at that time.

We first consider the effect of not upsampling the backward velocity. We set $N_{\boldsymbol{v}} = 256$, and still use degree $p_{\text{high}} = 7$ interpolation for the Hamiltonian. The spectral Jacobian error is plotted in Figure 5.6a. We see the spectral Jacobian error quickly leave the region of machine precision. We next consider using lower degree interpolation: we now keep $N_{\boldsymbol{v}} = 2048$, and use degree $p_{\text{high}} = 3, 5$ interpolation. The spectral Jacobian error is plotted in Figure 5.6b. We see that the error is satisfactory only in the very short term for $p_{\text{high}} = 5$, and is never so for $p_{\text{high}} = 3$. Indeed, with $p_{\text{high}} = 3$, the spectrum of the interpolated velocity decays only like $\mathcal{O}\left(|\boldsymbol{k}|^{-2}\right)$, since we lose one order when taking the polynomial gradient. This insufficient decay carries over to the backward map, and so we see a large spectral Jacobian error. The situation is better for $p_{\text{high}} = 5$, where the spectrum decays like $\mathcal{O}\left(|\boldsymbol{k}|^{-4}\right)$, but this is still insufficient. Finally, we consider using weaker mollification. This is an important consideration, as we need to be careful to not remove important details from the velocity. We take $K_{\text{mol}} = 32$, which is one quarter of the Nyquist frequency of the map grid. The spectral Jacobian error is plotted in Figure 5.6c. Here we see that at later times, too many fine details from the velocity are carried over to the maps, raising their spectra. In conclusion, we see that the parameters of Section 5.1 are indeed necessary to achieve an acceptable spectral Jacobian error.

5.3 Vortex merger problem

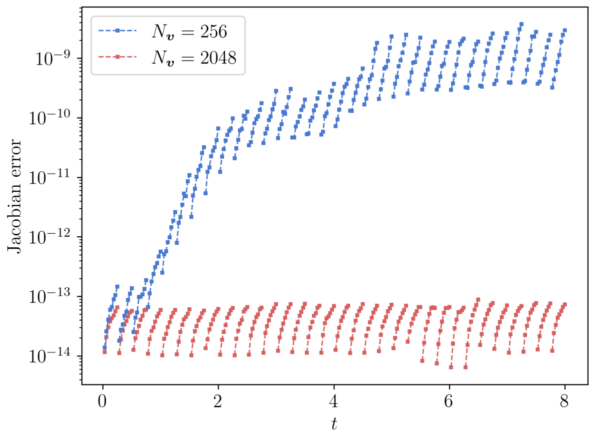
We next run a vortex merger test, imitating the test from [45] with the Characteristic Mapping Method. The initial condition consists of two Gaussians, each with $\sigma = 0.07$, placed at a distance of 0.3 from each other in the periodic unit square. We take a sum of the quickly-decaying Gaussians to make the vorticity periodic. To be exact, we take

$$\omega_0(\boldsymbol{x}) = -\sum_{\boldsymbol{m}\in\mathbb{Z}^2} \left(g(\boldsymbol{x} - (0.15, 0) - \boldsymbol{m}) + g(\boldsymbol{x} + (0.15, 0) - \boldsymbol{m})\right)$$

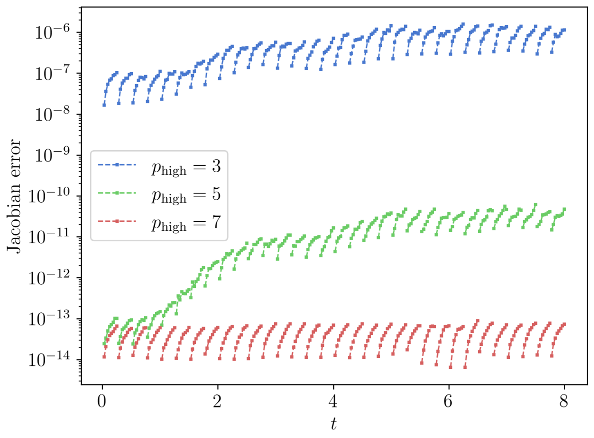
with

$$g(\boldsymbol{x}) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\frac{|\boldsymbol{x}|^2}{\sigma^2}\right).$$

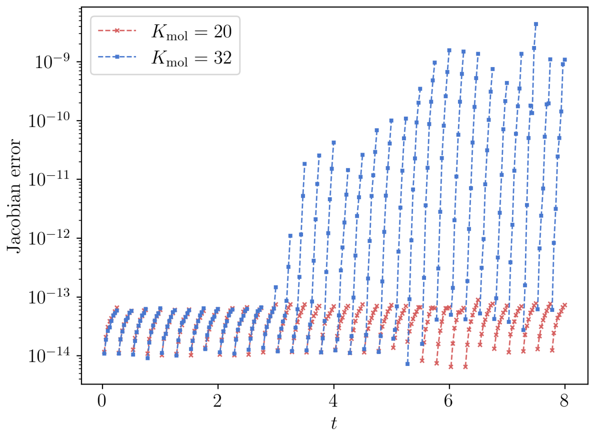
In this configuration we have two vortices of the same sign, and so the vortices have a propensity of merging. In the presence of viscosity, these vortices would merge and eventually vanish under



(a) $N_{\boldsymbol{v}} = 256$ (no upsampling) vs. $N_{\boldsymbol{v}} = 2048$.



(b) $p_{\text{high}} = 3, 5, 7$.



(c) $K_{\text{mol}} = 32$ vs. $K_{\text{mol}} = 20$.

Figure 5.6: Spectral Jacobian errors of submaps over time, with optimal parameters (red) versus some varied parameter (other colours).

Time step Δt	1/32	Time interpolation order $s+1$	4
Map grid $N_{\boldsymbol{\chi}}$	512	Integrator order p	4
Stream function grid N_ψ	512	Mollification cutoff K_{mol}	32
Upsampled velocity grid $N_{\boldsymbol{v}}$	4096	Fixed-point convergence ε_{sym}	10^{-13}
Spatial interpolation degree p_{low}	3	Jacobian remapping condition δ_{det}	$5 \cdot 10^{-13}$
Spatial interpolation degree p_{high}	7		

Table 5.3: Parameters for the vortex merger test.

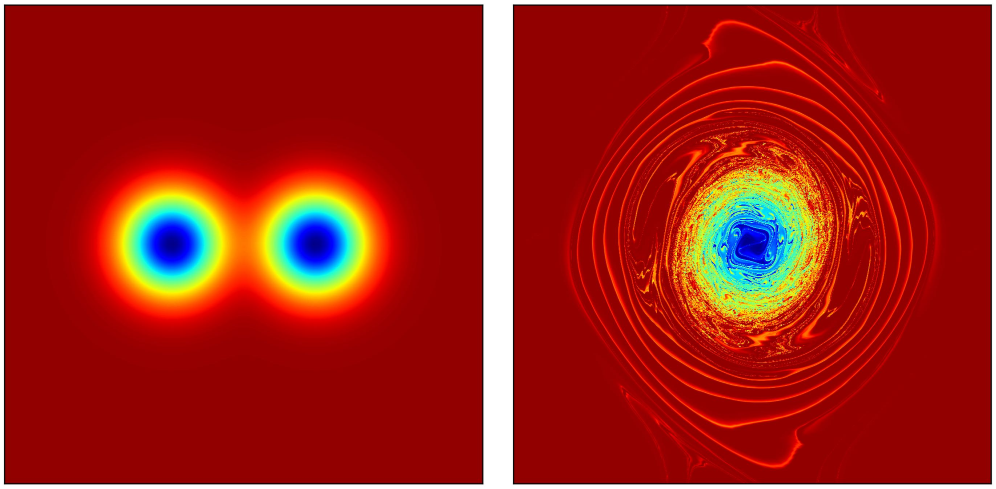
the effect of diffusion. However, since the Euler equations are exactly inviscid, the vortices will not dissipate and will instead keep circling around each other, generating many instabilities and fine scale features as time goes on.

This test is known to be very unstable. Small changes in the parameters yield large differences in the solution. As a result, our plots quickly begin to look different from those of [45], especially when it comes to the fine scales. We include this test for two reasons. First, we show that even in the presence of such large instabilities, we still have conservation of the Jacobian. The conservation is independent of any other qualities of the solution. Next, we would like to illustrate the arbitrary spatial resolution of the method, and the fact that we can represent features much smaller than the grid size. We do this by gradually zooming into the final vorticity plot. We run the test until $t = 25$, as opposed to $t = 20$ as in [45]. This is because the instabilities develop at different rates between the two methods. The parameters for this test are shown in Table 5.3.

5.3.1 Vorticity and Jacobian

We first plot the vorticity. The initial ($t = 0$) and final ($t = 25$) vorticities are plotted in Figure 5.7. The vorticity at various times in between is plotted in Figure 5.8. The vorticity is sampled by the backward maps by the same composition strategy as for the four-modes test. We see the flow develop extremely fine features over time. Representing the solution on a fixed grid would require an exponentially-growing grid size over time. On the other hand, using the semigroup decomposition of the backward map, we are able to compute the solution on a fixed grid, using a linear amount of submaps over time.

We next inspect the spectral Jacobian error of each submap; this is plotted in Figure 5.9. We see that the number of time steps per submap is relatively consistent over time, despite the fine



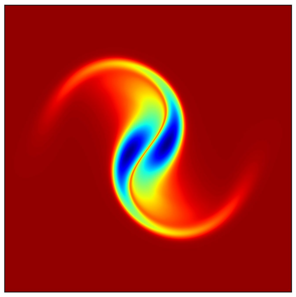
(a) $t = 0$ (b) $t = 25$

Figure 5.7: Vortex merger test: initial and final vorticity.

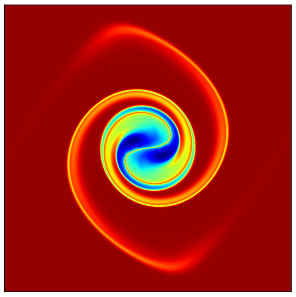
instabilities that develop. This confirms that conservation of the Jacobian is independent of other properties of the solution. A total of 131 submaps is required to reach $t = 25$. This averages to 5.45 submaps per unit of time. To reach $t = 20$, 109 submaps are required. In the comparable test of [45] with the Characteristic Mapping Method, 605 submaps are used to reach $t = 20$. We thus again see that we are able to use less submaps than [45], due to the lack of spatial projections at each time step. In particular, this results in significant savings in memory.

5.3.2 Illustration of arbitrary subgrid resolution

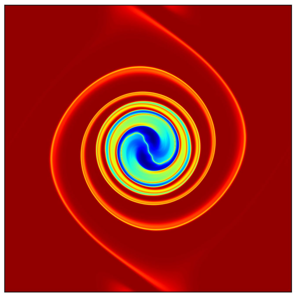
Finally, we illustrate the ability to resolve arbitrary sub-grid features of the solution. We recreate the test performed in [45]. We pick an arbitrary location, and zoom in to the final vorticity by sampling the submap composition on gradually smaller frames. The sampling is done by the same composition strategy as in the four-modes test. The gradual zooms are plotted in Figure 5.10. In particular, the final frame has a width of 2^{-13}. We are thus able to resolve solution features that are much smaller than the size of the grid on which the backward maps are computed. We reiterate that this is possible since the discrete flow is exactly inviscid, and thus the fine scales are not lost to numerical diffusion.



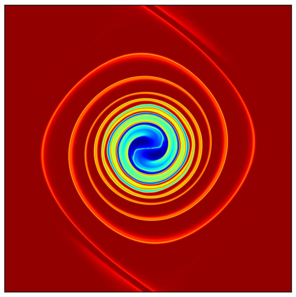
(a) $t = 2$



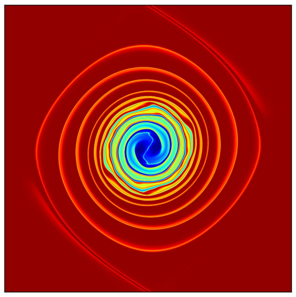
(b) $t = 4$



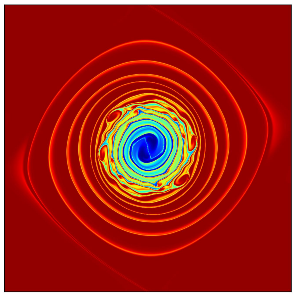
(c) $t = 6$



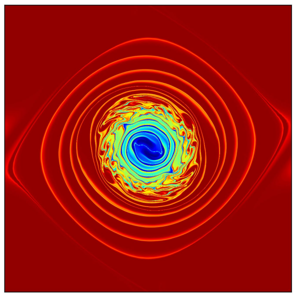
(d) $t = 8$



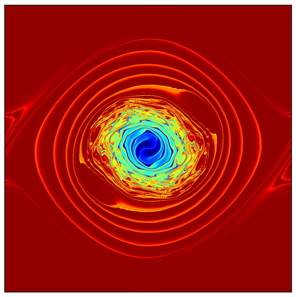
(e) $t = 10$



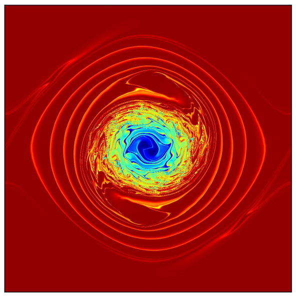
(f) $t = 12$



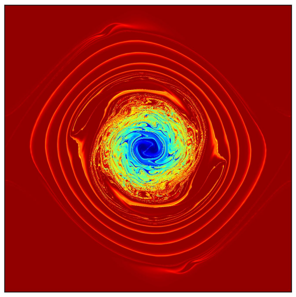
(g) $t = 14$



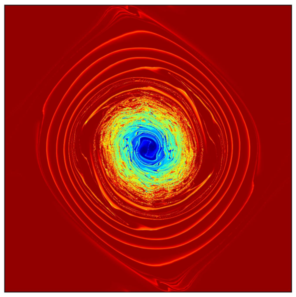
(h) $t = 16$



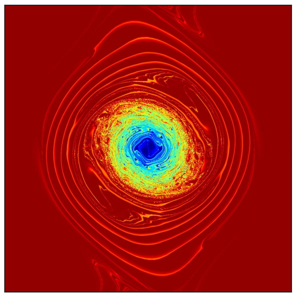
(i) $t = 18$



(j) $t = 20$



(k) $t = 22$



(l) $t = 24$

Figure 5.8: Vortex merger test: vorticity over time.

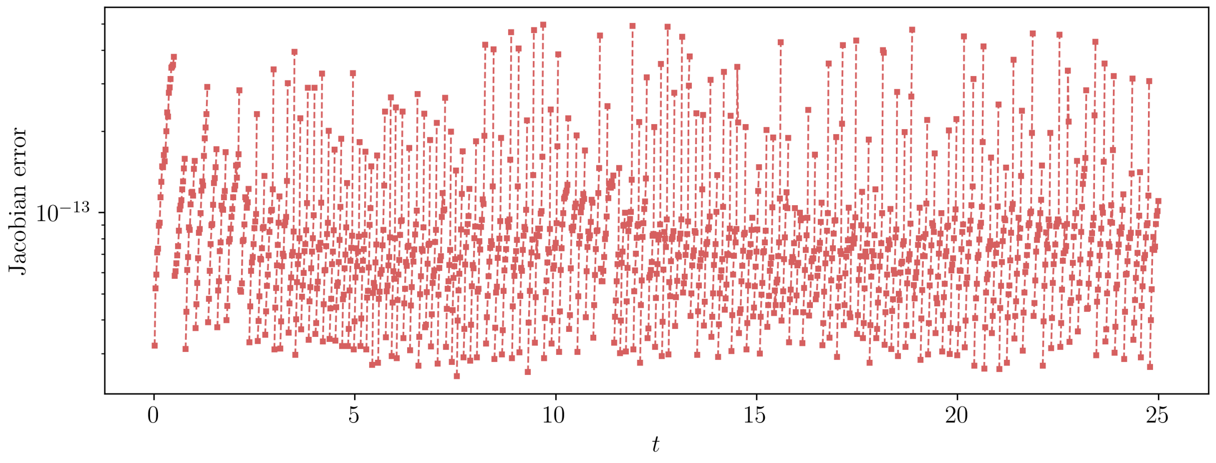
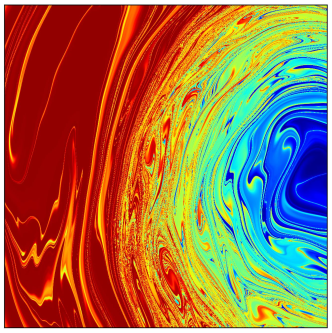
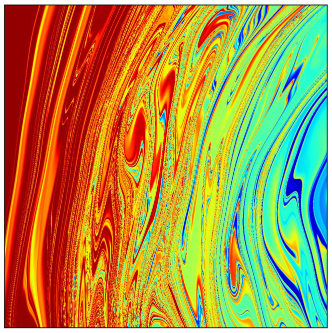
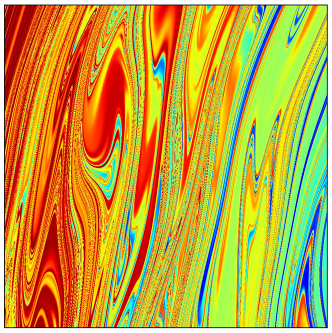
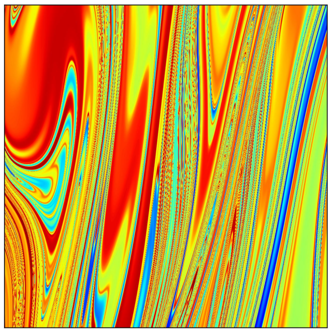


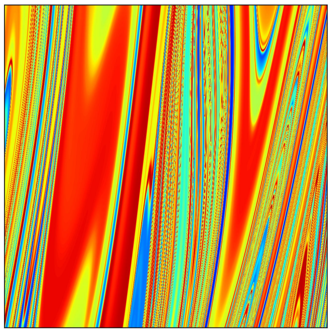
Figure 5.9: Vortex merger test: spectral Jacobian error of the current submap at each time step.

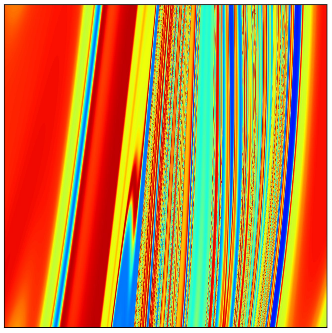

(a) Width $= 2^{-2}$

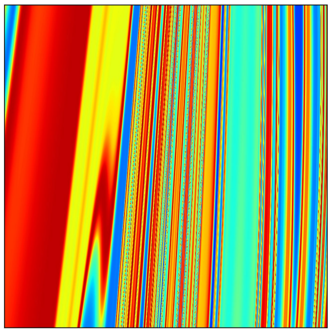

(b) Width $= 2^{-3}$

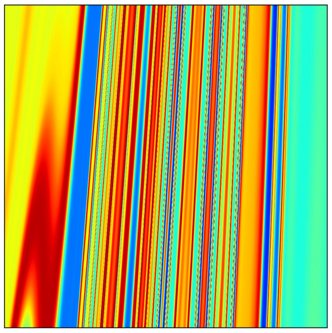

(c) Width $= 2^{-4}$

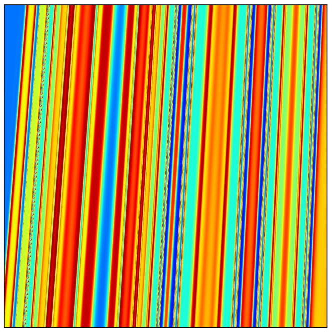

(d) Width $= 2^{-5}$

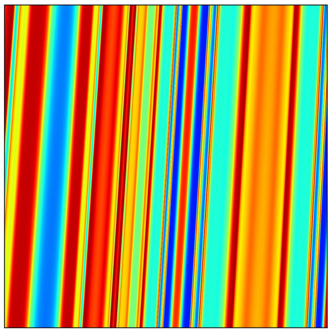

(e) Width $= 2^{-6}$

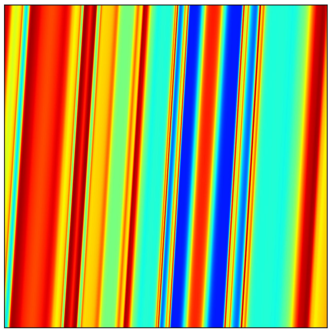

(f) Width $= 2^{-7}$


(g) Width $= 2^{-8}$


(h) Width $= 2^{-9}$


(i) Width $= 2^{-10}$


(j) Width $= 2^{-11}$


(k) Width $= 2^{-12}$

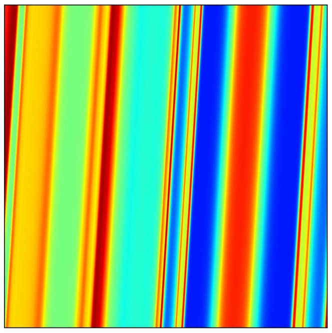

(l) Width $= 2^{-13}$

Figure 5.10: Gradual zoom into vorticity at $t = 25$.

6 Conclusion

We have seen how the time integration method used in the Characteristic Mapping Method for the 2D incompressible Euler equations [45] is incompatible with conservation. This is due to a spatial projection at each time step, which is common for semi-Lagrangian schemes. We have reformulated the CMM to decouple the time integration from space, enabling conservative integration and integration in the presence of boundaries. The time integration of the submaps is now totally Lagrangian; we have, in effect, minimized the overall number of spatial projections required over time. Symplectic integrators were utilized for exactly volume-preserving integration. We saw that the Fourier spectral Jacobian error gives an indication of the spatial resolution of the submaps, motivating a spectral Jacobian-based decomposition of the backward map. We utilized high-degree polynomial interpolation and Fourier upsampling in the interpolation of the velocity to minimize the spatial truncation error in the submaps. As such, the submaps can be interpolated to a high order of conservation, using the same upsampling and high-degree polynomial techniques. Exactly-conservative interpolation is however not possible at this stage. These constructions were applied to simulate the incompressible Euler equations on the torus. In numerical experiments, we saw that the spectral Jacobian error of the submaps is independent of other solution features; in particular, we can resolve very fine-scale features while retaining exact conservation. We finally saw that our interpolation scheme gives an essentially-conservative composition across submaps. The final composition can be sampled pointwise at arbitrary locations. To see conservation, however, the composition must be sampled on a sufficiently fine grid.

There are many possible avenues for further work. One extension to the method that is already in progress is an extension to the unit square. In this domain, the Fourier basis is replaced by the Chebyshev basis, which enables the same upsampling that was utilized here. One could also envision an extension to the sphere S^2, following the geometric formulation of the CMM for 2D incompressible Euler of [40]. In this approach, one could utilize spherical harmonics for upsampling, and the spherical midpoint symplectic integrator ([29]) for volume preservation. A generalization of Proposition (2.1), where we derived the velocity of the backward map, to arbitrary dimensions or geometries, would be useful. Another direction would consider a different equation, such as the Vlasov-Poisson equations, similar to their CMM implementation in [24], as well as other such flows of diffeomorphisms. One could moreover be interested in the conservation of a different quantity than the Jacobian; for this, the alternate remapping criterion suggested in Section 5.1.3 could be of use. Finally, with the ideas of Section 2.1.3, one could consider a CMM over complicated domains with boundary; the flow around an obstacle, for instance.

Appendix A Periodic spline interpolation

For spatial interpolation in this thesis, we use periodic bivariate cardinal B-spline interpolation. In this appendix we review the spline basis and the construction of spline coefficients in Fourier space, and see an error estimate.

A.1 Cardinal spline basis

We begin by defining a convenient set of basis functions. Denote by $M_p : \mathbb{R} \to \mathbb{R}$ the central cardinal B-spline of degree p ([38]). This is defined recursively by

$$\begin{aligned} M_0(x) &= \begin{cases} 1 & \text{if } -\frac{1}{2} \leqslant x < \frac{1}{2} \\ 0 & \text{otherwise} \end{cases} \\ M_{p+1}(x) &= \frac{\frac{p}{2}+1+x}{p+1} M_p\left(x+\tfrac{1}{2}\right) + \frac{\frac{p}{2}+1-x}{p+1} M_p\left(x-\tfrac{1}{2}\right). \end{aligned} \tag{A.1}$$

We note that the p here is off by one compared to [38]. Here we match p to the polynomial degree. Moreover, Equation (A.1) is an instance of the more general Cox-de Boor formula applied to uniformly-spaced knots.

For example, the central cardinal B-splines of degrees 1-3 can be seen to be

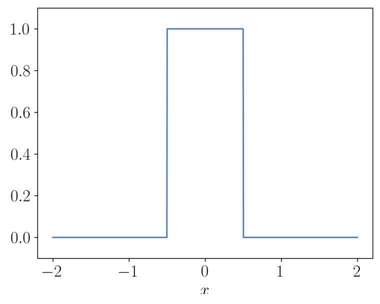
$$M_1(x) = \begin{cases} 1+x & -1 \leqslant x < 0 \\ 1-x & 0 \leqslant x < 1 \end{cases} \qquad M_2(x) = \begin{cases} \frac{1}{2}\left(\frac{3}{2}+x\right)^2 & -\frac{3}{2} \leqslant x < -\frac{1}{2} \\ \frac{1}{2}\left(\frac{3}{2}-x^2\right) & -\frac{1}{2} \leqslant x < \frac{1}{2} \\ \frac{1}{2}\left(\frac{3}{2}-x\right)^2 & \frac{1}{2} \leqslant x < \frac{3}{2} \end{cases}$$

$$M_3(x) = \begin{cases} \frac{1}{6}(2+x)^3 & -2 \leqslant x < -1 \\ \frac{1}{6}(4-6x^2-3x^3) & -1 \leqslant x < 0 \\ \frac{1}{6}(4-6x^2+3x^3) & 0 \leqslant x < 1 \\ \frac{1}{6}(2-x)^3 & 1 \leqslant x < 2 \end{cases}$$

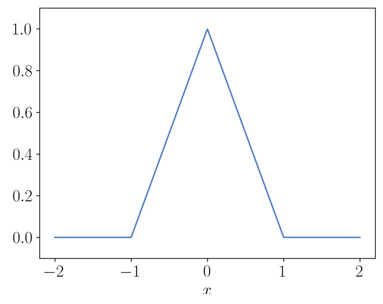
The central cardinal B-splines of degrees 0-3 are shown in Figure A.1a-A.1d.

Some important properties of the central cardinal B-splines, from [38], are as follows. Define the space of splines over $\mathbb{R}$, with the integers as knots, by

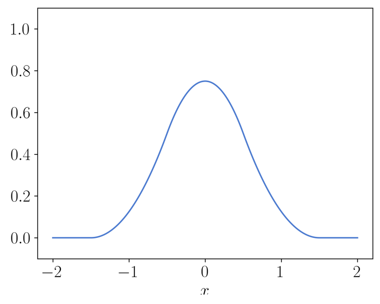
$$\mathcal{S}_p = \{q : q \in C^{p-1}(\mathbb{R}), q \text{ is a polynomial of degree } p \text{ in each } (n, n+1), n \in \mathbb{Z}\}.$$



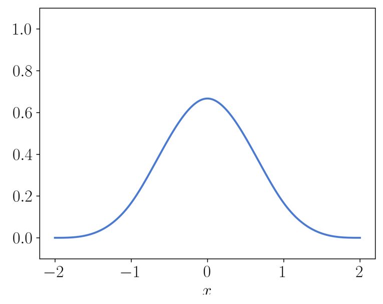
(a) $M_0(x)$



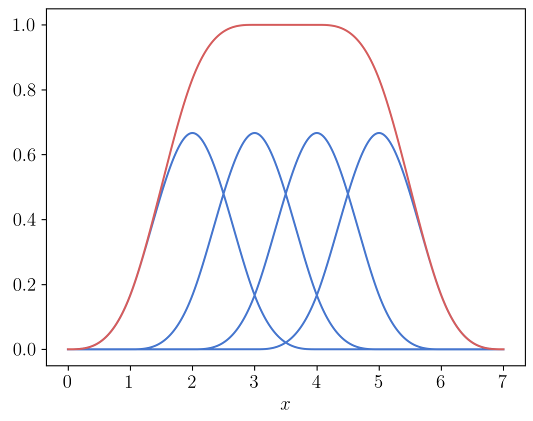
(b) $M_1(x)$



(c) $M_2(x)$



(d) $M_3(x)$



(e) Four central cardinal B-splines of degree 3 shifted by integers (blue), and their sum (red), forming a partition of unity.

Figure A.1: The first central cardinal B-splines M_0-M_3, and the partition of unity property.

Analogously, there is the shifted class of splines,

$$\mathcal{S}_p^* = \{q : \mathbb{R} \to \mathbb{R} : q\left(x + \tfrac{1}{2}\right) \in \mathcal{S}_p\}.$$

Then, for odd p, the set of integer shifts of M_p forms a basis of $\mathcal{S}_p$, and for even p, a basis of $\mathcal{S}_p^*$. In particular each M_p is globally C^{p-1}. Moreover, for any function $f : \mathbb{R} \to \mathbb{R}$, there exist unique spline coefficients a_k such that the piecewise-polynomial

$$q(x) = \sum_{k \in \mathbb{Z}} a_k M_p(x - k)$$

interpolates f at the integers. Next, the integer-shifted M_p form a partition of unity over $\mathbb{R}$; this is illustrated in Figure A.1e. Finally, the support of M_p is $\left[-\frac{p+1}{2}, \frac{p+1}{2}\right]$.

To evaluate the central basis splines, de Boor's algorithm is available ([25]). The recursive algorithm computes $M_p(x)$ in a manner very similar to the Cox-de Boor formula A.1. This algorithm is numerically stable, avoiding the evaluation of finite differences, which are used in other definitions of the basis splines.

We next define a basis of interpolants over a grid on the torus $T^2 = \mathbb{R}^2/2\pi\mathbb{Z}^2$. Take a grid Γ of $N \times N$ points,

$$\Gamma = \{\boldsymbol{x_j} := h\boldsymbol{j} : 0 \leqslant j_1, j_2 < N\},$$

where $h = \frac{2\pi}{N}$. We define a basis for interpolants over Γ as follows. First, define a 1-dimensional basis function $b_{p,k}$ by starting with a degree p cardinal basis spline, and then scaling, shifting and periodicizing:

$$b_{p,k}(x) = \sum_{m \in \mathbb{Z}} M_p\left(\tfrac{x}{h} - k - 2\pi m\right).$$

Then, abusing notation, we also write $b_{p,\boldsymbol{k}}$ for the tensor product of these:

$$b_{p,\boldsymbol{k}}(\boldsymbol{x}) := b_{p,k_1}(x_1) b_{p,k_2}(x_2). \tag{A.2}$$

The $b_{p,\boldsymbol{k}}$ form a basis of the space of periodic bivariate splines of degree p over Γ ([19]), which we denote by

$$\mathrm{Sp}^{(p)} := \mathrm{span}\{b_{p,\boldsymbol{k}}(\boldsymbol{x}) : 0 \leqslant \boldsymbol{k} < N\}.$$

We omit the dependence of $\mathrm{Sp}^{(p)}$ on Γ. We also denote by $\mathrm{Sp}^{(p)}[\cdot]$ the projection of functions or grid data onto this space.

Finally, a well-known ([38]) expression for the derivative of a centered cardinal B-spline is

$$\frac{d}{dx}M_p(x) = M_{p-1}\left(x+\tfrac{1}{2}\right) - M_{p-1}\left(x-\tfrac{1}{2}\right). \tag{A.3}$$

Thus, derivatives of splines are shifted splines of lower degree. It follows that for our spline basis over T^2 we have

$$\frac{\partial}{\partial x_1} b_{p,\boldsymbol{k}}(\boldsymbol{x}) = \frac{1}{h}\left(b_{p-1,k_1}\left(x_1+\tfrac{h}{2}\right) - b_{p-1,k_1}\left(x_1-\tfrac{h}{2}\right)\right) b_{p,k_2}(x_2), \tag{A.4}$$

and analogously for $\frac{\partial}{\partial x_2} b_{p,\boldsymbol{k}}$.

A.2 Construction of periodic spline interpolants

Let $f : T^2 \to \mathbb{R}$ be a function. We would like to interpolate f on the grid Γ by a piecewise-polynomial in $\mathrm{Sp}^{(p)}$. Define a candidate interpolant $q(\boldsymbol{x}) \in \mathrm{Sp}^{(p)}$ as

$$q(\boldsymbol{x}) = \sum_{0 \leqslant \boldsymbol{k} < N} a_{\boldsymbol{k}} b_{p,\boldsymbol{k}}(\boldsymbol{x}).$$

We seek spline coefficients $a_{\boldsymbol{k}}$ so that $q(\boldsymbol{x_j}) = f(\boldsymbol{x_j})$ for all $\boldsymbol{j}$. This condition can be written as

$$f(\boldsymbol{x_j}) = \sum_{0 \leqslant \boldsymbol{k} < N} a_{\boldsymbol{k}} b_{p,\boldsymbol{k}}(h\boldsymbol{j}). \tag{A.5}$$

Equation (A.5) is a linear system for the coefficients $a_{\boldsymbol{k}}$. The matrix in question is block-circulant with circulant blocks, and can be solved with a sparse solver.

The spline coefficients may be solved for much more efficiently, however, by noting following. The basis functions $b_{p,\boldsymbol{k}}$ have a shift property: $b_{p,\boldsymbol{k}}(\boldsymbol{x} + h\boldsymbol{j}) = b_{p,\boldsymbol{k}-\boldsymbol{j}}(\boldsymbol{x})$. Moreover the basis functions are symmetric in $\boldsymbol{k}$ at $\boldsymbol{x} = 0$: $b_{p,-\boldsymbol{k}}(0) = b_{p,\boldsymbol{k}}(0)$, since M_p is even. Thus Equation (A.5) can be rewritten as

$$f(\boldsymbol{x_j}) = \sum_{0 \leqslant \boldsymbol{k} < N} a_{\boldsymbol{k}} b_{p,\boldsymbol{k}-\boldsymbol{j}}(0) = \sum_{0 \leqslant \boldsymbol{k} < N} a_{\boldsymbol{k}} b_{p,\boldsymbol{j}-\boldsymbol{k}}(0).$$

We recognize the right-hand side as a circular convolution. Thus, defining the kernel

$$K_p = \begin{pmatrix} b_{p,(0,0)}(0) & \cdots & b_{p,(0,N-1)}(0) \\ \vdots & \ddots & \\ b_{p,(N-1,0)}(0) & & b_{p,(N-1,N-1)}(0) \end{pmatrix} \tag{A.6}$$

the coefficients $a_{\boldsymbol{k}}$ are related to the grid data by

$$f|_\Gamma = K_p * a.$$

In particular, this means we can efficiently solve for the spline coefficients from the grid data with the FFT by elementwise division in frequency space,

$$a = \mathcal{F}^{-1}(\mathcal{F}(f|_\Gamma) \,/\, \mathcal{F}(K_p)).$$

A.3 Interpolation error

Periodic uniform spline interpolation satisfies the usual estimates expected of polynomial interpolation. Here we list some 1-dimensional estimates in the case of odd-degree splines. These easily extend to multiple dimensions. The following is a theorem of [16].

Theorem A.1. *Let p be odd, $f \in C^{p+1}(S^1)$, and let q be the periodic spline of degree p interpolating f on a uniform grid. Let $0 \leqslant r \leqslant p$. Then*

$$\|f^{(r)} - q^{(r)}\|_{L^\infty} = \mathcal{O}(h^{p+1-r})$$

as the grid size $h \to 0$.

Some best constants for small p have been found. For instance, for the quintic case $p = 5$, it is shown in [19] that

$$\|f^{(r)} - q^{(r)}\|_{L^\infty} \leqslant \varepsilon_r \|f^{(6)}\|_{L^\infty} h^{6-r},$$

with constants $\varepsilon_0 \approx 0.071$, $\varepsilon_1 \approx 0.22$, and so on, independent of the function f to interpolate.

References

[1] S. A. Denisov, *Infinite superlinear growth of the gradient for the two-dimensional Euler equation*, Discrete & Continuous Dynamical Systems - A, 23 (2009), pp. 755–764.

[2] V. I. Arnold and B. A. Khesin, *Topological Methods in Hydrodynamics*, vol. 125 of Applied Mathematical Sciences, Springer International Publishing, Cham, 2021.

[3] C. W. Bardos and E. S. Titi, *Mathematics and turbulence: where do we stand?*, Journal of Turbulence, 14 (2013), pp. 42–76.

[4] J. T. Beale and A. Majda, *High order accurate vortex methods with explicit velocity kernels*, Journal of Computational Physics, 58 (1985), pp. 188–208.

[5] G. Benettin and A. Giorgilli, *On the Hamiltonian interpolation of near-to-the identity symplectic mappings with application to symplectic integration algorithms*, Journal of Statistical Physics, 74 (1994), pp. 1117–1143.

[6] J. Bergmann, T. Maurel-Oujia, Xi-Yuan, Yin, J.-C. Nave, and K. Schneider, *Singularity formation of vortex sheets in 2D Euler equations using the characteristic mapping method*, 2024. arXiv preprint, arXiv:2404.02008.

[7] J. P. Boyd, *Chebyshev and Fourier Spectral Methods: Second Revised Edition*, Courier Corporation, June 2013.

[8] C. Canuto, M. Y. Hussaini, A. Quarteroni, and T. A. Zang, *Spectral Methods in Fluid Dynamics*, Springer, Berlin, Heidelberg, 1988.

[9] F. Cardin, *On symplectomorphisms and Hamiltonian flows*, Journal of Fixed Point Theory and Applications, 24 (2022), p. 33.

[10] E. Celledoni, R. Mclachlan, D. Mclaren, B. Owren, G. Quispel, and W. Wright, *Energy-preserving Runge-Kutta methods*, ESAIM: Mathematical Modelling and Numerical Analysis, 43 (2009), pp. 645 – 649.

[11] P. J. Channell and C. Scovel, *Symplectic integration of Hamiltonian systems*, Nonlinearity, 3 (1990), p. 231.

[12] R. Courant, E. Isaacson, and M. Rees, *On the solution of nonlinear hyperbolic differential equations by finite differences*, Communications on Pure and Applied Mathematics, 5 (1952), pp. 243–255. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/cpa.3160050303.

[13] D. G. Ebin and J. Marsden, *Groups of Diffeomorphisms and the Motion of an Incompressible Fluid*, Annals of Mathematics, 92 (1970), pp. 102–163.

[14] L. C. Evans, *Partial Differential Equations*, American Mathematical Society, 2010.

[15] U. Frisch, *Turbulence: The Legacy of A. N. Kolmogorov*, Cambridge University Press, 1995.

[16] M. Golomb, *Approximation by periodic spline interpolants on uniform meshes*, Journal of Approximation Theory, 1 (1968), pp. 26–65.

[17] T. Grafke, H. Homann, J. Dreher, and R. Grauer, *Numerical simulations of possible finite time singularities in the incompressible Euler equations: Comparison of numerical methods*, Physica D: Nonlinear Phenomena, 237 (2008), pp. 1932–1936.

[18] E. Hairer, C. Lubich, and G. Wanner, *Geometric Numerical Integration: Structure-Preserving Algorithms for Ordinary Differential Equations*, Springer Science & Business Media, Mar. 2013.

[19] C. A. Hall, *Error bounds for periodic quintic splines*, Communications of the ACM, 12 (1969), pp. 450–452.

[20] F. H. Harlow and J. E. Welch, *Numerical Calculation of Time-Dependent Viscous Incompressible Flow of Fluid with Free Surface*, The Physics of Fluids, 8 (1965), pp. 2182–2189.

[21] W. Hundsdorfer and J. Verwer, *Numerical Solution of Time-Dependent Advection-Diffusion-Reaction Equations*, vol. 33 of Springer Series in Computational Mathematics, Springer, Berlin, Heidelberg, 2003.

[22] F. Kang and S. Zai-jiu, *Volume-preserving algorithms for source-free dynamical systems*, Numerische Mathematik, 71 (1995), pp. 451–463.

[23] A. Kiselev and V. Šverák, *Small scale creation for solutions of the incompressible two-dimensional Euler equation*, Annals of Mathematics, 180 (2014), pp. 1205–1220.

[24] P. Krah, X.-Y. Yin, J. Bergmann, J.-C. Nave, and K. Schneider, *A Characteristic Mapping Method for Vlasov–Poisson with Extreme Resolution Properties*, Communications in Computational Physics, 35 (2024), pp. 905–937. Publisher: Global Science Press.

[25] E. T. Y. Lee, *A simplified B-spline computation routine*, Computing, 29 (1982), pp. 365–371.

[26] A. J. Majda and A. L. Bertozzi, *Vorticity and Incompressible Flow*, Cambridge Texts in Applied Mathematics, Cambridge University Press, Cambridge, 2001.

[27] J. E. MARSDEN and S. SHKOLLER, *The Anisotropic Lagrangian Averaged Euler and Navier-Stokes Equations*, Archive for Rational Mechanics and Analysis, 166 (2003), pp. 27–46.

[28] J. E. Marsden and M. West, *Discrete mechanics and variational integrators*, Acta Numerica, 10 (2001), pp. 357–514.

[29] R. McLachlan, K. Modin, and O. Verdier, *A minimal-variable symplectic integrator on spheres*, Mathematics of Computation, 86 (2016), pp. 2325–2344.

[30] O. Mercier, X.-Y. Yin, and J.-C. Nave, *The Characteristic Mapping Method for the Linear Advection of Arbitrary Sets*, SIAM Journal on Scientific Computing, 42 (2020), pp. A1663–A1685.

[31] K. Modin, *Geometric Hydrodynamics: from Euler, to Poincaré, to Arnold.*, arXiv: Mathematical Physics, (2019).

[32] J.-C. Nave, R. R. Rosales, and B. Seibold, *A gradient-augmented level set method with an optimally local, coherent advection scheme*, Journal of Computational Physics, 229 (2010), pp. 3802–3827.

[33] R. A. Norton and G. R. W. Quispel, *Discrete gradient methods for preserving a first integral of an ordinary differential equation*, Discrete and Continuous Dynamical Systems, 34 (2014), pp. 1147–1170.

[34] O. Podvigina, V. Zheligovsky, and U. Frisch, *The Cauchy–Lagrangian method for numerical analysis of Euler flow*, Journal of Computational Physics, 306 (2016), pp. 320–342.

[35] J. F. Price, *12.808 Supplemental Material, Topics in Fluid Dynamics: Dimensional Analysis, a Coriolis tutorial, and Lagrangian and Eulerian Representations*,

2022. https://ocw.mit.edu/courses/res-12-001-topics-in-fluid-dynamics-fall-2023 (accessed 10/06/2024).

[36] S. S. Ray, U. Frisch, S. Nazarenko, and T. Matsumoto, *Resonance phenomenon for the Galerkin-truncated Burgers and Euler equations*, Physical Review E, 84 (2011), p. 016301.

[37] N. Saber, *Two-dimensional Characteristic Mapping Method with inertial particles on GPU using CUDA*, Master's thesis, I2M-AMU, July 2021.

[38] I. J. Schoenberg, *Cardinal Spline Interpolation*, Society for Industrial and Applied Mathematics, 1973.

[39] A. Staniforth and J. Côté, *Semi-Lagrangian Integration Schemes for Atmospheric Models—A Review*, Monthly Weather Review, 119 (1991), pp. 2206 – 2223.

[40] S. Taylor and J.-C. Nave, *A characteristic mapping method for incompressible hydrodynamics on a rotating sphere*, Oct. 2023. arXiv preprint, arXiv:2302.01205.

[41] ——, *A projection-based Characteristic Mapping method for tracer transport on the sphere*, Journal of Computational Physics, 477 (2023), p. 111905.

[42] A. T. S. Wan, A. Bihlo, and J.-C. Nave, *Conservative Methods for Dynamical Systems*, SIAM Journal on Numerical Analysis, 55 (2017), pp. 2255–2285.

[43] A. T. S. Wan and J.-C. Nave, *On the Arbitrarily Long-Term Stability of Conservative Methods*, SIAM Journal on Numerical Analysis, 56 (2018), pp. 2751–2775.

[44] X.-Y. Yin, L. Chen, and J.-C. Nave, *A Diffusion-Driven Characteristic Mapping Method for Particle Management*, SIAM Journal on Scientific Computing, 43 (2021), pp. A3155–A3183.

[45] X.-Y. Yin, O. Mercier, B. Yadav, K. Schneider, and J.-C. Nave, *A Characteristic Mapping Method for the two-dimensional incompressible Euler equations*, Journal of Computational Physics, 424 (2021), p. 109781.

[46] X.-Y. Yin, K. Schneider, and J.-C. Nave, *A Characteristic Mapping Method for the three-dimensional incompressible Euler equations*, Journal of Computational Physics, 477 (2023), p. 111876.

[47] V. I. Yudovich, *The flow of a perfect, incompressible liquid through a given region*, Dokl. Akad. Nauk SSSR, 146 (1962), pp. 561–564.

[48] G. Zhong and J. E. Marsden, *Lie-Poisson Hamilton-Jacobi theory and Lie-Poisson integrators*, Physics Letters A, 133 (1988), pp. 134–139.