

An Affine Semi-Lagrangian Advection Method

by

Jade Marcoux-Ouellet

A thesis
presented to the University of Waterloo
in fulfilment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2018

© Jade Marcoux-Ouellet 2018

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

In computer graphics, the standard semi-Lagrangian advection as in the work of Stam [35] is a widespread unconditionally stable transport scheme used in incompressible fluid solvers. Due to its stability, which disconnects the grid resolution from the time step required to prevent the numerical solution from blowing up, the method provides a good artistic control over the quality-performance trade-off. However, it is also notoriously known to include a great amount of artificial dissipation into the solution, hence destroying fine-scale details, and making simulated fluids appear overly viscous.

Previous research efforts to counteract this unfortunate side effect have been spent notably on reinserting lost small-scale features, and on adapting different parts of the method to improve its accuracy. As part of the latter group, we present an affine semi-Lagrangian advection method, which we refer to as the ASLAM (pronounced “ay-slam”). This novel ASLAM adapts the locally affine descriptor of velocity from the affine particle-in-cell (APIC) method, a hybrid approach by Jiang et al. [27], to the particle-free context of the Eulerian framework. We analyse the ASLAM’s behaviour on a selection of testing scenarios, and evaluate it both qualitatively and quantitatively against a range of competing techniques, showing that it successfully reduces the artificial dissipation arising from standard semi-Lagrangian advection.

Acknowledgements

As I began my journey into this new endeavour (i.e., grad school), I was given three pieces of advice:

1. Do it for yourself.
2. Get your act together.
3. Do not romanticize your thesis.

While thanks are due to the authors of these life-saving tips, the following work would not have been possible without the support and contribution of many.

First and foremost, I am forever grateful to my supervisor Christopher Batty for accepting me as his Master’s student, for invaluable guidance throughout my degree as well as for his precious insights into research and academia. To quote an accidental oxymoron from one of my colleagues: Christopher is solid in fluids.

I am indebted to my fellow members of the computational motion group (CMG) for their much appreciated assistance and joyful comradeship. Thank you: JC, Ryan, Omar, Yipeng, Yu, Michael, and Henry. I will miss our engaged discussions and our colourful post-“reading group” conversations. I feel privileged to have been part of such a dynamic ensemble.

I am also thankful to the many ever-changing occupants of the scientific computation (SciComp) lab for bringing the place to life with their around-the-clock schedules, dispelling any emerging feeling of solitude, as well as to the delightful computer graphics lab (CGL) people, whose enthusiasm for different areas of graphics is contagious.

During my time in Waterloo, I met incredible passionate people who brightened up my life outside the lab. For their time and their friendships, thanks to Sharon, Charu, Monika, Jacob, Danielle, and Nadine. I hope our paths cross again in the future.

Finally, I would like to acknowledge the support and patience of my immediate family: my sister Shany, and my parents, Carolle and Rino.

· ~ ~ ~ ·

The research presented in this document is supported by the Natural Sciences and Engineering Research Council of Canada (NSERC), the Fonds de recherche du Québec –Nature et technologies (FRQNT) as well as the David R. Cheriton Graduate Scholarship Funds.

Dedication

With my unwavering gratitude,
to
Stéphane Côté
and
Meagan Yarmey

Table of Contents

List of Tables	viii
List of Figures	ix
List of Algorithms	xi
Acronyms	xii
Symbols	xiii
1 Introduction	1
1.1 Notation	3
2 Fluids	5
2.1 What is a Fluid?	5
2.2 How to Simulate Fluids?	9
3 Advection	16
3.1 Semi-Lagrangian Advection	16
3.2 Predictor-Corrector Algorithms	27
4 An Affine Semi-Lagrangian Advection Method	31
4.1 Fictitious Particle Set	34
4.2 Particle-to-Grid Weights	37

5	Results	40
5.1	Testing Scenarios and Performance Metrics	40
5.2	ASLAM Parameter Assessment	44
5.3	Comparative Analysis of Advection Methods	55
5.4	Practical Application Scenario	63
5.5	Additional Investigations	64
6	Conclusion	71
6.1	Future Work	72
	References	74
	APPENDICES	78
A	Cubic Interpolation Schemes	79
A.1	Modified Catmull-Rom Interpolation	81
A.2	Bridson’s Cubic Interpolation	83
A.3	Fritsch-Carlson Interpolation	84
A.4	Cubic B-Spline Interpolation	85
A.5	Bicubic Interpolation	87
B	Complementary Figures	90
B.1	ASLAM Parameter Assessment	90
B.2	Practical Application Scenario	94

List of Tables

4.1	Number of Stencil Particles According to Stencil Distribution Type	36
5.1	Description of Testing Icons	41
5.2	L^1 -norm, L^2 -norm, and L^∞ -norm Error Values and Convergence Rates Under Grid Refinement	54
5.3	Abbreviations for the Advection Methods	56
A.1	Weights for the Modified Catmull-Rom Interpolation	82
A.2	Knot Vectors to Generate the Basis Functions for the Cubic B-Spline Interpolation	85
A.3	Basis Functions for the Cubic B-Spline Interpolation	87

List of Figures

2.1	Mapping Between Particles and Grid	13
3.1	Method of Characteristics as a Foundation for Semi-Lagrangian Advection	17
3.2	Back-Tracing Functions	19
3.3	Linear and Cubic Interpolation Mechanisms in 2d	21
3.4	Impact of Higher-Order Accurate Interpolation Scheme on Semi-Lagrangian Advection in 2d	22
3.5	Comparison Between Interpolation Schemes in 1d	24
3.6	Monotonicity Region Based on Derivatives for Cubic Interpolation	25
3.7	Semi-Lagrangian Multi-Step Advection Methods	28
3.8	Comparison Between Semi-Lagrangian Advection Methods in 2d	30
4.1	Schematic Diagram of the ASLAM	32
4.2	Stencil Distribution Types	36
4.3	Weighting Techniques	38
5.1	Analytical Solution to the 1d-TRANSLATION Test	41
5.2	Analytical Solution to the 2d-TRANSLATION Test	41
5.3	Nonlinear Flow Used for the 2d-ROTATION Test	42
5.4	Pseudo-Analytical Solution to the 2d-ROTATION Test	43
5.5	Impact of Parametric Size by Weighting Type for the 1d-TRANSLATION Test	45
5.6	Performance by Weighting Type for the 1d-TRANSLATION Test	46

5.7	Impact of Weighting Type by Parametric Size for the 1d-TRANSLATION Test	47
5.8	Performance by Parametric Size for the 1d-TRANSLATION Test	48
5.9	Impact of Parametric Size and Weighting Type for the 2d-TRANSLATION Test	50
5.10	Impact of Parametric Size and Weighting Type for the 2d-ROTATION Test .	51
5.11	Performance by Parametric Size for the 2d-TRANSLATION Test	52
5.12	Performance by Parametric Size for the 2d-ROTATION Test	53
5.13	Convergence of the ASLAM	55
5.14	Impact of Advection Method for the 1d-TRANSLATION Test	56
5.15	Performance by Metric for the 1d-TRANSLATION Test	57
5.16	Impact of Advection Method for the 2d-TRANSLATION Test	58
5.17	Performance by Metric for the 2d-TRANSLATION Test	59
5.18	Impact of Advection Method for the 2d-ROTATION Test	61
5.19	Performance by Metric for the 2d-ROTATION Test	62
5.20	Initial Conditions for a Rising Plume Scenario	63
5.21	Impact of Advection Method on a Rising Plume Scenario (Beginning) . . .	65
5.22	Comparison Between SL-LIN, ASLAM-0, and ASLAM	66
5.23	Comparison Between MC-LIN, MC-ASLAM, and ASLAM	68
5.24	Comparison Between MC-LIN, MC-ASLAM-CLAMPED, and ASLAM	69
B.1	Full Performance by Weighting Type for the 1d-TRANSLATION Test	91
B.2	Full Performance by Parametric Size for the 1d-TRANSLATION Test	92
B.3	Full Performance by Parametric Size for the 2d-TRANSLATION Test	93
B.4	Full Density Conservation by Parametric Size for the 2d-ROTATION Test . .	94
B.5	Performance by Weighting Type for the 2d-TRANSLATION Test	95
B.6	Performance by Weighting Type for the 2d-ROTATION Test	96
B.7	Impact of Advection Method on a Rising Plume Scenario (End)	97

List of Algorithms

2.1	Simulation Loop for One Time Step	10
2.2	Hybrid Simulation Loop for One Time Step	14
3.1	Standard Semi-Lagrangian Advection	18
3.2	Back and Forth Error Compensation and Correction Method	28
3.3	MacCormack Advection	29
4.1	Affine Semi-Lagrangian Advection	33

Acronyms

APIC affine particle-in-cell

ASLAM affine semi-Lagrangian advection method

BFEC back and forth error correction and compensation

CFD computational fluid dynamics

FEM finite element method

FLIP fluid-implicit-particle

MPM material point method

PDE partial differential equation

PIC particle-in-cell

PolyPIC polynomial particle-in-cell

RHS right-hand side

SPH smooth particle hydrodynamics

Symbols

d Number of spatial dimensions, $d = 1, 2, 3$

\mathbf{x} Position vector, a position in \mathbb{R}^3 such that $\mathbf{x} = (x, y, z)$

\mathbf{u} Velocity vector, a velocity in \mathbb{R}^3 such that $\mathbf{u} = (u, v, w)$

ρ Density, a scalar defined as $\rho = \frac{m}{V}$ where V is volume, a scalar representing the amount of space occupied by a substance

P Pressure, a scalar representing the force applied per unit area, always perpendicular to the surface

\mathbf{f}_b Body force vector, a cumulative force in \mathbb{R}^3 including all forces acting on each element of the fluid

m Mass, a scalar

\mathcal{V} Arbitrary fixed volume

Ω Closed surface of \mathcal{V}

J Density flux, a scalar

∇ Gradient operator, whose vector form in \mathbb{R}^3 is $\nabla = \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z} \right)$

$\nabla \cdot$ Divergence operator, whose vector form in \mathbb{R}^3 is $\nabla \cdot = \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z} \right) \cdot$

$\mathbf{u} \cdot \nabla$ Advection operator, such that $\mathbf{u} \cdot \nabla = u \frac{\partial}{\partial x} + v \frac{\partial}{\partial y} + w \frac{\partial}{\partial z}$, also the source of non-linearity in the fluids' equations of motion

$\frac{D}{Dt}$ Material derivative as defined by Equation 2.3, also known as the Lagrangian derivative

\mathcal{A} Forward step in time according to given back-tracing function and interpolation scheme; the reverse operator \mathcal{A}^r corresponds to a backward step in time according to the same back-tracing function and interpolation scheme

\mathcal{F} Numerical scheme to solve a PDE

Δt A time step such that $\Delta t = t_{n+1} - t_n$

$\Delta \mathbf{x}$ Grid spacing vector, dimension of each grid cell in \mathbb{R}^3 such that $\Delta \mathbf{x} = (\Delta x, \Delta y, \Delta z)$

q Arbitrary advectable scalar quantity; for advectable vector quantities, q can be interpreted as any component of the vector

w Weight, a scalar

\mathbf{i} Index vector, an index in \mathbb{R}^3 such that $\mathbf{i} = (i, j, k)$

p A particle and, when used as a subscript, indicates that the quantity is attached to a particle p

\mathbf{x}_i Grid node position in \mathbb{R}^3

$p \rightarrow \mathbf{i}$ As a subscript, a particle-to-grid quantity or operator

$\mathbf{i} \rightarrow p$ As a subscript, a grid-to-particle quantity or operator

$\bar{\mathbf{x}}$ Normalized position such that $\bar{\mathbf{x}} \in [0, 1]^3$

\mathcal{H} Interpolation kernel, a set of grid nodes required to implement a given interpolation scheme

\mathbf{x}_h Interpolation kernel node position such that $\mathbf{x}_h \in \mathcal{H}$

\mathcal{E} Set of grid nodes immediately enclosing the location to interpolate such that $\mathcal{E} \subset \mathcal{H}$ and $|\mathcal{E}| = 2^d$

\mathbf{x}_e Enclosing grid node position such that $\mathbf{x}_e \in \mathcal{E}$

\mathbf{x}_i^b Back-traced position of grid node position \mathbf{x}_i

\mathbf{C}_p Locally affine descriptor of velocity \mathbf{u} , defined by Equation 2.24

\mathcal{S} A stencil, a set of fictitious particles

\mathbf{x}_s Stencil particle position such that $\mathbf{x}_s \in \mathcal{S}$

\mathbf{x}_s^* Stencil particle position collocated with the grid node such that $\mathbf{x}_s^* \in \mathcal{S}$ and $\mathbf{x}_s^* = \mathbf{x}_i$

\mathbf{x}_s^b Back-traced position of stencil particle \mathbf{x}_s

\mathbf{c} Locally affine descriptor of velocity \mathbf{u} , defined by Equation 4.2

\mathbf{x}_e^b Enclosing grid node position for the back-traced position \mathbf{x}_i^b

ζ Parametric size of stencil \mathcal{S} , $\zeta \in \mathbb{N}^0$

Δ_r Radius of a grid cell's circumscribed circle (or sphere in \mathbb{R}^3) such that $\Delta_r = \frac{\|\Delta \mathbf{x}\|_2}{2}$

D_i Difference between values at x_i and x_{i+1}

δ_i Derivative at x_i

δ_{i+1} Derivative at x_{i+1}

α_i Ratio of δ_i to D_i , always non-negative

β_i Ratio of δ_{i+1} to D_i , always non-negative

\mathcal{M} Subset of the monotonicity region depicted in Figure 3.6

Q Analytical solution of a PDE

L^1 -norm Error metric defined by Equation 5.2, also known as the absolute value norm

L^2 -norm Error metric defined by Equation 5.3, also known as the Euclidean norm

L^∞ -norm Error metric defined by Equation 5.4, also known as the maximum norm

$\%^\rho$ Conservation of density expressed as a percentage, defined by Equation 5.5

Chapter 1

Introduction

Fluids are ubiquitous. We are surrounded by them since around 70% of our planet's surface is covered in water [1], and our bodies, notably the circulatory system, are biological fluid engines. Yet, we often forget that we live in a fluid too: the atmosphere. This mixture of gases which contributes to making our planet a hospitable home is perhaps one fluid we subconsciously care about the most on a daily basis. It drives the weather we experience, from the puffy cumulus clouds to the thunderstorm's show of fury. Evidently, we live in a world dominated by fluids.

In scientific computing, understanding fluids is of interest in the quest to reveal the laws governing our reality. For instance, atmospheric scientists analyse air flows, among others, to improve weather forecasting models [1]. Studying the motion of fluids, compressible or not, is the heart of fluid dynamics, and, in computational fluid dynamics (CFD), numerical analysis obviously drives the investigation. One of the core pillars of CFD are the Navier-Stokes equations, the partial differential equations (PDEs) governing the behaviour of viscous fluids, which have to be approximated by discretization [42]. Several approaches, such as finite differences, finite volumes, and finite element method (FEM), have been developed to numerically solve said PDEs. In this work, we concentrate on incompressible fluids discretized via finite differences, which are both dominant in computer graphics.

Incidentally, in graphics applications, fluids are pursued for the completeness they bring to a scene due to their participation in many natural phenomena. Initial attempts at computer-generated fluids focused on depicting them based on appearance or visual properties using nonphysics-based procedures, for example procedural approaches to generate textures mapped on simple primitives. While these predominantly phenomenological techniques did not bother with dynamics, before long fluid animations appeared. The first

animations of fluids were based on simplified wave models¹ [17] for liquids, and spectral-inspired algorithms coupled with the advection-diffusion equation² [36, 37] for gases.

Early methods subsequently evolved to account for the PDEs underlying the motion of fluids. For example, in Kass and Miller [28], the shallow water approximation is used in lieu of the Navier-Stokes equations, and Gamito et al. [21] propose a particle-grid model to simulate gases based on the vorticity equation, which is derived from the Navier-Stokes equations. The pioneering work of Foster and Metaxas [16], which presents a complete discretization of the full tridimensional Navier-Stokes equations using finite differencing, established that challenging fluid behaviours could be reproduced realistically and with acceptable efficiency for computer animation. Numerous methods have since been developed to tackle a variety of phenomena, such as, but not limited to, films and foams, multi-phase flows, and high-viscosity fluids.

One important landmark in computer graphics is standard semi-Lagrangian advection by Stam [35], which nowadays dominates over purely Eulerian (i.e., grid-based) transport schemes. In CFD, semi-Lagrangian schemes have a long history predating Stam [35], but they were not especially prevalent, except perhaps in the atmospheric sciences [38]. Hence, Stam [35] deserves credit for wide-spreading stable fluids in the graphics community. Since it does not rely on persistent particles, the method is particularly suited to the simulation of gases, which have no interface. The most highlighted advantageous feature of semi-Lagrangian advection is undeniably its unconditional stability, which decouples the time step from the spatial resolution, as opposed to explicit/direct-solving schemes from CFD literature. This uncoupling provides a better control for balancing quality and efficiency, an asset in artistically-driven pipelines common in graphics.

However, semi-Lagrangian advection comes with a major drawback: the introduction of massive amounts of artificial dissipation. This numerical artefact decreases the visual quality of the simulation by deleting small-scale features, and increasing the fluid's apparent viscosity. Whilst this is inappropriate for engineering applications requiring high-accuracy, it can be compensated for, in the context of computer-generated animations, by adding fictitious forces, like the vorticity confinement suggested by Fedkiw et al. [14]. Further, the technique's different components may be modified (e.g., high-order interpolation schemes, etc.) to reduce numerical dissipation. On the other hand, Lagrangian approaches, based on persistent particles, and hybrid methods, which leverage the particle-provided benefits of the former, can perform advection much better (i.e., with less dissipation) than their Eulerian counterparts, including semi-Lagrangian ones.

¹They since have grown into quite sophisticated real-time techniques too as in Jeschke and Wojtan [26].

²Spectral methods also have spawn a number of neighbouring techniques, such as Cui et al. [8].

In light of this, we develop a new technique, the **affine semi-Lagrangian advection method** (ASLAM)³, to perform advection under an Eulerian setting. Our technique, which adapts the locally affine velocity descriptor from the hybrid approach of Jiang et al. [27] into an advection update formula, exhibits less numerical dissipation than the standard semi-Lagrangian advection of Stam [35]. Moreover, unlike Lagrangian and hybrid methods, it annihilates the need to track a persistent set of particles throughout the simulation.

Chapter 2 begins with a slightly more formal introduction to fluids than the above rough sketch. In particular, fluids and their dynamics are explained along with simple mathematical definitions. Then, we explain the foundations of numerical simulations in the context of computer graphics applications, and survey major fluid paradigms. Subsequently, in Chapter 3, we review different semi-Lagrangian advection techniques and interpolation schemes, and discuss their limitations. This is followed by the introduction of our proposed method, the ASLAM, in Chapter 4, where we present our dissipation-reducing approach. Afterwards, in Chapter 5, we examine some results illustrating the performance of our method, investigate its convergence, and show a comparative analysis with a selected subset of existing techniques as well as ASLAM-based variations. Lastly, we conclude by providing some orientation towards future investigation avenues in Chapter 6.

1.1 Notation

Before getting to the core of this thesis, we quickly survey some important notation-related details. Lowercases are used for the majority of scalar quantities while **bold** lowercases are exclusively saved for vectors, which are always column matrices, even if written as $\mathbf{x} = (x, y, z)$ for brevity. **Bold** uppercases are used for matrices, and, when elements are explicitly enumerated, they are enclosed by square brackets.

Parentheses are reserved for grouping, precedence rules, components of vector quantities, binomial coefficients, and to denote a function’s or an operator’s parameters if the function or operator has either more than one parameter, or a parameter not corresponding to a spatial position. For spatial parameters, a vertical line rather than parentheses is used to indicate the position at which a value is taken. Therefore, $q|_{\mathbf{x}}$ refers to the value of quantity q at the position \mathbf{x} .

Unless mentioned otherwise, subscripts correspond to positional information, and superscripts correspond to temporal information. In particular, positional information located on particles is subscripted with p while that related to the grid is assigned numerous

³Pronounced “ay-slam”.

subscripts contextually, with **i** being the most common. Regarding temporal information, n is the elected superscript, but note that regular n has other meanings described in due time.

A list of symbols along with short descriptions is available in the preamble. Further, they are grouped in such a way that related symbols are closer to one another, and organized according to the number of occurrences as well as their order of appearance in the text.

Chapter 2

Fluids

In the sections hereafter, we first outline the properties of the fluids we are interested in, including a derivation of the equations involved along with an intuitive description, albeit rooted in physics, of their origin. Then, we dive into the details of the particular solver used to simulate fluids in this thesis, and review important hybrid approaches in computer graphics.

2.1 What is a Fluid?

When asked this section’s eponymous question, one common answer is to define fluids qualitatively as substances that flow. Moreover, while the term “fluid” most often brings to mind liquids, like water, it encompasses gases too¹. The flowing behaviour exhibited by fluids results, in part, from differences in pressure.

Pressure is a crucial actor in defining fluids quantitatively. As a matter of fact, the study of fluids is often divided according to whether the studied substance is compressible or not. Despite the fact that most fluid substances are truly, in some respect, compressible, this behaviour is only relevant to a handful of cases in computer graphics applications. Bridson lists sonic booms and blast waves as such scenarios [4], both of which would be better depicted artistically than physically for entertainment purposes. Indeed, artistic

¹Plasma, the so-called “fourth phase” of matter, is also considered as to be a fluid [33]. However, we will not discuss it further due to its conducting nature; plasma both affects and is affected by electro-magnetic forces. Thus, its behaviour is not entirely described by the model presented herein.

representations of these phenomena are more visually significant to the general public than accurate physically-based animations.

CFD is the study of fluid flow problems using numerical methods, and at its core are the Navier-Stokes equations, the widely-accepted fluid model in physics and mathematics, which describes the motion of viscous fluids. In this document, we limit our fluid simulations to incompressible inviscid fluids, a subset of fluid substances whose behaviours fall under the Navier-Stokes equations. In particular, having discarded viscosity, we adopt the incompressible Euler equations as a description of their behaviour:

$$\nabla \cdot \mathbf{u} = 0, \tag{2.1}$$

$$\frac{D}{Dt} \mathbf{u} = -\frac{1}{\rho} \nabla P + \mathbf{f}_b. \tag{2.2}$$

where \mathbf{u} is velocity, ρ is density, P is pressure, and \mathbf{f}_b combines all body forces applied to the fluid. Along with boundary conditions, the above PDEs (i.e., Equation 2.1 and Equation 2.2) govern the behaviour of fluids in terms of motion; hence, they are known as the equations of motion. Equation 2.2 includes the material derivative $\frac{D}{Dt}$, defined as

$$\frac{D}{Dt} = \frac{\partial}{\partial t} + (\mathbf{u} \cdot \nabla), \tag{2.3}$$

where $\mathbf{u} \cdot \nabla$ is the advection operation, the source of non-linearity in the equations of motion. The material derivative provides the relationship between the rates of change of fluid-relevant quantities under the Eulerian and Lagrangian viewpoints, which are further explained later. Several paths can be taken to derive the governing PDEs [4, 41, 42]. We choose to present a brief origin story of these equations using a conservative approach.

Equation 2.1 is the mass continuity equation, and represents conservation of mass. Imagine an arbitrary fixed volume of fluid \mathcal{V} with closed surface Ω , then the density flux J across the boundary is

$$J = \int_{\Omega} \rho \mathbf{u} \cdot d\Omega. \tag{2.4}$$

Using the divergence theorem, the above expression can be turned into a volume integral

$$J = \int_{\mathcal{V}} \nabla \cdot (\rho \mathbf{u}) d\mathcal{V}. \tag{2.5}$$

The fluid flow across the boundary must be balanced out by a change in mass m . Yet, since the volume is fixed at \mathcal{V} , the change in mass presents itself as a change in density.

Therefore, we have that the fluid loss is also

$$J = -\frac{dm}{dt} = -\frac{d}{dt} \int_{\mathcal{V}} \rho d\mathcal{V} = -\int_{\mathcal{V}} \frac{\partial \rho}{\partial t} d\mathcal{V}. \quad (2.6)$$

Thus, the density flux J can be explained either by fluid flowing across the boundary, or by a change in density. Equating Equation 2.5 to Equation 2.6 yields

$$\int_{\mathcal{V}} \nabla \cdot (\rho \mathbf{u}) d\mathcal{V} = -\int_{\mathcal{V}} \frac{\partial \rho}{\partial t} d\mathcal{V}. \quad (2.7)$$

Because the volume \mathcal{V} is arbitrary, we have that

$$\nabla \cdot (\rho \mathbf{u}) = -\frac{\partial \rho}{\partial t}. \quad (2.8)$$

Given that we are interested in incompressible fluids, the variation in density ρ is negligible, and density is taken to be constant. As a result, Equation 2.8 is transformed into Equation 2.1, which is often referred to as an incompressibility constraint. This equation requires the velocity field to be divergence-free, meaning that it has neither sources nor sinks. Hence, one can intuitively see how a divergence-free velocity field enforces conservation of mass.

Equation 2.2 is the momentum equation, and represents conservation of momentum. In physics, momentum \mathbf{p} quantifies the amount of motion, and is defined as

$$\mathbf{p} = m\mathbf{u}. \quad (2.9)$$

Thereby, Equation 2.2 is also a vector equation. Similarly as for Equation 2.1, let us consider an arbitrary fixed volume of fluid \mathcal{V} with closed surface Ω . The total amount of momentum is

$$\int_{\mathcal{V}} \rho \mathbf{u} d\mathcal{V}. \quad (2.10)$$

We can use the material derivative $\frac{D}{Dt}$ to express the variation of momentum over time. Mimicking Newton's second law of motion $\mathbf{f} = m\mathbf{a}$, where \mathbf{f} is force and \mathbf{a} is acceleration, we have

$$\frac{D}{Dt} \int_{\mathcal{V}} \rho \mathbf{u} d\mathcal{V} = \int_{\mathcal{V}} \mathbf{f} d\mathcal{V}. \quad (2.11)$$

Whereinbefore, \mathbf{f} is the force per unit volume, but, once again, since the volume \mathcal{V} is

arbitrary, and density ρ is constant, we have that

$$\frac{D}{Dt}\mathbf{u} = \frac{\mathbf{f}}{\rho}. \quad (2.12)$$

Equation 2.12 is a general form of the momentum equation (i.e., Equation 2.2) prior to defining relevant forces. Different forces can come into play when dealing with fluids. Here, we focus on pressure and body forces.

The pressure gradient determines the force arising from fluid pressure. This force is a surface stress. It acts on the surface of different regions of the fluid unlike body forces, such as gravity, which are applied throughout its entire volume \mathcal{V} . The pressure force on a fluid is the integral of the pressure applied to its boundary

$$- \int_{\Omega} P d\Omega. \quad (2.13)$$

With the fundamental theorem of calculus, this surface integral can be expressed as a volume integral

$$- \int_{\mathcal{V}} \nabla P d\mathcal{V}. \quad (2.14)$$

The above force is in the direction opposite to the pressure gradient. It works to make the pressure distribution uniform by moving fluid from regions of high pressure to regions of low pressure.

Regarding body forces, they most often include the gravitational force $\rho\mathbf{g}$, where \mathbf{g} is a force vector whose magnitude is equal to the gravitational acceleration and whose orientation is downwards. In this thesis, we generalize body forces to \mathbf{f}_b , such that the combined effects of all body forces on the fluid can be expressed as

$$\int_{\mathcal{V}} \mathbf{f}_b d\mathcal{V}. \quad (2.15)$$

Note that there is no need to convert to a volume integral as body forces, by definition, already apply to the fluid as a whole. With the pressure force and body forces being defined over an arbitrary volume \mathcal{V} (i.e., Equation 2.14 and Equation 2.15), they can be easily substituted in Equation 2.11, transforming Equation 2.12 into Equation 2.2. If the right-hand side (RHS) of the Equation 2.2 were zero, then we would have a steady system.

Before moving on to describing how to solve Equation 2.1 and Equation 2.2 numerically, we briefly address boundary conditions, which are essential to fluid simulation. Not only

are they an inherent part of the problem’s model, but they also provide users with a certain amount of control over the simulation as thoroughly explained by Foster and Metaxas [16]. Indeed, inflow and outflow can help achieve specific effects, and solid boundaries considerably influence a fluid’s behaviour. In this document, we only concern ourselves with enforcing free-slip boundary conditions for stationary axis-aligned solid obstacles, or periodic boundary conditions otherwise.

2.2 How to Simulate Fluids?

As presented in Chapter 1, animating fluids has been a long-standing research topic in computer graphics. In light of the definitions provided in the previous section, we adopt a physically-based approach to the simulation of fluid dynamics by trying to numerically solve Equation 2.1 and Equation 2.2. In particular, the algorithm we concentrate on relies on operator splitting, a popular technique in computer graphics since its introduction by Stam [35], although much earlier instances existed in CFD.

As described by Bridson [4], splitting amounts to decomposing a complex problem into smaller simpler components, and solving these subparts. In the present context, splitting is applied to the equations of motion with respect to how they update velocity \mathbf{u} in time. The splitting of the operators specifically targets the momentum equation. Equation 2.2 can be expressed as

$$\frac{\partial \mathbf{u}}{\partial t} = -\mathbf{u} \cdot \nabla \mathbf{u} - \frac{1}{\rho} \nabla P + \mathbf{f}_b. \quad (2.16)$$

Recall that the advection term $\mathbf{u} \cdot \nabla$ is hidden in the material derivative $\frac{D}{Dt}$. The terms on the RHS of Equation 2.16 correspond to the three different subproblems issued from operator splitting:

$$\frac{\partial \mathbf{u}}{\partial t} = \mathbf{f}_b, \quad (2.17a)$$

$$\frac{\partial \mathbf{u}}{\partial t} + \frac{1}{\rho} \nabla P = 0 \quad \text{such that} \quad \nabla \cdot \mathbf{u} = 0, \quad (2.17b)$$

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = 0. \quad (2.17c)$$

First, the velocity field is updated according to the body forces acting on the fluid, as described by Equation 2.17a. Most often, this is achieved using forward Euler as a time

Algorithm 2.1 SIMULATION LOOP FOR ONE TIME STEP

Require: \mathbf{u}^n **Ensure:** \mathbf{u}^{n+1}

- | | |
|----------------------------------------------------------------|----------------|
| 1: Apply body forces | Equation 2.17a |
| 2: Enforce incompressibility by performing pressure projection | Equation 2.17b |
| 3: Advect velocity field and, possibly, other quantities | Equation 2.17c |
-

integration method

$$\mathbf{u}^{n+1} = \mathbf{u}^n + \Delta t \mathbf{f}_b|_{\mathbf{x}}. \quad (2.18)$$

Stam [35] mentions that a first-order accurate procedure, like Equation 2.18, is sufficient as \mathbf{f}_b is not expected to vary significantly during the time step Δt , at least for scenarios pertaining to graphics applications. Of course, higher-order time integration methods can be used, should the need for it arise.

In this type of solver, incompressibility is enforced by using the pressure force as a correction mechanism. This second subproblem is called pressure projection for reasons that are further and better explained by Bridson [4]. In the context of this work, Equation 2.17b is transformed into a linear system, which accounts for boundary conditions, and is solved directly via LDLT decomposition. Alternatively, it is also possible to solve the linear system iteratively with preconditioned conjugate gradient for instance.

At last, advection is performed according to Equation 2.17c, the steady state version of the momentum equation. We leave the discussion of the details to Chapter 3, where several transport algorithms are surveyed. However, two points should be highlighted. While Equation 2.17c is specific to the velocity field, the simulation may include other advectable quantities q , such as temperature, concentrations of various substances, etc. Transport of these quantities should also be performed during this step. More generally, Equation 2.17c becomes

$$\frac{\partial q}{\partial t} + \mathbf{u} \cdot \nabla q = 0, \quad (2.19)$$

which is the focus of Chapter 3. Furthermore, for conservation purposes, advection should always be performed using a divergence-free velocity field. This requirement is satisfied by carrying out the pressure projection step immediately before the advection step.

Equation 2.17 translates into the simulation loop's core, illustrated by Algorithm 2.1. Regardless each step's accuracy in Algorithm 2.1, solving the PDEs via operator splitting can only be so accurate, as any similar approximations. In particular, the above splitting is limited to first-order accuracy in time as pointed out in [4], wherein which Bridson mentions

the existence of other higher-order accurate splitting methods. However, a discussion on the effectiveness and accuracy of operator splitting in comparison to other approaches, such as the advection-reflection solver of Zehnder et al. [43], is beyond the scope of this document. We are content in knowing that the applicability of first-order accurate splitting-based solvers in computer graphics has been proven many times to be sufficient.

Practically, the fluid simulation outlined above can be achieved under two distinct settings: the Eulerian or the Lagrangian framework. These viewpoints give rise to two distinctive families, namely grid-based and particle-based methods. Monitoring weather provides a good analogy to understand the difference between both frameworks [4, 41].

Typically, a weather forecast is presented by a few easily countable points on a map; data from fixed local weather stations is used to forecast future atmospheric conditions. Where there is no local weather station, information from neighbouring stations is combined to produce the forecast. Similarly, in the Eulerian framework, the simulation domain is mapped to a predetermined structure made up of a few nodes whose spatial connectivity is known. Each node corresponds to a location where a quantity is observed. A regular grid with cells of dimensions $\Delta \mathbf{x}$ is a commonly used discretization. Grid-based approaches reinforce the idea of fluids as continua by representing them as combinations of scalar and vector fields; a scalar field for density and a vector field for velocity would be the minimal set.

Occasionally, atmospheric scientists release weather balloons into the atmosphere to better study certain phenomena. These balloons are carried by the winds, travel great distances, and transmit back information regarding the state of their surroundings. They are also equipped with some tracking device allowing the data they collect to be geolocated. Likewise, in the Lagrangian framework, particles are used to keep track of the relevant quantities as they are transported along the flow rather than being tethered to a fixed grid. The fluid is represented by all the particles, each of which is individually aware of the quantities relevant to its simulation, velocity and mass for instance. Particle-based approaches stem from the fact that a fluid is a collection of distinctly identifiable fluid parcels or elements.

While some methods are purely Eulerian or purely Lagrangian, both representations are useful in understanding the behaviour of fluids as evidenced by the presence of the material derivative $\frac{D}{Dt}$ in the governing PDEs. As introduced earlier, the material derivative relates the rate of change of fluid-relevant quantities under both viewpoints. Imagine a function $q(\mathbf{x}, t)$ which gives the value of quantity q at time t for a particle located at position \mathbf{x} . Investigating the rate of change of q and using the chain rule, we can easily

retrieve the definition of the material derivative:

$$\begin{aligned}
 \frac{dq}{dt} &= \frac{\partial q}{\partial t} + \nabla q \cdot \frac{d\mathbf{x}}{dt} \\
 &= \frac{\partial q}{\partial t} + \nabla q \cdot \mathbf{u} \\
 &= \frac{\partial q}{\partial t} + \mathbf{u} \cdot \nabla q \\
 &= \frac{D}{Dt}.
 \end{aligned}
 \tag{2.20}$$

These two families of approaches (i.e., grid-based/Eulerian and particle-based/Lagrangian) fully cover the dichotomy between the advection step and the pressure solve.

On one hand, the former can enforce incompressibility easily at the expense of often introducing tremendous numerical dissipation in the solution, as we further examine in Chapter 3. As explained by Bridson [4], the solution to the advection equation without viscosity (i.e., Equation 2.19) computed using semi-Lagrangian advection for instance appears to include viscosity. This phenomenon is named “numerical dissipation”.

On the other hand, particle-based approaches can perform transport intuitively, but struggle to achieve pressure projection effectively, since it involves vector calculus operators, like the gradient ∇ , which are less trivial to approximate on unstructured points than on grids. Indeed, purely Lagrangian schemes are best suited for advection since they transport data along the flow without resampling, which is a major source of artificial dissipation [4].

Hybrid methods attempt to leverage the advantages of both families by combining grids and particles. Advection is performed on particles whereas pressure projection and body forces are solved for on a grid. Hybrid techniques introduce additional grid-to-particle and particle-to-grid transfer operations, on which they heavily rely. These transfer operations allow the mapping of quantities between the two representations as shown in Figure 2.1. Hence, the simulation loop of Algorithm 2.1 is slightly modified to account for these additional steps as seen in Algorithm 2.2.

As will be made clear from briefly surveying popular hybrid methods in computer graphics, their attempts at decreasing the numerical dissipation introduced by the advection step are not always successful in fully eradicating said byproduct.

The particle-in-cell (PIC) method is such a hybrid approach from early CFD [23]. The transfer operations can be as simple as trilinear interpolation, as in Zhu and Bridson [44]. In spite of being stable, PIC also produces excessive numerical dissipation due to the accumulated interpolations, which generate unwanted smoothing. In particular, the mapping

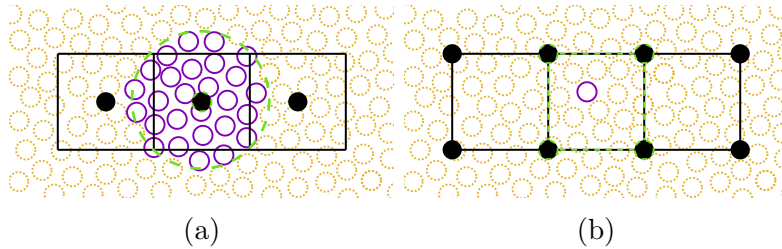


Figure 2.1: *Mapping between particles and grid.* Particles are represented by yellow dotted circles or purple circles according to their utility. The grid is represented by a black mesh with black disks standing for the nodes; nodes relevant to the transfer operation have green dashed contours. (a) Particle-to-grid transfer is achieved by weighting the contribution of all the particles surrounding the grid node of interest. The neighbourhood of particles relevant to a given grid node can be determined in many ways. In the above example, we only consider particles within a given radius, depicted by a green dashed circle, of the relevant grid node. (b) In grid-to-particle transfer, the cell containing the particle of interest is used to identify the relevant nodes; herein, only the immediate grid nodes, attached to the cell containing the particle, are used, but a greater ring could be considered.

of velocity between both representations causes the velocity field to lose angular momentum. Indeed, a grid node is incapable of storing angular momentum; therefore, vortices are potentially obliterated by the mapping from particles to grid. On account of the transfers between particles and grid occurring twice per time step, PIC's undesirable smoothing is worse than in semi-Lagrangian advection where interpolation is only performed once, as further explained in Chapter 3.

In an attempt to decrease PIC's artificial dissipation, the fluid-implicit-particle (FLIP) method was introduced by Brackbill and Ruppel [3]. Instead of overwriting the full velocities during grid-to-particle transfers, FLIP only transmits a correction to the particles' velocities. This correction successfully reduces the amount of numerical dissipation, but it has the adverse side effect of not preserving the stability guaranteed by PIC as the correction may be non-conservative, and particle quantities may carry sub-grid scale noise. Zhu and Bridson [44] propose to combine both techniques (i.e., PIC and FLIP) with a weighted sum to exploit their respective advantages. Even if the combined PIC-FLIP procedure provides some flexibility, it can still be noisy and unstable.

The search for a hybrid approach with both good stability and little additional dissipation led to the development of the affine particle-in-cell (APIC) method by Jiang et al. [27]. The idea behind APIC is that particles should carry a locally affine descriptor of velocity rather than representing velocity as a unique constant vector value. Thus, each particle p

Algorithm 2.2 HYBRID SIMULATION LOOP FOR ONE TIME STEP

Require: A set of particles each with m_p^n and \mathbf{u}_p^n

Ensure: A set of particles each with m_p^{n+1} and \mathbf{u}_p^{n+1}

- 1: Transfer particle quantities to grid
 - 2: Apply body forces
 - 3: Enforce incompressibility by performing pressure projection
 - 4: Transfer grid quantities to particles
 - 5: Advect velocity field and, possibly, other quantities
-

has position \mathbf{x}_p , mass m_p , velocity \mathbf{u}_p , and locally affine velocity descriptor \mathbf{C}_p . The original APIC [27] targeting fluids was formulated for staggered grids unlike the hereinafter formulation which adopts a generalized view. The simulation loop begins with particle-to-grid transfers (i.e., step 1 of Algorithm 2.2) performed according to

$$m_{\mathbf{i}}^n = \sum_p m_p w_{p \rightarrow \mathbf{i}}^n, \quad (2.21)$$

$$m_{\mathbf{i}}^n \mathbf{u}_{\mathbf{i}}^n = \sum_p m_p w_{p \rightarrow \mathbf{i}}^n (\mathbf{u}_p^n + \mathbf{C}_p^n (\mathbf{x}_{\mathbf{i}} - \mathbf{x}_p^n)), \quad (2.22)$$

where $w_{p \rightarrow \mathbf{i}}$ are particle-to-grid weights, and quantities subscripted with \mathbf{i} are located on grid nodes $\mathbf{x}_{\mathbf{i}}$.

Recalling the definition of momentum from Equation 2.9, Equation 2.22 expresses the mapping of momentum from particles to grid. Velocity can be retrieved by dividing by the mass $m_{\mathbf{i}}^n$. Note that the descriptor \mathbf{C}_p is not transferred to the grid, but rather used in the mapping of momentum to enforce information conservation.

Equation 2.22 can further be interpreted as a first-order Taylor series approximation of the local velocity field. In [20], Fu et al. generalize the locally affine velocity descriptor \mathbf{C}_p to a locally polynomial representation. The resulting polynomial particle-in-cell (PolyPIC) method amounts to including additional terms in the series.

After steps 2 and 3 of Algorithm 2.2 are performed on the grid, the divergence-free velocity field $\tilde{\mathbf{u}}^{n+1}$ only needs to be mapped back to the particles using

$$\mathbf{u}_p^{n+1} = \sum_{\mathbf{x}_{\mathbf{h}} \in \mathcal{H}} \tilde{\mathbf{u}}^{n+1} \Big|_{\mathbf{x}_{\mathbf{h}}} w_{\mathbf{i} \rightarrow p}^n \Big|_{\mathbf{x}_{\mathbf{h}}}, \quad (2.23)$$

where $w_{\mathbf{i} \rightarrow p}$ are grid-to-particle weights. Remember that the notation $q|_{\mathbf{x}}$ indicates that

the quantity q is evaluated at the position \mathbf{x} . Therefore, the above equation amounts to a weighted sum.

On the other hand, new descriptors are also assigned to each particle according to

$$\mathbf{C}_p^{n+1} = \sum_{\mathbf{x}_h \in \mathcal{H}} \tilde{\mathbf{u}}^{n+1} \Big|_{\mathbf{x}_h} \left(\nabla w_{i \rightarrow p}^n \Big|_{\mathbf{x}_h} \right)^T. \quad (2.24)$$

\mathbf{C}_p is a tensor essentially based on the gradient of the grid-to-particle weights $w_{i \rightarrow p}^n$. It defines the gradient used in first-order term of the Taylor series in Equation 2.22.

The good properties of APIC impose several restrictions on the choice of weights $w_{i \rightarrow p}$

$$\begin{aligned} \sum_{\mathbf{x}_h \in \mathcal{H}} w_{i \rightarrow p}^n \Big|_{\mathbf{x}_h} &= 1, \\ \sum_{\mathbf{x}_h \in \mathcal{H}} \mathbf{x}_h w_{i \rightarrow p}^n \Big|_{\mathbf{x}_h} &= \mathbf{x}_p^n, \\ \sum_{\mathbf{x}_h \in \mathcal{H}} (\mathbf{x}_h - \mathbf{x}_p^n) w_{i \rightarrow p}^n \Big|_{\mathbf{x}_h} &= 0. \end{aligned} \quad (2.25)$$

If all parts of Equation 2.25 are satisfied, then APIC simulations exhibit less numerical dissipation than PIC simulations. Weights issued from multi-linear interpolation, employed in the original formulation [27], meet Equation 2.25 for example.

Unlike FLIP, APIC prevents instabilities from arising. Additionally, Jiang et al. [27] demonstrate, in the supplementary material, that APIC exactly conserves angular momentum when transferring from particles to grid and vice-versa, which is pertinent when dealing with nonlinear flows.

Chapter 3

Advection

In this chapter, we review several advection algorithms to solve

$$\frac{\partial q}{\partial t} + \mathbf{u} \cdot \nabla q = 0, \quad (3.1)$$

where q represents an advectable scalar quantity. For vector quantities, q can be interpreted as any component of the vector. We begin with a presentation of the standard semi-Lagrangian advection method as introduced to computer graphics by Stam [35]. Then, we explore alternative techniques popular in computer animation, namely predictor-corrector algorithms. In addition to inspiring our proposed technique, presented in Chapter 4, all these methods provide a baseline against which to compare our technique’s numerical performance, notably in terms of accuracy, as later discussed in Chapter 5.

3.1 Semi-Lagrangian Advection

In grid-based approaches prior to the unconditionally stable semi-Lagrangian advection method of Stam [35], Equation 3.1 was solved with finite differences, drawing purely Eulerian methods from CFD [42]. Explicit schemes, such as upwind, Lax-Friedrichs, and Lax-Wendroff, would be chosen according to their accuracy, efficiency, or complexity. However, these purely Eulerian schemes all share a common caveat: guarantee of stability is achieved by restraining the size of the time step taken Δt to a function of the grid spacing $\Delta \mathbf{x}$. In other words, smaller grid spacings require smaller time steps to ensure stability; failure to meet time step constraints can result in important visual artefacts. For instance, when

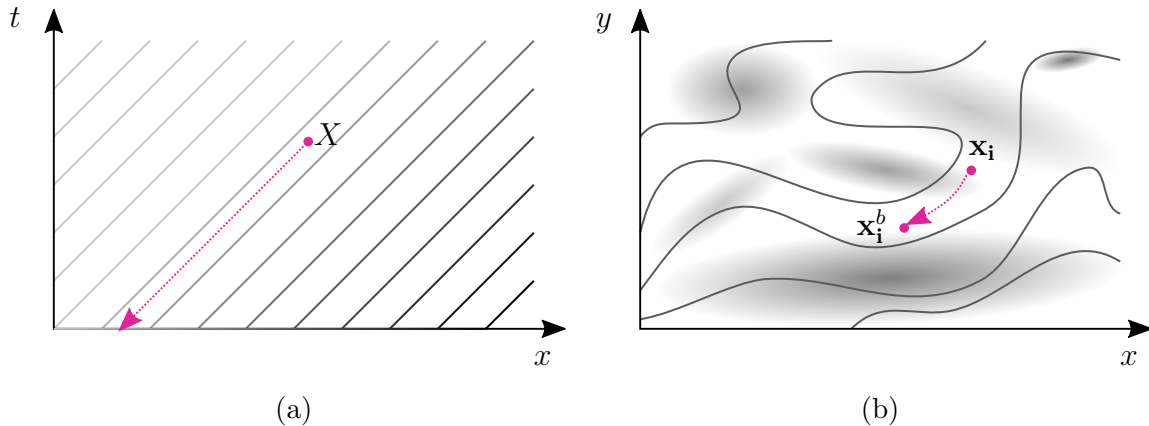


Figure 3.1: *Method of characteristics as a foundation for semi-Lagrangian advection.* (a) Characteristics for Equation 3.2 (i.e., unidimensional linear advection) are drawn as grayscale isocontours. Grayscale initial conditions lay on the horizontal axis, and are evolved forward in time, according to the PDE given by Equation 3.2, along the positive vertical t -axis. Characteristics originating from the t -axis correspond to boundary conditions. It is possible to find the value at any point X at any time by walking along the characteristics back to the initial condition at $t = 0$, as shown by the pink dotted path. (b) Streamlines of a random bidimensional velocity field \mathbf{u}^n are overlaid on a grayscale representation of a scalar field q . Here the axes represent the simulation domain, while the streamlines and background correspond to a snapshot of a scenario in two dimensions at some arbitrary time t . In semi-Lagrangian advection, \mathbf{x}_i is back-traced following the pink dotted path to \mathbf{x}_i^b , where q is interpolated.

high-resolution or adaptive grids are used, such a poor scalability can be a prominent hindrance. More importantly, in the balancing act between computational cost and quality, adjusting the time step is an intuitive and readily available trade-off from an artistic usability standpoint. Therefore, removing the time step dependence of the solution's stability can be helpful in preventing unfortunate explosive errors caused by too large time steps.

The approach proposed by Stam [35] is unconditionally stable, meaning that it remains stable for arbitrarily large time steps. Consequently, it excels at meeting the needs of computer graphics applications. It is based on the method of characteristics. A characteristic is a curve of constant quantity q , where q is the quantity being advected in the present case. The information from the problem's initial conditions propagate along the characteristics

Algorithm 3.1 STANDARD SEMI-LAGRANGIAN ADVECTION

Require: q^n, \mathbf{u}^n **Ensure:** q^{n+1}

- 1: $\mathbf{x}_i^b \leftarrow \text{BACK_TRACE}(\mathbf{x}_i, \mathbf{u}^n)$
 - 2: $q^{n+1} \leftarrow \text{INTERPOLATE}(\mathbf{x}_i^b, q^n)$
-

over time. Figure 3.1a shows the characteristics for the unidimensional linear advection

$$\frac{\partial q}{\partial t} + \frac{\partial q}{\partial x} = 0. \quad (3.2)$$

For any time t and any point x on the domain, the value q can be determined by travelling along the characteristics back to time $t = 0$.

The semi-Lagrangian advection method proceeds almost identically, as shown in Figure 3.1b. Here, the superscript b indicates not a temporal step, but that the position is the back-traced position corresponding to the non-superscripted variable. First, the grid positions \mathbf{x}_i^b are computed by back-tracing all positions \mathbf{x}_i through the velocity field \mathbf{u}^n , following the characteristics backwards in time. Then, the values $q|_{\mathbf{x}_i^b}$ are determined by interpolating from the grid. Semi-Lagrangian advection is summarized in Algorithm 3.1.

One important downside of semi-Lagrangian transport, as singled out by Stam [35], is its introduction of a significant amount of artificial dissipation in the numerical solution. In visual terms, quantities spread outwards. Even in an inviscid context, flows appear viscous. As a result, small-scale features dampen, and, eventually, disappear. Therefore, as noted by Stam [35], the proposed method is unsuitable for engineering applications, where greater accuracy may be required. However, he suggests that this effect could be mitigated, for computer graphics applications, by applying additional compensating forces. Vorticity confinement by Fedkiw et al. [14] is such a force; it only serves to enhance the simulation by adding small-scale features. While the physical nature of vorticity confinement is debatable, it definitely does not correct the artificial damping produced by semi-Lagrangian advection. In the best case, it compensates for it by introducing an additional quantity to produce more details.

Numerical dissipation can be reduced by choosing both an appropriate approximation of the characteristics and a good interpolation scheme. The simplest way to perform back-tracing is to take an Euler step backwards:

$$\text{BACK_TRACE}(\mathbf{x}_i, \mathbf{u}^n) = \text{EULER}(\mathbf{x}_i, \mathbf{u}^n) = \mathbf{x}_i - \Delta t \mathbf{u}_i^n. \quad (3.3)$$

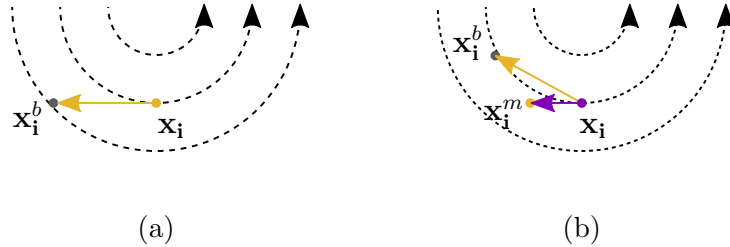


Figure 3.2: *Back-tracing functions*. Dashed arrows represent the streamlines of a nonlinear flow. Solid coloured arrows represent velocities as sampled from the point of the same colour. A simple Euler step (a) (i.e., Equation 3.3) does not approximate well nonlinear flows while the midpoint method (b) (i.e., Equation 3.4) can better capture a flow’s non-linearity.

Equation 3.3 is first-order accurate, which is an insufficient approximation for nonlinear flows as illustrated in Figure 3.2a. In practice, most flows are not purely linear. Thus, the second-order Runge-Kutta method, also known as the midpoint method, is typically used:

$$\text{BACK_TRACE}(\mathbf{x}_i, \mathbf{u}^n) = \text{MIDPOINT}(\mathbf{x}_i, \mathbf{u}^n) = \mathbf{x}_i - \Delta t \mathbf{u}^n|_{\mathbf{x}_i^m}, \quad (3.4)$$

where \mathbf{x}_i^m is the midpoint back-traced position of grid node \mathbf{x}_i , such that

$$\mathbf{x}_i^m = \mathbf{x}_i - \frac{1}{2} \Delta t \mathbf{u}_i^n. \quad (3.5)$$

As depicted in Figure 3.2b, the midpoint method yields a more accurate approximation of the flow than the single Euler step. Evidently, the closer the back-tracing function fits the PDE’s characteristics the better.

However, the accuracy of standard semi-Lagrangian advection is not only determined by the back-tracing function’s accuracy, but also by the interpolation scheme’s. Although it does not readily apply to all interpolation schemes, interpolation can be expressed as a weighted sum

$$q|_{\mathbf{x}} = \sum_{\mathbf{x}_h \in \mathcal{H}} w|_{\mathbf{x}_h} q|_{\mathbf{x}_h}, \quad (3.6)$$

where \mathcal{H} is the interpolation kernel, that is to say the set of grid nodes required to implement a given interpolation scheme. Each grid node \mathbf{x}_h in the interpolation kernel \mathcal{H} contributes to the interpolated value at \mathbf{x} according to weight $w|_{\mathbf{x}_h}$. In the nearest neighbour interpolation for instance, $|\mathcal{H}|$ is one, as the interpolated value is equal to the value

of the closest neighbouring grid node (i.e., $w = 1$).

For dimensions greater than one, if the interpolation scheme is applied sequentially along each dimension, the weights can be viewed as a product of the weights from applying the scheme along each dimension

$$w|_{\mathbf{x}_h} = w|_{x_h} w|_{y_h} w|_{z_h}. \quad (3.7)$$

The stability of semi-Lagrangian transport is in part determined by the monotonicity of the interpolation scheme used. Besides, discussing monotonicity is relevant as it directly relates to the conservation of advected quantities q . A necessary condition for a monotone interpolation scheme is

$$\min_{\mathbf{x}_e \in \mathcal{E}} q|_{\mathbf{x}_e} \leq q|_{\mathbf{x}} \leq \max_{\mathbf{x}_e \in \mathcal{E}} q|_{\mathbf{x}_e}. \quad (3.8)$$

In other words, a monotone scheme prohibits under/overshooting by limiting the interpolated value to the range defined by the set \mathcal{E} of its enclosing grid nodes \mathbf{x}_e .

Often, when defining interpolation schemes, a normalized position $\bar{\mathbf{x}}$ is used

$$\bar{\mathbf{x}} = (\bar{x}, \bar{y}, \bar{z}) = \frac{\mathbf{x} - \mathbf{x}_{i,j,k}}{\Delta \mathbf{x}} = \left(\frac{x - x_{i,j,k}}{\Delta x}, \frac{y - y_{i,j,k}}{\Delta y}, \frac{z - z_{i,j,k}}{\Delta z} \right), \quad (3.9)$$

where, for a given position \mathbf{x} and grid origin $\mathbf{x}_0 = (x_0, y_0, z_0)$,

$$\mathbf{x}_{i,j,k} = \Delta \mathbf{x} \left\lfloor \frac{\mathbf{x} - \mathbf{x}_0}{\Delta \mathbf{x}} \right\rfloor + \mathbf{x}_0 = \left(\Delta x \left\lfloor \frac{x - x_0}{\Delta x} \right\rfloor + x_0, \Delta y \left\lfloor \frac{y - y_0}{\Delta y} \right\rfloor + y_0, \Delta z \left\lfloor \frac{z - z_0}{\Delta z} \right\rfloor + z_0 \right). \quad (3.10)$$

The normalized position simplifies the notation since $\bar{\mathbf{x}} \in [0, 1]^3$.

In semi-Lagrangian advection methods, linear interpolation is a popular choice. It is monotone, simple to implement, and can be easily generalized to two and three dimensions (i.e., respectively bilinear and trilinear interpolation). For bilinear interpolation, $|\mathcal{H}| = 2^d$ as illustrated in Figure 3.3a. In higher dimensions, linear interpolation is performed along each dimension sequentially. Therefore, in two dimensions, each grid node contributes to the interpolated value according to the following weights:

$$\begin{aligned} w_{i,j} &= (1 - \bar{x})(1 - \bar{y}), \\ w_{i+1,j} &= \bar{x}(1 - \bar{y}), \\ w_{i,j+1} &= (1 - \bar{x})\bar{y}, \\ w_{i+1,j+1} &= \bar{x}\bar{y}. \end{aligned} \quad (3.11)$$

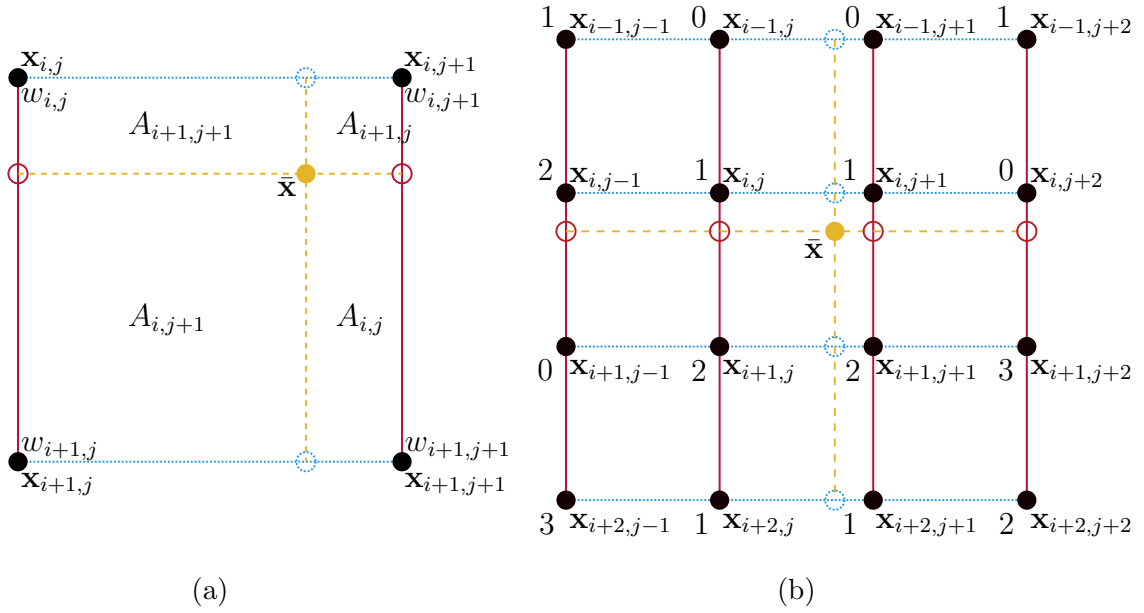


Figure 3.3: *Linear and cubic interpolation mechanisms in 2d.* Grid nodes represented by black disks form the set \mathcal{H} for (a) linear interpolation and for (b) cubic interpolation. Interpolation is performed at point $\bar{\mathbf{x}}$ identified by a yellow disk. The two possible orderings of dimensions are illustrated: along x first in solid red and along y first in dotted blue. Accordingly, interpolating along x/y first as shown by solid red/dotted blue lines gives the values located at the temporary intermediate points represented by empty solid red/dotted blue circles. These are then interpolated along the y/x dimension to give the value at $\bar{\mathbf{x}}$. In (a), the weights are proportional to the area opposite to their location. In (b), the weights (not shown) for each grid node depend on the interpolation scheme used.

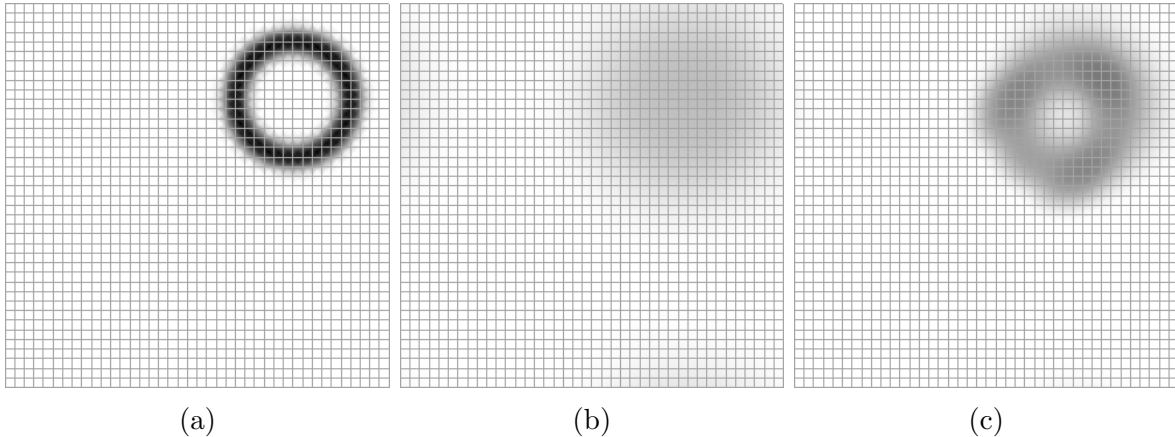


Figure 3.4: *Impact of higher-order accurate interpolation scheme on semi-Lagrangian advection in 2d.* An annulus positioned at the bottom-left corner of the grid is advected to $n = 1200$ with $\Delta t = \frac{1}{60}$ by a constant linear flow $\mathbf{u} = (1, 1, 0)$. (a) shows the analytical solution while both (b) and (c) show the numerical solution resulting from using semi-Lagrangian advection with respectively bilinear interpolation and the monotone cubic interpolation of Fritsch and Carlson [19]. The result from using higher-order interpolation (c) exhibits less numerical dissipation than its bilinear counterpart (b) as can be seen from the clearer empty region and better-preserved darker colour.

As obvious from Equation 3.11, the interpolation result is independent of the order in which the unidimensional interpolations are performed along each dimension. Since multiplication is commutative, $w|_{x_h}$, the x -part of the weights (i.e., the first operand in parentheses), can be switched with $w|_{y_h}$, the y -part of the weights (i.e., the second operand in parentheses).

Higher-order interpolation is a way to improve the accuracy of semi-Lagrangian transport with relatively little effort. Higher-order schemes increase the interpolated value's accuracy, as shown in Figure 3.4. For a cubic interpolation scheme, in two dimensions, $|\mathcal{H}| = 4^d$ as illustrated in Figure 3.3b. In one dimension, recall (or see Appendix A for a detailed reminder) that the general form for piecewise cubic polynomial interpolation is given by

$$q|_x = (\delta_i \Delta x + \delta_{i+1} \Delta x - 2D_i) \bar{x}^3 + (3D_i - 2\delta_i \Delta x - \delta_{i+1} \Delta x) \bar{x}^2 + (\delta_i \Delta x) \bar{x} + q|_{x_i}, \quad (3.12)$$

where

$$D_i = q|_{x_{i+1}} - q|_{x_i}, \quad (3.13)$$

and δ_i is the derivative at grid node x_i , likewise for δ_{i+1} . For cubic Hermite interpolation,

δ_i , likewise for δ_{i+1} , is computed using central finite differences

$$\delta_i = \frac{q|_{x_{i+1}} - q|_{x_{i-1}}}{2\Delta x}. \quad (3.14)$$

Together, Equation 3.12, Equation 3.13, and Equation 3.14 define the cubic Catmull-Rom interpolating spline [4].

Unless they are monotone, higher-order interpolation schemes should be applied with caution as they may generate new extrema. In such cases, clamping can be used to maintain the interpolated value within the range of acceptable values. Most often, said range is given by the values at $\mathbf{x}_e \in \mathcal{E}$, as expressed in Equation 3.8. Although clamping corrects under/overshooting, if need be, it breaks the continuity of the interpolating polynomial's derivatives, a sacrifice with consequences yet to be thoroughly investigated.

In computer graphics, one predominant cubic interpolation scheme, which is based on cubic Hermite spline interpolation, is that of Fedkiw et al. [14]. The authors highlight the necessary conditions for a monotone piecewise cubic polynomial interpolation scheme

$$\begin{cases} \text{sign}(\delta_i) = \text{sign}(\delta_{i+1}) = \text{sign}(D_i) & D_i \neq 0, \\ \delta_i = \delta_{i+1} = 0 & D_i = 0. \end{cases} \quad (3.15)$$

According to Fedkiw et al. [14], enforcing Equation 3.15 produces a monotone version of the Catmull-Rom spline interpolation, but this is insufficient. Despite being widely used within the computer graphics community, our experiments uncovered situations in which the modified Catmull-Rom scheme of Fedkiw et al. [14] yielded non-monotonic results. This notably occurs when the magnitudes of the derivatives δ_i and δ_{i+1} are disproportionate with respect to the magnitude of D_i . Indeed, we can design several examples, as shown in Figure 3.5, to illustrate the approach's non-monotonicity. We subsequently found confirmation of this observation in the graphics literature, in work by Ihm et al. [25].

Using the following non-negative parameters to simplify their monotonicity investigation,

$$\alpha_i = \frac{\delta_i}{D_i}, \quad (3.16)$$

and

$$\beta_i = \frac{\delta_{i+1}}{D_i}, \quad (3.17)$$

Fritsch and Carlson [19] prove that enforcing Equation 3.15 to ensure monotonicity is only

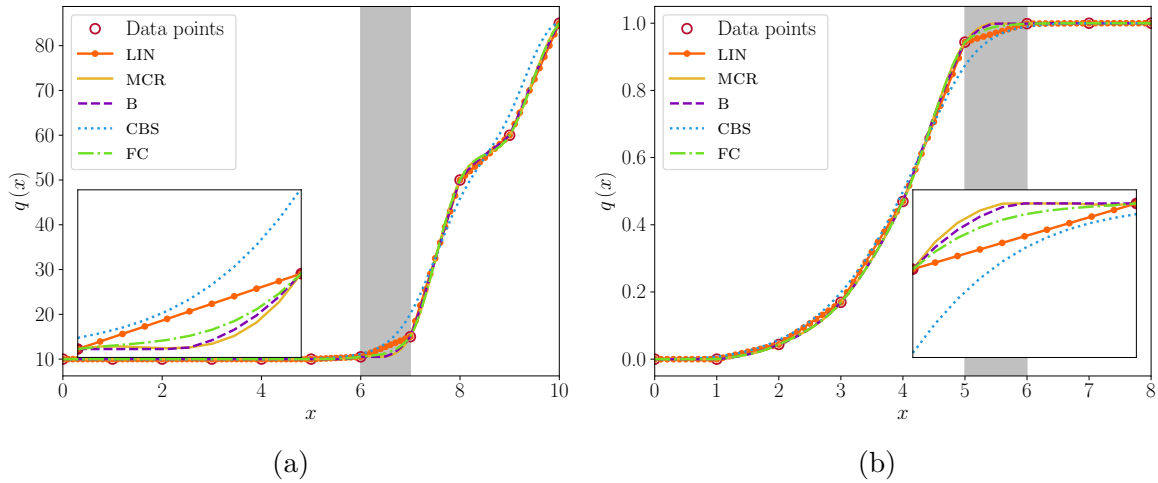


Figure 3.5: *Comparison between interpolation schemes in 1d.* Two different sets of data points, represented by empty red circles, are interpolated using linear interpolation (LIN in solid orange with circular markers), modified Catmull-Rom interpolation as presented by Fedkiw et al. [14] (MCR in solid yellow), Bridson’s cubic interpolation from the second edition of [4] (B in dashed purple), cubic B-spline interpolation as used in Stomakhin et al. [40] (CBS in dotted blue), and the monotone cubic interpolation of Fritsch and Carlson [19] (FC in dot-dashed green). Data sets are taken respectively from (a) the first example presented by Akima [2] and from (b) the second example of Fritsch and Carlson [19], but the points are evenly spaced as in uniform-grid simulations. For each subfigure, the inset plot shows a zoomed-in version of the region shaded in gray. In (a), monotonicity is violated by MCR on the interval $x \in [6, 7]$ as evidenced by the slight trough. B also does not respect the extrema given by the interval’s endpoints on $x \in [6, 7]$ as undershooting may be observed. Similarly, in (b), MCR and B overshoot on the interval $x \in [5, 6]$. In both cases, FC does not overshoot, and does preserve monotonicity while remaining interpolating whereas CBS is approximating rather than interpolating.

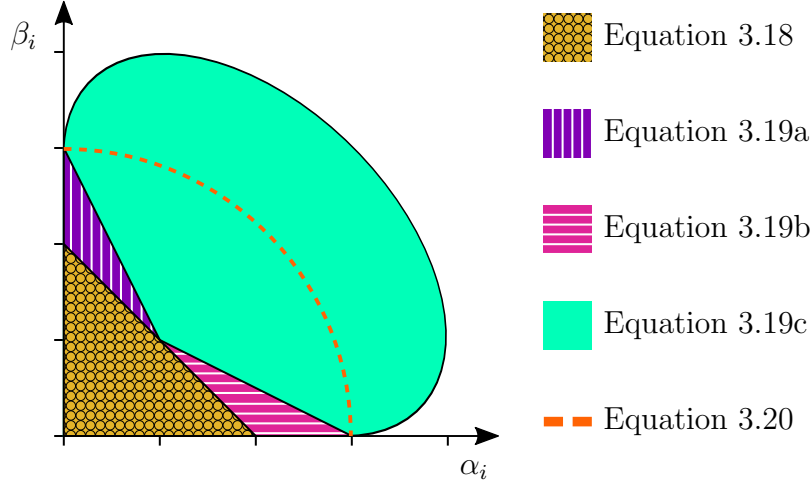


Figure 3.6: *Monotonicity region based on derivatives for cubic interpolation.* With the non-negative parameters α_i and β_i , a region of monotonicity can be defined in $\alpha\beta$ -space. The monotonicity region represents the (α_i, β_i) pairs which yield a monotone interpolant; it is the union of different sufficient conditions. Fritsch and Carlson [19] choose to restraint α_i and β_i to a subset \mathcal{M} of the monotonicity region corresponding to the inside of an origin-centred circle of radius 3, depicted by an orange dashed contour line.

valid if

$$\alpha_i + \beta_i - 2 \leq 0, \quad (3.18)$$

when the necessary conditions given by Equation 3.15 are also sufficient. Consequently, the modified Catmull-Rom interpolation of Fedkiw et al. [14] is not monotone when Equation 3.18 is not satisfied (i.e., when $\alpha_i + \beta_i - 2 > 0$) even though Equation 3.15 is satisfied. For cases when Equation 3.18 is not met, Fritsch and Carlson [19] derive other sufficient conditions:

$$2\alpha_i + \beta_i - 3 \leq 0, \quad (3.19a)$$

$$\alpha_i + 2\beta_i - 3 \leq 0, \text{ or} \quad (3.19b)$$

$$\alpha_i - \frac{(2\alpha_i + \beta_i - 3)^2}{3(\alpha_i + \beta_i - 2)} \geq 0. \quad (3.19c)$$

Having sufficient conditions for all possible cases, as illustrated in Figure 3.6, allowed Fritsch and Carlson [19] to design an algorithm to produce monotone piecewise cubic interpolants. In this thesis, we follow their approach, which we outline thereafter.

First, the derivatives δ_i and δ_{i+1} are initialized according to Equation 3.14. Then, sufficient conditions are enforced; Equation 3.15 is satisfied in accordance with Fedkiw et al. [14] while Equation 3.19 is met by restricting α_i and β_i to a specific monotone region of $\alpha\beta$ -space. In particular, as favoured by Fritsch and Carlson [19] and depicted in Figure 3.6, we choose the monotonicity subregion \mathcal{M} to be defined as the origin-centred circle of radius 3, yielding the constraint hereafter:

$$\mathcal{M} := \alpha_i^2 + \beta_i^2 \leq 9. \quad (3.20)$$

If α_i and β_i do not satisfy Equation 3.20, they are mapped to the monotonicity subregion \mathcal{M} by finding the closest point to the boundary of \mathcal{M} . Given our choice of \mathcal{M} (i.e., Equation 3.20), this corresponds to finding the intersection between the line joining the origin and the non-monotonic (α_i, β_i) pair. This can be achieved with a multiplicative corrective factor

$$\tau_i = \frac{3}{\sqrt{\alpha_i^2 + \beta_i^2}}, \quad (3.21)$$

which is applied such that

$$\alpha_i^{\text{corrected}} = \alpha_i \tau_i, \quad (3.22a)$$

$$\beta_i^{\text{corrected}} = \beta_i \tau_i. \quad (3.22b)$$

The correction τ_i can be propagated to δ_i and δ_{i+1} back into Equation 3.12 yielding

$$q|_x = (\tau_i \delta_i \Delta x + \tau_i \delta_{i+1} \Delta x - 2D_i) \bar{x}^3 + (3D_i - 2\tau_i \delta_i \Delta x - \tau_i \delta_{i+1} \Delta x) \bar{x}^2 + \tau_i (\delta_i \Delta x) \bar{x} + q|_{x_i}. \quad (3.23)$$

We note three drawbacks¹ in using the monotone cubic interpolation scheme of Fritsch and Carlson [19]. First, due to the nonlinear corrective factor defined in Equation 3.21, the scheme cannot be expressed in the form of Equation 3.6 (i.e., as a simple weighted sum). Secondly, it exhibits dimension ordering dependence, meaning that the result depends on the order in which interpolation is performed along each dimension in multidimensional problems. This can be at least partially caused by the algorithm's dependence on the grid's values, as further explained in Appendix A. We arbitrarily choose to perform interpolation along x first and then along y . Finally, given the apparent notoriety of clamping in computer graphics (e.g., Bridson's cubic scheme and the modified Catmull-Rom scheme from Fedkiw et al. [14], to name a few), one may question whether using Fritsch and Carlson's algorithm (or any similar attempt at built-in monotonicity) is worth the (slight)

¹These are distinct from the three flaws identified by Fritsch and Butland [18].

computational overhead.

In summary, semi-Lagrangian advection is an unconditionally stable highly dissipative transport method, whose accuracy depends on the back-tracing function and the interpolation chosen. To facilitate future notation, we define an operator \mathcal{A} combining the steps of Algorithm 3.1, such that we can succinctly write

$$q^{n+1} = \mathcal{A}(q^n). \tag{3.24}$$

The velocity field is implicitly taken to be \mathbf{u}^n . Operator \mathcal{A} both approximates the characteristics, and applies the interpolation scheme to retrieve the values q^n at corresponding back-traced positions \mathbf{x}_i^b .

3.2 Predictor-Corrector Algorithms

Prior to the introduction of unconditionally stable semi-Lagrangian advection in computer graphics, purely Eulerian methods from CFD were used to solve PDEs. Among grid-based methods tackling Equation 3.1, are multi-step advection techniques accounting for their own error; we concentrate specifically on predictor-corrector algorithms, like that of Dupont and Liu [10]. The procedure proposed by Dupont and Liu [10] accounts for the transport scheme’s built-in error while solving for the next time step. It leverages the fact that reversible differential equations can be evolved in both time directions using the same numerical scheme.

Imagine solving for \hat{q}^{n+1} by moving q^n forwards in time with a scheme \mathcal{F} . Similarly, solving for \hat{q}^n by moving \hat{q}^{n+1} backwards in time with the same scheme \mathcal{F} would give back exactly q^n if scheme \mathcal{F} had no numerical error. In practice however, \hat{q}^n is not equal to q^n . Since scheme \mathcal{F} is employed both for the forward and backward steps, the difference between \hat{q}^n and q^n is about twice the error of \mathcal{F} , assuming that both steps, forward and backward, incur identical numerical errors. In the algorithm of Dupont and Liu [10], this error is utilized to correct the original q^n . The corrected quantity \tilde{q}^n is then moved forwards in time using \mathcal{F} to produce the updated q^{n+1} .

This method requires three steps in time: one step forward, like the standard semi-Lagrangian advection, then one step backward to compute the error, and finally another step forward to advect the corrected quantity. Hence, it is named the back and forth error correction and compensation (BF ECC) method. Figure 3.7a illustrates the scheme’s steps. BF ECC has been shown to be second-order accurate when using a first-order accurate scheme \mathcal{F} . The increased accuracy comes only at thrice the computational cost of using

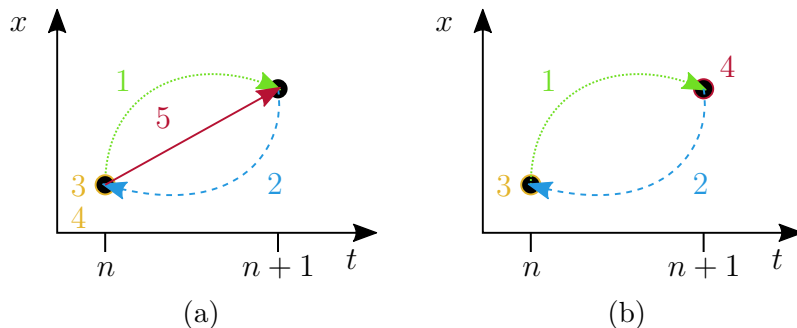


Figure 3.7: *Semi-Lagrangian multi-step advection methods*. (a) and (b) illustrate the BFEC and MacCormack advection schemes respectively. The steps from Algorithm 3.2 for (a) and from Algorithm 3.3 for (b) are numbered in colour. Arrows of different colours represent different time steps. The advection schemes are identical until step 3, inclusively, where the advection error is computed. They diverge afterwards in the way they incorporate said error. The MacCormack advection requires one step in time less than the BFEC advection.

Algorithm 3.2 BACK AND FORTH ERROR COMPENSATION AND CORRECTION METHOD

Require: q^n, \mathbf{u}^n

Ensure: q^{n+1}

- 1: $\hat{q}^{n+1} \leftarrow \mathcal{A}(q^n)$
 - 2: $\hat{q}^n \leftarrow \mathcal{A}^r(\hat{q}^{n+1})$
 - 3: $2e \leftarrow \hat{q}^n - q^n$
 - 4: $\tilde{q}^n \leftarrow q^n - e$
 - 5: $q^{n+1} \leftarrow \mathcal{A}(\tilde{q}^n)$
-

solely the original scheme \mathcal{F} (i.e., three steps in time against one). More importantly, in a subsequent paper, Dupont and Liu [11] show the validity of the results when using semi-Lagrangian advection as scheme \mathcal{F} (i.e., $\mathcal{F} = \mathcal{A}$), yielding the procedure described in Algorithm 3.2. Note that the resulting transport scheme is not monotone; consequently, it may produce new extrema, which can be eliminated using clamping. Another way to address the issues arising from non-monotonicity is to revert to a first-order technique when under/overshooting is detected as suggested by Selle et al. [34], who stressed the advantage of such an approach, notably in region near solid boundaries.

Selle et al. [34] also suggest a performance improvement to the BFEC algorithm. First, they assume that the operator \mathcal{A} is (nearly) linear, allowing step 5 of Algorithm 3.2

Algorithm 3.3 MACCORMACK ADVECTION

Require: q^n, \mathbf{u}^n **Ensure:** q^{n+1}

- 1: $\hat{q}^{n+1} \leftarrow \mathcal{A}(q^n)$
 - 2: $\hat{q}^n \leftarrow \mathcal{A}^r(\hat{q}^{n+1})$
 - 3: $2e \leftarrow \hat{q}^n - q^n$
 - 4: $q^{n+1} \leftarrow \hat{q}^{n+1} - e$
-

to be written as

$$q^{n+1} \leftarrow \mathcal{A}(q^n - e) = \mathcal{A}(q^n) - \mathcal{A}(e) = \hat{q}^{n+1} - \mathcal{A}(e). \quad (3.25)$$

Then, they stress that there is no compelling evidence that the advection error is time-dependent, yielding

$$q^{n+1} \leftarrow \hat{q}^{n+1} - e. \quad (3.26)$$

Accounting for those assumptions, the BFECC’s handling of the built-in advection error is modified into a “weaker” form, named by Selle et al. [34] the semi-Lagrangian MacCormack scheme, drawing parallels with the two steps in time of the Eulerian MacCormack method. For succinctness, we abandon the “semi-Lagrangian” qualifier, and refer to the method as MacCormack advection, implying that it is performed in a semi-Lagrangian fashion, except if explicitly mentioned otherwise.

The MacCormack advection begins identically to BFECC, initially solving for \hat{q}^{n+1} by moving q^n forwards in time with \mathcal{A} . Then, solving for \hat{q}^n by moving \hat{q}^{n+1} backwards in time with \mathcal{A}^r allows for the computation of the advection error. Diverging from BFECC, instead of using the error to correct the original q^n , it is utilized to correct the advected quantity \hat{q}^{n+1} , resulting in the updated q^{n+1} . The MacCormack advection procedure is described in Algorithm 3.3.

Its computational cost is decreased by a third compared to BFECC, and is only twice that of solely using standard semi-Lagrangian advection. Figure 3.7b illustrates the steps of the MacCormack advection. Akin to BFECC, it is also shown to be second-order accurate when using a first-order accurate operator \mathcal{A} . Hence, it exhibits similar reduced dissipation and comparable improved accuracy to BFECC, at only twice the cost of standard semi-Lagrangian advection. Unfortunately, it also shares BFECC’s non-monotonicity, which can, nevertheless, be handled similarly.

Furthermore, Selle et al. [34] generalize the MacCormack results to multidimensional advection. The corresponding theorem [34] states that if the technique employed to perform

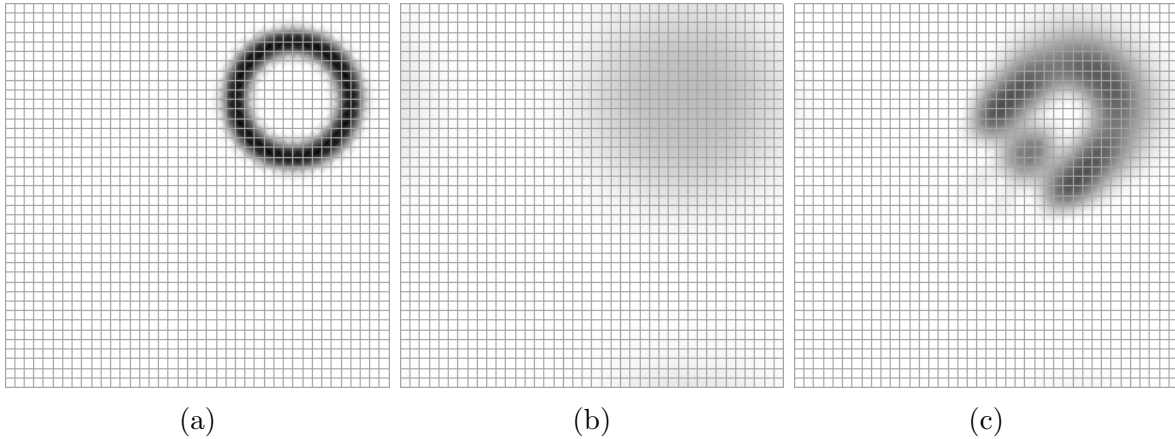


Figure 3.8: *Comparison between semi-Lagrangian advection methods in 2d.* An annulus positioned at the bottom-left corner of the grid is advected to $n = 1200$ with $\Delta t = \frac{1}{60}$ by a constant linear flow $\mathbf{u} = (1, 1, 0)$. (a) shows the analytical solution while both (b) and (c) show the numerical solutions respectively resulting from using (b) standard semi-Lagrangian advection and (c) MacCormack advection with semi-Lagrangian steps. In both cases, the midpoint method and linear interpolation are used. The result from MacCormack advection (c) exhibits less numerical dissipation than the result from standard semi-Lagrangian advection (b) as can be seen from the clearer empty region and better-preserved darker colour.

the time stepping is accurate to an order n , where n is a positive odd integer, then the MacCormack scheme's order of accuracy is $n + 1$. While first-order Eulerian schemes, like upwind, can be used as stepping methods, the authors emphasize that using semi-Lagrangian stepping does not impact the method's desirable properties. Hence, the overall procedure can be easily made unconditionally stable by using standard semi-Lagrangian as a stepping scheme. Figure 3.8 illustrates the improvement gained by using MacCormack advection with semi-Lagrangian steps on simply using standard semi-Lagrangian transport with the midpoint method and linear interpolation.

Chapter 4

An Affine Semi-Lagrangian Advection Method

We propose a method to reduce the numerical dissipation introduced by performing semi-Lagrangian advection. In Chapter 2, we discussed the advantages of hybrid approaches pertaining to advection. Broadly speaking, particles were the source of the improved transport exhibited by hybrid techniques compared to purely Eulerian algorithms. However, they are also the origin of some of their shortcomings¹.

Indeed, in gaseous simulation, where semi-Lagrangian advection is still commonly used (see examples in Zehnder et al. [43] for instance), there is no surface and, therefore, no need for particles tracking the interface between the fluid and the non-fluid regions of the simulation domain. In this context, the particles' only usefulness is merely to facilitate transport, and one can argue that the resources they require outweigh their utility. Another challenge stemming from the use of particles is the resampling needed to ensure they are well distributed, or else suffer from ensuing artefacts. Although the particle count can be restricted as in narrow-band FLIP [15] where particles are used solely near the interface, the treatment of the interior, which can correspond to a significant portion of the simulation domain, could still benefit from a less dissipative transport scheme.

Therefore, our main contribution is a method aiming both to reduce dissipation arising from semi-Lagrangian advection and to remove the need for particles, indirectly disposing of their associated flaws. Our method adapts the ideas of APIC, more specifically the locally affine velocity descriptor \mathbf{C}_p defined by Equation 2.24, to semi-Lagrangian advection under

¹The drawbacks of hybrid methods arising from the use of particles are evidently shared by Lagrangian approaches as well.

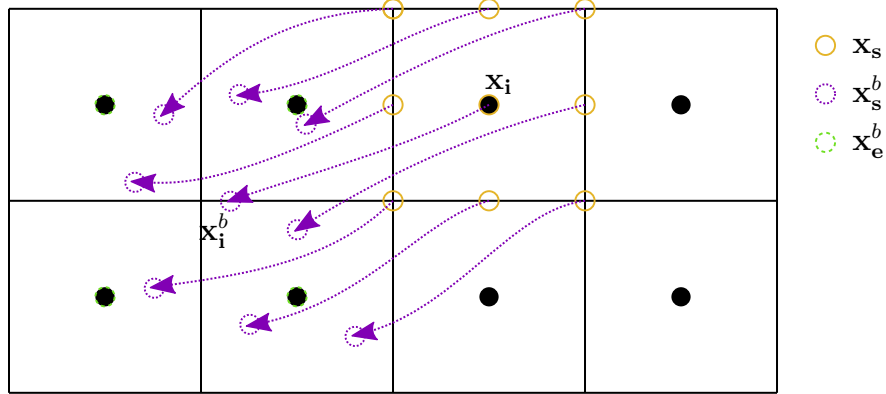


Figure 4.1: *Schematic diagram of the ASLAM.* A cell-centred grid has nodes depicted by black disks. Empty yellow circles represent the positions of fictitious particles \mathbf{x}_s associated with grid node \mathbf{x}_i . These fictitious particles are back-traced through the velocity field (not shown) following the purple dotted arrows to the positions \mathbf{x}_s^b corresponding to the empty dotted purple circles. The update’s result is clamped against the values at \mathbf{x}_e^b depicted by empty dashed green circles.

an Eulerian setting. Recall the two steps in the semi-Lagrangian advection method: back-tracing and interpolation, both outlined in Algorithm 3.1. At the back-traced position \mathbf{x}_i^b , in addition to interpolating the value of the advected quantity q , we interpolate a locally affine velocity descriptor \mathbf{c} . Furthermore, instead of back-tracing a single sample \mathbf{x}_i per grid node, we back-trace several temporary samples \mathbf{x}_s per grid node as shown in Figure 4.1. The combined use of the descriptor \mathbf{c} with the additional samples \mathbf{x}_s yields a more accurate reconstruction of the updated field q^{n+1} . Our method, summarized in Algorithm 4.1, is hence named the **affine semi-Lagrangian advection method** (ASLAM) as it basically transforms APIC into a particle-free method.

In a purely Eulerian framework, like the ASLAM, since there are no persistent particles, transfer operations, fundamental in hybrid approaches, are undefined. To address this issue, we define \mathcal{S} to be a set of fictitious particles \mathbf{x}_s in the vicinity of \mathbf{x}_i , as illustrated in Figure 4.1. To adapt APIC to the advection step, we modify Equation 2.22, converting the transfer from particles to grid into an update (i.e., step 6 of Algorithm 4.1) for a scalar

Algorithm 4.1 AFFINE SEMI-LAGRANGIAN ADVECTION

Require: $q^n, \mathbf{u}^n, \mathcal{S}$
Ensure: q^{n+1}

```

1: for all  $\mathbf{x}_i$  do
2:   for all  $\mathbf{x}_s \in \mathcal{S}$  do
3:      $\mathbf{x}_s^b \leftarrow \text{BACK\_TRACE}(\mathbf{x}_s, \mathbf{u}^n)$ 
4:      $q^n|_{\mathbf{x}_s^b}, \mathbf{c}^n|_{\mathbf{x}_s^b} \leftarrow \text{INTERPOLATE}(\mathbf{x}_s^b, q^n)$ 
5:   end for
6:    $q^{n+1}|_{\mathbf{x}_i} \leftarrow \text{UPDATE}\left(q^n|_{\mathbf{x}_s^b \forall \mathbf{x}_s \in \mathcal{S}}, \mathbf{c}^n|_{\mathbf{x}_s^b \forall \mathbf{x}_s \in \mathcal{S}}\right)$ 
7: end for

```

quantity q located at grid node \mathbf{x}_i

$$q^{n+1}|_{\mathbf{x}_i} = \frac{1}{\sum_{\mathbf{x}_s \in \mathcal{S}} w_{p \rightarrow i}|_{\mathbf{x}_s}} \sum_{\mathbf{x}_s \in \mathcal{S}} \left(q^n|_{\mathbf{x}_s^b} + \mathbf{c}^n|_{\mathbf{x}_s^b} (\mathbf{x}_i - \mathbf{x}_s) \right) w_{p \rightarrow i}|_{\mathbf{x}_s}. \quad (4.1)$$

Equation 4.1 is a weighted average of Taylor series expansions from data of the fictitious particles positioned at \mathbf{x}_s to the grid node of interest \mathbf{x}_i . Here we use a first-order approximation as in APIC [27] although it could be possible to use higher-order approximation as in PolyPIC [20]. The locally affine velocity descriptor \mathbf{c} as well as the quantity q are both defined in the same manner as in APIC. The former is computed using the gradient of the weights used to interpolate from grid to particles

$$\mathbf{c}^n|_{\mathbf{x}_s^b} = \sum_{\mathbf{x}_h \in \mathcal{H}} q^n|_{\mathbf{x}_h} \nabla \left(w_{i \rightarrow p}^n|_{\mathbf{x}_h} \right)^T. \quad (4.2)$$

The latter is simply interpolated from the grid values

$$q^n|_{\mathbf{x}_s^b} = \sum_{\mathbf{x}_h \in \mathcal{H}} q^n|_{\mathbf{x}_h} w_{i \rightarrow p}^n|_{\mathbf{x}_h}. \quad (4.3)$$

In both Equation 4.2 and Equation 4.3, the weights from linear interpolation (e.g., Equation 3.11 in two dimensions) are used as grid-to-particle weights $w_{i \rightarrow p}$.

Similar to the predictor-corrector algorithms presented in Chapter 3, the update of Equation 4.1 can also produce new extrema. To eliminate those, we clamp with respect to the set of enclosing nodes \mathbf{x}_e^b at the back-traced position \mathbf{x}_i^b , as shown in Figure 4.1. In the ASLAM, the number of additional advection steps, compared to standard semi-Lagrangian

advection, depends on the total number of fictitious particles for all grid nodes \mathbf{x}_i , which, as demonstrated later, is not necessarily equivalent to $|\mathcal{S}| \cdot N$, where N is the number of grid nodes \mathbf{x}_i .

Although Equation 4.1, Equation 4.2, and Equation 4.3 target scalar quantities q , they are valid for vector quantities as well since q can be taken as a vector quantity’s component. In such cases, note that the descriptor \mathbf{c} is a tensor rather than a vector. While the closeness of our formulation to the original APIC formulation [27] should preserve the scheme’s good stability and its reduced artificial dissipation, our formulation differs from the original APIC [27] in one key aspect.

In hybrid methods, including APIC, advection is performed on the particles using the velocity field provided by the grid. The transport mechanism, whether Runge-Kutta or another forward integration method, is unaffected by the transfer operations defined between grid and particle representations. Particles only passively carry their descriptors \mathbf{C}_p , which are not used to execute the transport. APIC’s transfer operations improve the accuracy of the velocity field by minimizing the error inherent to mapping between representations; their positive effect on the advection results is indirect. Indeed, the decreased numerical dissipation in the advection is caused by the improved velocity field, not by an improvement of the particle transport mechanism itself as it is unaltered (i.e., particles’ trajectories are defined in the same manner). In the ASLAM, this indirection is removed. The transport mechanism is modified to integrate the locally affine descriptor \mathbf{c} and additional samples \mathbf{x}_s into a single update equation (i.e., Equation 4.1).

Besides, APIC’s conservative properties, conservation of linear and angular momenta for instance, are only guaranteed for the mapping between representations, which occurs within a single time step, rather than from one time step to another. In proposing to transform APIC into an advection formula, we hope these properties can be carried over several time steps.

In the following sections, we discuss in depth two important features of the ASLAM. First, we investigate some challenges related to \mathcal{S} , the set of fictitious particles. Then, we present different options for the weights $w_{p \rightarrow i}$ from fictitious particles to grid node.

4.1 Fictitious Particle Set

Fictitious particles are the core of Equation 4.1, and hence the core of the ASLAM. One crucial question hidden in the formulation of Equation 4.1 is whether all grid nodes should have identically distributed fictitious particles. Although it would be possible to handle

grid nodes differently based on their material, velocity gradient, or other local properties, we treat all grid nodes identically, for simplicity.

Given that all grid nodes are assigned the same local distribution of fictitious particles \mathbf{x}_s , we refer to \mathcal{S} as a stencil and its members \mathbf{x}_s as stencil particles. The same stencil is used for all grid nodes, which allows us to focus on a single cell with grid node \mathbf{x}_i . Two important issues arise in trying to define a good stencil:

1. How large should \mathcal{S} be?
2. Where should the stencil particles \mathbf{x}_s be located with respect to the grid node of interest \mathbf{x}_i ?

While we tackle these challenges conjointly, we examine the distribution of fictitious particles first, and explain why this choice is tightly integrated with the overall size of the stencil.

First of all, we determine that a stencil should not cover more space than the grid spacing $\Delta\mathbf{x}$. Indeed, for a regular grid, the control volume associated to each grid node extends exactly half $\Delta\mathbf{x}$ in each direction. So, all stencil particles should be located at most half $\Delta\mathbf{x}$ in each direction from the grid node \mathbf{x}_i they are associated with in order to fully sample its control volume.

In addition, it is reasonable to assume that the distribution of stencil particles changes according to the number of said particles, especially since we want to enforce a good coverage of the grid cell. The distribution thus depends on the number of dimensions d and a parametric size $\zeta \in \mathbb{N}^0$. Note that when ζ is zero, then the stencil's size $|\mathcal{S}|$ should be one, and the only fictitious particle should be collocated with the grid node, yielding back the standard semi-Lagrangian advection method. In such a case, when $\exists! \mathbf{x}_s \in \mathcal{S} : \mathbf{x}_s = \mathbf{x}_i$, we refer to such a stencil particle as \mathbf{x}_s^* .

As a starting point for the construction of stencil \mathcal{S} , we design three different possible distribution types: CHESSBOARD, MANHATTAN, and SUBSAMPLING. The CHESSBOARD and MANHATTAN distribution types originate from the chessboard and Manhattan distances respectively. The SUBSAMPLING distribution is inspired from the super-sampling in ray-tracing algorithms. All these stencil distribution types are illustrated in Figure 4.2, for $d = 2$ and parametric sizes $\zeta = 1, 2$. Table 4.1 shows how the number of dimensions d and the parametric size ζ are related to the stencil's size $|\mathcal{S}|$ for the three distribution types. Note that the parametric size ζ has a different meaning for each distribution type.

For the CHESSBOARD distribution type, ζ is the maximum chessboard distance between \mathbf{x}_i and any \mathbf{x}_s . Since the corresponding Euclidean distance is half the grid spacing, ζ can be

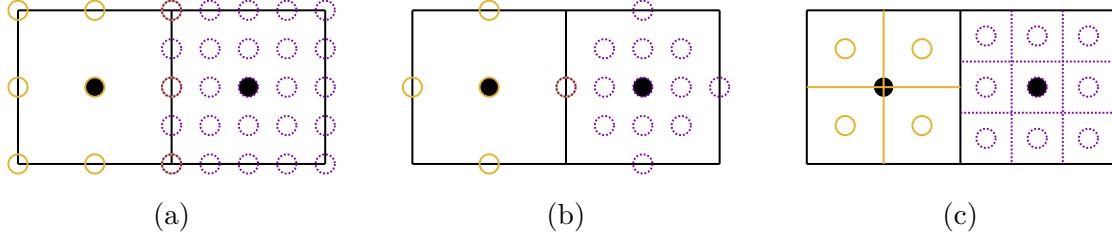


Figure 4.2: *Stencil distribution types*. Empty **yellow** circles represent the positions of stencil particles associated with a stencil of parametric size $\zeta = 1$ for the left grid node while empty dotted **purple** circles represent the positions of stencil particles associated with a stencil of parametric size $\zeta = 2$ for the right grid node for (a) CHESSBOARD, (b) MANHATTAN, and (c) SUBSAMPLING distribution types respectively. For (a) and (b), neighbouring grid nodes share stencil particles for all values of ζ , but this is never the case in (c).

Distribution Type	Number of Stencil Particles $ \mathcal{S} $
CHESSBOARD	$(2\zeta + 1)^d$
MANHATTAN	$\text{DELANNOY_NUMBER}(d, \zeta) = \sum_{k=0}^{\min(d, \zeta)} \binom{d+\zeta-k}{d} \binom{d}{k}$
SUBSAMPLING	$(\zeta + 1)^d$

Table 4.1: *Number of stencil particles according to stencil distribution type*. The number of stencil particles depends on the number of dimensions d , the parametric size ζ , and the stencil distribution type. Notice that the number of stencil particles is strictly increasing as a function of the parametric size ζ .

thought of as the halfsize of the stencil along each dimension. Likewise for the MANHATTAN distribution type, ζ is the maximum Manhattan distance between \mathbf{x}_i and any \mathbf{x}_s , whose Euclidean counterpart is also half the grid spacing. For both these distribution types, the grid node of interest \mathbf{x}_i is always included as a stencil particle. However, for the SUBSAMPLING distribution type, where ζ is the number of subdivisions per dimension, the grid node of interest is only included if ζ is even. According to the ASLAM’s update formula, Equation 4.1, for \mathbf{x}_s^* , the Taylor expansion reduces to the first term only. There is no contribution from the descriptor \mathbf{c} at the grid node’s position.

Typically, in hybrid methods, the set of particles relevant to a grid node \mathbf{x}_i changes at every time step as Lagrangian particles are relentlessly transported around. As a direct consequence, their contributions to different grid nodes, as depicted in Figure 2.1a, change too. By contrast, the computational cost of these operations is amortized in the ASLAM since the stencil particles are given fixed starting positions. As opposed to traditional particles, there is no need to track stencil particles on the long term, or to identify those relevant to a given grid node. The relationship between stencil particles and grid nodes is unchanging, and is set up at the beginning of the simulation. The weights are also unchanging, and can be pre-computed, which is computationally advantageous.

To conclude this section, many distributions other than the three aforementioned are possible. For instance, if we discard our initial assumption to treat all grid nodes identically and independently, it would be possible to define a single stencil particle distribution for an entire domain instead of using a cell-sized stencil. Alternatively, stencil particles could be distributed more densely in regions of interest while reverting back to semi-Lagrangian in regions of lesser importance. Such explorations are left for future endeavours.

4.2 Particle-to-Grid Weights

Each stencil particle contributes to the update according to Equation 4.1. The magnitude of the contribution is determined by the weights $w_{p \rightarrow i}$. We propose three different weighting techniques: UNIFORM, SPH, and GAUSSIAN. These are depicted in Figure 4.3, and further described below.

The first technique consists of having equal contributions from all stencil particles, such that

$$w_{p \rightarrow i} |_{\mathbf{x}_s} = \frac{1}{|\mathcal{S}|} \quad \forall \mathbf{x}_s. \quad (4.4)$$

This amounts to averaging all the data from the stencil particles, which, in practice, presumably smooths any existing extrema. However, it is not computationally demanding,

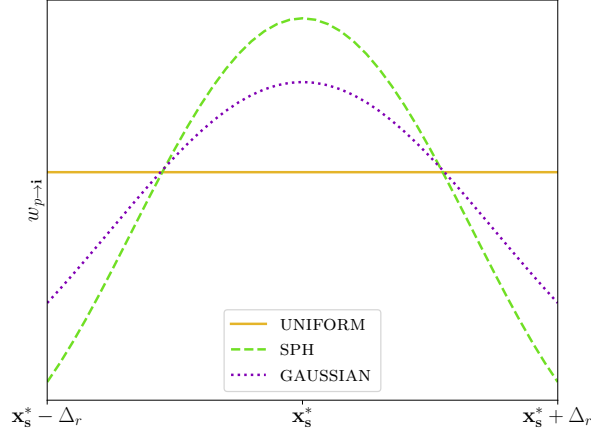


Figure 4.3: *Weighting techniques*. Different weighting techniques for determining particle-to-grid weights, UNIFORM weighting in solid yellow, SPH weighting in dashed green, and GAUSSIAN weighting in dotted purple, are compared. Here the weights $w_{p \rightarrow i}$ were normalized such that $\sum_{\mathbf{x}_s \in \mathcal{S}} w_{p \rightarrow i} |_{\mathbf{x}_s} = 1$, where \mathbf{x}_s are positions of stencil particles.

and may be sufficient for our purposes. We name this technique the UNIFORM weighting, due to its similarity with the discrete uniform distribution from probability theory and statistics.

On the other hand, one can argue that the approximation at \mathbf{x}_s^* , which is collocated with grid node \mathbf{x}_i , should not be too “bad”. Indeed, existing techniques use this approximation, and still manage to produce good results. This means that the other stencil particles, the ones not collocated with the grid node, should only have a supporting role. Their weights should be inferior to that of \mathbf{x}_s^* .

As a second weighting technique, we suggest using the main smoothing kernel in smooth particle hydrodynamics (SPH) designed by Müller et al. [32]

$$w_{p \rightarrow i} |_{\mathbf{x}_s} = \frac{315}{64\pi\Delta^9} \begin{cases} (\Delta^2 - (\mathbf{x}_s - \mathbf{x}_i)^2)^3 & 0 \leq (\mathbf{x}_s - \mathbf{x}_i) \leq \Delta, \\ 0 & \text{otherwise.} \end{cases} \quad (4.5)$$

where $\Delta = \kappa\Delta_r$, with $\kappa = 1.5$ in this document. This technique is named SPH weighting as a reference to the origin of Equation 4.5.

Originally, in Müller et al. [32], the smoothing kernels are designed have a zero value at the boundary. In this context, we absolutely want to avoid this because the stencil particles

on a cell's boundary should contribute to the approximation at \mathbf{x}_i . Otherwise, they would be a complete waste of computational resources since they would still be back-traced, but their contribution would never be used.

By using Δ_r , the cell's circumscribed circle's radius, we extend the kernel's effective region to the cell's corners as opposed to using

$$\max(\Delta x, \Delta y, \Delta z), \quad (4.6)$$

which would discard them. By setting $\kappa > 1$, we secure all stencil particles' contributions to the approximation. Moreover, by choosing $\kappa = 1.5$, we guarantee that the smallest weight corresponds to at least 10% of the largest weight. Knowing that the maximum value is located at the stencil's centre and that the minimum value is located at the boundary, we can use Equation 4.5 to derive a condition on κ

$$\left(1 - \frac{1}{\kappa^2}\right)^3 \geq 10\%. \quad (4.7)$$

According to Equation 4.7, different κ values result in different ratios between the smallest and largest possible weights. While we could investigate the effect of varying κ , we adapt the Gaussian kernel, popular in digital signal processing [22].

This is our last proposed weighting technique, which we name GAUSSIAN weighting. Using \mathcal{N} , the normal (Gaussian) distribution probability density function

$$\mathcal{N}(\mathbf{x}, \mu, \sigma) = \frac{\exp\left(-\frac{(\mathbf{x}-\mu)^2}{2\sigma^2}\right)}{\sigma\sqrt{2\pi}}, \quad (4.8)$$

we define the weights as

$$w_{p \rightarrow i} |_{\mathbf{x}_s} = \mathcal{N}\left(\frac{\mathbf{x}_s - \mathbf{x}_s^*}{\Delta_r}, 0, \sigma\right), \quad (4.9)$$

where σ is Gaussian standard deviation, taken to be 0.75. This value was chosen to make the GAUSSIAN weighting technique approximatively halfway between UNIFORM and SPH weighting techniques, as illustrated in Figure 4.3.

Chapter 5

Results

This chapter reports on the evaluation of our proposed ASLAM. First, we describe the testing scenarios as well as the metrics employed to analyse an exploration of different parameter combinations, and to assess which combination yields overall results superior to the others. These testing scenarios and metrics are then used to compare our method to existing transport techniques. Afterwards, we repeat the comparison with results from a practical application scenario, and discuss the ASLAM’s qualitative performance against the other techniques’. Finally, we recount the outcome of two minor connected investigations.

5.1 Testing Scenarios and Performance Metrics

The testing scenarios introduced hereinafter evaluate the advection of scalar quantities. Consequently, neither pressure projection nor external forces are applied in the corresponding simulations. In particular, unless explicitly mentioned otherwise, all results are produced using $\Delta t = \frac{1}{60}$, and 1200 time steps are computed. Moreover, the simulation domain covers $(x, y) \in [0, 40]^2$, using a cell-centred grid with $\Delta x = \Delta y = 1$ and 40 grid nodes per dimension. Note that we adopt periodic boundaries for all testing scenarios. Thus, all flows are taken to be periodic in space and time. To further explicit the results being presented, we juxtapose icons to the figures, following the descriptions given in Table 5.1.

Our first testing scenario is a simple unidimensional example using the steady linear flow $\mathbf{u} = (1, 0, 0)$. The initial shape matches the radial cross-section of an annulus of inner radius 4 and outer radius 8, centred at $x = 10$. Figure 5.1 shows animation frames of the expected behaviour. Note that the flow preserves the initial condition’s profile.










Icon	Description
	1d-TRANSLATION test
	2d-TRANSLATION test
	2d-ROTATION test
	Analytical solution
	$n \in [1, 60]$, Z stands for “zoom”
	$n \in [1, 1200]$, F stands for “full”

Table 5.1: *Description of testing icons.* To emphasize which test results are displayed, we use the icons depicted herein. Note that  and  are only used to qualify performance charts. Moreover, since charts with icon  also includes interval $n \in [1, 60]$, said interval is shaded in gray to highlight the fact that it was shown in greater details at some point.

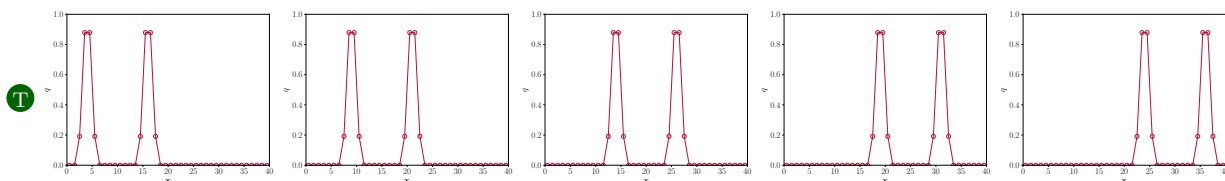


Figure 5.1: *Analytical solution to the 1d-TRANSLATION test.* The initial shape, corresponding to the radial cross-section of an annulus of inner radius 4 and outer radius 8, centred at $x = 10$, is advected by a constant flow $\mathbf{u} = (1, 0, 0)$. The analytical solution moves the initial condition in the positive x -axis direction, without deformation, as shown for (from left to right) $n = 0, 300, 600, 900, 1200$.

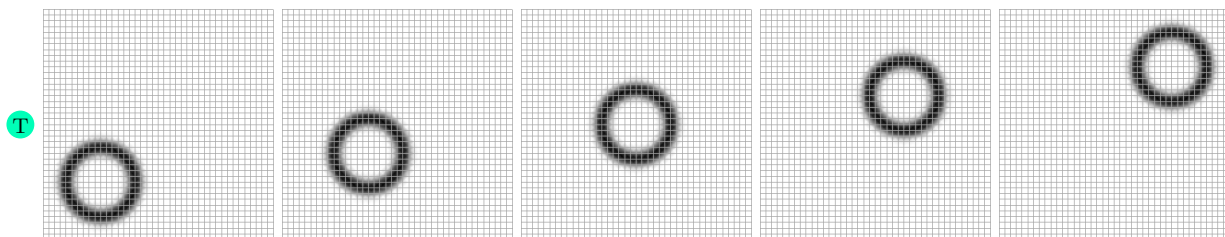


Figure 5.2: *Analytical solution to the 2d-TRANSLATION test.* The initial shape, an annulus of inner radius 4 and outer radius 8, positioned at the bottom left corner of the grid, is advected by a constant flow $\mathbf{u} = (1, 1, 0)$. The analytical solution moves the initial condition from the bottom left corner to the top right corner of the grid, without deformation, as shown for (from left to right) $n = 0, 300, 600, 900, 1200$.

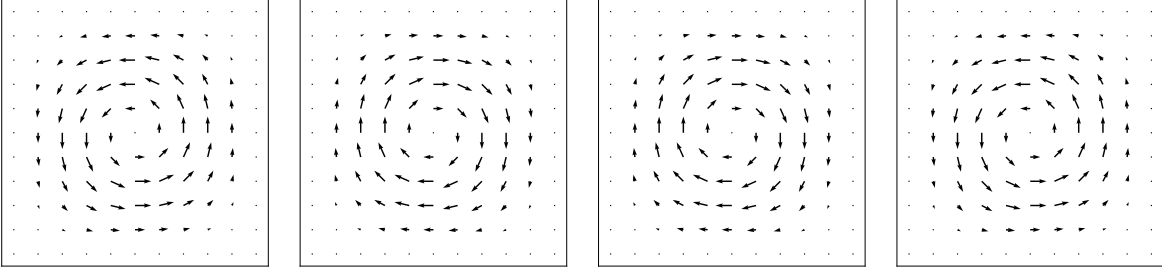


Figure 5.3: *Nonlinear flow used for the 2d-ROTATION test.* The vortex-in-a-box flow of Enright et al. [13], as defined by Equation 5.1 with $N = 600$, is shown for (from left to right) $n = 150, 450, 750, 1050$. First, the initial conditions are rotated counter-clockwise with respect to centre of the simulation domain. Then, they are brought back to their original location with an inverse clockwise rotation with respect to the centre of the simulation domain. They are further rotated clockwise with respect to the centre of the simulation domain before being brought back once more to their original location by a counter-clockwise rotation with respect to centre of the simulation domain. Note that the vortex-in-a-box flow does not preserve the shape of the initial conditions (i.e., it is not a rigid body rotation).

In two dimensions, the first testing scenario is the bidimensional equivalent of the 1d-TRANSLATION test. The initial shape is an annulus of inner radius 4 and outer radius 8, whose centre is positioned at the bottom left corner of the grid. It is transported by linear flow $\mathbf{u} = (1, 1, 0)$, as illustrated in Figure 5.2. Similar to the unidimensional scenario, the translational flow preserves the shape of the annulus.

The last testing scenario, also the second test in two dimensions, uses the vortex-in-a-box flow of Enright et al. [13], defined by

$$\mathbf{u} = 2 \cos\left(\frac{\pi n}{N}\right) (\sin^2(\pi \bar{x}) \sin(\pi \bar{y}) \cos(\pi \bar{y}), -\sin(\pi \bar{x}) \cos(\pi \bar{x}) \sin^2(\pi \bar{y}), 0), \quad (5.1)$$

where N is the period time, chosen to be 600 for all the examples. \bar{x} and \bar{y} are normalized position coordinates, such that $(\bar{x}, \bar{y}) \in [0, 1]^2$. This nonlinear flow (i.e., Equation 5.1) is illustrated for a few time steps in Figure 5.3.

Unlike in the 1d-TRANSLATION and 2d-TRANSLATION tests, the shape of the initial condition is not preserved throughout the entire simulation. The initial shape corresponds to an annulus of inner radius 4 and outer radius 8, whose centre is positioned at the top centre of the grid. We do not know the analytical solution to this setup for all time steps. We only

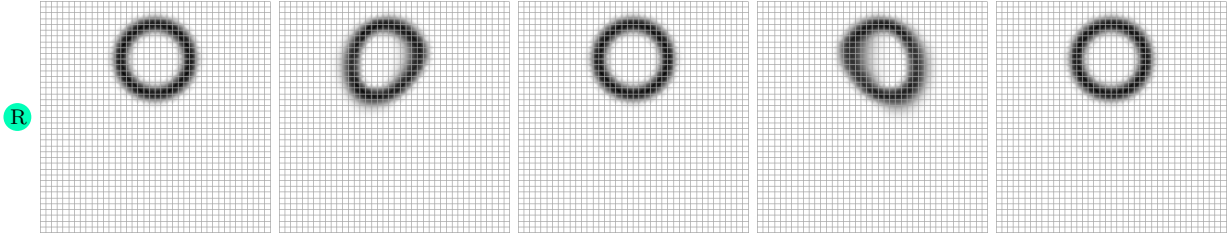


Figure 5.4: *Pseudo-analytical solution to the 2d-ROTATION test.* The initial shape, an annulus of inner radius 4 and outer radius 8, positioned at the top centre of the grid, is advected by the flow given by Equation 5.1 as shown for (from left to right) $n = 0, 300, 600, 900, 1200$. In this context, the analytical solution is only known at $n = 0, 600, 1200$, when the initial conditions are recovered. Thus, frames shown for $n = 300, 900$, which are taken from the numerical solution computed using MacCormack advection with semi-Lagrangian advection as a stepping scheme, are merely illustrative. The flow first deforms the annulus by dragging it counter-clockwise. This deformation is then reversed, hence restoring the annulus to its original position and shape. Next, the flow deforms the annulus by dragging it clockwise before once again bringing it back to its initial configuration.

have the analytical solution for time steps which are multiples of N (i.e., $n = 0, 600, 1200$ in this case). Figure 5.4 presents a tentative analytical solution.

In order to assess our method's performance numerically, we use three different norm-based metrics, namely the L^1 -norm, the L^2 -norm, and the L^∞ -norm, as well as a conservation metric, the conservation of density $\% \rho$. We define Q to be the analytical solution. For all norm-based metrics, we compute each metric's value at each time step for which we are aware of the analytical solution. For the 1d-TRANSLATION and 2d-TRANSLATION tests, we always know the analytical solution, but, for the 2d-ROTATION test, as mentioned previously, this is only true at $n = 600$ and $n = 1200$. The L^1 -norm is

$$\sum_{\mathbf{i}} \text{abs}(q|_{\mathbf{x}_i} - Q|_{\mathbf{x}_i}). \quad (5.2)$$

It expresses the total error in the grid as the sum of each grid cell's error. The L^2 -norm is

$$\sqrt{\sum_{\mathbf{i}} (q|_{\mathbf{x}_i} - Q|_{\mathbf{x}_i})^2}. \quad (5.3)$$

Also commonly known as the Euclidean norm, it, not unlike the L^1 -norm, measures the total error in the grid using each grid cell's error. Both the L^1 -norm and the L^2 -norm

account for both the positional and amplitude inaccuracies. The L^∞ -norm is defined as

$$\max_{\mathbf{i}} \text{abs} (q|_{\mathbf{x}_i} - Q|_{\mathbf{x}_i}). \quad (5.4)$$

It corresponds to the worst error in the grid, and illustrates the general worst case scenario. For these norm-based metrics (i.e., Equation 5.2, Equation 5.3, and Equation 5.4), lower values clearly translate into better accuracy.

Finally, in all our testing scenarios, the quantity q corresponds to density ρ . An ideal scheme would conserve density. At each time step, we compare the amount of density to the initial amount ρ^0 , and express it as a percentage

$$\% \rho = 100 \frac{\rho}{\rho^0}. \quad (5.5)$$

Evidently, smaller variations in conservation of density $\% \rho$ are preferred. Even if our scheme is not exactly conservative by design, as is the case for some schemes (e.g., Lentine et al. [30]), a low degree of deviation is nonetheless desirable because advected quantities should be conserved by divergence-free flows.

5.2 ASLAM Parameter Assessment

In this section, we investigate two parameters of the ASLAM, namely parametric size ζ and particle-to-grid weights $w_{p \rightarrow \mathbf{i}}$. More specifically, we limit the parametric size to $\zeta = 1, 2, 3, 4$, and often drop the qualifying adjective “parametric” for brevity. The weights are computed according to the three weighting techniques described in Section 4.2. We deliberately choose to restrict our stencil distribution to the CHESSBOARD type, as it covers the whole area of the cell, unlike the MANHATTAN type, and it always includes the point of interest, unlike odd-size SUBSAMPLING type (refer to Figure 4.2 for a reminder). Indeed, preliminary results (not shown in this thesis) suggest that the stencil distribution should at least fully sample the control volume associated with the grid node, and include a stencil particle at the location of the grid node. The CHESSBOARD type meets both these requirements.

Before presenting the results, we would like to address a few items regarding the interpretation of the subsequent performance charts (i.e., charts reporting metric measurements as a function of time step n). Since the advection error accumulates identically at each time step, performance charts should always be read from left to right, with more importance given to the earliest time steps. To facilitate this, the upcoming performance charts’

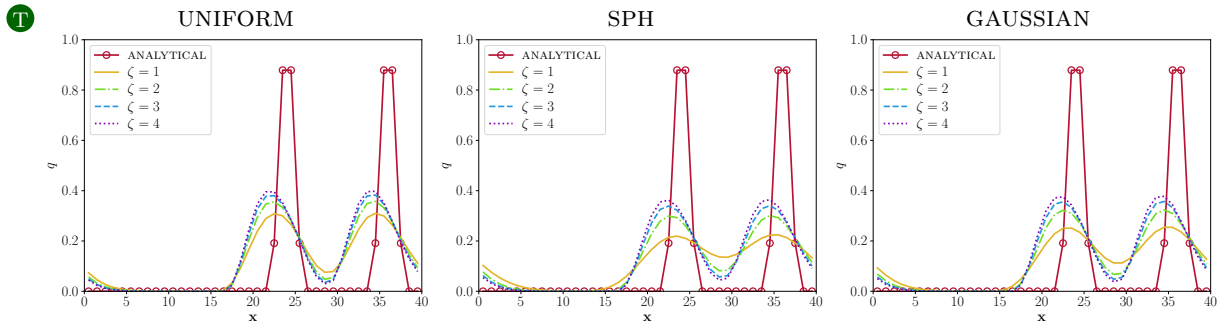


Figure 5.5: *Impact of parametric size by weighting type for the 1d-TRANSLATION test.* At $n = 1200$, the analytical solution in solid red with empty circular markers is compared to the ASLAM with parametric size $\zeta = 1, 2, 3, 4$ respectively in solid yellow, dot-dashed green, dashed blue, and dotted purple for (from left to right) UNIFORM, SPH, and GAUSSIAN weights. For all weighting techniques, increasing ζ improves the numerical solution's accuracy in terms of amplitude, but shifts the numerical solution upwind the analytical's.

n -axes are restricted to $n \in [1, 60]$, although simulations are run for 1200 time steps. The complete performance charts, showing $n \in [1, 1200]$, are included in Appendix B.

We start with some results for the previously described 1d-TRANSLATION test. Figure 5.5 shows that, for all weighting techniques, increasing ζ better preserves the analytical solution's extrema. This observation concurs with the intuition that more stencil particles should be able to capture a better picture of the scalar field than fewer. Moreover, given that, for each unit increment of ζ , two more particles are added to the stencil (refer to Table 4.1 for a reminder), the improvement is more visually significant between parametric sizes $\zeta = 1$ and $\zeta = 2$. However, solutions with larger ζ also exhibit greater deceleration as evidenced by the fact that the extrema are positioned slightly to the left of their analytical counterparts.

As depicted in Figure 5.6, the performance, as measured by the norm-based metrics, is not significantly different across weighting types. For norm-based metrics, sizes $\zeta > 1$ typically produce very similar curves, much harder to distinguish from one another than from the $\zeta = 1$ curve. In some cases, such as the L^∞ -norm and L^2 -norm for UNIFORM weights, the smallest size, $\zeta = 1$, even outperforms the largest size, $\zeta = 4$, in the earliest time steps. Additionally, larger sizes do not always translate to better performance according to the norm-based metrics. For instance, $\zeta = 2$ outperforms larger sizes in L^∞ -norm and L^2 -norm for both GAUSSIAN and SPH weights. Interestingly, the smallest size is better at preserving density for all weighting types; as ζ increases, conservation of density worsens. The complete performance charts in Figure B.1 concur with the above observations.

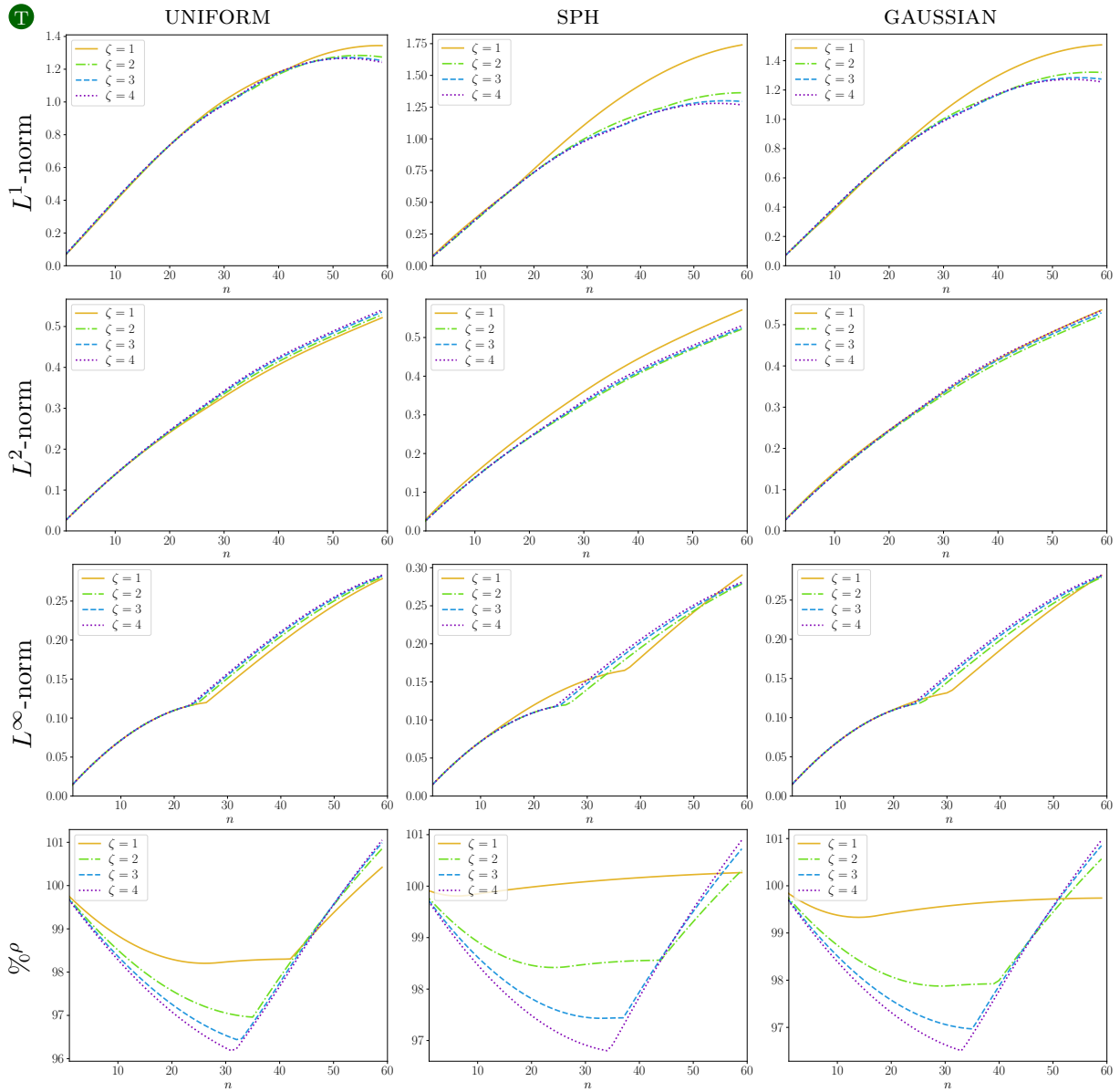


Figure 5.6: *Performance by weighting type for the 1d-TRANSLATION test.* Performance for Figure 5.5 is measured by (from top to bottom) the L^1 -norm, the L^2 -norm, the L^∞ -norm, and density conservation $\%^\rho$ for (from left to right) UNIFORM, SPH, and GAUSSIAN weights with parametric size $\zeta = 1, 2, 3, 4$ respectively in solid yellow, dot-dashed green, dashed blue, and dotted purple. For all norms, $\zeta = 1$ distinguishes itself from $\zeta > 1$. Larger sizes are not always more accurate, and worsen density conservation. The complete performance charts are available in Figure B.1.

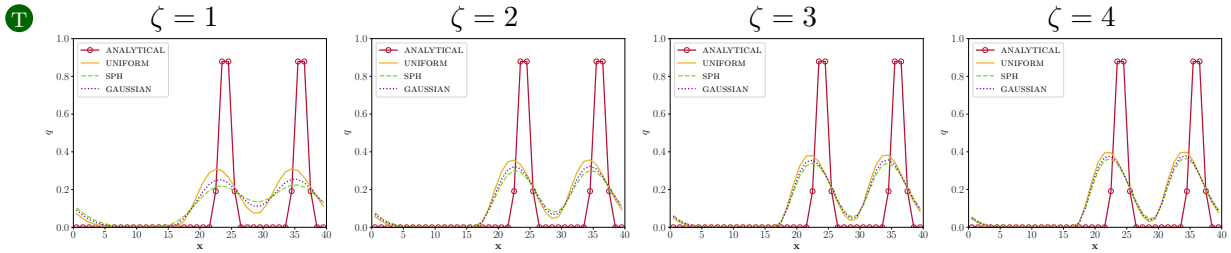


Figure 5.7: *Impact of weighting type by parametric size for the 1d-TRANSLATION test.* At $n = 1200$, the analytical solution in solid red with empty circular markers is compared against the ASLAM with UNIFORM weights in solid yellow, SPH weights in dashed green, and GAUSSIAN weights in dotted purple for (from left to right) parametric size $\zeta = 1, 2, 3, 4$. Across all sizes, UNIFORM weights better preserve the analytical solution’s extrema while SPH weights are the worst, but the discrepancy in accuracy decreases with increasing size.

Plotting the results by size rather than by weight, as in Figure 5.7, allows for a better comparison of the different weighting types. For all sizes considered, the UNIFORM weights appear to better preserve the analytical solution’s extrema, but this advantage is reduced as the parametric size increases. Nevertheless, SPH weighting results in the more damped numerical solution in terms of amplitude, but the less decelerated one as evidenced by its extrema, which are more collocated with the analytical solution’s than when other weighting techniques are employed.

Looking at the corresponding norm-based metrics in Figure 5.8, the performance is very similar across different parametric sizes. Besides, across sizes $\zeta > 1$, for the norm-based metrics, all weighting techniques typically produce very similar curves, much harder to distinguish from one another than from the $\zeta = 1$ curve, for which UNIFORM weighting is best. However, in terms of density conservation, the UNIFORM weights’ performance is surpassed by the other two weighting techniques, SPH being the best all sizes considered. Figure B.2, showing the complete performance charts, agrees with these remarks too.

Summarizing our observations for the 1d-TRANSLATION test, we conclude that, although increasing the parametric size ζ increases the apparent matching of the numerical solution with the analytical solution, the improvement is more significant between $\zeta = 1$ and $\zeta = 2$ according to the metrics. Therefore, since increasing the size also increases the computational cost, and smaller sizes are positively correlated with better density conservation, a size $\zeta = 2$ is preferable. Regarding the different weighting techniques available, SPH weights outperform both UNIFORM and GAUSSIAN weights in terms of density conservation for all sizes, but they sometimes perform slightly worse in the L^1 -norm, for all sizes,

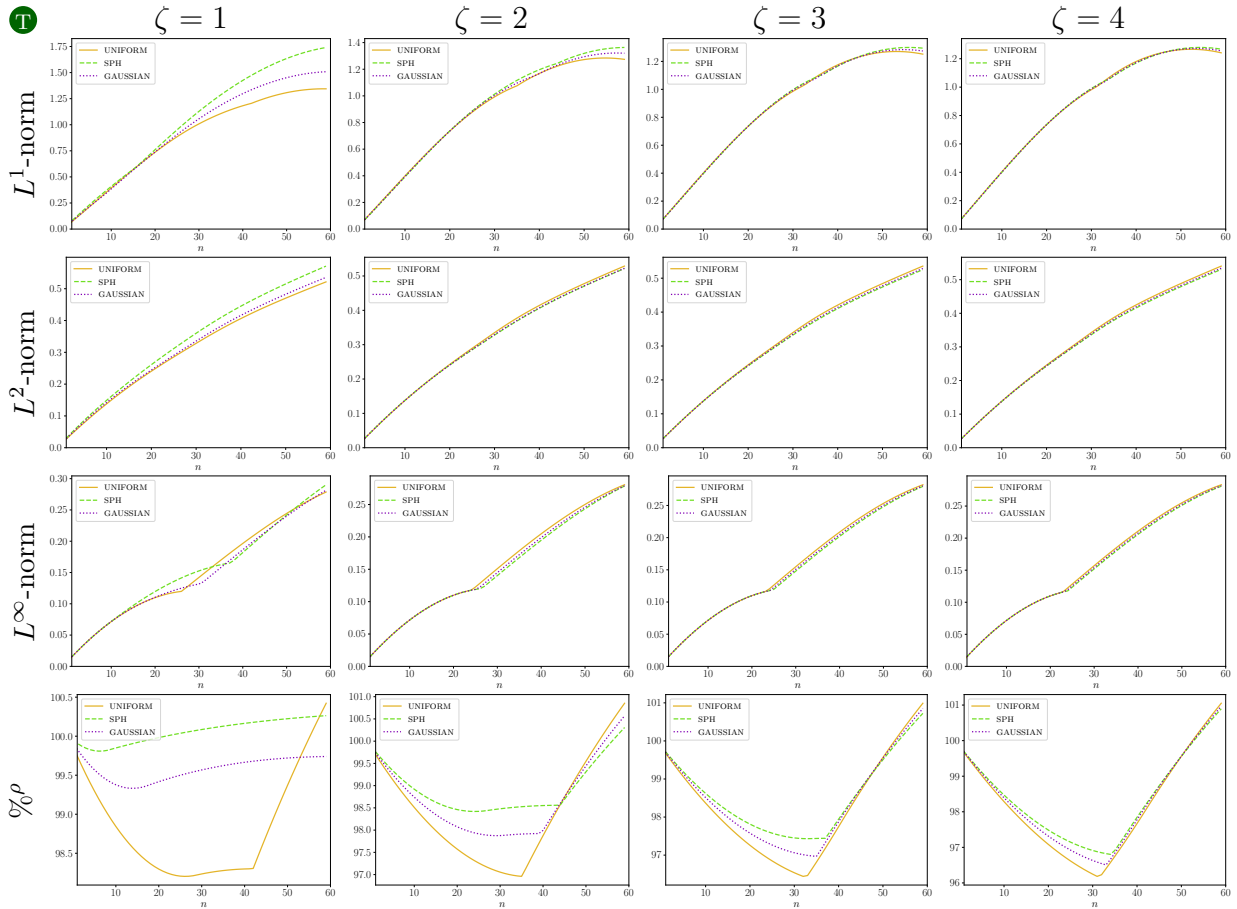


Figure 5.8: *Performance by parametric size for the 1d-TRANSLATION test.* Performance for Figure 5.7 is measured by (from top to bottom) the L^1 -norm, the L^2 -norm, the L^∞ -norm, and density conservation $\% \rho$ for (from left to right) parametric size $\zeta = 1, 2, 3, 4$ with UNIFORM, SPH, and GAUSSIAN weights respectively in solid yellow, dashed green, and dotted purple. Performance is very similar across sizes, but, under the norm-based metrics, UNIFORM weights are better when $\zeta = 1$. They are nevertheless the worst at conserving density, where SPH weights are best. Figure B.2 presents the complete performance charts.

and for $\zeta = 1$.

Now that we have a general idea of the ASLAM’s performance in one dimension, we want to determine if our conclusions directly extend to two dimensions. Unlike in one dimension, we can assess the method’s performance in the context of nonlinear flows, such as the 2d-ROTATION test described earlier. Given that the original APIC method was known to exactly conserve angular momentum, we expect the ASLAM to handle well nonlinear flows.

While Figure 5.9 shows the bidimensional results for the 2d-TRANSLATION test, Figure 5.10 shows the bidimensional results for the 2d-ROTATION test. In both cases and identically to the unidimensional results, increasing parametric size ζ decreases dissipation, as evidenced by the better-preserved empty region in the middle of the annulus. Yet again, the improvement appears more significant between $\zeta = 1$ and $\zeta = 2$, although $8s$ stencil particles are added for each ζ unit increment to $\zeta = s$. This confirms that there is a limit after which increasing the number of stencil particles does not provide any additional visual improvement of the numerical solution, but still costs additional computational resources to perform the advection. Nonetheless, for the 2d-ROTATION test, we observe a “squarification”, for lack of a better word, of the annulus (e.g., the annulus appears to have corners and flat boundaries). This “squarification” is perceptibly worse as ζ increases. Perhaps the reduced dissipation brings to light this other, formerly hidden, artefact, or it is a byproduct of the stencil distribution type. In terms of weighting techniques, it is visually rather difficult to pick a best option. Besides, neither Figure 5.9 nor Figure 5.10 is particularly suited to determine how well collocated the numerical solution is with the analytical solution. Thus, we turn to the metrics to address both these shortfalls.

We have twice determined that increasing the parametric size reduces dissipation, and that $\zeta = 2$ offers the best improvement relative to the number of stencil particles added. Therefore, we focus on comparing weighting techniques as illustrated by Figure 5.11 and Figure 5.12, respectively for the 2d-TRANSLATION test and the 2d-ROTATION test. Under a purely translational (i.e., linear) flow, SPH weighting bests both the other weighting techniques for all metrics computed. Under a nonlinear flow, this observation holds except for $\zeta = 1$ and $\zeta = 2$ according to the L^∞ -norm. So, we can reasonably conclude that SPH weighting probably behaves best under more generic cases. The complete performance charts for Figure 5.11 and Figure 5.12 are displayed respectively in Figure B.3 and Figure B.4. We also include the comparison of parametric size ζ in Appendix B.

In summary, we argue that the optimal parameter combination for the ASLAM is a parametric size $\zeta = 2$ with SPH weights. Henceforth, we shall use these settings to first investigate the stability of our proposed method, and later to compare it to existing

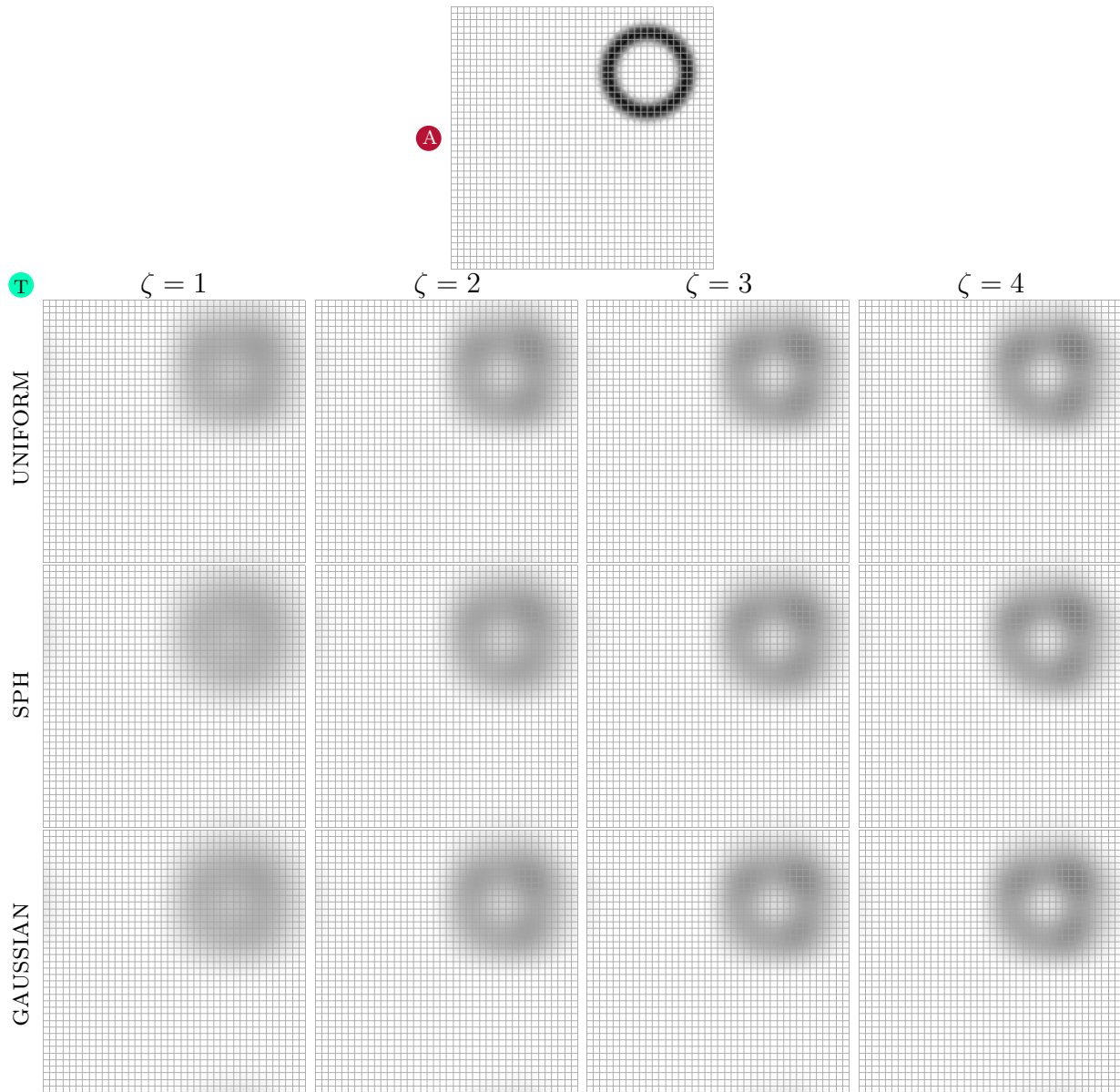


Figure 5.9: *Impact of parametric size and weighting type for the 2d-TRANSLATION test.* At $n = 1200$, the analytical solution (top) is compared to numerical solutions computed with (from left to right) parametric size $\zeta = 1, 2, 3, 4$ and (from top to bottom) UNIFORM, SPH, and GAUSSIAN weights. Extrema are better preserved, and less dissipation is incurred with increasing ζ as evidenced by the darker regions and emptier annuli's centres. However, results are very similar across weighting types.

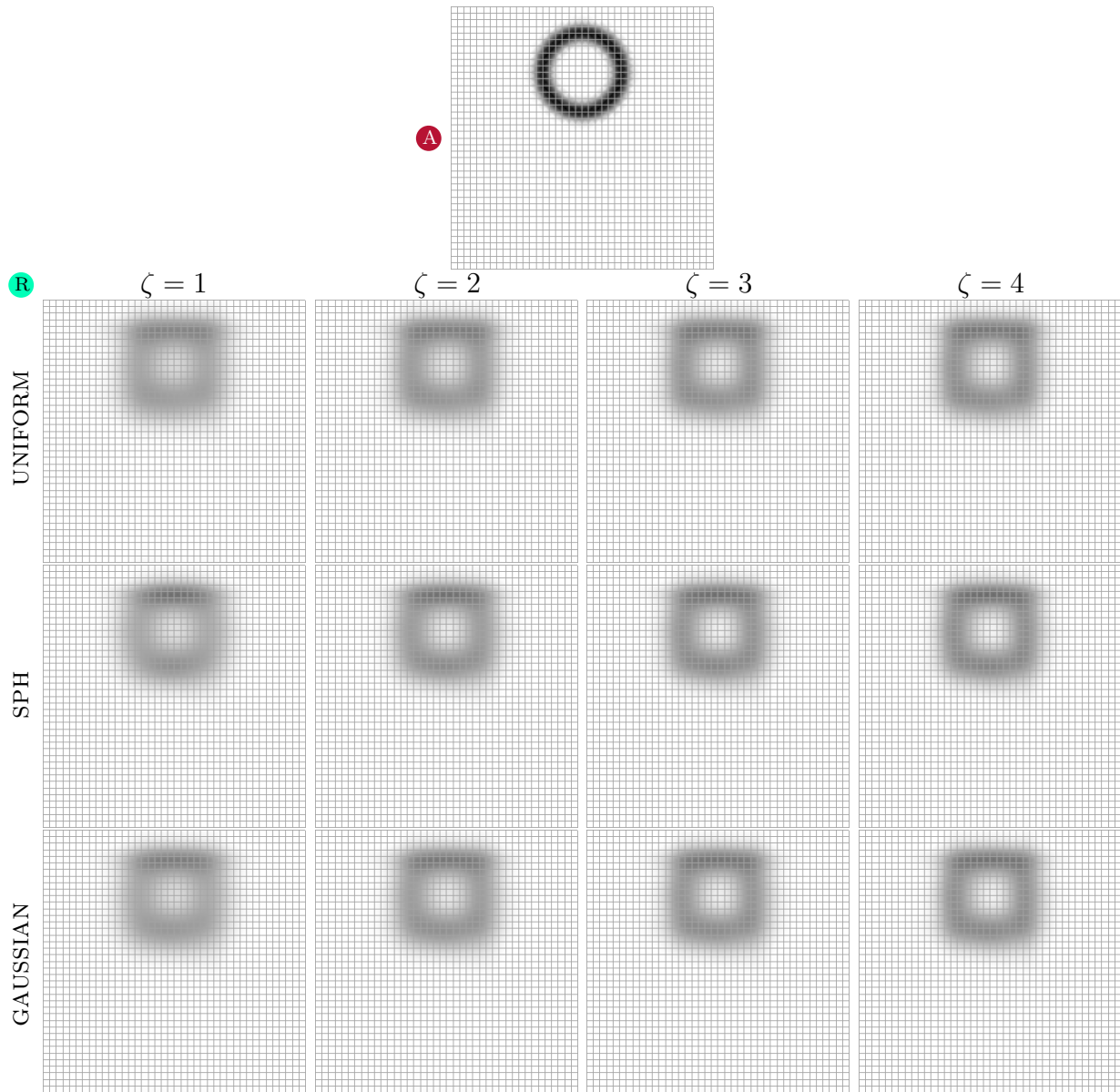


Figure 5.10: *Impact of parametric size and weighting type for the 2d-ROTATION test.* At $n = 1200$, the analytical solution (top) is compared to numerical solutions computed with (from left to right) parametric size $\zeta = 1, 2, 3, 4$ and (from top to bottom) UNIFORM, SPH, and GAUSSIAN weights. Increasing ζ better preserves extrema, and is less dissipative as depicted by the darker regions and emptier annuli's centres. Yet, it also introduces some “squarification” of the initial shape. Otherwise, results look similar across weighting types.

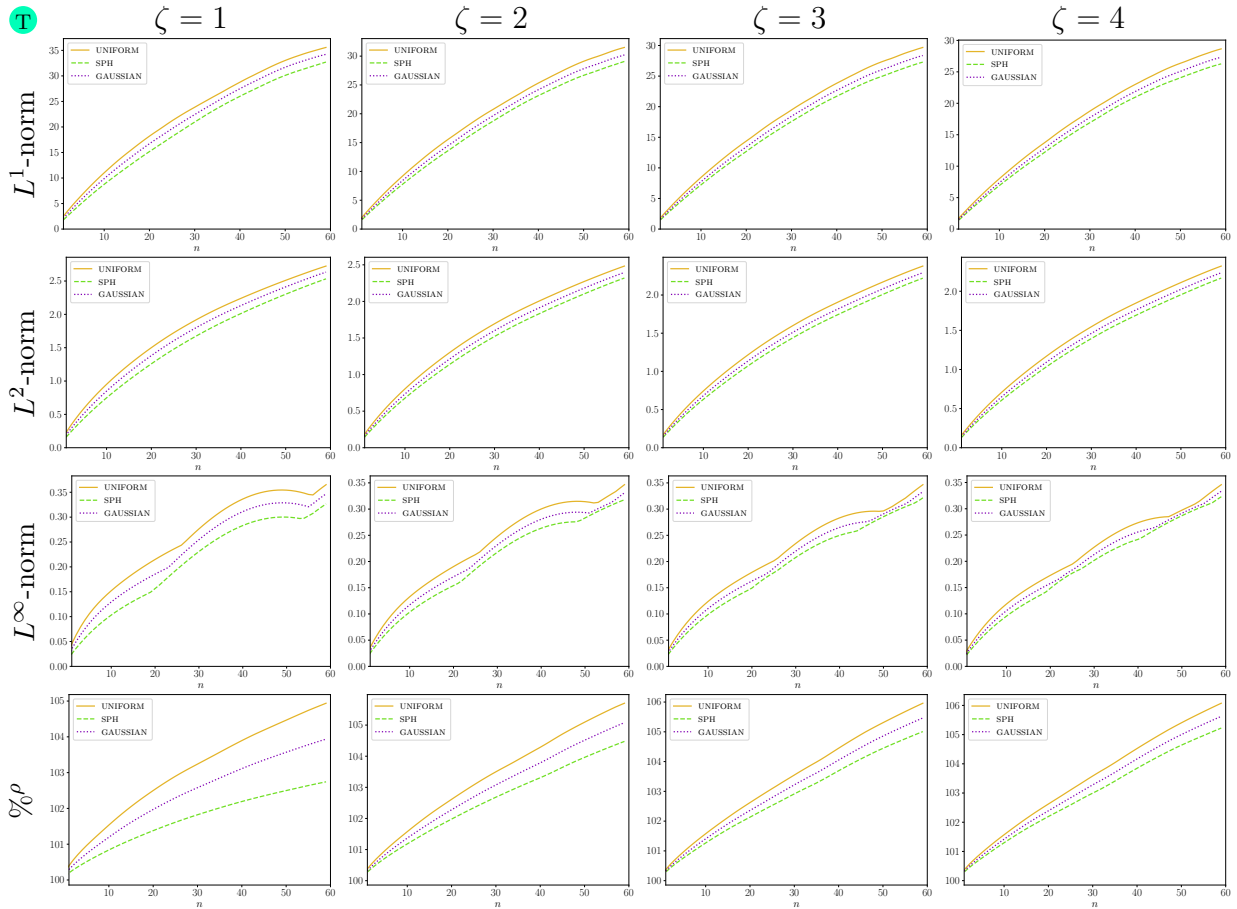


Figure 5.11: *Performance by parametric size for the 2d-TRANSLATION test.* Performance for Figure 5.9 is measured by (from top to bottom) the L^1 -norm, the L^2 -norm, the L^∞ -norm, and density conservation $\% \rho$ for (from left to right) parametric size $\zeta = 1, 2, 3, 4$ with UNIFORM, SPH, and GAUSSIAN weights respectively in solid yellow, dashed green, and dotted purple. For all metrics and all sizes, SPH weights are more accurate than both UNIFORM and GAUSSIAN weights. The complete performance charts are available in Figure B.3.

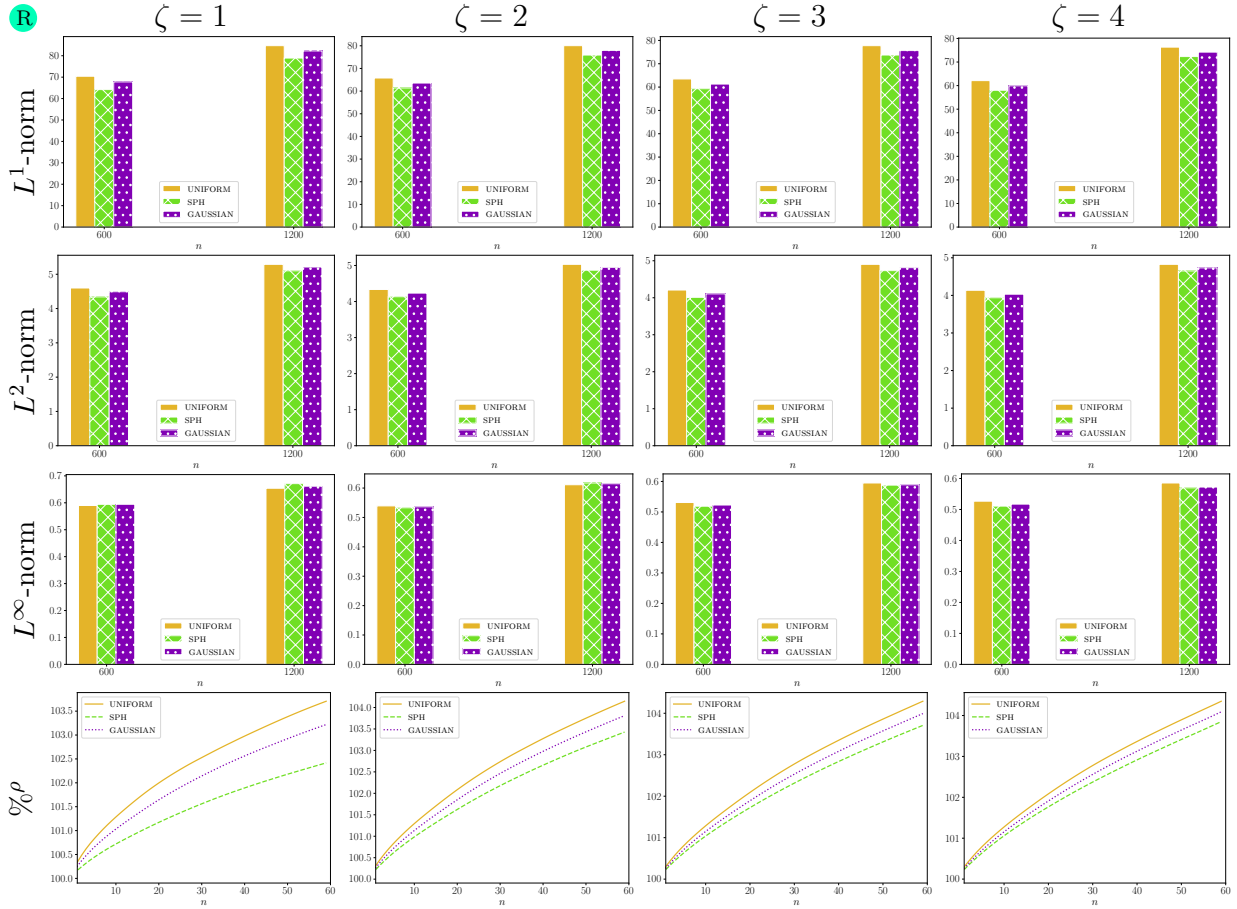


Figure 5.12: *Performance by parametric size for the 2d-ROTATION test.* Performance for Figure 5.10 is measured by (from top to bottom) the L^1 -norm, the L^2 -norm, the L^∞ -norm, and density conservation $\%^\rho$ for (from left to right) parametric size $\zeta = 1, 2, 3, 4$ with UNIFORM, SPH, and GAUSSIAN weights respectively in solid yellow, dashed green, and dotted purple. For all metrics and all sizes, except for the L^∞ -norm with sizes $\zeta = 1, 2$, SPH weights are more accurate than both UNIFORM and GAUSSIAN weights. Figure B.4 displays the complete performance charts.

Δx	L^1 -norm	Convergence Rate	L^2 -norm	Convergence Rate	L^∞ -norm	Convergence Rate
1	7.591e+01	-	4.869e+00	-	6.173e-01	-
1/2	4.109e+01	8.856e-01	3.073e+00	6.641e-01	4.228e-01	5.460e-01
1/4	1.930e+01	1.090e+00	1.618e+00	9.256e-01	2.503e-01	7.562e-01
1/8	8.838e+00	1.127e+00	7.880e-01	1.038e+00	1.234e-01	1.020e+00
1/16	3.249e+00	1.444e+00	3.004e-01	1.392e+00	5.115e-02	1.271e+00

Table 5.2: L^1 -norm, L^2 -norm, and L^∞ -norm error values and convergence rates under grid refinement. The error values are measured at $n = 1200$ for the 2d-ROTATION test. For all norms, the error decreases with decreasing grid cell size.

advection techniques. So, we study the numerical rate of convergence of our method under grid refinement using norm-based metrics. To account for the change in grid size $\Delta \mathbf{x}$, we redefine the L^1 -norm and L^2 -norm respectively as

$$L^1\text{-norm} = \sum_{\mathbf{i}} \text{abs}(q|_{\mathbf{x}_i} - Q|_{\mathbf{x}_i}) \Delta, \quad (5.6)$$

and

$$L^2\text{-norm} = \sqrt{\sum_{\mathbf{i}} (q|_{\mathbf{x}_i} - Q|_{\mathbf{x}_i})^2 \Delta}, \quad (5.7)$$

where Δ is the size of a single grid cell. Since we use the previously defined 2d-ROTATION test, Δ corresponds to the area of a grid cell. The above definitions (i.e., Equation 5.6 and Equation 5.7) ensure that we are measuring the error in an integral sense, independently of the grid's resolution. We compute the convergence rate under grid refinement for each norm-based metric. The results are presented in Table 5.2, which confirms that the error decreases as the cell size decreases, thus meeting the expected behaviour.

According to the L^1 -norm and L^2 -norm, our ASLAM appears to exhibit linear convergence, perhaps even slightly more than linear as can be seen from Figure 5.13. Even the worst case scenario, represented by the L^∞ -norm, shares this behaviour. However, our proposed method is definitely not second-order accurate. While PIC is known to be at least first-order accurate, and can be improved as in Edwards and Bridson [12], the order of APIC itself has not been investigated, to our knowledge. In the ASLAM, as illustrated many times, the error, for all norms considered, clearly depends on the parametric size ζ , on the weighting technique utilized as well as on the stencil distribution type, whose anal-

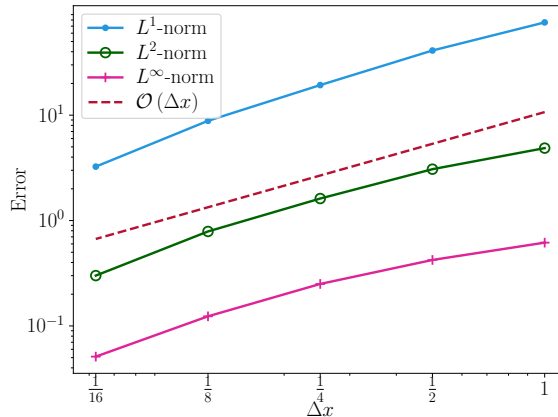


Figure 5.13: *Convergence of the ASLAM at $n = 1200$ for the 2d-ROTATION test.* The error is computed by the L^1 -norm in solid blue with circular markers, the L^2 -norm in solid green with empty circular markers, and the L^∞ -norm in solid pink with plus markers. Linear convergence is plotted in dashed red; the slope of the curve is of interest, not its vertical positioning. Under all norms, the convergence rate is slightly superior to linear. Note that the axes are logarithmically scaled.

ysis we discarded early on. As a result, the exact convergence rate of our method may not be known with greater certainty until further investigations are conducted.

5.3 Comparative Analysis of Advection Methods

Having assessed the ASLAM’s various parameter combinations, we now compare our proposed method to two other unconditionally stable Eulerian advection techniques most commonly used in computer animation, namely semi-Lagrangian advection (refer to Section 3.1 for a reminder) and MacCormack advection (refer to Section 3.2 for a reminder), both with linear and cubic interpolation. For succinctness, we abbreviate each method in accordance with Table 5.3.

Furthermore, we reuse the testing scenarios described in Section 5.1. However, since different methods are assessed, their advection error may not accumulate identically. Therefore, we analyse their performance in the scope of both the earliest time steps $n \in [1, 60]$ and the full simulation $n \in [1, 1200]$. To further emphasise the difference between both settings, performance charts are labelled with an icon, presented in Table 5.1, indicating which time interval is displayed.

Abbreviation	Method
SL-LIN	Semi-Lagrangian advection with linear interpolation
ASLAM	The ASLAM with linear interpolation and a CHESSBOARD stencil of size $\zeta = 2$ as well as SPH weights
MC-LIN	MacCormack advection with semi-Lagrangian advection as a stepping scheme and linear interpolation
SL-FC	Semi-Lagrangian advection with the monotone cubic interpolation of Fritsch and Carlson [19]
MC-FC	MacCormack advection with semi-Lagrangian advection as a stepping scheme and the monotone cubic interpolation of Fritsch and Carlson [19]

Table 5.3: *Abbreviations for the advection methods.* The above abbreviations refer to the advection methods being compared. Save for ASLAM, we use the hyphen to separate between the advection scheme itself and the interpolation method used.

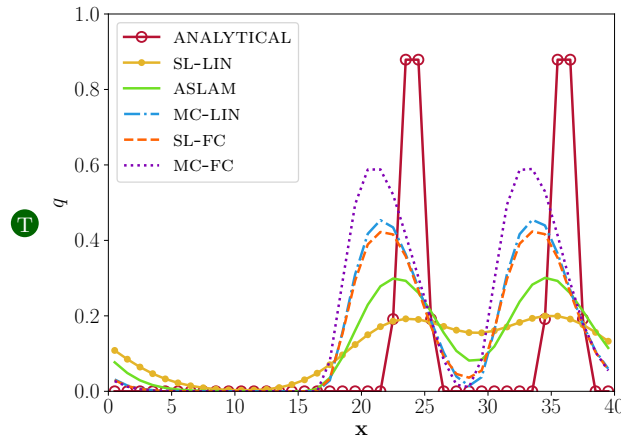


Figure 5.14: *Impact of advection method for the 1d-TRANSLATION test.* The analytical solution in solid red with empty circular markers is compared to the numerical solutions computed with SL-LIN in solid yellow with circular markers, ASLAM in solid green, MC-LIN in dot-dashed blue, SL-FC in dashed orange, and MC-FC in dotted purple. MacCormack advection (i.e., MC-LIN and MC-FC) is less dissipative than semi-Lagrangian advection (i.e., SL-LIN and SL-FC); ASLAM only outperforms SL-LIN. The monotone cubic interpolation of Fritsch and Carlson [19] (i.e., SL-FC and MC-FC) also reduces dissipation compared to linear interpolation (i.e., SL-LIN, ASLAM, and MC-LIN). However, less dissipative solutions exhibit greater deceleration with respect to the analytical solution.

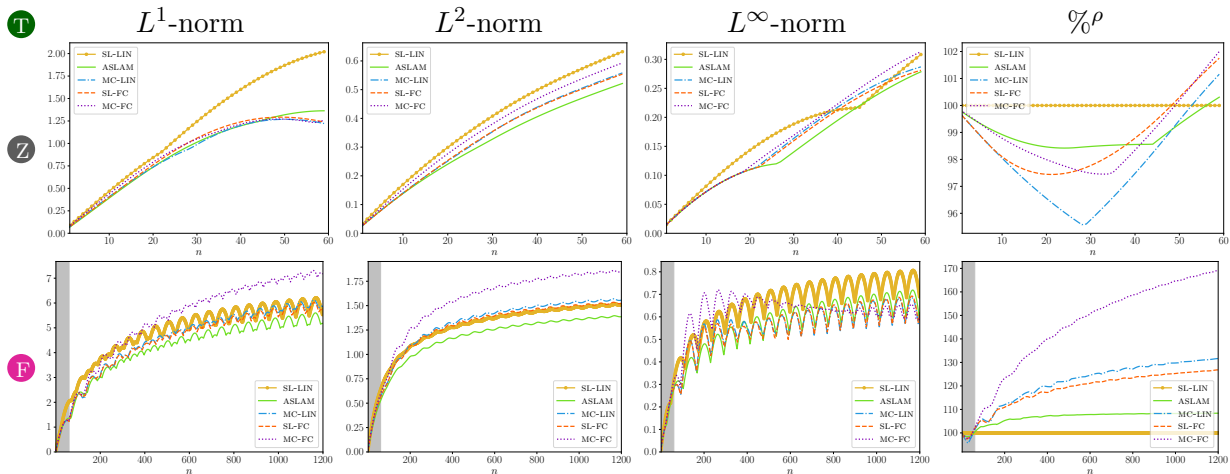


Figure 5.15: *Performance by metric for the 1d-TRANSLATION test.* Performance for Figure 5.14 measured by (from left to right) the L^1 -norm, the L^2 -norm, the L^∞ -norm, and density conservation $\% \rho$ for SL-LIN in solid yellow with circular markers, ASLAM in solid green, MC-LIN in dot-dashed blue, SL-FC in dashed orange, and MC-FC in dotted purple is shown for (from top to bottom) $n \in [1, 60]$ and $n \in [1, 1200]$. Note that the top row charts are zoomed-in versions of the area shaded in gray in the bottom row charts. ASLAM bests all other methods in terms of norms for the earliest time steps, but SL-LIN better conserves density.

Similar to the parameter assessment, we begin with an example in one dimension. Figure 5.14 illustrates that the MacCormack advection (i.e., MC-LIN and MC-FC) better preserves the solution’s extrema than the other approaches (i.e., SL-LIN, ASLAM, and SL-FC). As expected, using a higher-order accurate interpolation scheme (i.e., the monotone cubic interpolation of Fritsch and Carlson [19]) also reduces the numerical dissipation. We can see that SL-FC and MC-FC both outperform their linear interpolation counterparts, SL-LIN and MC-LIN. Visually, ASLAM bests SL-LIN since the latter almost destroys the gap between the maximum extrema. Once again, we observe that better-preserved extrema are less collocated with the analytical solution’s.

This explains why, as we look at the metrics in Figure 5.15, ASLAM outperforms all other methods, for all norm-based metrics, in both the earliest time steps and the full simulation. Regarding conservation of density, however, it follows in second best place behind SL-LIN, and MC-FC is the worst. We also note that MC-LIN performs better than its cubic counterpart, MC-FC, under all the norms, probably because its extrema are more collocated with the analytical solution’s.

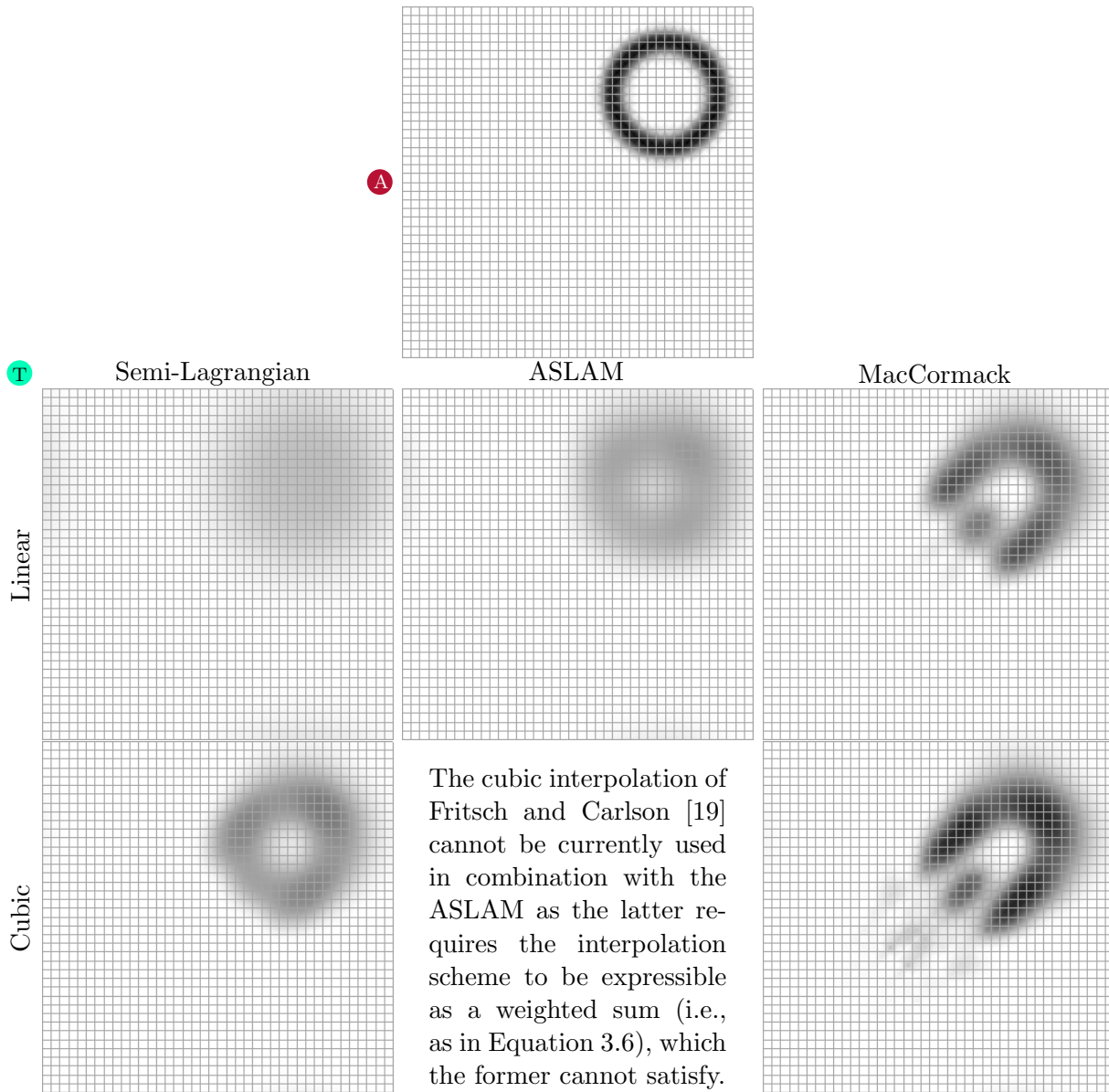


Figure 5.16: *Impact of advection method for the 2d-TRANSLATION test.* At $n = 1200$, the analytical solution (top) is compared to numerical solutions computed using (from left to right and from top to bottom) SL-LIN, ASLAM, MC-LIN, SL-FC, and MC-FC. ASLAM is less dissipative than SL-LIN, but more dissipative than SL-FC, as evidenced by the better-preserved annulus' empty centre. MacCormack advection (i.e., MC-LIN and MC-FC) better preserves extrema as shown by the darker colour, but also exhibits a trailing artefact.

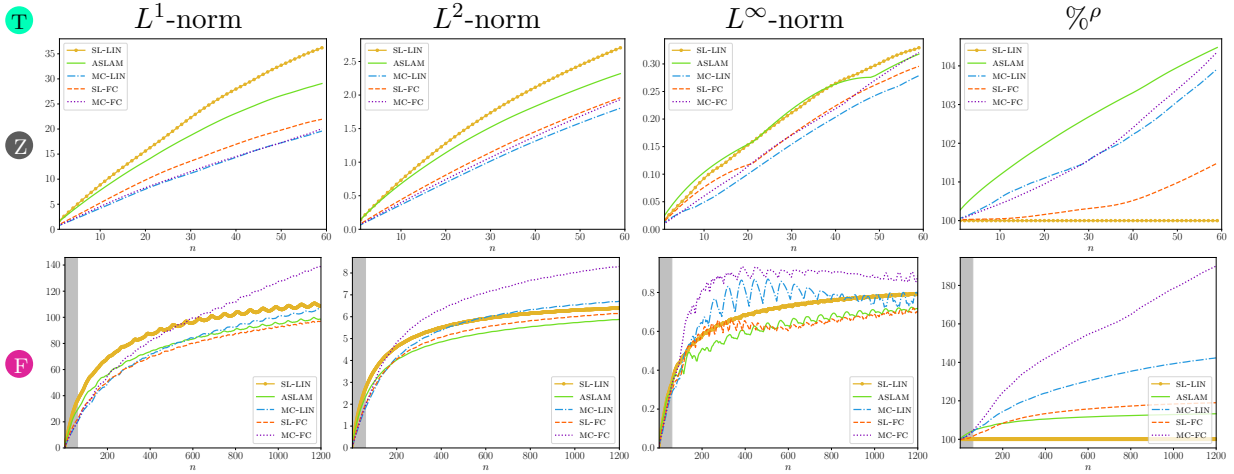


Figure 5.17: *Performance by metric for the 2d-TRANSLATION test.* Performance for Figure 5.16 measured by (from left to right) the L^1 -norm, the L^2 -norm, the L^∞ -norm, and density conservation $\% \rho$ for SL-LIN in solid yellow with circular markers, ASLAM in solid green, MC-LIN in dot-dashed blue, SL-FC in dashed orange, and MC-FC in dotted purple is shown for (from top to bottom) $n \in [1, 60]$ and $n \in [1, 1200]$. Note that the top row charts are zoomed-in versions of the area shaded in gray in the bottom row charts. For norm-based metrics, MacCormack advection (i.e., MC-LIN and MC-FC) appears best in the earliest time steps, but ASLAM and SL-FC perform better considering the full simulation. In terms of density conservation, no method surpasses SL-LIN. Finally, the trailing artefact previously identified for MC-LIN and MC-FC is only present in the conservation of density, where the final densities are respectively over 140% and 180% of the initial value.

In two dimensions, we first consider the 2d-TRANSLATION test. ASLAM further reduces dissipation compared to SL-LIN as evidenced by the annulus' better-preserved empty region seen in Figure 5.16. However, it is not superior to the same advection method with a higher-order accurate interpolation scheme (i.e., SL-FC). Under the 2d-TRANSLATION test, MC-LIN and MC-FC introduce some trailing artefact. More notably, with the higher-order accurate interpolation scheme, MC-FC aggravates the visual artefact. As a result, we can reasonably claim that ASLAM better safeguards the integrity of the annulus' shape although the MacCormack advection (i.e., MC-LIN and MC-FC) undeniably appears to preserve better the maximum extrema as evidenced by the darker colour of the corresponding annuli.

As shown in Figure 5.17, while ASLAM outperforms SL-LIN in the L^1 -norm and L^2 -norm, it is not the case in the L^∞ -norm if only the earliest time steps are considered. Moreover, ASLAM appears no better than the other methods (i.e., SL-FC, MC-LIN, and

MC-FC). However, if we look at the full simulation, we can see that ASLAM closely follows SL-FC, and may even be second best, even in terms of density conservation. Regarding the trailing artefact observed in Figure 5.16 for MC-LIN and MC-FC, it is interestingly not reflected in the norm-based metrics. Indeed, in the earliest time steps, MacCormack advection (i.e., MC-LIN and MC-FC) bests all other methods, except for SL-FC in the L^∞ -norm. In the full simulation, variations of the error may not be sufficient to explain the trailing artefact since other methods also exhibit comparable variations, but no artefact. Nonetheless, the worsening of the trailing artefact is apparent in the norm-based metrics, which indicate that, rather counter-intuitively but consistently with the observations, MC-LIN outperforms MC-FC. Yet, one could reasonably argue that the artefact is mirrored in the density conservation metric, where the performance of the MacCormack advection (i.e., MC-LIN and MC-FC) is below that of the semi-Lagrangian advection (i.e., SL-LIN and SL-FC) and ASLAM.

For the 2d-ROTATION test, results are shown in Figure 5.18. The MacCormack advection (i.e., MC-LIN and MC-FC) yields extremely good visual results, superior to ASLAM's unlike in the 2d-TRANSLATION test. Both the empty region as well as the maximum values look very similar to the analytical annulus, and no artefact is seen. Otherwise, ASLAM positions itself between SL-FC, which is less dissipative as seen from the annulus' darker colour, and SL-LIN, which is more dissipative as evidenced by the disappearing empty centre of the annulus.

In terms of norm-based metrics for the 2d-ROTATION test, MacCormack advection (i.e., MC-LIN and MC-FC) would be considered the best, as opposed to semi-Lagrangian advection (i.e., SL-LIN and SL-FC) and ASLAM, as shown in Figure 5.19. Except for the L^∞ -norm where MC-LIN results in a lower error, MC-FC performs better. ASLAM again outperforms SL-LIN in the L^∞ -norm and L^2 -norm, but not in the L^1 -norm. No method appears especially good at preserving density. In the full simulation, we observe an oscillation of the curves associated with SL-FC, MC-LIN, and MC-FC around $n = 600$, when the annulus is restored to its initial configuration. In light of this, SL-FC is the best at preserving density, bringing density conservation closest to 100% at $n = 600, 1200$. We note that MacCormack advection (i.e., MC-LIN and MC-FC) does increase the total density significantly as in the 2d-TRANSLATION test. Hence, the increase in density described in Figure 5.17 cannot be explained solely by the presence of the artefact sighted in Figure 5.16.

To conclude this section, it is clear that ASLAM almost always outperforms SL-LIN, except perhaps in terms of density conservation. Otherwise, it is both visually and quantitatively superior. Given the restrictions regarding monotonicity when using higher-order accurate interpolation schemes, which have yet to be thoroughly investigated, ASLAM can be an interesting alternative to using a higher-order accurate interpolation with semi-

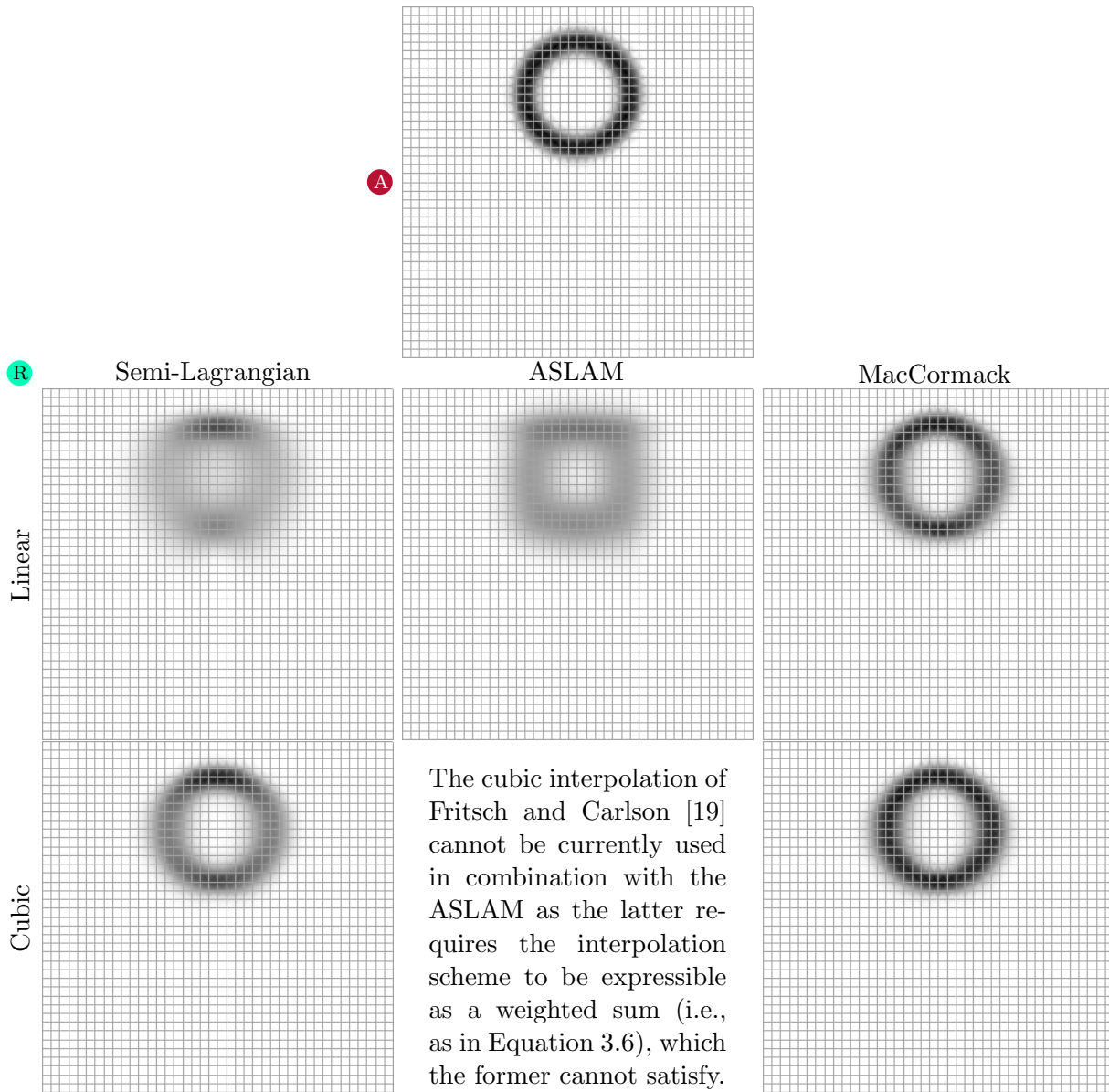


Figure 5.18: *Impact of advection method for the 2d-ROTATION test.* At $n = 1200$, the analytical solution (top) is compared to numerical solutions computed using (from left to right and from top to bottom) SL-LIN, ASLAM, MC-LIN, SL-FC, and MC-FC. The MacCormack advection (i.e., MC-LIN and MC-FC) is superior to all other methods in returning the annulus to its original shape and position. ASLAM is still less dissipative than SL-LIN, but, in this regard, it is also outdone by SL-FC.

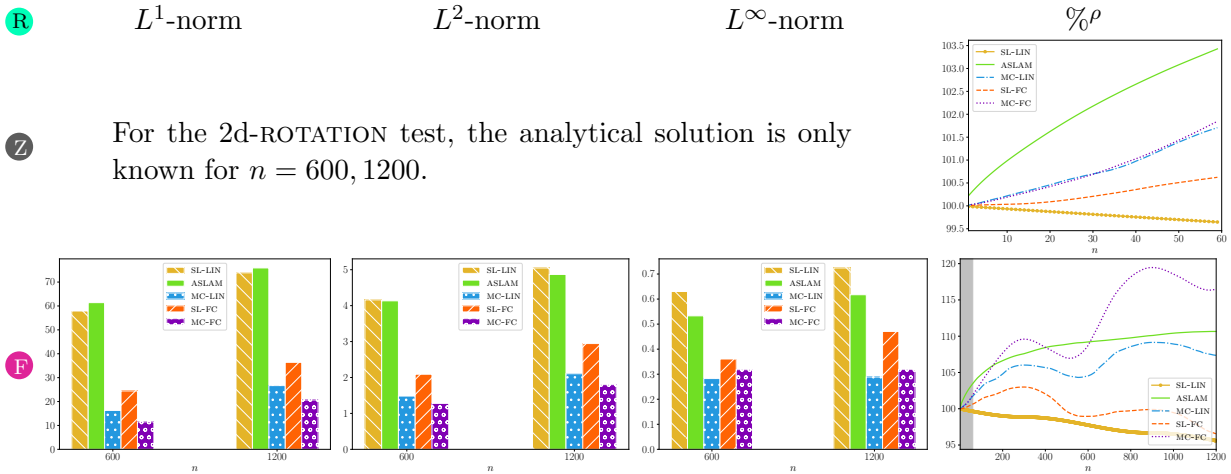


Figure 5.19: *Performance by metric for the 2d-ROTATION test.* Performance for Figure 5.18 is measured by (from left to right) the L^1 -norm, the L^2 -norm, the L^∞ -norm, and density conservation $\% \rho$ for SL-LIN in solid yellow with circular markers, ASLAM in solid green, MC-LIN in dot-dashed blue, SL-FC in dashed orange, and MC-FC in dotted purple. For the conservation of density, note that the top row chart is a zoomed-in version of the area shaded in gray in the corresponding bottom row chart. In terms of norms, MacCormack advection (i.e., MC-LIN and MC-FC) is better than semi-Lagrangian advection (i.e., SL-LIN and SL-FC) and ASLAM. However, in terms of density conservation, SL-FC is best, almost preserving the total initial density at $n = 600$, when the annulus is brought back to its original setting. Besides, ASLAM is the only method not to show oscillations in density conservation, steadily increasing instead.

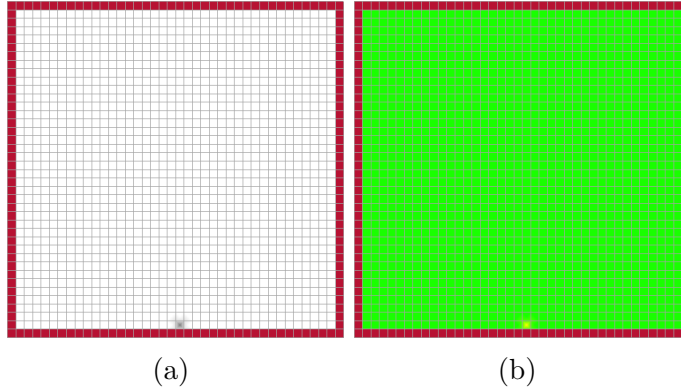


Figure 5.20: *Initial conditions for a rising plume scenario.* (a) shows density ρ on a white-to-black coloured scale, mapping $\rho \in [0, 1]$, while (b) shows temperature T on a blue-to-red coloured scale, mapping $T \in [-100, 100]$. The emitter has density $\rho = 0.5$ and temperature $T = 50$; the background temperature is $T = 5$. Solid boundary conditions on all four walls of the domain are depicted in red. Velocity is taken to be zero initially on all the simulation domain.

Lagrangian advection (i.e., SL-FC), although it does not produce as high-quality results. On the other hand, the MacCormack advection, regardless of the interpolation scheme with which it is coupled (i.e., MC-LIN or MC-FC), is better at preserving extrema than all the other methods considered. Since MacCormack advection’s performance has been shown to be quite dependent on the flow, we are reserving for later a more detailed judgement on whether it is superior or not to our method.

5.4 Practical Application Scenario

We compare the proposed ASLAM to different advection methods (i.e., the same as the ones presented in Section 5.3, with the same abbreviations too) in a rising plume scenario. A source is placed at the bottom of a closed box, and emits warm smoke for half the simulation time (i.e., for 600 time steps). The initial conditions are illustrated in Figure 5.20. Note that, given that we have an even number of grid cells (i.e., 40 per dimension), the setting is not entirely symmetric. Both scalar quantities (i.e., density and temperature) are stored on cell-centred grids while the velocity, the only vector quantity, is stored on a staggered grid. In this application scenario, the smoke’s density as well as the fluid’s temperature and velocity all are advected quantities.

The simulation proceeds as outlined in Algorithm 2.1. External forces include gravity and buoyancy as approximated in Fedkiw et al. [14]. The buoyancy model of Fedkiw et al. [14] accounts for the effects due to temperature (e.g., hot air rises while cold air sinks) and density (e.g., heavy smoke falls) without interfering with the pressure component of the simulation. Therefore, the smoke remains incompressible.

We focus our analysis on the first half of the simulation, when the smoke source emits, but the second half of the simulation is presented in Figure B.7. Resulting density fields for the first half of the simulation are presented in Figure 5.21.

At the beginning of the simulation, the different methods yield smoke plumes resembling one another, but the visual discrepancy increases as time progresses, making it harder to compare the results. However, it is evident that, identically to the testing scenarios discussed previously, ASLAM is less dissipative than SL-LIN. The resulting plume is less fuzzy, and shows more details especially as the box fills with smoke. In this regard, we can safely claim that the MacCormack advection, regardless of the interpolation scheme used (i.e., MC-LIN or MC-FC), is less dissipative than both semi-Lagrangian advection (i.e., SL-LIN and SL-FC) and ASLAM. It better preserves extrema as evidenced by the darker-coloured bow-shaped part of the plume. Additionally, the corresponding plumes (i.e., MC-LIN or MC-FC) take longer to reach the top of the box, and show more fine details. Yet, for MC-FC, the resulting plume does not look quite natural. In the later frames (i.e., bottom rows), we can see a rather unsmooth pattern which degenerates moving onwards (not shown in Figure 5.21, but in Figure B.7). ASLAM is comparable to SL-FC. They exhibit comparable levels of detail, but one might argue that ASLAM is slightly blurrier although it is not significant.

The rising plume scenario results are consistent with the analysis of the results obtained in the 1d-TRANSLATION test, 2d-TRANSLATION test, and 2d-ROTATION test. ASLAM surpasses SL-LIN, and is in close competition with SL-FC. However, in our opinion, the MC-LIN produces the less dissipative and more detailed results. We notably prefer it to MC-FC, which amplifies and/or generates potential unnatural-looking artefacts.

5.5 Additional Investigations

In Chapter 4, we mentioned that the ASLAM only uses a first-order Taylor series approximation in the update (refer to Equation 4.1 for a reminder). We briefly discussed including more terms in a PolyPIC fashion [20], but one could wonder if simply using stencil particles without the first derivative term (i.e., involving the descriptor \mathbf{c} and, indirectly, the gra-

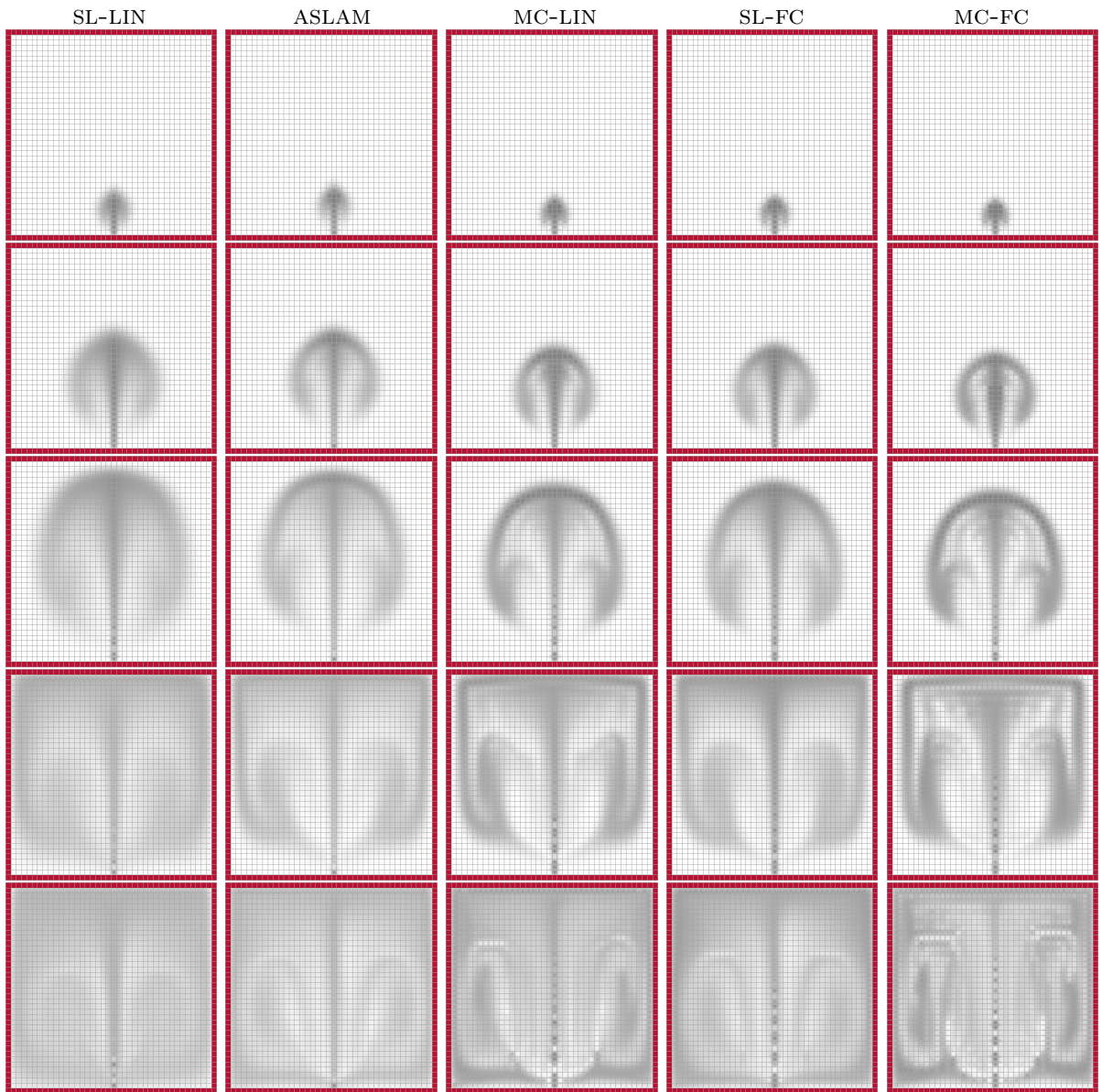


Figure 5.21: *Impact of advection method on a rising plume scenario.* Frames at (from top to bottom) $n = 120, 240, 360, 480, 600$ from a rising smoke plume in a closed box (in solid red) are simulated using (from left to right) SL-LIN, ASLAM, MC-LIN, SL-FC, and MC-FC. MC-LIN and MC-FC show more details earlier on than the other methods, but MC-FC develops some visual artefacts. In particular, SL-LIN is the most dissipative while ASLAM grows to be comparable to SL-FC. Subsequent frames are shown in Figure B.7.

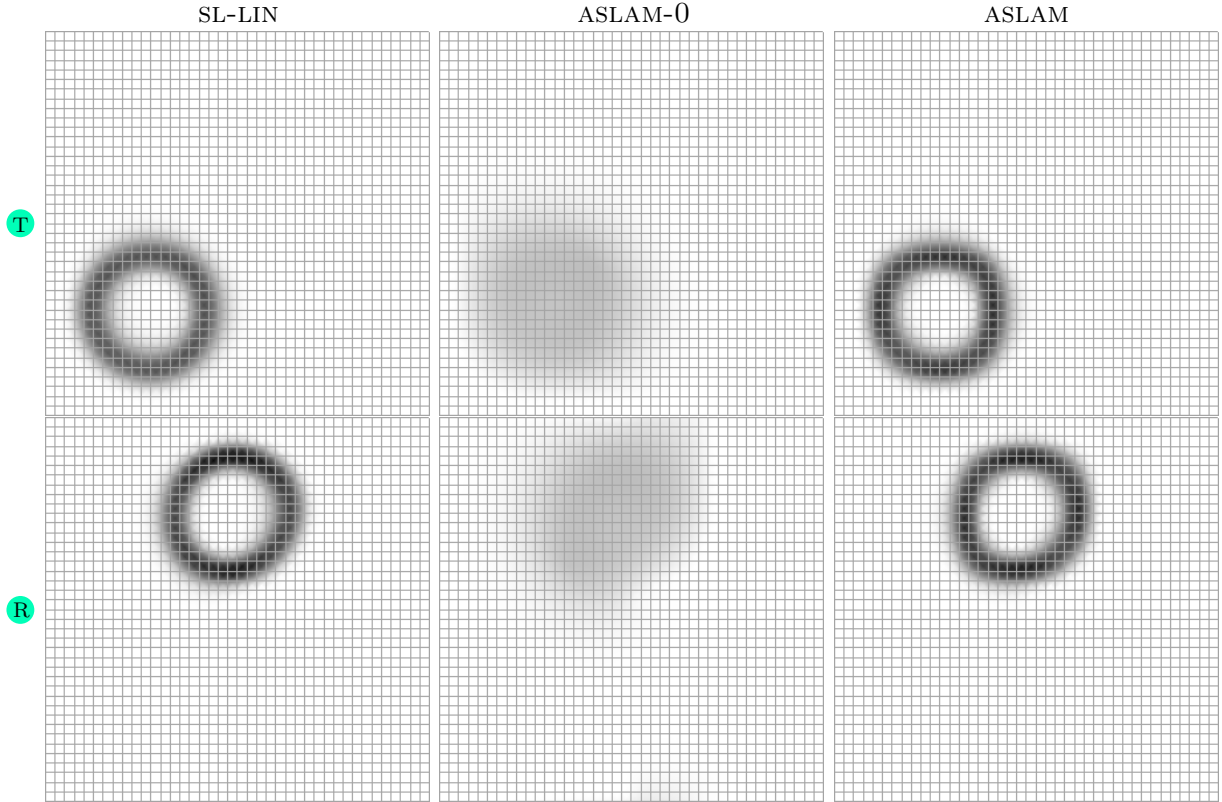


Figure 5.22: *Comparison between SL-LIN, ASLAM-0, and ASLAM.* The numerical solutions at $n = 60$ to (from top to bottom) the 2d-TRANSLATION test and the 2d-ROTATION test are computed using (from left to right) SL-LIN, ASLAM-0, and ASLAM. For both tests considered, ASLAM-0 is clearly overly dissipative, even SL-LIN is best.

dients of the weights) would also yield some improvement over standard semi-Lagrangian advection. In such a case, the update would be

$$q^{n+1}|_{\mathbf{x}_i} = \frac{1}{\sum_{\mathbf{x}_s \in \mathcal{S}} w_{p \rightarrow i}|_{\mathbf{x}_s}} \sum_{\mathbf{x}_s \in \mathcal{S}} \left(q^n|_{\mathbf{x}_s^b} \right) w_{p \rightarrow i}|_{\mathbf{x}_s}. \quad (5.8)$$

Equation 5.8 amounts to a super-sampled form of semi-Lagrangian advection, which we nickname ASLAM-0. We test this hypothesis with a CHESSBOARD stencil of parametric size $\zeta = 2$ with SPH weights, and the results are presented in Figure 5.22 for $n = 60$.

It is obvious that not including the first derivative term yields more dissipative results

for both the 2d-TRANSLATION test and the 2d-ROTATION test. This can be explained by the fact that Equation 5.8 amounts to a simple weighted average. We know that additional averaging operations lead to increased smoothing, which is what we are observing in Figure 5.22 after which only 60 time steps are completed. Of course, from a Taylor series approximation perspective, Equation 5.8 is a less accurate approximation than Equation 4.1, which directly translates into poorer and overly dissipative results. Furthermore, it is clear that simply adding stencil particles is not equivalent to increasing grid resolution.

In another short experiment, we test if the ASLAM could replace standard semi-Lagrangian advection in the MacCormack advection algorithm (refer to Algorithm 3.3 for a reminder). Since the ASLAM is also unconditionally stable due to clamping, the corresponding MacCormack advection, which we refer to as MC-ASLAM, would also be unconditionally stable. We use a CHESSBOARD stencil of parametric size $\zeta = 2$ and SPH weights as parameters for the ASLAM as a stepping scheme. Implementation-wise, combining the ASLAM with MacCormack advection requires some modifications.

Recall that we clamp the result of the update produced by Equation 4.1 to prevent under/overshooting with respect to the enclosing grid nodes \mathbf{x}_e^b (refer to Figure 4.1 for a reminder). This clamping clearly modifies the advection error of the ASLAM. Since the MacCormack scheme uses said error, we remove the ASLAM’s clamping to produce a better estimate of the advection error incurred by the ASLAM. However, we keep the clamping applied after the MacCormack advection, whose purpose is also to prevent similar under/overshooting.

Bidimensional results, shown in Figure 5.23, are extremely puzzling, hinting at the fact that combining MacCormack advection with the ASLAM as a stepping scheme may not be as trivial as it sounds. We can see that MC-ASLAM results in some serious problems. After a certain number of time steps $n < 60$, the annulus stops moving, appearing unaffected by the flow in both the 2d-TRANSLATION test and the 2d-ROTATION test. Upon investigation of this phenomenon, we note that MC-ASLAM ends up requiring clamping on all grid nodes. This drives the simulation into a stationary state where the post-advection grid is identical to the pre-advection grid. This state leads the scheme into an infinite loop with no visible effect, regardless of the flow.

Since clamping is only used when the advection error introduces under/overshooting, we decided to bring back the clamping in the ASLAM step in a new MC-ASLAM-CLAMPED combination, although we know that it tampers the “real” advection error. The results are shown in Figure 5.24. Obviously, this is not a good idea as evidenced by the very disturbing artefacts, which initially start as a mere “squarification”, and quickly overtake the simula-

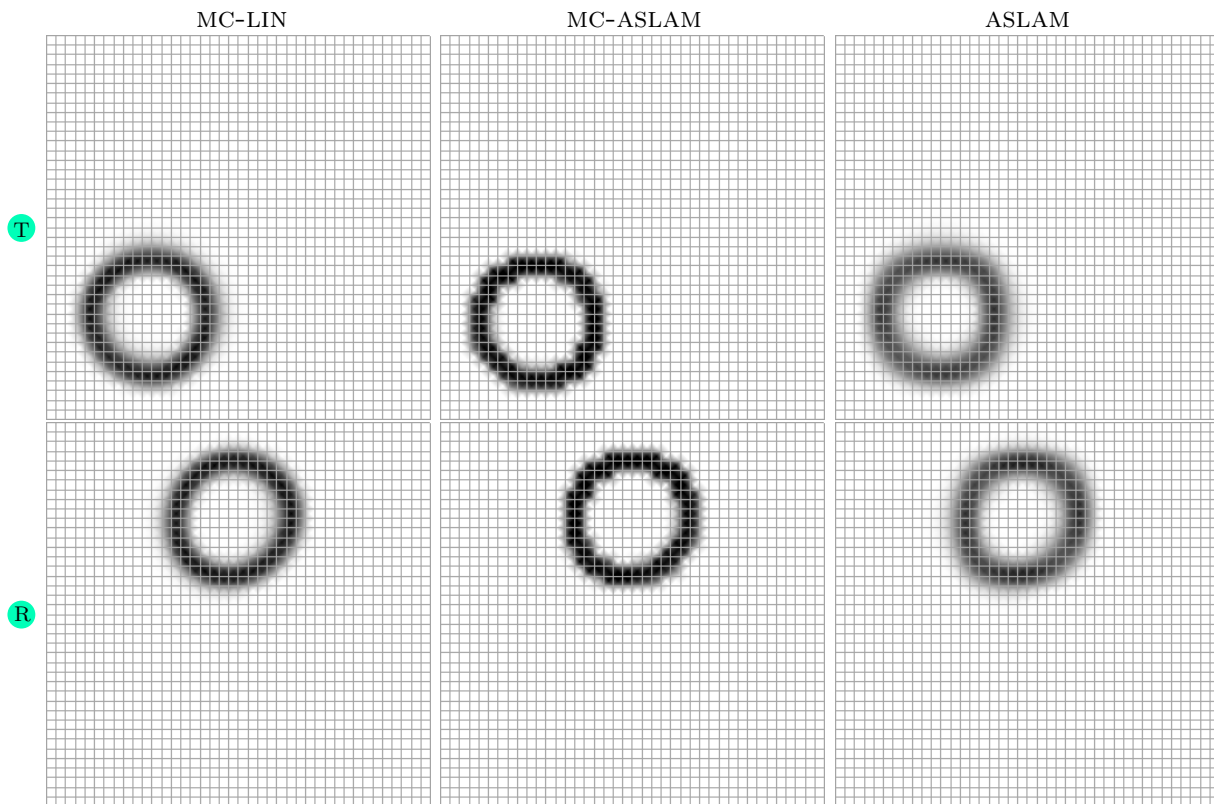


Figure 5.23: *Comparison between MC-LIN, MC-ASLAM, and ASLAM.* The numerical solutions at $n = 60$ to (from top to bottom) the 2d-TRANSLATION test and the 2d-ROTATION test are computed using (from left to right) MC-LIN, MC-ASLAM, and ASLAM. After a few time steps, the MC-ASLAM simulation gets trapped in a stationary state, where clamping from the MacCormack scheme annihilates the effect of advection, preventing the annulus from moving.

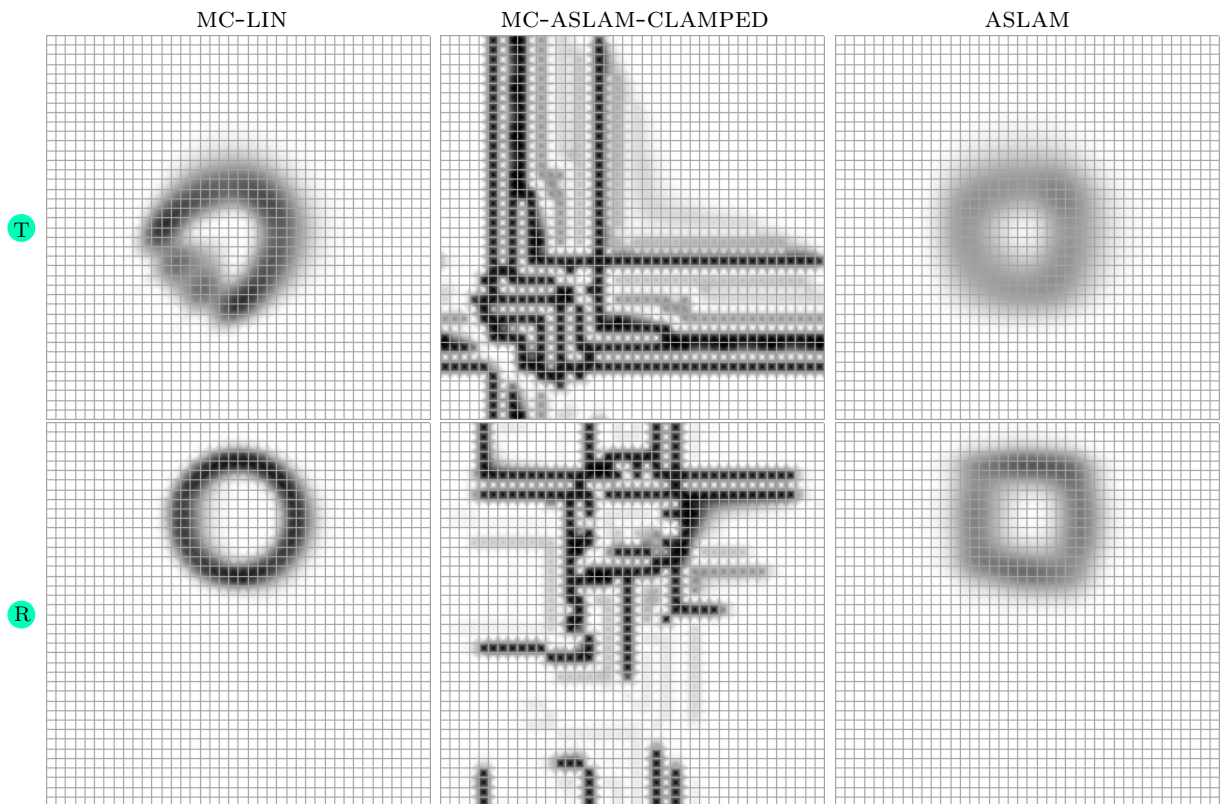


Figure 5.24: *Comparison between MC-LIN, MC-ASLAM-CLAMPED, and ASLAM.* The numerical solutions at $n = 600$ to (from top to bottom) the 2d-TRANSLATION test and the 2d-ROTATION test are computed using (from left to right) MC-LIN, MC-ASLAM-CLAMPED, and ASLAM. In MC-ASLAM-CLAMPED, restoring clamping in the ASLAM as a stepping scheme clearly produces some disturbing visual artefacts aggravated with time and, in this case, by periodic boundaries.

tion. In Figure 5.24, MC-ASLAM-CLAMPED's unnatural results are further exacerbated by the periodicity of the boundaries.

From these simple experiments, it is evident that combining the ASLAM with the MacCormack advection is not trivial, and, if pursued, will require a more detailed analysis of both methods. Among others, we believe that the stencil particles, due to their fixed starting locations, may break the correlation between the error incurred during the forward advection and the backward advection of the MacCormack scheme. This potentially makes the ASLAM incompatible with predictor-corrector algorithms as described in Section 3.2. Moreover, because the ASLAM is more computationally expensive than semi-Lagrangian advection, regardless of the interpolation scheme used, the improvement produced by an ASLAM-MacCormack union would have to be impressive to justify the choice of the ASLAM as a stepping scheme instead of semi-Lagrangian advection.

Chapter 6

Conclusion

In this thesis, we presented an **affine semi-Lagrangian advection method** (ASLAM), a semi-Lagrangian particle-free advection scheme based on an adaptation of APIC to the Eulerian framework. As illustrated for several testing scenarios and a practical application scenario, the ASLAM is less dissipative than standard semi-Lagrangian advection with linear interpolation, which was introduced in computer graphics by Stam [35]. Nevertheless, it is simultaneously poorer at successfully enforcing conservation of the advected scalar quantities. In comparison to the MacCormack advection and to semi-Lagrangian advection with the use of higher-order interpolation schemes, our ASLAM fails to reduce numerical dissipation as much. Yet, qualitative results on the rising plume scenario show that it is competitive with semi-Lagrangian advection with the monotone cubic interpolation of Fritsch and Carlson [19]. However, it is equivalently good or better than those schemes at conserving advected scalar quantities.

Since it substitutes fixed sample points for the set of traceable particles in hybrid and Lagrangian approaches, our ASLAM is particularly suited for the simulation of gases, where no interface or surface needs to be reconstructed. Despite these supplementary sample points, which are treated semi-Lagrangianly, the ASLAM offers a better alternative than super-sampled advection, which we demonstrated to be more dissipative. As a matter of fact, the APIC-based update of Equation 4.1 using stencil \mathcal{S} is the sole feature distinguishing the ASLAM from standard semi-Lagrangian advection. The ASLAM is otherwise composed of repeated semi-Lagrangian advectons on samples $\mathbf{x}_{\mathcal{S}}$. As such, better conservation of advected quantities could be possibly achieved by using a fully conservative semi-Lagrangian advection scheme, such as the one of Lentine et al. [30].

While it is possible to configure our proposed ASLAM with different parameters, our

investigation outlined an “optimal” parameter combination, which accounts for both qualitative and quantitative performances. Moreover, for the chosen parameter combination, we have shown our method’s stability as it reaches linear convergence. In light of the discussion regarding said parameters, guidelines can be derived to further develop the ASLAM’s parameter space. For instance, we are aware of the limitations related to increasing the parametric size ζ , but this was closely tied to the stencil particle distribution analysed.

Although we briefly hinted, in Section 4.1, at several possible alternative stencil particle distributions, this topic remains largely unexplored. Among others, we believe that a less regular distribution than the CHESSBOARD type, like a uniformly distributed noise, may help diminishing/eradicating the “squarification” introduced by the ASLAM. Since the stencil particles are fixed, replacing the stencil particle distribution does not affect the complexity of the algorithm, but may boost (or equally likely tank) the accuracy of the ASLAM.

6.1 Future Work

An evident extension to our work would be to use the ASLAM in a tridimensional scenario. While we expect the stencil \mathcal{S} to naturally extend to an additional spatial dimension, as it did from the unidimensional tests to the bidimensional tests, a parameter combination other than the “optimal” one identified in this thesis may be best. Should the ASLAM be used in tridimensional applications, this would have to be investigated. Moreover, our worked focused specifically on gases, but our method can presumably substitute for advection in a number of scenarios, including the simulation of liquids.

In presenting several interpolation schemes, we have informally tackled the idea of monotonicity. It remains to be determined, in the context of fluid simulations for graphics applications, whether clamping a non-monotonic interpolation scheme is preferable or not to using a scheme with a built-in monotonicity guarantee. Clamping inevitably introduces discontinuities in the derivatives within the clamped interpolated interval, but innately-monotone schemes may also lead to derivative discontinuities at the boundaries between grid cells.

Regardless, it is evident that defining a higher-order accurate monotone interpolation scheme is not trivial. While such schemes exist as evidenced by [18, 19] in one dimension and [5, 6] (with additional restrictions on the data) in two dimensions, an important issue arising is whether a dimension-specific interpolation scheme should be preferred to a unidimensional scheme applied along each dimension. This question is particularly relevant

as the number of dimensions is increased to three. Indeed, given how non-trivial producing a monotone bicubic patch is, one can reasonably expect the difficulty to increase for a monotone tricubic patch, what is more without constraints on the data.

The ASLAM results shown in Chapter 5 were all produced using linear interpolation to compute weights $w_{i \rightarrow p}$ from grid to fictitious particles. We can assume that higher-order interpolation schemes could be used to further decrease numerical dissipation, as long as the weights satisfy the conditions outlined by the original APIC method (i.e., Equation 2.25). This, however, remains to be investigated, especially in light of the challenges inherent to using higher-order interpolation schemes. Hence, finding a unidimensional monotone cubic interpolation scheme which does not exhibit the dimension ordering issue remains relevant. However, the problem of ordering-dependency could be discarded if one can prove that the ordering of the dimensions does not matter, or that it is of little significance.

Furthermore, our proposed ASLAM requires an interpolation scheme which can be expressed as a weighted sum as in Equation 3.6. Thus, the choices of interpolation schemes are severely limited. Currently, appropriate schemes need to be monotone (preferably without clamping), be independent of dimension ordering or tailored to a given number of dimensions, and fit the form given by Equation 3.6. Linear interpolation meets all aforementioned criteria, but none of the other schemes presented in this thesis do.

Alternatively, we could investigate a way to perform the ASLAM weightlessly or, at the very least, redefine the locally affine velocity descriptor \mathbf{c} . Indeed, if said descriptor \mathbf{c} were defined without the gradient of the interpolation weights, the constraints on the interpolation scheme would be lessened, especially regarding having to be expressed as a weighted sum. As aforementioned, PolyPIC by Fu et al. [20] also offers an interesting avenue to further diminish artificial dissipation, and to refine the descriptor \mathbf{c} .

Another interesting research avenue would be to leverage, via hybridization in a narrow-band FLIP [15] fashion, the compatible nature of the ASLAM and “true” (i.e., Lagrangian) APIC methods, which are already hybrids. In this yet hypothetical setup, the Lagrangian APIC is used near the liquid’s surface whilst the Eulerian ASLAM is applied on the interior of the simulation domain. Given the conceptual closeness of both techniques, the resulting hybrid method seems likely to simplify the transitions between Lagrangian and Eulerian regions of the domain as the surface moves.

References

- [1] C. Donald Ahrens. *Meteorology Today: An Introduction to Weather, Climate, and the Environment*. Brooks/Cole, CengageLearning, Belmont, CA, 9th ed. edition, 2009.
- [2] Hiroshi Akima. A new method of interpolation and smooth curve fitting based on local procedures. *Journal of the ACM (JACM)*, 17(4):589–602, 1970.
- [3] Jeremiah U. Brackbill and Hans M. Ruppel. Flip: A method for adaptively zoned, particle-in-cell calculations of fluid flows in two dimensions. *Journal of Computational Physics*, 65(2):314–343, 1986.
- [4] Robert Bridson. *Fluid Simulation for Computer Graphics*. A K Peters, Wellesley, Mass., 2008.
- [5] Ralph E. Carlson and Frederick N. Fritsch. Monotone piecewise bicubic interpolation. *SIAM Journal on Numerical Analysis*, 22(2):386–400, 1985.
- [6] Ralph E. Carlson and Frederick N. Fritsch. An algorithm for monotone piecewise bicubic interpolation. *SIAM Journal on Numerical Analysis*, 26(1):230–238, 1989.
- [7] Edwin Catmull and Raphael Rom. A class of local interpolating splines. In Robert E. Barnhill and Richard F. Riesenfeld, editors, *Computer Aided Geometric Design*, pages 317 – 326. Academic Press, 1974.
- [8] Qiaodong Cui, Pradeep Sen, and Theodore Kim. Scalable laplacian eigenfluids. *ACM Trans. Graph.*, 37(4):87:1–87:12, July 2018.
- [9] Carl de Boor. *A Practical Guide to Splines*. Applied mathematical sciences; 27. Springer-Verlag, New York, 1978.
- [10] Todd F. Dupont and Yingjie Liu. Back and forth error compensation and correction methods for removing errors induced by uneven gradients of the level set function. *Journal of Computational Physics*, 190(1):311–324, 2003.

- [11] Todd F. Dupont and Yingjie Liu. Back and forth error compensation and correction methods for semi-lagrangian schemes with application to level set interface computations. *Mathematics of Computation*, 76(258):647–668, 2007.
- [12] Essex Edwards and Robert Bridson. A high-order accurate particle-in-cell method. *International Journal for Numerical Methods in Engineering*, 90(9):1073–1088, 2012.
- [13] Douglas Enright, Ronald Fedkiw, Joel Ferziger, and Ian Mitchell. A hybrid particle level set method for improved interface capturing. *Journal of Computational Physics*, 183(1):83–116, 2002.
- [14] Ronald Fedkiw, Jos Stam, and Henrik Wann Jensen. Visual simulation of smoke. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, pages 15–22, New York, NY, USA, 2001. ACM.
- [15] Florian Ferstl, Ryoichi Ando, Chris Wojtan, Rüdiger Westermann, and Nils Thuerey. Narrow band flip for liquid simulations. *Computer Graphics Forum*, 35(2):225–232, 2016.
- [16] Nick Foster and Dimitri Metaxas. Realistic animation of liquids. *Graphical Models and Image Processing*, 58(5):471–483, 1996.
- [17] Alain Fournier and William T. Reeves. A simple model of ocean waves. *SIGGRAPH Comput. Graph.*, 20(4):75–84, August 1986.
- [18] Frederick N. Fritsch and J. Butland. A method for constructing local monotone piecewise cubic interpolants. *SIAM Journal on Scientific and Statistical Computing*, 5(2):300–304, 1984.
- [19] Frederick N. Fritsch and Ralph E. Carlson. Monotone piecewise cubic interpolation. *SIAM Journal on Numerical Analysis*, 17(2):238–246, 1980.
- [20] Chuyuan Fu, Qi Guo, Theodore Gast, Chenfanfu Jiang, and Joseph Teran. A polynomial particle-in-cell method. *ACM Trans. Graph.*, 36(6):222:1–222:12, November 2017.
- [21] Manuel Noronha Gamito, Pedro Faria Lopes, and Mário Rui Gomes. Two-dimensional simulation of gaseous phenomena using vortex particles. In Demetri Terzopoulos and Daniel Thalmann, editors, *Computer Animation and Simulation '95*, pages 3–15, Vienna, 1995. Springer Vienna.

- [22] Rafael C. Gonzalez. *Digital Image Processing*. Pearson, New York, NY, 2018.
- [23] Francis H. Harlow. The particle-in-cell method for numerical solution of problems in fluid dynamics. 3 1962.
- [24] Yuanming Hu, Yu Fang, Ziheng Ge, Ziyin Qu, Yixin Zhu, Andre Pradhana, and Chenfanfu Jiang. A moving least squares material point method with displacement discontinuity and two-way rigid body coupling. *ACM Trans. Graph.*, 37(4):150:1–150:14, July 2018.
- [25] Insung Ihm, Deukhyun Cha, and Byungkwon Kang. Controllable local monotonic cubic interpolation in fluid animations. *Computer Animation and Virtual Worlds*, 16(34):365–375, 2005.
- [26] Stefan Jeschke and Chris Wojtan. Water wave packets. *ACM Trans. Graph.*, 36(4):103:1–103:12, July 2017.
- [27] Chenfanfu Jiang, Craig Schroeder, Andrew Selle, Joseph Teran, and Alexey Stomakhin. The affine particle-in-cell method. *ACM Trans. Graph.*, 34(4):51:1–51:10, July 2015.
- [28] Michael Kass and Gavin Miller. Rapid, stable fluid dynamics for computer graphics. *SIGGRAPH Comput. Graph.*, 24(4):49–57, September 1990.
- [29] Doyub Kim. *Fluid Engine Development*. Taylor & Francis: Informa plc, Boca Raton, Fla., 2017.
- [30] Michael Lentine, Jón Tómas Grétarsson, and Ronald Fedkiw. An unconditionally stable fully conservative semi-lagrangian method. *Journal of Computational Physics*, 230(8):2857–2879, April 2011.
- [31] Louis M. Milne-Thomson. *The Calculus of Finite Differences*. AMS Chelsea Publishing Series. AMS Chelsea Pub., 2000.
- [32] Matthias Müller, David Charypar, and Markus Gross. Particle-based fluid simulation for interactive applications. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '03, pages 154–159, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.
- [33] Glenn Research Center NASA. Phases of matter. <https://www.grc.nasa.gov/www/k-12/airplane/state.html>.

- [34] Andrew Selle, Ronald Fedkiw, Byungmoon Kim, Yingjie Liu, and Jarek Rossignac. An unconditionally stable maccormack method. *Journal of Scientific Computing*, 35 (2-3):350–371, June 2008.
- [35] Jos Stam. Stable fluids. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '99, pages 121–128, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.
- [36] Jos Stam and Eugene Fiume. Turbulent wind fields for gaseous phenomena. In *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '93, pages 369–376, New York, NY, USA, 1993. ACM.
- [37] Jos Stam and Eugene Fiume. Depicting fire and other gaseous phenomena using diffusion processes. In *Proceedings of the 22Nd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '95, pages 129–136, New York, NY, USA, 1995. ACM.
- [38] Andrew Staniforth and Jean Côté. Semi-lagrangian integration schemes for atmospheric models – a review. *Monthly Weather Review*, 119(9):2206–2223, 1991.
- [39] Michael Steffen, Robert M. Kirby, and Martin Berzins. Analysis and reduction of quadrature errors in the material point method (mpm). *International Journal for Numerical Methods in Engineering*, 76(6):922–948, 2008.
- [40] Alexey Stomakhin, Craig Schroeder, Lawrence Chai, Joseph Teran, and Andrew Selle. A material point method for snow simulation. *ACM Trans. Graph.*, 32(4):102:1–102:10, July 2013.
- [41] Geoffrey K. Vallis. *Atmospheric and Oceanic Fluid Dynamics: Fundamentals and Large-Scale Circulation*. Cambridge University Press, Cambridge, UK; New York, 2006.
- [42] John F. Wendt, editor. *Computational Fluid Dynamics: An Introduction*. A von Karman Institute Book. Springer, Berlin, 3rd ed. edition, 2009.
- [43] Jonas Zehnder, Rahul Narain, and Bernhard Thomaszewski. An advection-reflection solver for detail-preserving fluid simulation. *ACM Trans. Graph.*, 37(4):85:1–85:8, July 2018.
- [44] Yongning Zhu and Robert Bridson. Animating sand as a fluid. In *ACM SIGGRAPH 2005 Papers*, SIGGRAPH '05, pages 965–972, New York, NY, USA, 2005. ACM.

APPENDICES

Appendix A

Cubic Interpolation Schemes

In this appendix, we discuss several cubic interpolation schemes. Most formulations covered, except the cubic B-spline and bicubic interpolations, follow Equation 3.12, the general model, introduced in Chapter 3, for piecewise cubic polynomials. Equation 3.12 can be derived using divided differences as in de Boor [9]. The k -th divided difference of a function f at nodes x_i, \dots, x_{i+k} is defined to be the leading coefficient of the polynomial of order $k + 1$ agreeing with f at x_i, \dots, x_{i+k}

$$[x_i, \dots, x_{i+k}]f = \begin{cases} \frac{f^{(k)}(x_i)}{k!} & \text{if } x_i = \dots = x_{i+k} \\ & \text{and } f \in C^{(k)}, \\ \frac{[x_i, \dots, x_{r-1}, x_{r+1}, \dots, x_{i+k}] - [x_i, \dots, x_{s-1}, x_{s+1}, \dots, x_{i+k}]}{x_s - x_r} & \text{if } x_r \neq x_s. \end{cases} \quad (\text{A.1})$$

Often, x_r is taken to be x_i , and x_s is taken to be x_{i+k} yielding

$$[x_i, \dots, x_{i+k}]f = \begin{cases} \frac{f^{(k)}(x_i)}{k!} & \text{if } x_i = \dots = x_{i+k} \text{ and } f \in C^{(k)}, \\ \frac{[x_{i+1}, \dots, x_{i+k}] - [x_i, \dots, x_{i+k-1}]}{x_{i+k} - x_i} & \text{otherwise.} \end{cases} \quad (\text{A.2})$$

Milne-Thomson [31] presents a very useful generalized formula for the divided difference when nodes are repeated as in $\underbrace{x_1, \dots, x_1}_{\eta_1}, \dots, \underbrace{x_p, \dots, x_p}_{\eta_p}$

$$[x_1, \dots, x_1, \dots, x_p, \dots, x_p] = \frac{1}{(\eta_1 - 1)! \dots (\eta_p - 1)!} \frac{\partial^{(\eta_1 + \dots + \eta_p - p)}}{\partial x_1^{(\eta_1 - 1)} \dots \partial x_p^{(\eta_p - 1)}} [x_1, \dots, x_p] \quad (\text{A.3})$$

To derive Equation 3.12, we consider a single interval $[x_i, x_{i+1}]$ on which a cubic polynomial f satisfies

$$\begin{aligned} f|_{x_i} &= q|_{x_i}, \\ f|_{x_{i+1}} &= q|_{x_{i+1}}, \\ f'|_{x_i} &= \delta_i, \\ f'|_{x_{i+1}} &= \delta_{i+1}. \end{aligned} \tag{A.4}$$

We further define $\Delta x = x_{i+1} - x_i$. In de Boor [9], the Newton form is used to retrieve the coefficients of f

$$\begin{aligned} f|_x &= f|_{x_i} + (x - x_i) [x_i, x_i]f + (x - x_i)^2 [x_i, x_i, x_{i+1}]f \\ &\quad + (x - x_i)^2 (x - x_{i+1}) [x_i, x_i, x_{i+1}, x_{i+1}]f, \end{aligned} \tag{A.5}$$

where

$$[x_i, x_i]f = f'|_{x_i} = \delta_i, \tag{A.6a}$$

$$\begin{aligned} [x_i, x_i, x_{i+1}]f &= \frac{([x_i, x_{i+1}]f - [x_i, x_i]f)}{\Delta x} \\ &= \frac{1}{\Delta x} \left(\frac{[x_{i+1}]f - [x_i]f}{\Delta x} - \delta_i \right) \\ &= \frac{1}{\Delta x} \left(\frac{f|_{x_{i+1}} - f|_{x_i}}{\Delta x} - \delta_i \right) \end{aligned} \tag{A.6b}$$

$$\begin{aligned} [x_i, x_i, x_{i+1}, x_{i+1}]f &= \frac{([x_i, x_{i+1}, x_{i+1}]f - [x_i, x_i, x_{i+1}]f)}{\Delta x} \\ &= \frac{1}{(\Delta x)^2} ([x_{i+1}, x_{i+1}]f - [x_i, x_{i+1}]f - [x_i, x_{i+1}]f + [x_i, x_i]f) \\ &= \frac{1}{(\Delta x)^2} \left(f'|_{x_{i+1}} - \frac{2D_i}{\Delta x} + f'|_{x_i} \right) \\ &= \frac{1}{(\Delta x)^2} \left(\delta_{i+1} - \frac{2D_i}{\Delta x} + \delta_i \right). \end{aligned} \tag{A.6c}$$

Remember that $x_{i+1} = x_i + \Delta x$. By carefully substituting Equation A.6 into Equation A.5, and rearranging the terms to express f in terms of shifted powers, we get

$$\begin{aligned}
f|_x &= q|_{x_i} + (x - x_i) \delta_i + \frac{(x - x_i)^2}{\Delta x} \left(\frac{D_i}{\Delta x} - \delta_i \right) + \frac{(x - x_i)^2 (x - x_{i+1})}{(\Delta x)^2} \left(\delta_{i+1} - \frac{2D_i}{\Delta x} + \delta_i \right) \\
&= q|_{x_i} + (x - x_i) \delta_i + \frac{(x - x_i)^2}{\Delta x} \left(\frac{D_i}{\Delta x} - \delta_i + \frac{(x - x_{i+1})}{(\Delta x)} \left(\delta_{i+1} - \frac{2D_i}{\Delta x} + \delta_i \right) \right) \\
&= q|_{x_i} + (x - x_i) \delta_i + \frac{(x - x_i)^2}{\Delta x} \left(\frac{D_i}{\Delta x} - \delta_i + \frac{(x - x_i - \Delta x)}{(\Delta x)} \left(\delta_{i+1} - \frac{2D_i}{\Delta x} + \delta_i \right) \right) \\
&= q|_{x_i} + (x - x_i) \delta_i + \frac{(x - x_i)^2}{\Delta x} \left(\frac{3D_i}{\Delta x} - 2\delta_i - \delta_{i+1} + \frac{(x - x_i)}{(\Delta x)} \left(\delta_{i+1} - 2\frac{D_i}{\Delta x} + \delta_i \right) \right) \\
&= q|_{x_i} + (x - x_i) \delta_i + \frac{(x - x_i)^2}{\Delta x} \left(\frac{3D_i}{\Delta x} - 2\delta_i - \delta_{i+1} \right) + \frac{(x - x_i)^3}{(\Delta x)^2} \left(\delta_{i+1} - 2\frac{D_i}{\Delta x} + \delta_i \right) \\
&= q|_{x_i} + \bar{x} (\delta_i \Delta x) + \bar{x}^2 (3D_i - 2\delta_i \Delta x - \delta_{i+1} \Delta x) + \bar{x}^3 (-2D_i + \delta_i \Delta x + \delta_{i+1} \Delta x),
\end{aligned} \tag{A.7}$$

which is exactly Equation 3.12. If D_i and δ_i are defined by Equation 3.13 and Equation 3.14 respectively, then the general form for piecewise cubic polynomial interpolation can be rewritten in terms of weights, as in Bridson [4], to fit the form given by Equation 3.6:

$$\begin{aligned}
w_{i-1} &= -\frac{\bar{x}}{2} (\bar{x} - 1)^2, \\
w_i &= 1 + \frac{1}{2} \bar{x}^2 (3\bar{x} - 5), \\
w_{i+1} &= -\frac{\bar{x}}{2} (3\bar{x}^2 - 4\bar{x} - 1), \\
w_{i+2} &= \frac{\bar{x}^2}{2} (\bar{x} - 1).
\end{aligned} \tag{A.8}$$

Note that, depending on the choice of D_i and, most importantly, of δ_i and δ_{i+1} , it is not always trivial, or even possible, to express the interpolation scheme as a weighted sum.

A.1 Modified Catmull-Rom Interpolation

Although the monotonicity of the interpolation scheme proposed by Fedkiw et al. [14] has been disproved, the scheme can still be used along with appropriate clamping. The pseudo-

Case	w_{i-1}	w_i	w_{i+1}	w_{i+2}
C1	0	1	0	0
C2	0	$(\bar{x} - 1)^2 (2\bar{x} + 1)$	$\bar{x}^2 (3 - 2\bar{x})$	0
C3	0	$1 + \frac{\bar{x}^2}{2} (3\bar{x} - 5)$	$\bar{x}^2 (3 - 2\bar{x})$	$\frac{\bar{x}^2}{2} (\bar{x} - 1)$
C4	$-\frac{\bar{x}}{2} (\bar{x} - 1)^2$	$(\bar{x} - 1)^2 (2\bar{x} + 1)$	$\frac{\bar{x}}{2} (3\bar{x}^2 - 4\bar{x} - 1)$	0
C5	Weights can be taken as in Equation A.8			

Table A.1: *Weights for the modified Catmull-Rom interpolation.* The weights for the different cases of the modified Catmull-Rom interpolation of Fedkiw et al. [14] result from enforcing Equation 3.15. Clamping may be needed if Equation 3.18 is not satisfied.

code outlined by Kim [29] translates the necessary conditions, given by Equation 3.15, into the five following cases, with accompanying weights shown in Table A.1.

C1: $D_i = 0$

This is a constant interval since it implies that $x_i = x_{i+1}$; thus, we set $\delta_i = \delta_{i+1} = 0$.

C2: $D_i \neq 0$, $\text{sign}(D_i) \neq \text{sign}(\delta_i)$, and $\text{sign}(D_i) \neq \text{sign}(\delta_{i+1})$

Both x_i and x_{i+1} are extrema; thus, we set $\delta_i = \delta_{i+1} = 0$.

C3: $D_i \neq 0$ and $\text{sign}(D_i) \neq \text{sign}(\delta_i)$

x_i is an extremum; thus, we set $\delta_i = 0$.

C4: $D_i \neq 0$ and $\text{sign}(D_i) \neq \text{sign}(\delta_{i+1})$

x_{i+1} is an extremum; thus, we set $\delta_{i+1} = 0$.

C5: Otherwise.

Inherently to these five cases, the weights now depend on the values at all $\mathbf{x}_h \in \mathcal{H}$. Implementation-wise, this prevents the abstraction of grid's values when interpolating, but, since clamping is required to preserve monotonicity, the grid's values already need to be known.

A more critical repercussion of this modified version of Catmull-Rom interpolation is the loss of commutativity of the weights for each dimension. Indeed, the ordering of the dimensions along which interpolation is performed is now relevant, and impacts the numerical results of the overall scheme. For the example depicted in Figure 3.3b, the ordering of the dimensions is only irrelevant if $\bar{x} = \bar{y}$ as evidenced by the interpolation formulas obtained by substituting the appropriate weights in Equation 3.6:

$$\begin{aligned} q|_{\mathbf{x}} &= -\bar{y}^3 + \bar{y}^2 + \bar{y} + 1 \quad \text{along } x \text{ first, then along } y, \\ q|_{\mathbf{x}} &= -\bar{x}^3 + \bar{x}^2 + \bar{x} + 1 \quad \text{along } y \text{ first, then along } x. \end{aligned} \tag{A.9}$$

While it is unclear whether this loss of commutativity is visually significant or not, this dilemma only arises from modifying the Catmull-Rom interpolation scheme as in Fedkiw et al. [14].

To conclude this section, it would perhaps be worth spending more time investigating the relevance of the interpolation scheme proposed by Fedkiw et al. [14], given that it is not monotone unless clamping is used, and that we do not know the impact of dimension ordering on the interpolation.

A.2 Bridson's Cubic Interpolation

In the second edition of [4], Bridson presents a cubic interpolation scheme as a less dissipative alternative to the approach of Fedkiw et al. [14]. While the proposed scheme does not exhibit the dimension ordering oddness which arose from modifying Catmull-Rom interpolation, it remains non-monotonic (refer to Figure 3.5 for a reminder). Assuming $\Delta x = 1$, this scheme's weights are given by:

$$\begin{aligned} w_{i-1} &= -\frac{\bar{x}}{3} + \frac{\bar{x}^2}{2} - \frac{\bar{x}^3}{6}, \\ w_i &= 1 - \bar{x}^2 + \frac{\bar{x}^3 - \bar{x}}{2}, \\ w_{i+1} &= \bar{x} + \frac{\bar{x}^2 - \bar{x}^3}{2}, \\ w_{i+2} &= \frac{\bar{x}^3 - \bar{x}}{6}. \end{aligned} \tag{A.10}$$

Inspection of the weighting functions reveals that this scheme is not monotone. Indeed, like the interpolation scheme proposed by Fedkiw et al. [14], some weights are not non-negative.

Upon observation of this fact, Bridson [4] says:

[The interpolated signal] slightly goes below zero, although the initial data was all non-negative. At a theoretical level, with certain nonlinear equations (with additional terms beyond just advection), this raises a greater risk of instabilities developing – but in practice, for the fluid solvers this book discusses, it appears not to be an issue.

Hence, as also noted by Bridson, clamping is required to prevent under/overshooting. To our knowledge, no one has investigated whether clamping is theoretically better than using a monotone interpolation scheme. We know only that, in practice, clamping works. The real question here is perhaps how to determine the correctness of an interpolated result. Indeed, most often, we do not have the analytical solution to the transport problems of interest. Thus, we cannot construct the appropriate polynomial to guarantee a specific order of accuracy. In the context of smoke simulation, a vorticity-preserving interpolation scheme yielding swirlier motion would perhaps be most useful, even if it is not accurate.

A.3 Fritsch-Carlson Interpolation

The approach proposed by Fritsch and Carlson [19] is monotone by design. In this thesis, we mimic it in choosing the subregion of monotonicity \mathcal{M} (i.e., Equation 3.20) based on their assessment that it produces more “visually pleasing” results than the alternative choices they propose. However, we would like to point out that this conclusion may have been biased by how Fritsch and Carlson determine the modification to the derivative each region warrants.

When required, the derivatives’ update is performed by finding the intersection between the boundary of the monotonicity subregion and the line joining the non-monotonic (α_i, β_i) pair to the origin of the $\alpha\beta$ -space (refer to Figure 3.6 for a reminder). For the monotonicity subregion \mathcal{M} , described by Equation 3.20, the intersection corresponds to the closest point on the subregion’s boundary and, hence, to the smallest possible change of the derivatives’ values. This unfairly favours the origin-centred circular region compared to the other monotonicity subregions proposed in Fritsch and Carlson [19].

In our opinion, finding the closest point to the boundary of each monotonicity subregion considered would have been a more objective way to compare said subregions. Intuitively, we believe that minimizing the changes in derivatives would be preferable, not only because derivative updates directly translate into slightly more computation, but also because each

Node	Knot Vector(s)
Internal	$\{x_{i-2}, x_{i-1}, x_i, x_{i+1}, x_{i+2}\}$
Left boundary	$\{x_i, x_i, x_i, x_{i+1}, x_{i+2}\} + \{x_i, x_i, x_i, x_i, x_{i+1}\}$
Right boundary	$\{x_{i-2}, x_{i-1}, x_i, x_i, x_i\} + \{x_{i-1}, x_i, x_i, x_i, x_i\}$
Left external	$\{x_{i-1}, x_{i-1}, x_i, x_{i+1}, x_{i+2}\}$
Right external	$\{x_{i-2}, x_{i-1}, x_i, x_{i+1}, x_{i+1}\}$

Table A.2: *Knot vectors to generate the basis functions for the cubic B-spline interpolation.* To generate the different basis functions $B(\tilde{x})$, presented in Table A.3, different nodes use different knot vectors; some even use more than one. Left external nodes are one node away from the left boundary while right external nodes are one node away from the right boundary. Knot vectors are given by Steffen et al. [39].

interval should be handled as similarly as possible. Hence, the monotonicity subregion corresponding to the square $[0, 3] \times [0, 3]$ in $\alpha\beta$ -space may be best as the update would be simpler (i.e., restricting α_i and β_i to be at most 3). As mentioned in Fritsch and Carlson [19], it is also the largest subregion satisfying the properties required to ensure consistency between intervals. Moreover, unlike the nonlinear update described in Chapter 3, this update would be linear, most likely allowing the interpolation scheme to be expressed as in Equation 3.6 (i.e., a weighted sum). In [18], Fritsch and Butland propose a method to construct the derivatives such that $(\alpha_i, \beta_i) \in [0, 3] \times [0, 3]$. If the corresponding interpolation scheme can be written as a weighted sum, then we could have a cubic version of the ASLAM.

A.4 Cubic B-Spline Interpolation

In computer graphics, cubic B-spline interpolation has been used along with the material point method (MPM) as in Stomakhin et al. [40]. In particular, the basis functions $B(\tilde{x})$ from Steffen et al. [39] can be used as weights, such that cubic B-spline interpolation satisfies Equation 3.6. Moreover, in higher dimensions, the weights can be expressed as in Equation 3.7 using dyadic products. In one dimension, the basis functions are splines of degree 3 with compact support. Steffen et al. [39] provides the knot vectors, as presented in Table A.2, to derive the basis functions for different grid nodes.

For succinctness, we refer to nodes one node away from any boundary as external nodes, which allows us to distinguish them from both boundary nodes and internal nodes. Furthermore, it is convenient to use a normalized position $\tilde{x} \in [-2, 2]$, and to generalize

the notation for the knot vector. For instance, the knot vector for internal nodes can be expressed as

$$\begin{aligned} \{x_{i-2}, x_{i-1}, x_i, x_{i+1}, x_{i+2}\} &= \{\gamma_i, \gamma_{i+1}, \gamma_{i+2}, \gamma_{i+3}, \gamma_{i+4}\} \\ &= \{-2, -1, 0, 1, 2\}. \end{aligned} \quad (\text{A.11})$$

With this new notation, deriving the basis functions from the knot vectors is simplified. For internal nodes, we can use the recursive formula of de Boor [9] for B-splines

$$B_{i,k}(x) = \left(\frac{x - \gamma_i}{\gamma_{i-1+k} - \gamma_i} \right) B_{i,k-1}(x) + \left(\frac{\gamma_{i+k} - x}{\gamma_{i+k} - \gamma_{i+1}} \right) B_{i+1,k-1}(x), \quad (\text{A.12})$$

where k is the spline's order (e.g., $k = 4$ for cubic B-splines), and the base case is

$$B_{i,1}(x) = \begin{cases} 1 & x \in [\gamma_i, \gamma_{i+1}], \\ 0 & \text{otherwise.} \end{cases} \quad (\text{A.13})$$

However, the recursive formula of Equation A.12 is ill-suited for knot vectors including repeated points, due to unavoidable divisions by zero. Fortunately, de Boor [9] also provides a B-spline definition in terms of divided differences

$$B_{i,k}(x) = (\gamma_{i+k} - \gamma_i) [\gamma_i, \dots, \gamma_{i+k}] b, \quad (\text{A.14})$$

where

$$b = (\gamma_j - x)_+^{k-1} \quad j \in \{i, \dots, i+k\}, \quad (\text{A.15})$$

with

$$(x)_+ = \begin{cases} x & x \geq 0, \\ 0 & x < 0. \end{cases} \quad (\text{A.16})$$

Using Equation A.12, Equation A.13, Equation A.14, Equation A.15, and Equation A.16, where appropriate, the cubic basis functions for each node identified in Table A.2 can be derived. We do not show the complete derivation here, but the results are presented in Table A.3.

However, as can be observed from Figure 3.5, using these basis functions as weights in Equation 3.6 does not yield an interpolating curve. Indeed, the resulting curve does not pass through all the data points, and, therefore, it is only approximating. Catmull and Rom [7] had already observed that cubic B-splines are approximating rather than interpolating. In the presentation of their work at SIGGRAPH 2018 (see their slides¹), Hu

¹The slides are available on the first author's GitHub page: <https://github.com/yuanming-hu/>

Node	$\tilde{x} \in [-2, -1]$	$\tilde{x} \in [-1, 0]$	$\tilde{x} \in [0, 1]$	$\tilde{x} \in [1, 2]$
Internal	$\frac{(\tilde{x}+2)^3}{6}$	$\frac{4-3\tilde{x}^2(\tilde{x}+2)}{6}$	$\frac{3\tilde{x}^2(\tilde{x}-2)+4}{6}$	$\frac{(2-\tilde{x})^3}{6}$
Left boundary	0	0	$\frac{3\tilde{x}^2}{2} \left(\frac{\tilde{x}}{2} - 1 \right) + 1$	$\frac{(2-\tilde{x})^3}{4}$
Right boundary	$\frac{(\tilde{x}+2)^3}{4}$	$1 - \frac{3\tilde{x}^2}{2} \left(\frac{\tilde{x}}{2} + 1 \right)$	0	0
Left external	0	$\frac{(\tilde{x}+1)^2(7-11\tilde{x})}{12}$	$\frac{7(\tilde{x}^3+1)+3\tilde{x}(1-5\tilde{x})}{12}$	$\frac{(2-\tilde{x})^3}{6}$
Right external	$\frac{(\tilde{x}+2)^3}{6}$	$\frac{7(1-\tilde{x}^3)-3\tilde{x}(5\tilde{x}+1)}{12}$	$\frac{(\tilde{x}-1)^2(11\tilde{x}+7)}{12}$	0

Table A.3: *Basis functions for the cubic B-spline interpolation.* Different nodes have different basis functions $B(\tilde{x})$, generated from the corresponding knot vectors shown in Table A.2. Under periodic boundary conditions, all nodes are taken to be internal nodes. The basis functions for internal nodes are from Steffen et al. [39].

et al. [24] mentioned similarly and unsuccessfully attempting to use B-spline interpolation in an APIC setting. They further add that it would not have angular momentum conservation. In light of this, pursuing B-splines interpolation outside a FEM context is all but promising.

A.5 Bicubic Interpolation

So far, the cubic interpolation schemes discussed could be applied dimension by dimension. Bicubic interpolation entirely discards the issue of dimension ordering, at the cost of not easily (or possibly) fitting the form given by Equation 3.6. The scheme, tailored for domains in two dimensions, is a type of polynomial interpolation, and can be expressed as

$$f|_{\mathbf{x}} = \sum_{m=0}^3 \sum_{n=0}^3 a_{m,n} \bar{x}^m \bar{y}^n. \quad (\text{A.17})$$

where $a_{m,n}$ is the coefficient of the product of m -th power of \bar{x} with the n -th power of \bar{y} .

taichi_mpm.

Given the interpolation kernel \mathcal{H} , as shown in Figure 3.3b, we know

$$\begin{aligned} f|_{\mathbf{x}_e} &= q|_{\mathbf{x}_e}, \\ \frac{\partial f}{\partial x}\bigg|_{\mathbf{x}_e} &= \frac{\partial q}{\partial x}\bigg|_{\mathbf{x}_e}, \\ \frac{\partial f}{\partial y}\bigg|_{\mathbf{x}_e} &= \frac{\partial q}{\partial y}\bigg|_{\mathbf{x}_e}, \\ \frac{\partial^2 f}{\partial x \partial y}\bigg|_{\mathbf{x}_e} &= \frac{\partial^2 q}{\partial x \partial y}\bigg|_{\mathbf{x}_e}. \end{aligned} \tag{A.18}$$

The information in Equation A.18 allows us to express Equation A.17 as a linear system

$$f|_{\mathbf{x}} = [1 \quad \bar{x} \quad \bar{x}^2 \quad \bar{x}^3] \mathbf{F}^T \mathbf{A} \mathbf{F} \begin{bmatrix} 1 \\ \bar{y} \\ \bar{y}^2 \\ \bar{y}^3 \end{bmatrix}, \tag{A.19}$$

where

$$\mathbf{A} = \begin{bmatrix} q|_{\mathbf{x}_{i,j}} & q|_{\mathbf{x}_{i,j+1}} & \frac{\partial q}{\partial y}\bigg|_{\mathbf{x}_{i,j}} & \frac{\partial q}{\partial y}\bigg|_{\mathbf{x}_{i,j+1}} \\ q|_{\mathbf{x}_{i+1,j}} & q|_{\mathbf{x}_{i+1,j+1}} & \frac{\partial q}{\partial y}\bigg|_{\mathbf{x}_{i+1,j}} & \frac{\partial q}{\partial y}\bigg|_{\mathbf{x}_{i+1,j+1}} \\ \frac{\partial q}{\partial x}\bigg|_{\mathbf{x}_{i,j}} & \frac{\partial q}{\partial x}\bigg|_{\mathbf{x}_{i,j+1}} & \frac{\partial^2 q}{\partial x \partial y}\bigg|_{\mathbf{x}_{i,j}} & \frac{\partial^2 q}{\partial x \partial y}\bigg|_{\mathbf{x}_{i,j+1}} \\ \frac{\partial q}{\partial x}\bigg|_{\mathbf{x}_{i+1,j}} & \frac{\partial q}{\partial x}\bigg|_{\mathbf{x}_{i+1,j+1}} & \frac{\partial^2 q}{\partial x \partial y}\bigg|_{\mathbf{x}_{i+1,j}} & \frac{\partial^2 q}{\partial x \partial y}\bigg|_{\mathbf{x}_{i+1,j+1}} \end{bmatrix}, \tag{A.20}$$

and

$$\mathbf{F} = \begin{bmatrix} 1 & 0 & -3 & 2 \\ 0 & 0 & 3 & -2 \\ 0 & 1 & -2 & 1 \\ 0 & 0 & -1 & 1 \end{bmatrix}. \tag{A.21}$$

The bicubic interpolation as described above is not monotone. Therefore, it requires clamping.

In a follow-up paper, Carlson and Fritsch [5] extend their ideas from [19] to two dimensions, hence establishing necessary and sufficient conditions on the derivatives to engineer a monotone bicubic interpolating polynomial on rectangular domains. Carlson and Fritsch [5] explain that defining monotonicity on a bidimensional rectangular domain is not as trivial as the unidimensional counterpart. In particular, they choose to ensure that all slices

taken parallel to each coordinate axis are monotone, and further require that the data be consistent. This consistency constraint on the data does not allow for cases where opposite data points share the same value, but different between both diagonals.

Therefore, the resulting algorithm, which they detail in Carlson and Fritsch [6], cannot handle some extreme cases: those with potential inflection, or saddle points in the interpolating polynomial among others. In addition to having to identify these cases, another interpolation scheme would be needed to handle them, which raises a question about the relevance of having monotonicity embedded in the interpolation scheme in the context of our proposed method.

Appendix B

Complementary Figures

In this appendix, we provide figures that complement those presented in Chapter 5, whose structure, when relevant, is mirrored thereafter.

B.1 ASLAM Parameter Assessment

The full performance plots, displaying $n \in [1, 1200]$ rather than $n \in [1, 60]$, for the testing scenarios in Section 5.2, are rendered with interval $n \in [1, 60]$ shaded in gray.

In one dimension, observations made from Figure 5.6 remain valid in the long run as shown in Figure B.1. Indeed, the results are consistent across weighting techniques. Moreover, parametric size $\zeta = 1$ distinguishes itself more than parametric sizes $\zeta > 1$. While density increases, it also seems to stabilize itself after numerous time steps.

The same comments also apply to Figure B.2, which presents the long term results of Figure 5.8. The results are similar across parametric sizes, with UNIFORM weights being better for $\zeta = 1$ under the norm-based metrics. However, in terms of density conservation, SPH weights best the other two weighting techniques, and UNIFORM weights are the worst.

In two dimensions, we can see that SPH weights' relative improvement in terms of norm-based metrics decreases with time. The curves for all weighting techniques nearly coincide with one another as shown in Figure B.3. However, SPH weights remain better at conserving density for both the 2d-TRANSLATION test and the 2d-ROTATION test. Conservation of density for the latter is illustrated in Figure B.4.

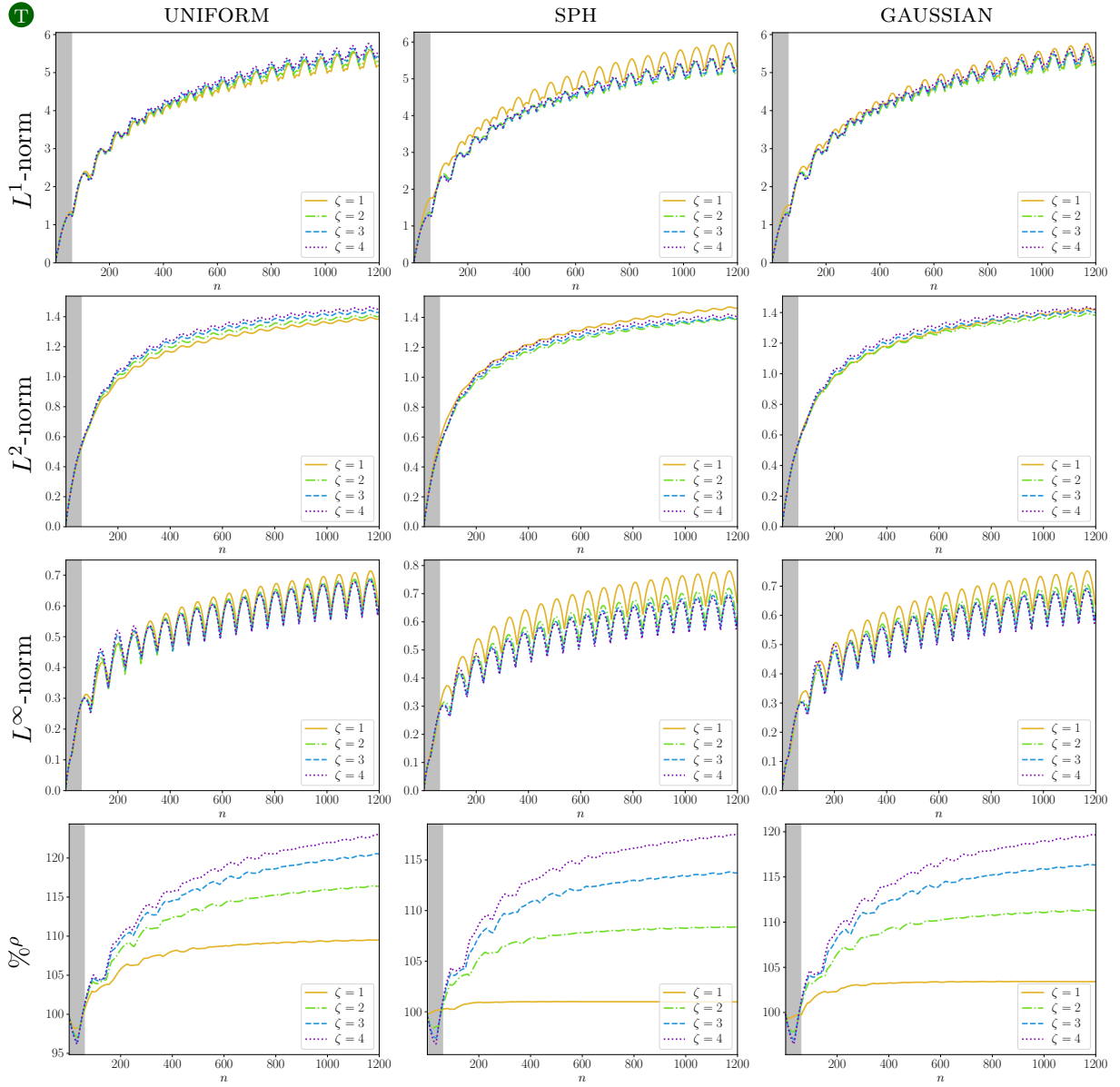


Figure B.1: *Full performance by weighting type for the 1d-TRANSLATION test.* Full performance for Figure 5.5 measured by (from top to bottom) the L^1 -norm, the L^2 -norm, the L^∞ -norm, and density conservation $\% \rho$ for (from left to right) UNIFORM, SPH, and GAUSSIAN weights with parametric size $\zeta = 1, 2, 3, 4$ respectively in solid yellow, dot-dashed green, dashed blue, and dotted purple is consistent with the earliest time steps shaded in gray. Earliest time steps are also shown in Figure 5.6.

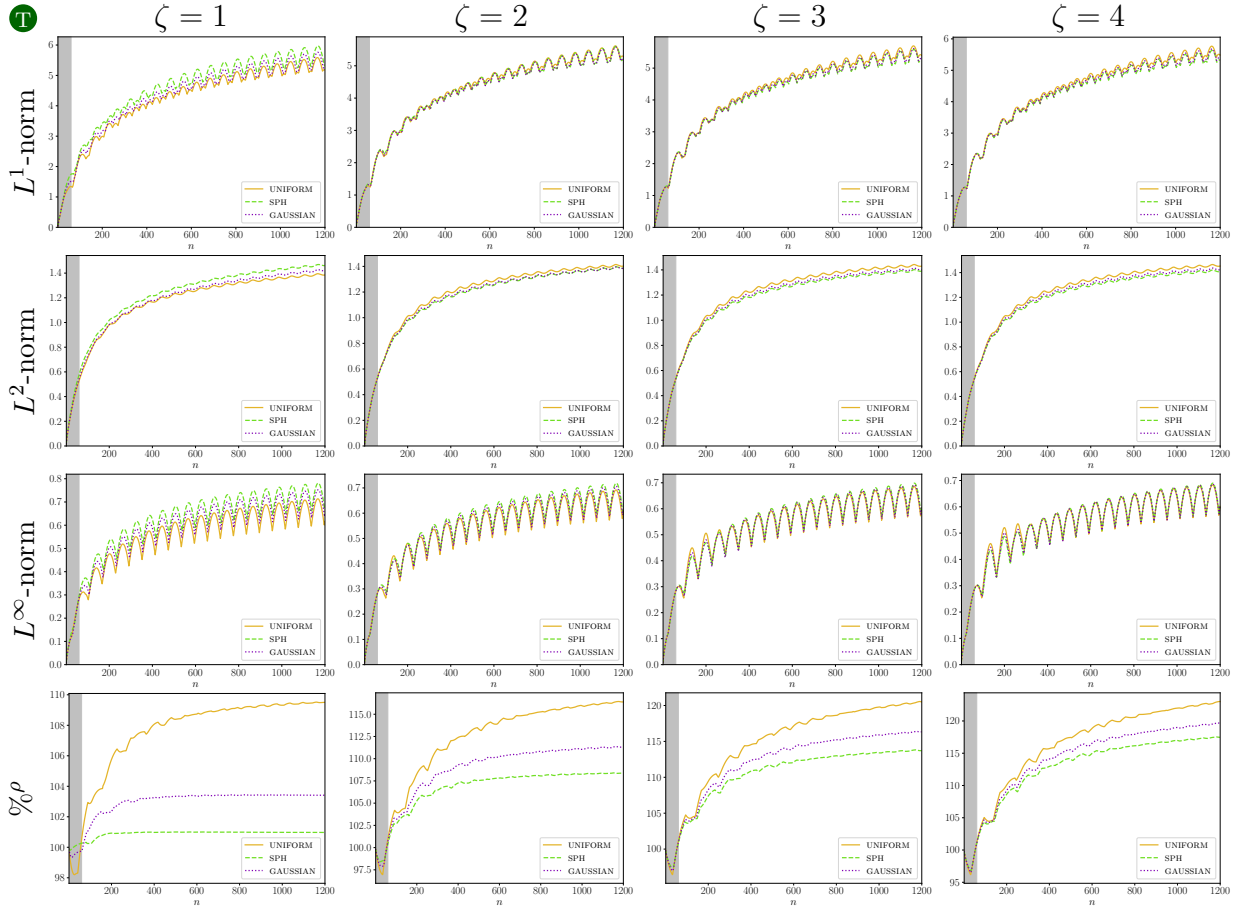


Figure B.2: *Full performance by parametric size for the 1d-TRANSLATION test.* Full performance for Figure 5.7 measured by (from top to bottom) the L^1 -norm, the L^2 -norm, the L^∞ -norm, and density conservation $\% \rho$ for (from left to right) parametric size $\zeta = 1, 2, 3, 4$ with UNIFORM, SPH, and GAUSSIAN weights respectively in solid yellow, dashed green, and dotted purple is consistent with the earliest time steps shaded in gray. Earliest time steps are also shown in Figure 5.8.

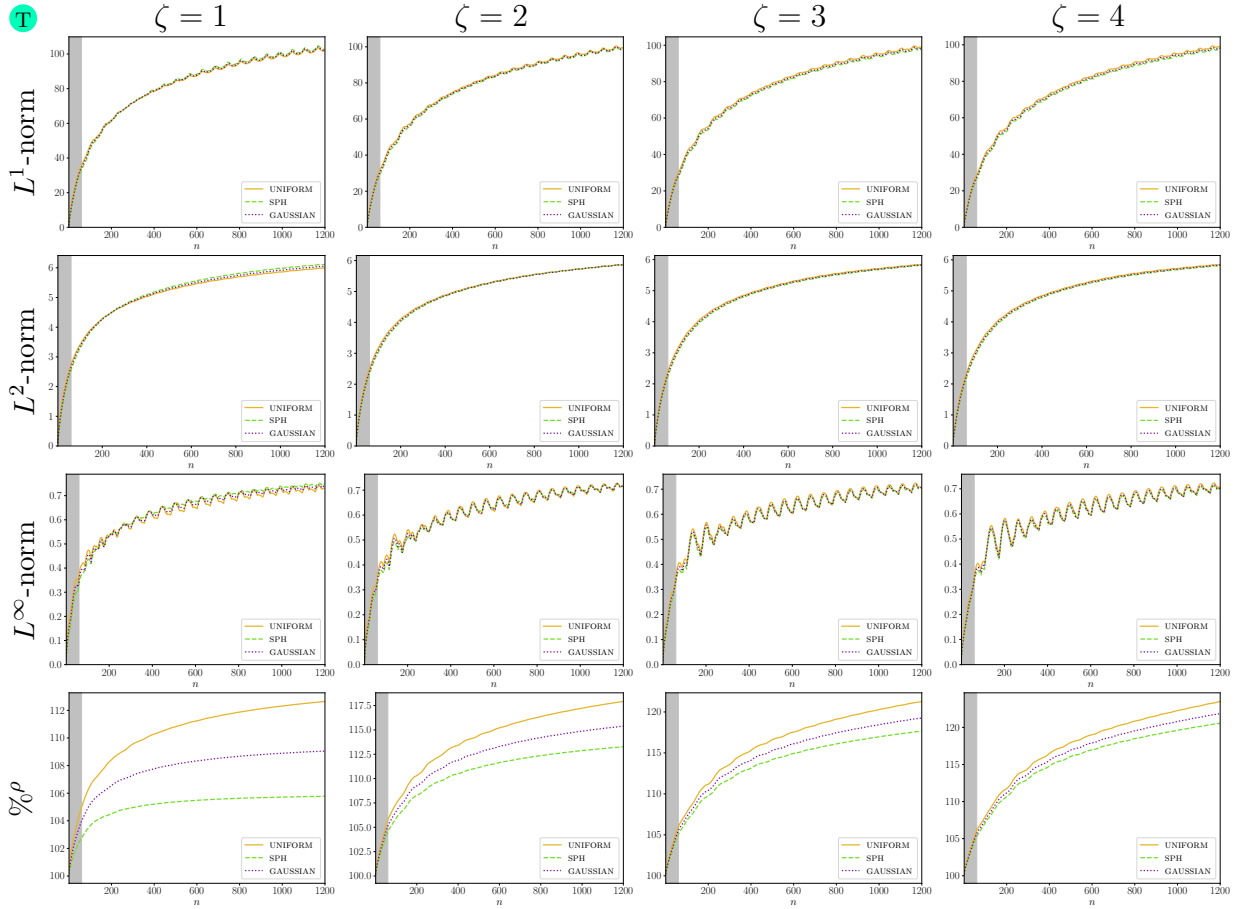


Figure B.3: *Full performance by parametric size for the 2d-TRANSLATION test.* Full performance for Figure 5.9 measured by (from top to bottom) the L^1 -norm, the L^2 -norm, the L^∞ -norm, and density conservation $\% \rho$ for (from left to right) parametric size $\zeta = 1, 2, 3, 4$ with UNIFORM, SPH, and GAUSSIAN weights respectively in solid yellow, dashed green, and dotted purple is consistent with the earliest time steps shaded in gray. Earliest time steps are also shown in Figure 5.11.

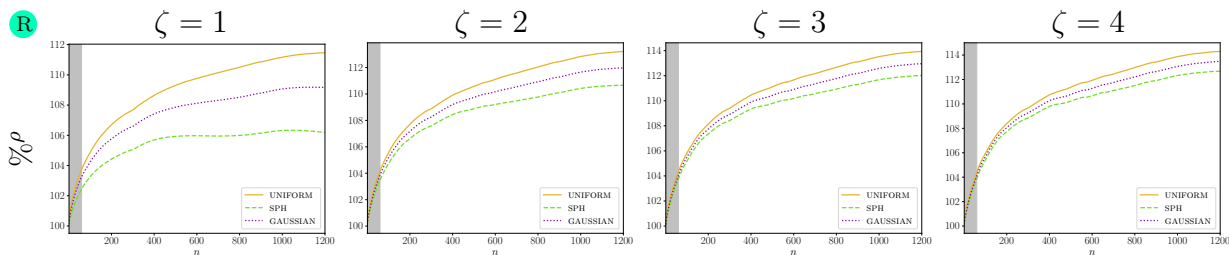


Figure B.4: *Full density conservation by parametric size for the 2d-ROTATION test.* Full performance for Figure 5.10 measured by density conservation $\% \rho$ for (from left to right) parametric size $\zeta = 1, 2, 3, 4$ with UNIFORM, SPH, and GAUSSIAN weights respectively in solid yellow, dashed green, and dotted purple is consistent with the earliest time steps shaded in gray. Earliest time steps are also shown in Figure 5.12.

In Section 5.2, we discarded a comparison by weighting type on account that it was formerly discussed in the unidimensional case. We include it here for thoroughness. Figure B.5 and Figure B.6 present the performance for the 2d-TRANSLATION and 2d-ROTATION tests respectively. In particular, they focus on the earliest time steps $n \in [1, 60]$ since they were not previously shown. The complete interval $n \in [1, 1200]$ is included as an inset with the first steps shaded in gray, where applicable. The results are consistent across both tests as well as across weighting techniques. In terms of norm-based metrics, increasing the parametric size ζ leads to a reduced error. However, it has the reverse effect on density conservation, which varies less with smaller sizes.

B.2 Practical Application Scenario

In Section 5.4, we presented the practical application scenario of a rising plume in a closed box. Figure B.7 shows the second half of the simulation. SL-LIN is clearly the most dissipative method with an almost-uniform density distribution towards the end of the simulation. ASLAM and SL-FC, which were comparable in the first half of the simulation, start to diverge noticeably, with SL-FC keeping more details. MC-FC worsens the visual artefacts introduced earlier in the simulation while MC-LIN seems to develop a similar artefact although to a lesser extent.

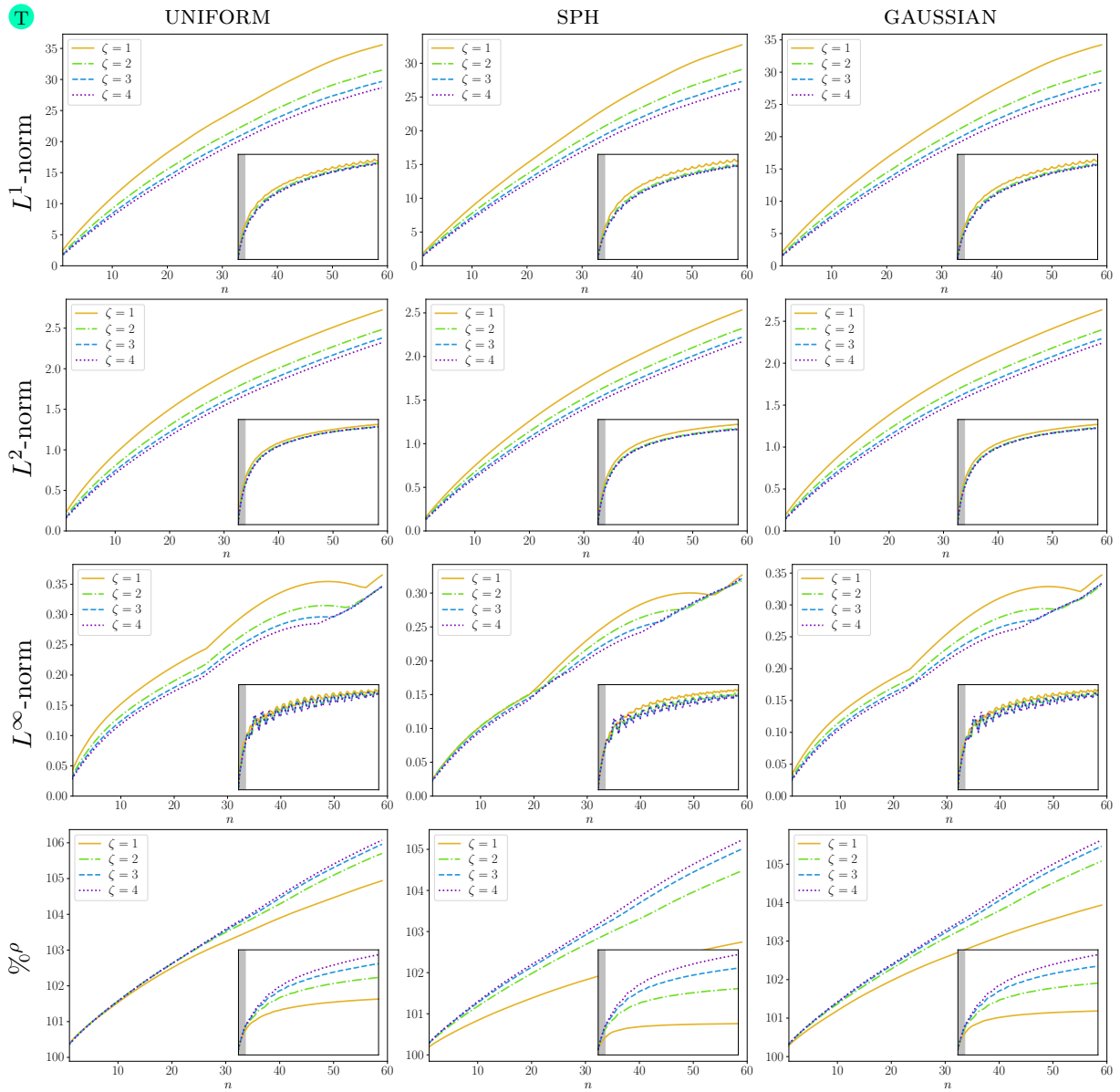


Figure B.5: *Performance by weighting type for the 2d-TRANSLATION test.* Performance for Figure 5.9 measured by (from top to bottom) the L^1 -norm, the L^2 -norm, the L^∞ -norm, and density conservation $\% \rho$ for (from left to right) UNIFORM, SPH, and GAUSSIAN weights with parametric size $\zeta = 1, 2, 3, 4$ respectively in solid yellow, dot-dashed green, dashed blue, and dotted purple agrees with the unidimensional case shown in Figure 5.6. Insets show $n \in [1, 1200]$ with the earliest time steps $n \in [1, 60]$ shaded in gray.

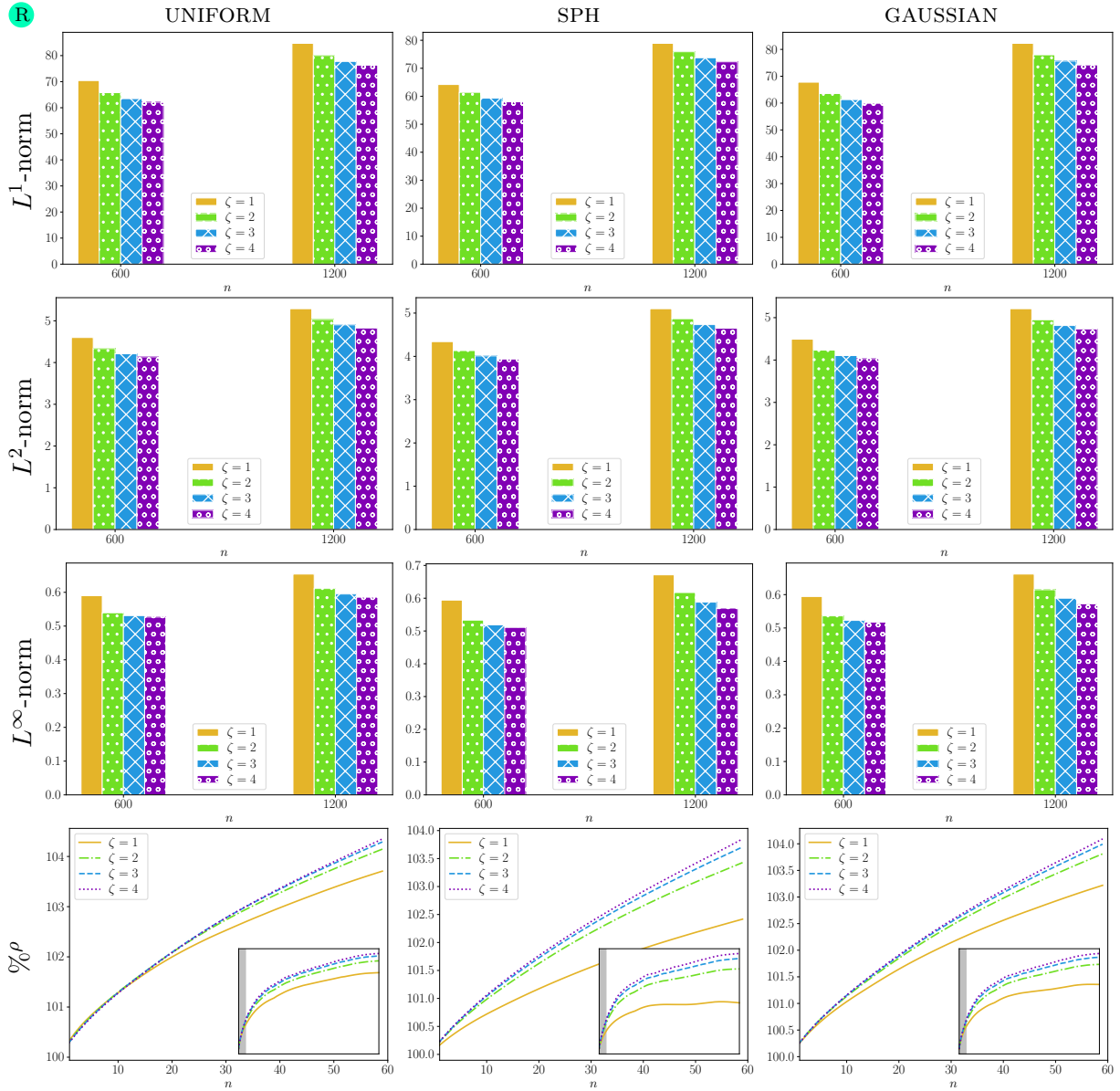


Figure B.6: *Performance by weighting type for the 2d-ROTATION test.* Performance for Figure 5.10 measured by (from top to bottom) the L^1 -norm, the L^2 -norm, the L^∞ -norm, and density conservation $\% \rho$ for (from left to right) UNIFORM, SPH, and GAUSSIAN weights with parametric size $\zeta = 1, 2, 3, 4$ respectively in solid yellow, dot-dashed green, dashed blue, and dotted purple agrees with the unidimensional case shown in Figure 5.6. Insets show $n \in [1, 1200]$ with the earliest time steps $n \in [1, 60]$ shaded in gray.

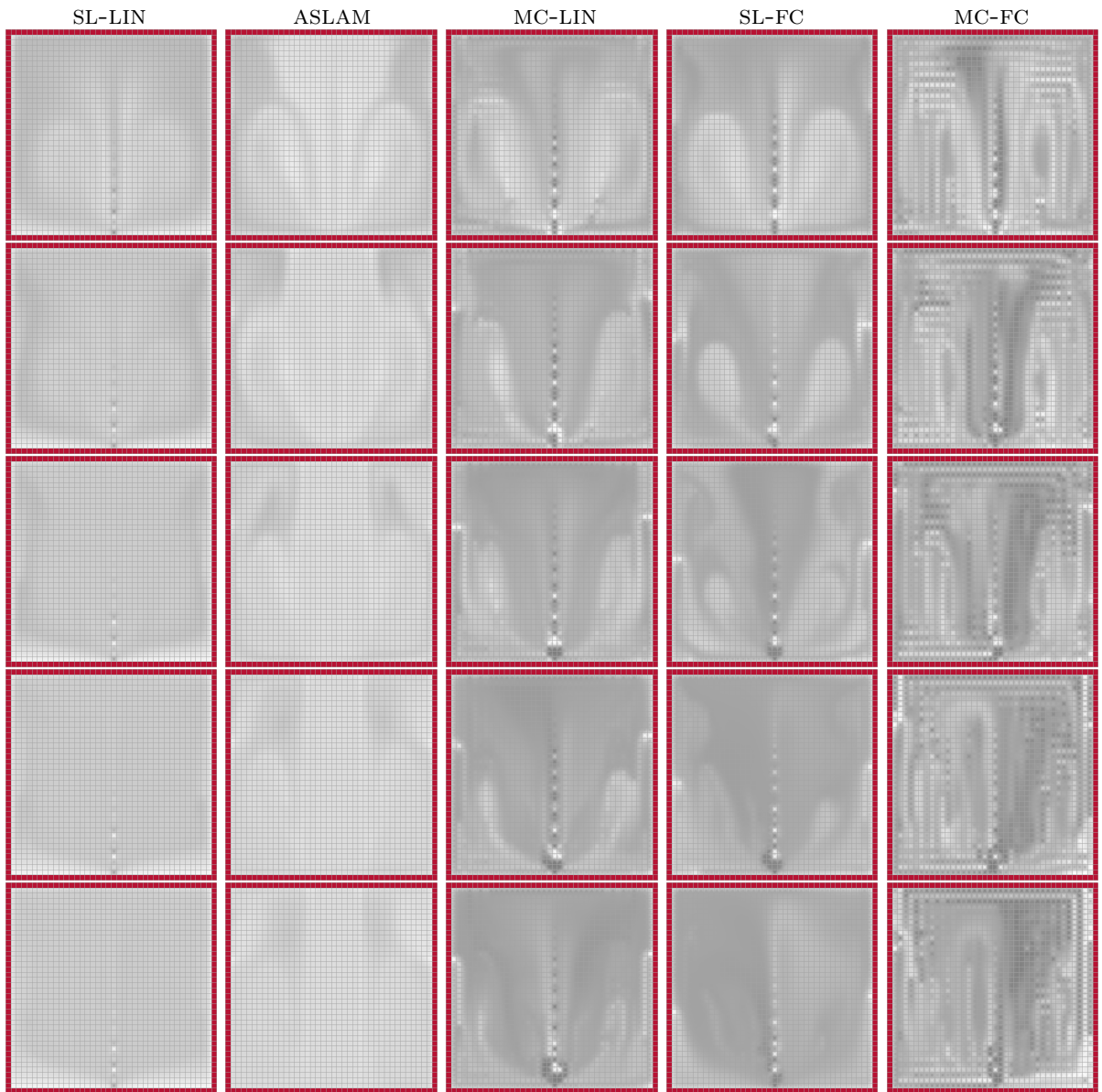


Figure B.7: *Impact of advection method on a rising plume scenario.* Remaining frames at (from top to bottom) $n = 720, 840, 960, 1080, 1200$ from a rising smoke plume in a closed box (in solid red) are simulated using (from left to right) SL-LIN, ASLAM, MC-LIN, SL-FC, and MC-FC. MC-FC aggravates the existing visual artefacts while MC-LIN starts to develop some. SL-LIN remains the most dissipative. ASLAM and SL-FC start to diverge, with SL-FC retaining more details. See Figure 5.21 for the beginning of the simulation.