Instituto Alberto Luiz Coimbra de Pós-Graduação e Pesquisa de Engenharia COPPE UFRJ

A STABLE TENSOR-BASED DEFLECTION MODEL FOR CONTROLLED FLUID SIMULATIONS

Marcelo Caniato Renhe

Orientadores: Claudio Esperança
Marcelo Bernardes Vieira

Rio de Janeiro
Junho de 2017

A STABLE TENSOR-BASED DEFLECTION MODEL FOR CONTROLLED FLUID SIMULATIONS

Marcelo Caniato Renhe

TESE SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

RIO DE JANEIRO, RJ – BRASIL
JUNHO DE 2017

Renhe, Marcelo Caniato

A stable tensor-based deflection model for controlled fluid simulations/Marcelo Caniato Renhe. – Rio de Janeiro: UFRJ/COPPE, 2017.

XII, 82 p.: il.; $29,7$cm.

Orientadores: Claudio Esperança
Marcelo Bernardes Vieira

Tese (doutorado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2017.

Referências Bibliográficas: p. 71 – 78.

1. fluid simulation. 2. tensor field. 3. fluid control. I. Esperança, Claudio *et al.* II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

Agradecimentos

Diante do fim de mais esta etapa, eu gostaria de agradecer a todos os que de alguma forma foram fundamentais para que eu chegasse até aqui. Agradeço a Deus, aos familiares e especialmente aos amigos que me apoiaram nos momentos mais tensos do doutorado, me estimulando a seguir adiante. Obrigado por compreenderem o distanciamento, necessário muitas vezes para que eu pudesse focar e dar andamento ao trabalho. Alguns acompanharam mais de perto que outros: Crys, Renata, Madeira, Douglas, Sara, Marco, Sanderson, Marina, Lucas, Emília, Isabela, Gabi, Schots... Agradeço também aos meus orientadores Claudio Esperança e Marcelo Bernardes, por seu acompanhamento atento durante todo esse tempo. Um agradecimento mais que especial ao Prof. Antônio Oliveira, que foi meu orientador no mestrado e no começo do doutorado, mas infelizmente não pôde acompanhar o desfecho dessa jornada. Agradeço também àqueles que ajudaram em pequenos detalhes, mas que fizeram toda diferença: ao Allan, pela ajuda na geração dos resultados e a confecção das figuras da tese; ao Luiz Maurilio, pela companhia nas várias idas e vindas ao Rio de Janeiro; aos professores do Departamento de Ciência da Computação da UFJF que, mais de uma vez, colaboraram nos momentos em que eu estive mais atarefado com o doutorado; a todos os meus alunos, que pelo contato em sala de aula ou nos corredores da UFJF muitas vezes acabavam sem querer aliviando o stress de estar o tempo inteiro focado no tema da tese; aos funcionários da secretaria do PESC que por diversas vezes se mostraram atenciosos no atendimento às inúmeras dúvidas com os processos internos do doutorado; e a todos os outros que porventura não venham à memória neste momento, mas que também contribuíram para que esse momento se concretizasse. Muito obrigado!

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.)

UM MODELO ESTÁVEL DE DEFLEXÃO BASEADO EM TENSORES PARA SIMULAÇÕES CONTROLADAS DE FLUIDOS

Marcelo Caniato Renhe

Junho/2017

Orientadores: Claudio Esperança
Marcelo Bernardes Vieira

Programa: Engenharia de Sistemas e Computação

A associação entre fluidos e tensores pode ser observada em algumas situações práticas, como em ressonância magnética por tensores de difusão ou em escoamento permeável. Para fins de simulação, tensores podem ser usados para restringir o escoamento do fluido ao longo de direções específicas. Este trabalho visa explorar esta relação tensor-fluido e propor um método para controlar o escoamento usando um campo de tensores de orientação. Para atingir nossos objetivos, nós expandimos a formulação matemática que governa a dinâmica de fluidos para alterar localmente o momento, defletindo o fluido para trajetórias desejadas. Tomando como base abordagens clássicas para simulação de fluidos em computação gráfica, o método numérico é alterado para acomodar a nova formulação. Controlar o processo de difusão pode também ajudar na visualização de campos tensoriais, onde frequentemente busca-se detectar e realçar caminhos de interesse. Os experimentos realizados mostram que o fluido, induzido pelo campo tensorial subjacente, percorre trajetórias significativas, resultando em um método que é numericamente estável e adequado para fins de visualização e animação.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

A STABLE TENSOR-BASED DEFLECTION MODEL FOR CONTROLLED FLUID SIMULATIONS

Marcelo Caniato Renhe

June/2017

Advisors: Claudio Esperança
Marcelo Bernardes Vieira

Department: Systems Engineering and Computer Science

The association between fluids and tensors can be observed in some practical situations, such as diffusion tensor imaging and permeable flow. For simulation purposes, tensors may be used to constrain the fluid flow along specific directions. This work seeks to explore this tensor-fluid relationship and to propose a method to control fluid flow with an orientation tensor field. To achieve our purposes, we expand the mathematical formulation governing fluid dynamics to locally change momentum, deflecting the fluid along intended paths. Building upon classical computer graphics approaches for fluid simulation, the numerical method is altered to accomodate the new formulation. Gaining control over fluid diffusion can also aid on visualization of tensor fields, where the detection and highlighting of paths of interest is often desired. Experiments show that the fluid adequately follows meaningful paths induced by the underlying tensor field, resulting in a method that is numerically stable and suitable for visualization and animation purposes.

Contents

List of Figures **ix**

1 Introduction **1**

1.1 Related works . . . 3

1.1.1 Controlled fluid simulation . . . 4

1.1.2 Anisotropic diffusion . . . 5

1.1.3 Tensor field processing . . . 6

1.2 Organization . . . 12

2 Fundamentals **13**

2.1 Tensors . . . 13

2.1.1 Concept . . . 14

2.1.2 Tensor-related quantities . . . 14

2.1.3 Orientation tensors . . . 15

2.1.4 Tensor fields . . . 17

2.1.5 Anisotropy metrics . . . 18

2.2 Fluid dynamics . . . 21

2.2.1 Fluids basics . . . 21

2.2.2 Navier-Stokes equations . . . 24

2.2.3 Boundary conditions . . . 26

2.3 Stable fluid simulation . . . 27

2.3.1 Data representation . . . 27

2.3.2 Numerical solution . . . 29

2.3.3 Advection . . . 30

2.3.4 Diffusion . . . 32

2.3.5 Projection . . . 33

3 Proposed method **34**

3.1 Proposed continuous formulation . . . 37

3.2 Customized advection . . . 38

3.2.1 Numerical solution . . . 39

3.2.2 Analytical solution 41
3.2.3 Stability discussion 42
3.3 Customized diffusion 44
3.3.1 Proposed anisotropic diffusion 46
3.3.2 Stability discussion 50
3.4 Customized projection 52

4 Results and discussions **54**
4.1 Fluid dynamics comparisons 54
4.1.1 Diffusion and viscosity 55
4.1.2 Advection 57
4.2 Tensor field design for fluid control 59
4.3 General 3D tensor fields 64
4.4 Known issues 67

5 Conclusions **69**

Bibliography **71**

A Anisotropic diffusion masks **79**
A.1 Asymmetric scheme 79
A.2 Symmetric scheme 80
A.3 Diffusion masks 81

List of Figures

2.1 The geometric interpretation of tensors. According to their eigenvalues, the tensors can be described as linear, planar or spherical. 17

2.2 Examples of tensor fields: (a) 2D tensor field generated from 3 reference points; (b) a helical tensor field available at [1]; (c) a DT-MRI brain field; (d) a tensor field constructed for the Bunny geometry. Glyphs as defined in [2] were used for the representation of each individual tensor. 18

2.3 The helical tensor field with colors mapped according to different anisotropy measures. Colors vary from blue (lowest) to red (highest). 22

2.4 The difference between storing velocity values at the center of a cell and at its faces. 28

2.5 Semi-Lagrangian advection process: a particle is traced backwards in time and the new value is obtained by interpolating from the neighboring cells at the particle's old position. 31

3.1 Six cases of a velocity vector being transformed by a tensor. Each case shows a different situation of tensor-vector alignment, producing deflected vectors that may be scaled or shrinked, according to the tensor eigenvalues. 35

3.2 Geometrical interpretation of Equation 3.2. 36

3.3 Illustration of the customized advection process, using Eq. 3.9. 40

3.4 Illustration of the enhanced customized advection, now using Eq. 3.5. 41

3.5 As Δt approaches 0, $\ln \Delta t$ goes to infinity. Thus, we can guarantee there is always a valid time step for our advection. When $\lambda_j > 0$, the linear function is increasing, so there is no limit on Δt. This makes sense, since a positive λ_j indicates a purely dissipative tensor. 45

3.6 Asymmetric finite difference scheme. Source: [3] 47

3.7 Symmetric finite difference scheme. Source: [3] 47

3.8 Two sequences of frames for diffusion in the presence of a linear tensor, oriented horizontally: the top sequence used the asymmetric scheme, while the bottom sequence used th symmetric one. 47

3.9 Two sequences of frames for diffusion in the presence of a linear tensor, oriented diagonally: the top sequence used the asymmetric scheme, while the bottom sequence used th symmetric one. 48
3.10 Two sequences of frames for diffusion in the presence of an isotropic tensor: the top sequence used the asymmetric scheme, while the bottom sequence used th symmetric one. 48
3.11 Sequences of frames for the averaged scheme, using the same three tensors of Figs. 3.8, 3.9 and 3.10. The topmost sequence is the isotropic tensor, the center sequence is the horizontal linear tensor and the bottom sequence is the diagonal linear tensor. 49

4.1 Two tensor fields used for comparison with ordinary fluid simulations. They are shown here in reduced resolution to clarify their structures. Colors indicate the tensor norm. (a) 3-point field; (b) Vertical tube field. 55
4.2 Diffusion sequence for the vertical tube in Fig. 4.1(b). Densities are inserted at grid center. (a) Simulation using Stam's diffusion. The smoke is diffused isotropically, completely unaware of the tensor field influence. (b) Simulation using the anisotropic masks for diffusion. The smoke is dispersed predominantly in the direction of the tube. Parameters: $\Delta t = 0.2$, $S = 10$. 56
4.3 A completely diagonal tensor field and a frame of its correspondent diffusion simulation. Simulation parameters: $\Delta t = 0.2$, $S = 10$. 56
4.4 Diffusion sequence with densities being constantly inserted in the grid center. (a) Simulation using Stam's diffusion, as in Fig. 4.2(a). (b) Simulation using the anisotropic diffusion for the 3-point field shown in 4.1(a). The fluid gradually assumes the shape of the central region of the field. Parameters: $\Delta t = 0.2$, $S = 10$. 57
4.5 Comparison between isotropic (a) and anisotropic viscosity (b). While isotropic viscosity just reduces momentum equally in all directions, the anisotropic viscosity reduces it in specific directions. We show the evolution of both velocity fields at 3 iterations ($t = 5$, $t = 15$ and $t = 25$). The third frame is very close to steady state in both cases. Colors indicate the velocity norm. 58

4.6 (a) Our 32×32 tensor field used in the simulation; (b) Classical fluid simulation, without using Eq. 3.2; (c) Simulation using Eq. 3.2 as an external force; (d) Simulation using Eq. 3.2 in advection. Notice how the simulations in Figs. (c) and (d) generate a velocity field with a shape that resembles the colinear regions of the tensor field in (a). We can also see that this shape is much more accentuated when Eq. 3.2 is used as an advection (d) than as an external force (c). Colors indicate level of anisotropy in (a) and density values in (b), (c) and (d). 59

4.7 Animation sequence for the vertical tube in Fig. 4.1(b). (a) Simulation using Stam's advection. There is no tensor affecting the advection, thus the fluid loses its vertical path after being propelled by a force in a different direction. (b) Simulation using tensor advection. The method ensures that the fluid returns to its intended path, even after being deviated by the external force. No dissipation function was used, reason why densities accumulated outside the path after the force was applied. (c) Simulation using tensor advection, but with dissipation instead. The dissipation is applied everywhere outside the tube. Simulation parameters: $\Delta t = 0.05$, $S = 10$, $\beta = 1.3$ (β does not influence simulation (a)). 60

4.8 48×48 tensor field (d) composed by the sum of fields (a), (b) and (c). For tensor field (a), $\mu = 0$, $\sigma = 0.4$, $\lambda_1 = f$ and $\lambda_2 = 0.1$; for (b), $\mu = 1$, $\sigma = 0.3$, $\lambda_1 = 0.1$ and $\lambda_2 = f$; for (c), $\mu = 0$, $\sigma = 0.1$, $\lambda_1 = f$ and $\lambda_2 = 1 - f$. As this is a bidimensional field, λ_3 was set to 0 for all of them. 61

4.9 Velocity field in six different frames of a simulation with the disk tensor field. Color indicates densities, flowing from the center towards the borders, gradually filling the region. Simulation parameters: $\Delta t = 0.05$, $\beta = 3.0$, and $\alpha = 0.5$, $S = 100$. 63

4.10 Rendered densities for the disk simulation. The fluid gradually fills the region corresponding to the tensor field in Fig. 4.8(d). The fluid starts slowly, building momentum until it reaches a "final" state, where the shape is already formed but the fluid continues its circling motion in a stable manner. Frame (h) shows this state. Simulation parameters: $\Delta t = 0.05$, $\beta = 3.0$, $\alpha = 0.5$, $S = 100$. 63

4.11 $20 \times 20 \times 20$ wafer tensor field, composed solely of planar tensors. Near the center, the tensors have higher norm. The goal was to diffuse the fluid radially and keep it contained in the planar layers. 64

4.12 $20 \times 20 \times 20$ twister field. This field consists of linear tensors oriented so that a vortex can be formed in its core. At the boundaries, planar tensors are used to contain the fluid and an additional set of external linear tensors is used to generate finer motion detail around the twister. 65
4.13 Rendered densities for the wafer simulation. Parameters: $\Delta t = 0.1$, $\beta = 1.3$, $\alpha = 0.5$, $S = 100$. 65
4.14 Rendered densities for the twister simulation. Parameters: $\Delta t = 0.2$, $\beta = 5.0$, $\alpha = 0.1$, $S = 100$. 66
4.15 $20 \times 20 \times 20$ sphere field. The interior of the sphere is composed of linear tensors. These tensors are surrounded by a shell of planar tensors. 66
4.16 Rendered densities for the sphere simulation. Parameters: $\Delta t = 0.1$, $\beta = 3.42$, $\alpha = 0.18$, $S = 1$. 67
4.17 Helix tensor field visualization with direct application of our method. Parameters $\Delta t = 0.05$, $\beta = 6.0$ and $\alpha = 10.0$. (a) $38 \times 39 \times 40$ tensor field. (b) iteration 651. 67

A.1 Asymmetric finite difference scheme. Source: [3] 79
A.2 Symmetric finite difference scheme. Source: [3] 80

Chapter 1

Introduction

Fluids have been a recurring research topic in the computer graphics field for the last couple of decades. Works focused on various kinds of fluids can be encountered in the literature: water, smoke, viscous fluids, turbulent fluids, non-Newtonian fluids, and so on. Many works also focus on secondary fluid phenomena, like bubbles or wave formation, or on the interaction of fluids and solids. The available literature on the subject is quite extensive and spans across many different areas, with the level of strictness of the employed methods depending on the problem being attacked. In solving engineering problems, for example, care must be taken about the physical accuracy of these simulations. Usually, these works are interested in devising methods to predict the behavior of fluids in real scenarios. This poses a serious challenge for researchers, due to the complexities involved in accurately simulating fluid flow. Also, fluid simulations are significantly costly in terms of processing.

Instead of exploring robust approaches for creating physically accurate simulations, many works sacrifice physical realism to produce fluid animations with interesting visual results. These fall on a different category of methods, which try to emulate physical behavior by combining physics with artificial solutions. Without the restrictions imposed by more rigorous applications, these physically inspired methods allow one to disconsider some physics-related parts of the simulations. However, this does not necessarily reduce the complexity of the whole process, since coupling physical and non-physical mechanisms is a delicate procedure.

This kind of combined approach is suitable for visual effects applications. For instance, fluid dynamics could be tweaked in order to keep the fluid contained to a specific path or to make it converge to a predetermined shape. This would enable the animator to produce fluid-like objects and animation sequences. In this context, methods for controlling fluid motion become relevant.

Despite the seeming artificiality of these control mechanisms, the idea of a constrained or directed fluid flow can be observed in certain real scenarios. For example, in diffusion tensor imaging, tensor fields obtained from DT-MRI, a magnetic resonance imaging technique, describe the flow patterns of water inside organic tissue. These diffusion tensors represent the probability of water flowing in specific directions. Interestingly, if we look into porous media flow, tensors are also present as the permeability, which is a measure of the medium telling how much fluid can flow through the pores. So, given this relationship between fluid flow and tensors, a natural question arises: would it be viable to use tensors to control and direct a fluid along paths of interest? How can we provide a straightforward way of simulating anisotropic transport in an easy-to-use and stable manner?

This thesis focuses on exploring the association of fluid dynamics and tensor fields. Our focus is on a special category of tensors defined by WESTIN *et al.* [4], namely the orientation tensors. These are symmetric positive semidefinite rank-2 tensors usually related to covariance estimations. Tractography methods often use this kind of tensor to detect fibrous structures in diffusion tensor fields from MRI procedures. The main objective is to understand how the multivariate nature of tensors may affect fluid behavior during simulation and how it can help on tackling the problem of modeling anisotropic diffusion. This problem has been investigated by physics researchers, such as in [3]. To achieve our purposes, we seek to adapt a common fluid simulation method first proposed by STAM [5], by customizing the mathematical model behind it so that the tensor information is used to locally alter fluid momentum. We also propose a new discretization for the basic simulation steps of Stam's approach in order to reflect the adapted equations. Finally, we want our method to enable the production of fluid-like animations by designing tensor fields suited to the needs of the user.

What we really want to demonstrate in this work is that tensors can be successfully used to control fluids, whether it be for simulation or animation applications. Even though the tensor concept is not well known to the general public, modeling tensor fields for that purpose is a feasible approach, since their design can be automated. Potential users are not required to have any knowledge about tensors. For example, an interface could be provided so that the user would be able to specify a set of curves or surfaces used as basis for the tensor field generation.

Although the idea of constructing tensor fields for controlling fluids may sound too animation-oriented, it helps us in more general contexts. Take as an example the area of tensor field visualization. Fluid motion could be used as a catalyst for perception. Taking into consideration that tensors can be associated with diffusion, colinear and coplanar structures could be highlighted by introducing fluid motion on the tensor field and observing its flow patterns. Regions with more prominent diffusion would be perceived more emphatically by a human observer, due to the natural sensitivity of our visual system to motion. Tensor fields containing real data, though, tend to be very noisy and, consequently, are difficult to visualize. So, designed tensor fields can be useful as a controlled scenario for evaluating the tensor effect on fluids. By inserting tensor information into the fluid simulation, we add uncertainty to flow direction, allowing us to evaluate how a tensor can alter fluid flow.

We can summarize the main contributions of this thesis as follows:

1. a system of partial differential equations whose solution lead to a tensor-based customized description of fluid flow,
2. a stable numerical method for the aforementioned system of equations,
3. a method that allows for constructing tensor fields aimed at controlling fluids,
4. a method potentially applicable to the visualization of diffusion tensor fields.

Although we propose a flexible method, applicable both to simulations and animations, we do not concern about strict physical accuracy in this work, since our problems of interest are mainly visual (animation and tensor field visualization). We are focused on producing interesting animations and on correctly recognizing important information in tensor fields. A more accurate method could possibly enhance the results, especially in the visualization case, but this is not a requirement. For visualization purposes, as long as we are able to highlight the relevant information in the field, physical accuracy should not be a problem.

As it currently stands, the method here proposed is capable of successfully generating directed flow simulations for general 2D tensor fields and is somewhat capable for 3D fields. The higher number of degrees of freedom in 3D may require additional control mechanisms to keep the fluid contained to the tensor field region. In the results section, we will discuss and present the issues related to the 3D simulations.

1.1 Related works

Fluid simulations, in general, are rather unpredictable. To be able to foresee fluid behavior opens a huge array of possibilities. There is a profusion of works in literature intended on devising methods to control fluid flow in order to preset its final configuration. The available approaches are very diverse, as are the applications towards which they are aimed. In this section, the main advancements related to the subject in hand are presented. Since this has been a topic of research for already more than a decade, the list of related works is quite extensive. The bibliographical survey performed here focused on categorizing trends and identifying seminal and relevant works published across the years, as well as the most recent developments regarding the subject. To the best of our knowledge, the main references are considered below. We divided this section into smaller subsections, each one dedicated to a specific area of research that is relevant to our work. Section 1.1.3 is not specifically related to fluids, but represents an important part of this thesis.

1.1.1 Controlled fluid simulation

The work by STAM [5] is a classical reference when it comes to simulating fluids using an Eulerian representation. The semi-Lagrangian advection introduced by Stam results in an unconditionally stable method, which is still widely used as the basis for many other works. Since its publication, many variations and improvements to the method were proposed, such as in [6] and [7]. BRIDSON [8] revisits techniques developed in many works published through the years, and although it does not set itself as a complete survey of the field, it is a good reference for the current state of fluid simulation developments in computer graphics. Also, a recent survey for smoke simulation can be found in [9]. In this work, since we do not intend to simulate complex fluids, we follow Bridson's presentation of the basic algorithm proposed by Stam.

Using fluids to produce controlled animations started with TREUILLE *et al.* [10], which employed user-defined keyframe images to guide the simulation. A more recent keyframe-based approach is available in [11], where stylized animations based on hand-drawn artworks are produced by adapting two other techniques: regenerative morphing [12] and image melding [13]. These and other works, like [14], [15] and [16], were target-driven. This means that the fluid followed arbitrary paths in order to reach and form a pre-specified target shape. FATTAL e LISCHINSKI [17] followed this pattern, but used external forces instead of keyframes to direct the fluid. A similar approach was presented in [18], using a distance function as the external force. There are also some works, like [19], that employ methods to produce high quality animations that are interactively edited by the user, in a process called fluid sculpting.

The works mentioned above are mainly concerned with the final state of the fluid, letting the fluid flow freely during the intermediate steps. Here, we are interested in controlling fluid flow throughout the entire simulation. Most methods related to this problem use external forces to restrict fluid flow to specific paths. In KIM *et al.* [20], for example, a control force is defined to make the velocity field converge to a speficied target flow. Guiding forces are also present in [21], where predesigned flow patterns are used to control high-resolution and turbulent simulations by means of a technique called Lagrangian Coherent Structure. In [22], control forces based on a signed distance field are employed in order to control both path and shape at the same time. There are also methods based on deformation of the underlying grid. SATO *et al.* [23] allows the user to interactively deform the grid, and the velocity fields are adapted accordingly by applying a curl operator to scalar stream functions obtained from the original velocity fields. A similar approach, using vector potentials, is used in [24] to deform 3D flow. Both of these works produce deformed fields without losing incompressibility. Interpolation of flow fields were also attempted in [25] in order to generate incompressible fluid animations.

In our approach, fluid flow is controlled by tensor fields, which are interpreted as an inherent part of the medium. The fluid in our simulations is not rigidly bound to a path, as in typical force-based methods, nor is deformed by precomputed velocity fields. Hence the use of the word **deflection**. We leverage on the probabilistic nature of tensors to smoothly deviate fluid towards paths of interest. Similar rationale have been used before in visualization contexts, where fluids are influenced by tensors to follow specific paths, whether by altering viscosity and pressure [26] or by molding advection and introducing tensor-based external forces [27]. In visualization of diffusion tensor fields, certain paths are associated with fibers, which make highlighting of these paths relevant. Many other works, not necessarily fluid-related, have been proposed to tackle this complex problem. While visualization is not the focus of our method, the model we propose may be adapted for that purpose. Section 1.1.3 deals with these works. In the next section, we discuss research made on anisotropic diffusion, another important part of our proposal.

1.1.2 Anisotropic diffusion

Anisotropic diffusion is a challenging problem. Many physics researchers have proposed solutions to this problem. This kind of diffusion is not rare in nature. It is commonly observed in contexts such as porous media flow and heat conduction in fusion plasmas. Viscosity is also an example of anisotropic process in fusion plasmas. The diffusion of a quantity is considered anisotropic when it changes according to its direction. This means that diffusion may be larger in one direction than on another.

A group of methods used to solve anisotropic diffusion is called the Mimetic Finite Difference (MFD) methods. Mimetic methods are usually focused on mimicking properties of the continuous model of mathematical and physical systems, preserving them at the discrete level [28]. In [29], a discretization scheme is proposed to deal with heat diffusion in magnetised plasmas, without recurring to aligned coordinates, a necessity in the context of magneto-hydrodynamics. VAN ES *et al.* [3] tried to replicate GÜNTER *et al.* [29]'s scheme accuracy properties for a discretization performed in the direction of the strongest diffusion. The number of available MFD methods is quite large, so a good reference for the interested reader is the review article by LIPNIKOV *et al.* [30].

Porous flow is also highly anisotropic. The permeability is defined by a symmetric tensor. Porous flow simulations is often used in reservoir simulation, for example. A method commonly used in these cases is the two-point flux approximation (TPFA) finite-volume scheme [31]. This method uses two reference points to approximate the flux. For simulations in non-orthogonal grids, the multipoint flux approximation (MPFA) method is preferred. It presents good results in the presence of diffusion tensor discontinuity, but generates a non-symmetric diffusion operator, an issue solved later by AAVATSMARK [32]. Specifically for computer graphics, LENAERTS *et al.* [33] propose simulating porous flow through a deformable material by using the Smoothed Particle Hydrodynamics (SPH) method, considering the pores macroscopically. Since it is a work focused on computer graphics applications, they are able to abstract from small details in order to obtain plausible porous simulations with low computational effort.

In our work, we intend to deal with anisotropic diffusion by employing some discretizations proposed by GÜNTER *et al.* [29] and discussed in [3]. Section 3.3 will discuss them in detail.

1.1.3 Tensor field processing

During the development of this thesis, the area of tensor field processing for visualization had an important role in our research. Although our focus is not on visualization, we tried many ideas from works related to this subject. So, we provide in the following subsections an extensive review of the literature involving tensor field visualization and tractography methods.

Tensor field visualization

Methods for visualizing tensor fields are extremely relevant in the context of diffusion tensor imaging (often labeled as DTI). Through magnetic resonance, it is possible to obtain diffusion patterns in organic tissue. The diffusion of water molecules is limited by the cell membranes and by the presence of large protein molecules, resulting in an anisotropic diffusion. The geometrical and physical properties of the tissue also affect diffusion orientation, so that we can use the diffusion information acquired from MRI to infer the structure of the tissue [34].

The simplest way of visualizing tensor information in MRI images is by mapping isolated scalar quantities onto the image itself, using some kind of shading to represent the desired values. The quantities usually used in these visualizations are anisotropy measures as the ones presented in Section 2.1.5. These were among the first attempts on attacking the problem of visualization of tensor fields, seeking to aid on interpretation of DTI images. VILANOVA *et al.* [34] discusses these and many other approaches that appeared in this first moment. Although simple and easy to interpret, the extraction and separate visualization of these pieces of information does not give much insight on connectivity relationships and diffusion direction.

The same anisotropy measures mentioned above can be used in a volume rendering scheme. Usually, this volume rendering approach is combined with other elements, such as tensor glyphs, to convey more information [34]. Glyphs are graphical representations of information. As for diffusion tensors, glyphs usually encode information such as anisotropy and diffusivity through shape characteristics like size, color and position, for instance. Different kinds of glyphs have been used in the past for tensor visualization. Pierpaoli *et al* [35] and Laidlaw *et al* used ellipsoids. However, disadvantages associated with the use of this shape led the research community to search for other kinds of glyphs to represent tensors. A common problem with ellipsoids is ambiguity due to viewer position. Kindlmann [2] decided to employ superquadrics instead, taking the positive characteristics of ellipsoids, but avoiding its flaws.

The glyph-based techniques mentioned above fall into a category of methods that approach the problem in a discrete fashion. They are very effective on displaying local information, but they fail on portraying an accurate view of the entire tensor field. Being able to capture a meaningful global view of the field can be very useful in identifying colinear and coplanar structures. More specifically, in the case of DT-MRI images, these structures are often associated with fibers. For this reason, methods trying to map the connectivity between diffusion tensors are commonly referred to as fiber tracking (or tractography) methods [34].

The main idea behind these tracking methods is to determine a set of continuous paths through the field, starting from initial points called seeds and ending in regions where certain stopping criteria apply. Usually, these seed points are defined by the user, but there are also a number of publications focused on the development of new methods for automatic generation of seeds [36–38]. The stopping criteria ensures that the tracking will not continue in regions which are not of interest, such as isotropic regions. Thus, a common way of defining a stopping criterion is by making use of some anisotropy measure. The most straightforward way of tracking is to use the main eigenvector of a tensor to guide the generation of the streamlines. As regions of high anisotropy are commonly associated with location of fibers in DT-MRI images [39, 40], a vector field can be extracted from the tensors and used to guide the tracking procedure. The paths obtained, then, can describe these fibers with a good amount of accuracy.

The FACT algorithm [41], which relies on the kind of approach discussed above, is among the most used fiber tracking methods in the literature. Despite having being published several years ago, it is still widely used due to its simplicity, being the standard tracking algorithm in some tractography toolkits [42]. The acronym FACT stands for fiber assignment by continuous tracking. The algorithm connects voxels in a discrete environment by tracking a continuous vector field instead of a discrete one. This ensures that the correct voxels will be connected, which would not be the case if the voxels were selected by simply observing the direction to which tensors point in each voxel. Naturally, despite simple and efficient, FACT has limitations, especially where it concerns resolution and direction determination. The first one is associated with the imaging process itself, while the second one is a direct limitation of the method, which cannot differentiate between afferent and efferent fibers.

Delmarcelle and Hesselink [43] extended the basic idea of streamlines to what they called a **hyperstreamline**. While streamlines are usually defined based on partial tensor information (the most common piece of information used is the main eigenvector), a hyperstreamline aims on encoding the full tensor information in a continuous path. This is done by plotting the line along the main eigenvector direction, but also expanding it laterally using the minor eigenvectors. The hyperstreamline produced is then represented in the form of a tube or a helix, each one having its own advantages and disadvantages in certain situations. Their method can even visualize unsymmetric tensor fields by means of field decomposition.

Another attempt on tackling the tracking problem was made by WEINSTEIN *et al.* [44], with the **Tensorlines** method. Differently from previous works, the authors seek to maintain and stabilize the tracking through regions comprised of isotropic tensors. These regions are often avoided in the hyperstreamlines method by the employment of termination criteria. Obviously, the tensorlines method also makes use of specific conditions to halt the tracking procedure, but it tries to pierce through isotropic regions that may be associated with acquisition noise. In order to make it possible, they devised two additional terms, which they refer to as advection vectors, to be incorporated into the propagation equation. The probability that the propagation will puncture through the planar and isotropic tensors is defined by an user-specified parameter, which can be adjusted according to the dataset.

Some tracking algorithms make a distinction between linear and coplanar structures representations by defining streamsurfaces alongside streamlines. The idea behind streamsurfaces is to provide a means of representing coplanar structures more naturally. Diffusion in coplanar structures are better represented by surfaces than by a set of aligned streamlines. ZHANG *et al.* [45] implement this idea by visualizing connectivities in a DT-MRI brain image using streamtubes and streamsurfaces. They also define some criteria to determine if a tube or surface should be displayed, in order to reduce cluttering. Plus, anatomical landmarks are provided so physicians can have a better interpretation of the visualized structure. Another similar work, presented in [46], uses streamsurfaces to identify regions of crossing fibers. These methods were merely concerned in representing regions with planar anisotropy as a continuous surface. However, no treatment of the anisotropy variation along each surface was considered. More recently, SONG *et al.* [47] proposed visualizing anisotropy of diffusion on streamsurfaces through two different methods: one using a merging ellipsoids technique, and another one using LIC (line integral convolution) textures.

KONDRATIEVA *et al.* [48] proposed the usage of particle tracing on GPU in order to achieve a dynamic visualization of the tensor field. They adapted a particle engine designed to interactively visualize 3D flow [49] so it could be applied to DT-MRI visualization. They based themselves in Weinstein's work [44] to extract a vector field from the tensors. For the rendering part, they use a number of different visualization options, such as texture splats, streamlines and streamtubes.

Later, LEONEL *et al.* [50] also pursued a dynamic visualization through particle tracing by developing an adaptive method, based on a series of user-defined parameters. These parameters are used to keep a priority list of locations where particles should be inserted. The method takes into account not only tensor characteristics, but also the user viewpoint, in order to determine which features should be highlighted in the field. Although it shows good results, the method is highly dependent on the parameters. This requires that users have a previous knowledge of the field to achieve interesting results. In an attempt to reduce parameter dependency and improve the particle creation criterion, an extension to that work was presented in [51]. This improved work uses multiresolution by decomposing the tensor field in subsamples. The distance to the observer is used as a parameter to determine the level of detail necessary for the presentation of tensors, reducing cluttering and enhancing the visualization.

As stated in the introduction, this thesis is concerned with the problem of identifying these regions of interest, but without the focus on visualization. Nonetheless, the work here presented could be adapted to visualization purposes if needed. In [27], for instance, a preliminary version of the developed method was used to visualize synthetic tensor fields. Colinear structures were tracked through an Eulerian fluid simulation approach. The method was also tested on real DT-MRI brain tensor fields. Although quite capable for synthetic fields, the brain example required specific adjustments in order to obtain acceptable visualizations, due to discontinuities commonly present in those kinds of fields.

Yet another, more recent, fiber tracking method is presented in [52]. The algorithm, called EuDX, is a fast deterministic method which can satisfactorily handle regions of fiber crossings. The authors argue that they can correctly identify these crossing fibers, assuming the reconstruction algorithm supports them. This tracking method is part of the DIPY tractography library [53], which also includes a clustering method called Quick Bundles [54]. The idea of fiber clustering comes from the knowledge that fibers are usually oriented as bundles in organic tissue. So, it makes sense to search for similarities between adjacent fiber tracts in order to determine these bundles. These methods are intimately related to fiber tracking approaches, since most of them rely on tracking procedures to construct the bundles. In general, clustering methods use distance and shape comparisons between fiber pairs in order to achieve their goals. The next section presents a brief discussion about some of these methods.

Fiber clustering

ZHANG e LAIDLAW [55] were among the first to deal with this subject. The authors use an agglomerative hierarchical clustering method in order to cluster the streamtubes [45] mentioned in the previous section. They define a simple criterion based on distance for clusterization. Later, DING *et al.* [56] define a different similarity measure between fibers by introducing the concept of a corresponding segment, which is based on a point-wise correspondence along portions of each fiber. This measure is used to clusterize fibers using a hierarchical method similar to the K-nearest-neighbors algorithm. They then calculate a number of quantifiable properties which characterize each bundle, such as curvature, torsion and mean diffusivity.

While DING *et al.* [56] try to model bundle geometry with a series of averaged parameters, COROUGE *et al.* [57] seek to preserve local shape characteristics for each curve in a bundle. The clustering itself is accomplished through the use of alternative distance metrics, instead of the Euclidean metric used in previous works. Bundles with low cardinality are considered outliers and thus discarded. The authors focus on obtaining meaningful anatomical bundles for white matter tracts in brain images.

WELDESELASSIE e HAMARNEH [58] adapt graph cuts to DT-MRI data in order to find a global optimal solution for the task of segmenting DT-MRI images into meaningful anatomical regions. The method depends on user input for defining major constraints for the segmentation procedure. Although this method does not rely on fiber tracking, it may be considered as a different approach for the identification of fiber clusters.

Another work, presented by HLAWITSCHKA *et al.* [59], introduces a local coherence measure for fiber tracts based on deviations between neighboring tracts. The Tensorlines method [44] is used to generate the fiber tracts in some of their results. The coherence measure itself can be used as a scalar field for visualization.

For a full three-dimensional view of the dataset, they use volume rendering with a transparency mapping based on coherence values, naturally reducing noise and clutter.

Usually, clustering results are dependent on a ground truth obtained via manual classification [60]. On this matter, several works focused on developing methods which could automatically determine anatomically correct clusters. DODERO *et al.* [61] present an approach for automatically detecting bundles through a framework called Dominant Sets. The authors argue that two specific characteristics of this framework make it a good choice for the problem in hand. First, it is robust to noise, and second, the inherent similarity of the data is the only information needed for determining the number of clusters. They validate their method against both synthetic and real datasets. ZHANG *et al.* [62], on the other hand, use a proximity-based metric, just as in [57], to interactively identify bundles, aided by an expert, generating a fiber bundle template. This template is then used to guide new clusterings, allowing an automatic labelling procedure to take place. Their method labels correctly major bundles, but is not very successful with smaller ones.

As already mentioned, recently GARYFALLIDIS *et al.* [54] published the QuickBundles method for fiber clustering. The authors aspired to obtain a simple and efficient method for generating clusters. In order to achieve this, the method does not allow any reassignment to kernels or updating phases. They also developed their own distance metric, which they call MDF (minimum average direct flip). The algorithm also allows customization of the threshold parameter, making the clusterizations more or less simple. This may be used to reveal interesting features about the dataset or to change the level of detail by splitting or merging bundles. The method presents linear complexity in a best-case scenario.

Similarity metrics play a central role in these clustering methods. Thus, some works have attempted to evaluate and validate these metrics and algorithms. MOBERTS *et al.* [60] propose a framework to validate different clustering methods. They devised a measure which captures the difference between the established ground truth and the obtained clusters. The measure was fine-tuned with the help of physicians and then used to evaluate some clustering algorithms. SILESS *et al.* [63], in a more recent work, compared the classical k-means algorithm with the Quick Bundles [54] clustering method mentioned above using a metric called Point Density Model. They also compared the most common distance metrics used in fiber clustering methods.

As streamline visualization is useful not only to tractography, other works can also be found in the literature focused on streamline clustering in different contexts. As an example, YU *et al.* [64] presents a hierarchical method for clustering streamlines and simplifying visualization of three-dimensional flow fields, using similarity metrics based on neighborhood and geometry. Several other works can be found on this matter. They will not be presented here, since they are outside the scope of our work.

1.2 Organization

This thesis is organized in the following way: Chapter 2 discusses all the fundamental concepts and methods used in the development of our work. We introduce the basics of orientation tensors and tensor fields, as well as the theoretical background related to fluid dynamics and fluid simulation for computer graphics. In Chapter 3, we present our proposed deflection model for fluids based on tensor fields. We show how the Navier-Stokes equations can be adapted to account for the tensor influence, both in the continuous and discretized models, and a customized simulation method is proposed. We then present our results in Chapter 4, comparing it to traditional fluid simulation and discussing the limitations of our method. We finish this thesis in Chapter 5, mentioning some different approaches that have been attempted and projecting some possible future lines of work to further improve our developments.

Chapter 2

Fundamentals

2.1 Tensors

When we deal with physical quantities with an associated direction, we resort to the concept of vectors to represent them. A vector, as a mathematical entity that has magnitude and direction, is capable of representing quantities that are slightly more complex than those that can be fully described by scalars. Some examples of vector quantities include force, velocity, acceleration and eletrical field. Taking velocity as example, if we imagine a vehicle travelling with a speed of $50m/s$, this single piece of information is not enough to fully describe the trajectory of the vehicle. Besides the speed, which is the velocity magnitude, we need to specify towards which direction the vehicle is moving. Hence, a **vector** is necessary to adequately represent the velocity quantity.

However, in various situations, a vector may also be not enough. Some physical quantities can vary in different directions, while still in the same position in space. This means that, for a point (x, y) in a 2D Euclidean space, this kind of quantity may have an associated value in x direction and a different value in y direction. Obviously, a vector, having a single associated direction, will not be able to fully represent this quantity. In this context, it becomes mandatory to use a more comprehensive mathematical structure, capable of representing more information than a vector. That structure is what we call a **tensor**.

This work deals with simulations involving tensor fields. In this section, we will present the tensor concept and the definition of tensors fields. In particular, we will discuss about the orientation tensor, which is the kind of tensor we are interested here.

2.1.1 Concept

A tensor is an abstract mathematical entity. It can be defined as a linear transformation between vector spaces. It can also be understood as a generalization of scalars and vectors. A scalar can be seen as a rank 0 tensor, while vectors are associated to tensors of rank 1. The tensor rank determines its number of components. A rank n tensor has 3^n components. Rank 2 tensors, or second-order tensors, are called **dyadic tensors** in tensor terminology [65]. The tensors we will be discussing throughout this work fit in this category.

A second-order tensor possesses $3^2 = 9$ components and it can be represented by a 3×3 matrix. As mentioned before, tensors are especially useful to represent quantities that vary in more than one direction for a single point. In the specific case of a second-order tensor, there exist two associated directions. A classical example, mentioned by [65], is the stress applied to an object. The relationship between force and stress is given as:

$$F = T \cdot A,$$

where T is the applied stress, A is the area over which the stress is applied and F is the resulting force.

However, there isn't only a single kind of stress. Whenever a stress is applied to an object, there is a possibility that we have two kinds of stress acting on it: a tensile stress, which produces a force component normal to the surface, and a shear stress, which generates a tangential force. In order to S to embrace, simultaneously, these two contributions in distinct directions, the solution is to use a tensor for its representation. Thus, the previous equation assumes the following form:

$$d\mathbf{F} = \mathbf{T} \cdot d\mathbf{A},$$

where $d\mathbf{F}$ and $d\mathbf{A}$ are vectors and $\mathbf{T}$ is a tensor. The stress was the first physical quantity to be described as a tensor. The latin word for stress (*tensus*) is the origin for the term[65].

2.1.2 Tensor-related quantities

Before we delve into specifics, let us first introduce some additional concepts and quantities regarding tensors. There are a number of useful quantities for analyzing a tensor characteristics. These quantities make up the fabric of several important measures, such as the ones presented ahead in this section.

As mentioned before, tensors can be seen as a generalization of simpler mathematical elements. Taking vectors as an example, we consistently make use of auxiliary vector-related opterations, such as a vector magnitude, its dot product or its cross product, just to name a few. Some of these are also available when we are dealing with tensors.

Let us consider two second-order tensors **T** and **U**. We can define a dot product between these two tensors. This product is a scalar invariant, i.e., it does not depend on position or coordinate system orientation. Its mathematical definition is presented below:

$$\mathbf{T} : \mathbf{U} = \sum_{i=1}^{3} \sum_{j=1}^{3} T_{ij} U_{ji}. \tag{2.1}$$

Notice that for symmetric tensors (the kind of tensors we are interested in this work), we can disregard the order of the indices i and j for **U** in the summation above. Similarly to vectors, the tensor magnitude can also be defined as the square root of its dot product. Thus, taking symmetry into account, we obtain:

$$|\mathbf{T}| = \sqrt{\mathbf{T} : \mathbf{T}} = \sqrt{\sum_{i=1}^{3} \sum_{j=1}^{3} T_{ij}^2}. \tag{2.2}$$

The equation above defines the tensor magnitude in terms of its individual elements. We can also define the tensor norm in terms of its eigenvalues as follows:

$$||\mathbf{T}|| = |\lambda_{max}|, \tag{2.3}$$

where λ_{max} is the largest eigenvalue.

The eigenvalues can be used in many ways in order to analyze a tensor. In a diffusion context, for example, the eigenvalues represent the principal diffusivities [66]. They are also used to identify in a tensor its degree of **anisotropy**, which is the dependence on direction observed in certain physical properties. Plus, by summing them up, we obtain the **trace** of the tensor, which is also an invariant:

$$Tr(\mathbf{T}) = \lambda_1 + \lambda_2 + \lambda_3. \tag{2.4}$$

The trace, as well as the other relations discussed so far, appear in the calculation of many of the metrics we usually see in the literature. We will get a glimpse of their usage when we discuss anisotropy metrics in Section 2.1.5.

2.1.3 Orientation tensors

Inividually, tensors can be easily described and analyzed. When looking at a tensor field, however, the information of each individual tensor alone is usually not enough to attain a meaningful understanding of the field. Identifying relations between different tensors may provide additional insight on the field. For instance, information like connectivity between tensors can be of significant value in some contexts.

In [4], a particular kind of tensor, namely the **orientation tensor**, is introduced. This is a special case of a non-negative symmetric rank 2 tensor capable of estimating orientations in a field, which makes it especially interesting for our work. Additionally, these orientations allow for capturing the desired connectivity information and establishing continuous paths along the field.

Mathematically, a local orientation tensor can be defined as follows:

$$\mathbf{B} = \sum_{n}^{i=1} \lambda_i \mathbf{e}_i \mathbf{e}_i^T, \tag{2.5}$$

where λ_i are the eigenvalues and e_i are the associated eigenvectors. Consequently, $\mathbf{B}$ is a linear combination of rank-1 tensors formed by $\mathbf{e}_i$ and $\mathbf{e}_i^T$ with a single non-zero eigenvalue λ_i. The tensor eigenvectors define the principal directions of the tensor and form an orthonormal basis. The tensor itself is a linear transformation that amplifies a vector along the main axes $\mathbf{e}_i$ according to eigenvalue λ_i.

The tensor fields we use here are initially normalized. We can, however, amplify the tensor effect by scaling each tensor by a boosting factor, which we will refer to as β. So, the tensor $\mathbf{T}$ we actually use throughout our simulations can be defined as:

$$\mathbf{T} = \beta \mathbf{B}. \tag{2.6}$$

In order to represent T in terms of its linear, planar and spherical features, a decomposition of the equation above can be obtained in $\mathbb{R}^3$. Thus, $\mathbf{T}$ is now defined as:

$$\mathbf{T} = (\lambda_1 - \lambda_2)\mathbf{T}_l + (\lambda_2 - \lambda_3)\mathbf{T}_p + \lambda_3 \mathbf{T}_s. \tag{2.7}$$

The main benefit of this representation is we obtain an insight about the geometrical characteristics of the tensor. By comparing its eigenvalues, we can determine if a tensor is approximately linear or if its shape is closer to a sphere, for example. Assuming $\lambda_1 \geq \lambda_2 \geq \lambda_3$, if we have the first eigenvalue much greater than the other two, then the tensor has an approximately linear shape. On the other hand, if $\lambda_1 \approx \lambda_2 >> \lambda_3$, then we have a planar tensor. Finally, if all eigenvalues are approximately equal, then the tensor shape tends to a sphere. Figure 2.1 shows each kind of tensor with its respective geometric representation.

Determining these shapes is extremely relevant for applications in which tensors represent local water diffusion. This kind of analysis can provide information on diffusion direction. While linear and planar tensors have preferred directions for diffusion, spherical tensors have no main orientation associated. Some metrics are usually employed in order to correctly classify each tensor geometry. These anisotropy metrics are presented in the next section.

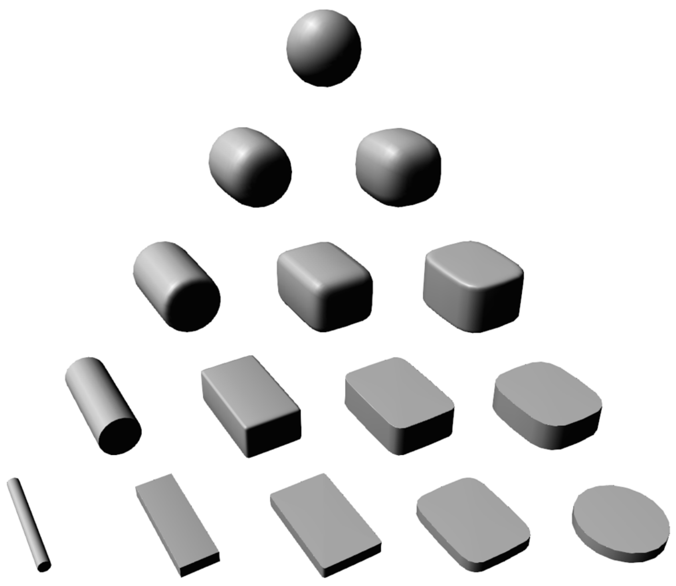


Figure 2.1: The geometric interpretation of tensors. According to their eigenvalues, the tensors can be described as linear, planar or spherical.

2.1.4 Tensor fields

Let us now consider a set of orientation tensors disposed in a subset of the Euclidean space. The concept is similar to the one used for vector and scalar fields. For each point $\mathbf{x}$ in a subset S of $\mathbb{R}^n$, there is a tensor $\mathbf{T}$ assigned. This forms what we call tensor fields. Figure 2.2 shows some examples of fields, using glyphs (mentioned in Section 1.1) to represent them.

Tensor fields can be generated in many ways. For example, they can be constructed directly from chosen mathematical relations, or they can be derived from real data, like in diffusion magnetic resonance imaging. As explained in Section 1.1.3, this is a process where images are produced by capturing the diffusion of water in human tissue and encoding the information in a tensor. This procedure is highly subject to noise, and tensor fields generated from it usually have poor resolution. This complicates the task of correctly estimating individual tensors characteristics, as well as identifying diffusion patterns across the tensor fiels. So, this motivated the development of metrics for minimizing these issues, which will be addressed in the next section.

In our work, tensor fields are interpreted as an inherent part of the medium, naturally influencing a fluid flowing through the region. This will be discussed in detail when we get to our method, in Chapter 3.

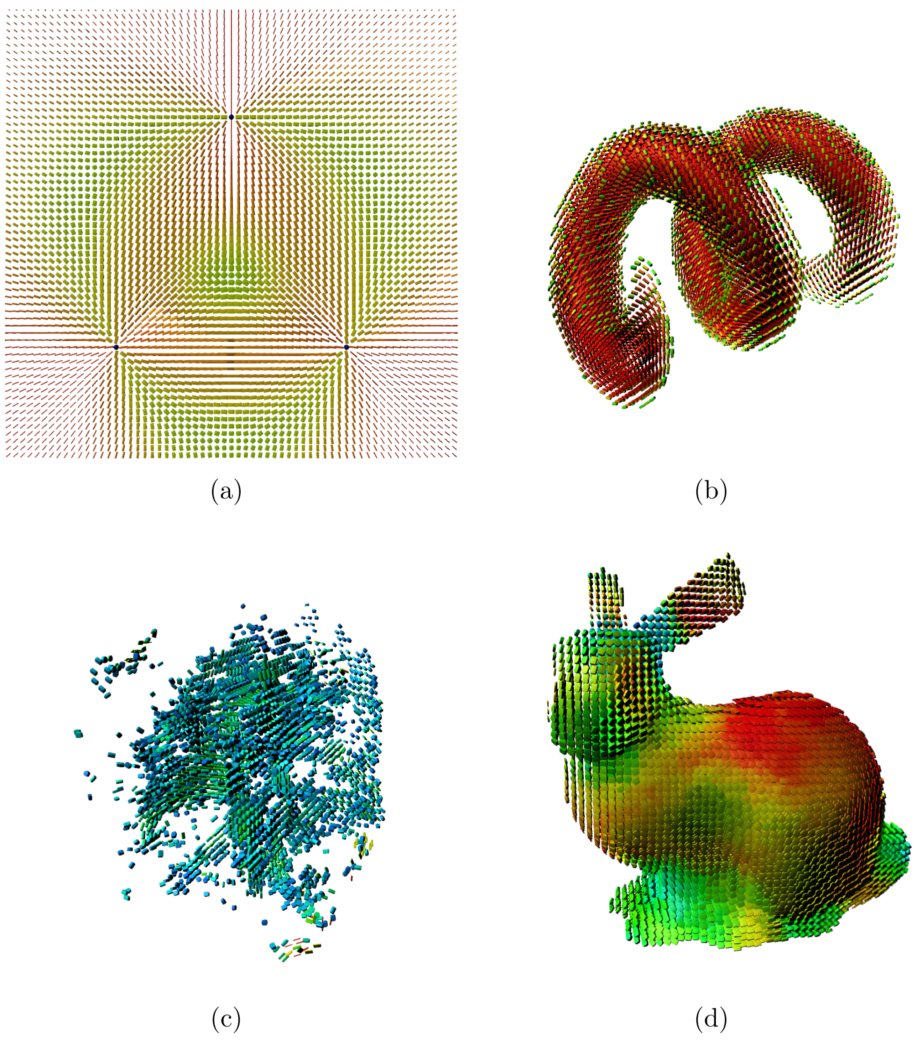


Figure 2.2: Examples of tensor fields: (a) 2D tensor field generated from 3 reference points; (b) a helical tensor field available at [1]; (c) a DT-MRI brain field; (d) a tensor field constructed for the Bunny geometry. Glyphs as defined in [2] were used for the representation of each individual tensor.

2.1.5 Anisotropy metrics

There are a wide array of metrics available in the literature for extracting relevant information about anisotropy in a tensor field. Some of these metrics consider individual tensors, while others use knowledge about some neighborhood of a tensor. In this section, a small subset of these metrics is presented. The anisotropy coefficients are the most basic ones. They are present in a multitude of fiber tracking techniques, since they encode essential information about each tensor in a field. Some other more complex metrics are also presented. These play a supporting role for the correct functioning of our method.

WESTIN *et al.* [4] proposes a set of geometric measures in order to determine the shapes described in the previous section. These measures are based on the eigenvalues of the diffusion tensor. They provide the ability of determining a tensor's degree of anisotropy. This anisotropy is mapped by the use of three coefficients: a linear, a planar and a spherical one.

The c_l coefficient measures linear anisotropy. These are tensors that may be geometrically interpreted as an elongated ellipsis, or a cylinder. This means that, if we have any kind of information flowing through a field composed mainly of this kind of tensor, this information will likely follow a line passing through these tensors. In certain regions of a field, however, the information will not follow a linear path. It may choose more than one direction instead, while still restricted to a nearly bidimensional diffusion. In this case, we have a planar tensor, which is identified by the c_p coefficient. Its geometrical interpretation is that of a disk. When the information is able to flow in any direction, with no restrictions at all, we then have a region composed mainly of isotropic tensors, or spherical tensors. These are represented by the c_s coefficient.

It is important to note there is no such thing as a purely linear, planar or spherical tensor in a field constructed from real acquired data. Each tensor has a predominant direction, which is determined by analyzing the values of each relationship. The more linear a tensor is, the higher will be its c_l value. The same applies to c_p and c_s. These three coefficients sum to 1, and they are calculated by using the eigenvalues extracted from the tensor matrix. By comparing the eigenvalues of a tensor, we can identify its shape. If the first eigenvalue λ_1 is much greater than the other two, we have a predominantely linear tensor. If λ_1 and λ_2 have approximate values and both are much greater than λ_3, we have a predominantely planar tensor. And if no significant distinction exist between the three eigenvalues, then the tensor is mostly isotropic. The mathematical definition of each coefficient is presented below:

$$\begin{aligned} c_l &= \frac{\lambda_1 - \lambda_2}{\lambda_1 + \lambda_2 + \lambda_3} \\ c_p &= \frac{2(\lambda_2 - \lambda_3)}{\lambda_1 + \lambda_2 + \lambda_3} \\ c_s &= \frac{3\lambda_3}{\lambda_1 + \lambda_2 + \lambda_3} \end{aligned}$$

Another metric commonly used in fiber tracking methods is the fractional anisotropy (FA). It describes variation between diffusion in different directions. The FA index is calculated by dividing the magnitude of the anisotropic part of a tensor by the magnitude of the full tensor. So, in order to present the mathematical expression for this index, we need to first obtain the expression for the anisotropic part.

Any tensor $\mathbf{T}$ can be decomposed into two separate tensors: an isotropic one and an anisotropic one [66]. The isotropic tensor is defined as the multiplication of the mean diffusivity of $\mathbf{T}$ by the identity tensor I. If we recall the already mentioned meaning of the eigenvalues in a diffusion context, we can define the mean diffusivity as:

$$D = \frac{Tr(\mathbf{T})}{3} = \frac{\lambda_1 + \lambda_2 + \lambda_3}{3} \tag{2.8}$$

The anisotropic part $\mathbf{T}_A$ of $\mathbf{T}$, given by $\mathbf{T}_A = \mathbf{T} - D\mathbf{I}$, is called the diffusion deviatoric or diffusion deviation tensor. This name derives from the fact that $\mathbf{T}_a$ measures how much $\mathbf{T}$ deviates from the isotropy. Using Equation 2.2, its magnitude can be defined as:

$$|\mathbf{T}_A| = \sqrt{\mathbf{T}_A : \mathbf{T}_A} = \sqrt{\sum_{i=1}^{3}\sum_{j=1}^{3}(T_{ij} - DI_{ij})^2} \tag{2.9}$$

By applying Equation 2.8 and taking symmetry into consideration, we can represent $\mathbf{T}_A$ in terms of $\mathbf{T}$ eigenvalues, as discussed in [66]. With a bit of additional algebraic manipulation, we obtain the following expression for the magnitude of $\mathbf{T}_A$:

$$|\mathbf{T}_A| = \sqrt{\mathbf{T}_A : \mathbf{T}_A} = \sqrt{\frac{1}{3}((\lambda_1 - \lambda_2)^2 + (\lambda_2 - \lambda_3)^2 + (\lambda_3 - \lambda_1)^2)} \tag{2.10}$$

With Equation 2.10 in hand, we can finally present the fractional anisotropy. This index measures how much of the magnitude of $\mathbf{T}$ is associated with anisotropic diffusion. The equation for FA, as introduced in [66], follows:

$$FA = \sqrt{\frac{3}{2}\frac{|\mathbf{T}_A|}{\mathbf{T}}} = \sqrt{\frac{1}{2}\left(\frac{(\lambda_1 - \lambda_2)^2 + (\lambda_2 - \lambda_3)^2 + (\lambda_3 - \lambda_1)^2}{\lambda_1^2 + \lambda_2^2 + \lambda_3^2}\right)} \tag{2.11}$$

Another similar index, called the relative anisotropy (RA), can also be defined using the magnitudes presented earlier. However, this index computes the ratio of the magnitudes of the anisotropic and isotropic parts, instead of using the full tensor magnitude. Thus, the RA index can be expressed as:

$$RA = \frac{|\mathbf{T}_A|}{|D\mathbf{I}|} = \frac{1}{3}\sqrt{\frac{(\lambda_1 - \lambda_2)^2 + (\lambda_2 - \lambda_3)^2 + (\lambda_3 - \lambda_1)^2}{D}}. \tag{2.12}$$

Both FA and RA indices are 0 for isotropic tensors. Their values increase according to the degree of anisotropy of the tensor. For cylindrical tensors, FA approaches 1. These two indices are called intravoxel measures, which means they map diffusion anisotropy by relying on the analyzed voxel alone. There are, however, other metrics that do not rely solely on a single tensor, but on a neighborhood instead, in order to detect diffusion patterns.

All metrics discussed so far used the three eigenvalues of $\mathbf{T}$ for the calculations, ignoring the eigenvectors in the process. This makes sense, since only the eigenvalues are related to diffusion anisotropy. However, one of the problems with the aforementioned metrics is that they can be innacurate in the presence of noise. This is a common problem for fields obtained from diffusion tensor imaging. Thus, PIERPAOLI e BASSER [67] propose using more information to improve anisotropy estimates. This is done by a different metric, called the lattice index (LI) , which makes use of the directional information contained in the tensor to minimize the influence of noise in the estimations. If a certain tensor is product of noise, its eigenvectors would be uncorrelated and there would be no orientational coherence between this tensor and its neighbors. So, this index is capable of measuring the level of collinearity between voxels. The lattice index can be mathematically defined as:

$$\mathrm{LI} = \frac{\sum_{n=1}^{8} a_n \mathrm{LI}_n}{\sum_{n=1}^{8} a_n}, \tag{2.13}$$

where n is the number of neighbors, a_n is a weighting factor and LI_n is the basic element of the index calculation, defined as follows:

$$\mathrm{LI}_n = \frac{\sqrt{3}}{\sqrt{8}} \left(\frac{\sqrt{\mathbf{T}_A : \mathbf{T}_{An}}}{\sqrt{\mathbf{T} : \mathbf{T}_n}} + \frac{\sqrt{\mathbf{T}_A : \mathbf{T}_{An}}\sqrt{\mathbf{T}_A : \mathbf{T}_{An}}}{\mathbf{T} : \mathbf{T}} \right). \tag{2.14}$$

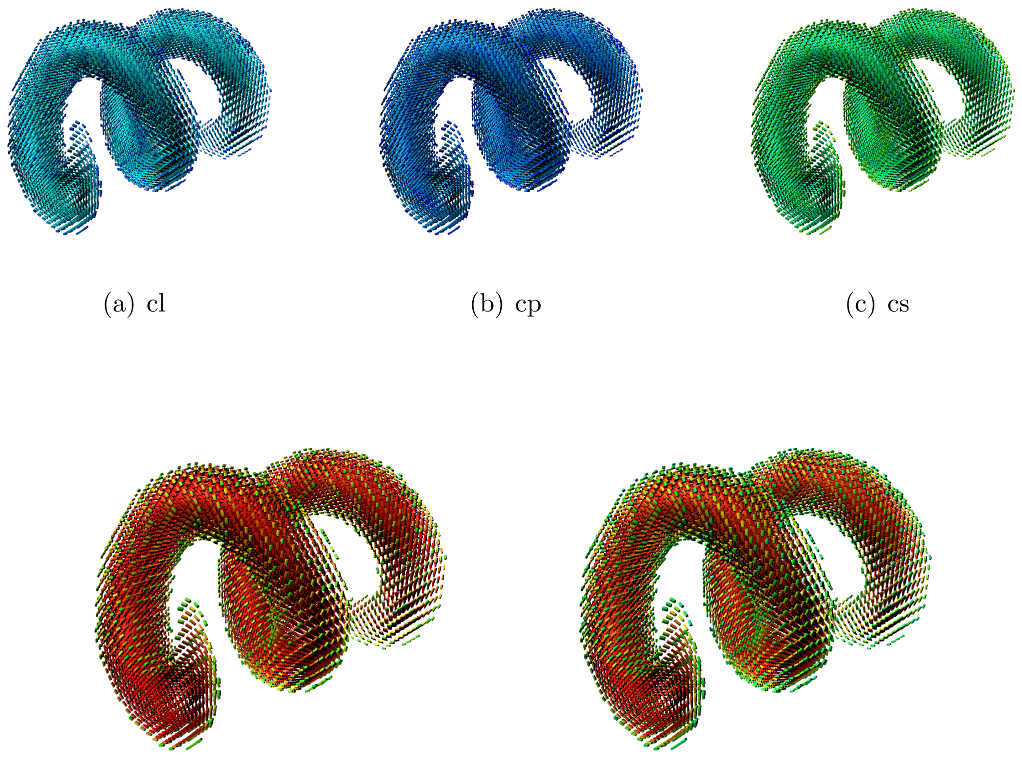
Figure 2.3 displays the helical field shaded according to the anisotropy measure used.

2.2 Fluid dynamics

In physics, studies involving fluids are carried out by a subarea called Fluid Dynamics. This area is interested in describing how fluids behave when they are subject to external forces, bringing them out of rest. In other words, there is no state of equilibrium in this case. These fluids are governed by the Navier-Stokes equations, which are a group of two partial differential equations that fully describe fluid motion. In this section, we will discuss basic fluid properties and definitions and also analyze how these equations map fluid flow.

2.2.1 Fluids basics

FERZIGER e PERIĆ [68] define a **fluid** as a substance with a molecular structure organized in a way that offers no resistance to external shearing forces. This means that, any shear force applied to the fluid, no matter how low its magnitude, will be able to deform the fluid. This is what gives them the ability of flowing, which allows the fluid (especially gases) to assume the available volume of its recipient. In the



(a) cl (b) cp (c) cs

(d) FA (e) LI

Figure 2.3: The helical tensor field with colors mapped according to different anisotropy measures. Colors vary from blue (lowest) to red (highest).

case of liquids, the fluid is also bounded by a so-called free surface, i.e., a surface created by the fluid itself that does not correspond to any solid boundary in the recipient. Gases have no such surface, reason why they are able to fully occupy a volume where it is contained. In this work, in all experiments we simulated the flow of smoke. Although smoke *per se* is not a fluid, we can interpret it as such, since smoke is just particles suspended in the air, which is the invisible flowing medium we will be simulating in fact.

When analyzing fluids, a common practice is to consider them as continuous substances, instead of a set of molecules. This is the basis of the Continuum Hypothesis. The purpose is to simplify the analysis. All fluid quantities are then defined inside a representative element of volume, usually referred to as REV. This is the smallest possible portion of space considered. In this volume, each quantity represents an average of the quantities of molecules contained in the region. This allows us to consider all fluid properties as continuous functions of time and space [69].

The main quantities we are interested in this work are **density** and **velocity**. These are fundamental to any fluid discussion. Other variables, like viscosity for instance, while also having an important role, may be ignored in specific scenarios. In fact, **inviscid fluids** are those with zero viscosity. In nature, practically no fluid can be considered truly inviscid, but for analysis and simulation purposes, it is perfectly plausible to assume the existence of such a fluid.

Density can be defined as the mass present in a volume element. Differently from solids, for which we can define the mass of a body, for fluids this mass is dispersed across the recipient volume. Assuming once again the REV volume, we can define a density value for each defined REV. Similarly, we can define for each REV a velocity value. Velocity is a quantity defined as a variation in space during a limited timeframe. If we imagine the REV as fluid particle, the velocity in a point in space (x, y, z) will be the instantaneous velocity of the particle passing through this point in a given time t. Both densities and velocities may vary in time. Thus, they are both functions of 4 variables in $\mathbb{R}^3$:

$$\rho = \rho(x, y, z, t), \tag{2.15}$$

$$\mathbf{u} = \mathbf{u}(x, y, z, t). \tag{2.16}$$

Fluids and their flow can also be categorized according to their characteristics. For our purposes, we will be dealing with **incompressible Newtonian fluids**. A Newtonian fluid has a deformation rate directly proportional to the applied stress. In this case, its viscosity is constant as long as there is no other influencing factor (apart from shear stress) affecting it, such as temperature. Non-Newtonian fluids, on the contrary, are the ones that do not follow Newton's viscosity law, which means that the applied stress can change the fluid viscosity.

It was said that an inviscid fluid is a fluid with no viscosity. The effect of viscosity on fluids can be determined by the Reynolds number:

$$Re = \frac{\rho V L}{\mu},$$

where μ is the viscosity, V is the flow speed and L the flow width. If this number is too high, the effect of viscosity is negligible, and thus the fluid may be considered as inviscid. As stated in [68], in most cases the effects of viscosity are more important near the boundaries, so it usually can be disregarded elsewhere.

Finally, a fluid is incompressible if density variations in the fluid may be neglected. While gases are usually compressible, they can also be considered incompressible when heat transfer is kept to a minimum and flow speed is little when compared to the velocity of sound, a relation defined by the number of Mach. Whenever this number is less than 0.3, the flow can be treated as incompressible. Besides, even for water, unless the fluid is subject to extreme conditions, the volume variation in fluids is usually very small. For fluid simulations that are more focused on the visual results than on physical accuracy, there would be not much gain in simulating compressible flow, as it is expensive and it does not visually affect significantly our simulations [8]. Thus, for our purposes, we will consider both liquids and gases as incompressible.

2.2.2 Navier-Stokes equations

The incompressible Navier-Stokes equations are a set of two equations that describe fluid flow and guarantees that it obeys basic conservation laws. They can be presented as:

$$\begin{cases} \frac{\partial \mathbf{u}}{\partial t} = -(\mathbf{u} \cdot \nabla)\mathbf{u} - \frac{1}{\rho}\nabla p + \nu \nabla^2 \mathbf{u} + \mathbf{f} \\ \nabla \cdot \mathbf{u} = 0 \end{cases} \tag{2.17}$$

The above presentation varies a little, according to the reference, but the general form is the same. In order to keep the objectivity of this work, we will not show derivation of these equations, but they can be found in [8] or in [69].

Consider a control volume Ω bounded by a control surface $\partial\Omega$. The analysis of fluid flow by using a control volume consists in selecting an arbitrary volume in space through which fluid is flowing. This is also called the Eulerian viewpoint. The alternative would be to consider the fluid as a collection of particles that carry the quantities by themselves, named the Lagrangian viewpoint. They are just different ways of looking at the fluid. Some things may be easier to analyze when choosing a viewpoint or another.

The second part of Eq. 2.17 is the incompressibility condition. It states that the amount of fluid that gets inside the control volume must be the same that gets out of it. In other words, the volume is not changing. As shown in [8], the measure of how much the volume is changing can be calculated by the following equation:

$$\frac{d}{dt}V(\Omega) = \int\int_{\partial\Omega} \mathbf{u} \cdot \hat{n},$$

where $V(\Omega)$ is the volume of Ω and $\hat{n}$ is the normal component of the velocity. For an incompressible fluid, the rate of change on the right side of the equation must be equal to zero. By applying the divergence theorem, we get to:

$$\int\int\int_{\Omega} \nabla \cdot \mathbf{u} = 0.$$

As this must hold to every region in the fluid, the integrand has to be zero everywhere, resulting in the condition $\nabla \cdot \mathbf{u} = 0$. This is also called the **continuity equation**. A vector field (in our case, the velocity field) that satisfies this condition is called **divergence-free**.

The first part of Eq. 2.17 is the one responsible for actually describing fluid motion. The equation can be broken down in separate blocks, to facilitate the analysis. The f term is the simplest, gathering all the external forces that act on the fluid. These may be the gravity, wind or artificial forces we use in our simulations. The pressure term $(-\frac{1}{\rho}\nabla p)$ plays important role in ensuring the incompressibility condition. The pressure can be understood as something that keeps the velocity divergence-free.

The first term of the equation is the main responsible for motion, since it describes the transport of quantities across the fluid. This transport is called **advection**. This term is intrinsically related to the material derivative, which is responsible for connecting the two viewpoints mentioned before: the Eulerian and the Lagrangian [8]. If we consider a quantity q carried by a particle (Lagrangian viewpoint), the function $q(t, \mathbf{x})$ defines the value of q at time t for a particle at position $\mathbf{x}$. However, $\mathbf{x}$ is an Eulerian variable, since it does not change with particles. To determine the rate of change of q for the particle in $\mathbf{x}$, we must take the total derivative of q, given by:

$$\frac{d}{dt}q(t, \mathbf{x}) = \frac{\partial q}{\partial t} + \nabla q \cdot \frac{d\mathbf{x}}{dt} = \frac{\partial q}{\partial t} + \nabla q \cdot \mathbf{u} = \frac{Dq}{Dt}. \qquad (2.18)$$

The first term indicates the variation of q at the fixed point $\mathbf{x}$, while the second term describes how much of q's variation is caused by the fluid that flows through that point. This material derivative is what describes advection in the Navier-Stokes equation. The **advection equation**, as pointed out in [8], is an equation that uses the material derivative by setting it to zero. This means that the q variation at $\mathbf{x}$ summed with its variation due to fluid flowing by $\mathbf{x}$ is equal to zero. In other words, there is movement happening at $\mathbf{x}$, but the quantity is not in fact changing, if we look at it from the Lagrangian viewpoint. So, the advection equation can be stated as:

$$\frac{Dq}{Dt} = \frac{\partial q}{\partial t} + \nabla q \cdot \mathbf{u} = 0. \qquad (2.19)$$

Notice that Eq.2.19 describes the advection in terms of a generic quantity q. Naturally, any fluid-related quantities may be advected (for instance, density or temperature). We just have to substitute q with the appropriate variable. As we are not considering temperature in this thesis, we only advect densities and velocities. The velocity advection is also called **self-advection**. It means the velocity is advected through the fluid by itself. The advection equations for velocities and densities are shown below:

$$\frac{d\mathbf{u}}{dt} + \mathbf{u} \cdot \nabla \mathbf{u} = 0, \tag{2.20}$$

$$\frac{d\rho}{dt} + \mathbf{u} \cdot \nabla \rho = 0. \tag{2.21}$$

The notation for the advection used in the Navier-Stokes equations is just a simplification of the used for the material derivative equations. The advective term can be represented in two ways: $\mathbf{u} \cdot (\nabla \mathbf{u})$, which involves a dyadic tensor, or $(\mathbf{u} \cdot \nabla)\mathbf{u}$, which is simpler and usually preferred in fluid dynamics references [70].

The only contribution left is the viscosity. The Laplacian term $\nu \nabla^2 \mathbf{u}$ is responsible for adding the viscous effect to the velocity. The constant ν is the kinematic viscosity, defined as μ/ρ. Viscosity acts as a diffusive term, dissipating energy in the fluid. The larger the viscosity is, the quicker velocities will dissipate. For quantities such as density, this term in the Navier-Stokes equation represents a diffusion term, where the quantity gradually dissipates to its neighbors.

As said before, the viscosity term may be suppressed if we are not interested in simulating it. In this case, Eq. 2.17 reduces to:

$$\frac{\partial \mathbf{u}}{\partial t} = -(\mathbf{u} \cdot \nabla)\mathbf{u} - \frac{1}{\rho}\nabla p + \mathbf{f}. \tag{2.22}$$

Together with the incompressibility condition, this forms the **Euler equations**, which is just the name given to the inviscid Navier-Stokes equations.

2.2.3 Boundary conditions

An important part of fluid simulation is to determine what happens to the fluid at the boundaries of the volume. The way these boundary conditions are set influence the general behavior of the fluid. We will not consider in this work free surfaces or boundaries between two different fluids, since they do not apply to our context. Since we are focusing on simulating smoke, the only boundary conditions that matter are the ones between the fluid and the control volume.

A possibility would be to consider the fluid as if it were wrapping around itself, i.e., leaving the volume in a boundary and re-entering it through the opposite boundary. Another option is to fix the velocity on parts of the boundary, simulating a constant inflow, as in a wind tunnel [71]. Instead, we use a solid wall boundary, which forbids any fluid from leaving or entering the volume. This is also called a no-stick condition, since it restricts the normal component of the velocity. As a result, the fluid is able to move freely in the tangential direction. This does not necessarily relate to the velocity of the boundary. If the boundary is also moving, the normal component of the fluid velocity must be equal to that of the solid. In our case, we consider a stationary boundary. Thus, the boundary condition reduces to:

$$\mathbf{u} \cdot \hat{n} = 0$$

2.3 Stable fluid simulation

In the last chapter, we presented the mathematical formulation behind the motion of fluids. For simulating fluids, we must solve the Navier-Stokes equations, presented in Section 2.2.2, to obtain the values of velocities and densities for each point in the volume. However, these equations are difficult to solve analytically. Besides, fluid dynamics is usually very complex, so that simulation experiments tend to be computationally expensive and hard to control.

To properly simulate fluid flow, the Navier-Stokes must be discretized and methods for numerically solving them must be employed. The Computational Fluid Dynamics field focuses on providing solutions to problems such as this. As we mentioned in the introduction, this thesis is not concerned with strict physical accuracy, so that we resort to approaches focused on computer graphics applications. Specifically, we based our method on the Stable Fluids method, proposed by STAM [5] and also discussed in [8]. The bulk of our basic fluid simulator is based on an implementation of the Stable Fluids method presented in [71].

In this chapter, we will discuss the fluid data representation and the methods used to solve for each term of the Navier-Stokes equations. This will serve as the basis for the proposed method of this thesis.

2.3.1 Data representation

In order to simulate fluid flow, we need to decide on what kind of representation to use for storing fluid data. The chosen representation directly affects the choice of the numerical method used for solving the equations. Two forms of representation are commonly used, each one of them following one of the viewpoint approaches for analyzing fluid flow mentioned in Chapter 2.2.

The Lagrangian approach use the idea of observing moving particles to represent the data, storing velocity and density values in each particle. Although not very practical to use in a physics context, in a computational simulation this kind of approach becomes feasible, since it is much easier to manage a large number of particles computationally than it is to do it in traditional physics experiments. This approach is used in the well-known Smoothed Particle Hydrodynamics method, usually abbreviated as SPH. This is an interpolation method that was first developed

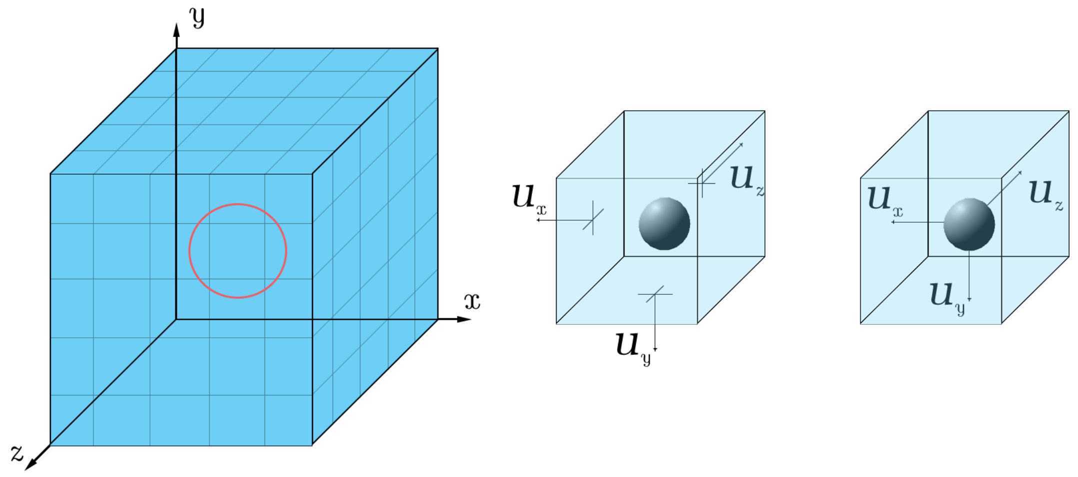


Figure 2.4: The difference between storing velocity values at the center of a cell and at its faces.

to simulate astrophysical phenomena, being later incorporated by the computer graphics community. Since this is out of the scope of this work, we will not delve into details, referring the interested reader to the work by DA SILVA NETO *et al.* [72], which provides a good introduction to the subject. For a more thorough covering of works using the SPH method, there is a recent survey by WEAVER e XIAO [73].

The Eulerian approach follows the idea of analyzing a fixed region in space through which fluid can flow. Numerically speaking, we discretize the space into a grid and store the relevant information in each cell. The quantities, as in the control volume analysis described in Section 2.2.2, are averaged values representing that quantity in that region of space limited by the cell boundaries. Thus, we are looking at the fluid state in that whole region, and not in a sole particle.

Since we want to use the tensor field as an inherent part of the medium, we opted for using the Eulerian approach. In this way, the tensor is stored along with the fluid quantities, like velocity and density. Differently from those quantities, the tensor is a function of space alone, while velocity and density are functions of space and time.

The obvious configuration is the storage of values in the center of each cell. However, in some situations this may lead to a special kind of instability called the checkerboard instability, due to how derivatives are calculated when velocities are represented at the center of cells. A common solution to this issue is to use a staggered arrangement, also called a MAC (marker-and-cell) grid, where velocities are split into their components, and each component is stored in a cell face. Figure 2.4 shows the differences between both representations.

2.3.2 Numerical solution

We mentioned that the Navier-Stokes equations have a difficult analytical solution. One way of tackling this problem is by dividing it into smaller manageable steps. This is the principle behind the operator splitting technique, which may be used to solve partial differential equations composed by a sum of different terms. Simply put, it consists of solving each term in the equation with a proper numerical method, incrementally summing their contributions along the way.

Consider an operator $\mathcal{L}$ applied to $\mathbf{u}$ in the following partial differential equation:

$$\frac{\partial \mathbf{u}}{\partial t} = \mathcal{L}\mathbf{u}.$$

If $\mathcal{L}$ can be written as a linear sum of m terms, where each term contributes to the value of u, we can use different methods for solving each one of them and account for that in a series of update steps [74].

$$\begin{aligned} \mathbf{u}^{n+(1/m)} &= \mathcal{M}_1(u^n, \Delta t) \\ \mathbf{u}^{n+(2/m)} &= \mathcal{M}_2(u^{n+(1/m)}, \Delta t) \\ &\cdots \\ \mathbf{u}^{n+1} &= \mathcal{M}_m(u^{n+[(m-1)/m}, \Delta t) \end{aligned}$$

Here we follow the sequence of steps employed in [5]. The first step in the simulation process consists in the application of the external forces. Any kind of force can be applied in this step. For example, wind-like forces would be added here. This opens the possibility for the addition of artificial forces designed with specific purposes in mind. For instance, the vorticity confinement technique [75] consists in the application of an artificial force that reintroduces in the simulation swirling details lost due to numerical dissipation. Similarly, in [18] a force based on a distance field is used to direct the fluid towards a target shape.

The next step is the diffusion, responsible for accounting for the viscosity effects. As we are simulating smoke, the diffusive term can also be applied to the densities, smoothing out them according to a diffusion constant. For velocities, the viscosity works in a similar way, dampening them quickly, significantly reducing fluid motion, as it should happen in a viscous fluid. This is a completely optional step, only making sense for fluids that are not inviscid, or if density diffusion for smoke is not desired.

The velocity field produced by the previous step is then taken as input for the advection step. In this step, we employ an unconditionally stable method presented in [5], which guarantees that no output velocity is bigger than the maximum velocity in the input field. This feature prevents the simulation from blowing off. The process is simple: we trace the advected property back in time to find what was its value in its previous position in the grid. Then, we use that value interpolated with its neighbors to update the current value of the property. This will be discussed in detail in Section 2.3.3.

An important part of the simulation is the projection. The projective step should be performed both after the advection and the diffusion. This step enforces the incompressibility condition, by taking the resulting velocity field from the previous step and making it divergence-free. This is done by calculating the pressure gradient and subtracting it from the velocity. Projection will be discussed in Section 2.3.5.

2.3.3 Advection

The advective term is the one responsible for transporting quantities along the fluid. Sometimes called convection, or even transport, the advection carries substances and other values (like temperature, for instance) through the velocity field. As we mentioned in the previous chapter, the velocities are also advected in a process called self-advection. This is how we update velocity values during simulation.

The most straightforward way of updating the velocities would be to calculate them by using a finite differences scheme, immediately obtaining the next velocity value. However, this is highly unstable, since velocities may grow excessively at each iteration if the time step used is too large, causing the simulation to eventually blow up. So, STAM [5] proposed using an alternate approach, which is unconditionally stable, independently of how large the time step is. Imagine a fluid particle p in position $\mathbf{x}_G$, where the new velocity $\mathbf{u}_G^{t+1}$ must be calculated. The idea is to trace the particle's movement backward in time to determine its original position $\mathbf{x}_P$ in time t. In other words, the question is: which velocity was responsible for moving the particle from its previous position to its current position? This velocity is used to update the current velocity. The logic behind it is that the advection transported this value from position $\mathbf{x}_P$ to current position $\mathbf{x}_G$. So, essentially, $\mathbf{u}_G^{t+1}$ at $\mathbf{x}_G$ is the same value as $\mathbf{u}_P^t$ at $\mathbf{x}_P$. Mathematically, the backward transport is calculated as follows:

$$\mathbf{x}_P = \mathbf{x}_G - \Delta t \mathbf{u}_G^t. \tag{2.23}$$

The problem is that $\mathbf{x}_P$ may not be an exact grid cell center. Thus, we need to interpolate between its neighboring cells in order to obtain the velocity value that will be used to update the cell at $\mathbf{x}_G$. So, the final velocity is defined as:

$$\mathbf{u}_G^{t+1} = \text{interp}(\mathbf{u}^t, \mathbf{x}_P), \tag{2.24}$$

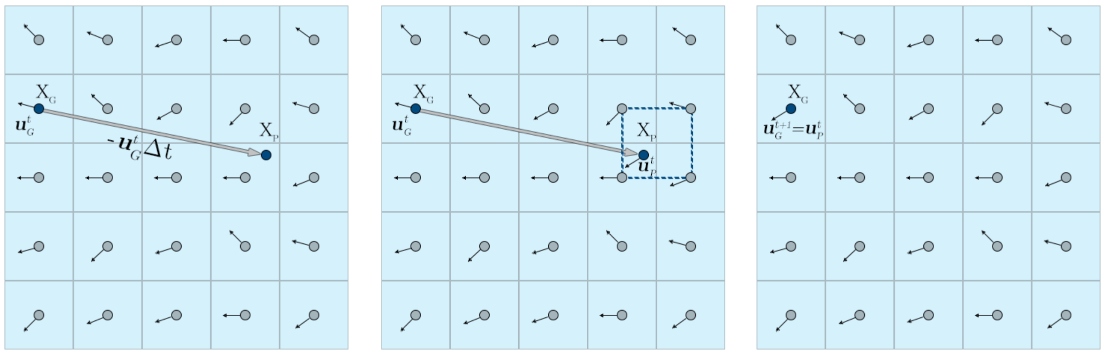


Figure 2.5: Semi-Lagrangian advection process: a particle is traced backwards in time and the new value is obtained by interpolating from the neighboring cells at the particle's old position.

where interp$(\mathbf{u}^t, \mathbf{x}_P)$ is an interpolation function between neighbors of $\mathbf{x}_P$.

The same procedure is used for other fluid quantities. If we want to calculate the new density value, for instance, we need to trace a particle back in time through the velocity field using Eq. 2.23 and interpolate the densities at $\mathbf{x}_P$. Because we use the concept of a particle to update velocities in the grid, this process is called a semi-Lagrangian advection, i.e., we are using the Lagrangian viewpoint within an Eulerian context. Figure 2.5 illustrates the process.

This procedure is explained by the so-called method of characteristics. Consider the advection equation presented in Eq. 2.19 rewritten as below:

$$\frac{\partial q(\mathbf{x}, t)}{\partial t} = -\mathbf{u}(\mathbf{x}) \cdot \nabla q(\mathbf{x}, t),$$

where $\mathbf{u}(\mathbf{x})$ is a steady vector field. Consider also that $q(\mathbf{x}, 0) = q_0(\mathbf{x})$, with q_0 being the scalar field q at time $t = 0$. The method of characteristics is capable of solving equations of this form. In order to do this, let us consider $\mathbf{p}(\mathbf{x}_0, t)$ as the characteristics of the vector field $\mathbf{u}$ flowing through $\mathbf{x}_0$ at $t = 0$:

$$\frac{d}{dt}\mathbf{p}(\mathbf{x}_0, t) = \mathbf{u}(\mathbf{p}(\mathbf{x}_0, t)),\ \mathbf{p}(\mathbf{x}_0, 0) = \mathbf{x}_0.$$

Finally, consider $\overline{q}(\mathbf{x}_0, t) = q(\mathbf{p}(\mathbf{x}_0, t), t)$, i.e., $\overline{q}$ is the field value along characteristic $\mathbf{p}(\mathbf{x}_0, t)$. We can calculate the variation of $\overline{q}$ as:

$$\frac{d\overline{q}}{dt} = \frac{\partial q}{\partial t} + \mathbf{u} \cdot \nabla q = 0.$$

This means that q is not varying over time along the streamlines. So, we have:

$$\overline{q}(\mathbf{x}, t) = \overline{q}(\mathbf{x}, 0) = q_0(\mathbf{x}_0).$$

This guarantees that we can trace a particle back in time to find the initial point $\mathbf{x}_0$ ($\mathbf{x}_P$ in the semi-Lagrangian advection) and use its value to update the current position [5].

2.3.4 Diffusion

Diffusion is an alternative way of moving substances through the fluid. It is intrinsically different from the advection, though. While in the advection fluid quantities are moved through the velocity field, here the quantities are isotropically dispersed through time, from regions of higher concentration to regions with lower concentration.

In nature, the diffusion process is essentially random. It is related to Brownian motion, which is the random motion performed by independent particles in a fluid. The Brownian motion is the effect of collisions between molecules of the fluid, resulting in a motion that is fundamentally random. If a particle is observed during a specific time interval, the average displacement by the particle is 0, meaning the particle will not follow any specific direction. So, diffusion is nothing more than a large number of particles following the rules of Brownian motion over a timespan.

In order to solve the diffusion term during the simulation, STAM [71] proposes solving it implicitly. The problem with directly solving it, by computing the exchanges between neighboring cells and updating the central cell with the new density value, is that it is extremely unstable. Once a large diffusion rate is set, density values tend to oscillate and eventually diverge. Thus, following similar rationale used for the advection, the diffusion solver finds the previous densities that generated the current ones, by travelling back in time. This results in a linear system where the density values are unknowns. This system can be easily solved by using the iterative Gauss-Seidel method. This solution is stable, no matter how large the diffusion rate chosen. The implicit equation being solved is the following:

$$(\mathbf{I} - \nu \Delta t \nabla^2)\mathbf{u}'(x) = \mathbf{u}(x), \tag{2.25}$$

where $\mathbf{I}$ is the identity operator, $\mathbf{u}'(x)$ is the diffused velocity field and $\mathbf{u}(x)$ is the field received from the previous simulation step. Naturally, the diffusion term in the equations can be applied both for velocities and for densities. The difference lies in the coefficient. While for velocities this coefficient is the kinematic viscosity ν, for densities it means a diffusion rate.

2.3.5 Projection

An essential part of the simulation is the projection step. It must be performed to ensure velocities are kept divergence-free. Performing the advection on a velocity field that is not divergence-free can lead to problems. So, two projection steps occur in each iteration: one after the advection and another after the diffusion. This is how we enforce the incompressibility condition on our simulation.

The first step is to calculate pressure by plugging the velocity field obtained from the previous step (whether it be the advection or the diffusion) into the following Poisson equation [7]:

$$\nabla^2 p = \frac{1}{\Delta t} \nabla \cdot \mathbf{u}. \tag{2.26}$$

The solution to this equation is obtained by computing the velocity divergence and solving a linear system similar to the one used for diffusion, where the pressure values are the unknowns.

Next, the pressure gradient is calculated and subtracted from the velocity. This results in our divergence-free velocity field. The principle behind this procedure is the Helmholtz-Hodge decomposition, which states that any vector field can be decomposed into a sum between a divergence-free vector field and a scalar field. So, by using our original velocity field and subtracting from it our pressure field (a scalar field), we guarantee that our velocities obey the incompressibility condition in Eq. 2.17.

Chapter 3

Proposed method

The purpose of this work is to adapt the mathematical model and the numerical simulation of fluids to account for the information obtained from tensor fields. Here we are dealing specifically with local orientation tensors, introduced in Section 2.1.3. This kind of tensor evokes a geometric interpretation that can be associated with direction of fluid diffusion. The probabilistic nature of the tensor allows us to influence fluid flow, deflecting it along paths of interest in a tensor field. This could be useful for DT-MRI, where tensors encode water diffusion inside organic tissue. Usually, these tensor fields suffer with information cluttering, making it difficult for physicians to visualize and interpret structures of interest. Adding motion (fluid motion, in our case) to the visualization process helps drawing the attention of the observer to relevant regions in the field. Porous media flow could also, in theory, be used for enhancing visualization, since it is intrinsically related to the concept of permeability, a tensor quantity, as briefly discussed in Section 1.1.2. The permeability tensor is of the exact same form as the tensors obtained from DT-MRI. So, it would make sense to treat diffusion tensor fields as permeability fields, and use the idea of fluid flowing through pores to add motion to the tensor field and stimulate human perception in the process.

The example applications above are specific, but tensor fields could be used in a more general way. Since they indicate diffusion direction, we can use them to intentionally move the fluid along specific paths. This kind of fluid control can be useful, for example, in computer animations. Thus, we argue that if we simulate and visualize fluid flow influenced by a tensor field, we can analyze fluid behavior to not only emphasize colinear and coplanar structures in DT-MRI, but also to create fluid-like animations, given a specific tensor field designed for that purpose. The challenge lies in determining the best way to add tensor information to the simulation.

We based ourselves on the classic method proposed by STAM [5] to produce an adapted method. We use an Eulerian representation of the fluid, aligning the grid

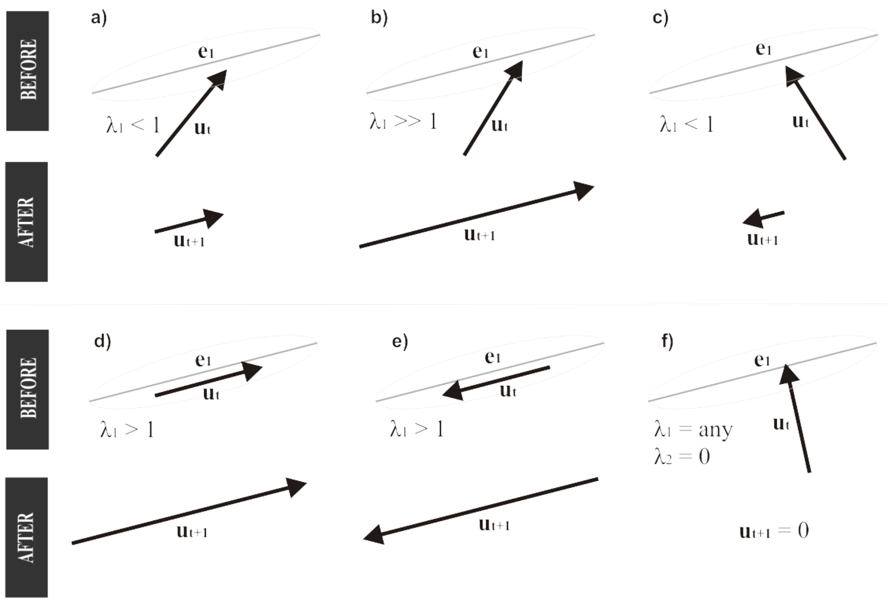


Figure 3.1: Six cases of a velocity vector being transformed by a tensor. Each case shows a different situation of tensor-vector alignment, producing deflected vectors that may be scaled or shrinked, according to the tensor eigenvalues.

cells with the field, so that each tensor is located at the center of its corresponding cell. We work on the idea of the tensor acting as something that deflects velocities passing through its cell, locally reducing or amplifying momentum in the process. For example, fluid arriving at a cell with a rank-1 tensor oriented perpendicularly to its velocity would have its momentum locally reduced to zero. Depending on the degree of alignment between tensor and velocities, the tensor could be physically interpreted as a pump (alignment to eigenvectors with high eigenvalues, for instance) or a sink (velocities perpendicular to main eigenvector with low eigenvalue). Figure 3.1 shows some examples.

In order for the simulation to be successful, it is desirable that the tensor field be continuous everywhere. Although in certain cases good results can still be obtained even in the presence of discontinuities, the method is not prepared for these situations yet. The presence of null tensors could cripple the simulation, especially regarding the projection. The problems with discontinuous fields in the projection will be discussed in detail in Section 3.4.

In [27], tensors were used to directly transform velocities during the interpolation step of the semi-Lagrangian advection, forcing velocity vectors to abide by the tensor field characteristics. While it could definitely produce good visual results, that was a rather arbitrary approach. It would be interesting to produce the same effect using a more physically-motivated formulation leading to a certain level of numerical stability. Thus, we propose to use the tensor as a local momentum modulator for the fluid as it moves along the field. So, we begin by introducing the following

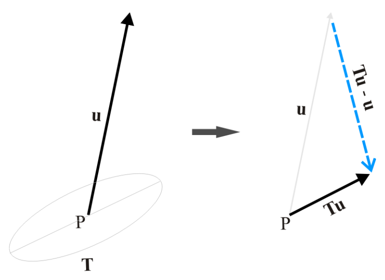


Figure 3.2: Geometrical interpretation of Equation 3.2.

differential equation:

$$\frac{\partial \mathbf{u}}{\partial t} = \mathbf{K}\mathbf{u}, \tag{3.1}$$

where $\mathbf{u}$ is measured in m/s and $\mathbf{K}$ is a matrix measured in s^{-1}. We propose $\mathbf{K} = \mathbf{T} - b\mathbf{I}$, where b is a scaling factor, to transform the velocity $\mathbf{u}$ in the following form:

$$\frac{\partial \mathbf{u}}{\partial t} = (\mathbf{T} - b\mathbf{I})\mathbf{u},$$

which, setting $b = 1$ and putting physical units aside, can be viewed as:

$$\frac{\partial \mathbf{u}}{\partial t} = \mathbf{T}\mathbf{u} - \mathbf{u}. \tag{3.2}$$

This is the version we will use for the remainder of this text, always considering the units specified in Eq. 3.1: $[\mathbf{T}] = \mathrm{s}^{-1}, b = \mathrm{s}^{-1}, [\mathbf{u}] = \mathrm{m\ s}^{-1}$. The dimensional equation for Eq. 3.2 can be written as $[L^1T^{-2}] = [T^{-1}][L^1T^{-1}] - [T^{-1}][L^1T^{-1}]$. We include this last equation in the original Navier-Stokes formulation, as an additional equation to the system. This equation works as a restriction on the velocity. The main insight behind this restriction is that a tensor should decelerate velocities that are not aligned with its main eigenvector, while also bending them into alignment. So, we determine that the acceleration of the fluid must obey Eq. 3.2. If we look at Fig. 3.2, we see that the transformation of a velocity vector $\mathbf{u}$ by an orientation tensor $\mathbf{T}$ results in a new velocity $\mathbf{T}\mathbf{u}$, which is aligned with the orientation of $\mathbf{T}$ and has a norm influenced by the eigenvalues of $\mathbf{T}$. The vector $\mathbf{T}\mathbf{u} - \mathbf{u}$, which takes $\mathbf{u}$ to $\mathbf{T}\mathbf{u}$, can be interpreted as an acceleration vector. Depending on the eigenvalues of $\mathbf{T}$ and on the angle between $\mathbf{u}$ and the main eigenvector $\mathbf{e}_1$ of $\mathbf{T}$, this vector could locally accelerate the fluid in the direction of $\mathbf{e}_1$ or it could decelerate it, reducing fluid momentum in the nearby region. This physical interpretation of the tensor effect on the fluid is much more interesting than the approach adopted in [27].

Eq. 3.2 is a homogeneous ordinary differential equation with constant coefficients in time, i.e., $\partial T/\partial t = 0$. It governs fluid acceleration when coupled with the Navier-Stokes equations. But how do we enforce this restriction on the system? There are different ways to achieve that, and it depends on how we interpret the tensor role in the system. One could think of adding the restriction to the system as an external force, multiplying the resulting acceleration by the fluid density. This would be described by the equation below:

$$\frac{\partial \mathbf{u}}{\partial t} = -\mathbf{u} \cdot \nabla t + \nu \nabla^2 \mathbf{u} + \mathbf{T}\mathbf{u} - \mathbf{u}. \tag{3.3}$$

The main advantage of this approach is that it can be directly applied to the Navier-Stokes equations. However, there are a number of disadvantages: it requires very small time steps to remain stable and fluid control is only exerted during instantaneous force applications. So, we can say that using Equation 3.2 as a force does not establish a strong link between the tensor field and the fluid. If it were applied as a force, the tensor effect would cease as soon as the force stopped acting, since the advection would be able to transport quantities without restrictions. However, if we take as a premise that the tensor field is an inherent part of the environment, thinking of it as a property of the medium, the tensor becomes an integral part of fluid motion. The fact that tensor fields are kept static during our simulations also favors the adoption of this point of view. By choosing this approach, what we are saying is that all aspects of the fluid dynamics are bound to the tensor field characteristics. This places a strong bias on fluid motion, making it likely to follow a predefined path. The drawback of this approach is that care must be taken on how to insert the tensor information into the equations, since they govern fluid motion in a coupled way.

A good way to do this is by adding this as an additional equation to the Navier-Stokes system of equations. We also customized some terms of Eq. 2.17 in order to insert the tensor information in each step of the numerical simulation. This is the core of our proposal, which we will discuss in the following sections.

3.1 Proposed continuous formulation

In order to consider the tensor as a property of the medium through which the fluid flows, we need to adapt the Navier-Stokes equations to account for the tensor information. We propose the usage of the following customized system of equations for describing the dynamics of fluids. Notice the addition of our Eq. 3.2:

$$\begin{cases} \frac{\partial \mathbf{u}}{\partial t} = -(\mathbf{u} \cdot \nabla)\mathbf{u} - \frac{1}{\rho}\nabla p + \nabla \cdot (\mathbf{T}\nabla \mathbf{u}) + \mathbf{f} \\ \frac{\partial \mathbf{u}}{\partial t} = \mathbf{T}\mathbf{u} - \mathbf{u} \qquad \text{(advective-only)} \\ \nabla \cdot \mathbf{u} = 0 \\ \frac{\partial \rho}{\partial t} = -(\mathbf{u} \cdot \nabla)\rho + \nabla \cdot (\mathbf{T}\nabla \rho) - \alpha\rho + S \\ \mathbf{u}(0) = \mathbf{u}_0 \end{cases} \tag{3.4}$$

The system of equations above accounts for both the velocity and the density update. The acceleration equation presented in the previous section enters here as a purely advective equation, working as a restriction to fluid motion. Since the advection term is the one responsible for the main transport of the fluid, it is sensible that we add to it this restriction. Eq. 3.2 should, considering the underlying tensor field, discourage fluid motion along certain regions and deflect the fluid towards intended paths. We do this by adapting the numerical method that solves the advection part. This will be addressed in Section 3.2.

The other term affected by the tensors is the diffusive term, both for velocities and for densities. The tensor is introduced in the equation by multiplying the gradient of the fluid quantity being diffused. This results in a diffusion which is no longer isotropic. Thus, the viscosity ν (for velocities) and the diffusion rate κ (for densities) are now replaced by the tensor $\mathbf{T}$. It is the tensor who determines the degree of diffusiveness applied in each direction for the quantity being evaluated. This is a classic formulation for anisotropic diffusion. Our contribution is related to how we handle this term, by precomputing Laplacian masks derived from the tensor information. Section 3.3 discusses the details of this part of the proposal.

Finally, in the density equation, we added a source term S, for inserting fluid in the system, and a dissipation term $-\alpha\rho$, as used in [5], to avoid the excessive accumulation of densities outside the regions of interest of the tensor field. Since the tensor advection tends to reduce motion in these regions, densities who eventually fall outside the controlled paths will hardly be able to return to the tensor-controlled regions. This could be reduced by using smaller time steps, but the dissipation function provides a good alternative instead.

3.2 Customized advection

The advection part is the core of our proposal. This is the main responsible for controlling the fluid by the tensor field. Although the anisotropic diffusion can also aid on this matter, it is just a secondary process. The advection is the principal means of transport in a fluid. So, we will present two approaches for the problem of introducing tensors to the advection.

First, let us consider again Eq. 3.2, coupled with the condition $\mathbf{u}(0) = \mathbf{u}_0$ included in the system of equations above. If we put it in the form $\mathbf{u}' + \mathbf{A}\mathbf{u} = 0$, which is a homogeneous equation with constant coefficients, we obtain:

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{I} - \mathbf{T})\mathbf{u} = 0.$$

Thus, $\mathbf{A} = \mathbf{I} - \mathbf{T}$. Assuming $\mathbf{A}$ is a diagonalizable matrix, then there is $\mathbf{D}$ such that $\mathbf{D} = \text{diag}(\lambda_1, \lambda_2, \cdots, \lambda_n)$, the diagonal matrix of A's eigenvalues, and there is a matrix $\mathbf{P} = [\mathbf{e}_1, \mathbf{e}_2, \cdots, \mathbf{e}_n]$ such that $\mathbf{P}$ is invertible and $\mathbf{A} = \mathbf{PDP}^{-1}$.

Let us define $e^{-t\mathbf{D}} = \text{diag}(e^{-t\lambda_1}, \cdots, e^{-t\lambda_n})$ and $e^{-t\mathbf{A}} = \mathbf{P}e^{-t\mathbf{D}}\mathbf{P}^{-1}$. Then we have:

$$\mathbf{u}(t) = e^{-t\mathbf{A}}\mathbf{u}_0.$$

So, the solution for Eq. 3.2 is:

$$\mathbf{u}(t) = e^{-t(\mathbf{I}-\mathbf{T})}\mathbf{u}_0. \tag{3.5}$$

We now have two options: using this equation directly or implement a numerical solution. We will begin the discussion with the numerical one. This solution involves obtaining a discretized version of Eq. 3.2, reusing the operator splitting technique presented before. Afterwards, we will show the analytical solution, which uses Eq. 3.2.

3.2.1 Numerical solution

In Chapter 2.2 we explained how the technique called operator splitting works. Basically, it allows us to separately calculate the contribution of each term in a partial differential equation. We followed the same procedure to discretize and solve Eq. 3.2. Since it is composed of two terms, we can solve it in two steps:

$$\tilde{\mathbf{u}} = \mathbf{u}^t + \Delta t \mathbf{T}\mathbf{u}^t,$$
$$\mathbf{u}^{t+1} = \tilde{\mathbf{u}} - \Delta t \tilde{\mathbf{u}}.$$

where t indicates a time instant, or an iteration. By plugging the first equation into the second, we obtain the following expression for $\mathbf{u}^{n+1}$:

$$\mathbf{u}^{t+1} = \mathbf{u}^t + \Delta t \mathbf{T}\mathbf{u}^t - \Delta t(\mathbf{u}^t + \Delta t \mathbf{T}\mathbf{u}^t),$$

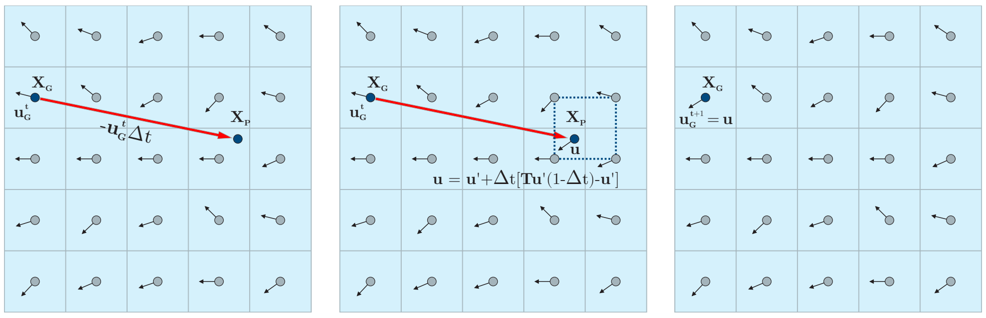


Figure 3.3: Illustration of the customized advection process, using Eq. 3.9.

Rearranging it, we get to the final update equation:

$$\mathbf{u}^{t+1} = \mathbf{u}^t + \Delta t[\mathbf{T}\mathbf{u}^t(1 - \Delta t) - \mathbf{u}^t], \tag{3.6}$$

which can be directly plugged into the advection step. Equation 3.6 defines how velocities are transported between time t and $t+1$. So, we want to evaluate this equation whenever velocities suffer any variation during the advection. The original semi-Lagrangian advection proposed by [5] consisted of a particle in cell location $\mathbf{x}_G$ propagated backwards in search of its previous position $\mathbf{x}_P$. The velocity in $\mathbf{x}_P$ was then interpolated with those at its neighbors, and this new value was used to update the velocity in $\mathbf{x}_G$. In our setting, we use Eq. 3.6 whenever velocities are calculated during the advection process, affecting both the backtraced position $\mathbf{x}_P$ and the final velocity value used to update $\mathbf{x}_G$. This process is illustrated in Fig. 3.3. So, we have:

$$\mathbf{x}_P^t = \mathbf{x}_G^t - \Delta t(\mathbf{u}_G^t + \Delta t[\mathbf{T}\mathbf{u}_G^t(1 - \Delta t) - \mathbf{u}_G^t]). \tag{3.7}$$

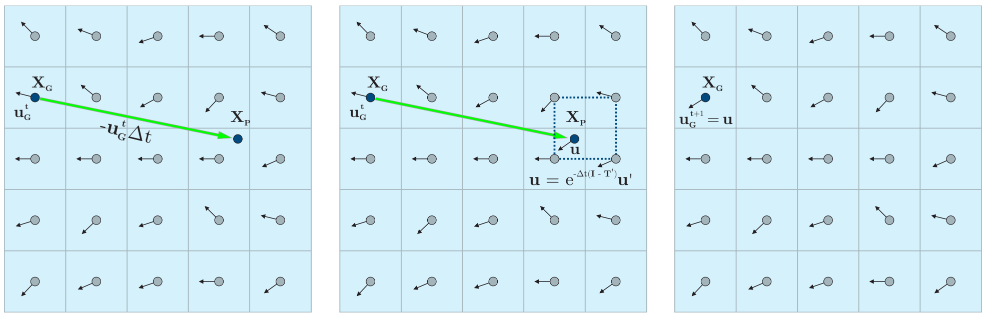
Position $\mathbf{x}_P$ may not fall exactly at the center of a cell. Thus, we interpolate the values from its neighbors. We perform this interpolation on both original and tensor-transformed values. However, in our case, we need to use as parameters velocities obtained from time instant $t-1$. The interpolated velocities are:

$$\mathbf{u}' = \text{interp}(\mathbf{u}_P^{t-1}, \mathbf{x}_P), \mathbf{T}\mathbf{u}' = \text{interp}(\mathbf{T}\mathbf{u}_P^{t-1}, \mathbf{x}_P), \tag{3.8}$$

where $\text{interp}(\mathbf{u}, \mathbf{x})$ is a bilinear or trilinear interpolation function, depending on the simulation.

Finally, we plug these new interpolated velocity values into our Eq. 3.6 to obtain the updated $\mathbf{u}_G$ value:

$$\mathbf{u}_G^{t+1} = \mathbf{u}' + \Delta t[\mathbf{T}\mathbf{u}'(1 - \Delta t) - \mathbf{u}']. \tag{3.9}$$



$\mathbf{x}_G$
$\mathbf{u}_G^t$
$-\mathbf{u}_G^t\Delta t$
$\mathbf{x}_P$
$\mathbf{x}_G$
$\mathbf{u}_G^t$
$\mathbf{x}_P$
$\mathbf{u}$
$\mathbf{u} = e^{-\Delta t(\mathbf{I}-\mathbf{T}')}\mathbf{u}'$
$\mathbf{x}_G$
$\mathbf{u}_G^{t+1} = \mathbf{u}$

Figure 3.4: Illustration of the enhanced customized advection, now using Eq. 3.5.

The problem with this solution is that, despite providing much better control than the external forces approach discussed in the beginning of this chapter, it is still limited by the time step. Of course, time steps in this case are not that prohibitive as they are when Eq. 3.2 is used as a force, but they still should be kept low for the simulation to work adequately. Next, we provide a better solution to the problem of adding tensor information to the advection.

3.2.2 Analytical solution

An alternative to the previous solution is to replace Eq. 3.9 with Eq. 3.5 in the advection process. Observing Figure 3.4, the backtracing procedure is the same. The position $\mathbf{x}_P$ is encountered by moving the particle back in time. However, upon finding $\mathbf{x}_P$, the new velocity value used to update $\mathbf{u}(\mathbf{x}_G)$ at time $t+1$ is the result of a direct application of Eq. 3.5. Mathematically speaking, this leaves us with the following update equation for the velocity:

$$\mathbf{u}_G^{t+1} = e^{-\Delta t(\mathbf{I}-\mathbf{T}_P)}\mathbf{u}_P^t. \tag{3.10}$$

During the semi-Lagrangian advection process, we mentioned there is always the possibility of $\mathbf{x}_P$ not falling in an exact cell center. In fact, that is the most likely possibility. So, we need to use again the interpolated velocity $\mathbf{u}'$ for our calculations. As in this equation the tensor is not directly applied to the velocity (it is part of the exponential function instead), we also need to obtain a tensor $\mathbf{T}' = \text{interp}(\mathbf{T}_P, \mathbf{x}_P)$ interpolated among its neighbors. So, the final update equation becomes:

$$\mathbf{u}_G^{t+1} = e^{-\Delta t(\mathbf{I}-\mathbf{T}')}\mathbf{u}'. \tag{3.11}$$

This solution is more interesting for a number of reasons. First, differently from the numerical solution, where the original velocities were also part of the calculation, Eq. 3.10 depends only on the tensor-transformed velocity in $\mathbf{x}_P$. Second, the time-dependence on Eq. 3.9 was much more restrictive than in this new solution. In that case, the multiplication of **Tu** by $(1 - \Delta t)$ created a problem where an increase in the time step would reduce the tensor participation in the advection. So, although this can produce good results for small time steps ($\Delta t < 0.1$), it seriously reduces our freedom when setting the simulation parameters. Obviously, there is also the issue of a negative value for $\Delta t > 1$, but since such a high time step is not common practice, this concern can be disregarded.

Another problem involving Eq. 3.9 is that the tensor contribution is added to the current velocity. This means that the acceleration provided by the tensor term is acccumulated during iterations. So, if a higher time step is employed, the advected quantities will propagate faster, reducing the tensor influence on the field. Thus, once more we have an issue with higher time steps. In order for the tensor to really affect the velocities, the time step would need to be reduced, so that the tensor influence could accumulate over time for that cell. As for Eq. 3.5, the tensor is directly applied to the initial velocity, and the result of this transformation is used as the new velocity, without summing it with the current one. So, the tensor instantaneously affects that velocity in the analytical solution. Plus, in Eq. 3.5, we have an exponential function, which results in a much stronger tensor influence.

The numerical solution, however, is simpler and less costly. For time steps smaller than 0.1, the numerical solution approaches very closely the analytical one. As the time step increases, the solution begins to diverge. So, for small time steps, the numerical solution could be used to compensate for the performance hit caused by the exponential in Eq. 3.5.

3.2.3 Stability discussion

Adding tensors to the semi-Lagrangian advection could, in theory, disavow the unconditional stability claim of the original method. Dealing with normalized tensor fields would not present an issue regarding stability. However, we apply the scaling factor β to an input normalized tensor field in order to stimulate motion, so that the maximum amplification for fluid velocities is exactly β. The boosting factor poses a valid concern on the possibility of the simulation eventually blowing up. Even though we can satisfactorily control the simulation by carefully choosing the scaling factor, we attempt to ensure that stability holds for our customized advection by following standard literature on that matter and applying the CFL (Courant–Friedrichs–Lewy) condition, which limits time step to a maximum value based on grid spacing. Although the CFL condition is not a stability condition *per* *se*, it truly reduces the probability of a simulation blowing up [76]:

$$\Delta t \leq \frac{C\Delta x}{|\mathbf{u}|}.$$

The velocity $\mathbf{u}$ in the above equation is the maximum velocity encountered in the fluid in a single iteration. FOSTER e FEDKIW [6] suggests, based on experience, a value around 5 for the CFL number C, so that the backtracing particle trajectory length is no longer than five grid cells. Considering the tensor effect in our velocities and the analytical solution presented in Eq. 3.5, the CFL condition assumes the following form:

$$\Delta t \leq \frac{5\Delta x}{||\mathrm{e}^{-\Delta t(\mathbf{I}-\mathbf{T})}\mathbf{u}||}. \tag{3.12}$$

In our case, the worst-case scenario is when the velocity vector is fully aligned with the tensor's main eigenvector, and this tensor has its main eigenvalue greater than 1. This could significantly magnify the velocity. So, for our CFL condition, we consider the maximum velocity as the one that, when transformed by the tensor through Eq. 3.5, assumes the biggest norm in the grid. Now, consider a valid vectorial norm $||\cdot||$ that induces a matricial norm defined as:

$$||\mathbf{M}|| = \max_{\mathbf{x}\in\mathbb{R}^n, \mathbf{x}\neq 0} \frac{||\mathbf{Mx}||}{||\mathbf{x}||},$$

or, equivalently:

$$||\mathbf{M}|| = \max_{\mathbf{x}\in\mathbb{R}^n, ||\mathbf{x}||=1} ||\mathbf{Mx}||.$$

Given a matricial norm, it can be proven that:

$$||\mathbf{Ax}|| \leq ||\mathbf{A}|| \cdot ||\mathbf{x}||. \tag{3.13}$$

Let $\mathbf{A}$ a symmetric matrix that defines an operator in $\mathbb{R}^n$. We can also prove the following statement:

$$||\mathbf{A}|| = \max_j ||\lambda_j||, \tag{3.14}$$

where λ_j are the eigenvalues of A. So, let us look again at the stability condition in Eq. 3.12. From Eq. 3.13, we know that:

$$||\mathrm{e}^{-\Delta t(\mathbf{I}-\mathbf{T})}\mathbf{u}|| \leq ||\mathrm{e}^{-\Delta t(\mathbf{I}-\mathbf{T})}|| \cdot ||\mathbf{u}||.$$

From Eq. 3.14, we also know that:

$$\Delta t|\lambda_{max}| \leq \frac{5\Delta x}{||\mathbf{u}||},$$

where λ_{max} is the largest eigenvalue of $\mathrm{e}^{\Delta t(\mathbf{I}-\mathbf{T})} = \mathbf{P}\mathrm{e}^{\Delta d\mathbf{D}}\mathbf{P}^{-1}$, $\mathbf{P}$ is the matrix of eigenvectors of $(\mathbf{I}-\mathbf{T})$ and $\mathbf{D} = \mathrm{diag}(\lambda_1, \lambda_2, \cdots, \lambda_n)$ are the eigenvalues of $(\mathbf{I}-\mathbf{T})$. So, we obtain:

$$\Delta t|\mathrm{e}^{-\Delta t\lambda_j}| \leq \frac{5\Delta x}{||\mathbf{u}||}, \tag{3.15}$$

where λ_j is the smallest eigenvalue in $\mathbf{D}$. Thus, we have:

$$\ln \Delta t - \Delta t\lambda_j \leq \ln\left(\frac{5\Delta x}{||\mathbf{u}||}\right), \tag{3.16}$$

which does not have a closed form solution. Rearranging the terms, we can rewrite Eq. 3.17 as:

$$\ln \Delta t \leq \Delta t\lambda_j + \ln\left(\frac{5\Delta x}{||\mathbf{u}||}\right), \tag{3.17}$$

where the L.H.S. is a logarithmic function of Δt and R.H.S. is linear function of Δt. To find the intersection between these two functions is not complicated and can be done efficiently. The smallest Δt can be easily obtained as the intersection of the L.H.S. and the R.H.S., for $||\mathbf{u}|| \neq 0$. Notice that:

$$\lim_{\Delta t \to 0} \ln \Delta t = -\infty,$$

while the R.H.S., when $\Delta t = 0$, is $\ln \frac{5\Delta x}{||\mathbf{u}||}$, which is constant. Thus, it means there is always a Δt which satisfies inequality Eq. 3.17. So, at each iteration, it is easy to find the maximum Δt that guarantees stability. This is illustrated in Fig. 3.5.

3.3 Customized diffusion

In Section 2.3.4, we discussed the process used by STAM [5] to solve the diffusion step of the simulation. In summary, what we want to solve is the contribution of the term $\nu\nabla^2\mathbf{u}$ from the Navier-Stokes equations. We mentioned that solving it directly is not stable, and one should adopt an implicit approach to this problem, by solving Eq. 2.25, shown below:

$$(\mathbf{I} - \nu\Delta t\nabla^2)\mathbf{u}'(x) = \mathbf{u}(x).$$

The equation above is specific for the diffusion of momentum, with ν being the fluid viscosity. Let us consider a general equation for the diffusion of any fluid-related quantity. We will define it as:

$$(\mathbf{I} - \phi\Delta t\nabla^2)q'(x) = q(x), \tag{3.18}$$

where ϕ is a diffusion coefficient, and $q'(x)$ and $q(x)$ are the diffused and original

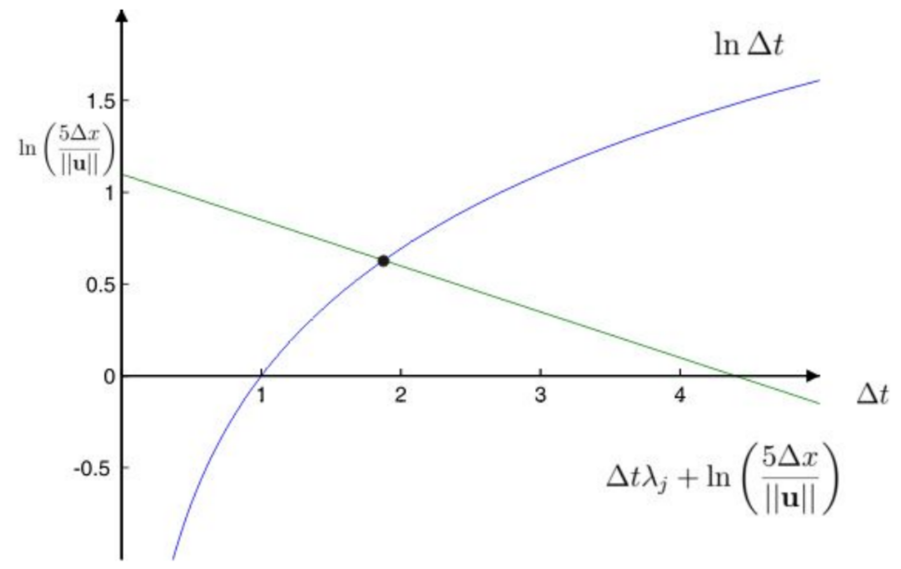


Figure 3.5: As Δt approaches 0, $\ln \Delta t$ goes to infinity. Thus, we can guarantee there is always a valid time step for our advection. When $\lambda_j > 0$, the linear function is increasing, so there is no limit on Δt. This makes sense, since a positive λ_j indicates a purely dissipative tensor.

quantities, respectively.

STAM [71] details the reasons why the implicit formulation should be used. If we take the direct approach, we simply need to compute the contributions of each cell to its neighbors. Thus, considering a 4-neighborhood in a 2D scenario, the exchanges between the center cell (where we want to compute the new value) and its four neighbors will result in a standard Laplacian mask. The value at cell (i, j) will be diffused to the neighbors, but this cell will also receive contributions from each one of them. This summarizes into the following update equation:

$$q_{i,j}^{t+1} = q_{i,j}^{t} + \phi \Delta t \Delta x \Delta y (q_{i-1,j}^{t} + q_{i+1,j}^{t} + q_{i,j-1}^{t} + q_{i,j+1}^{t} - 4q_{i,j}^{t}), \tag{3.19}$$

where t indicates the time instant and i and j are the respective x and y indices for the grid cells.

The problem with this approach is that as ϕ increases, q starts to oscillate until it diverges. So, we take the implicit road and look back in time to find which values q^{t+1}, when diffused backward in time, produce the value q^t. So, the variables in Eq. 3.19 have their time instant t switched:

$$q_{i,j}^{t} = q_{i,j}^{t+1} - \phi \Delta t \Delta x \Delta y (q_{i-1,j}^{t+1} + q_{i+1,j}^{t+1} + q_{i,j-1}^{t+1} + q_{i,j+1}^{t+1} - 4q_{i,j}^{t+1}).$$

This leaves us with a linear system with unknowns $q_{i,j}^{t+1}$. By rearranging the equation and applying an iterative solver like the Gauss-Seidel relaxation, we have to solve the following update equation:

$$q_{i,j}^{t+1} = \frac{q_{i,j}^{t} + \phi \Delta t \Delta x \Delta y (q_{i-1,j}^{t+1} + q_{i+1,j}^{t+1} + q_{i,j-1}^{t+1} + q_{i,j+1}^{t+1})}{1 + 4\phi \Delta t \Delta x \Delta y}, \tag{3.20}$$

This equation is solved over a predefined number of iterations. In [71], the default number used is 20. The advantage of this formulation is that it is completely stable for any values of ϕ or Δt.

The above formulation is great for traditional fluid diffusion, but it is problematic to our purposes, because this process is essentially isotropic. It uses only the nearest neighbors, so it cannot capture the Laplacian in some directions. The interesting thing about using the tensor field to model fluid flow is to influence flow direction by propelling the fluid in particular directions. We saw how this is done in advection, but for that to work also in diffusion, we will need to adopt an alternate approach. So, in the next section we present our proposal for turning diffusion anisotropic.

3.3.1 Proposed anisotropic diffusion

In Section 1.1.2, we presented a brief review of works dealing with the problem of anisotropic diffusion. This is a common problem in physics, so it has been extensively studied, with several published works available in the literature. We decided to use the discretization schemes proposed by GÜNTER *et al.* [29]: the asymmetric and the symmetric ones. First, let us review the altered diffusion term presented in Eq. 3.4. For a generic quantity q, we can define it as:

$$\frac{\partial q}{\partial t} = \nabla \cdot (\mathbf{T} \nabla q)$$

The finite difference schemes proposed by GÜNTER *et al.* [29] solve this equation by directly calculating the product between the diffusion tensor and the gradient of q, and then finally calculating the divergence. The asymmetric scheme treats the flux values differently for x and y, using positions $(i \pm 1/2, j)$ and $(i, j \pm 1/2)$, while the symmetric one uses position $i + 1/2, j + 1/2$ for both differentials. Figures 3.6 and 3.7 illustrate both schemes.

Each one of these schemes has advantages and disadvantages. For example, the symmetric scheme works better for tensors oriented in the diagonal, while these tensors have poor result for axis-aligned tensors. The symmetric approach also fails with isotropic tensors, forming a checkerboard pattern due to the lack of exchanges between the nearest neighbors. Actually, the checkerboard pattern plagues this scheme for most tensors with low levels of anisotropy. The asymmetric scheme,

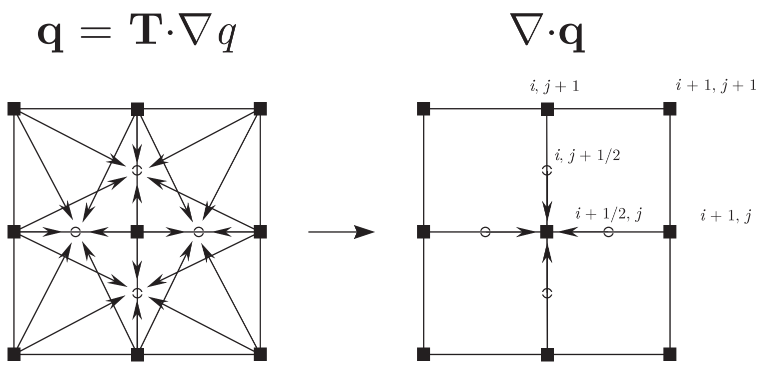


Figure 3.6: Asymmetric finite difference scheme. Source: [3]

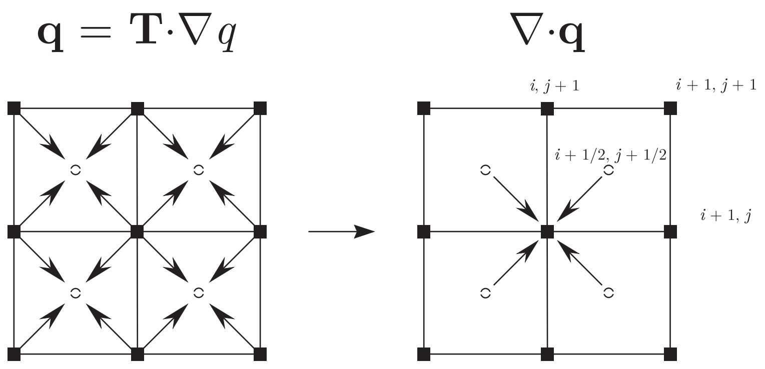


Figure 3.7: Symmetric finite difference scheme. Source: [3]

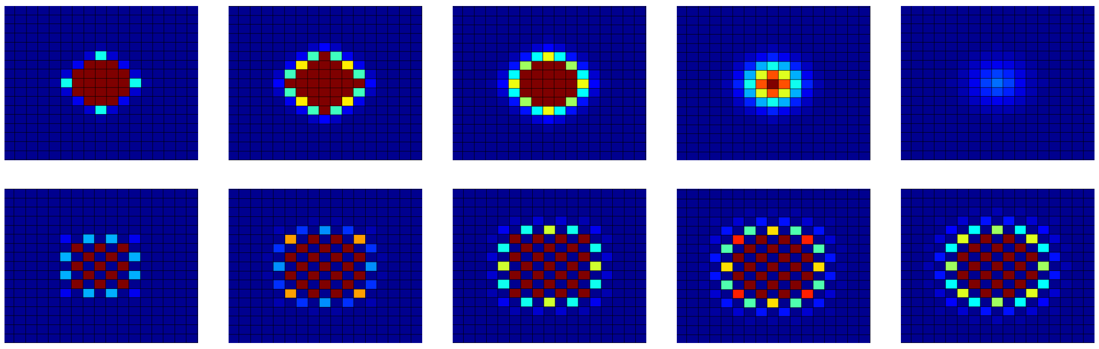


Figure 3.8: Two sequences of frames for diffusion in the presence of a linear tensor, oriented horizontally: the top sequence used the asymmetric scheme, while the bottom sequence used th symmetric one.

however, works fine for both isotropic and highly anisotropic axis-aligned tensors. In the diagonal, the asymmetric still offers good results, but produces some artifacts. Figures 3.8, 3.9 and 3.10 shows various configurations of a Matlab simulation of heat transfer using these two schemes.

As we can see, no scheme alone is ideal for tensors oriented arbitrarily. Thus, we propose to use a combination of these two schemes. Experiments showed that a simple average between these two is enough to enhance the results. The asymmetric scheme provides the bigger contribution to the overall result. Although the average

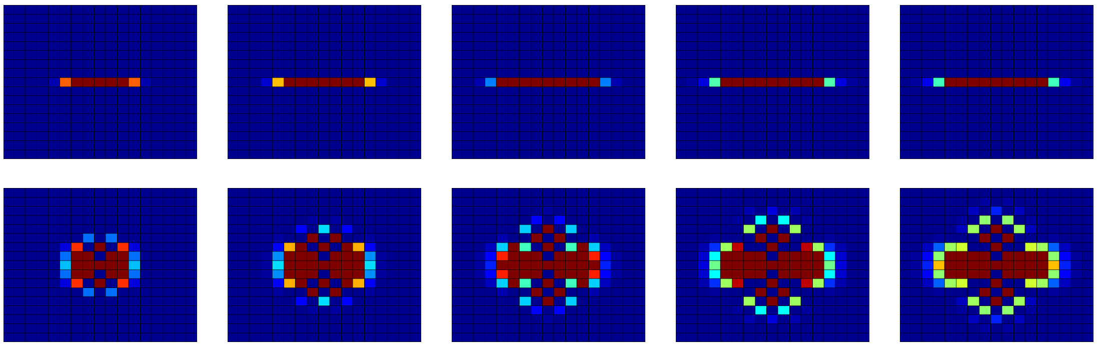


Figure 3.9: Two sequences of frames for diffusion in the presence of a linear tensor, oriented diagonally: the top sequence used the asymmetric scheme, while the bottom sequence used th symmetric one.

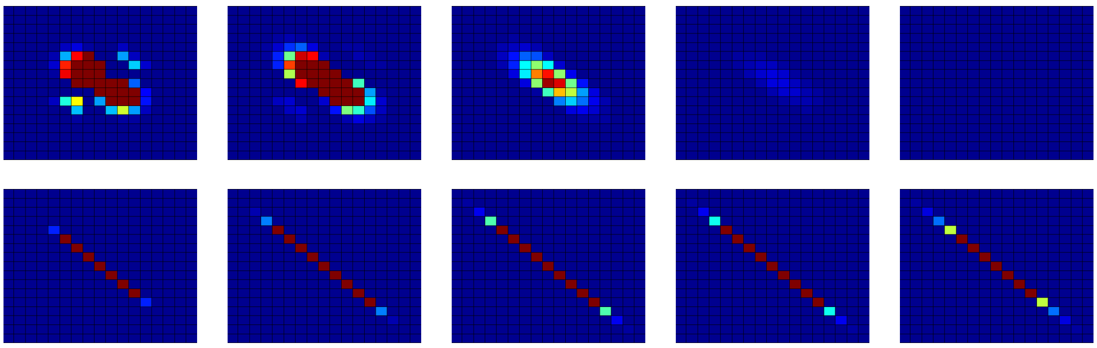


Figure 3.10: Two sequences of frames for diffusion in the presence of an isotropic tensor: the top sequence used the asymmetric scheme, while the bottom sequence used th symmetric one.

is not as good as the best case scenario for the symmetric scheme, it alleviates the problems encountered in this case with the asymmetric one. Besides, the majority of the tensors we use in practice are not purely linear. So, the symmetric scheme in the average serves as a way of compensating for the few situations where the asymmetric presents issues. Figure 3.11 shows the outcome of the averaged scheme.

As our tensor fields are static throughout the simulation, we can preprocess these operators and obtain a Laplacian mask to be directly applied during the diffusion step. This allows us to reutilize the diffusion solution used in the classic fluid simulation. So, the Gauss-Seidel relaxation scheme is once again used to implicitly solve the diffusion equation. However, now instead of using a simple isotropic mask, we propose to use masks computed from the differential operators proposed by GÜNTER *et al.* [29] to calculate the new value of q. Let $S(\mathbf{T})$ be the symmetric mask calculated for $\mathbf{T}$ and $A(\mathbf{T})$ be the asymmetric mask. The equations used for calculating the masks can be encountered in Appendix A. The final mask $M(\mathbf{T})$ will

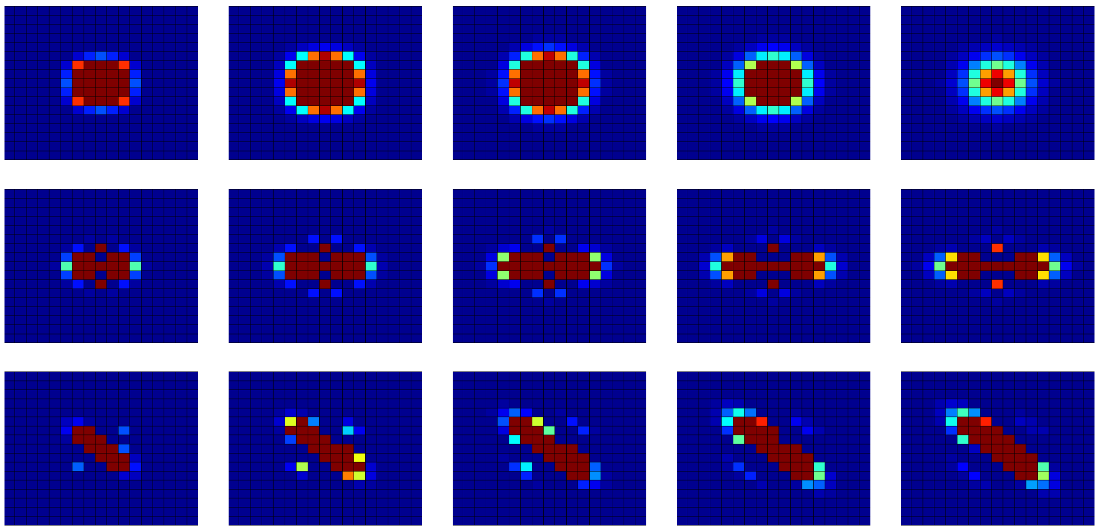


Figure 3.11: Sequences of frames for the averaged scheme, using the same three tensors of Figs. 3.8, 3.9 and 3.10. The topmost sequence is the isotropic tensor, the center sequence is the horizontal linear tensor and the bottom sequence is the diagonal linear tensor.

be defined as:

$$M(\mathbf{T}) = \frac{S(\mathbf{T}) + A(\mathbf{T})}{2}.$$

The update equation for the anisotropic Gauss-Seidel relaxation technique will be as follows:

$$q_{i,j}^{t+1} = \frac{q_{i,j}^{t} + \sum_{i=1}^{3}\sum_{j=1}^{3} M_{i,j} Q_{i,j}}{(1 - q_{i,j}^{t+1})}, \tag{3.21}$$

where Q is the following matrix containing the values of q for neighbors:

$$Q = \begin{bmatrix} q_{i-1,j-1}^{t+1} & q_{i-1,j}^{t+1} & q_{i-1,j+1}^{t+1} \\ q_{i,j-1}^{t+1} & 0 & q_{i,j+1}^{t+1} \\ q_{i+1,j-1}^{t+1} & q_{i+1,j}^{t+1} & q_{i+1,j+1}^{t+1} \end{bmatrix}$$

Notice the diffusion coefficient ϕ is gone. In fact, we do not need it anymore. The tensor acts now as the diffusion coefficient. When the diffusion becomes isotropic, the coefficient reduces to a scalar. So, the tensor-based diffusion will be stronger or weaker depending on the orientation of the tensor eigenvectors and on its eigenvalues. This anisotropic diffusion complements the idea of a tensor as a property of the medium. Depending on the quantity q, the tensor will acquire a different meaning. For velocities, for instance, it becomes the viscosity tensor, reducing or amplifying momentum in the specific directions according to the tensor orientation.

An important concern about anisotropic diffusion, discussed in [3], is the possibility of large numerical errors introduced in the system due to the presence of highly anisotropic tensors. Our main interest is in the linear tensors, so this is a valid concern. However, the Laplacian is dissipative, locally reducing the error in a monotonic fashion at each iteration. Besides, the Laplacian always has a negative value at its center, which forces the denominator in the update equation to always be greater than 1, ensuring the dissipation and the convergence. Thus, we obtain a valid Laplacian mask, regardless of the tensor orientation.

Drawbacks

There are a few drawbacks related to this solution. Even for the averaged scheme, highly anisotropic tensors may present artifacts during the diffusion process. Extremely linear tensors are difficult to handle, as already documented in the specialized literature. Plus, there is the problem of drifting. When diffusion takes place, the quantity values slightly drift laterally, which is certainly not a desired behavior. Anisotropic diffusion is a complex problem with vast literature, so more complex schemes may be able to work around these issues. Further investigation shall be done in the future to improve the results obtained so far.

3.3.2 Stability discussion

Once again, as we are inserting extraneous information to the original stable fluid simulation method, we need to analyze if stability still holds after the proposed alterations. BRIDSON [76] demonstrates the stability of the implicit treatment of diffusion for the classic method. By analyzing a one-dimensional version of the diffusion problem, he shows that the weights in the update equation presented in Eq. 3.20 are all positive and they sum to 1. This guarantees that the new value for q is bounded by its minimum and maximum old values. In 2D, the update equation can be rewritten as follows to evidence the weights:

$$
\begin{aligned}
q_{i,j}^{t+1} = \left(\frac{1}{1+4\phi\Delta t\Delta_x\Delta_y}\right) q_{i,j}^{t} &+ \left(\frac{\phi\Delta t\Delta x\Delta y}{1+4\phi\Delta t\Delta_x\Delta_y}\right) q_{i-1,j}^{t+1} \\
&+ \left(\frac{\phi\Delta t\Delta x\Delta y}{1+4\phi\Delta t\Delta_x\Delta_y}\right) q_{i+1,j}^{t+1} \\
&+ \left(\frac{\phi\Delta t\Delta x\Delta y}{1+4\phi\Delta t\Delta_x\Delta_y}\right) q_{i,j-1}^{t+1} \\
&+ \left(\frac{\phi\Delta t\Delta x\Delta y}{1+4\phi\Delta t\Delta_x\Delta_y}\right) q_{i,j+1}^{t+1}.
\end{aligned}
$$

We reproduce this analysis for our tensor-based diffusion. In our case, we consider this 2D version of this problem, since our problem is minimally bidimensional (there can be no anisotropy in 1D). By expanding and reorganizing the anisotropic Laplacian equation for each one of the discretization schemes presented in the previous section, we get to two tensor-based update equations.

First, let us analyze the symmetric scheme. Let a be the product $\phi \Delta t \Delta x \Delta y$. Discretizing $\nabla \cdot (\mathbf{T}\nabla q)$, we get to the following equation for $q_{i,j}$:

$$\begin{aligned} q_{i,j}^{t+1} = & \frac{q_{i,j}^{t} + \frac{a}{4}(T_{xx}+T_{xy}+T_{yx}+T_{yy})(q_{i+1,j+1}^{t+1}+q_{i-1,j-1}^{t+1})}{1+a(T_{xx}+T_{yy})} \\ & + \frac{\frac{a}{4}(T_{xx}-T_{xy}-T_{yx}+T_{yy})(q_{i+1,j-1}^{t+1}+q_{i-1,j+1}^{t+1})}{1+a(T_{xx}+T_{yy})} \\ & + \frac{\frac{a}{4}(2T_{xx}-2T_{yy})(q_{i+1,j}^{t+1}+q_{i-1,j}^{t+1})}{1+a(T_{xx}+T_{yy})} \\ & + \frac{\frac{a}{4}(2T_{yy}-T_{xx})(q_{i,j+1}^{t+1}+q_{i,j-1}^{t+1})}{1+a(T_{xx}+T_{yy})}. \end{aligned} \tag{3.22}$$

Notice that we now have 8 neighbors contributing to the new value. We want to verify if the weights sum to 1. Putting the weights in evidence, we can then rewrite the previous equation as:

$$\begin{aligned} q_{i,j}^{t+1} = \left(\frac{1}{1+a(T_{xx}+T_{yy})}\right) q_{i,j}^{t} & + \left(\frac{a(T_{xx}+T_{xy}+T_{yx}+T_{yy})}{4(1+a(T_{xx}+T_{yy}))}\right)(q_{i+1,j+1}^{t+1}+q_{i-1,j-1}^{t+1}) \\ & + \left(\frac{a(T_{xx}-T_{xy}-T_{yx}+T_{yy})}{4(1+a(T_{xx}+T_{yy}))}\right)(q_{i+1,j-1}^{t+1}+q_{i-1,j+1}^{t+1}) \\ & + \left(\frac{a(2T_{xx}-2T_{yy})}{4(1+a(T_{xx}+T_{yy}))}\right)(q_{i+1,j}^{t+1}+q_{i-1,j}^{t+1}) \\ & + \left(\frac{a(2T_{yy}-2T_{xx})}{4(1+a(T_{xx}+T_{yy}))}\right)(q_{i,j+1}^{t+1}+q_{i,j-1}^{t+1}). \end{aligned}$$

Considering that each weight appears twice in the equation (each one multiplying two different neighbors), if we sum them all and simplify the opposing terms, we obtain:

$$\frac{1}{1+a(T_{xx}+T_{yy})} + \frac{4a(T_{xx}+T_{yy})}{4(1+a(T_{xx}+T_{yy}))} = \frac{1+a(T_{xx}+T_{yy})}{1+a(T_{xx}+T_{yy})} = 1$$

Similarly, we can discretize the anisotropic Laplacian for the asymmetric scheme. Reorganizing the equation so that the weights be in evidence, we have:

$$\begin{aligned} q_{i,j}^{t+1} = & \left(\frac{1}{1+2a(T_{xx}+T_{xy}+T_{yx}+T_{yy})}\right) q_{i,j}^{t} \\ & + \left(\frac{a(T_{xx}+T_{yx})}{1+2a(T_{xx}+T_{xy}+T_{yx}+T_{yy})}\right) Y \\ & + \left(\frac{a(T_{xy}+T_{yy})}{1+2a(T_{xx}+T_{xy}+T_{yx}+T_{yy})}\right) X, \end{aligned} \tag{3.23}$$

where $Y = q_{i+1,j}^{t+1} + q_{i+1,j-1}^{t+1} + q_{i+1,j+1}^{t+1} + q_{i-1,j-1}^{t+1} + q_{i-1,j}^{t+1} + q_{i-1,j+1}^{t+1} - 2(q_{i,j-1} + q_{i,j+1})$ and $X = q_{i-1,j+1}^{t+1} + q_{i,j+1}^{t+1} + q_{i+1,j+1}^{t+1} + q_{i-1,j-1}^{t+1} + q_{i,j-1}^{t+1} + q_{i+1,j-1}^{t+1} - 2(q_{i-1,j} + q_{i+1,j})$.

Summing the weights, we obtain:

$$\frac{1 + 6a(T_{xx} + T_{yx}) + 6a(T_{xy} + T_{yy}) - 4a(T_{xx} + T_{yx}) - 4a(T_{xy} + T_{yy})}{1 + 2a(T_{xx} + T_{xy} + T_{yx} + T_{yy})} =$$
$$= \frac{1 + 2a(T_{xx} + T_{yx}) + 2a(T_{xy} + T_{yy})}{1 + 2a(T_{xx} + T_{xy} + T_{yx} + T_{yy})} =$$
$$= \frac{1 + 2a(T_{xx} + T_{yx} + T_{xy} + T_{yy})}{1 + 2a(T_{xx} + T_{xy} + T_{yx} + T_{yy})} = 1.$$

As we can see, both discretizations result in a set of tensor-based weights that sum to 1. By combining the weights for our averaged scheme we obtain the same result. So, we can ensure that our diffusion method is bounded by the old values of q, being stable no matter how large the diffusion coefficient ϕ. In our method, the tensor boost β works as our diffusion coefficient instead. This can be easily demonstrated by extracting it from $\mathbf{T}$ and putting it outside the parentheses, as part of the factor a.

If we look at the results of the computation of the Laplacian masks, we will also notice that, in all cases, the central element is negative, compensating for the contributions of its neighbors. Thus, there is no possibility whatsoever that our customized diffusion method can hinder the stability present in the original method. Moreover, the matrix of the system solved by the Gauss-Seidel iterative method is diagonally-dominant, which means it always converge to the solution. Our method is stable even for null tensors. It is worth noting, however, that this case would produce a zero line in the Gauss-Seidel system. This could impact the exchanges between neighbors, since some cells would be able to share energy without receiving it. From a formal point of view, this is not exactly interesting. A more careful evaluation is needed in order to assess the local implications of this scenario.

3.4 Customized projection

The last part of our method involves the projection. We already discussed its importance in keeping the velocity field divergence-free, a prerequisite for mass conservation in the fluid. Even though we defined smoke as our fluid of choice for all our simulation experiments, we can consider smoke as an incompressible fluid. As mentioned in Section 2.2.1, the visual effect of considering a fluid as compressible is negligible. Plus, the conditions under which we perform our simulations also allows us to discard the compressibility. So, for our purposes, there is no gain in dealing with the complexities and the computational costs involved in simulating compressible flow.

The problem is that we now have a tensor field acting on the fluid. The original pressure solve, if used in such a context, may soften the tensor influence provided by the previous steps of the simulation. Furthermore, it makes sense that the pressure also be subject to the tensor field, since we interpret it as a property of the medium. Thus, the pressure should not be isotropic. The solution to the Poisson problem now is a bit more complex, since a new restriction were added to the system. We have to find a pressure p with a gradient in a specific direction who will take the velocities and enforce incompressibility by making them divergence-free.

The Poisson equation used for solving the pressure was defined in Section 2.3.5. We now reproduce it here:

$$\nabla^2 p = \frac{1}{\Delta t}\nabla \cdot \mathbf{u}.$$

The velocity vector $\mathbf{u}$ is now bound to the tensors. So, we must change equation above to also account for the effect of tensors in the pressure. So, the original equation becomes something like the following:

$$\nabla \cdot (\hat{\mathbf{T}}\nabla p) = \frac{1}{\Delta t}\nabla \cdot \mathbf{u}, \tag{3.24}$$

where $\hat{\mathbf{T}} = \mathbf{T}/|\mathbf{T}|$. The use of $\hat{\mathbf{T}}$ is due to the fact that in projection the tensor should be normalized. Indeed, we want a pressure field capable of dissipating energy, but the main goal is that it respects the principal directions of the tensor. Since the projection is just a step to enforce incompressibility, it does not make sense for it to be subject to any kind of gain or loss of momentum.

In [71], the projection is solved using the same Gauss-Seidel relaxation technique for calculating the pressures from the divergences. Here, we can do the same thing, using the Laplacian mask we calculated for the diffusion. Instead of using a matrix Q with all the values of the generic quantity q for the neighbors, we will use a matrix P for the neighboring pressures. The equation for calculating the pressure is then defined as:

$$p_{i,j}^{t+1} = \frac{\nabla \cdot \mathbf{u}_{i,j}^{t} + \sum_{i=1}^{3}\sum_{j=1}^{3} M_{i,j}P_{i,j}}{1 - p_{i,j}^{t+1}}. \tag{3.25}$$

As indicated before, discontinuities in the tensor field can negatively affect our projection. The tensor already imposes a bigger restriction to the Poisson solver. This means that pressure exchanges cannot occur in some directions, which may lead, in certain cases, to a system that has no solution. If, besides that, we also have null tensors in the field, we are further complicating things, since we are saying that in some specific points in space there are no pressure exchanges at all. The inability of correctly calculating pressures would result in a velocity field that cannot be made divergence-free. This can cause serious problems with advection, producing some small artifacts or even severely hampering the simulation. BRIDSON [76] recommends that advection be **always** performed in a divergence-free velocity field.

Chapter 4

Results and discussions

In this chapter we will present the results obtained with our method proposed. The experiments here described were conceived as a proof of concept of our work. We divided the presentation in two main sections. The first one is dedicated to comparing our proposal to Stam' original method. In Section 4.2, we show how we can design tensor fields with the purpose of controlling simulations. Apart from our smoke renderings, which were produced via the Mitsuba [77] rendering software, all other figures use colors to indicate values of interest, such as fluid speed or tensor anisotropy. Colors range from blue (lowest value) to red (highest value), with intermediate values set on green.

In all our simulations, we used a no-stick boundary condition, with stationary solid walls. Fluid is initially at rest and is put in motion by the application of some external force. Density insertion varies through different simulations, so we will describe it separately for each experiment. All simulations were run in a computer with an Intel Core i7-4700MQ processor, 8GB RAM and an NVIDIA GeForce GTX 765M video card. For most cases here presented, we were able to run at interactive times. This was more difficult for higher resolution fields, such as the 128×128 2D disk field and the helical field, but this can be circumvented by taking advantage of parallelization. Also, the additional computations involving tensors do not affect significantly the performance during the steps of the simulation. For the disk field, for example, we obtained an average 4fps with or without the tensor advection. For the diffusion and projection parts, since our Laplacian masks are precomputed, the performance hit is minimum.

4.1 Fluid dynamics comparisons

The simulations produced by our method are quite different from a classic fluid simulation. The effects of our tensor-based advection become especially apparent when compared with the standard Stable Fluids method [5]. To illustrate this, we used two tensor fields: one defined by 3 points and a simple vertical field composed only of linear tensors in a tube surrounded by null tensors. Both fields are depicted here with reduced resolution so that individual tensors are more easily distinguishable. They are displayed in Fig. 4.1. These fields are colored based on their norm, defined in Section 2.1.2.

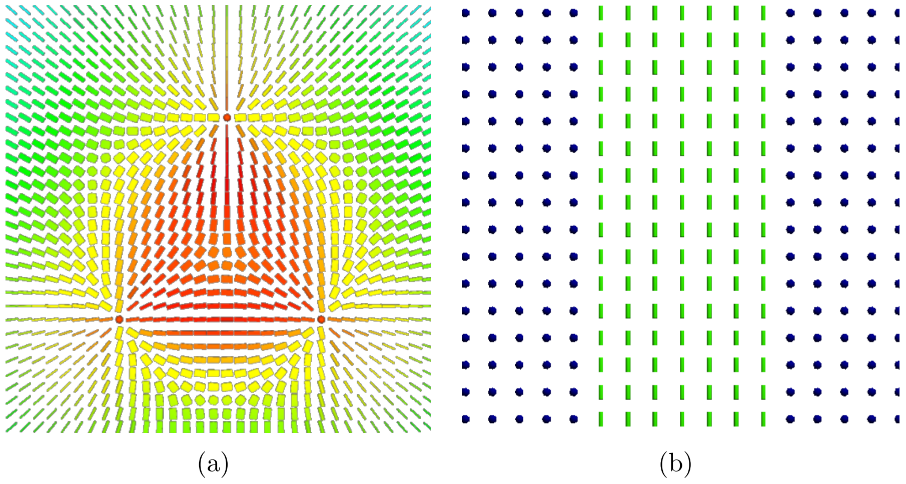


Figure 4.1: Two tensor fields used for comparison with ordinary fluid simulations. They are shown here in reduced resolution to clarify their structures. Colors indicate the tensor norm. (a) 3-point field; (b) Vertical tube field.

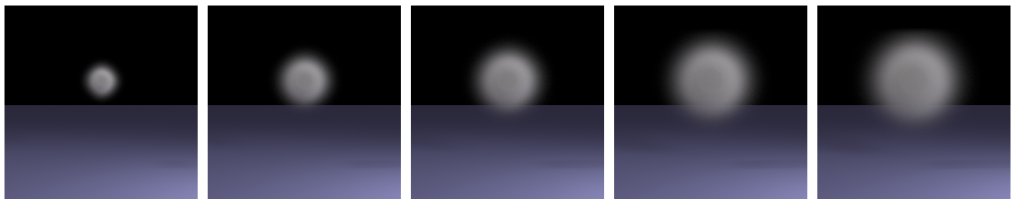
We separated the experiments in two parts. First, we leave the classic Stam advection in place and apply our anisotropic diffusion, comparing it to the original isotropic one. Then, we invert the setup: we use the tensor advection, without altering the diffusion step, to discuss the role of the tensor field in the advection process. In all cases, we leave the projection step untouched.

4.1.1 Diffusion and viscosity

In this subsection, we present the results of our anisotropic diffusion applied both to mass diffusion (density) and to momentum diffusion (viscosity). First, let us consider the vertical tube field. Figure 4.2 shows the difference between the basic diffusion employed by STAM [5] and the anisotropic diffusion proposed in Section 3.3.

For a tensor field where all tensors are diagonal (i.e., no isotropic tensors surrounding them), the directional diffusion becomes even more apparent. Figure 4.3 shows the tensor field and a selected frame of the simulation.

A similar comparison is performed for the 3-point field. This is a synthetic bidimensional tensor field, defined by three main points $\mathbf{q}_1 = (16, 16)$, $\mathbf{q}_2 = (48, 16)$ and $\mathbf{q}_3 = (32, 48)$ in a 64×64 grid. For each point $\mathbf{p}_{ij}$ in the grid, a tensor is


(a)

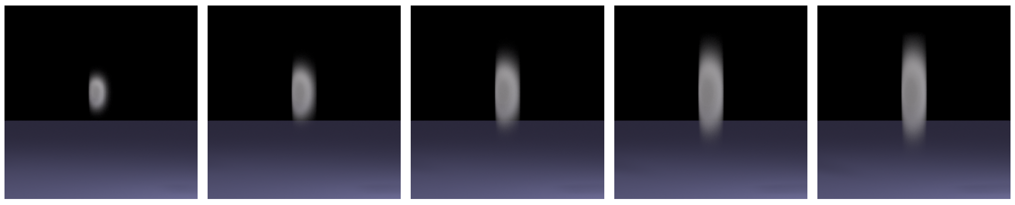
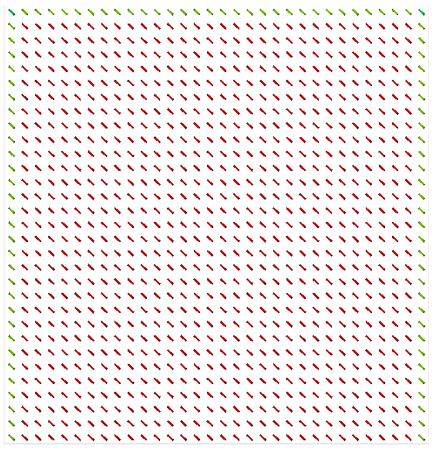

(b)

Figure 4.2: Diffusion sequence for the vertical tube in Fig. 4.1(b). Densities are inserted at grid center. (a) Simulation using Stam's diffusion. The smoke is diffused isotropically, completely unaware of the tensor field influence. (b) Simulation using the anisotropic masks for diffusion. The smoke is dispersed predominantly in the direction of the tube. Parameters: $\Delta t = 0.2$, $S = 10$.


(a)

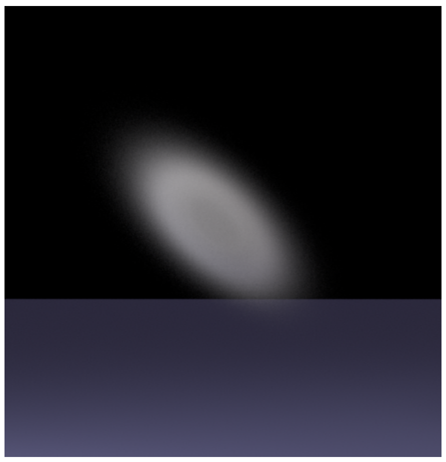

(b)

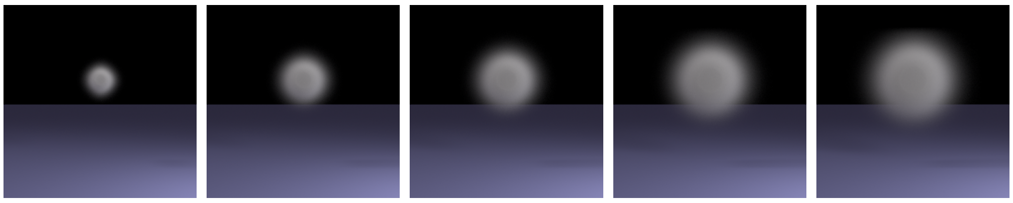
Figure 4.3: A completely diagonal tensor field and a frame of its correspondent diffusion simulation. Simulation parameters: $\Delta t = 0.2$, $S = 10$.

computed as follows:

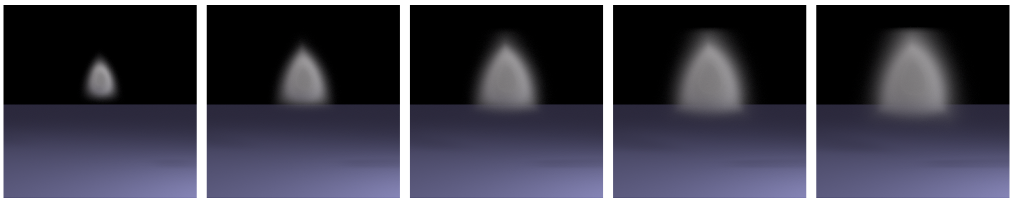
$$\mathbf{T}_{ij} = \sum_{k=1}^{3} \frac{(\mathbf{q}_k - \mathbf{p}_{ij}) \cdot (\mathbf{q}_k - \mathbf{p}_{ij})^T}{||(\mathbf{q}_k - \mathbf{p}_{ij})||}, \tag{4.1}$$

where $\mathbf{q}_k \neq \mathbf{p}_{ij}$. $\mathbf{T}_{ij}$ captures the uncertainty of the direction between grid point $\mathbf{p}_{ij}$ and every point $\mathbf{q}_k$. This uncertainty is weighed by the Euclidean distances between the points. Figur 4.4 shows the compared results.

As explained in Section 3.3, viscosity is also affected by the tensor field, using



(a)



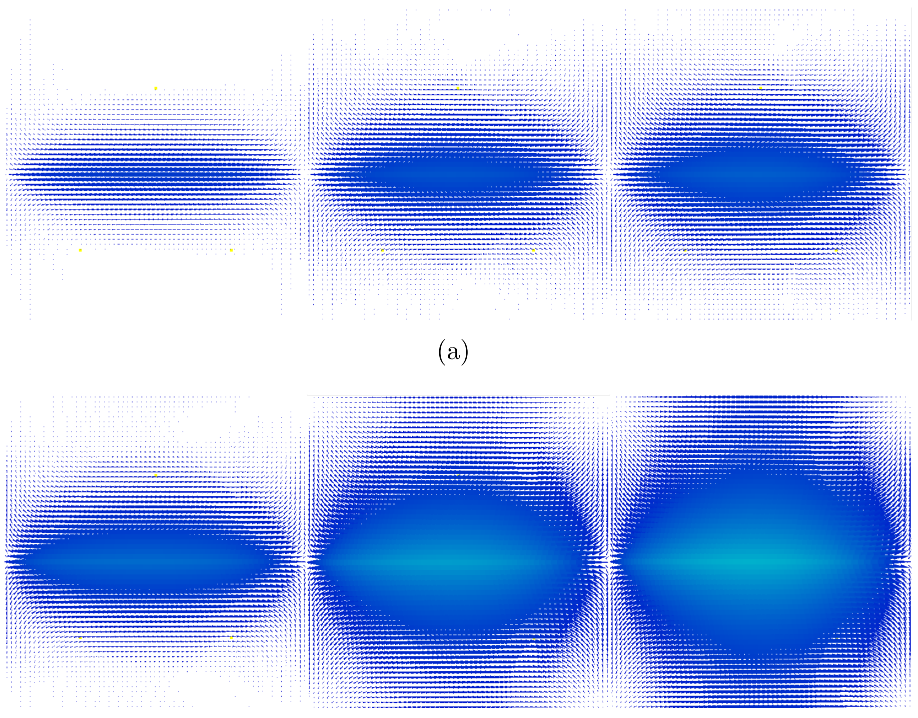
(b)

Figure 4.4: Diffusion sequence with densities being constantly inserted in the grid center. (a) Simulation using Stam's diffusion, as in Fig. 4.2(a). (b) Simulation using the anisotropic diffusion for the 3-point field shown in 4.1(a). The fluid gradually assumes the shape of the central region of the field. Parameters: $\Delta t = 0.2$, $S = 10$.

the same diffusion mask applied to the densities. In order to verify the effect of tensors in the viscosity, we used the vertical tensor field with a constant force applied perpendicularly to the field. As we can see in Figure 4.5, our method diffuses the velocities in the direction of the tensors, while the original one just reduces momentum isotropically. The direction followed by the velocities in the default method is just the direction of the applied force. With anisotropic viscosity, the tensor is also taken into account in this process.

4.1.2 Advection

Focusing now on the main part of our method, we begin by showing the advantage of using the customized advection proposed in Section 3.2 over the possibility of using Eq. 3.2 as a force. Figure 4.6 shows the differences between these two approaches. Velocity vectors are displayed for each grid cell in Figs. 4.6(b), 4.6(c) and 4.6(d), with colors mapping the amount of density in the cell. In Fig. 4.6(a), colors indicate the degree of colinearity in a neighborhood of the tensor, measured by the lattice index defined in Section 2.1.5. In Fig. 4.6(b), the advection acts freely, exactly as in the Stable Fluids method. In that case, the simulation is completely unaware of the underlying tensor field. In Figs. 4.6(c) and 4.6(d), instead, Eq. 3.2 is acting on the fluid, constraining fluid motion to the colinear regions marked in red in Fig. 4.6(a). The difference between the two tensor simulations is in the way tensor affects the fluid. The left image shows a simulation where Eq. 3.2 was applied as an external force (following the formulation in Eq. 3.3), while in the right one the advection is being molded by the proposed differential equation. While both approaches are


(a)

(b)

Figure 4.5: Comparison between isotropic (a) and anisotropic viscosity (b). While isotropic viscosity just reduces momentum equally in all directions, the anisotropic viscosity reduces it in specific directions. We show the evolution of both velocity fields at 3 iterations ($t = 5$, $t = 15$ and $t = 25$). The third frame is very close to steady state in both cases. Colors indicate the velocity norm.

.

capable of successfully controlling the fluid, affecting the advection enable a slightly finer control. As can be seen in Fig. 4.6, the tensor force tends to oscillate a bit more, forcing us to use very small time steps, so that the fluid can be adequately contained in the paths. Velocities in the tensor advection, however, are more directed, allows for larger time steps and dissipate quicker in planar regions.

Another way to illustrate the effects of our method is by showing how densities are affected. In this case, we use the vertical tube field. Two separate simulations were conducted. Results are provided in Figures 4.7(a) and 4.7(b). The simulation in Fig. 4.7(a) shows the result of Stam's classical advection, while Fig. 4.7(b) shows our results. Fluid is inserted at the center of the grid and propelled upwards. At a certain moment, an external force is applied perpendicularly to the fluid path, in an attempt to change its trajectory. Notice how the fluid in Stam's simulation assumes a completely different direction when compelled by an external force. In our method, after the initial disturbance ceases, the fluid returns to its original path, responding to the tensor influence. There are some densities, though, that remain still outside the tube's path. As all tensors are null in those regions, advection does not influence them. A damping function can be used to smoothly dissipate these

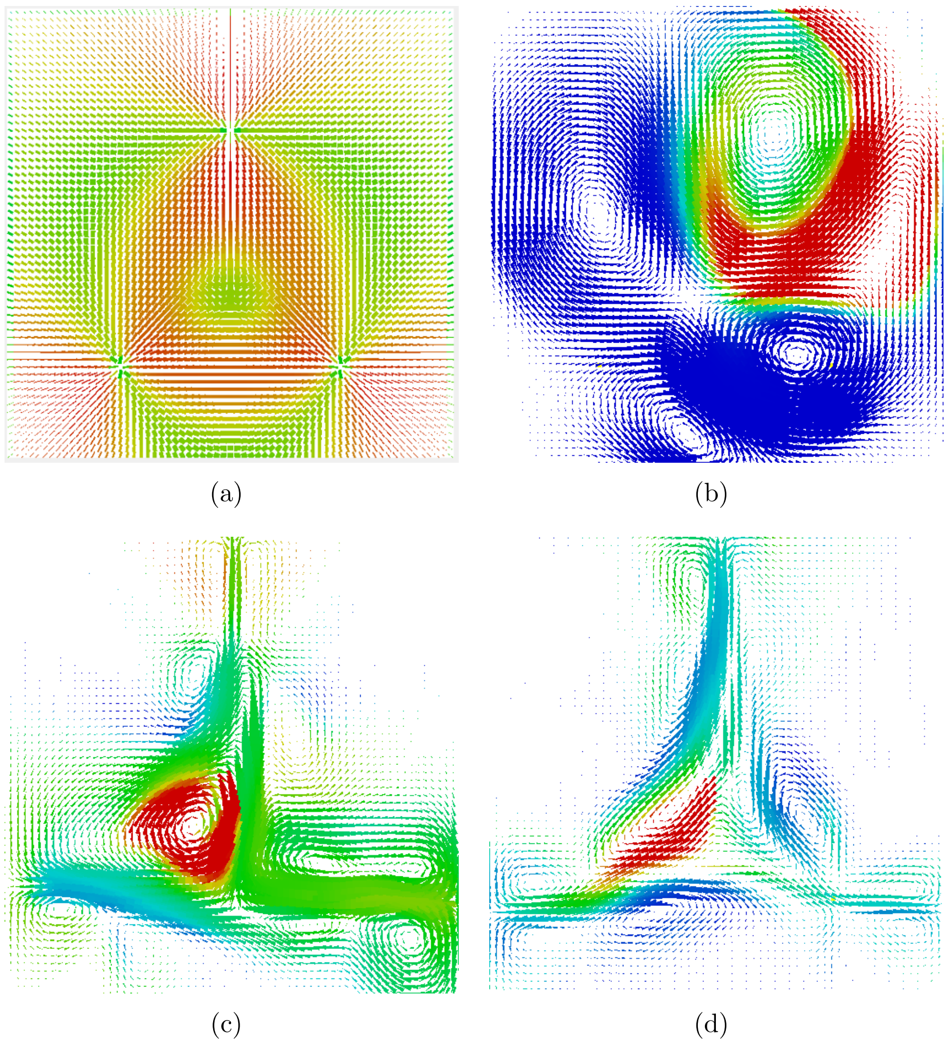

(a) (b) (c) (d)

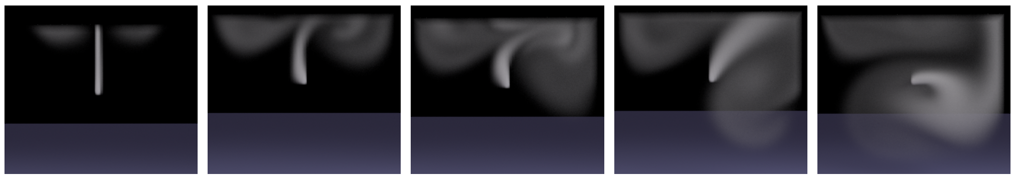
Figure 4.6: (a) Our 32×32 tensor field used in the simulation; (b) Classical fluid simulation, without using Eq. 3.2; (c) Simulation using Eq. 3.2 as an external force; (d) Simulation using Eq. 3.2 in advection. Notice how the simulations in Figs. (c) and (d) generate a velocity field with a shape that resembles the colinear regions of the tensor field in (a). We can also see that this shape is much more accentuated when Eq. 3.2 is used as an advection (d) than as an external force (c). Colors indicate level of anisotropy in (a) and density values in (b), (c) and (d).

densities. Stam's method used such a function for dissipating densities in every cell in the grid, while we only apply it to null tensor regions. The damping factor $-\alpha\rho$ is included in Eq. 3.4 in the section that describes our proposed method.

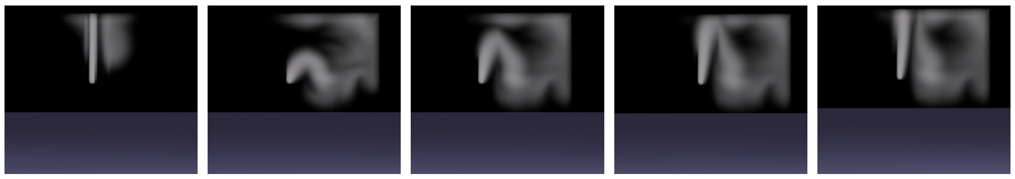
4.2 Tensor field design for fluid control

The addition of the tensor field complements the classic formulation, where we had basically two main variables: density and velocity. Now, we have three elements:

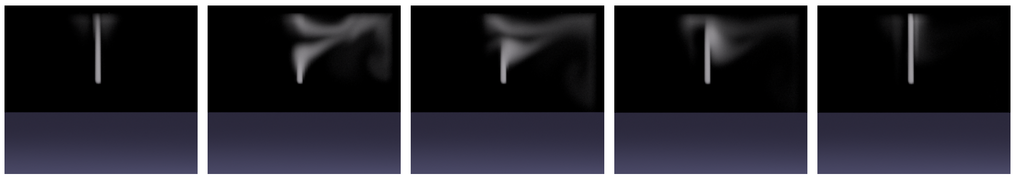
- the density field, with selected points for fluid injection;



(a)



(b)



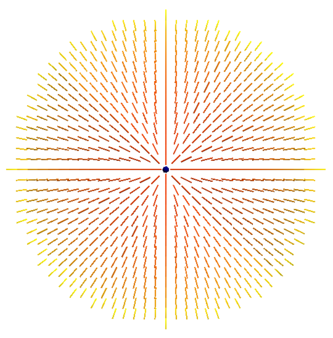
(c)

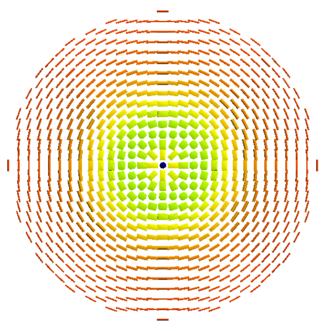
Figure 4.7: Animation sequence for the vertical tube in Fig. 4.1(b). (a) Simulation using Stam's advection. There is no tensor affecting the advection, thus the fluid loses its vertical path after being propelled by a force in a different direction. (b) Simulation using tensor advection. The method ensures that the fluid returns to its intended path, even after being deviated by the external force. No dissipation function was used, reason why densities accumulated outside the path after the force was applied. (c) Simulation using tensor advection, but with dissipation instead. The dissipation is applied everywhere outside the tube. Simulation parameters: $\Delta t = 0.05$, $S = 10$, $\beta = 1.3$ (β does not influence simulation (a)).

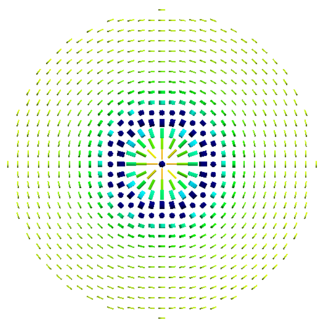
- the velocity field, which represents fluid momentum generated by external forces;
- and the tensor field, which limits fluid flow, i.e., it introduces new boundary conditions.

Thus, having these elements in mind, one could think of designing tensor fields to produce controlled fluid animations. Tensor information could be used to determine fluid injection or the external forces, with the orientational information of the tensor used to deflect the fluid, keeping it contained to the desired paths or regions. According to the motion pattern the user wants to apply to the fluid, different levels of anisotropy may be deemed necessary. In a general manner, linear tensors would be preferable for directed flow, while planar tensors are more useful for fluid containment. Isotropic tensors do not restrict fluid flow, so it is a good choice for filling regions with fluid. For example, if the user wants to simulate fluid flowing through pipes, linear tensors in the direction of intended flow should be considered. If, however, the user desires to create a fluidic shape by filling it with fluid, a certain dose of planar or isotropic tensors may be needed to adequately diffuse the fluid to all regions of the shape. Also, we could define the boundaries of a three-dimensional object by setting planar tensors around it to deflect the fluid along the borders, forming a kind of wall. Naturally, most practical fields will be a combination of these kinds of tensors in order to obtain interesting simulations.

An example is the disk-shaped tensor field, composed by the sum of the 3 fields depicted in Figure 4.8. For visual clarity, we show these tensor fields in a smaller resolution. The field is composed of radial linear tensors near the center and perpendicular linear tensors following the circumference near the border, forming a transition section formed by planar tensors. The rationale here is that we want a tensor field that initially propels the fluid radially from the insertion point and then keeps it in a circling pattern, forming a disk shape. The mid region (depicted in dark blue) allows the fluid to flow more freely, with no specific associated direction. Further from the center, the field becomes more aligned with the disk tangent in order to trap the fluid and reveal its external boundary.


(a)


(b)


(c)

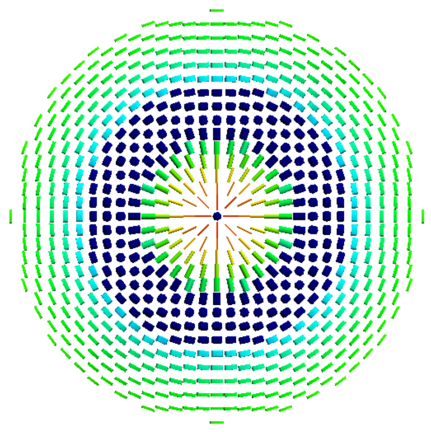

(d) Final tensor field

Figure 4.8: 48 × 48 tensor field (d) composed by the sum of fields (a), (b) and (c). For tensor field (a), $\mu = 0$, $\sigma = 0.4$, $\lambda_1 = f$ and $\lambda_2 = 0.1$; for (b), $\mu = 1$, $\sigma = 0.3$, $\lambda_1 = 0.1$ and $\lambda_2 = f$; for (c), $\mu = 0$, $\sigma = 0.1$, $\lambda_1 = f$ and $\lambda_2 = 1 - f$. As this is a bidimensional field, λ_3 was set to 0 for all of them.

Let d be the distance between grid cell $\mathbf{P}(i,j)$ and grid center $\mathbf{C}(N_x/2,\ N_y/2)$, where N_x and N_y are the grid dimensions. In order to obtain our disk, we define tensors for all cells within some radius r from the center. All tensors outside the disk are set to isotropic tensors with negligible norm. The distance d is defined as:

$$d = |\mathbf{C} - \mathbf{P}| = \sqrt{(C_x - P_x)^2 + (C_y - P_y)^2}.$$

The calculated distances are all normalized to the interval $[0,1]$, resulting in the distance d_n below:

$$d_n = \frac{|\mathbf{C} - \mathbf{P}|}{d_{max}}.$$

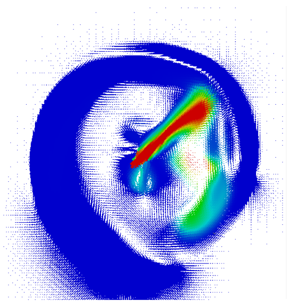
Finally, d_n is used in a Gaussian function to generate a factor f used as the eigenvalues needed to create the tensors, using Eq. 2.5. By varying the standard deviation σ, the mean μ and the eigenvalues to which we apply the Gaussian factor, we can obtain different distributions of tensors. If we look at Fig. 4.8, we see how changing these parameters influence the field design. In the case of Fig. 4.8(c), for example, as $\lambda_1 = f$ and $\lambda_2 = 1 - f$ are varying inversely, the orientation of the tensors gradually change from the center to the borders. Near the border, λ_2 is much bigger than λ_1, so the second eigenvector $\mathbf{e}_2$, which is perpendicular to the main eigenvector $\mathbf{e}_1$, will define a linear tensor oriented perpendicularly to the radial linear tensors near the disk center.

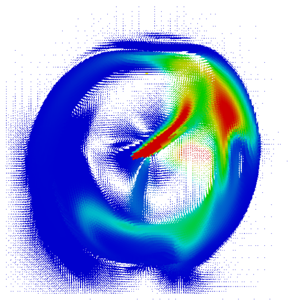
Naturally, this is just a way of mathematically generating a tensor field. Any other function could be used instead to produce tensors, like in the 3-points tensor field (Fig. 4.6). The central idea behind the construction of all these fields is the definition of eigenvalues and eigenvectors for each tensor in a way that it serves the purposes of the simulation.

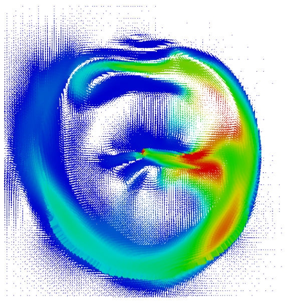
$$f = \mathrm{e}^{\frac{-(\mu - d_n)^2}{\sigma^2}}$$

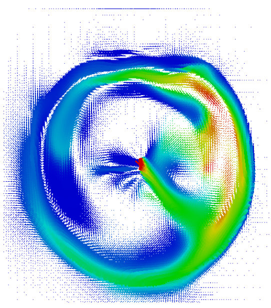
The velocity field for some frames of the disk simulation are displayed in Figure 4.9. For the simulation, we used a higher resolution tensor field (128×128). Notice the circling pattern at its borders. Fluid density is inserted at the disk center and is propelled outwardly by the radial linear tensors at the core. Upon reaching the external layers, the fluid starts to move along the boundaries, circling the disk, as imposed by the tensors in that region. Figure 4.10 shows the rendered smoke for this example. In this case, the damping function was used to gradually dissipate densities outside the disk.

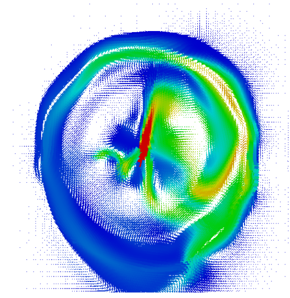
We have also designed some 3D fields to test our method. The wafer field (Fig 4.11) is composed of parallell layers of planar tensors, with their norms reducing radially from the center. Planar tensors encourage diffusion in more than one direction, so we used them to stimulate fluid diffusion along the layer, entirely filling the


(a)


(b)


(c)


(d)


(e)

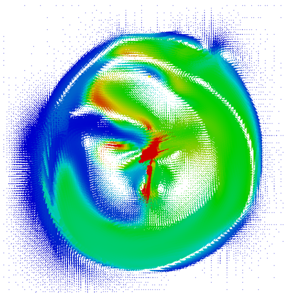
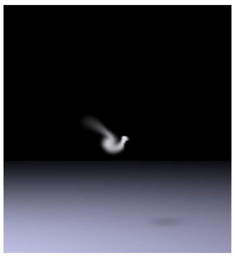
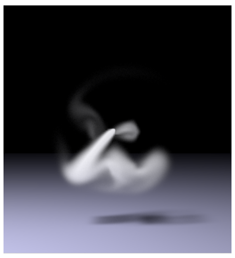
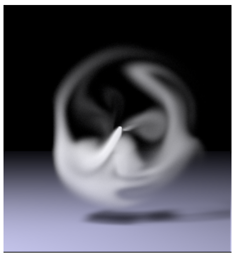
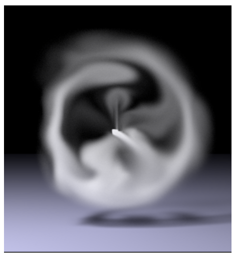

(f)

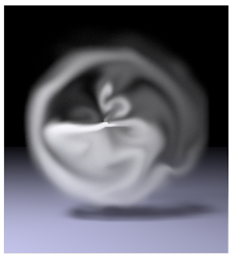
Figure 4.9: Velocity field in six different frames of a simulation with the disk tensor field. Color indicates densities, flowing from the center towards the borders, gradually filling the region. Simulation parameters: $\Delta t = 0.05$, $\beta = 3.0$, and $\alpha = 0.5$, $S = 100$.

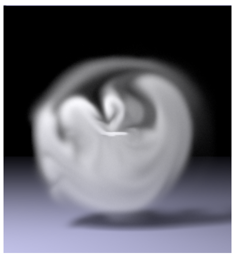

(a) $t = 50$

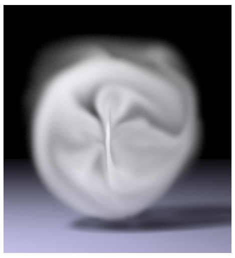

(b) $t = 200$


(c) $t = 300$


(d) $t = 450$


(e) $t = 550$


(f) $t = 650$


(g) $t = 800$

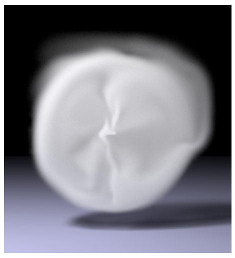

(h) $t = 1100$

Figure 4.10: Rendered densities for the disk simulation. The fluid gradually fills the region corresponding to the tensor field in Fig. 4.8(d). The fluid starts slowly, building momentum until it reaches a "final" state, where the shape is already formed but the fluid continues its circling motion in a stable manner. Frame (h) shows this state. Simulation parameters: $\Delta t = 0.05$, $\beta = 3.0$, $\alpha = 0.5$, $S = 100$.

shape. Fig. 4.13 shows some frames of our simulation for the wafer field.

In Fig. 4.14 we can see the result of simulating a twister field. In this case, we applied a constant external force at the basis of the twister, which is also the density insertion point. The force is directed towards the top. Despite being vertically directed, the linear tensors that form the core of the twister produce a swirling motion, generating the expected vortex. The planar tensors at the boundaries of the tensor field are used to better contain the fluid. We also added a spiral of linear tensors around the twister so we could introduce some finer detail to the fluid motion at the boundaries.

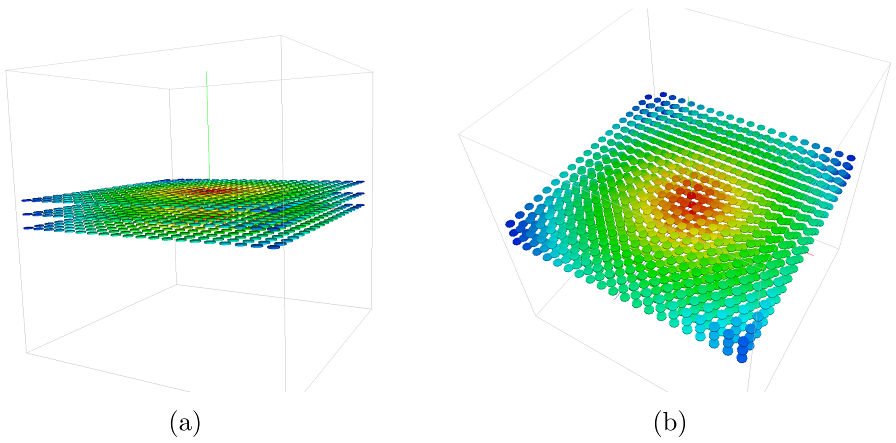


Figure 4.11: $20 \times 20 \times 20$ wafer tensor field, composed solely of planar tensors. Near the center, the tensors have higher norm. The goal was to diffuse the fluid radially and keep it contained in the planar layers.

Finally, we created a spherical field. It follows a similar rationale used in the twister field. The interior of the sphere is composed of linear tensors, while the surface is made of planar tensors. The linear tensors here also have the objective of producing circular motion around the sphere. We applied radial forces in the center of the sphere, propelling the fluid outwards. The tensor advection generates velocities that encircle the sphere, forcing the fluid to assume a circular motion. Fluid is inserted throughout the whole volume of the sphere. We can notice some fluid moving around the surface. Figs. 4.15 and 4.16 shows, respectively, the tensor field and some frames of its simulation.

4.3 General 3D tensor fields

A well known 3D dataset for testing DT-MRI based methods is the helix tensor field [1]. This synthetic field represent continous trajectories and patches that are not intended for keeping the fluid inside the helix volume. However, it is possible to use our method to partially visualize the volume without any change or processing

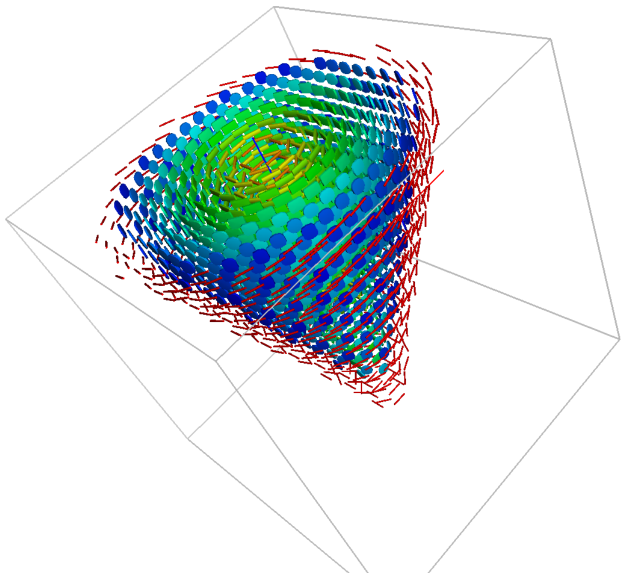
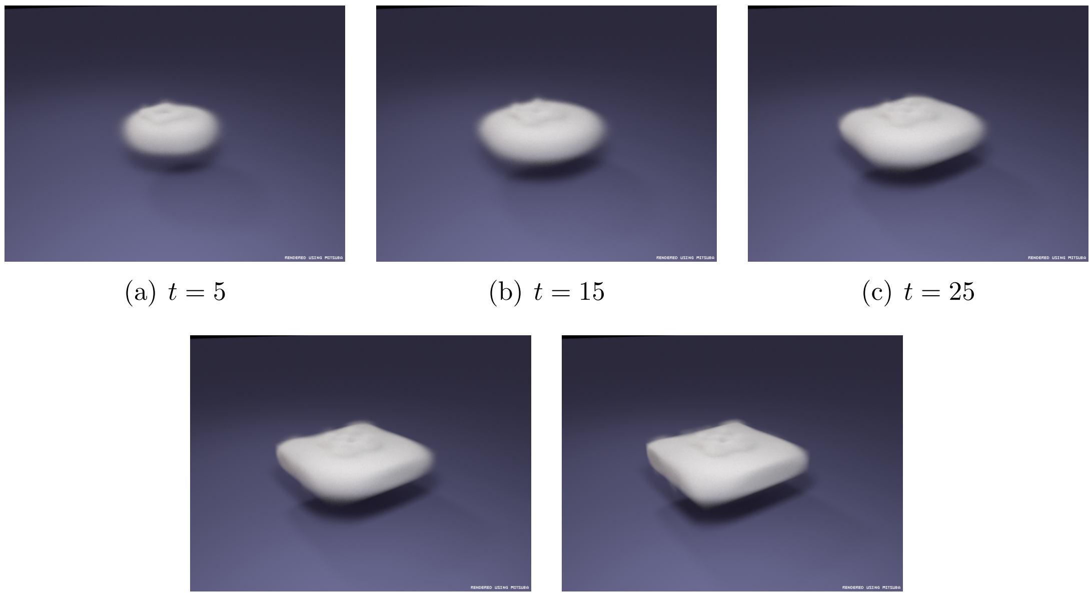


Figure 4.12: $20 \times 20 \times 20$ twister field. This field consists of linear tensors oriented so that a vortex can be formed in its core. At the boundaries, planar tensors are used to contain the fluid and an additional set of external linear tensors is used to generate finer motion detail around the twister.



(a) $t = 5$

(b) $t = 15$

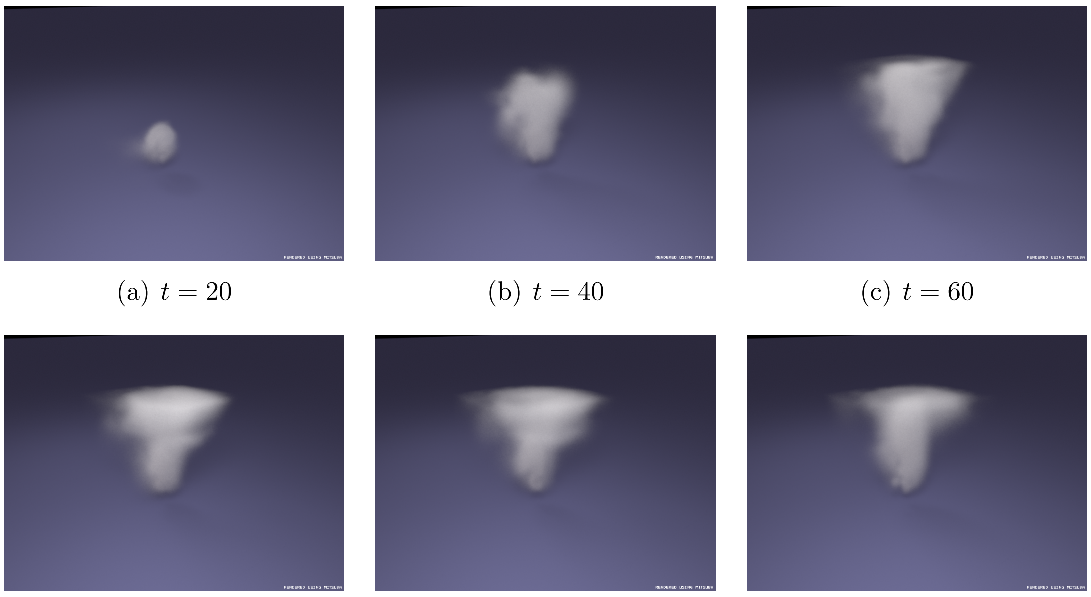
(c) $t = 25$

(d) $t = 35$

(e) $t = 45$

Figure 4.13: Rendered densities for the wafer simulation. Parameters: $\Delta t = 0.1$, $\beta = 1.3$, $\alpha = 0.5$, $S = 100$.

of the original field. To do this, we inserted 6 sources of density throughout the medial axis of the volume. This was needed to form a completely connected flow inside the helix since a long range current is not possible with only one source. The field trajectories tend to eject the density outside the volume. Notice that a large damping factor ($\alpha = 10.0$) is needed to visualize the field. The result, however, does



(a) $t = 20$ (b) $t = 40$ (c) $t = 60$

(d) $t = 80$ (e) $t = 100$ (f) $t = 120$

Figure 4.14: Rendered densities for the twister simulation. Parameters: $\Delta t = 0.2$, $\beta = 5.0$, $\alpha = 0.1$, $S = 100$.

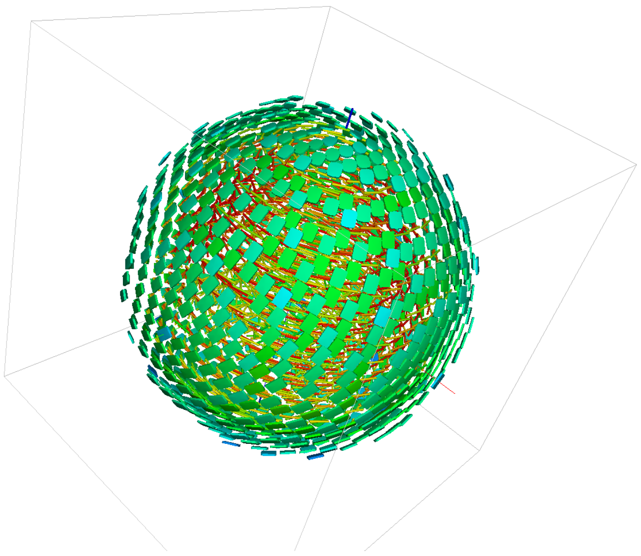
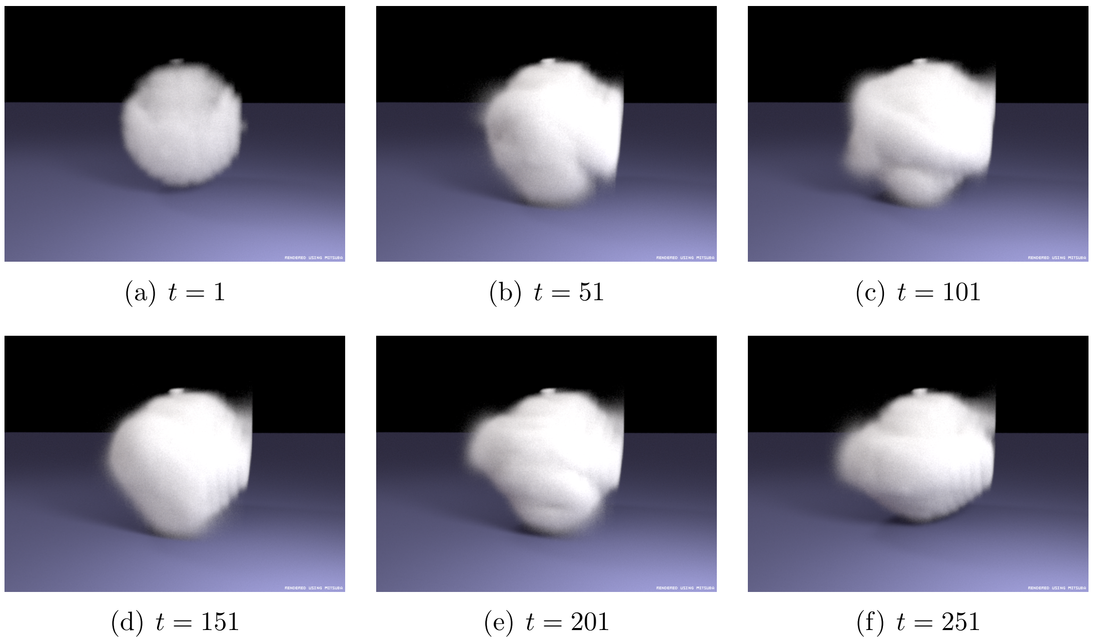


Figure 4.15: $20 \times 20 \times 20$ sphere field. The interior of the sphere is composed of linear tensors. These tensors are surrounded by a shell of planar tensors.

not show the details of the field trajectories.

The simulation result shown in Fig. 4.17 is just a selected frame where the helix is visibly recognizable. Unfortunately, the simulation is not able to keep the helix connected through all iterations, as we did in other experiments. The 3D tensor fields are much more challenging and require further improvements so we can generate volumes made of fluid.


(a) $t = 1$ (b) $t = 51$ (c) $t = 101$

(d) $t = 151$ (e) $t = 201$ (f) $t = 251$

Figure 4.16: Rendered densities for the sphere simulation. Parameters: $\Delta t = 0.1$, $\beta = 3.42$, $\alpha = 0.18$, $S = 1$.

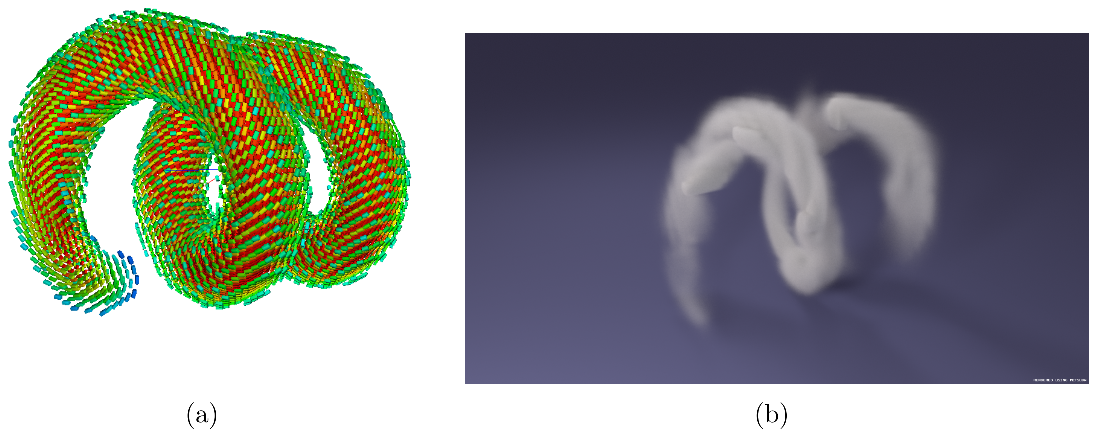

(a) (b)

Figure 4.17: Helix tensor field visualization with direct application of our method. Parameters $\Delta t = 0.05$, $\beta = 6.0$ and $\alpha = 10.0$. (a) $38 \times 39 \times 40$ tensor field. (b) iteration 651.

4.4 Known issues

As we mentioned in the previous section, the results of our method for 3D fields still needs improvements. Additional control mechanisms may be required in order to acquire the level of control seen in the 2D simulations, since in 3D we have a higher number of degrees of freedom. When simulating fluid for the sphere field, for example, fluid can be seen circling the sphere's surface in thin sheets, but it fails to enclose the sphere as a whole unless we provide enough points of density insertion.

By carefully adjusting simulation parameters, we can obtain results such as those demonstrated in Fig. 4.16, but in most cases we end up with a deformed sphere.

Further investigation must also be done regarding the application of the customized projection. Our proposed projection is mathematically consistent, but, in this thesis, we did not analyze what are the conditions the tensor field has to abide to in order for the projection to work in any situation. For example, discontinuities in the field may lead to problems, such as small localized instabilities, which may be aggravated depending on the simulation parametrization.

Chapter 5

Conclusions

This thesis investigated the relationship between symmetric positive-definite tensors and fluids, and how to couple these two elements to produce controlled fluid simulations. We presented a mathematical model for simulating fluid flow based on tensor fields, by proposing a set of partial differential equations that adapt the classic Navier-Stokes equations. This resulted in the development of a customized version of the Stable Fluids [5] method, where tensors dictate fluid behavior in every step of the simulation. We provided the means for changing both advection and diffusion, treating tensors as a property of the medium, in order to gain better control over the fluid. We discussed two working forms of altering advection and pondered on their advantages and disadvantages. For the diffusion, we showed how to construct Laplacian masks for implementing anisotropic diffusion, an open problem still under intense research in the physics community. Naturally, due to the intricacies inherent to the Navier-Stokes equations, controlling the fluid proves to be challenging, since the advection process imposes some dynamics of its own. So, any modifications to the equations must be done with care to not lose physical plausibility. The goal here was to obtain control without losing the fluid-like effect. Finding this balance is not an easy task.

As shown in the previous section, results demonstrate that it is possible to use tensors to direct fluids through intended paths. By carefully designing our tensor fields, we can produce interesting animation sequences or model fluidic shapes, like in the disk example. As the method produces well-defined flow paths influenced by the tensor field, this kind of simulation is also potentially useful in the scope of tensor field visualization, where highlighting colinear paths is usually the focus of interest. Tensor fields used in these cases, however, tend to be extremely noisy. Some preprocessing would probably be necessary in order to obtain satisfactory results. How to control fluids with tensors in a noisy environment still require investigation, and will be addressed in the future.

We also showed the results of our anisotropic diffusion method. The fluid clearly diffuses accordingly, considering the underlying tensor field. As discussed in Section 3.3, we opted for using the finite difference operators supplied by GÜNTER *et al.* [29] and derived Laplacian masks from it. We used a simple averaging scheme between the asymmetric and symmetric operators. This worked well in all our experiments (including 3D), but it would be interesting to devise or test other forms of combination and also more complete discrete Laplacian estimators. We mentioned the problems involving both the symmetric and the asymmetric scheme. The averaging reduces the issues of both, but does not fully eliminate them. So, we intend to explore alternatives to our current diffusion setting. This is extremely relevant, since we conjecture that diffusion may be paramount to ensure that densities fill regions of the volume that are not reached by the advection alone.

The projection is another topic of concern of this thesis. We provided a formulation to also adapt the original one, since we believe that the standard projection may hamper the tensor effect on the tensor-transformed velocity field. However, in Section 4.4, we mentioned we still have to investigate the necessary characteristics of the tensor field so that it does not harm the projection. This problem shall be addressed shortly, since the new projection may also contribute to better control the fluid.

Although our method can produce interesting results in 2D simulations, we saw that it still needs improvements in 3D. For now, it excessively relies on parameter configuration to obtain acceptable results. Although every fluid simulation is anchored on a proper choice of parameters, in the 2D case we are able to produce good results with a larger array of parameter combinations. We wish to explore ways of reducing the excessive dependence on parameter configuration for the 3D simulations, enabling our method to work in less controlled scenarios.

Another interesting topic of discussion is the design of tensor fields. Our method cannot produce interesting animations if the tensor field is not adequately constructed. An interesting line of work is to investigate the best ways to construct these tensor fields. A well-thought tensor field may help the proposed method on obtaining good simulations.

Finally, it would be interesting to apply our method to different contexts in the future. Porous flow simulations are a possibility, considering the permeability tensor is exactly of the same form as the tensors we used in this work. Also, as already discussed throughout the text, the presented method may be very useful for enhancing DT-MRI tensor field visualizations. The poor resolution and information cluttering of these tensor fields often lead medical practitioners to resort to visualization techniques to better understand the data. However, these are also the reasons why it is difficult to provide good visualizations. We hope an adapted version of our method can also contribute to this scenario.

Bibliography

[1] "Diffusion tensor MRI datasets". `http://www.sci.utah.edu/~gk/DTI-data/`, 05 2017.

[2] KINDLMANN, G. "Superquadric Tensor Glyphs". In: *Proceedings of IEEE TVCG/EG Symposium on Visualization 2004*, pp. 147–154, May 2004.

[3] VAN ES, B., KOREN, B., DE BLANK, H. J. "Finite-difference schemes for anisotropic diffusion", *Journal of Computational Physics*, v. 272, pp. 526–549, 2014.

[4] WESTIN, C., PELED, S., GUDBJARTSSON, H., et al. "Geometrical diffusion measures for MRI from tensor basis analysis". In: *Proceedings of ISMRM*, v. 97, p. 1742, 1997.

[5] STAM, J. "Stable fluids". In: *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pp. 121–128. ACM Press/Addison-Wesley Publishing Co., 1999.

[6] FOSTER, N., FEDKIW, R. "Practical animation of liquids". In: *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pp. 23–30. ACM, 2001.

[7] FEDKIW, R., STAM, J., JENSEN, H. W. "Visual simulation of smoke". In: *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pp. 15–22. ACM, 2001.

[8] BRIDSON, R. *Fluid simulation for computer graphics.* CRC Press, 2015.

[9] HUANG, Z., GONG, G., HAN, L. "Physically-based smoke simulation for computer graphics: a survey", *Multimedia Tools and Applications*, v. 74, n. 18, pp. 7569–7594, 2015.

[10] TREUILLE, A., MCNAMARA, A., POPOVIĆ, Z., et al. "Keyframe control of smoke simulations". In: *ACM Transactions on Graphics (TOG)*, v. 22, pp. 716–723. ACM, 2003.

[11] BROWNING, M., BARNES, C., RITTER, S., et al. "Stylized keyframe animation of fluid simulations". In: *Proceedings of the Workshop on Non-Photorealistic Animation and Rendering*, pp. 63–70. ACM, 2014.

[12] SHECHTMAN, E., RAV-ACHA, A., IRANI, M., et al. "Regenerative morphing". In: *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pp. 615–622. IEEE, 2010.

[13] DARABI, S., SHECHTMAN, E., BARNES, C., et al. "Image melding: Combining inconsistent images using patch-based synthesis." *ACM Trans. Graph.*, v. 31, n. 4, pp. 82–1, 2012.

[14] MCNAMARA, A., TREUILLE, A., POPOVIĆ, Z., et al. "Fluid control using the adjoint method". In: *ACM Transactions On Graphics (TOG)*, v. 23, pp. 449–456. ACM, 2004.

[15] BONILLA, D., VELHO, L., NACHBIN, A., et al. "Fluid warping". In: *Proceedings of the IV Iberoamerican Symposium in Computer Graphics*, 2009.

[16] BONILLA, D., VELHO, L. "Control methods for fluid-based image warping". In: *Proceedings of WTD-Workshop of Theses and Dissertations-SIBGRAPI 2011*, 2011.

[17] FATTAL, R., LISCHINSKI, D. "Target-driven smoke animation". In: *ACM Transactions on Graphics (TOG)*, v. 23, pp. 441–448. ACM, 2004.

[18] RENHE, M. C., OLIVEIRA, A., ESPERANÇA, C., et al. "Enhanced Target Driven Smoke Morphing". In: *Graphics, Patterns and Images (SIBGRAPI), 2012 25th SIBGRAPI Conference on*, pp. 213–220. IEEE, 2012.

[19] MANTEAUX, P.-L., VIMONT, U., WOJTAN, C., et al. "Space-time sculpting of liquid animation". In: *Proceedings of the 9th International Conference on Motion in Games*, pp. 61–71. ACM, 2016.

[20] KIM, Y., MACHIRAJU, R., THOMPSON, D. "Path-based control of smoke simulations". In: *Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pp. 33–42. Eurographics Association, 2006.

[21] YUAN, Z., CHEN, F., ZHAO, Y. "Pattern-guided smoke animation with lagrangian coherent structure". In: *ACM transactions on graphics (TOG)*, v. 30, p. 136. ACM, 2011.

[22] YANG, B., LIU, Y., YOU, L., et al. "A unified smoke control method based on signed distance field", *Computers & Graphics*, v. 37, n. 7, pp. 775–786, 2013.

[23] SATO, S., DOBASHI, Y., IWASAKI, K., et al. "Deformation of 2D flow fields using stream functions". In: *SIGGRAPH Asia 2014 Technical Briefs*, p. 4. ACM, 2014.

[24] SATO, S., DOBASHI, Y., YUE, Y., et al. "Incompressibility-preserving deformation for fluid flows using vector potentials", *The Visual Computer*, v. 31, n. 6-8, pp. 959–965, 2015.

[25] SATO, S., DOBASHI, Y., NISHITA, T. "A combining method of fluid animations by interpolating flow fields". In: *SIGGRAPH ASIA 2016 Technical Briefs*, p. 4. ACM, 2016.

[26] HAGEMAN, N. S., TOGA, A. W., NARR, K. L., et al. "A diffusion tensor imaging tractography algorithm based on Navier–Stokes fluid mechanics", *IEEE transactions on medical imaging*, v. 28, n. 3, pp. 348–360, 2009.

[27] RENHE, M. C., DE SOUZA FILHO, J. L., VIEIRA, M. B., et al. "Tensor field visualization using Eulerian fluid simulation". In: *Computational Science and Its Applications–ICCSA 2013*, Springer, pp. 332–347, 2013.

[28] BONELLE, J. *Compatible Discrete Operator schemes on polyhedral meshes for elliptic and Stokes equations.* Tese de Doutorado, Université Paris-Est, 2014.

[29] GÜNTER, S., YU, Q., KRÜGER, J., et al. "Modelling of heat transport in magnetised plasmas using non-aligned coordinates", *Journal of Computational Physics*, v. 209, n. 1, pp. 354–370, 2005.

[30] LIPNIKOV, K., MANZINI, G., SHASHKOV, M. "Mimetic finite difference method", *Journal of Computational Physics*, v. 257, pp. 1163–1227, 2014.

[31] AARNES, J. E., GIMSE, T., LIE, K.-A. "An introduction to the numerics of flow in porous media using Matlab". In: *Geometric modelling, numerical simulation, and optimization*, Springer, pp. 265–306, 2007.

[32] AAVATSMARK, I. "Multipoint flux approximation methods for quadrilateral grids". In: *9th International forum on reservoir simulation, Abu Dhabi*, pp. 9–13, 2007.

[33] LENAERTS, T., ADAMS, B., DUTRÉ, P. “Porous flow in particle-based fluid simulations”, *ACM Transactions on Graphics (TOG)*, v. 27, n. 3, pp. 49, 2008.

[34] VILANOVA, A., ZHANG, S., KINDLMANN, G., et al. “An introduction to visualization of diffusion tensor imaging and its applications”, *Visualization and Processing of Tensor Fields*, pp. 121–153, 2006.

[35] PIERPAOLI, C., BASSER, P. J. “Toward a quantitative assessment of diffusion anisotropy”, *Magnetic resonance in Medicine*, v. 36, n. 6, pp. 893–906, 1996.

[36] EDMUNDS, M., LARAMEE, R. S., MALKI, R., et al. “Automatic stream surface seeding: A feature centered approach”. In: *Computer Graphics Forum*, v. 31, pp. 1095–1104. Wiley Online Library, 2012.

[37] MCLOUGHLIN, T., JONES, M. W., LARAMEE, R. S., et al. “Similarity measures for enhancing interactive streamline seeding”, *Visualization and Computer Graphics, IEEE Transactions on*, v. 19, n. 8, pp. 1342–1353, 2013.

[38] BRAMBILLA, A., HAUSER, H. “Expressive seeding of multiple stream surfaces for interactive flow exploration”, *Computers & Graphics*, v. 47, pp. 123–134, 2015.

[39] MORI, S., VAN ZIJL, P. “Fiber tracking: principles and strategies–a technical review”, *NMR in Biomedicine*, v. 15, n. 7-8, pp. 468–480, 2002.

[40] BEAULIEU, C. “The basis of anisotropic water diffusion in the nervous system–a technical review”, *NMR in Biomedicine*, v. 15, n. 7-8, pp. 435–455, 2002.

[41] MORI, S., CRAIN, B. J., CHACKO, V., et al. “Three-dimensional tracking of axonal projections in the brain by magnetic resonance imaging”, *Annals of neurology*, v. 45, n. 2, pp. 265–269, 1999.

[42] WANG, R., BENNER, T., SORENSEN, A., et al. “Diffusion toolkit: a software package for diffusion imaging data processing and tractography”. In: *Proc Intl Soc Mag Reson Med*, v. 15, 2007.

[43] DELMARCELLE, T., HESSELINK, L. “Visualizing second-order tensor fields with hyperstreamlines”. In: *IEEE Computer Graphics and Applications, Volume 13, Issue 4*, pp. 25–33, Los Alamitos, CA, USA, 1993. IEEE Computer Society Press.

[44] WEINSTEIN, D., KINDLMANN, G., LUNDBERG, E. "Tensorlines: advection-diffusion based propagation through diffusion tensor fields". In: *VIS '99: Proceedings of the conference on Visualization '99*, pp. 249–253, Los Alamitos, CA, USA, 1999. IEEE Computer Society Press. ISBN: 0-7803-5897.

[45] ZHANG, S., DEMIRALP, C., LAIDLAW, D. H. "Visualizing diffusion tensor MR images using streamtubes and streamsurfaces", *Visualization and Computer Graphics, IEEE Transactions on*, v. 9, n. 4, pp. 454–462, 2003.

[46] VILANOVA, A., BERENSCHOT, G., VAN PUL, C. "DTI visualization with streamsurfaces and evenly-spaced volume seeding". In: *Proceedings of the Sixth Joint Eurographics-IEEE TCVG conference on Visualization*, pp. 173–182. Eurographics Association, 2004.

[47] SONG, W.-J., CHEN, W., CHEN, H.-D., et al. "Visualizing diffusion tensor fields on streamsurfaces with merging ellipsoids and LIC texture", *Applied Mathematics-A Journal of Chinese Universities*, v. 29, n. 4, pp. 399–409, 2014.

[48] KONDRATIEVA, P., KRUGER, J., WESTERMANN, R. "The application of GPU particle tracing to diffusion tensor field visualization". In: *Visualization, 2005. VIS 05. IEEE*, pp. 73–78. IEEE, 2005.

[49] KRUGER, J., KIPFER, P., KONCLRATIEVA, P., et al. "A particle system for interactive visualization of 3D flows", *Visualization and Computer Graphics, IEEE Transactions on*, v. 11, n. 6, pp. 744–756, 2005.

[50] LEONEL, G. A., PEÇANHA, J. P., VIEIRA, M. B. "A viewer-dependent tensor field visualization using particle tracing". In: *Proceedings of the 2011 international conference on Computational science and its applications - Volume Part I*, ICCSA'11, pp. 690–705. Springer-Verlag, 2011. ISBN: 978-3-642-21927-6.

[51] DE SOUZA FILHO, J. L. R., RENHE, M. C., VIEIRA, M. B., et al. "A viewer-dependent tensor field visualization using multiresolution and particle tracing". In: *Proceedings of the 12th international conference on Computational Science and Its Applications - Volume Part II*, ICCSA'12, pp. 712–727, Berlin, Heidelberg, 2012. Springer-Verlag. ISBN: 978-3-642-31074-4.

[52] GARYFALLIDIS, E. *Towards an Accurate Brain Tractography*. Cambridge, University of Cambridge, 2013. Disponível em: <https://books.google.com.br/books?id=zbLQoQEACAAJ>.

[53] GARYFALLIDIS, E., BRETT, M., AMIRBEKIAN, B., et al. "Dipy, a library for the analysis of diffusion MRI data", *Frontiers in neuroinformatics*, v. 8, 2014.

[54] GARYFALLIDIS, E., BRETT, M., CORREIA, M. M., et al. "Quickbundles, a method for tractography simplification", *Frontiers in neuroscience*, v. 6, 2012.

[55] ZHANG, S., LAIDLAW, D. H. "Hierarchical clustering of streamtubes", *Brown University, Providence, RI, United States*, 2002.

[56] DING, Z., GORE, J. C., ANDERSON, A. W. "Classification and quantification of neuronal fiber pathways using diffusion tensor MRI", *Magnetic Resonance in Medicine*, v. 49, n. 4, pp. 716–721, 2003.

[57] COROUGE, I., GOUTTARD, S., GERIG, G. "Towards a shape model of white matter fiber bundles using diffusion tensor MRI". In: *Biomedical Imaging: Nano to Macro, 2004. IEEE International Symposium on*, pp. 344–347. IEEE, 2004.

[58] WELDESELASSIE, Y. T., HAMARNEH, G. "DT-MRI segmentation using graph cuts". In: *Medical Imaging*, pp. 65121K–65121K. International Society for Optics and Photonics, 2007.

[59] HLAWITSCHKA, M., GARTH, C., TRICOCHE, X., et al. "Direct visualization of fiber information by coherence", *International journal of computer assisted radiology and surgery*, v. 5, n. 2, pp. 125–131, 2010.

[60] MOBERTS, B., VILANOVA, A., VAN WIJK, J. J. "Evaluation of fiber clustering methods for diffusion tensor imaging". In: *Visualization, 2005. VIS 05. IEEE*, pp. 65–72. IEEE, 2005.

[61] DODERO, L., VASCON, S., GIANCARDO, L., et al. "Automatic white matter fiber clustering using dominant sets". In: *Pattern Recognition in Neuroimaging (PRNI), 2013 International Workshop on*, pp. 216–219. IEEE, 2013.

[62] ZHANG, S., CORREIA, S., LAIDLAW, D. H. "Identifying white-matter fiber bundles in DTI data using an automated proximity-based fiber-clustering

method", *Visualization and Computer Graphics, IEEE Transactions on*, v. 14, n. 5, pp. 1044–1053, 2008.

[63] SILESS, V., MEDINA, S., VAROQUAUX, G., et al. "A comparison of metrics and algorithms for fiber clustering". In: *Pattern Recognition in Neuroimaging (PRNI), 2013 International Workshop on*, pp. 190–193. IEEE, 2013.

[64] YU, H., WANG, C., SHENE, C.-K., et al. "Hierarchical streamline bundles", *Visualization and Computer Graphics, IEEE Transactions on*, v. 18, n. 8, pp. 1353–1367, 2012.

[65] KOLECKI, J. C. *An introduction to tensors for students of physics and engineering.* Relatório técnico, NASA, 2002.

[66] BASSER, P. J., PIERPAOLI, C. "Microstructural and physiological features of tissues elucidated by quantitative-diffusion-tensor MRI", *Journal of magnetic resonance*, v. 213, n. 2, pp. 560–570, 2011.

[67] PIERPAOLI, C., BASSER, P. J. "Toward a quantitative assessment of diffusion anisotropy", *Magnetic resonance in Medicine*, v. 36, n. 6, pp. 893–906, 1996.

[68] FERZIGER, J. H., PERIĆ, M. *Computational methods for fluid dynamics.* 3 ed. , Springer, 2002. ISBN: 3-540-42074-6.

[69] FOX, R. W., MCDONALD, A. T., PRITCHARD, P. J. *Introduction to fluid mechanics.* 6 ed. , John Wiley & Sons, 2004. ISBN: 0-471-20231-2.

[70] EMANUEL, G. *Analytical Fluid Dynamics.* 2 ed. , CRC Press LLC, 2001. ISBN: 0-8493-9114-8.

[71] STAM, J. "Real-time fluid dynamics for games". In: *Proceedings of the game developer conference*, v. 18, p. 25, 2003.

[72] DA SILVA NETO, A. A., RODRIGUES, P. S. S., GIRALDI, G. A., et al. "Animação computacional de fluidos via smoothed particle hydrodynamics". In: *SIBGRAPI '05: Digital Proceedings of the Brazilian Symposium on Computer Graphics and Image Processing*, 2005.

[73] WEAVER, T., XIAO, Z. "Fluid Simulation by the Smoothed Particle Hydrodynamics Method: A Survey." VISIGRAPP 2016-Proceedings of the 11th Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications 2016, 2016.

[74] PRESS, W., FLANNERY, B., TEUKOLSKY, S., et al. *Numerical Recipes in C: The Art of Scientific Computing.* Cambridge University Press, 10 1992. ISBN: 0521431085.

[75] STEINHOFF, J., UNDERHILL, D. "Modification of the Euler equations for "vorticity confinement": Application to the computation of interacting vortex rings", *Physics of Fluids*, v. 6, n. 8, pp. 2738–2744, 1994.

[76] BRIDSON, R. *Fluid Simulation for Computer Graphics.* CRC Press, 2008.

[77] JAKOB, W. "Mitsuba renderer". 2010. http://www.mitsuba-renderer.org.

Appendix A

Anisotropic diffusion masks

The purpose of this appendix is to present the calculations performed to obtain the Laplacian mask used in the anisotropic diffusion method proposed in Section 3.3. These masks were based on the differential operators introduced by GÜNTER *et al.* [29]. We will divide this appendix in three sections. The first two sections present the equations for computing the operators, and the last section is dedicated to constructing the masks.

A.1 Asymmetric scheme

To present the calculation for the asymmetric scheme from [29], let us show again the schematics depicted in Figure A.1.

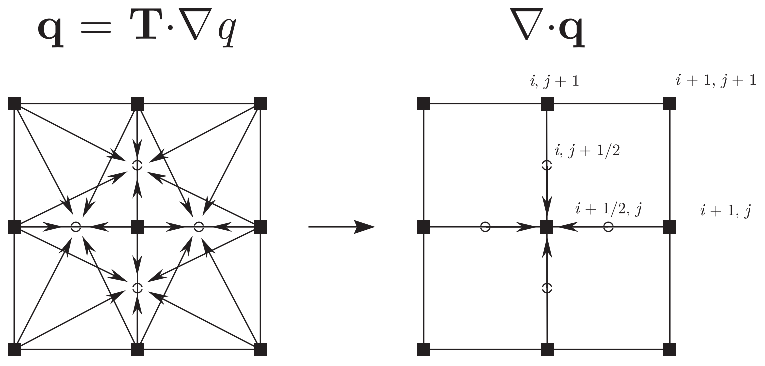


Figure A.1: Asymmetric finite difference scheme. Source: [3]

As defined in the original reference, we first calculate the following set of equations:

$$\nabla q_{i+\frac{1}{2},j} = \left(\frac{q_{i+1,j} - q_{i,j}}{\Delta x}, \frac{q_{i+1,j+1} + q_{i,j+1} - q_{i,j-1} - q_{i+1,j-1}}{4\Delta y} \right)$$
$$\nabla q_{i,j+\frac{1}{2}} = \left(\frac{q_{i+1,j+1} + q_{i+1,j} - q_{i-1,j+1} - q_{i-1,j}}{4\Delta x}, \frac{q_{i,j+1} - q_{i,j}}{\Delta y} \right)$$
$$\nabla q_{i-\frac{1}{2},j} = \left(\frac{q_{i,j} - q_{i-1,j}}{\Delta x}, \frac{q_{i-1,j+1} + q_{i,j+1} - q_{i,j-1} - q_{i-1,j-1}}{4\Delta y} \right)$$
$$\nabla q_{i,j-\frac{1}{2}} = \left(\frac{q_{i+1,j} + q_{i+1,j-1} - q_{i-1,j} - q_{i-1,j-1}}{4\Delta x}, \frac{q_{i,j} - q_{i,j-1}}{\Delta y} \right)$$

Next, we transform the gradients by the tensor:

$$\mathbf{q}_{i+\frac{1}{2},j} = -\mathbf{T}_{i+\frac{1}{2},j} \cdot (\nabla q_{i+\frac{1}{2},j})^T$$
$$\mathbf{q}_{i,j+\frac{1}{2}} = -\mathbf{T}_{i,j+\frac{1}{2}} \cdot (\nabla q_{i,j+\frac{1}{2}})^T$$
$$\mathbf{q}_{i-\frac{1}{2},j} = -\mathbf{T}_{i-\frac{1}{2},j} \cdot (\nabla q_{i-\frac{1}{2},j})^T$$
$$\mathbf{q}_{i,j-\frac{1}{2}} = -\mathbf{T}_{i,j-\frac{1}{2}} \cdot (\nabla q_{i,j-\frac{1}{2}})^T$$

Finally, we calculate the divergence of the vector $\mathbf{q}$, which is exactly the anisotropic diffusion equation $\nabla \cdot (\mathbf{T}\nabla q)$.

$$\nabla \cdot \mathbf{q} = \frac{(q_1)_{i+\frac{1}{2},j} - (q_1)_{i-\frac{1}{2},j}}{\Delta x} + \frac{(q_2)_{i,j+\frac{1}{2}} - (q_2)_{i,j-\frac{1}{2}}}{\Delta y} \tag{A.1}$$

A.2 Symmetric scheme

The symmetric scheme follow a similar sequence of steps for its calculation. Figure A.2 shows once again the idea of this operator.

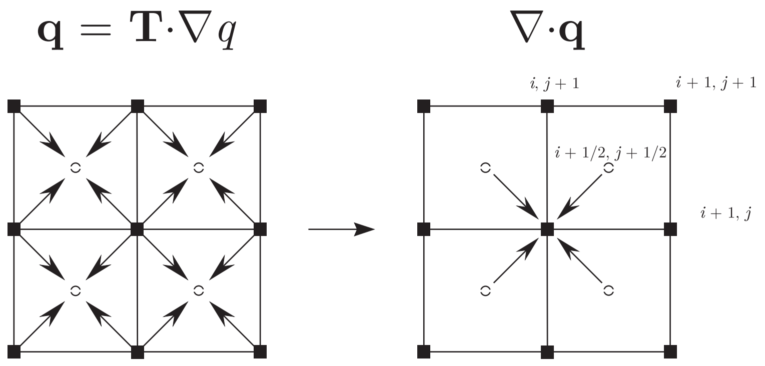


Figure A.2: Symmetric finite difference scheme. Source: [3]

The first step is calculating the gradients for scalar value q. The symmetric scheme, however, calculates them at the diagonal neighbors of the center cell.

$$\begin{aligned}
\nabla q_{i+\frac{1}{2},j+\frac{1}{2}} &= \left(\frac{q_{i+1,j+1}+q_{i+1,j}-q_{i,j+1}-q_{i,j}}{2\Delta x}, \frac{q_{i+1,j+1}+q_{i,j+1}-q_{i+1,j}-q_{i,j}}{2\Delta y}\right) \\
\nabla q_{i+\frac{1}{2},j-\frac{1}{2}} &= \left(\frac{q_{i+1,j-1}+q_{i+1,j}-q_{i,j-1}-q_{i,j}}{2\Delta x}, \frac{q_{i,j}+q_{i+1,j}-q_{i,j-1}-q_{i+1,j-1}}{2\Delta y}\right) \\
\nabla q_{i-\frac{1}{2},j+\frac{1}{2}} &= \left(\frac{q_{i,j+1}+q_{i,j}-q_{i-1,j+1}-q_{i-1,j}}{2\Delta x}, \frac{q_{i-1,j+1}+q_{i,j+1}-q_{i-1,j}-q_{i,j}}{2\Delta y}\right) \\
\nabla q_{i-\frac{1}{2},j-\frac{1}{2}} &= \left(\frac{q_{i,j}+q_{i,j-1}-q_{i-1,j}-q_{i-1,j-1}}{2\Delta x}, \frac{q_{i,j}+q_{i-1,j}-q_{i,j-1}-q_{i-1,j-1}}{2\Delta y}\right)
\end{aligned}$$

Applying the tensor to the gradients, we obtain:

$$\begin{aligned}
\mathbf{q}_{i+\frac{1}{2},j+\frac{1}{2}} &= -\mathbf{T}_{i+\frac{1}{2},j+\frac{1}{2}} \cdot (\nabla q_{i+\frac{1}{2},j+\frac{1}{2}})^T \\
\mathbf{q}_{i+\frac{1}{2},j-\frac{1}{2}} &= -\mathbf{T}_{i+\frac{1}{2},j-\frac{1}{2}} \cdot (\nabla q_{i+\frac{1}{2},j-\frac{1}{2}})^T \\
\mathbf{q}_{i-\frac{1}{2},j+\frac{1}{2}} &= -\mathbf{T}_{i-\frac{1}{2},j+\frac{1}{2}} \cdot (\nabla q_{i-\frac{1}{2},j+\frac{1}{2}})^T \\
\mathbf{q}_{i-\frac{1}{2},j-\frac{1}{2}} &= -\mathbf{T}_{i-\frac{1}{2},j-\frac{1}{2}} \cdot (\nabla q_{i-\frac{1}{2},j-\frac{1}{2}})^T
\end{aligned}$$

And we finish it by computing the divergence of $\mathbf{q}$:

$$\nabla \cdot \mathbf{q} = \frac{(q_1)_{i+\frac{1}{2},j+\frac{1}{2}} + (q_1)_{i+\frac{1}{2},j-\frac{1}{2}} - (q_1)_{i-\frac{1}{2},j+\frac{1}{2}} - (q_1)_{i-\frac{1}{2},j-\frac{1}{2}}}{2\Delta x} + \frac{(q_2)_{i+\frac{1}{2},j+\frac{1}{2}} + (q_2)_{i-\frac{1}{2},j+\frac{1}{2}} - (q_2)_{i+\frac{1}{2},j-\frac{1}{2}} - (q_2)_{i-\frac{1}{2},j-\frac{1}{2}}}{2\Delta y} \tag{A.2}$$

A.3 Diffusion masks

In order to obtain the anisotropic masks to insert in the Gauss-Seidel relaxation performed during the diffusion step, we use the calculations presented in the previous sections and construct a matrix representing the mask. The symmetric mask $S(\mathbf{T})$ is defined as:

$$S(\mathbf{T}) = \begin{bmatrix} \mathrm{sym}(Q_1,\mathbf{T}) & \mathrm{sym}(Q_2,\mathbf{T}) & \mathrm{sym}(Q_3,\mathbf{T}) \\ \mathrm{sym}(Q_4,\mathbf{T}) & \mathrm{sym}(Q_5,\mathbf{T}) & \mathrm{sym}(Q_6,\mathbf{T}) \\ \mathrm{sym}(Q_7,\mathbf{T}) & \mathrm{sym}(Q_8,\mathbf{T}) & \mathrm{sym}(Q_9,\mathbf{T}) \end{bmatrix}, \tag{A.3}$$

where $\mathrm{sym}(Q_k,\mathbf{T})$ is a function that calculates the symmetric operator for tensor $\mathbf{T}$ and the scalar matrix Q_k, with $k = 1..9$, which is a 3×3 matrix with elements $q_{i,j}$, $i = 1..3$, $j = 1..3$ and:

$$q_{i,j} = \begin{cases} 1, & \text{if } 3(i-1)+j = k \\ 0, & \text{otherwise.} \end{cases}$$

The asymmetric mask is computed in the same way, but using the asymmetric function $\text{asym}(Q_k, \mathbf{T})$ instead:

$$A(\mathbf{T}) = \begin{bmatrix} \text{asym}(Q_1, \mathbf{T}) & \text{asym}(Q_2, \mathbf{T}) & \text{asym}(Q_3, \mathbf{T}) \\ \text{asym}(Q_4, \mathbf{T}) & \text{asym}(Q_5, \mathbf{T}) & \text{asym}(Q_6, \mathbf{T}) \\ \text{asym}(Q_7, \mathbf{T}) & \text{asym}(Q_8, \mathbf{T}) & \text{asym}(Q_9, \mathbf{T}) \end{bmatrix}. \tag{A.4}$$