A Framework for Solving Parabolic Partial Differential Equations on Discrete Domains

by

Leticia Mattos Da Silva

B.S., University of California, Los Angeles (2021)

Submitted to the Department of Electrical Engineering and Computer Science

in partial fulfillment of the requirements for the degree of

Master of Science in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2023

Authored by: Leticia Mattos Da Silva
Department of Electrical Engineering and Computer Science
May 19, 2023

Certified by: Justin Solomon
Associate Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by: Leslie A. Kolodziejski
Professor of Electrical Engineering and Computer Science
Chair, Department Committee for Graduate Students

A Framework for Solving Parabolic Partial Differential Equations on Discrete Domains

by

Leticia Mattos Da Silva

Submitted to the Department of Electrical Engineering and Computer Science
on May 19, 2023, in partial fulfillment of the
requirements for the degree of
Master of Science

Abstract

We introduce a framework for solving a class of parabolic partial differential equations on triangle mesh surfaces, including the Hamilton-Jacobi equation and the Fokker-Planck equation. Certain PDE in this class often have nonlinear or stiff terms that cannot be resolved with standard methods on triangle mesh surfaces. To address this challenge, we leverage a splitting integrator combined with a convex optimization step to solve these PDE. Our machinery can be used to compute entropic approximation of optimal transport distances on geometric domains, overcoming the numerical limitations of the state-of-the-art method. In addition, we demonstrate the versatility of our method on a number of linear and nonlinear PDE that appear in diffusion tasks in geometry processing.

Thesis Supervisor: Justin Solomon
Title: Associate Professor of Electrical Engineering and Computer Science

Acknowledgments

This thesis is dedicated to my parents, who gave up everything so I could achieve this dream. I would like to thank my advisor Justin Solomon for his never ending patience and his belief on my potential as a researcher—it was really a marathon and not a sprint. I would like to thank Oded Stein for the mentorship he provided and collaboration on the project from which the body of this thesis is derived. I would also like to thank Nestor Guillen for the very insightful discussions on partial differential equations. I would like to thank my dear friends Brooke McGoldrick, Ryan Deng, and Vinith Suriyakumar for their unconditional support during the time I worked on this thesis. Finally, I would like to acknowledge the generous support of the Schwarzman College of Computing Fellowship supported by Google and the EECS Distinguished Graduate Fellowship for funding my research studies at MIT.

Contents

Introduction

The analysis of partial differential equations (PDE) is a ubiquitous technique in computer graphics, geometry processing, and adjacent fields. In particular, parabolic PDE describe a wide variety of phenomena. For example, instances of the Hamilton-Jacobi equation model the time evolution of flame front propagation and the evolution of functions undergoing nonlinear diffusion. The Fokker-Planck equation describes the evolution of density functions driven by stochastic processes. Both of these equations have recently provided important means to study problems in image processing, computer vision, statistics, and machine learning [Osher et al., 2023, Mokrov et al., 2021, Wang and Hancock, 2008, Leatham et al., 2022]. Hence, methods for accurately and efficiently solving this class of PDE over geometric domains is a central goal in geometry processing.

Myriad numerical algorithms have been proposed for solving PDE in geometry processing. Unfortunately, the most popular algorithms are unsuitable for important regimes, such as capturing infinitesimal or nonlinear phenomena. For instance, the convolutional Wasserstein distance method for barycenter computation [Solomon et al., 2015] is built on tiny amounts of diffusion. The aforementioned method, however, relies on heuristics for choosing diffusion times that are not too small–leading to numerical inaccuracies–and not too large–creating approximations that quantitatively appear wrong. Moreover, nonlinear PDE that describe certain types of flow also require special care. Indeed, explicit integrators used for these PDE require time step restrictions to avoid numerical instability, and implicit integration schemes are no longer equivalent to solving a single linear system of equations, turning out to be too

expensive.

To address these limitations, we propose a framework leveraging a splitting integration strategy and an appropriate spatial discretization to solve parabolic PDE over discrete geometric domains. The splitting allows us to leverage the implicit integration of a well-known PDE, the heat equation, and use a convex relaxation to deal with the challenging piece of the parabolic PDE. Empirically, our method improves performance compared to the state-of-the-art in geometry processing. We apply our framework to challenging parabolic PDE: log-domain heat diffusion, the nonlinear G-equation, and the Fokker-Planck equation, all of which we solve efficiently on a variety of domains and time steps.

Contributions. Our main contributions are:

- A framework to solve linear and nonlinear parabolic PDE on triangle mesh surfaces with efficiency matching that of conventional methods in geometry processing.
- A log-domain diffusion algorithm that improves performance of geometry processing methods relying on tiny amounts of diffusion, demonstrated on optimal transport tasks.
- We apply our framework to the G-equation and the Fokker-Planck equation on triangle grids and surface meshes for various initial conditions and time steps, as well as different choices of vector fields and viscosity values.

Related Work

2.1 Second-order Parabolic PDE

Second-order parabolic PDE are a general extension of the heat equation. The solutions to PDE in this class are in some ways related to the solutions of the heat equation; these equations appear in close applications where the object of interest is the density of a quantity on a domain, evolving forward in time. While the heat equation is a well-studied PDE with a wide array of tools available to solve it on discrete domains, second-order parabolic PDE in general are more challenging, in particular in the presence of additional terms that add nonlinearity or chaotic behaviour.

The typical strategy for spatial discretization of second-order parabolic PDE are Garlekin methods [Evans, 2010]. Common numerical integration techniques are finite difference schemes, including forward Euler, explicit Runge-Kutta formulas, backward Euler, and the Crank-Nicolson method. Still, many PDE in this family cannot be efficiently solved on triangle meshes with these discretization methods. Some of the difficulties that arise include: the strong stiffness of certain equations in this family is not suitable for methods such as Runge-Kutta [Sommeijer et al., 1998]; equations such as the Hamilton-Jacobi do not have a conservative form, making it difficult to write out fluxes for discontinuous Garlekin methods adapted to irregular domains [Yan and Osher, 2011].

2.1.1 Heat diffusion and entropy-regularized optimal transport

The entropy-regularized approximation of transport distances provides a method for solving various optimal transportation problems by realizing them as optimization problems in the space of probability measures equipped with the Kullback-Leibler (KL) divergence. We refer the reader to [Villani, 2003] for a complete introduction to the optimal transportation problem, which roughly seeks to transport all the mass from a source distribution to a target distribution with minimal cost. Benamou et al. [2015] links entropy-regularized transport to minimizing a KL divergence.

Cuturi [2013] proposed a fast computational method to compute transport distances using entropic regularization based on the Sinkhorn algorithm. Solomon et al. [2015] extended this work by showing that the kernel in the Sinkhorn algorithm for the 2-Wasserstein distance can be approximated with the heat kernel, which can be computed using PDE solution techniques over geometric domains. This perspective is also echoed in Schrödinger bridge formulations of transport, whose static form includes an identical substitution of the heat kernel [Léonard, 2014].

2.1.2 G-equation

The G-equation is a level-set Hamilton-Jacobi equation introduced by F.A. Williams in [Buckmaster, 1985] to model turbulent-flame propagation in combustion theory. While some finite difference schemes have succeeded in solving various cases of the G-equation on regular grids [Liu et al., 2013], no study to our knowledge has considered the G-equation on triangle surface meshes. Several challenges arise in the latter setting: for certain values of flow intensity, the constraint on time step—given by the Courant–Friedrichs–Lewy (CFL) condition—is quite restrictive on explicit methods; and the nonlinearity of the equation yields an open problem for implicit methods.

2.1.3 Fokker–Planck equation

The Fokker-Planck equation is a linear parabolic partial differential equation describing the time evolution of the probability density function of a process driven by a stochastic

differential equation (SDE). Processes driven by SDE appear in many contexts, including mathematical finance [Boghosian, 2014], in which methods for numerically solving the Fokker-Planck equation are highly valued. The Fokker-Planck equation also has applications in computer vision and image processing [Smolka and Wojciechowski, 1997, Leatham et al., 2022].

2.2 PDE-driven geometry processing

PDE are a fundamental component of many geometry processing algorithms. PDE-based approaches have been used for surface fairing [Desbrun et al., 1999, Bobenko and Schröder, 2005], surface reconstruction [Duan et al., 2004, Xu et al., 2006], and physics-based simulation [Chen et al., 2013, O'Brien et al., 2002], to cite a few examples. Hence, there has been a considerable amount of work focused on developing accurate and robust PDE solvers. The most popular PDE methods, however, such as the famous "heat method" for geodesic distance approximations [Crane et al., 2013], are concerned with solving linear PDE problems.

2.3 Convex Optimization for Regularized Geodesics

Perhaps the closest related work to this thesis is [Edelstein et al., 2023], which proposes a convex optimization framework for extracting regularized geodesic distances on triangle meshes. We extend this convex optimization strategy to derive one of the steps in a splitting technique to solve a larger class of equations. Their paper generalizes a PDE-based result from [Belyaev and Fayolle, 2020] for distance function estimation to general regularizers under some mild conditions. This kind of variational approach for approximating distance functions on surfaces was first proposed and backed by theoretical results in [Belyaev and Fayolle, 2015].

Mathematical Preliminaries

3.1 General Preliminaries

Let $\mathcal{M} \subset \mathbb{R}^3$ be a surface embedded in $\mathbb{R}^3$, possibly with boundary $\partial\mathcal{M}$; we will use $\hat{n}(x)$ to denote the unit normal to the boundary $\partial\mathcal{M}$ at $x \in \partial\mathcal{M}$. Take ∇ to be the gradient operator and Δ to be the Laplacian operator, with the convention that Δ has nonnegative eigenvalues. We refer the reader to [do Carmo, 1976] for a general introduction to Laplacian operators, which roughly map functions to their total second derivative pointwise.

In this thesis we consider time-varying second-order parabolic PDE of the following form:

$$\frac{\partial u}{\partial t} = -\varepsilon\Delta u + H(x, u(x), \nabla u(x)) \tag{3.1}$$

where the unknown function $u(x,t)\colon \mathcal{M} \times [0,\infty) \to \mathbb{R} \times \mathbb{R}_+$ is twice differentiable on $\mathcal{M}$, $\varepsilon \geq 0$, and $H(x,p,q)$ is jointly convex in $(p,q) \in \mathbb{R} \times \mathbb{R}^3$ for fixed $x \in \mathcal{M}$.

We tackle the Cauchy problem for (3.1), that is, with a prescribed initial condition $u(x,0) = u_0(x)$, where $u_0\colon \mathcal{M} \to \mathbb{R}$ is some scalar function, and boundary conditions determining the behavior of $u(x,t)$ at $x \in \partial\mathcal{M}$. Two common choices for boundary conditions are *Dirichlet* conditions, which prescribe the values of $u(x,t)$ for all $(x,t) \in \partial\mathcal{M} \times (0,\infty)$, and *Neumann* conditions, which specify $\nabla u(x,t) \cdot \hat{n}(x) \equiv 0$ for all $x \in \partial\mathcal{M}$.

The parameter $\varepsilon \geq 0$ in (3.1) is called the viscosity parameter. As $\varepsilon \to 0$, the

function u converges to a solution of

$$\frac{\partial u}{\partial t} = H(x, u(x), \nabla u(x)), \tag{3.2}$$

which is known as the Hamilton-Jacobi equation [Evans, 2010]. This convergence result is given by the stochastic differential game method in [Fleming and Souganidis, 1989].

3.2 Example PDE

Our framework arose in our study of functions $u(x,t)$ satisfying the following PDE, which arises in the subsequent proposition:

$$\frac{\partial u}{\partial t} = -\Delta u + \|\nabla u\|_2^2. \tag{3.3}$$

Note that for this equation, we have $H(x,p,q) = \|q\|_2^2$.

Proposition 3.2.1. *Suppose $v(x,t)$ is such that $v(x,t) > 0$ for all $(x,t) \in \mathcal{M} \times [0,\infty)$ and v satisfies the heat equation:*

$$\frac{\partial v}{\partial t} = -\Delta v. \tag{3.4}$$

Define $u(x,t) := \log v(x,t)$. Then, $u(x,t)$ satisfies (3.3). Moreover, if v satisfies Dirichlet or Neumann boundary conditions, then u satisfies the same boundary conditions up to applying log*. In sum, u satisfies the boundary conditions satisfied by v up to applying* log*.*

This proposition is a direct result of the chain rule. It only applies to strictly positive functions $v(x,t)$; by the maximum principle, it is sufficient to check $v(x,0) > 0$ to guarantee this condition for all $t \in [0,\infty)$. Also, u is a supersolution to the heat equation since clearly $\partial u/\partial t \geq -\Delta u$.

The overarching theme of this thesis is that the same procedure that solves (3.3)

can be used to handle other parabolic problems. In particular, we consider three choices of H:

3.2.1 Nonlinear diffusion

First, motivated by equation (3.3), we consider $H(x, p, q) = \|q\|_2^2$. In §6.1, we will show that this Hamilton-Jacobi equation and Proposition 3.2.1 play a key role in overcoming the limitations faced by methods relying in small amounts of diffusion to compute entropy regularized transport distances on discrete domains.

3.2.2 G-equation

Second, we consider $H(x, p, q) = -\Phi(x) \cdot q + \|q\|_2$, which corresponds to the G-equation (see §2.1.2):

$$\frac{\partial u}{\partial t} = -\Phi(x) \cdot \nabla u + \|\nabla u\|_2 - \varepsilon \Delta u. \tag{3.5}$$

When $\varepsilon = 0$, this equation is known as the *inviscid* G-equation. For $\varepsilon > 0$, it is known as the *viscous* G-equation. Certain types of incompressible flows are of special interest in applications where the G-equation is found. In particular, the two-dimensional cellular flow is defined as

$$\Phi(x) = (-A \sin(2\pi x) \cos(2\pi y),\ A \cos(2\pi x) \sin(2\pi y)). \tag{3.6}$$

3.2.3 Fokker-Planck equation

Finally, we consider $H(x, p, q) = p \nabla \Phi(x) + \Phi(x) q$, which corresponds to the Fokker-Planck equation (see §2.1.3):

$$\frac{\partial u}{\partial t} = u \nabla \Phi(x) + \Phi(x) \cdot \nabla u - \varepsilon \Delta u. \tag{3.7}$$

Here u is the density function of the trajectories of the following SDE:

$$\mathrm{d}x = -\nabla \Phi(x) \mathrm{d}t + \sqrt{2\varepsilon}\, \mathrm{d}W, \tag{3.8}$$

where W is a Wiener process. We refer the reader to [Medved et al., 2020] for a review on the relationship between equations (3.7) and (3.8). The vector field Φ is typically known as the drift vector and ε as the diffusion coefficient, but we will call the latter the viscosity parameter for consistency throughout this thesis.

Method

Let $\Omega = (V, E, F)$ be a triangle mesh with vertices $V \subset \mathbb{R}^3$, edges $E \subseteq V \times V$, and triangular faces $F \subset V \times V \times V$. In this section, we develop a general approach to approximating solutions to equations of the form (3.1) on triangle meshes, given $u(x, 0)$ discretized on M using one value per vertex, $u_0 \in \mathbb{R}^{|V|}$. First, we describe our method for time integration, and then, we define a spatio-dicretization strategy suitable to the different terms associate with PDE of the form in (3.1).

4.1 Time Integration

In what follows, we give a brief overview of classical time integration approaches and outline a method to address the challenges faced by classical approaches when solving second-order parabolic PDE.

4.1.1 Classical approaches

Two typical means of solving PDE in time are explicit time integration, such as the *forward Euler* method, and implicit integration, such as the *backward Euler* method. Forward Euler often imposes a strict restriction (upper bound) on h for stability. Runge–Kutta and other variations of this integrator can improve stability and accuracy of forward Euler integration, but almost all require that we take many small steps ($h \ll 1$) to avoid introducing instability. Backward Euler is unconditionally stable and first-order accurate in h, which often suffices for small $h > 0$. If the PDE

contains a nonlinear term, however, backward Euler and related implicit integration schemes are no longer equivalent to solving a single linear system of equations; in this case, implicit integration leads to a nonlinear root-finding problem.

4.1.2 Strang splitting

We would like to leverage the effectiveness of implicit integration for the simplest version of (3.1) where $H = 0$, but without having to solve a nonlinear system of equations. To this end, we propose using *Strang–Marchuk splitting* (also known as leapfrog or Störmer–Verlet integration), originally proposed in [Strang, 1964, Marchuk, 1988]. Generically, operator splitting methods solve differential equations $dx/dt = f(x) + g(x)$ by alternating steps in which the terms f and g on the right-hand side are treated individually. This splitting scheme is justified by the Lie–Trotter–Kato formula [Trotter, 1959, Kato, 1974]. In our case, we choose a splitting so that one piece of the splitting resembles solving the heat equation and the other benefits from the solution techniques introduced in [Edelstein et al., 2023].

In particular, suppose $u_{n-1} \in \mathbb{R}^{|V|}$ is our estimate of u at time $h(n-1)$. We estimate u_n via the following three steps:

I. First, apply half a time step of implicit heat diffusion, approximating the solution to $\partial u/\partial t = -\varepsilon \Delta u$ at $t = h(n-1) + h/2$:

$$u_n^{(1)} = \left(M + h\frac{\varepsilon L}{2}\right)^{-1} M u_{n-1}. \tag{4.1}$$

II. Next, apply a full time step approximating the solution to $\partial u/\partial t = H(x, u, \nabla u)$, as detailed in §4.1.3, to obtain $u_n^{(2)} \in \mathbb{R}^{|V|}$.

III. Finally, apply a second half-step of implicit heat diffusion:

$$u_n = \left(M + h\frac{\varepsilon L}{2}\right)^{-1} M u_n^{(2)}. \tag{4.2}$$

Note that (4.1) and (4.2) solve the same linear system with different right-hand sides,

allowing us to pre-factor the matrix $M + {}^{h\varepsilon L}/_2$ if we plan to apply our operator more than once.

4.1.3 Strang-Marchuk splitting, step II

We extend the convex optimization problem in [Edelstein et al., 2023] to take a time step of the first-order equation ${}^{\partial u}/_{\partial t} = H(x, u, \nabla u)$, including the case where this is a nonlinear first-order problem. In this section, we describe an implicit integrator leveraging the fact that $H(x, p, q)$ is jointly convex in its arguments (p, q).

In this step, our goal is to obtain $u_n^{(2)}$ by integrating the first-order equation for time h starting at function $u_n^{(1)}$. The simplest implicit integration strategy is *backward Euler* integration, which would solve the following root-find problem:

$$\frac{u_n^{(2)} - u_n^{(1)}}{h} = H(x, u_n^{(2)}, \nabla u_n^{(2)}). \tag{4.3}$$

Here, we will assume the $u_n^{(i)}$ are functions on the surface $\mathcal{M}$, but an identical formulation will apply after spatial discretization in §4.2. This problem is nonlinear whenever H is nonlinear, making it difficult to solve.

Instead, we observe that relaxing the equality in (4.3) to an inequality leads to a *convex* constraint on the unknown function $u_n^{(2)}$. Taking inspiration from [Edelstein et al., 2023], we arrive at the following optimization problem to advance forward in time:

$$u_n^{(2)} = \left\{ \begin{array}{ll} \arg\min_u & \int_{\mathcal{M}} u(x)\,\mathrm{dVol}(x) \\ \text{subject to} & H(x, u, \nabla u) - \frac{1}{h}(u - u_n^{(1)}) \leq 0 \\ & \qquad \text{for all } x \in \mathcal{M}. \end{array} \right. \tag{4.4}$$

Notice that the constraint in (4.4) yields a *supersolution* of the implicit problem (4.3). Moreover, formulation (4.4) is a convex problem for u since it has a linear objective and a pointwise convex constraint.

While we leave formal proof of well-posedness of (4.4) before spatial discretization to future work, we note that works like [Fathi, 2020] justify existence of viscosity solutions to Hamilton-Jacobi equations like (4.3); from there, it may be possible to

extend the argument in [Edelstein et al., 2023] to show that the inequality constraint in our convex relaxation becomes tight. In the spatially-discrete case (see §4.2), well-posedness of our convex program is immediate since it has a linear objective and satisfiable convex constraints. Empirically, we find that our discretization of this time step closely resembles ground-truth when it is available.

4.2 Spatial Discretization

In geometry processing, the most common discretization of the PDE on triangle meshes uses the *finite element method* (FEM). Following [Botsch et al., 2010], take $L \in \mathbb{R}^{|V|\times|V|}$ to be the cotangent Laplacian matrix associated to our mesh (with appropriate boundary conditions), and take $M \in \mathbb{R}^{|V|\times|V|}$ to be the diagonal matrix of voronoi cell areas. Also, take $G \in \mathbb{R}^{3|F|\times|V|}$ to be the matrix mapping vertex scalar values to per face gradients.

We can approximate solutions to equation (3.1) by solving the ordinary differential equation:

$$\frac{du}{dt} = -\varepsilon M^{-1} L u + H(x, u, Gu). \tag{4.5}$$

We will apply the Strang splitting strategy from §4.1.2 to this ODE, yielding steps nearly identical to those outlined in the previous section.

If we use a naïve piecewise-linear finite element method (FEM) to discretize (3.1) spatially, the $\partial u/\partial t$ and $-\varepsilon\Delta u$ terms are associated to the vertices of our mesh, while $H(x, u, Gu)$ includes terms that are associated to both the vertices (x, u) and the triangular faces (Gu). Hence, we propose an alternate spatio-discretization strategy to map any face valued quantities in H to per vertex values. In particular, on a mesh, we reformulate (4.5) via

$$\frac{du_i}{dt} = -\varepsilon(M^{-1} L u(t))_i + \omega_i \sum_{j\sim i} a_j H_{ij}(u_i(t), (Gu)_j(t)). \tag{4.6}$$

where $\omega_i = 1/\sum_{j\sim i} a_j$ and a_i is the area of the triangle j in the one-ring neighborhood of vertex i. The right-hand side of equation (4.6) is convex whenever H_{ij} is jointly

convex in its inputs. Hence, our discretizations below of (3.2), (3.5), and (3.7) are convex since H_{ij} is convex by design whenever H is convex in the continuous case.

Next, we define H discretized as a per-vertex quantity for each of the three example PDE given in sections §3.2.1–§3.2.3.

4.2.1 Nonlinear diffusion

We can now discretize $H = \|q\|_2^2$ on the right-hand side of (3.3) as follows

$$H_i = \omega_i \sum_{j \sim i} \sum_{k=1}^{3} (G_k u \odot G_k u)_j \tag{4.7}$$

where $\odot$ denotes element-wise multiplication, $G_k \in \mathbb{R}^{|F| \times |V|}$ is the matrix mapping a function on the vertices V of our mesh to the kth components of its per-triangle gradient, and W is the diagonal matrix whose elements are the weights ω_i.

4.2.2 G-equation

Similarly, we can discretize the H function for (3.5) as follows

$$H_i = \omega_i \sum_{j \sim i} \Big(\textstyle\sum_{k=1}^{3} (-\Phi_k \odot G_k u)_j + \sqrt{\sum_{k=1}^{3} (G_k u)_j \odot (G_k u)_j} \Big), \tag{4.8}$$

where $\Phi_k \in \mathbb{R}^{|F|}$ is the kth component of a vector field $\Phi(x) \in \mathbb{R}^{3|F|}$.

4.2.3 Fokker-Planck equation

Finally, discretize the divergence operator $\nabla \cdot$ by $G^{\mathrm{T}} M_F \in \mathbb{R}^{|V| \times 3|F|}$, where M_F is the diagonal matrix of triangle areas. The per-vertex discretization of the H functional on the right-hand side of equation (3.7) becomes:

$$H_i = (u \odot (G^{\mathrm{T}} M_F \Phi))_i + \omega_i \sum_{j \sim i} \sum_{k=1}^{3} (\Phi_k \odot G_k u)_j. \tag{4.9}$$

Implementation Details

After introducing the spatial discretization techniques from §4.2, our nonlinear time step (4.4) becomes a finite-dimensional convex optimization problem:

$$\begin{aligned} \text{minimize}_{u_n^{(2)}} \quad & \sum_i \left(u_n^{(2)}\right)_i \\ \text{subject to} \quad & H - u_n^{(2)} + u_n^{(1)} \leq 0, \end{aligned} \tag{5.1}$$

where H denotes the convex function of u and its gradients (see §4.2.1-§4.2.3).

Our implementation uses the CVX software library [Grant and Boyd, 2014, 2008] equipped with the default conic solver SDPT3 [Toh et al., 1999, Tütüncü et al., 2003]. To be concrete, we formulate the convex optimization problem for each of the example parabolic PDE of the form in (3.1) as follows:

Nonlinear diffusion In this case, we have quadratic constraints:

$$\begin{aligned} \arg\min_{u^{(2)}} \quad & \sum_i \left(u_n^{(2)}\right)_i \\ \text{subject to} \quad & \omega_i \sum_{j\sim i} \sum_{k=1}^{3} (G_k u \odot G_k u)_j - u_n^{(2)} + u_n^{(1)} \leq 0, \\ & \qquad\qquad\qquad\qquad\qquad\qquad i = 1, \ldots, |V|. \end{aligned} \tag{5.2}$$

While the program above is convex, the constraints involve general convex quadratic forms. It can be more efficient to convert such programs to second-order cone program (SOCP) standard form, an optimized form for standard solvers like SDPT3. Hence, we reformulate the above quadratic constraint as a second-order cone. In particular,

algebraic manipulation shows that (5.2) is equivalent to the following constraints

$$\left\| \begin{matrix} {}^{(1-z_j)}/_2 \\ (Gu)_j \end{matrix} \right\|_2 \leq \frac{(1+z_j)}{2},\ j = 1, \ldots, |F| \tag{5.3}$$

$$h\ \omega_i \sum_{j\sim i} z_j - \left(u_n^{(2)}\right)_i + \left(u_n^{(1)}\right)_i \leq 0,\ i = 1, \ldots, |V|. \tag{5.4}$$

G-equation In this case, we simply have constraints using norms, which is already in SOCP form. Hence, we can use the conic solver efficiently without further reformulation. We implement our optimization as follows

$$\begin{aligned} \arg\min_{u^{(2)}} \ & \textstyle\sum_i \left(u_n^{(2)}\right)_i \\ \text{subject to} \ & \omega_i \textstyle\sum_{j\sim i} \big(- \sum_{k=1}^{3} (\Phi_k \odot G_k u)_j + \|(Gu)_j\|_2 \big) \\ & -u_n^{(2)} + u_n^{(1)} \leq 0,\ i = 1, \ldots, |V|. \end{aligned} \tag{5.5}$$

Fokker-Planck equation This equation is linear and requires no reformulation. Its implementation becomes:

$$\begin{aligned} \arg\min_{u^{(2)}} \ & \textstyle\sum_i \left(u_n^{(2)}\right)_i \\ \text{subject to} \ & (u \odot (G^{\mathrm{T}} M_F \Phi))_i + \omega_i \textstyle\sum_{j\sim i} \sum_{k=1}^{3} (\Phi_k \odot G_k u)_j \\ & -u_n^{(2)} + u_n^{(1)} \leq 0,\ i = 1, \ldots, |V|. \end{aligned} \tag{5.6}$$

Experiments and Applications

Our experiments were carried out in MATLAB 2023a, on a macOS machine with 32GB memory. All tests were run with solver tolerance $\epsilon^{1/2}$, where $\epsilon = 2.22^{-16}$ is the machine precision. For the unit triangle grid with 131,072 triangles and 66,049 vertices, our implementation takes on average 60 seconds to solve equation (5.2).

6.1 Nonlinear Diffusion

The "log-sum-exp" trick is a standard method used to stabilize numerical algorithms, including the Sinkhorn algorithm when using small amounts of entropy. This computation, however, is not possible using the method adapted to triangle meshes proposed in [Solomon et al., 2015], which corresponds to taking the logarithm of a function undergoing tiny amounts of heat diffusion. We advocate for using our numerical framework and Proposition 3.2.1 to solve (3.3) in order to compute the result of heat diffusion on triangle meshes directly in the logarithmic domain, rather than diffusing in the linear domain and then taking the logarithm.

Figure 6-1: Interpolation between a conference logo and the map of Australia via our log-domain Wasserstein barycenter on a 300×300 triangle grid.

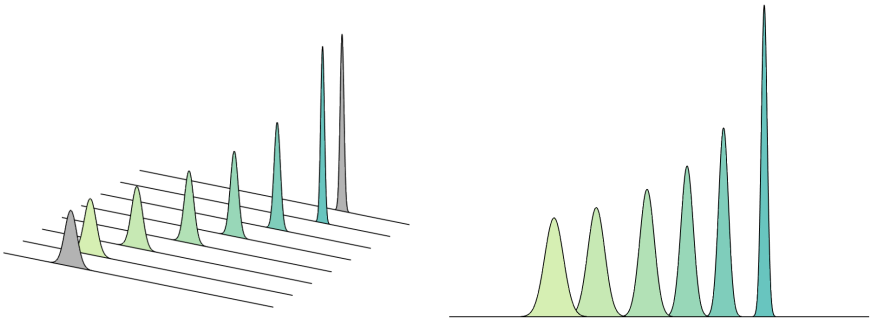

Figure 6-2: Wasserstein barycenters (green) between two distributions on the unit line (grey) with 1,000 elements for varying pairs of weights. Our method obtains these barycenters for a tiny amount of entropy, i.e., $\gamma = 10^{-7}$, whereas the method in Solomon et al. [2015] fails.

In Figure 6-3, we demonstrate how our log-domain diffusion algorithm is capable of computing Wasserstein barycenters for very small amounts of entropy. The method in [Solomon et al., 2015] fails to output a result for entropy values $\gamma = 10^{-7}$ or smaller. Even before its failure cases, the Wasserstein barycenters obtained via convolutional method look qualitatively wrong for a number of entropy values, while our method remains numerically stable and obtains the expected visual results given three Gaussian distributions (shown on the left) as inputs.

A similar result is shown in Figure 6-2, where the displacement interpolation between two Gaussian distributions on the line is shown. For the entropy value $\gamma = 10^{-7}$, our method obtains the barycenters for various pairs of weights as seen on the plot, while the convolutional method in [Solomon et al., 2015] fails to output a result.

Figure 6-4 shows that as the mesh is refined, the solutions to 5.2 converge to the numerical solution of highest resolution. In this experiment, we solve nonlinear diffusion of the logarithm of a Gaussian distribution centered at $(0.5, 0.5)$ on the unit grid with time step $h = 10^{-6}$ for 2^k number of time steps, $k = 1, \ldots, 10$. We start with a triangle unit grid with 128 triangle faces and vary its density via upsampling, each iteration of the upsampling increases the number of triangle faces by 4 times. The mesh with highest resolution has 131,072 triangle faces.

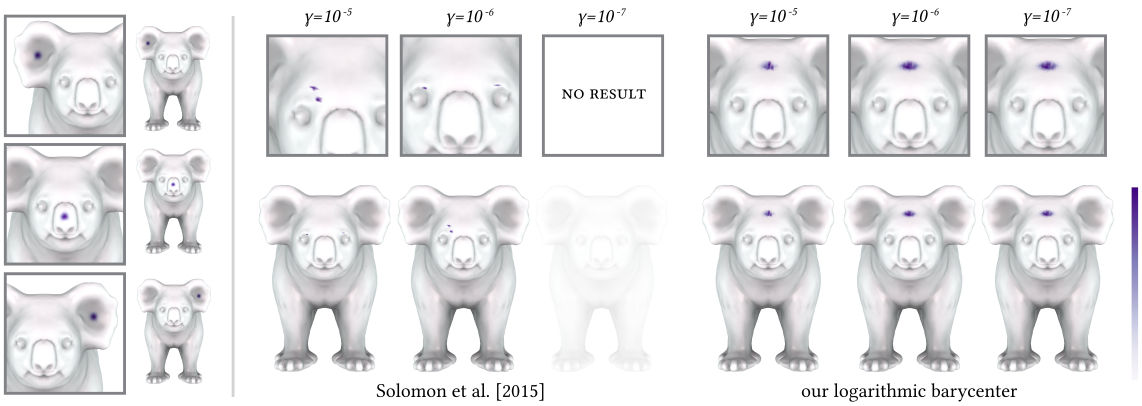

Figure 6-3: Entropic regularization can smooth out true Wasserstein distances creating approximations of the barycenter that qualitatively appear wrong. We present a method for logarithmic diffusion, which leads to sharper interpolants. Here, the Wasserstein barycenter of three input distributions (left) on a mesh with 99,994 triangle faces and 49,999 vertices is shown for different entropy values.

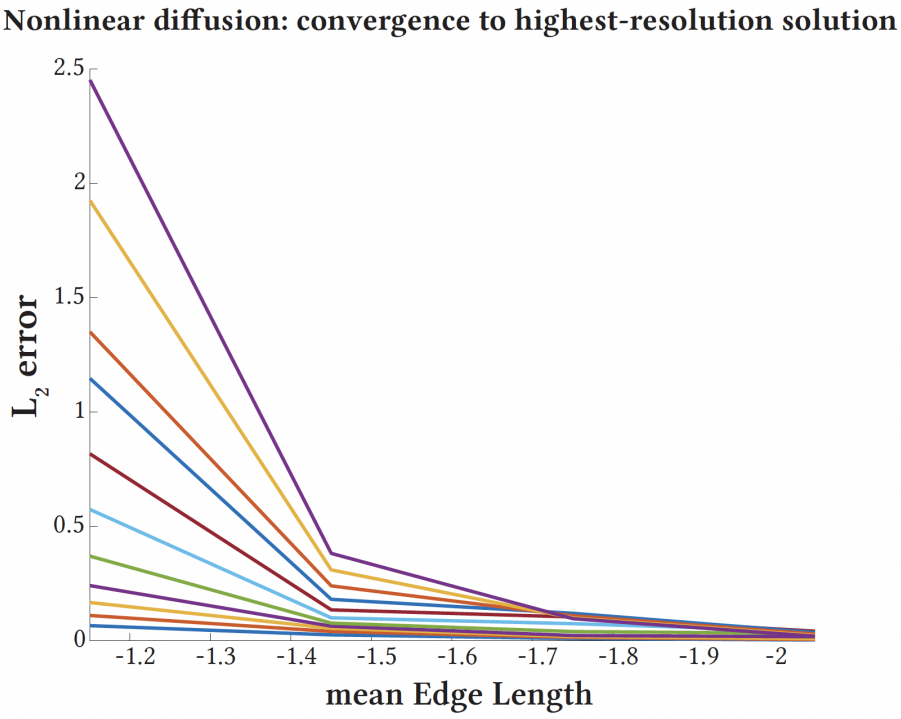

Figure 6-4: Error curves shown for different number of time steps. Each error curve is a function of the mean edge length in meshes obtained via upsampling. The solution converges to the highest resolution solution, and the error increases for larger number of steps given a fixed mean edge length. The mean edge length is shown in logarithmic scale.

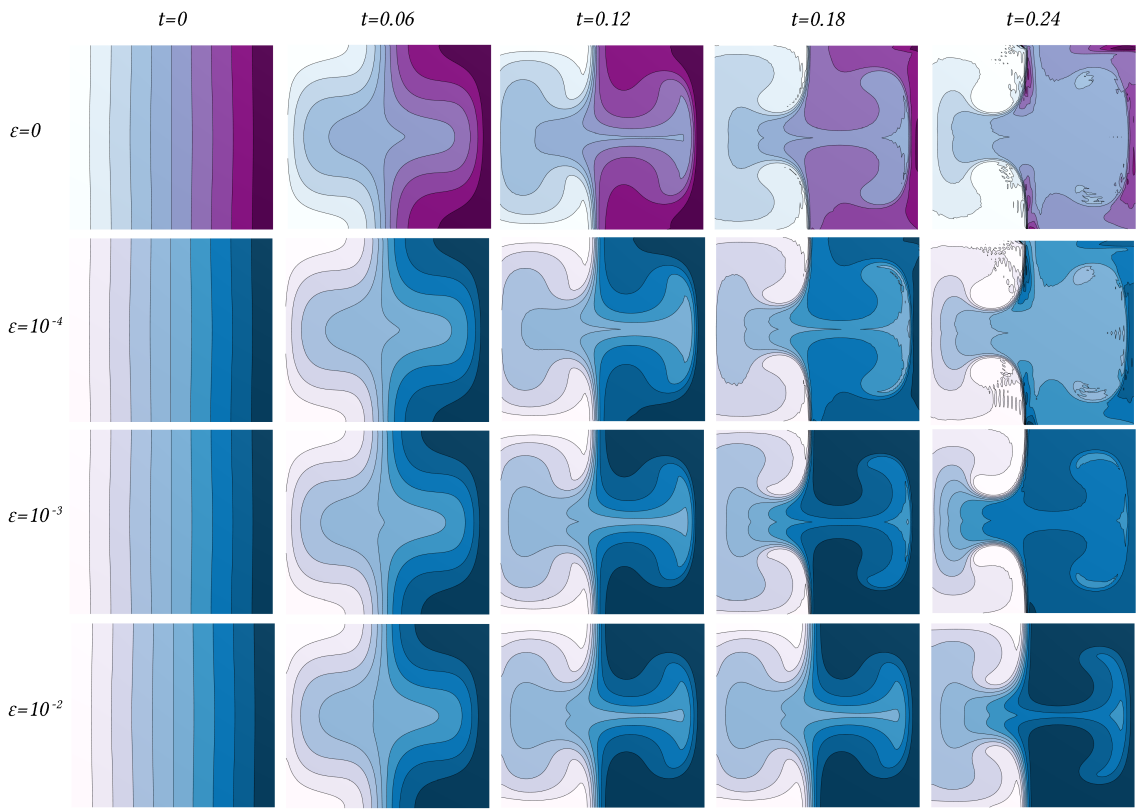

Figure 6-5: Level-sets of the G-equation under cellular flow obtained via our method with varying values of viscosity ε.

6.2 G-Equation

Typically, applications of the G-equation aim to understand the behavior of its function level-sets over time under incompressible flows. In Figure 6-5, we show level-set results for cellular flow (see equation (3.6)) with initial condition $u(x, 0) = P \cdot x$, where $P = (1, 0)$, obtained via our method in a grid with 20,000 triangle faces. We show that our method remains stable under varying amounts of viscosity, including the inviscid case when $\varepsilon = 0$.

In addition, we show in Figure 6-6 that our method can be used to obtain numerical solutions for the G-equation on triangle surface meshes. In this example, we evolve the G-equation with initial condition $u(x, 0) = P \cdot x$, where $P = (1, 0, 0)$, on the discrete torus given a velocity field $\Phi(x) = A(-\sin(2\pi x)\cos(2\pi y), \sin(2\pi x)\sin(2\pi y), 0)$, where $A = 2$. The vicosity value for this experiment is $\varepsilon = 10^{-3}$.

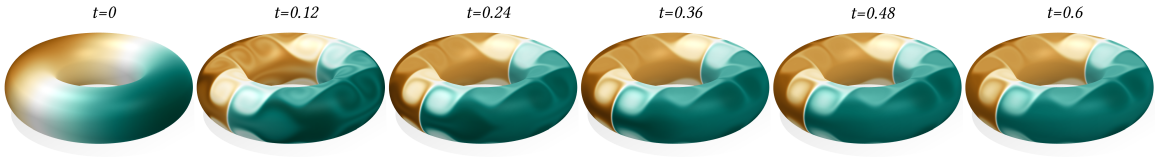

Figure 6-6: Time evolution of the G-equation (3.5) on the discrete torus with 200,000 triangle faces and 100,000 vertices under Kolmogorov flow using our method.

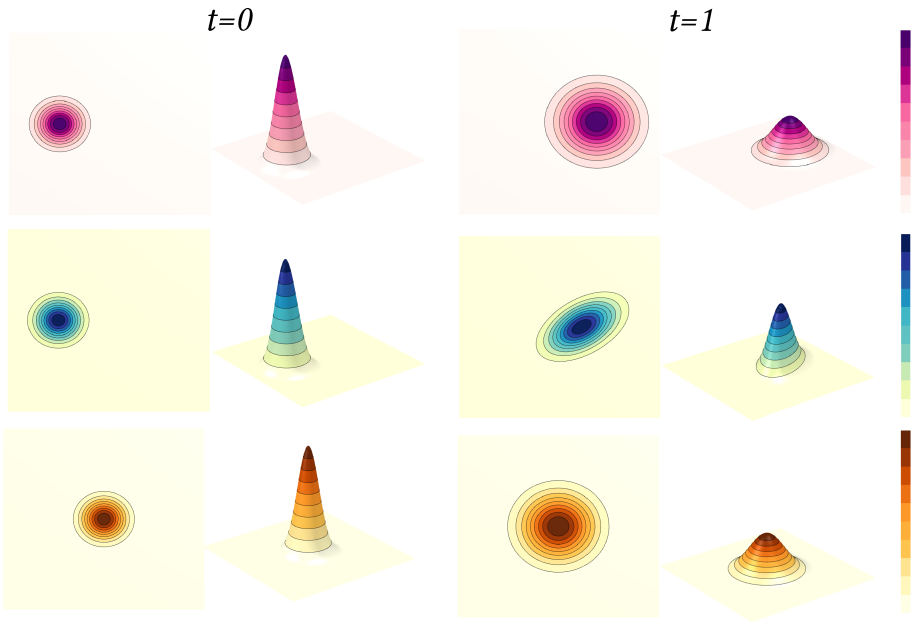

Figure 6-7: Time evolution of the Fokker-Planck equation (3.7) on a 100×100 triangle grid, obtained using our method, under constant drift (top), shear flow (middle), and no drift (bottom).

6.3 Fokker–Planck Equation

Given a velocity vector field $\Phi \in \mathbb{R}^{|V| \times 3|F|}$ defined per face on an underlying computational mesh, we can use our framework to obtain the numerical time evolution of the Fokker-Planck equation.

Following the expository article by Medved et al. [2020], in Figure 6-7, we demonstrate the use of our method for the time evolution of the Fokker-Planck equation in the following three cases: with no drift, under a constant drift in the positive x direction, and under shear flow. The viscosity parameter used in all three cases is $\varepsilon = 10^{-3}$ and the initial condition is a Gaussian distribution centered at $(0.5, 0.5)$; the

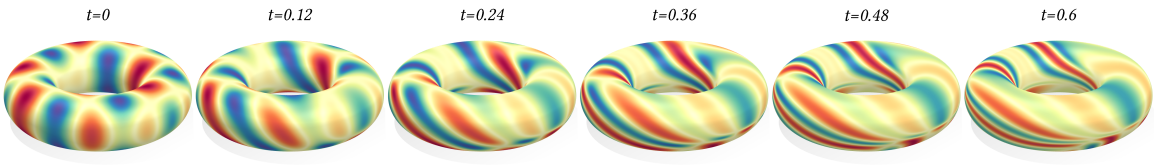

Figure 6-8: Time evolution of the Fokker-Planck equation (3.7) on the discrete torus with 200,000 triangle faces and 100,000 vertices under Kolmogorov flow using our method.

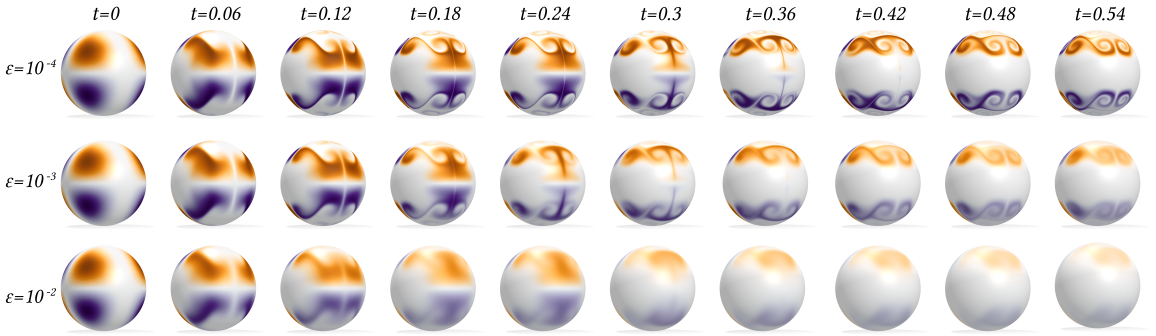

Figure 6-9: The effect of varying the viscosity parameter in the Fokker-Planck equation. Here, the equation is shown evolving in time on the discrete sphere with viscosity $\varepsilon = 10^{-4}$ (top row), $\varepsilon = 10^{-3}$ (middle row), and $\varepsilon = 10^{-2}$ (bottom row).

grid has 20,000 triangle faces.

Figure 6-8 shows the time evolution of the Fokker-Planck for a constant viscosity value of $\varepsilon = 10^{-4}$ on a torus with 200,000 triangle faces and 100,000 vertices. The flow used in this application is known as Komolgorov flow; it is defined as $\Phi(x) = (A\sin(z), 0, 0)$, where for the example shown in the figure $A = 8$; the initial condition is $u(x, 0) = \sin((16/\pi))x\cos((16/\pi)y)$.

In Figures 6-10 and 6-9, we demonstrate the use of our method for the Fokker-Planck equation under varying viscosity with initial condition $u(x, 0) = \sqrt{\frac{40}{32\pi}}\frac{x(-2y^2)z}{r^4}$, where r is the radius of the sphere. The flow chosen in Figure 6-10 is $\Phi(x) = (0, -z, y)$ normalized and scaled along the sphere. In Figure 6-9, the flow is defined by $\Phi(x) = (-A\sin(2\pi x)\cos(2\pi y),\ A\sin(2\pi x)\sin(2\pi y), 0)$, where $A = 8$ for this particular example.

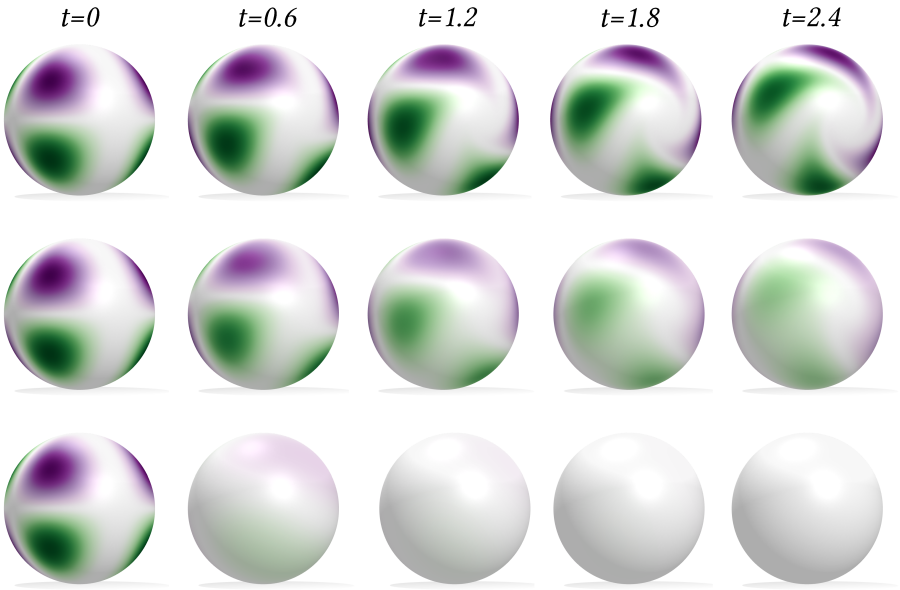

Figure 6-10: The effect of varying the viscosity parameter in the Fokker-Planck equation on a sphere with 81,920 triangle faces and 40,962 vertices. Here, the equation is shown evolving in time on the discrete sphere with viscosity $\varepsilon = 10^{-4}$ (top row), $\varepsilon = 10^{-3}$ (middle row), and $\varepsilon = 10^{-2}$ (bottom row).

Discussion and Conclusion

PDE appear everywhere in geometry processing and second-order parabolic PDE are no exception. While simpler PDE, such as the heat equation, count on a wide array of adequate tools, more general parabolic PDE are a challenge to standard methods of time integration and discretization schemes. Our work establishes an effective time integration and spatio-discretization strategy to solve this class of PDE under mild assumptions on triangle mesh surfaces.

An immediate avenue for future work is to derive an optimization algorithm to solve the problem in (4.4) along the lines of Edelstein et al. [2023] and compare it to our current off-the-shelf solution using CVX. Another line of future work is to extend our results to higher order parabolic equations with Laplacian terms, such as the *Kuramoto–Sivashinsky* equation. Several theoretical and numerical questions remain open to implement implement this extension. Within the realm of second-order parabolic PDE of the form 3.1, future work includes trying to solve system of coupled equations using our framework. In the specific case of the Fokker-Planck equations, further experimental work includes extending results to other versions of the Fokker-Planck equation, and using the solutions obtained via our method together with the relationship to (3.8) to simulate Brownian motion on triangle surface meshes.

Bibliography

Stanley Osher, Howard Heaton, and Samy Wu Fung. A hamilton–jacobi-based proximal operator. *Proceedings of the National Academy of Sciences*, 120(14), 2023.

Petr Mokrov, Alexander Korotin, Lingxiao Li, Aude Genevay, Justin Solomon, and Evgeny Burnaev. Large-scale wasserstein gradient flows, 2021.

Hongfang Wang and Edwin R. Hancock. Probabilistic relaxation labelling using the fokker–planck equation. *Pattern Recognition*, 41(11):3393–3411, 2008.

T. A. Leatham, D. M. Paganin, and K. S. Morgan. X-ray dark-field and phase retrieval without optics, via the fokker-planck equation, 2022.

Justin Solomon, Fernando De Goes, Gabriel Peyré, Marco Cuturi, Adrian Butscher, Andy Nguyen, Tao Du, and Leonidas Guibas. Convolutional Wasserstein distances: Efficient optimal transportation on geometric domains. *ACM Transactions on Graphics (ToG)*, 34(4):1–11, 2015.

Lawrence C Evans. *Partial Differential Equations*, volume 19. American Mathematical Soc., 2010.

B.P. Sommeijer, L.F. Shampine, and J.G. Verwer. Rkc: An explicit solver for parabolic pdes. *Journal of Computational and Applied Mathematics*, 88(2):315–326, 1998.

Jue Yan and Stanley Osher. A local discontinuous galerkin method for directly solving hamilton–jacobi equations. *Journal of Computational Physics*, 230(1):232—244, 2011.

Cédric Villani. *Topics in Optimal Transportation*, volume 58. American Mathematical Soc., 2003.

Jean-David Benamou, Guillaume Carlier, Marco Cuturi, Luca Nenna, and Gabriel Peyré. Iterative bregman projections for regularized transportation problems. *SIAM Journal on Scientific Computing*, 37(2), 2015.

Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. *Advances in neural information processing systems*, 26, 2013.

Christian Léonard. A survey of the schrödinger problem and some of its connections with optimal transport. *Discrete and Continuous Dynamical Systems*, 34(4):1533–1574, 2014.

John D. Buckmaster. *The Mathematics of Combustion.* Society for Industrial and Applied Mathematics, 1985.

Yu-Yu Liu, Jack Xin, and Yifeng Yu. A numerical study of turbulent flame speeds of curvature and strain g-equations in cellular flows. *Physica D: Nonlinear Phenomena*, 243(1):20–31, 2013.

Bruce Boghosian. Fokker–planck description of wealth dynamics and the origin of pareto's law. *International Journal of Modern Physics C*, 25(12), 2014.

Bogdan Smolka and Konrad W. Wojciechowski. Contrast enhancement of badly illuminated images based on gibbs distribution and random walk model. *Computer Analysis of Images and Patterns*, pages 271–278, 1997.

Mathieu Desbrun, Mark Meyer, Peter Schröder, and Alan H. Barr. Implicit fairing of irregular meshes using diffusion and curvature flow. *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, pages 317–324, 1999.

Alexander I. Bobenko and Peter Schröder. Discrete willmore flow. *Eurographics Symposium on Geometry Processing 2005*, 2005.

Ye Duan, Liu Yang, Hong Qin, and Dimitris Samaras. Shape reconstruction from 3d and 2d data using pde-based deformable surfaces. *Computer Vision — ECCV 2004*, pages 238–251, 2004.

Guoliang Xu, Qing Pan, and Chandrajit L. Bajaj. Discrete surface modelling using partial differential equations. *Computer Aided Geometric Design*, 23(2):125–145, 2006.

Zhili Chen, Renguo Feng, and Huamin Wang. Modeling friction and air effects between cloth and deformable bodies. *ACM Trans. Graph.*, 32(4), 2013.

James F. O'Brien, Adam W. Bargteil, and Jessica K. Hodgins. Graphical modeling and animation of ductile fracture. *ACM Trans. Graph.*, 21(3):291–294, 2002.

Keenan Crane, Clarisse Weischedel, and Max Wardetzky. Geodesics in heat: A new approach to computing distance based on heat flow. *ACM Transactions on Graphics (TOG)*, 32(5):1–11, 2013.

Michal Edelstein, Nestor Guillen, Justin Solomon, and Mirela Ben-Chen. A convex optimization framework for regularized geodesic distances. *Proc. SIGGRAPH*, 2023.

Alexander Belyaev and Pierre-Alain Fayolle. An admm-based scheme for distance function approximation. *Numer. Algorithms*, 84(3):983–996, 2020.

Alexander G. Belyaev and Pierre-Alain Fayolle. On variational and pde-based distance function approximations. *Computer Graphics Forum*, 34(8):104–118, 2015.

M. P. do Carmo. *Differential geometry of curves and surfaces.* Prentice Hall, 1976.

W. H. Fleming and P. E. Souganidis. On the existence of value functions of two-player, zero-sum stochastic differential games. *Indiana University Mathematics Journal*, 38 (2):293–314, 1989.

Andrei Medved, Riley Davis, and Paula A. Vasquez. Understanding fluid dynamics from langevin and fokker–planck equations. *Fluids*, (1), 2020.

Gilbert Strang. On the construction and comparison of different splitting schemes. *SIAM J. Numer. Anal.*, 5(3):506–517, 1964.

Gury Ivanovich Marchuk. Splitting methods. page 264, 1988.

H. F. Trotter. On the product of semi-groups of operators. *Proceedings of the American Mathematical Society*, 10(4):545–551, 1959.

Tosio Kato. On the trotter-lie product formula. *Proceedings of the Japan Academy*, 50(9):694–698, 1974.

Albert Fathi. Viscosity solutions of the hamilton-jacobi equation on a non-compact manifold. *preprint*, 2020.

Mario Botsch, Leif Kobbelt, Mark Pauly, Pierre Alliez, and Bruno Levy. *Partial Differential Equations*, volume 19. American Mathematical Soc., 2010.

Michael Grant and Stephen Boyd. CVX: Matlab software for disciplined convex programming, version 2.1. `http://cvxr.com/cvx`, March 2014.

Michael Grant and Stephen Boyd. Graph implementations for nonsmooth convex programs. In V. Blondel, S. Boyd, and H. Kimura, editors, *Recent Advances in Learning and Control*, Lecture Notes in Control and Information Sciences, pages 95–110. Springer-Verlag Limited, 2008. `http://stanford.edu/~boyd/graph_dcp.html`.

K. C. Toh, M. J. Todd, and R. H. Tütüncü. Sdpt3 — a matlab software package for semidefinite programming, version 1.3. *Optimization Methods and Software*, 11(1-4): 545–581, 1999.

Reha H. Tütüncü, Kim-Chuan Toh, and Michael J. Todd. Solving semidefinite-quadratic-linear programs using sdpt3. *Mathematical Programming*, 95:189–217, 2003.