Fellowship Statement

Leticia Mattos Da Silva
leticiam@mit.edu

Figure 1: The time evolution of the Fokker-Planck equation under a flow on a triangle mesh using our framework in MATLAB.

The analysis of partial differential equations (PDE) is a ubiquitous technique in computer graphics, geometry processing, and adjacent fields. In particular, *second-order parabolic* PDE describe a wide variety of interesting physical 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. As another example, 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 [4–6, 9]. Hence, methods for accurately and efficiently solving this class of PDE over geometric domains are central 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 nonlinear phenomena. Some of the difficulties that arise include the strong stiffness of certain equations in the family of *second-order parabolic* PDE. Even current solvers in MATLAB, such as ode23s, which is designed to handle problems with stiff terms, struggle to provide accurate solutions to these equations efficiently. My current research efforts are focused on developing a framework to solve second-order parabolic PDE on curved surfaces. In particular, in my most recent work, I address this challenge by leveraging a splitting integrator combined with a convex optimization step, which can readily be solved using optimization tools such as the ones provided in MATLAB.

In what follows, I describe the broader impact of my research work, noting how MATLAB has facilitated my computational work, and how my research has extended the software's capability to work with this family of PDE with higher accuracy.

Second-order Parabolic PDE. Second-order parabolic PDE may be thought of as generalizations of the heat equation; these equations appear in applications where the object of interest is the density of a quantity on a curved domain, evolving forward in time. While there is a vast array of tools available to solve the heat equation on discrete curved domains, less is known about solving more general second-order parabolic PDE, especially in the presence of terms that contribute nonlinearity or chaotic behavior.

Finite difference schemes, such as forward Euler, explicit Runge-Kutta formulas, backward Euler, and the Crank-Nicolson method, are commonly used numerical integration techniques. Many second-order parabolic PDE, however, cannot be efficiently solved on triangle meshes with these methods. The strong stiffness of certain equations in this family is not suitable for explicit methods such as Runge-Kutta [8].

It is for this reason that MATLAB's ode23s, which relies precisely on an implementation of Runge-Kutta [7], can only provide low accuracy solutions to equations with stiff terms [2]. In experiments, I have also found ode23s and other built-in solvers in MATLAB to be unable to find an appropriate time step to solve certain equations in this family.

New framework. To address the aforementioned limitations, I propose a framework leveraging a splitting integration strategy and an appropriate spatial discretization to solve parabolic PDE over discrete curved surfaces. The splitting allows me 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. To derive the convex relaxation step, I take inspiration from [1] but deal with a larger class of equations. I apply this new framework to three difficult second-order parabolic PDE: a nonlinear Hamilton-Jacobi equation, the nonlinear G-equation, and the Fokker-Planck equation, all of which are solved efficiently on a variety of curved domains and time steps using my framework. Empirically, my method improves performance compared to the state-of-the-art in geometry processing.

MATLAB-based software has been pivotal in the implementation of my framework to solve second-order parabolic PDE. In particular, I use CVX for disciplined convex optimization in MATLAB, allowing these second-order PDE to be solved in a few lines of code. The fact that important geometry processing toolboxes, including gptooolbox [3], are based in MATLAB also made it the most efficient software to implement my framework. It allowed me to focus on the research aspect of my project by obviating the need for me to build basic geometry processing functions from scratch. Although the recommended solvers in MATLAB were on their own inadequate to complete the task at hand, I nonetheless found it convenient to derive my new framework around them. MATLAB has also allowed for easy integration of my framework with visualization tools for sharing results [redacted]

Final Remarks. My line of research lays the foundation for graphics research to implement more advanced techniques using *second-order parabolic* PDE. In addition to its broader impact, my framework has advanced the use of MATLAB to solve these PDE with challenging nonlinear or stiff terms. An immediate avenue for future work is to derive an Alternating Direction Method of Multipliers (ADMM) optimization algorithm to solve the same convex relaxation problem with improved timing. Over the next few years, I am excited to work on many graphics and geometry processing applications that follow from my framework. These include but are not limited to: position-based flow using the G-equation for realistic models of thin flame propagation and fire, mesh editing via potential field manipulation using the Fokker-Planck equation, new approaches to stochastic heat kernel estimation on curved meshes using the Fokker-Planck equation, advances on texture synthesis based on nonlinear interactions, and medial axis detection on meshes using more general Hamilton-Jacobi equations. The aforementioned applications are far-reaching and impact a diverse set of areas, ranging from physics-based simulation to CAD design.

REFERENCES

[1] Michal Edelstein, Nestor Guillen, Justin Solomon, and Mirela Ben-Chen. 2023. A Convex Optimization Framework for Regularized Geodesic Distances. *Proc. SIGGRAPH* (2023).
[2] The MathWorks, Inc. 2023. Choose an ODE Solver. https://www.mathworks.com/help/matlab/math/choose-an-ode-solver.html.
[3] Alec Jacobson et al. 2021. gptoolbox: Geometry Processing Toolbox. http://github.com/alecjacobson/gptoolbox.
[4] T. A. Leatham, D. M. Paganin, and K. S. Morgan. 2022. X-ray dark-field and phase retrieval without optics, via the Fokker-Planck equation.
[5] Petr Mokrov, Alexander Korotin, Lingxiao Li, Aude Genevay, Justin Solomon, and Evgeny Burnaev. 2021. Large-Scale Wasserstein Gradient Flows.
[6] Stanley Osher, Howard Heaton, and Samy Wu Fung. 2023. A Hamilton–Jacobi-based proximal operator. *Proceedings of the National Academy of Sciences* 120, 14 (2023).
[7] Lawrence F. Shampine and Mark W. Reichelt. 1997. The MATLAB ODE Suite. *SIAM Journal on Scientific Computing* 18, 1 (1997), 1–22.
[8] B.P. Sommeijer, L.F. Shampine, and J.G. Verwer. 1998. RKC: An explicit solver for parabolic PDEs. *J. Comput. Appl. Math.* 88, 2 (1998), 315–326.
[9] Hongfang Wang and Edwin R. Hancock. 2008. Probabilistic relaxation labelling using the Fokker–Planck equation. *Pattern Recognition* 41, 11 (2008), 3393–3411.