Modeling Advection on Directed Graphs using Matérn Gaussian Processes for Traffic Flow

Danielle C. Maddix
Amazon Research
2795 Augustine Dr.
Santa Clara, CA 95054
dmmaddix@amazon.com

Nadim Saad
Stanford University
450 Serra Mall
Stanford, CA 94305
nsaad31@stanford.edu

Yuyang Wang
Amazon Research
2795 Augustine Dr.
Santa Clara, CA 95054
yuyawang@amazon.com

Abstract

The transport of traffic flow can be modeled by the advection equation. Finite difference and finite volumes methods have been used to numerically solve this hyperbolic equation on a mesh. Advection has also been modeled discretely on directed graphs using the graph advection operator [4, 18]. In this paper, we first show that we can reformulate this graph advection operator as a finite difference scheme. We then propose the Directed Graph Advection Matérn Gaussian Process (DGAMGP) model that incorporates the dynamics of this graph advection operator into the kernel of a trainable Matérn Gaussian Process to effectively model traffic flow and its uncertainty as an advective process on a directed graph.

1 Introduction

The continuous linear advection equation models the flow of a scalar concentration along a vector field. The solutions to this hyperbolic partial differential equation may develop discontinuities or shocks over time depending on the initial condition. These shocks can model the formation of traffic jams, and their propagation along a road [20]. Figure 1 illustrates an example, where initially the first half of the road is 70% occupied with cars, and the second half of the road is empty. The traffic propagates to the right until the whole road is 70% occupied. Classical methods, such as finite differences and finite volumes, have been used to predict the flow of traffic along a road [15, 20]. These classical numerical methods do not incorporate any randomness into the model, and can be limited in incorporating the uncertainty among different driver's behaviors [6].
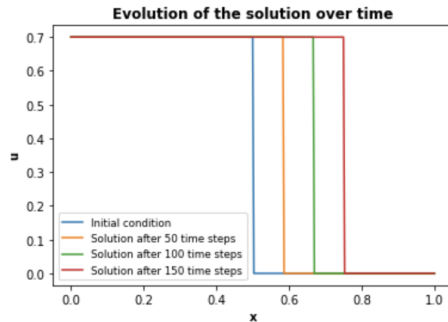

Figure 1: Propagation of cars on a road using an advection process.

Gaussian processes (GPs) [19] can learn unknown functions that allow use of prior information about their properties and for uncertainty modeling. Küper and Waldherr [10] propose the Gaussian Process Kalman Filter (GPKF) method to simulate spatiotemporal models, and test on the advection equation. Raissi et al. [17] train GPs on data to learn the underlying physics of non-linear advection-diffusion equations. Additional physics-based machine learning models [2] use the Matérn covariance function given below:

$$u \sim N\big(0, \big(\frac{2\nu}{\kappa^2} + \Delta\big)^{-\nu}\big), \qquad (1)$$

where u denotes an unknown function, $\nu < \infty$, $\kappa < \infty$ and Δ denotes the laplacian [1]. The Matérn

kernel captures physical processes due to its finite differentiability, and is also commonly used to define distances between two points that are d units distant from each other [2]. Gulian et al. [8] propose training joint Matérn GPs to model space-fractional differential equations, in which the advection-diffusion equation is a special case.

Recent works including [22] have studied solving partial differential equation (PDEs) on graphs. Chapman and Mesbahi [4], Rak [18] propose discrete advection and consensus operators to model advection and diffusion flows, respectively on directed graphs. Hošek and Volek [9] study the advection-diffusion equation on graphs using this discrete advection operator, and show that finite volume numerical discretizations can be reformulated as equations on graphs resulting in a corresponding maximum principle for this operator. Additional works have also looked at combining scientific computing and machine learning on graphs for spatiotemporal traffic modeling [12]. Chamberlain et al. [3] propose the Graph Neural Diffusion (GRAND) method, which combines traditional ODE solvers with graph neural networks (GNNs) to model diffusion on a undirected graph. Borovitskiy et al. [2] propose to replace the continuous laplacian Δ in (1) with the discrete graph laplacian operator L to model diffusion on undirected graphs, which can be limited for traffic modeling.

The goal of this paper is two-fold: to develop a model that effectively models traffic flow as an advective process on a directed graph and its uncertainty. We propose a novel method, Directed Graph Advection Matérn Gaussian Process (DGAMGP) that uses a symmetric positive definite variant of the graph advection operator L_{adv} as a covariance matrix in the Matérn Gaussian Process. We use the square of the singular values of L_{adv} to model the advection dynamics, and train a Matérn Gaussian Process to model the uncertainty. We also show the connection between consistent finite difference stencils for solving the linear advection equation and the graph advection operator. Our novel linkage helps improve the understanding and interpretability of this graph advection operator.

2 Understanding the directed graph advection operator

We aim to model the continuous advection equation for unknown scalar u under vector field v:

$$\frac{\partial u}{\partial t} = -\nabla \cdot (vu),$$

stochastically on a directed graph. We define a directed, weighted graph $\mathcal{G} = (V, E, W)$ with $|V| = n$ nodes and $|E| = |W| = m$ edges, where V denotes the vertex, E the edge, and W the edge weight sets, respectively. We discretize the flow vu along edge $(i, j) \in E$ with weight $w_{ji} \in W$ as $w_{ji}u_i(t)$, where $u_i(t)$ denotes the concentration u at node i and time t.

The graph advection operator L_{adv} is defined so that the flow into a node equals the flow out of it [4]:

$$\frac{du_i(t)}{dt} = \sum_{j:(j,i)\in E} w_{ij}u_j(t) - \sum_{j:(i,j)\in E} w_{ji}u_i(t) = -[L_{adv}u(t)]_i, \tag{2}$$

where $L_{adv} = D_{out} - A_{in}$ for diagonal out-degree matrix D_{out} and in-degree adjacency matrix A_{in}. For general directed graphs, L_{adv} belongs to the square, non-symmetric with non-negative real part eigenvalues [18] class of matrices in [14]. By design, L_{adv} is conservative, unlike the related diffusion or consensus operator $L_{cons} = D_{in} - A_{in}$, where D_{in} denotes the diagonal in-degree matrix [4, 18]. A main motivating reason for using L_{adv} to model traffic flow is that it results in a conservative scheme.

Reformulation of L_{adv} as finite difference on balanced graphs. We notice that L_{adv} at node i is a weighted linear combination of the other nodes adjacent to it, which resembles finite difference stencils of the unknown and its neighbors. We make this connection precise, and then construct example graphs where L_{adv} corresponds to common finite difference schemes for linear advection.

Theorem 2.1. *L_{adv} corresponds to a semi-discrete finite difference advection scheme, where the sum of the coefficients is zero if and only if the graph $\mathcal{G}$ is balanced, i.e. $L_{adv} = L_{cons}$.*

Proof. A finite difference approximation to the gradient can be written as the following weighted linear combination of its neighbors u_j for arbitrary coefficients $c_{ij} \in \mathbb{R}$:

$$-(u_x)_i \approx \sum_{j \neq i} c_{ij}u_j + c_{ii}u_i. \tag{3}$$

A consistent finite difference scheme is at least zero-th order accurate [11]. Since the derivative of a constant is 0, the coefficients must sum to 0, i.e $c_{ii} = -\sum_{j \neq i} c_{ij}$. Combining (2) with (3) gives:

$$(D_{out})_{ii} = \sum_{j:(i,j) \in E} w_{ji} = -c_{ii} = \sum_{j \neq i} c_{ij} = \sum_{j:(j,i) \in E} w_{ij} = (D_{in})_{ii}.$$

The graph $\mathcal{G}$ is balanced by definition, and it follows that $L_{adv} = L_{cons}$. The other direction follows similarly. □

Applying L_{adv} on the directed line graph in Figure 2(a) results in the first order upwind scheme with spatial step size Δx for $v > 0$ in (5) (See Appendix A and Figure 6 for the convergence study). Similarly, Figure 2(b) illustrates the directed graph in which L_{adv} gives the second order central difference scheme, where $(u_x)_i \approx (u_{i+1} - u_{i-1})/(2\Delta x)$ (See Appendix B for additional examples).
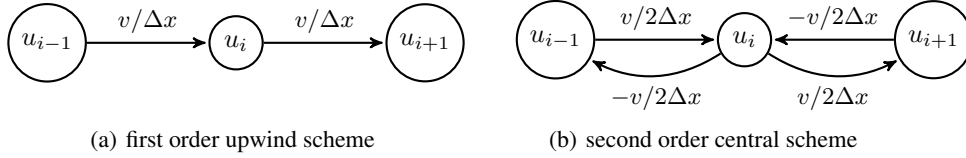

(a) first order upwind scheme (b) second order central scheme

Figure 2: Balanced graphs on which L_{adv} corresponds to finite difference stencils of linear advection.

3 Directed Graph Advection Matérn Gaussian Process (DGAMGP)

We propose the novel Directed Graph Advection Matérn Gaussian Process (DGAMGP) model, which uses the dynamics of L_{adv} to model advection stochastically on a directed graph through a discrete approximation to the continuous Laplacian Δ of the Matérn Gaussian Process in (1). The covariance matrix or kernel $\mathcal{K}$ of a Gaussian process needs to be symmetric and positive semi-definite. This leads to some challenges with the L_{adv} operator as it is not guaranteed in general to be symmetric or positive semi-definite (See Section 2). Note that using the graph Laplacian L in the covariance matrix in the undirected graph case is more straightforward since L is symmetric positive semi-definite.

In our directed graph case, we propose using $L_{adv}^T L_{adv}$ as the covariance matrix since it is symmetric positive definite, and hence orthogonally diagonalizable. Analogous to [2], we define a function ϕ of a diagonalizable matrix through Taylor series expansion. Then we can define its eigendecomposition as $L_{adv}^T L_{adv} = X_{adv} \Lambda_{adv} X_{adv}^T$, so that $\phi(L_{adv}^T L_{adv}) = X_{adv} \phi(\Lambda_{adv}) X_{adv}^T$, where $\phi(\Lambda_{adv})$ is computed by applying ϕ to the diagonal elements of Λ_{adv}.

We compute the eigendecomposition of $L_{adv}^T L_{adv} = V_{adv} \Sigma_{adv}^2 V_{adv}^T$, using the singular value decomposition (SVD) of $L_{adv} = U_{adv} \Sigma_{adv} V_{adv}^T$, where the eigenvalues and eigenvectors are the singular values squared and right singular vectors of L_{adv}, respectively. Hence, we model the advection dynamics using the square of the singular values of L_{adv}. Our approach can also be viewed as adding the square of the singular values of L_{adv} to the diagonal for regularization. Computing the thin-SVD is more computationally efficient and numerically stable, since we avoid explicitly forming the matrix-matrix product $L_{adv}^T L_{adv}$, which has double the condition number of L_{adv}, and the numerical issues with then computing its eigendecomposition.

We chose ϕ to be the Matérn covariance function in (1), and our DGAMGP model is given by:

$$u \sim N\big(0, \big(V_{adv}(\frac{2\nu}{\kappa^2} I + \Sigma_{adv}^2)^{-\nu} V_{adv}^T\big)\big). \tag{4}$$

This advective Gaussian Process is then trained on data by minimizing the negative log-likelihood of the Gaussian Process to learn the kernel hyperparameters ν and κ, and predict u [7]. For inference, we draw samples from the GP predictive posterior distribution with the learned hyperparameters [19]. See Algorithm 1 for details.

Choice of $L_{adv}^T L_{adv}$. There are alternate approaches to symmetrize L_{adv}. The first simple approach explored is to utilize $L_{sym} = (L_{adv}^T + L_{adv})/2$. This operator is not positive semi definite except

in the balanced graph case. The second approach is to use the symmetrizer method in [21], which generates a symmetric matrix L'_{sym} with the same eigenvalues as L_{adv} but is not always positive semi definite.

Algorithm 1 The Directed Graph Advection Matérn Gaussian Process (DGAMGP)

    Given a directed graph $\mathcal{G} = (V, E, W)$ and training data $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$.
1. Compute $L_{adv}(\mathcal{G}) = D_{out} - A_{in}$.
2. Compute the SVD of $L_{adv} = U_{adv}\Sigma_{adv}V_{adv}^T$.
3. Generate a DGAMGP model in (4).
4. Minimize the GP negative log marginal likelihood using $\mathcal{D}$ to learn ν, κ and σ [7].
5. Given test data $\{x_i^*\}$, draw samples from the GP predictive posterior distribution [19].

---

4 Numerical Results

In this section, we utilize our DGAMGP model for traffic modeling on synthetic and real-world directed traffic graphs. The data $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$ denotes the traffic flow speed in miles per hour y_i at location x_i. We test our model's predictive ability to predict the velocities of cars on a road at different positions. We use hold-out cross validation to split the data points generated into training (70% of the data) and testing data (30% of the data). We extend the code in [2] to compute the singular value decomposition of L_{adv} to train our DGAMGP model on a directed graph. The code is available at `https://github.com/advectionmatern/Modeling-Advection-on-Directed-Graphs-using-Mat-e-rn-Gaussian-Processes`, and the experiments are run on Amazon Sagemaker [13].

Regression results on synthetic graphs. We generate synthetic data that models traffic along a road, which has a relatively high density of cars in the first half and a low density of cars in the second half. We train and test our model on the upwind scheme in Figure 2(a), central scheme in Figure 2(b), an intersecting lane graph, where two lanes merge into one lane in Figure 3(a) and a loop graph representing the upwind scheme with periodic boundary conditions in Figure 3(b). Table 1 compares the results to the consensus baseline model of using the singular value decomposition of L_{cons} in Eqn. (4).

Model	Graph type	$n = 280$	$n = 325$	$n = 400$	ν	κ	σ
Advection	Upwind	0.52	0.45	**0.0005**	0.65	8.09	7.75
Consensus		**0.51**	**0.44**	**0.0005**	0.65	8.29	7.77
Advection	Central	1.31	0.85	8.41e-05	0.67	9.00	8.03
Consensus		**0.97**	**0.8**	**8.02e-05**	0.67	9.45	8.11
Advection	Intersection	0.96	**0.45**	**0.0005**	0.65	8.19	7.75
Consensus		**0.52**	0.46	**0.0005**	0.64	8.28	7.77
Advection	Loop	**0.47**	**0.41**	0.00045	0.65	8.49	7.76
Consensus		**0.47**	**0.41**	0.00045	0.65	8.49	7.76

Table 1: Comparison of l_2 test error on synthetic directed graphs with n nodes and the learned hyperparameters.
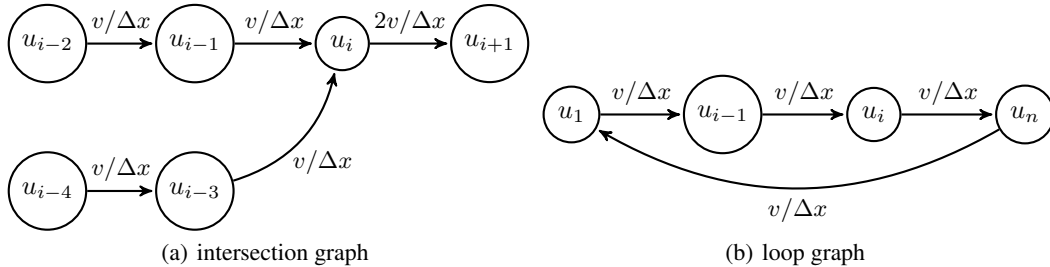

(a) intersection graph   (b) loop graph

Figure 3: Graphs representing two lanes merging into one (*left*) and a loop (*right*).

Regression results on a real-world traffic graph. We test on the real-world traffic data from the California Performance Measurement System [5] with the road network graph from the San Jose highways from Open Street Map [16] at a fixed time. Since our method supports directed graphs, we do not need to convert the raw directed traffic data to an undirected graph as in [2]. We use the same experimental setup from [2] to generate the train and test data. Figure 4 shows the resulting predictive mean and standard deviation of the speed on the San Jose highways using the visualization tools from [2]. We notice that the predictive standard deviation along the nodes is relatively small, and is larger on the points that are farther from the sensors.
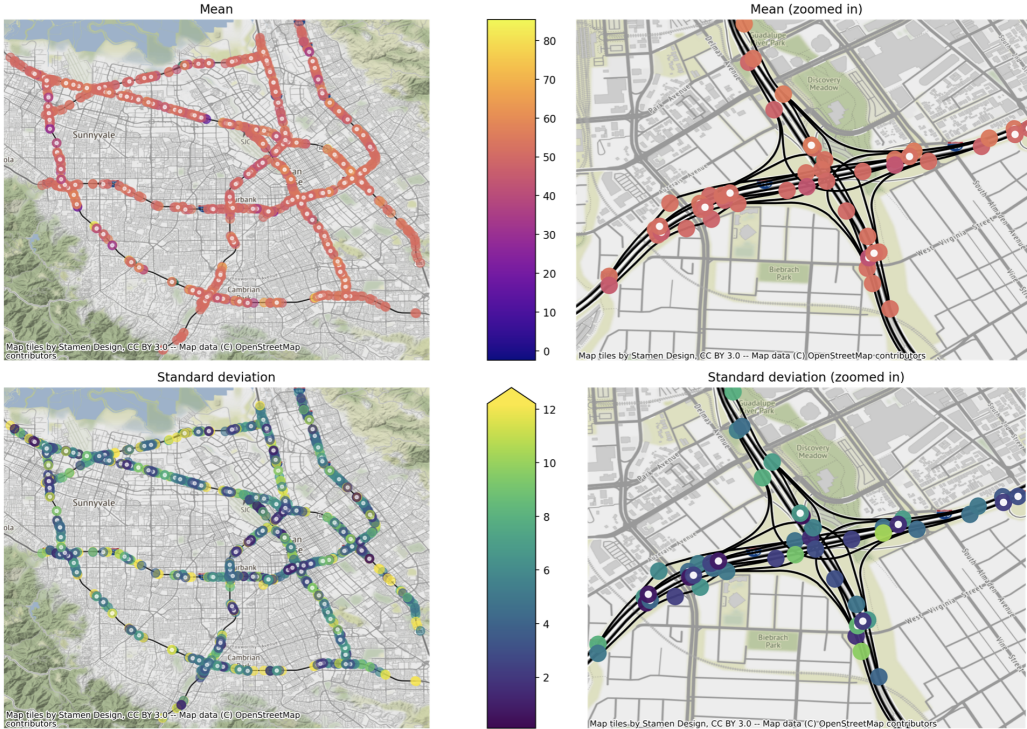

Figure 4: Traffic speed interpolation over a graph of San Jose highways using our DGAMGP method with $\nu = 0.35, \kappa = 1002.8, \sigma = 1.14$ and plotting tools from [2].

5 Conclusions

In this paper, we propose a novel method DGAMGP to model an advective process on a directed graph and its uncertainties. We show connections between finite differences schemes used to solve the linear advection equation and the graph advection operator L_{adv} employed in our model. We explore a regression problem on various graphs, and show that our proposed DGAMGP model performs similarly to other state-of-the-art models. Future work includes adding a time-varying component to our model, comparing our method to classical numerical methods for solving PDEs, and incorporating the behavior of the non-linear advection equation for traffic modeling.

References

[1] Bakka, H., Krainski, E., Bolin, D., Rue, H., and Lindgren, F. (2020). The diffusion-based extension of the matérn field to space-time. *arXiv:2006.04917*.

[2] Borovitskiy, V., Azangulov, I., Terenin, A., Mostowsky, P., Deisenroth, M., and Durrande, N. (2021). Matérn gaussian processes on graphs. In Banerjee, A. and Fukumizu, K., editors, *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pages 2593–2601. PMLR.

[3] Chamberlain, B., Rowbottom, J., Gorinova, M. I., Bronstein, M., Webb, S., and Rossi, E. (2021). Grand: Graph neural diffusion. In Meila, M. and Zhang, T., editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 1407–1418. PMLR.

[4] Chapman, A. and Mesbahi, M. (2011). Advection on graphs. *IEEE Conference on Decision and Control and European Control Confereence (CDC-ECC)*, 50:1461–1466.

[5] Chen, C., Petty, K., Skabardonis, A., Varaiya, P., and Jia, Z. (2001). Freeway performance measurement system: mining loop detector data. *Transportation Research Record*, 1748(1):96–102.

[6] Chen, Y., Sohani, N., and Peng, H. (2018). Modelling of uncertain reactive human driving behavior: a classification approach. In *2018 IEEE Conference on Decision and Control (CDC)*, pages 3615–3621.

[7] Gardner, J., Pleiss, G., Bindel, D., Weinberger, K., and Wilson, A. (2018). GPytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration. *32nd Conference on Neural Information Processing Systems (NIPS 2018) arXiv:1809.11165v2*.

[8] Gulian, M., Raissi, M., Perdikaris, P., and Karniadakis, G. (2019). Machine learning of space-fractional differential equations, SIAM Journal on Scientific Computing, Vol. 41, No. 4, Society for Industrial and Applied Mathematics. pages A2485–A2509.

[9] Hošek, R. and Volek, J. (2019). Discrete advection–diffusion equations on graphs: Maximum principle and finite volumes. *Applied Mathematics and Computation*, 361(C):630–644.

[10] Küper, A. and Waldherr, S. (2020). Numerical gaussian process kalman filtering. *21st IFAC World Congress*.

[11] LeVeque, R. J. (2007). *Finite Difference Methods for Ordinary and Partial Differential Equations: Steady-State and Time-Dependent Problems*. SIAM.

[12] Li, Y., Yu, R., Shahabi, C., and Liu, Y. (2018). Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. *International Conference on Learning Representations (ICLR)*.

[13] Liberty, E., Karnin, Z., Xiang, B., Rouesnel, L., Coskun, B., Nallapati, R., Delgado, J., Sadoughi, A., Astashonok, Y., Das, P., Balioglu, C., Chakravarty, S., Jha, M., Gautier, P., Arpin, D., Januschowski, T., Flunkert, V., Wang, Y., Gasthaus, J., Stella, L., Rangapuram, S., Salinas, D., Schelter, S., and Smola, A. (2020). Elastic machine learning algorithms in amazon sagemaker. In *2020 ACM SIGMOD International Conference on Management of Data, SIGMOD '20, New York, NY, USA. Association for Computing Machinery.*, pages 731–737.

[14] Liesen, J. and Parlett, B. N. (2008). On nonsymmetric saddle point matrices that allow conjugate gradient iterations. *Numer. Math.*, 108:605–624.

[15] Lighthill, M. and Whitham, G. (1955). On kinematic waves ii. a theory of traffic flow on long crowded roads. *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences*, 229:317 – 345.

[16] OpenStreetMap (2017). `https://www.openstreetmap.org`.

[17] Raissi, M., Perdikaris, P., and Karniadakis, G. (2019). Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707.

[18] Rak, A. (2017). Advection on graphs. `http://nrs.harvard.edu/urn-3:HUL.InstRepos:38779537`.

[19] Rasmussen, C. and Williams, C. (2006). *Gaussian Processes for Machine Learning*. MIT Press.

[20] Richards, P. (1956). Shock waves on the highway. *Operation Res.*, pages 42 – 51.

[21] Sen, S. and Venkaiah, V. C. (1988). On symmetrizing a matrix. *Indian J. pure appl. Math.*, 19(6):554–561.

[22] Solomon, J. (2015). PDE approaches to graph analysis. *ArXiv*, abs/1505.00185.

Checklist

1. For all authors...

   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]

   (b) Did you describe the limitations of your work? [Yes] See Section 3

   (c) Did you discuss any potential negative societal impacts of your work? [N/A]

   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]

2. If you are including theoretical results...

   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See Section 2

   (b) Did you include complete proofs of all theoretical results? [Yes] See Section 2

3. If you ran experiments...

   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See Appendix C

   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Section 4

   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See Section 4

   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Appendix C

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...

   (a) If your work uses existing assets, did you cite the creators? [Yes] See Section 4

   (b) Did you mention the license of the assets? [N/A]

   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] See References

   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A] Data was available to the public

   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...

   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A   Upwinding discretizations of linear advection

We discretize the 1D linear advection equation with velocity v:

$$u_t + vu_x = 0,$$

using the standard first order upwinding scheme on a simple uniform Cartesian mesh with spatial step size Δx. Then the classical finite difference first-order upwind scheme depends on the sign of v. For flow moving from left to right, $v > 0$, and we have the following semi-discrete discretization [11]:

$$\frac{du_i}{dt} + v\frac{u_i - u_{i-1}}{\Delta x} = 0, \quad \text{if} \quad v > 0,$$
$$\frac{du_i}{dt} + v\frac{u_{i+1} - u_i}{\Delta x} = 0, \quad \text{if} \quad v < 0. \tag{5}$$

Upwinding schemes are useful in the advection case since information is moving from left to right. The Courant-Friedrichs-Lewy (CFL) condition for stability of the first order upwinding scheme with Forward Euler time-stepping discretization with time step Δt is given by:

$$\left|\frac{v\Delta t}{\Delta x}\right| \leq 1 \iff \Delta t \leq \left|\frac{v}{\Delta x}\right|.$$

A less diffusive second order upwind scheme is also known as linear upwind differencing (LUD), and is given by:

$$\frac{du_i}{dt} = v\frac{-u_{i-2} + 4u_{i-1} - 3u_i}{2\Delta x}. \tag{6}$$

We can show that the scheme is second-order accurate using Taylor expansions. It is designed to be less diffusive because the u_{xx} term from the first-order upwinding scheme cancels. We have

$$\frac{u_{i-2} - 4u_{i-1} + 3u_i}{2\Delta x} = \frac{1}{2\Delta x}\left[\left(u - 2\Delta x u_x + \frac{4\Delta x^2}{2}u_{xx} - \frac{8\Delta x^3}{6}u_{xxx} + \mathcal{O}(\Delta x^4)\right)\right.$$

$$+ \left(-4(u - \Delta x u_x + \frac{\Delta x^2}{2}u_{xx} - \frac{\Delta x^3}{6}u_{xxx} + \mathcal{O}(\Delta x^4))\right) + 3u\right]$$

$$= u_x - \frac{\Delta x^2}{3}u_{xxx} + \mathcal{O}(\Delta x^4).$$

Hence, the scheme is second order accurate with a dispersive u_{xxx} leading error term.

B   Examples of L_{adv} on balanced graphs resulting in finite difference discretizations of linear advection
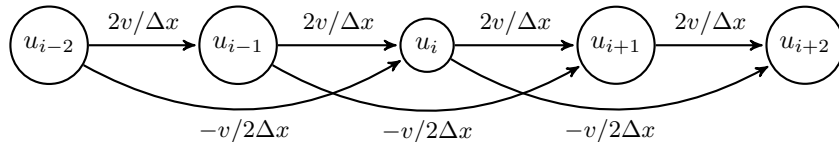
In addition to the finite difference schemes provided in Section 2, we also provide an example of a non-uniform mesh discretization:

$$\frac{du_i}{dx} \approx \frac{\frac{4}{3}u_{i+1/2} - u_i - \frac{1}{3}u_{i-1}}{\Delta x},$$

which results in the following graph, where the in-going and out-going edges from u_i:
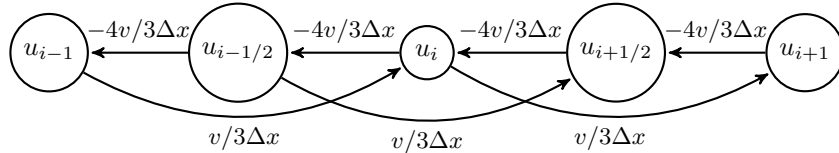
We can obtain the less diffusive second order upwind scheme (LUD) in (6) using the following graph:

C Additional Experiments

C.1 Gaussian Process prior results with DGAMGP

A main property of the Matérn Gaussian Process kernel is that it varies along Riemannian manifolds. The variance of the kernel is a function of degree, and depends on a complex manner on the graph. We show the results generated with a star graph directed towards the center node and a directed complete graph. Figure 5(a) shows that as expected for the complete graph, the nodes have the same variability, since for a random walk starting from any node, there is equal probability to get to another node. For the star graph in Figure 5(b), we observe that the center node has a variability of approximately 0 as starting from any node on the graph, the random walk always ends at the center.
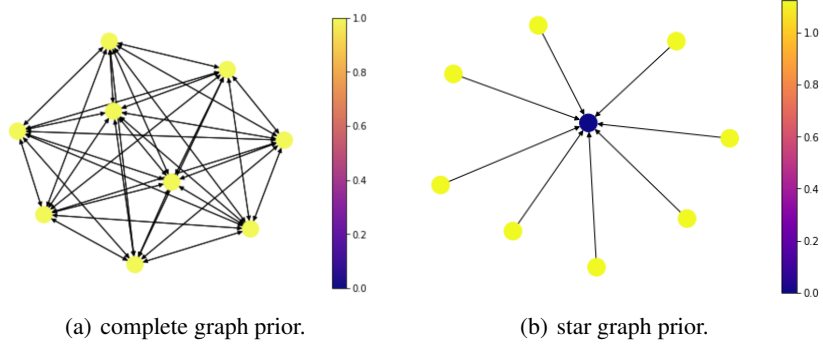

(a) complete graph prior.   (b) star graph prior.

Figure 5: Prior results using DGAMGP obtained using various graphs, and plotting tools from [2].

C.2 Convergence Studies

We conduct a convergence study of applying L_{adv} on the upwind graph in Figure 2(a), and show that it has first order convergence matching the performance of the equivalent first order upwind scheme. We use the same initial condition as in Figure 1. We then solve the resulting system of ODEs using the RK5 ODE solver. Figure 6(a) shows the solution at different time steps, and we see how the solution is propagating to the right. Figure 6(b) shows a loglog plot, where the error is decreasing linearly with a slope of 1 as the number of nodes n is increasing, as expected.

(a) Solution of the linear advection equation using (2)   (b) Convergence study in a log-log plot
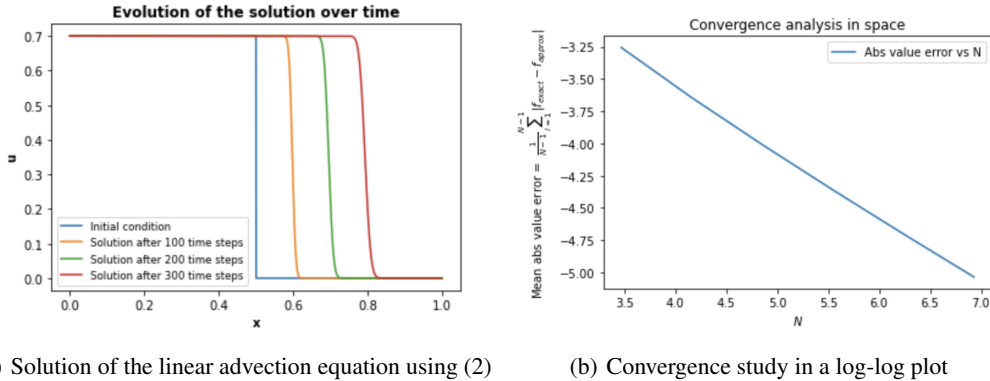
Figure 6: Upwinding solution with RK5 to the linear advection equation over time and corresponding convergence study.