Density Tracking By Quadrature for High Dimensional Stochastic Differential Equations

Ryleigh Moore and Akil Narayan, PhD

The University of Utah, Department of Mathematics

RMoore@math.utah.edu

The Problem

We wish to extend the results of Bhat and Madushani [1] to solve stochastic differential equations (SDEs) in high dimensions.
We are interested in solving SDEs of the form

$$dX_t = f(X_t)dt + g(X_t)dW_t, \quad X_0 = C \tag{1}$$

f : Drift $\quad g$: Diffusion

W_t : Brownian Motion $\quad X_t$: Itô diffusion

Furthermore, this method can be used to solve Fokker-Planck PDEs and is up to 100 times faster than other methods with the same accuracy [1]. The speed of DTQ makes this method attractive for solving inference problems.
This poster will often use 2-dimensional examples to illustrate methods that can be extended to higher dimensional cases.

Density Tracking by Quadrature

Density Tracking by Quadrature (DTQ) [1] is a convergent method that solves for the probability density function $p(x,t)$ of X_t.

1. **Discretize the SDE (1) in time** (e.g. Euler–Maruyama) and let $\tilde{p}$ denote its density

$$x_{n+1} = x_n + f(x_n)h + g(x_n)\sqrt{h}Z_{n+1} \tag{2}$$

$h > 0$: fixed time step $\quad Z_{n+1} \sim \mathcal{N}(0,1)$

From (2) we observe

$$\tilde{p}(x_{n+1} = x | x_n = y) = \mathcal{N}(y + f(y)h, hg^2(y)) =: G(x,y) \tag{3}$$

2. **Interpret (2) as a discrete-time Markov chain and consider the associated Chapman-Kolmogorov equation** for time evolution.

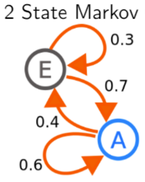
$$\tilde{p}(x, t_{n+1}) = \int_{\mathbb{R}^d} \tilde{p}(x_{n+1} = x | x_n = y)\tilde{p}(y, t_n)dy = \int_{\mathbb{R}^d} G(x,y)\tilde{p}(y,t_n)dy \tag{4}$$

3. **Discretize the Chapman-Kolmogorov equation (4) and the density $\tilde{p}$ in space** (e.g. using a mesh and numerical quadrature).

Markov Chain and Transition Matrix

A **Markov chain** is a stochastic model describing a sequence of possible events in which **the probability of the next event depends only on the current state**.

2 State Markov Chain



Source: Joxemai4

1 Step Transition Matrix

$$T = \begin{array}{c} \\ A \\ E \end{array} \begin{array}{c} \begin{array}{cc} A & E \end{array} \\ \begin{pmatrix} 0.6 & 0.7 \\ 0.4 & 0.3 \end{pmatrix} \end{array}$$

The n Step Transition Matrix is T^n and $T^{n+m} = T^n T^m$.

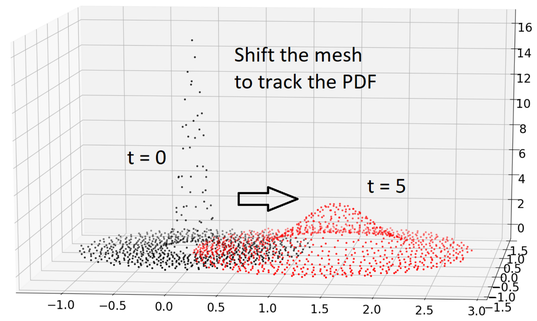
For equispaced and tensorized grids, $G(x,y)$ is a transition matrix computed from the Gaussian in equation (3) and the update for the spatially discretized PDF is given by

$$\hat{p}(x, t_{n+1}) = (\Delta x)^d G(x,y)\hat{p}(x,t_n)$$

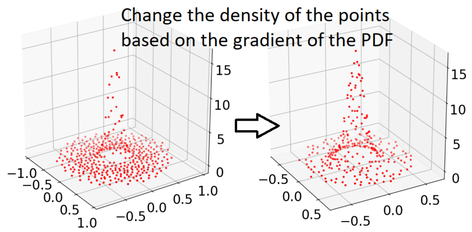
where d is the dimension of the SDE and Δx is the grid spacing.

Forming an Adaptive Mesh

We wish to utilize an adaptive mesh to track the density of the PDF solution while using as few points as possible.



If we don't adjust the mesh to track the PDF, the density may drift outside the domain of the computed solution.



We want to distribute points to create a mesh that is more dense in steep regions of the PDF and less dense in shallow regions in order to efficiently solve the SDE.

Weighted Leja Point Sequences

Tensorized grids are computationally expensive and quickly become an impractical mesh choice as the dimension increases. We instead use Leja sequences which are useful for creating adaptive, sparse meshes in high dimensions.

- Weighted Leja sequences are asymptotically distributed like weighted Gauss quadrature nodes.
- Any number of points may be added to an existing Leja sequence to create a new Leja sequence [3].

Let $w(z)$ be a weight function on a domain Ω, and let $\{z_n\}$ be a sequence of candidate samples from a weakly admissible mesh. A weighted Leja sequence [3] can be constructed via the optimization

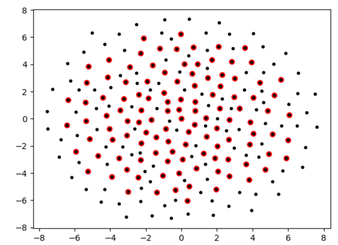
$$z_{N+1} = \underset{z \in \Omega}{\operatorname{argmax}} \sqrt{w(z)} \prod_{n=0}^{N} |z - z_n| \tag{5}$$

where z_0 is chosen as an initial point. In practice, we can solve equation (5) via an LU decomposition of a Vandermonde-like matrix [2]. The resulting row permutation information from the LU decomposition gives the Leja sequence ordering. This ordering essentially prescribes the importance of each point so that points earlier in the Leja sequence should be chosen as part of the mesh before points that appear later in the sequence.

Leja Sequences for Mesh Updates

The ordering and flexibility of Leja sequences make them useful for adding and removing points during mesh updates.

- When adding points to the existing mesh, we require points from the current mesh to be the initial points for a Leja sequence. We can then compute additional Leja points from a set of candidate samples to add new points to the mesh.
- When making the mesh less dense, we use points from the existing mesh as candidate samples for the Leja sequence. The ordering from the Leja procedure allows us to keep the most important points in the current mesh and remove points that are not as necessary.



The above figure shows a 230 point Gaussian weighed Leja sequence. The first 115 points are highlighted in red.

Leja Point Quadrature

To step the solution forward in time, we compute equation (4) for each point in the mesh via an interpolatory quadrature rule using a polynomial basis.

We use the points in the current mesh that are within a sufficient radius of the point we wish to update as the candidate samples for a Leja sequence. This allows us to select the best available subset of current mesh points to use for the quadrature rule.

Acknowledgements

I would like to thank my advisor Dr. Akil Narayan for his mentorship and support. This work was funded by ONR award N00014-19-1-2046-A00001 and the University of Utah Department of Mathematics. I would also like to thank ICERM for hosting the workshop.

References

[1] H. S. Bhat and R. W. M. A. Madushani. "Density Tracking by Quadrature for Stochastic Differential Equations". 2018.

[2] L. Bos, S. De Marchi, A. Sommariva, and M. Vianello. "Computing multivariate Fekete and Leja points by numerical linear algebra". In: *Journal on Numerical Analysis* 48.5 (2010), pp. 441–470.

[3] A. Narayan and J. D. Jakeman. "Adaptive Leja sparse grid constructions for stochastic collocation and high-dimensional approximation". In: *SIAM Journal on Scientific Computing* 36.6 (Apr. 2014), A2952–A2983.