Journal of Computational Physics 391 (2019) 142–154

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

GPU-accelerated particle methods for evaluation of sparse observations for inverse problems constrained by diffusion PDEs

Jeff Borggaard [a], Nathan Glatt-Holtz [b], Justin Krometis [c,*]

[a] Department of Mathematics, Virginia Tech, United States of America
[b] Department of Mathematics, Tulane University, United States of America
[c] Advanced Research Computing, Virginia Tech, United States of America

ARTICLE INFO

Article history:
Received 29 August 2018
Received in revised form 22 March 2019
Accepted 15 April 2019
Available online 18 April 2019

Keywords:
Inverse problems
Optimization
Scientific computing
Parallel computing
Passive scalars

ABSTRACT

We consider the inverse problem of estimating parameters of a driven diffusion (e.g., the underlying fluid flow, diffusion coefficient, or source terms) from point measurements of a passive scalar (e.g., the concentration of a pollutant). We present two particle methods that leverage the structure of the inverse problem to enable efficient computation of the forward map, one for time evolution problems and one for Dirichlet boundary-value problems. The methods scale in a natural fashion to modern computational architectures, enabling substantial speedup for applications involving sparse observations and high-dimensional unknowns. Numerical examples of applications to Bayesian inference and numerical optimization are provided.

1. Introduction

Much recent computational and theoretical work has been devoted to the inverse problem of the estimation of unknown or optimal model parameters from finite observations of the model output [1–4]. For example, recent works included reconstruction of a seismic wave speed field from waves recorded at a finite number of point receivers [5], estimation of an ice sliding coefficient field from finite velocity observations [6], and determination of the source of a chemical/biological attack from measurements of toxins [7].

Approaches to these inverse problems typically involve many evaluations of the model, also called the forward map. For example, each step of a Markov Chain Monte Carlo (MCMC) method [8] for Bayesian inference or iteration of a numerical optimization routine [9] will require evaluation of the forward map. When that map includes observations of the solution of a partial differential equation (PDE), these methods in turn require many solutions of the PDE. In many applications, these PDE solves are computationally-intensive and dominate the overall time to solution. For applications where the observations are sparse, i.e., low-dimensional, solving the PDE can be wasteful in the sense that a high-dimensional quantity is computed only to take a low-dimensional projection of it. It is therefore desirable to identify methods that allow computation of the parameter-to-observation map directly, without a full PDE solve.

In this work, we consider the problem of estimating the parameters of a driven diffusion (e.g., the background flow, diffusion coefficient, or sources) from point measurements of a passive scalar (e.g., the concentration of a contaminant). We

* Corresponding author.
E-mail address: jkrometis@vt.edu (J. Krometis).

https://doi.org/10.1016/j.jcp.2019.04.034
0021-9991/

present two numerical methods that can be used to compute observations for these inverse problems without computation of the full scalar field. These methods therefore bypass the need to approximate a high-dimensional PDE solution at each step of the inverse problem and instead replace the full PDE solve with an array of particle solutions that are much less computationally expensive. Moreover, since the particle simulations are decoupled, they can be parallelized in a straightforward manner on modern computational architectures. The result is a dramatic speedup, particularly for problems in which the dimension of the unknowns is significantly larger than the dimension of the observations.

2. Motivation

In this article, we consider two inverse problems:

Problem 2.1 *(Time-dependent Advection-Diffusion). Let $D = \mathbb{R}^n$ (or $D \subset \mathbb{R}^n$ with periodic boundary conditions). Let $\mathbf{v} : D \to \mathbb{R}^n$, $\sigma : D \to \mathbb{R}^{n\times n}$, and $\theta_0 : D \to \mathbb{R}$ be given functions. Assume that there exists a $\theta : \mathbb{R}^+ \times D \to \mathbb{R}$ satisfying*

$$\frac{d\theta}{dt}(t,\mathbf{x}) = -\mathbf{v}(\mathbf{x})\cdot\nabla\theta(t,\mathbf{x}) + \frac{1}{2}\sum_{i,j}(\sigma\sigma^T)_{i,j}(\mathbf{x})\frac{\partial^2}{\partial x_i \partial x_j}\theta(t,\mathbf{x}), \quad \theta(0,\mathbf{x}) = \theta_0(\mathbf{x}). \tag{1}$$

Goal: *Estimate unknown u (e.g., one or more of: advection field $\mathbf{v}$, diffusion coefficients σ, or initial condition θ_0) from finite, possibly noisy point observations $\theta(t_j, \mathbf{x}_j; u)$, $j = 1, \ldots, N$.*

Problem 2.2 *(Boundary Value (Dirichlet) Problem). Let D be an open, connected, bounded subset of $\mathbb{R}^n$. Let $\mathbf{v} : D \to \mathbb{R}^n$, $\sigma : D \to \mathbb{R}^{n\times n}$, $f : D \to \mathbb{R}$, and $\theta_{\partial D} : \partial D \to \mathbb{R}$ be given functions. Assume that there exists a $\theta : \overline{D} \to \mathbb{R}$ satisfying*

$$\begin{aligned} -\mathbf{v}(\mathbf{x})\cdot\nabla\theta(\mathbf{x}) + \frac{1}{2}\sum_{i,j}(\sigma\sigma^T)_{i,j}(\mathbf{x})\frac{\partial^2}{\partial x_i \partial x_j}\theta(\mathbf{x}) &= f(\mathbf{x}) \quad \mathbf{x} \in D \\ \theta(\mathbf{x}) &= \theta_{\partial D}(\mathbf{x}) \mathbf{x} \in \partial D. \end{aligned} \tag{2}$$

Goal: *Estimate unknown u (e.g., one or more of: advection field $\mathbf{v}$, diffusion coefficients σ, forcing f, or boundary condition $\theta_{\partial D}$) from finite point observations $\theta(\mathbf{x}_j; u)$, $j = 1, \ldots, N$.*

In each problem, the goal is to estimate the parameters of a driven diffusion from point measurements of the passive scalar θ. Solving these inverse problems in practice will typically involve many computations of the forward map from a given parameter u to its associated observations, which we denote by

$$\mathcal{G}(u) = \{\mathcal{G}_j(u)\}_{j=1}^N, \text{ where } \begin{cases} \mathcal{G}_j(u) = \theta(t_j, \mathbf{x}_j, u) & \text{for Problem 2.1} \\ \mathcal{G}_j(u) = \theta(\mathbf{x}_j, u) & \text{for Problem 2.2.} \end{cases} \tag{3}$$

$\mathcal{G}$ can be thought of as being composed of two operators

$$\mathcal{G}(u) = \mathcal{O}\circ\mathcal{S}(u) \tag{4}$$

where

- The solution operator $\mathcal{S} : u \mapsto \theta$ requires solving the PDE (1) or (2) for parameter u
- The observation operator $\mathcal{O}$ involves taking point observations from $\theta(u)$

As such, $\mathcal{G}$ is typically evaluated in two steps:

1. Compute $\theta(u)$ via some numerical PDE solver
2. Compute observations $\mathcal{G}(u)$ from $\theta(u)$

This natural approach has the benefit of allowing the application of third-party "black-box" PDE solvers to the inverse problem. However, computing $\mathcal{S}(u)$ involves approximating a solution that is infinite-dimensional, which can be very computationally expensive, requiring a discretization with many thousands or millions of degrees of freedom. By contrast, the evaluation of $\mathcal{O}$ then involves projecting that PDE solution into a finite-dimensional space. As a result, much of the work involved in approximating $\mathcal{S}$ is, in some sense, discarded in the application of $\mathcal{O}$. In the next section we present a numerical method for evaluation of $\mathcal{G}$ that breaks this two-step paradigm.

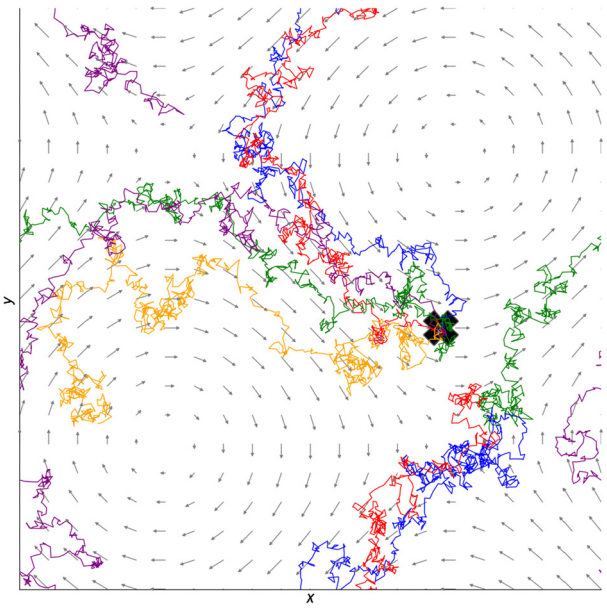

Fig. 1. Traces of 5 simulated Itô diffusions with the same final position.

3. Methods

In this section, we present a particle method that will allow point evaluation of θ directly from the unknown u without separate approximation of $\mathcal{S}$. The method will involve simulating an ensemble of particles (Itô diffusions), which is a well-known method for approximating θ; see, for example, [10], [11], or [12] for details. However, to leverage the sparse nature of the observations, for this application we will simulate the particles *backward in time* from their final condition to their initial condition. Doing so will allow us to avoid computing the entire field θ by computing it only where it is needed.

3.1. A particle method for Problem 2.1

In this section, we present a particle method for computing $\mathcal{G}(u)$ for the time-dependent problem Problem 2.1. A key ingredient is Kolmogorov's Backward Equation, which is presented in Theorem 3.1. See, for example, [13, Section 8.1] for details.

Theorem 3.1 *(Kolmogorov's Backward Equation). Let $\theta_0 \in C_b^2(D)$ and $\mathbf{v}, \sigma \in C_b^1(D)$.[1] Then there exists a bounded $\theta \in C^{1,2}(\mathbb{R}^+ \times D)$ satisfying (1), which is given by*

$$\theta(t, \mathbf{x}) = \mathbb{E}\theta_0(\mathbf{X}_t) \tag{5}$$

where $\mathbf{X}_t$ is the Itô diffusion

$$d\mathbf{X}_t = -\mathbf{v}(\mathbf{X}_t)dt + \sigma(\mathbf{X}_t)d\mathbf{W}_t, \quad \mathbf{X}_0 = \mathbf{x} \tag{6}$$

where $\mathbf{W}_t$ is n-dimensional Brownian motion.

Kolmogorov's Backward Equation tells us that the value of θ at a particular time and location $(t, \mathbf{x})$ is given by the average value of the initial condition evaluated at the position of the Itô diffusion (6) at time t when initialized at $\mathbf{x}$. This suggests a numerical method for evaluating $\mathcal{G}_j(u) = \theta(t_j, \mathbf{x}_j; u)$: (1) initialize a series of particles from $\mathbf{x}_j$; (2) simulate their movement to time t_j according to (6); (3) evaluate θ_0 at that location; and (4) take the average. See Fig. 1 for an illustration.

Numerical integration of (6) can be computed, for example, with an Euler-Maruyama approximation [14,15]

$$\mathbf{X}_{i+1} = \mathbf{X}_i - \mathbf{v}(\mathbf{X}_i)\Delta t_i + \sigma(\mathbf{X}_i)\sqrt{\Delta t_i}\xi_i, \tag{7}$$

or a Milstein approximation

[1] Throughout this paper, C^k (resp. C_b^k) denotes the space of functions with k continuous (resp. continuous and bounded) derivatives.

$$\mathbf{X}_{i+1} = \mathbf{X}_i - \mathbf{v}(\mathbf{X}_i)\Delta t_i + \sigma(\mathbf{X}_i)\sqrt{\Delta t_i}\xi_i + \frac{1}{2}\sigma(\mathbf{X}_i)\sigma'(\mathbf{X}_i)\left(\xi_i^2 - 1\right)\Delta t_i, \tag{8}$$

where $\mathbf{X}_i = \mathbf{X}(t_i)$, $\Delta t_i = t_{i+1} - t_i$, and $\xi_i \sim N(0,1)$. Higher-order integration methods are described in, e.g., [14, Chapter 14]. The resulting algorithm is described in Algorithm 3.1, where N_o is the number of observations and N_p is the number of particles used per observation.

Algorithm 3.1 Particle Method for Computing $\mathcal{G}_j(u)$.

1: Given u (e.g., $\mathbf{v}$, σ, and/or θ_0)
2: **for** $j = 1 \ldots N_o$ **do**
3: **for** $i = 1 \ldots N_p$ **do**
4: Set $\mathbf{X}_0^{(i)} = \mathbf{x}_j$
5: Simulate (6) with (7) or (8) to get $\mathbf{X}_{t_j}^{(i)}(u)$
6: Compute $g_j^{(i)}(u) = \theta_0\left(\mathbf{X}_{t_j}^{(i)}\right)$
7: **end for**
8: Compute $\mathcal{G}_j(u) = \theta(t_j, \mathbf{x}_j, u) \approx \frac{1}{N_p}\sum_i g_j^{(i)}(u)$
9: **end for**

3.2. A particle method for Problem 2.2

In this section, we present a particle method for computing $\mathcal{G}(u)$ for the boundary value problem Problem 2.2. The key idea is in the following theorem (see, e.g., [13, Corollary 9.1.2]), which identifies the solution to (2) in terms of Itô diffusions.

Theorem 3.2 *(Particle Solution to (2)). Let $f \in C_b(D)$, $\theta_{\partial D} \in C_b(\partial D)$, and $\mathbf{v}, \sigma \in C_b^1(D)$. As in (6), let $\mathbf{X}_t$ be the Itô diffusion*

$$d\mathbf{X}_t = -\mathbf{v}(\mathbf{X}_t)dt + \sigma(\mathbf{X}_t)d\mathbf{W}_t, \quad \mathbf{X}_0 = \mathbf{x}$$

and let τ_D denote the first exit time from D. Suppose $\tau_D < \infty$ almost surely for all $\mathbf{x} \in D$. Then if there exists bounded $\theta \in C^2\left(\overline{D}\right)$ satisfying (2), we have

$$\theta(\mathbf{x}) = \mathbb{E}\left[\theta_{\partial D}\left(\mathbf{X}_{\tau_D}\right)\right] - \mathbb{E}\left[\int_0^{\tau_D} f(\mathbf{X}_t)\,dt\right]. \tag{9}$$

See also [13, Sections 9.1-9.3] for related results on existence and uniqueness of solutions and [16, Chapter 9] for conditions guaranteeing that $\tau_D < \infty$ almost surely.

As in Section 3.1, this theorem presents a natural numerical approach to evaluating $\mathcal{G}_j(u) = \theta(\mathbf{x}_j, u)$: (1) initialize a series of particles from $\mathbf{x}_j$; (2) simulate their movement according to (6) until they exit D, integrating $f(\mathbf{X}_t)\,dt$ along the way; (3) evaluate $\theta_{\partial D}$ at the boundary location; (4) subtract the time integral of $f(\mathbf{X}_t)$; and (5) take the average. The resulting algorithm is described in Algorithm 3.2, where N_o is the number of observations and N_p is the number of particles used per observation.

Algorithm 3.2 Particle Method for Computing $\mathcal{G}_j(u)$ for Problem 2.2.

1: Given u (e.g., $\mathbf{v}$, σ, f, $\theta_{\partial D}$)
2: **for** $j = 1 \ldots N_o$ **do**
3: **for** $i = 1 \ldots N_p$ **do**
4: Set $\mathbf{X}_0^{(i)} = \mathbf{x}_j$
5: **while** $\mathbf{X}_t^{(i)} \in D$ **do**
6: Simulate particle position $\mathbf{X}_t^{(i)}$ with (7) or (8)
7: **end while**
8: Compute $g_j^{(i)}(u) = \theta_{\partial D}\left(\mathbf{X}_{\tau_D^{(i)}}^{(i)}\right) - \int_0^{\tau_D^{(i)}} f(\mathbf{X}_t^{(i)})\,dt$
9: **end for**
10: Compute $\mathcal{G}_j(u) = \theta(\mathbf{x}_j, u) \approx \frac{1}{N_p}\sum_i g_j^{(i)}(u)$
11: **end for**

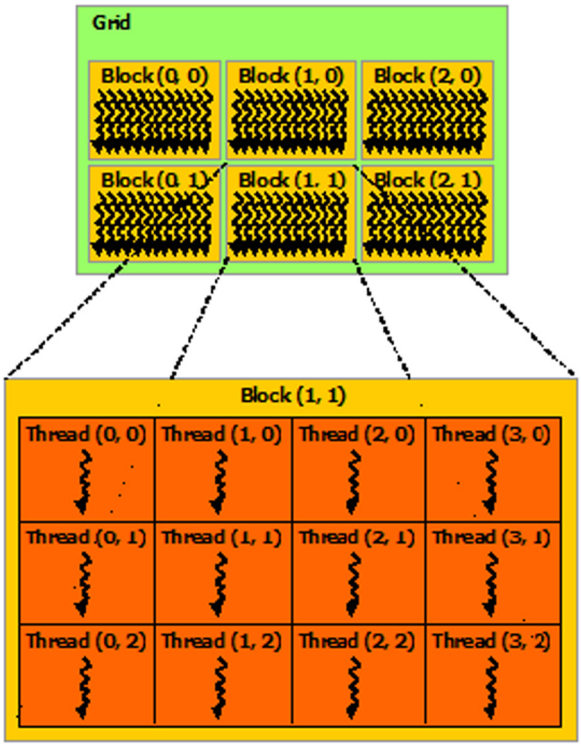

Fig. 2. GPU thread blocks [17].

3.3. Parallelization

Note that for a given observation point, all but the final step of Algorithm 3.1 or Algorithm 3.2 can be computed in parallel. In addition, the steps for separate observation points are entirely independent. As a result, the algorithms are embarrassingly parallel and can therefore be parallelized in a straightforward manner using any number of computational paradigms, such as message passing interface (MPI) processes or OpenMP threads, and naturally vectorize to leverage "single instruction multiple data" (SIMD) or "single instruction multiple thread" (SIMT) capabilities on modern CPUs or GPUs.

For example, Fig. 2, from NVIDIA's documentation [17], illustrates the layout of threads and blocks on an NVIDIA GPU. Each thread is a single execution unit that is grouped into a block and then run across one or more streaming multiprocessors. Algorithm 3.1 or Algorithm 3.2 can be ported to this architecture in a natural fashion: each observation can be assigned to a block or group of blocks, with each particle run in a separate thread. A single GPU can execute thousands of threads at a time, allowing thousands of particles to be simulated simultaneously on a single chip. Moreover, larger problems can be spread across multiple GPUs or multiple machines using MPI to gain even greater efficiency; the overhead is minimal as only the average value for each observation needs to be returned to the master process.

3.4. Computational complexity

In this section, we will compare the computational costs of Algorithm 3.1 to that of a reference Galerkin-based PDE solve. The analysis for Algorithm 3.2 is similar and we summarize the results at the end of this section. Consider the case where u is approximated by a basis expansion $u \approx \sum_{i=1}^{N_u} v_i \mathbf{e}_i$ and evaluation of each basis function $\mathbf{e}_i$ has computational cost C_b. We use the Euler-Maruyama approximation (7) and assume N_t timesteps per observation. Further, we assume evaluating $\mathcal{G}$ requires N_o observations and use N_p particles per observation. Then the computational cost of Algorithm 3.1 for P parallel processes/threads is given by

$$C_{particle} = O\left(\frac{1}{P} N_o N_p N_t N_u C_b\right). \tag{10}$$

Since the observations and particle simulations are almost entirely independent, they can be executed in parallel (see Section 3.3). So for large P we have the limit

$$C_{particle} \to O\left(N_t N_u C_b\right). \tag{11}$$

A more traditional method of solving the PDE would be to use a Galerkin projection, which would involve projecting the PDE (1) onto a set of basis functions $\{\phi\}_{l=1}^{N_b}$ to get a system of ODEs for the coefficients of θ:

$$\begin{aligned} M\dot{\Theta} &= A\Theta \\ M_{lm} &= \langle \phi_l, \phi_m \rangle \\ A_{lm} &= \left\langle \phi_l, -\mathbf{v}(\mathbf{x}) \cdot \nabla \phi_m + \frac{1}{2} \sum_{i,j} (\sigma \sigma^T)_{i,j}(\mathbf{x}) \frac{\partial^2}{\partial x_i \partial x_j} \phi_m \right\rangle. \end{aligned} \tag{12}$$

The bases ϕ_l could, for example, be spectral [18–20] or finite element basis functions. The system (12) is then integrated by repeated iteration of some combination of M, A (explicit, Runge-Kutta methods) and/or their inverses (implicit methods). Algorithm 3.3 outlines this algorithm for Explicit Euler time integration.

Algorithm 3.3 Reference Method for Computing $\mathcal{G}_j(u)$ (Galerkin, Explicit Euler).

1: Given u (e.g., $\mathbf{v}$, σ, and/or θ_0)
2: Project (1) onto $\{\phi\}_{l=1}^{N_b}$ to get system $M\dot{\Theta} = A\Theta$
3: **for** $i = 1 \dots N_t$ **do**
4: Multiply $\Theta_i = (M + \Delta t_i A)\, \Theta_{i-1}$
5: **end for**
6: **for** $j = 1 \dots N_o$ **do**
7: Compute $\mathcal{G}_j(u) = \sum_l \Theta_l(t_j)\phi_l(\mathbf{x}_j)$
8: **end for**

The computational costs of assembling the matrix A are heavily dependent on the choice of $\mathbf{v}$ and $\{\phi\}$ and therefore difficult to characterize in general. The cost of computing the observations is typically small. Ignoring these two factors, the computational cost of Algorithm 3.3 is dominated by the time integration of the system, which is made up of a series of matrix-vector multiplications. In general, the cost of this computation for N_b basis functions and N_t time steps is

$$C_{reference} \to O\left(N_t N_b^2\right). \tag{13}$$

Note also that to model an unknown of dimension N_u requires $N_b \geq N_u$. In practice, however, accurate modeling of θ may require $N_b \gg N_u$, particularly for small diffusion. Thus the ratio of the cost of the particle method (10) to the reference method (13) is

$$\frac{C_{particle}}{C_{reference}} = O\left(\frac{C_b N_u}{N_b^2}\right), \quad \text{so typically } \frac{C_{particle}}{C_{reference}} \ll O\left(\frac{C_b}{N_b}\right). \tag{14}$$

Thus for applications with a small enough number of observations or sufficient parallelism that a substantial proportion of the particles can be computed in parallel, the particle method should provide substantial speedup for large N_b and in particular for problems with high-dimensional unknowns.

We have of course made some simplifications in this analysis: For some choices of $\{\phi\}$ (e.g., discontinuous Galerkin [21]), the matrix A may be sparse or block diagonal, reducing the cost of the matrix multiply and increasing the effectiveness of parallelization. However, we have also ignored some of the costs of Algorithm 3.3 and most of the performance improvement comes from not computing the full field θ, so the computational complexity for large N_u or small N_o should be as indicated here for a large class of reference algorithms. A comparison of the computational cost of Algorithm 3.1 and Algorithm 3.3 for a sample problem is shown in Fig. 6.

The analysis for the boundary value method, Algorithm 3.2, is similar. The computational cost for the method is similar to that of Algorithm 3.1; the main difference is that the number of timesteps required to reach the boundary of D is uncertain and so may be larger or smaller than the largest t_j for the time dependent problem. A Galerkin approach to solving (2), meanwhile, would replace the time integration of the reference method Algorithm 3.3 with an iterative solver of a linear system $A\Theta = F$. Many efficient methods exist for solving these systems; however, because the number of basis functions required to represent θ would typically be much higher than the number of degrees of freedom in the parameter u, we still expect the particle method to yield significant efficiencies for applications with sparse observations (i.e., small N_o).

3.5. Discretization error and convergence

Numerical approximation of the true $\mathcal{G}(u)$ via Algorithm 3.1 or Algorithm 3.2 requires weak convergence of the underlying stochastic differential equation to the true value. As described in [14,15], this convergence typically involves two types of errors:

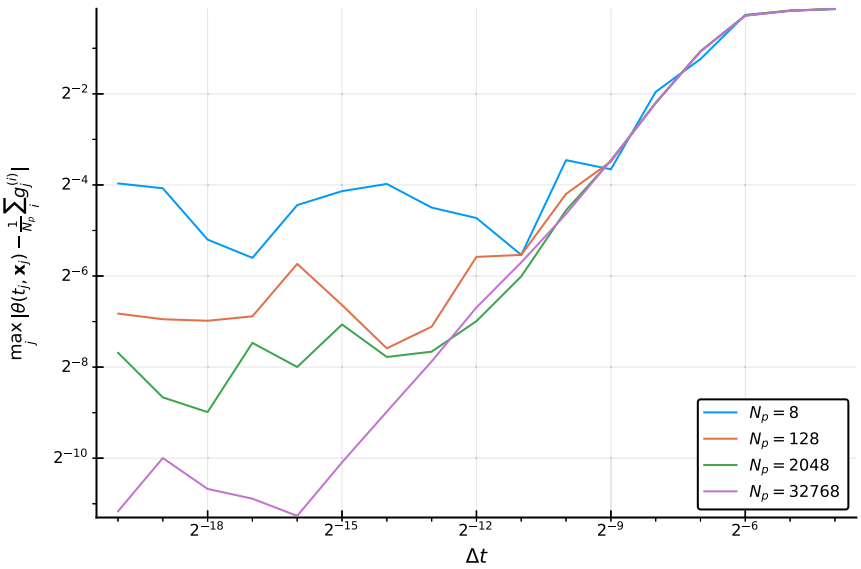

Fig. 3. Convergence for Algorithm 3.1 by time step Δt and number of particles N_p. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

1. *Discretization error* in approximating the time evolution of the underlying stochastic differential equation. The order of this error depends on the approximation used; for the Euler-Maruyama method (7), for example, weak convergence can be shown to be of order 1 (error proportional to Δt) when $\mathbf{v}, \sigma, \theta_0$ are sufficiently smooth (e.g., when $\mathbf{v}$, σ, and θ_0 are twice, twice, and four times continuously differentiable, respectively). See, e.g., [14, Chapter 14] for a description of this result for Euler-Maruyama and a description of methods with weak convergence of higher order in Δt.
2. *Sampling error* resulting from approximating the expected value by the mean of sample values. This error can be estimated by considering the variance of $\mathbb{E}g_j - \frac{1}{N_p}\sum_i g_j^{(i)}$, which according to the Central Limit Theorem scales like $\frac{1}{N_p}$. The sampling error therefore scales like $\frac{1}{\sqrt{N_p}}$.

In practice, the numerical error is a combination of these factors; the timestep Δt and the number of particles per observation N_p must be chosen to be sufficiently small and large, respectively, to obtain the desired level of approximation. To illustrate this point, Fig. 3 shows the convergence of Algorithm 3.1 for a sample problem with $\sigma^2 = 6 \times 10^{-5}$, $\theta_0 = \frac{1}{2} + \frac{1}{2}\sin(2\pi x)$, and $\mathbf{v}$ made up of 50 randomly-generated Fourier components (see (16)). Because an analytical solution was not available for this problem, the "true" value of $\theta(t_j, \mathbf{x}_j)$ was approximated via a simulation with $N_p = 2^{16}$ and $\Delta t = 2^{-20}$. The y axis shows the maximum error across $N_o = 8$ observation points. The time discretization in Algorithm 3.1 was implemented with the Euler-Maruyama method (7); as such, the error from time discretization is proportional to Δt when sufficiently many particles are used (see the trendline from top right to bottom left). The horizontal trends to the left of the main trendline (e.g., for $N_p = 8$ at error of roughly 2^{-4}) are cases where statistical error dominates discretization error – i.e., not enough particles were used to achieve optimal convergence. As expected, the statistical error (and therefore the meaning of "sufficiently many particles") scales like $\frac{1}{\sqrt{N_p}}$ – increasing the number of particles by 16 reduces the error by roughly a factor of 4.

For inverse problems that have been appropriately regularized, the effect of small errors in the forward model on the results should be small. See [1, Chapter 2] for a discussion of regularization; related theory for Bayesian inversion is provided in [2, Section 4.2]. In Bayesian problems, the numerical error may be modeled by probabilistic numerics as described in [22,23]; the distribution of results for individual particles (i.e., the distribution of $g_j^{(i)}, i = 1, \ldots, N_p$ in Algorithm 3.1 or Algorithm 3.2) would provide an estimate of the sampling error and therefore provide a reasonable input for probabilistic numerical models.

3.6. Limitations

For completeness, we identify three key limitations of using Algorithm 3.1 or Algorithm 3.2 to compute $\mathcal{G}$:

- As noted above, these inverse particle methods only apply to problems for which backward equation like Theorem 3.1 or Theorem 3.2 applies, i.e., where the underlying PDE is a driven diffusion.
- It is sometimes desirable to compute characteristics of θ beyond what is contained in $\mathcal{G}$. For example, [24] presents the statistics of the variance and variance dissipation of θ according to the Bayesian posterior on $\mathbf{v}$. Of course, because

these particle methods do not involve computing the full field θ, we cannot compute characteristics of θ beyond the values contained in $\mathcal{G}$. However, we note that the proposed particle algorithm could be used to compute the solution to the inverse problem, e.g., the maximum a posteriori u_{MAP}. Then a single, computationally-expensive PDE solve could be used to compute the characteristics of that solution – see, for example, the plot of $\theta(u_{MAP})$ in Fig. 7 or the plot of the optimal θ in Fig. 9.

- Many approaches to inverse problems (e.g., Hamiltonian Monte Carlo [25,26] for Bayesian inverse problems or Newton's method for optimization [9]) require computing the Fréchet derivative $D\mathcal{G}(u)$, which cannot be computed via the particle method in its current form. The gradient of $\mathcal{G}$ would therefore have to be approximated, e.g. via a finite-difference approximation involving multiple computations of $\mathcal{G}$.

4. Numerical examples

In this section, we provide examples demonstrating the applications and power of the particle methods Algorithm 3.1 and Algorithm 3.2. The examples were mostly written in the Julia numerical computing language [27]; the particle method was written in a combination of C and CUDA. The computations were run on the computational clusters at Virginia Tech,[2] with each particle computation using a single NVIDIA P100 GPU. We note that a multi-GPU implementation should yield further speedup for larger problem sizes, with minimal overhead due to the embarrassingly parallel nature of the particle method.

4.1. Example: Bayesian inference of fluid flows

In this section we present numerical examples of the application of Algorithm 3.1 to the Bayesian inverse problem [1,2] of estimating the background flow $\mathbf{v}$ from noisy point observations of θ with a constant diffusion coefficient on the two-dimensional periodic box $\mathbb{T}^2 = [0,1]^2$:

$$\frac{d\theta}{dt}(\mathbf{x}) = -\mathbf{v}(\mathbf{x}) \cdot \nabla\theta(t,\mathbf{x}) + \kappa\,\Delta\theta(t,\mathbf{x}), \quad \nabla\cdot\mathbf{v} = 0, \quad \theta(0,\mathbf{x}) = \theta_0(\mathbf{x}). \tag{15}$$

(In the notation of (1), $\sigma_{ij}(\mathbf{x}) = \sqrt{2\kappa}\delta_{ij}$; note that since σ is constant for this application, the Euler-Maruyama (7) and Milstein (8) approximations are equivalent.) Accurate simulation of the low-κ case is an active area of research [28–32]; in this case, the solution θ can become high-dimensional very quickly, making accurate PDE solves highly computationally expensive [24]. Thus the particle method will yield substantial benefit as the dimension of the background flow increases. We represent the background flow $\mathbf{v}$ via a divergence-free Fourier expansion in terms of wave numbers $\mathbf{k} \in \mathbb{Z}^2$

$$\mathbf{v}(\mathbf{x}) = \sum_{\mathbf{k}} v_{\mathbf{k}} \frac{\mathbf{k}^\perp}{\|\mathbf{k}\|} e^{2\pi i \mathbf{k}\cdot\mathbf{x}} \tag{16}$$

where $\mathbf{k}^\perp = [-k_y, k_x]$ so that $\mathbf{k}\cdot\mathbf{k}^\perp = 0$, and $v_{\mathbf{k}}$ obeys the reality condition $v_{\mathbf{k}} = -\overline{v_{-\mathbf{k}}}$. Thus, evaluating $\mathbf{v}(\mathbf{x})$ involves computing a series of sines and cosines; this represents the dominant cost in the particle method.

We will now compute the Bayesian posterior distribution for three example problems while using Algorithm 3.1 to compute the forward map $\mathcal{G}$. All results are computed using the preconditioned Crank-Nicholson Markov Chain Monte Carlo (MCMC) method. For the details of this particular application, we refer the reader to [24].

The first example will demonstrate consistency of the method with the traditional two-step method implemented in [24]. The second will demonstrate how the particle method allows extension of Example 1 to higher-dimensional vector fields. Finally, we conclude with an example where Algorithm 3.1 allows application to a problem where the true background flow contains many thousands of components. We note that for these Bayesian inference experiments, for simplicity we have used $N_b = N_u$ in the reference method; many applications would require $N_b \gg N_u$ (more basis functions to represent θ than $\mathbf{v}$), increasing the cost of the reference method.

4.1.1. Example 1: Consistency

In [24], the forward map $\mathcal{G}$ was computed with a two-step "solve, then observe" method: (15) was expanded in a Fourier basis $e^{2\pi i\mathbf{k}\cdot\mathbf{x}}$, projected into a system of ODEs, integrated in time using a Crank-Nicolson method, and then observed. In this section, we repeat the computations from Section 5.2 of that paper, in which $\mathbf{v}$ was assumed to have dimension of less than or equal to 197 and $\mathcal{G}$ involved 100 point observations of θ (i.e., $N_u = 197$ and $N_o = 100$). In this case, the dimension of $\mathbf{v}$ is low relative to the number of observations; thus the value of the particle method is limited – the PDE solves in [24] took approximately the same time as the GPU particle implementation. The results are shown simply to demonstrate that they are consistent for the two methods.

Fig. 4 shows the computed posterior one-dimensional and two-dimensional histograms for the first eight Fourier components of the background flow $\mathbf{v}$, for 10 million samples computed via the reference method from [24] and Algorithm 3.1.

[2] http://www.arc.vt.edu.

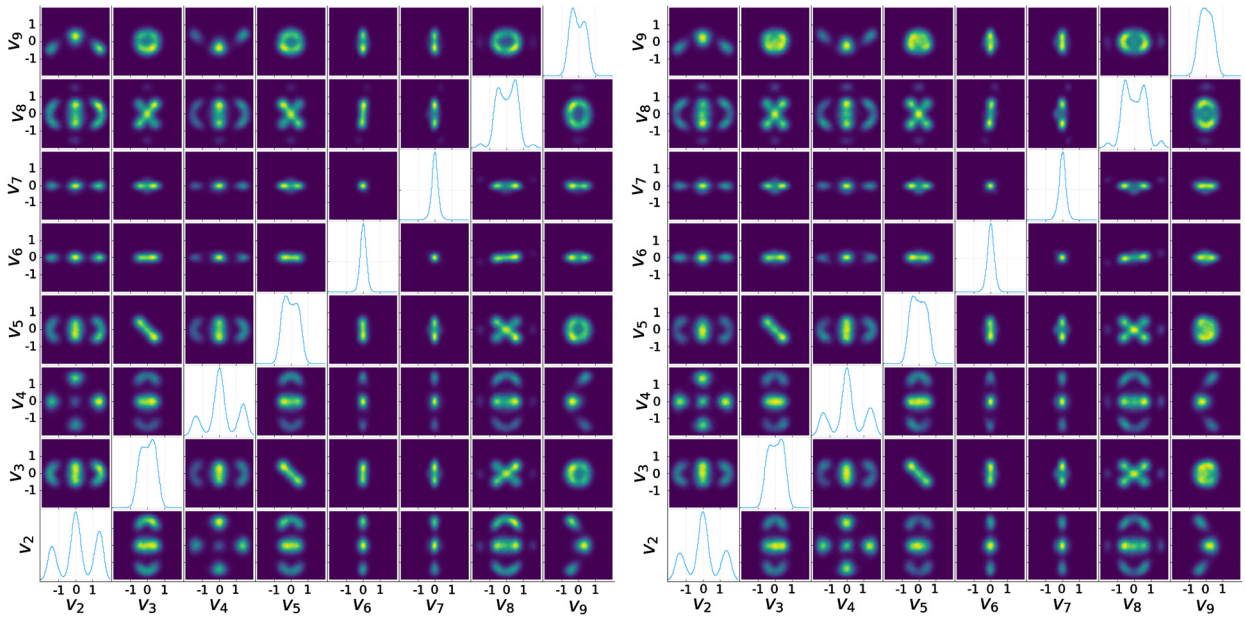

Fig. 4. Posterior histograms for the first eight components of $\mathbf{v}$ ($N_u = 197$). Left: Reference Method, see [24]. Right: Particle Method, Algorithm 3.1.

The posterior distributions have largely the same structure; smaller differences result from the fact that the posterior for this problem is highly complex and very slow to converge.

4.1.2. Example 2: Extension to higher-dimensions

In this example, we relax the assumption from Example 1 that the background flow has wave numbers with $\|\mathbf{k}\|_2 \leq 8$ ($N_u = 197$); here we allow $\|\mathbf{k}\|_2 \leq 32$ ($N_u = 3,209$). When computed using the reference method, the 16-fold increase in the dimension of $\mathbf{v}$ yielded an increase of over 300 in the computational cost of each sample, making it computationally intractable to generate enough samples to resolve the complex structure of the posterior distribution. By contrast, the computational cost of the GPU-based particle method only increased by a factor of 9, as some of the linear computational cost (10) was absorbed by the parallelism in the GPU.

To approximate the posterior, we generated 2.5 million samples via 100 separate 25,000-sample chains. Fig. 5 shows the resulting posterior histograms. When compared with Fig. 4, we see that the posterior shows additional regions of probability mass - see, for example, the peaks for $v_8 \approx \pm 1.5$. This indicates that the restriction of $\mathbf{v}$ to Fourier modes with $\|\mathbf{k}\|_2 \leq 8$ caused some possible candidate vector fields to be missed or de-emphasized.

4.1.3. Example 3: Turbulent background flows

We conclude the Bayesian examples by using the particle method to address a problem of much higher dimension than is considered in [24]. We again consider the Bayesian inverse problem of estimating $\mathbf{v}$ from measurements of θ (see (15)). However, this time we consider $\mathbf{v}$ made up of components of wave numbers with $\|\mathbf{k}\|_2 \leq 80$, a total of 20,081 components ($N_u = 20,081$). We use the same initial condition θ_0 and prior structure as in the previous two examples (though we allow the prior to extend to higher dimensions). In this example, we use $\kappa = 3 \times 10^{-5}$ from [33], which is typical for diffusion in water. We generate data by drawing a velocity field from the prior and computing θ at 13 evenly-spaced times between 0 and 0.5 at each of two observation locations: $[0, 0], [\frac{1}{2}, \frac{1}{2}]$.

Computing a Galerkin approximation of θ for this problem would require many matrix multiplications each of tens of thousands of rows and columns, making the computation of many thousands of samples, in general, computationally intractable. Using Algorithm 3.1 to compute the forward map $\mathcal{G}$, however, the computation scales roughly linearly with the number of unknowns, as shown in Fig. 6.

Fig. 7 shows the vorticity and norm of the maximum a priori (MAP) point from 100,000 computed pCN MCMC samples; this can be thought of as the background flow that best matches both the prior measure and the observations for a given model of observational noise. The figures show the complexity of the background flows that could be considered in the inference by allowing the inclusion of tens of thousands of parameters.

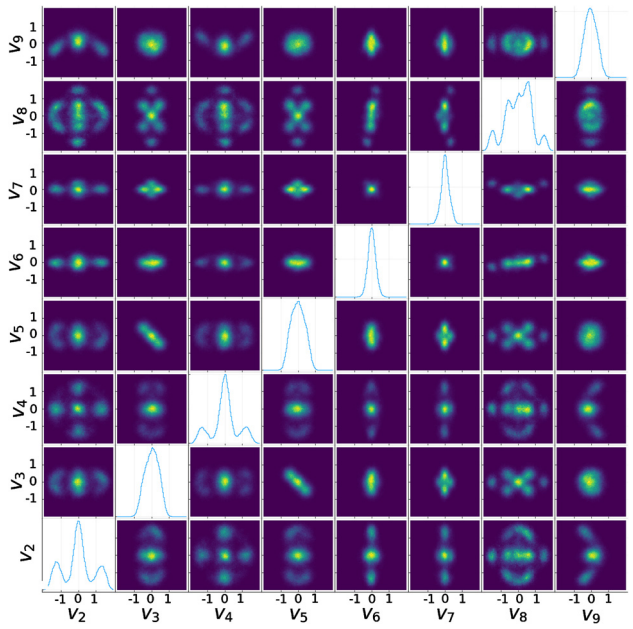

Fig. 5. Posterior histograms for the first eight components of $\mathbf{v}$ ($N_u = 3,209$).

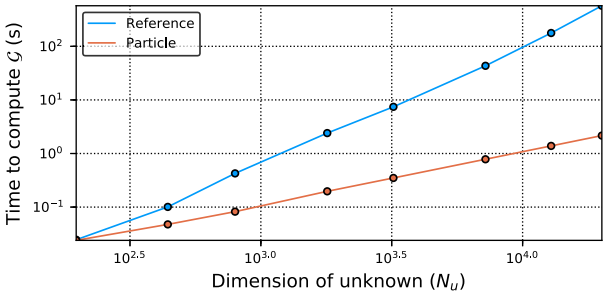

Fig. 6. Time in seconds to compute $\mathcal{G}$ by dimension of unknown (N_u) for reference (Fourier) and particle methods.

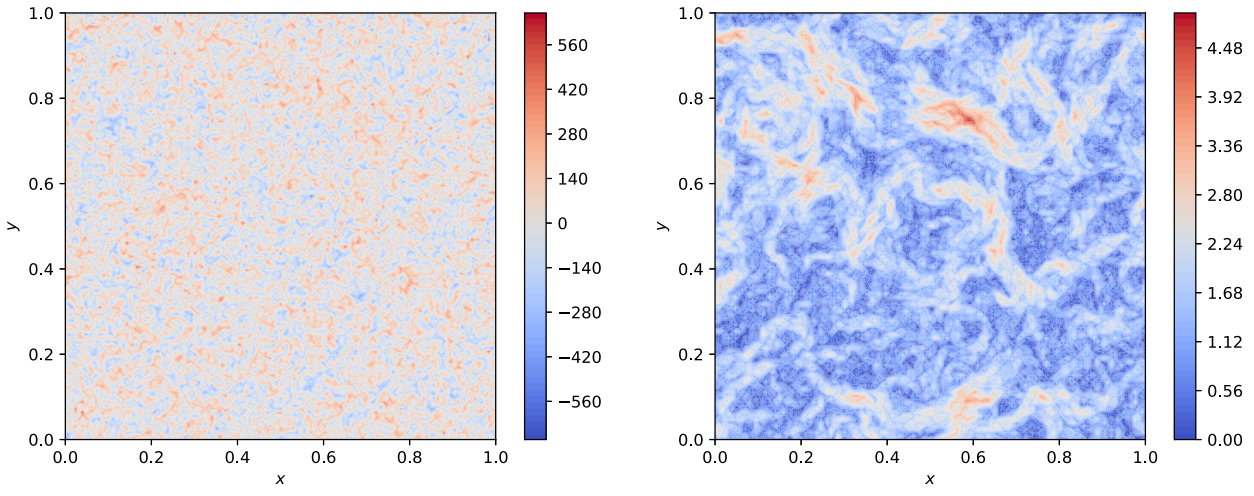

Fig. 7. Contour plots of vorticity (left) and norm (right) of $\mathbf{v}_{MAP}$.

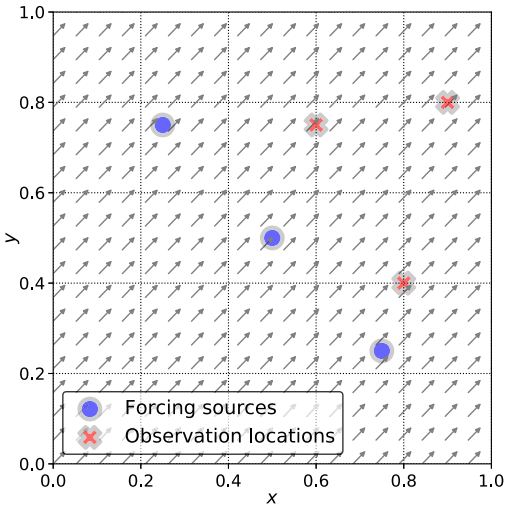

Fig. 8. Forcing locations, observation locations, and background flow.

4.2. Optimal forcing for boundary value problem

In this section, we provide an example application of boundary value method Algorithm 3.2. For this example, we consider a version of the boundary value problem Problem 2.2 with laminar background flow, constant diffusion, known boundary conditions, but to-be-determined forcing:

$$\begin{aligned} -\begin{bmatrix} 1 \\ 1 \end{bmatrix} \cdot \nabla\theta(\mathbf{x}) + \kappa\,\Delta\theta(\mathbf{x}) &= \sum_{j=1}^{3} f_j \exp\left(-4\left\|\mathbf{x}-\mathbf{x}_f^{(j)}\right\|_2^2\right) \quad \mathbf{x}\in D := [0,1]^2 \\ \theta(\mathbf{x}) &= \frac{1}{2}\left[\cos\left(\frac{\pi}{2}x\right)+\cos\left(\frac{\pi}{2}y\right)\right] \quad \mathbf{x}\in \partial D. \end{aligned} \tag{17}$$

The goal of the problem is to find the forcing coefficients $\mathbf{F} = [f_1, f_2, f_3]$ that produce the scalar field θ that best matches the target values $\mathcal{Y}$ at three observation locations $\left\{\mathbf{x}_{obs}^{(i)}\right\}$. That is, we seek

$$\mathbf{F}^\star := \arg\min_{\mathbf{F}} \|\mathcal{Y} - \mathcal{G}(\mathbf{F})\|, \quad \mathcal{G}(\mathbf{F}) = \left[\theta\left(\mathbf{x}_{obs}^{(1)},\mathbf{F}\right), \theta\left(\mathbf{x}_{obs}^{(2)},\mathbf{F}\right), \theta\left(\mathbf{x}_{obs}^{(3)},\mathbf{F}\right)\right]. \tag{18}$$

The problem is thus an optimal control problem and can be interpreted as, e.g.,

- Find the heat sources or sinks that produce the desired temperature at important locations in a room
- Find the forcing that minimizes the concentration of a chemical at key locations in a system (see, e.g., [34,35] for examples with time-varying forcing)

The forcing locations $\left\{\mathbf{x}_f^{(j)}\right\}$ and observation locations $\left\{\mathbf{x}_{obs}^{(i)}\right\}$ used in this example are shown in Fig. 8. For this problem, we use diffusion coefficient $\kappa = 0.282$, for water diffusing in air [36], and seek to match data $\mathcal{Y} = [0, 0, 0]$.

Recall that the forward map $\mathcal{G}$ is sparse for this problem - we only need θ evaluated at three points. We can therefore leverage Algorithm 3.2 to speed up the computations. (For this example, the particle exit time $\tau_D < \infty$ almost surely; see [16, Chapter 9] for details.) We use a Nelder-Mead simplex method [37,38] as implemented in Julia's Optim package [39] to seek the optimal $\mathbf{F}$. The method required evaluating $\mathcal{G}$ for thousands of possible values of $\mathbf{F}$. The requirement of multiple evaluations of the forward map is typical of many approaches to PDE-constrained optimization problems [3]. The choice of Nelder-Mead is merely used to demonstrate the effectiveness of Algorithm 3.2 within an optimization setting.

The results, θ and f, for the computed optimal forcing are shown in Fig. 9; notice that the observation points lie along the contour $\theta = 0$, indicating that we have found a set of parameters that are a good match for the data. (These plots, which required approximation of θ for the full domain, were generated via a finite element solver after $\mathbf{F}^\star$ was computed via the particle method; the plots required more work to generate than finding the optimal set of coefficients.)

We note that other, similar optimization problems – for example, fixing the forcing and instead seeking $\mathbf{v}$, κ, and/or boundary conditions that produce θ best matching the data – could be addressed via an analogous approach, again by leveraging Algorithm 3.2 to compute θ at the observation points for each value of the parameter considered by the optimization algorithm.

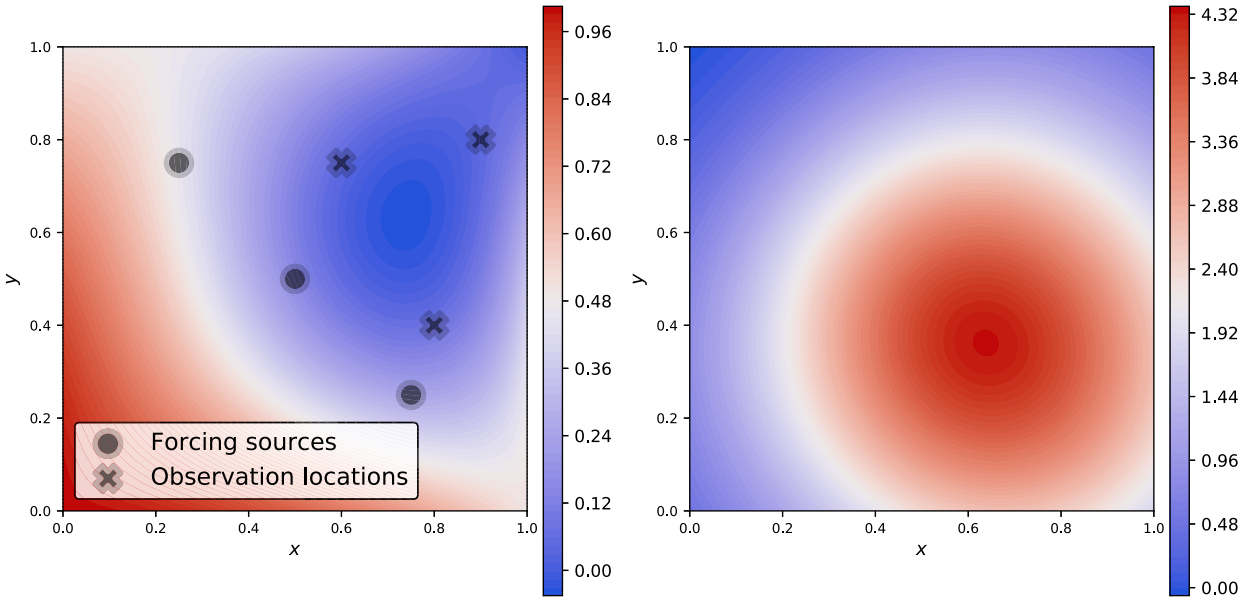

Fig. 9. Scalar field θ (left) and forcing function (right) for computed optimal coefficients $\mathbf{F}^*$.

Acknowledgements

This work was supported in part by the National Science Foundation under grants DMS-1313272, DMS-1522616, and DMS-1819110; the National Institute for Occupational Safety and Health under grant 200-2014-59669; and the Simons Foundation under grant 515990. We would also like to thank the Mathematical Sciences Research Institute, the Tulane University Math Department, and the International Centre for Mathematical Sciences (ICMS) where significant portions of this project were developed and carried out.

The authors acknowledge Advanced Research Computing at Virginia Tech[3] for providing computational resources and technical support that have contributed to the results reported within this paper.

References

[1] J. Kaipio, E. Somersalo, Statistical and Computational Inverse Problems, Applied Mathematical Sciences, vol. 160, Springer Science & Business Media, 2005.
[2] M. Dashti, A.M. Stuart, The Bayesian approach to inverse problems, in: Handbook of Uncertainty Quantification, 2017, pp. 311–428.
[3] M. Hinze, R. Pinnau, M. Ulbrich, S. Ulbrich, Optimization with PDE Constraints, Mathematical Modelling: Theory and Applications, vol. 23, Springer, New York, 2009.
[4] L.T. Biegler, O. Ghattas, M. Heinkenschloss, B. van Bloemen Waanders, Large-scale PDE-constrained optimization: an introduction, in: Large-Scale PDE-Constrained Optimization, Springer, 2003, pp. 3–13.
[5] T. Bui-Thanh, O. Ghattas, J. Martin, G. Stadler, A computational framework for infinite-dimensional Bayesian inverse problems Part I: The linearized case, with application to global seismic inversion, SIAM J. Sci. Comput. 35 (2013) A2494–A2523.
[6] N. Petra, J. Martin, G. Stadler, O. Ghattas, A computational framework for infinite-dimensional Bayesian inverse problems, Part II: Stochastic Newton MCMC with application to ice sheet flow inverse problems, SIAM J. Sci. Comput. 36 (2014) A1525–A1555.
[7] P.T. Boggs, K.R. Long, S.B. Margolis, P.A. Howard, Rapid source inversion for chemical/biological attacks, part 1: The steady-state case, SIAM J. Optim. 17 (2006) 430–458.
[8] S. Brooks, A. Gelman, G. Jones, X.-L. Meng, Handbook of Markov Chain Monte Carlo, CRC Press, 2011.
[9] J. Nocedal, S.J. Wright, Numerical Optimization, Springer, 2006.
[10] A.J. Chorin, Numerical study of slightly viscous flow, J. Fluid Mech. 57 (1973) 785–796.
[11] P. Degond, S. Mas-Gallic, The weighted particle method for convection-diffusion equations. I. The case of an isotropic viscosity, Math. Comput. 53 (1989) 485–507.
[12] A.J. Majda, A.L. Bertozzi, Vorticity and Incompressible Flow, vol. 27, Cambridge University Press, 2002.
[13] B. Øksendal, Stochastic Differential Equations. An Introduction with Applications, vol. 5, 6th ed., Springer, New York, 2013.
[14] P.E. Kloeden, E. Platen, Numerical Solution of Stochastic Differential Equations, Springer, 1999.
[15] D.J. Higham, An algorithmic introduction to numerical simulation of stochastic differential equations, SIAM Rev. 43 (2001) 525–546.
[16] A. Friedman, Stochastic Differential Equations and Applications, vol. 1, Academic Press, 1975.
[17] NVIDIA, NVIDIA CUDA C Programming Guide, NVIDIA Corporation, 2018.
[18] C. Canuto, M. Hussaini, A. Quarteroni, T. Zang, Spectral Methods: Fundamentals in Single Domains, Scientific Computation, Springer, Berlin Heidelberg, 2007.

[3] http://www.arc.vt.edu.

[19] C. Canuto, M. Hussaini, A. Quarteroni, T. Zang, Spectral Methods: Evolution to Complex Geometries and Applications to Fluid Dynamics, Scientific Computation, Springer, Berlin Heidelberg, 2007.
[20] D. Gottlieb, S. Orszag, S. for Industrial, A. Mathematics, Numerical Analysis of Spectral Methods: Theory and Applications, CBMS-NSF Regional Conference Series in Applied Mathematics, Society for Industrial and Applied Mathematics, 1977.
[21] J.S. Hesthaven, T. Warburton, Nodal Discontinuous Galerkin Methods: Algorithms, Analysis, and Applications, Springer Science & Business Media, 2007.
[22] P. Hennig, M.A. Osborne, M. Girolami, Probabilistic numerics and uncertainty in computations, Proc. R. Soc., Ser. A 471 (2015) 20150142.
[23] J. Cockayne, C. Oates, T. Sullivan, M. Girolami, Probabilistic numerical methods for PDE-constrained Bayesian inverse problems, AIP Conf. Proc. 1853 (2017) 060001.
[24] J. Borggaard, N. Glatt-Holtz, J. Krometis, A Bayesian approach to estimating background flows from a passive scalar, preprint, arXiv:1808.01084, 2018, submitted for publication.
[25] N. Bou-Rabee, J.M. Sanz-Serna, Geometric integrators and the Hamiltonian Monte Carlo method, Acta Numer. 27 (2018) 113–206.
[26] R.M. Neal, MCMC using Hamiltonian dynamics, in: Handbook of Markov Chain Monte Carlo, vol. 2, 2011, pp. 113–162.
[27] J. Bezanson, A. Edelman, S. Karpinski, V.B. Shah Julia, A fresh approach to numerical computing, SIAM Rev. 59 (2017) 65–98.
[28] M. Stynes, Numerical methods for convection-diffusion problems or the 30 years war, preprint, arXiv:1306.5172, 2013.
[29] K.W. Morton, Numerical Solution of Convection-Diffusion Problems, Chapman & Hall, 1996.
[30] R. Codina, On stabilized finite element methods for linear systems of convection–diffusion-reaction equations, Comput. Methods Appl. Mech. Eng. 188 (2000) 61–82.
[31] W. Hundsdorfer, J.G. Verwer, Numerical Solution of Time-Dependent Advection-Diffusion-Reaction Equations, vol. 33, Springer Science & Business Media, 2013.
[32] M. Mudunuru, K. Nakshatrala, On enforcing maximum principles and achieving element-wise species balance for advection–diffusion–reaction equations under the finite element method, J. Comput. Phys. 305 (2016) 448–493.
[33] S. Chen, R.H. Kraichnan, Simulations of a randomly advected passive scalar field, Phys. Fluids (1994) 10, 1998.
[34] L.-C. Chang, C.A. Shoemaker, P.L.-F. Liu, Optimal time-varying pumping rates for groundwater remediation: application of a constrained optimal control algorithm, Water Resour. Res. 28 (1992) 3157–3173.
[35] G.J. Whiffen, C.A. Shoemaker, Nonlinear weighted feedback control of groundwater remediation under uncertainty, Water Resour. Res. 29 (1993) 3277–3289.
[36] E.L. Cussler, Diffusion: Mass Transfer in Fluid Systems, Cambridge University Press, 2009.
[37] J.A. Nelder, R. Mead, A simplex method for function minimization, Comput. J. 7 (1965) 308–313.
[38] J.C. Lagarias, J.A. Reeds, M.H. Wright, P.E. Wright, Convergence properties of the Nelder-Mead simplex method in low dimensions, SIAM J. Optim. 9 (1998) 112–147.
[39] P.K. Mogensen, A.N. Riseth, Optim: a mathematical optimization package for Julia, J. Open Source Softw. 3 (2018) 615.