Available online at www.sciencedirect.com
ScienceDirect
Comput. Methods Appl. Mech. Engrg. 384 (2021) 113976
Computer methods in applied mechanics and engineering
www.elsevier.com/locate/cma

Solving inverse problems in stochastic models using deep neural networks and adversarial training

Kailai Xu[a], Eric Darve[a,b,*]

[a] Institute for Computational and Mathematical Engineering, Stanford University, Stanford, CA, 94305, United States of America
[b] Mechanical Engineering, Stanford University, Stanford, CA, 94305, United States of America

Available online xxxx

Abstract

Inverse problems associated with stochastic models constitute a significant portion of scientific and engineering applications. In such cases the unknown quantities are distributions. The applicability of traditional methods is limited because of their demanding assumptions or prohibitive computational consumption; for example, maximum likelihood methods require closed-form density functions, and Markov Chain Monte Carlo needs a large number of simulations. We propose a new method that estimates the unknown distribution by matching the statistical properties between observed and simulated random processes. We leverage the expressive power of neural networks to approximate the unknown distribution and use a discriminative neural network for computing the statistical discrepancies between the observed and simulated random processes. We demonstrated numerically that the proposed methods can estimate both the model parameters and learn complicated unknown distributions.
© 2021 Elsevier B.V. All rights reserved.

Keywords: Adversarial training; Neural networks; Automatic differentiation

1. Introduction

1.1. Problem statement

Almost all model-based problems in data analytics and scientific computing can be classified into two categories: forward problems and inverse problems. The forward problems produce predictions given model parameters, such as physical parameters in partial differential equations. However in the inverse problem, part of the parameters in the models are unknown but we observe (partial) outputs. The unknown parameters are calibrated from both the physical models and observations [1–3]. Among inverse problems, those associated with stochastic models are particularly interesting since stochasticity is indispensable for modeling real-world phenomenon.

Later on, we will provide a general setup for the stochastic inverse modeling problems we intend to solve. But to start, we describe some of the specific benchmarks we will use for our method. This will set the stage for discussing our approach:

* Corresponding author at: Mechanical Engineering, Stanford University, Stanford, CA, 94305, United States of America.
E-mail addresses: kailaix@stanford.edu (K. Xu), darve@stanford.edu (E. Darve).

https://doi.org/10.1016/j.cma.2021.113976
0045-7825/© 2021 Elsevier B.V. All rights reserved.

Table 1
The known random process w, unknown random variable θ and the corresponding distribution $\mathcal{P}$, and the output u. In the probability column, DNN (deep neural network) means the form of the probability distribution is not specified.

Example	w	θ	$\mathcal{P}$	$u(x,t)$
Uncertainty quantification	–	θ	DNN	$u(x)$
Stochastic differential equation	W_t	(κ, τ, σ)	Known functional form	r_t

- In **uncertainty quantification** [4,5], we have a partial differential equation (PDE) model

$$\begin{aligned} \mathcal{L}_\theta u(x) &= 0, \quad x \in \Omega, \\ \mathcal{B}u(x) &= 0, \quad x \in \partial\Omega, \end{aligned} \tag{1}$$

  where $\mathcal{L}_\theta$ is a differential operator parametrized by an unknown θ, and $\mathcal{B}$ is the operator associated with the boundary condition. Due to the stochastic nature of the physical process, θ is not deterministic and it has an unknown distribution $\mathcal{P}$. Therefore, the resultant solution u is a random variable. Given multiple observations u_i for u, we wish to reconstruct the unknown distribution $\mathcal{P}$. See Sections 4.1, 4.2, and 4.3 for numerical benchmarks.
- In **finance**, the stochastic differentiation equations can be used for modeling interest rates. For example, the square root process given by [6]

$$dr_t = \kappa(\tau - r_t)dt + \sqrt{r_t}\sigma dW_t, \tag{2}$$

  where r_t is the interest rate, $\{W_t, t \geq 0\}$ is a standard Brownian motion and $\theta = (\kappa, \tau, \sigma)$ are model parameters. To describe the dynamics of the interest rates, we wish to calibrate the parameters θ based on discrete observations $r_{t_1}, r_{t_2}, \ldots, r_{t_n}$ at time $t = t_1, t_2, \ldots, t_n$. See Section 4.4 for a numerical benchmark.

We can formulate Eqs. (1) and (2) using a unified mathematical model. We assume that we have a map F such that

$$F : (w, \theta) \mapsto u(\cdot, \cdot). \tag{3}$$

The variables w, θ and u (typically a function of a spatial variable x and time variable t) are defined as follows:

1. w is sampled from a known stochastic process, and we assume that we have a numerical procedure to generate w samples. However, the samples w themselves are not included in the observations.
2. θ is a random variable with a probability density function $\mathcal{P}$. In some special cases, the functional form of θ is known (or assumed), e.g., Gaussian distributions. In that case, we seek to estimate the parameters of the functional form, e.g., mean and variance of the Gaussian distribution. In the general case when the functional form is unknown, we propose to approximate θ with a generative deep neural network (DNN). The goal is then to estimate the weights and biases of the DNN.
3. $u(x,t)$ is the output of the mapping. It is also a random variable or process. u typically depends on the location x and time t; in some cases, u may only depend on location or time. Usually only a few samples of u are observable, and those samples are denoted as $\{u_i\}_{i=1}^n$.

The goal is to estimate the distribution $\mathcal{P}$ of θ from the observations $\{u_i\}_{i=1}^n$. Table 1 discusses the corresponding w, θ, and u in the previous examples.

When the distributions in the models are non-Gaussian, there are few analytical tools available and therefore we must resort to a numerical approach. Traditionally, numerical methods have been mostly developed for inverse problems associated with deterministic models or Gaussian-based stochastic models. Methods for non-Gaussian cases are relatively limited.

1.2. Review of existing methods and the state of the art

We now give a brief review of traditional methods for inverse modeling of Eq. (3). We discuss two cases separately.

Table 2
Comparison of AIM (this paper) and existing state-of-the-art methods.

Method	Requires sampling	Requires log likelihood	Has explicit loss functions	Parametrizes $\mathcal{P}$
Gaussian-mixture	✗	✓	✓	✓
Bayes inference	✗	✓	✓	✗
MCMC	✓	✗	✗	✗
AIM	✓	✗	✓	✓

(1) When the functional form of $\mathcal{P}$ is known, the inverse problem for Eq. (3) is reduced to a parameter estimation problem. Let the unknown parameters of $\mathcal{P}$ be $\boldsymbol{\theta}$. The traditional starting point of estimating $\boldsymbol{\theta}$ is the maximum likelihood estimator, which maximizes the log-likelihood function

$$\max_{\boldsymbol{\theta}} \; l_{\boldsymbol{\theta}}(u_1, u_2, \ldots, u_N) := \sum_{i=1}^{N} \log p(u_i|\boldsymbol{\theta}),$$

where $\{u_i\}_{i=1}^N$ are observed samples from the random process u.[1] The maximum likelihood method (MLE) is provably asymptotically efficient, that is, consistent and asymptotically normal with variance equal to the Cramér–Rao lower bound under certain conditions; typical conditions are given in [7–9]. However, MLE requires computing $p(u_i|\boldsymbol{\theta})$. When F is a very complicated model, it is difficult to obtain the analytical form of $p(u_i|\boldsymbol{\theta})$ or compute it numerically.

(2) In the second case, the functional form of $\mathcal{P}$ is unknown, we can approximate $\mathcal{P}$ using various functional forms. We describe three commonly used approaches in the following.

(a) Approximating $\mathcal{P}$ by a known distribution with tunable parameters. For example, Gaussian-mixture distributions [10] are used in variational inference for approximating the unknown distribution $\mathcal{P}$ [11]. Consequently, the problem is reduced to estimating the covariances $\boldsymbol{\Sigma}$ and the means $\boldsymbol{\mu}$ of the mixed Gaussian distributions. The surrogate mathematical model can be formulated as

$$u = F(w, \hat{\theta}(\boldsymbol{\Sigma}, \boldsymbol{\mu})),$$

where $\hat{\theta}(\boldsymbol{\Sigma}, \boldsymbol{\mu})$ is the Gaussian-mixture distribution and is used to approximate θ. Even though w is known, the log probability $p(u|\boldsymbol{\Sigma}, \boldsymbol{\mu})$ is usually analytically intractable due to the complicated nature of F, and therefore presents a major challenge for maximum likelihood methods for estimating $\boldsymbol{\Sigma}$, $\boldsymbol{\mu}$. Additionally, a Gaussian-mixture model may not be sufficient to approximate the unknown distribution $\mathcal{P}$.

(b) Bayes inference method [12–16]. In this approach, given new observations u_i, we update our estimation of θ following Bayes rule

$$p(\theta|u_i) = \frac{p(u_i|\theta)p(\theta)}{p(u_i)}.$$

The Bayes inference method requires us to specify a prior $p(\theta)$, which can be difficult to justify in the absence of a physical basis or a plausible scientific model; besides, it also requires calculating $p(u_i|\theta)$, which can be difficult to compute numerically.

(c) Markov Chain Monte Carlo (MCMC) [17–19]. MCMC can create samples from $\mathcal{P}$ by constructing a Markov chain. The method is based on sampling, may be slow to converge [20].

A comparison of the different approaches is given in Table 2.

1.3. Adversarial inverse modeling (AIM)

Our approach, adversarial inverse modeling (AIM), is based on minimizing the discrepancy between the actual random process $u = F(w, \theta)$ and estimated random process $\hat{u} = F(w, \hat{\theta})$. Here $\hat{\theta}$ is our current estimation of the true distribution θ and is updated as the minimization proceeds. AIM can be applied to estimate the distribution $\mathcal{P}$ of θ, whose functional form is either known or unknown. The key is that we can formulate the discrepancy

[1] Here log is the natural logarithm, sometimes denoted ln.

between statistical properties of u and $\hat{u}$ with a neural network, and the corresponding loss function is called the discriminator loss. The idea originates from **generative adversarial nets (GAN),** where a neural network D_ξ, parametrized by ξ, tries to discriminate the true random variable u from the estimated random variable $\hat{u}$. Meanwhile, we update $\hat{\theta}$ to generate samples $\hat{u}$ that mimic the distribution of u. By updating the neural network D_ξ and $\hat{\theta}$ simultaneously, at equilibrium, we obtain a $\hat{\theta}$ such that D_ξ cannot discriminate u and $\hat{u}$. In this case, the distribution $\hat{u}$ is indistinguishable from u, under the probability metric defined in the next paragraph.

How can the discriminator neural network measure the discrepancy between two probability distributions? The answer is that the choice of the loss functions $L^D(u, \hat{u}; \xi)$ for updating ξ (assuming $\hat{\theta}$ is fixed), and the loss functions $L^F(\hat{u}(\hat{\theta}))$ for updating $\hat{\theta}$ (assuming ξ is fixed) together determines the probability metric. For example, by properly choosing L^D and L^F (see Eq. (11) for such an example), we actually obtain the maximum likelihood estimator for θ at equilibrium and the corresponding probability distance metric is the Kullback–Leibler divergence. Other choices of divergence include the Jensen–Shannon divergence and the Wasserstein distance [21].

In the case where the functional form of $\mathcal{P}$ is known, we need to estimate the parameter $\boldsymbol{\theta}$ of the functional form. Estimating the distribution $\mathcal{P}$ with an unknown functional form is more challenging. When the functional form of $\mathcal{P}$ is known but $\mathcal{P}$ is **low-dimensional**, conventional functional forms such as linear basis functions, radial basis functions, and polynomial basis functions can be used to approximate the density function $\mathcal{P}$. The problem is then reduced to estimating some coefficients.

However, when $\mathcal{P}$ is a **high-dimensional** distribution and its functional form is unknown, the aforementioned methods can be too computationally expensive. For example, grid-based linear basis functions suffer from the curse of dimensionality because it requires tessellating a high dimensional domain. In this context, neural networks are very good candidates to approximate high-dimensional distributions by transforming simple distributions, such as Gaussian distributions, to very complex ones. For example, the neural networks are used to approximate very complex distributions in GANs, where they are called **generative neural networks**. We adopt this approach in AIM and use a neural network $G_\eta : \mathbb{R}^{d'} \to \mathbb{R}^d$ to represent d-dimensional unknown distributions $\mathcal{P}$ (d' and d are not necessarily equal). The generative neural net $G_\eta : \mathbb{R}^{d'} \to \mathbb{R}^d$ transforms a random variable u whose distribution is easy to sample from, e.g., a multivariate Gaussian distribution $\mathcal{N}(0, I_{d'})$, to another random variable $G_\eta(u)$. The goal is then to find optimal weights and biases η so that $G_\eta(u)$ has a distribution similar to θ. Fig. 1 illustrates the model formulation of AIM and Table 2 compares the difference between AIM and traditional approaches.

Organization of paper. This paper is organized as follows. In Section 2, we introduce adversarial inverse modeling, and propose an optimization algorithm for the framework. In Section 3, we analyze the convergence of AIM based on Kullback–Leibler divergence, a special case of AIM for application in parameter calibration of CIR processes. In Section 4, extensive numerical experiments are carried out, which include applications in uncertainty quantification, parameter inference, and option pricing. The numerical experiments also demonstrate our analysis in Section 3. Finally, we conclude with a general discussion of the method in Section 5.

2. Adversarial inverse modeling

2.1. Overview of the numerical algorithm

We now discuss how AIM can be numerically implemented. The idea is adversarial training, where we simultaneously (1) update $\hat{\theta}$ such that the discriminator D_ξ cannot distinguish u and $\hat{u}$; (2) update ξ so such the discriminator is better at distinguishing u and $\hat{u}$. At equilibrium, the generated outputs $\hat{u}$ will be indistinguishable from real random processes u in the sense of small probability discrepancy, determined by L^D and L^F. To be more concrete, assume that we have multiple observations $\{u_i\}$, the algorithm conducts the following steps iteratively until the convergence criterion has been met.

1. Generate realizations of w: w_i, $i = 1, 2, \ldots, N$.
2. Generate realizations of $z \sim \mathcal{N}(0, I_{d'})$: z_i, $i = 1, 2, \ldots, N$.
3. Compute $\hat{u}_i = F(w_i, G_\eta(z_i))$.
4. Compute the discrepancy between predictions $\{\hat{u}_i\}$ and observations $\{u_i\}$ in terms of probability distribution metrics: $d = L^D(\{u_i\}, \{\hat{u}_i\}; \xi)$, $i = 1, 2, \ldots, N$.
5. Update ξ ($\alpha_1 > 0$ is the step size):

$$\xi \leftarrow \xi - \alpha_1 L^D(\{u_i\}, \{\hat{u}_i\}; \xi); \tag{4}$$

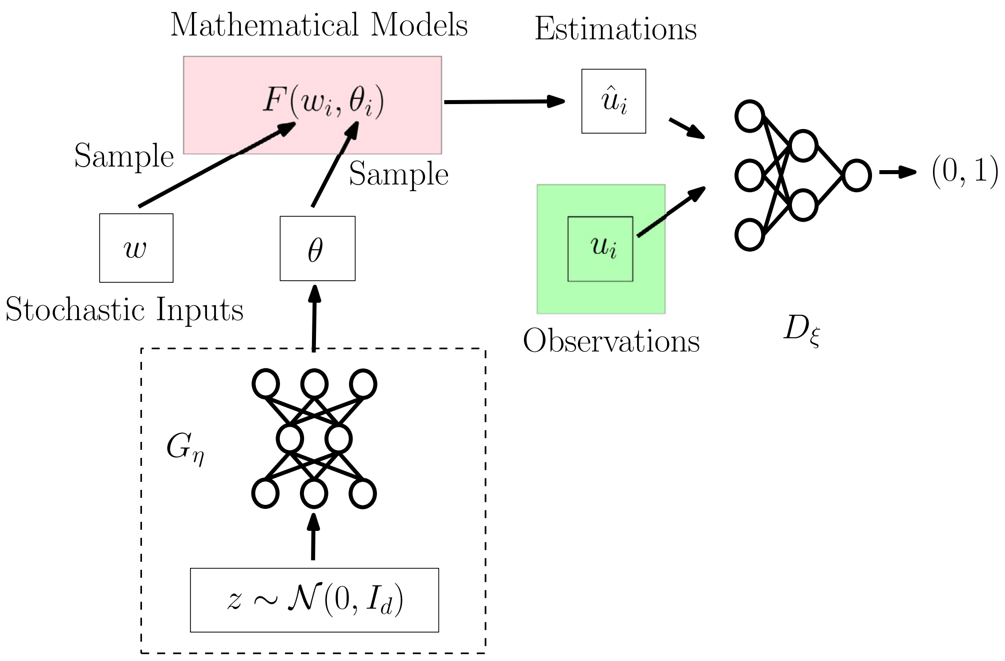

Fig. 1. Graphical illustration of adversarial inverse modeling. $\{u_i\}$ are observations, w_i, θ_i are realizations of w and θ respectively ($\theta = G_\eta(u)$ is an approximation to the true distribution $\mathcal{P}$). D_ξ is the discriminator. We parametrize $\mathcal{P}$ with a neural network G_η and the problem is reduced to estimating η (the dashed block). Once η is known, $G_\eta(u)$ is our approximation to the unknown random variable θ. When the functional form of $\mathcal{P}$ is known, we do not need a neural network and we optimize the parameters $\boldsymbol{\theta}$ of the functional form $\mathcal{P}$.

6. Repeat 1–4 to generate new w_i and u_i and update η according to the gradient ($\alpha_1 > 0$ is the step size)

$$\eta \leftarrow \eta - \alpha_2 \nabla_\eta L^F(\{F(w_i, G_\eta(z_i))\}_i). \tag{5}$$

Note we have abused the notation L^D, L^F for both random variables/processes and discrete samples. The discrete version can be viewed as the numerical approximation to the random variables/processes counterparts. The explicit expression is given in Section 2.3.

For the implementation, we use automatic differentiation [22] for computing Eqs. (4) and (5). The key is that the gradients must be back-propagated through both neural networks and physical or statistical models. For this purpose, we use ADCME [23] to both implement the numerical schemes and couple them with neural networks. The ADCME software is available at

https://github.com/kailaix/ADCME.jl

In the numerical examples, we consider two cases:

1. The functional forms of the distributions are known. In these cases, we calibrate the parameters of the given functional forms.
2. The functional forms of the distributions are unknown. In these cases, we use DNNs to approximate distributions with unknown functional forms.

We demonstrate that our algorithm is very effective in learning complex distributions for random variables within a partial differential equation system. We demonstrate numerically that the `L-BFGS-B` optimizer [24,25] performs equally well or significantly better in some cases compared to stochastic gradient descent optimizers. `L-BFGS-B` converges faster and is more stable than `ADAM` and `RMSProp`.

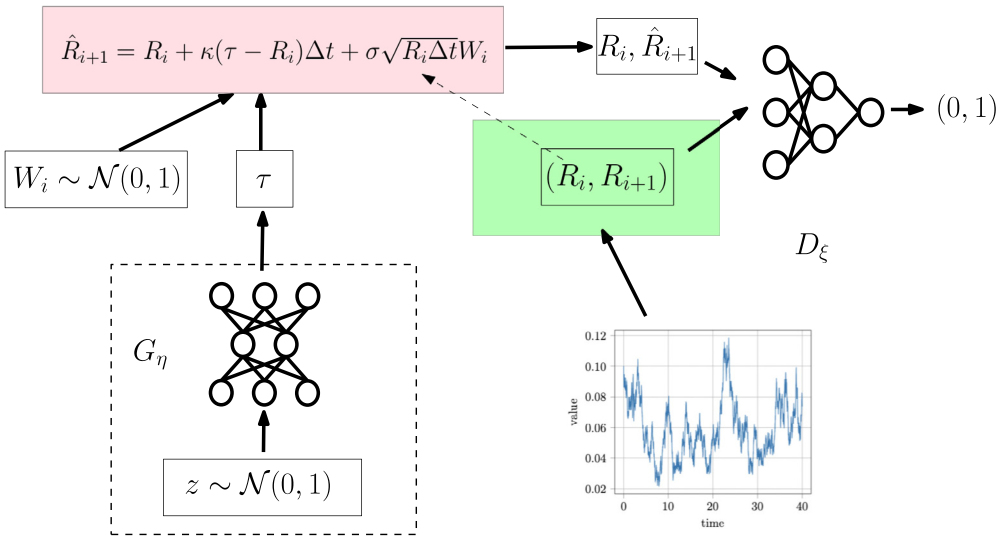
$\hat{R}_{i+1} = R_i + \kappa(\tau - R_i)\Delta t + \sigma\sqrt{R_i \Delta t} W_i$
$R_i, \hat{R}_{i+1}$
$(0,1)$
$W_i \sim \mathcal{N}(0,1)$
τ
(R_i, R_{i+1})
D_ξ
G_η
$z \sim \mathcal{N}(0,1)$
value
time

Fig. 2. AIM pipeline for Eq. (6). The outputs are obtained from simulated samples $\hat{R}_{i+1}$ from inputs R_i. D_ξ discriminates $\{(R_i, \hat{R}_{i+1})\}$ and $\{(R_i, R_{i+1})\}$. If τ is a parameter instead of a distribution, the neural network part is not needed.

2.2. Illustrative example

To illustrate the concept of adversarial inverse modeling, we consider the parameter calibration of the mean in the Cox, Ingersoll and Ross process (CIR process) [26]. The CIR process

$$dr_t = \kappa(\tau - r_t)dt + \sqrt{r_t}\sigma dW_t \tag{6}$$

is used for interest rate modeling [27], (κ, τ, σ) are model parameters, $\{W_t, t \geq 0\}$ is the standard Brownian motion, the interest rate r_t moves in the direction of its mean τ at speed κ, and $r_t\sigma^2$ is the diffusion function. Assume we are given a sample path $\mathbf{R} = \{R_1, R_2, \ldots, R_n\}$ with time interval Δt, and σ, κ are known. The task is to estimate the mean τ (see Fig. 2).

The idea of AIM is to match the distribution of simulated data and that of observed data. For this example, we define the observed data as $\{(R_i, R_{i+1})\}_{i=1}^n$. The numerical model corresponding to the continuous one Eq. (6) is the Euler Maruyama scheme

$$\hat{u}_i = F((x_i, W_i), \tau) = \left(x_i,\ \ x_i + \kappa(\tau - x_i)\Delta t + \sigma\sqrt{x_i \Delta t} W_i\right), \tag{7}$$

with stochastic input $W_i \sim \mathcal{N}(0,1)$ and x_i is randomly drawn from the sample path. A discriminative neural network $D_\xi : \mathbb{R}^2 \to (0,1)$ is used to measure the discrepancy between $\{u_i\}$ and $\{\hat{u}_i\}$. The goal of the neural network is to output 1 for u_i and 0 for $\hat{u}_i$, i.e.,

$$D_\xi(u_i) \approx 1, \qquad D_\xi(\hat{u}_i) \approx 0,$$

under this assumption. To this end, we can find ξ by minimizing

$$L^D(\{u_i\}, \{\hat{u}_i\}; \xi) = -\frac{1}{n}\sum_{i=1}^{n}\left(\log(D_\xi(u_i)) + \log(1 - D_\xi(\hat{u}_i))\right). \tag{8}$$

To combat the discriminator neural network D_ξ, one choice of the model loss function is (recall that $\hat{u}_i$ depends on τ)

$$L^F(\{\hat{u}_i\}) = -\frac{1}{n}\sum_{i=1}^{n}\log D_\xi(\hat{u}_i), \tag{9}$$

where we drive $D_\xi(\hat{u}_i)$ to 1 by minimizing L^F, and therefore we make the simulated outputs $\hat{u}_i$ seem more similar to the observed ones ($D_\xi(u_i) \approx 1$).

In the case where $\tau \sim \mathcal{P}$ is a nondegenerate distribution instead of a parameter, we parametrize $\mathcal{P}$ with the neural network G_η. The mathematical model for the stochastic process is converted to

$$\hat{u}_i = F((x_i, W_i), G_\eta(z)). \tag{10}$$

Here we use $z \sim \mathcal{N}(0, 1)$.

The optimization algorithm follows Algorithm 1, where we alternatively minimize Eqs. (8) and (9).

Algorithm 1 Optimization algorithm for AIM. The outer loop of the algorithm involves updating $\boldsymbol{\theta}$ and an inner loop, where the discriminator neural network is updated for k times. For `L-BFGS-B`, $k = 1$ is used. For Eq. (7), $\boldsymbol{\theta} = \tau$. For Eq. (10), $\boldsymbol{\theta} = \eta$ and we abuse the notation $F((x_i, W_i), \eta) = F((x_i, W_i), G_\eta(z))$.

```
1:  for $i = 1, 2, 3, \ldots$ do
2:      for $j = 1, 2, \ldots, k$ do
3:          Generate $n$ samples $\{w_1, w_2, \ldots, w_n\}$ from $\mathcal{Q}$
4:          Compute $\hat{u}_i \leftarrow F((x_i, w_i), \boldsymbol{\theta})$, $i = 1, 2, \ldots, n$
5:          Update the discriminator neural network parameters $\boldsymbol{\xi}$ using the gradients
                $\nabla_{\boldsymbol{\xi}} L^D(\{u_i\}, \{\hat{u}_i\}; \xi)$;
6:      end for
7:      Generate $n$ samples $\{w_1, w_2, \ldots, w_n\}$ from $\mathcal{Q}$
8:      Compute $\hat{u}_i \leftarrow F((x_i, w_i), \boldsymbol{\theta})$, $i = 1, 2, \ldots, n$
9:      Compute the gradients of $L^F$ using the adjoint state method (or automatic differentiation)
            $\mathbf{g} = \nabla_{\boldsymbol{\theta}} L^F(\{\hat{u}_i\})$;
10:     Update $\boldsymbol{\theta}$ with $\mathbf{g}$
11: end for
```

2.3. Choice of loss functions for AIM

The discriminator neural network D_ξ shares similar features as that in GANs. In this subsection, we propose three sets of loss functions, which all come from different GAN algorithms.

Recall that D_ξ is the discriminator neural network, u_i and $\hat{u}_i$ are observed and simulated data respectively. The major difference between the following GANs is their loss functions.

1. *Vanilla GAN* [28]. The output of D_ξ is a number in the range (0, 1), and the loss functions are
$$L^F(\{\hat{u}_i\}) = -\frac{1}{n}\sum_{i=1}^n \log D_\xi(\hat{u}_i),$$
$$L^D(\{u_i\}, \{\hat{u}_i\}; \xi) = -\frac{1}{n}\sum_{i=1}^n \left(\log(D_\xi(u_i)) + \log(1 - D_\xi(\hat{u}_i))\right).$$
At equilibrium, $L^F = \log 2$, $L^D = \log 4$.
2. *Wasserstein GAN* [29]. The last layer of the discriminator network is usually a linear layer, and the loss functions are
$$L^F(\{\hat{u}_i\}) = \frac{1}{n}\sum_{i=1}^n D_\xi(\hat{u}_i),$$
$$L^D(\{u_i\}, \{\hat{u}_i\}; \xi) = -\frac{1}{n}\sum_{i=1}^n D_\xi(u_i).$$
The discriminator neural network D_ξ is usually restricted to those whose norm of weights are within $[-c, c]$, $c > 0$. The norm of the weights in D_ξ are usually restricted to $[-c, c]$, $c > 0$, so that D_ξ belongs to the class of Lipschitz functions. In this case, the output of the neural network is not necessarily (0, 1). At equilibrium, $L^F = 0$, $L^D = 0$.

3. *Kullback–Leibler (KL) GAN* [21]. The output of D_ξ is a number in the range (0, 1), and the loss functions are

$$\begin{aligned} L^F(\{\hat{u}_i\}) &= \frac{1}{n}\sum_{i=1}^{n} \log \frac{1 - D_\xi(\hat{u}_i)}{D_\xi(\hat{u}_i)}, \\ L^D(\{u_i\}, \{\hat{u}_i\}; \xi) &= -\frac{1}{n}\sum_{i=1}^{n} \left(\log D_\xi(\hat{u}_i) + \log(1 - D_\xi(u_i))\right). \end{aligned} \tag{11}$$

At equilibrium, $L^F = 0$, $L^D = \log 4$.

Other variations of loss functions exist. A comprehensive comparison of the loss functions is out of scope but we refer readers to [21] for a detailed discussion.

The choice of optimizers is another concern for training in AIM. We attempt to compare the `L-BFGS-B` optimizer, which is extensively used in engineering, with other popular optimizers in machine learning. In our experiment, we mainly use three kinds of optimizers:

1. `ADAM` [30]. `ADAM` is an algorithm for first-order optimization of stochastic objective functions based on adaptive estimates of lower-order moments. We apply 5 updates to $\boldsymbol{\theta}$ and 1 update to D_ξ in each iteration.
2. `RMSProp` [31]. `RMSProp` divides the gradient by a running average of its recent magnitude. Usually it divides the learning rate by an exponentially decaying average of squared gradients. We apply 5 updates to $\boldsymbol{\theta}$ and 1 update to D_ξ in each iteration.
3. `L-BFGS-B` [32]. The limited Broyden–Fletcher–Goldfarb–Shanno algorithm is a powerful approach for finding a local minimum of a nonconvex objective function. The `L-BFGS-B` approximates the Hessian based on the most recent gradients. In our examples, `L-BFGS-B` is used for optimizing L^F and one `ADAM` update is applied to D_ξ in each iteration.

3. Convergence analysis

3.1. KL GAN produces maximum likelihood estimators

Thanks to the connection of AIM with GANs discussed in Section 1, many convergence analysis results for GANs are also applicable to AIM. From the perspective of divergence minimization [33], variants of GANs are proposed to minimize Kullback–Leibler distance (KL), the Jensen–Shannon distance, or the Wasserstein distance. For example, the vanilla GAN proposed in [28] is equivalent to minimizing the cross-entropy [34]. One particularly interesting case is minimizing the KL-divergence. It was shown [34] that it is equivalent to maximizing the log-likelihood function

$$\arg\min_\eta D_{KL}(\{u_i\}_{i=1}^n \parallel F(w, G_\eta(u))) = \arg\max_\eta \log \sum_{i=1}^{n} l_\eta(u_i), \tag{12}$$

where $l_\eta(u)$ is the log likelihood function and $\{u_i\}_{i=1}^n$ is the discrete distribution of the observations u. The optimal η is exactly the maximum likelihood estimator. [28] showed that using Eq. (11) is equivalent to minimizing Eq. (12) in expectation, under the assumption that the discriminator neural network is optimal.

It is known that under some regularity conditions on the family of distributions, the maximum likelihood estimator $\hat{\eta}_n$ is consistent, i.e., if η^* is the true parameter, we have $\hat{\eta}_n \to \eta^*$, $n \to \infty$ [35]. In addition, we have the asymptotic normality for maximum likelihood estimator [36]

$$\sqrt{n}(\hat{\eta}_n - \eta^*) \to \mathcal{N}\left(0, \frac{1}{I(\eta^*)}\right),$$

where $I(\eta)$ is the Fisher information

$$I(\eta) = -\mathbb{E}_u\left(\frac{\partial^2}{\partial \eta^2} \log l_\eta(u)\right).$$

Remark 1. The discussion above holds true if θ is subject to a Dirac delta distribution. In this case, the minimization problem becomes

$$\arg\min_{\theta} D_{KL}(\{u_i\}_{i=1}^n \parallel F(w,\theta)) = \arg\max_{\theta} \log \sum_{i=1}^{n} l_\theta(u_i),$$

and let the maximum likelihood estimator for n samples be $\hat{\theta}_n$, and the exact parameter is θ^*, then we have

$$\sqrt{n}(\hat{\theta}_n - \theta^*) \to \mathcal{N}\left(0, \frac{1}{I(\theta^*)}\right).$$

3.2. Theoretical analysis for the CIR example with KL divergence

In this section, we analyze the convergence of AIM for estimating θ and κ in Eq. (6). The analysis for other parameters is similar. The analysis also gives us some insights into when the mathematical problem is proposed and thus guides the design of algorithms.

In Section 3.1 we showed that if we use KL GAN, under the assumption that the discriminator neural network is optimal, κ, θ will converge to the maximum likelihood estimator in expectation. We assume that all those assumptions are satisfied, and therefore we can focus on analyzing the convergence of the maximum likelihood estimators for θ and κ. Note the choice of KL GAN is merely for analysis purposes and reconstructing the maximum likelihood estimator from AIM is not the final goal. But since the maximum likelihood estimator is well studied the analysis based on this particular choice helps us understand the theoretical aspect of AIM better.

3.2.1. Convergence for the τ estimator

We revisit the CIR example in Section 2.2. Recall in the CIR example, $u = F(w, \tau)$, samples of the known stochastic process w have the form $W_i \sim \mathcal{N}(0, 1)$ and the outputs (x_i, u_i) are two consecutive samples with time interval Δt. x_i and u_i are related by

$$u_i = x_i + \kappa(\tau - x_i)\Delta t + \sigma\sqrt{x_i \Delta t} W_i.$$

The probability density function of u_i given the observation x_i is given by

$$p(u|x) = \frac{1}{\sqrt{2\pi\sigma^2 x \Delta t}} \exp\left(-\frac{1}{2}\left(\frac{u - x - \kappa(\tau - x)\Delta t}{\sigma\sqrt{x\Delta t}}\right)^2\right).$$

The log likelihood function for the joint distribution (x, u) is

$$l_\tau(x, u) = -\frac{1}{2}\left(\frac{u - x - \kappa(\tau - x)\Delta t}{\sigma\sqrt{x\Delta t}}\right)^2 + \log f(x) - \log\left(\sigma\sqrt{2\pi x \Delta t}\right).$$

Here $f(x)$ is the probability density function of the random variable x. Therefore, assume we have observations (x_i, u_i), $i = 1, 2, \ldots, n$, which are consecutive samples with time interval Δt, the empirical log likelihood function is

$$\sum_{i=1}^{n} l_\tau(x_i, u_i) = -\frac{1}{2}\sum_{i=1}^{n}\left(\frac{u_i - x_i - \kappa(\tau - x_i)\Delta t}{\sigma\sqrt{x_i\Delta t}}\right)^2 + \sum_{i=1}^{n}\log f(x_i) - \sum_{i=1}^{n}\log\left(\sigma\sqrt{2\pi x_i \Delta t}\right).$$

The maximum likelihood estimator can be computed using

$$\sum_{i=1}^{n} l'_\tau(x_i, u_i) = 0 \Rightarrow \hat{\tau}_n = \frac{\frac{1}{n}\sum_{i=1}^{n}\frac{u_i}{x_i} + \kappa\Delta t - 1}{\left(\frac{1}{n}\sum_{i=1}^{n}\frac{1}{x_i}\right)\kappa\Delta t}. \tag{13}$$

Following this discussion, if we use KL divergence as the discrepancy measure of the real sample-path and generated ones, $\hat{\tau}$ at the Nash equilibrium will converge to $\hat{\tau}_n$ under mild assumptions. Let τ^* be the exact solution, we prove that as $\Delta t \to 0$, $n \to \infty$, $\hat{\tau}_n \to \tau^*$. We always assume convergence in the distribution for probability convergence. The proofs for the theorems are postponed to Appendices B and C.

Theorem 1. *Assume that* $\mathbb{E}\left(\frac{1}{x}\right) = X_{-1} \in (0, \infty)$, *for sufficiently small* $|\Delta t|$, *we have*

$$\hat{\tau}_n \to \tau^* + \mathcal{O}(\Delta t), \qquad n \to \infty. \tag{14}$$

In addition, the corresponding Fisher information is equal to

$$I(\tau) = \frac{\kappa^2 \Delta t X_{-1}}{\sigma^2},$$

and consequently

$$\sqrt{n}(\hat{\tau}_n - \tau^*) \to \mathcal{N}\left(0, \frac{\kappa^2 \Delta t}{\sigma^2 X_{-1}}\right).$$

Sketch of the proof. We apply the law of large numbers to both the numerator and denominator of Eq. (13).

Remark 2. The constraint $\mathbb{E}\left(\frac{1}{x}\right) = X_{-1} < \infty$ is satisfied asymptotically. Assume $2\kappa\tau > \sigma^2$, and n is large enough so x_i is subject to the asymptotic distribution of the CIR process, which is a gamma distribution with density function

$$h(r) = \frac{w^\nu}{\Gamma(\nu)} r^{\nu-1} e^{-wr}, \quad w = \frac{2\kappa}{\sigma^2}, \quad \nu = \frac{2\kappa\tau^*}{\sigma^2}. \tag{15}$$

Thus we have

$$\mathbb{E}\left(\frac{1}{x}\right) = \int_0^\infty \frac{1}{r} h(r) dr = \frac{1}{\tau^*} < \infty. \tag{16}$$

If x is sampled from the asymptotic distribution, the Fisher information is

$$I(\tau) = \frac{\kappa^2 \Delta t}{\sigma^2 \tau^*}.$$

Therefore, we have

$$\sqrt{n}(\hat{\tau}_n - \tau^*) \to \mathcal{N}\left(0, \frac{\sigma^2 \tau^*}{\kappa^2 \Delta t}\right). \tag{17}$$

The conditions in Eqs. (14) and (17) indicate that for $\hat{\tau}_n \to \tau^*$ as $\Delta t \to 0$, $n \to \infty$, the following condition is sufficient

$$n\Delta t \to \infty, \ \ \Delta t \to 0. \tag{18}$$

For example, we let $\Delta t = \frac{1}{\sqrt{n}}$, i.e., $n = \frac{1}{\Delta t^2}$, then the condition Eq. (18) is satisfied.

3.2.2. Convergence for the κ *estimator*

We assume that τ and σ are known and we want to estimate κ, whose true value is κ^*.

Theorem 2. *Assume that* $\mathbb{E}(x) = X_0 \in (0, \infty)$, $\mathbb{E}\left(\frac{1}{x}\right) = X_{-1} < \infty$, *for sufficiently small* $|\Delta t|$, *we have*

$$\hat{\kappa}_n \to \kappa^* + \mathcal{O}(\Delta t), \ \ n \to \infty.$$

In addition, the corresponding Fisher information is equal to

$$I(\kappa) = \frac{\Delta t}{\sigma^2}(\tau^2 X_{-1} - 2\tau + X_0),$$

and consequently

$$\sqrt{n}(\hat{\kappa}_n - \kappa^*) \to \mathcal{N}\left(0, \frac{\sigma^2}{\Delta t(\tau^2 X_{-1} - 2\tau + X_0)}\right), \qquad n \to \infty. \tag{19}$$

Sketch of the proof. To derive the Fisher information, compute the derivatives of the likelihood function. In the case where n is very large, $\frac{1}{n}\sum_{i=1}^{n}\frac{1}{x_i}$ and $\frac{1}{n}\sum_{i=1}^{n}x_i$ are approximately subject to the asymptotic distribution. Therefore, we have from Eq. (16)

$$X_{-1}\approx\frac{1}{\tau},\qquad X_0\approx\tau.$$

Consequently

$$I(\kappa)\approx 0,$$

which indicates that the estimator has infinite variance. However, as long as we resample from the observations such that the new training data satisfies $\tau^2X_{-1}-2\tau+X_0\neq 0$, we will obtain convergence result for κ estimator. One such technique is given in Section 4.4.

Remark 3. Given a parameter τ, let the family of distribution P_τ and P_{τ^*} be the true data distribution. The relationship between KL divergence and Fisher information can be described as follows for sufficiently small $|\tau-\tau^*|$ under mild assumptions

$$D_{KL}(P_{\tau^*}\parallel P_\tau)\approx\frac{1}{2}(\tau-\tau^*)^TI(\tau^*)(\tau-\tau^*).$$

Thus if $I(\tau^*)\approx 0$, we can see that $D_{KL}(P_{\tau^*}\parallel P_\tau)$ will have vanishing gradients and Hessians at $\tau=\tau^*$. This may lead to difficulty during optimization.

4. Numerical benchmarks

In this section, we present four numerical examples. The mathematical models for those problems can all be formulated in the form of $u=F(w,\theta)$ and therefore AIM is applicable for those cases. The extensive numerical results also demonstrate the generality of the method.

4.1. Estimation and uncertainty quantification of hidden parameters in PDEs

In this example, the unknown parameter θ consists of an unknown distribution and an unknown parameter. The unknown distribution is parametrized by a neural network. We simultaneously estimate both the distribution and the parameter.

Consider a Poisson equation with Dirichlet boundary condition

$$\begin{cases}-\nabla\cdot(a(x)\nabla u(x))=1, & x\in(0,1),\\ u(0)=u(1)=0, & \text{otherwise},\end{cases}\tag{20}$$

where

$$a(x)=1-0.9\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right).$$

Assume that μ and σ are both unknown, and there is model form uncertainty in μ, i.e., $\mu\sim\mathcal{P}$ for some unknown probability distribution $\mathcal{P}$. Our goal is that given many observations $\mathbf{u}_i$, $i=1,2,\ldots,N$, where each $\mathbf{u}_i\in\mathbb{R}^n$ is an observation of $[u(h)\ u(2h)\ \cdots\ u(nh)]$, $h=\frac{1}{n+1}$. The goal is to determine σ and $\mathcal{P}$, and the corresponding mathematical problem can be written as

$$\begin{aligned}&\min_{(\mathcal{P},\sigma)}\ \mathbb{E}_{\hat{u}\sim p_{\text{data}}}\text{Discrepancy}(u,\hat{u}),\\ &\quad\text{s.t. }\mu\sim\mathcal{P},\\ &\qquad a(x)=1-0.9\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right),\\ &\qquad-\nabla\cdot(a(x)\nabla u(x))=1,\quad x\in(0,1),\\ &\qquad u(0)=u(1)=0,\end{aligned}$$

where p_{data} is the distribution of the observed data.

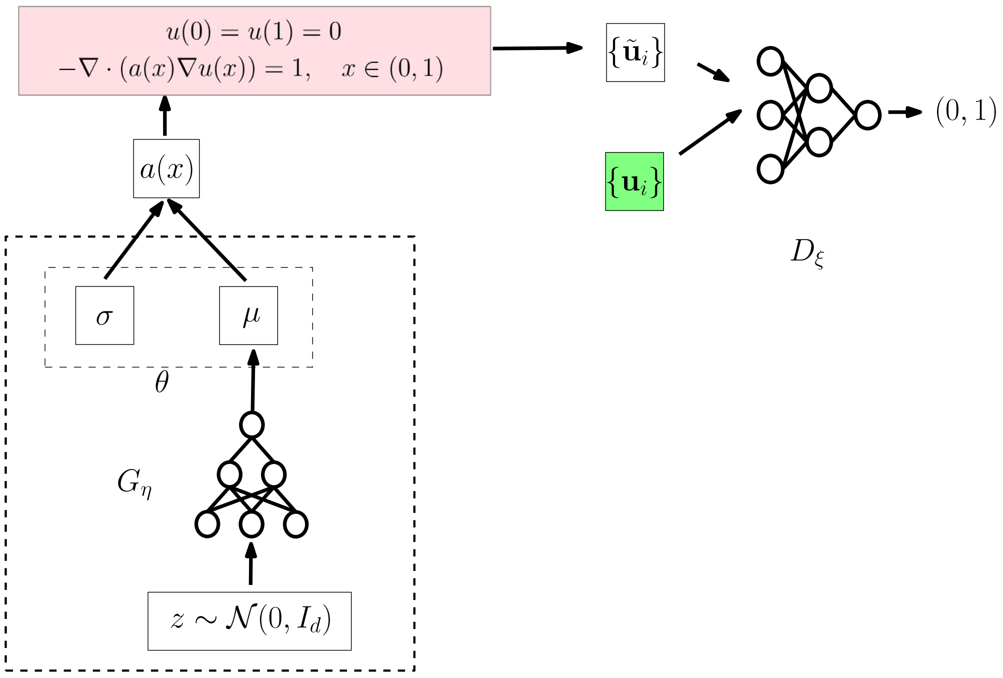
$u(0) = u(1) = 0$
$-\nabla \cdot (a(x)\nabla u(x)) = 1, \quad x \in (0,1)$
$\{\tilde{\mathbf{u}}_i\}$
$\{\mathbf{u}_i\}$
$(0,1)$
D_ξ
$a(x)$
σ
μ
θ
G_η
$z \sim \mathcal{N}(0, I_d)$

Fig. 3. The AIM computation pipeline for Eq. (21). In this case, the unknown distribution $\theta = (\sigma, \mu)$ consists of two components: σ is a deterministic but unknown number and μ is a distribution with an unknown functional form. We use a generative neural network G_η to generate samples for μ.

Traditionally, this can be done under the Bayesian framework where a priori information is imposed on $\mathcal{P}$ and the transformation of probability in the forward model is calculated. However, this approach may be infeasible for problems where the probabilistic relationship between data and parameter is analytically intractable, i.e., expressing $p(u(h), u(2h), \ldots, u(nh)|\ \mu, \sigma)$ in a closed-form. AIM is not limited by the availability of the analytical probabilistic relationship; indeed, all it needs is a model where forward computation and gradient back-propagation can be numerically performed.

To parametrize $\mathcal{P}$, we consider a neural network $N : \mathbb{R}^d \to \mathbb{R}$, $d = 10$; the inputs of neural network are samples from the uniform distribution $\mathcal{U}([-1, 1]^d)$ and the output is a generated sample of μ. To discretize Eq. (20), we consider the central difference scheme

$$\begin{aligned} -a_{i-\frac{1}{2}}u_{i-1} + (a_{i-\frac{1}{2}} + a_{i+\frac{1}{2}})u_i - a_{i+\frac{1}{2}}u_{i+1} = h^2, \qquad i = 1, 2, \ldots, n, \\ u_0 = u_{n+1} = 0, \end{aligned} \tag{21}$$

where $a_{j\pm\frac{1}{2}} = a(h(j \pm \frac{1}{2}))$, which depends on μ and σ. Eq. (21) leads to a tridiagonal system and can be solved using the Thomas algorithm [37]. The computation pipeline is shown in Fig. 3. The loss functions for F_θ and the discriminator neural network are chosen to be Wasserstein GAN losses. We use `RMSProp` with learning rate 10^{-4} and batch size 32. The true $\sigma^* = 0.1$ and the true distribution for μ is

$$\mu^* \sim \mathcal{N}(0.3, 0.1).$$

Fig. 4 shows the results of AIM. In the first plot, we show the model loss and discriminator loss. We see that both the discriminator loss and the model loss converge to 0, indicating that the optimization reaches the equilibrium. In the second plot, we see that the estimated σ converges to σ^* after around 1000 iterations. The values then oscillate around 0.1 due to the intrinsically adversarial optimization. The last plot shows the true distribution of μ^* and generated distribution for μ after iteration 38,000. We see that the generated distribution matches the true distribution reasonably well.

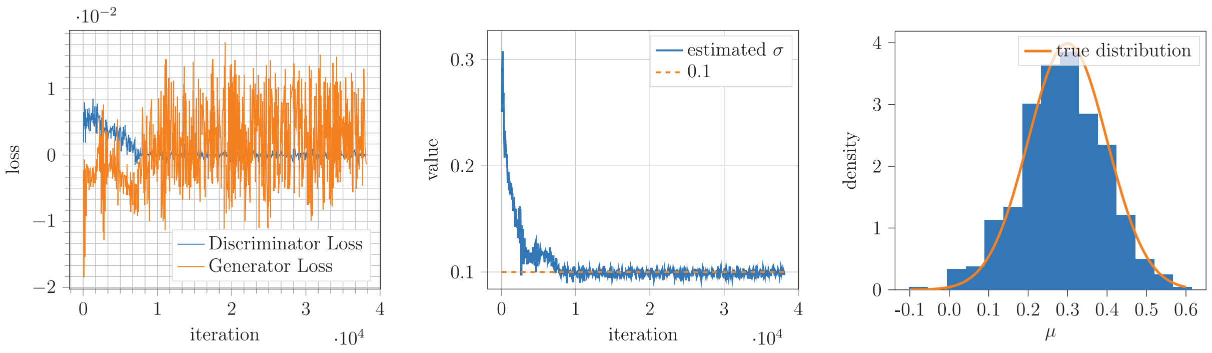

Fig. 4. AIM results for Eq. (21). In the first plot, we show the model loss and discriminator loss. In the second plot, we see that the estimated σ converges to σ^* after around 1000 iterations. The last plot shows the true distribution of μ^* and generated distribution for μ after iteration 38,000.

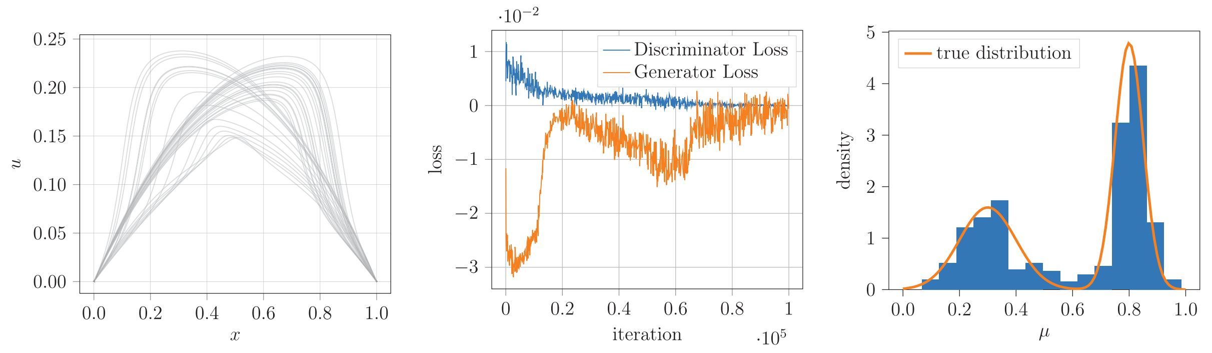

Fig. 5. AIM results for Eqs. (21) with (22). The first plot shows generated samples of $u(x)$. The second plot shows the loss functions. The last plot shows the generated distribution at iteration 100,000 together with the true distribution.

4.2. Uncertainty quantification with multimodal distribution

We consider a variant of Section 4.1. This example demonstrates that AIM can be used to learn a non-Gaussian distribution.

Different from the last section, we assume that σ is known. Besides, μ is subject to a nontrivial distribution, e.g., Gaussian mixture distribution,

$$p = 0.4\mathcal{N}(0.3, 0.1) + 0.6\mathcal{N}(0.8, 0.05). \tag{22}$$

The generated samples are shown in the first plot in Fig. 5. AIM is carried out with the same setting as in Section 4.1. We show the loss functions and the generated distribution in Fig. 5. We can see that the loss functions for both the model and the discriminator network converge to 0. The generated distribution also captures the multimodality of the distribution correctly.

Other distributions for μ are also tested with the same neural network architecture and optimizer. The results in Fig. 6 demonstrate that the proposed method is capable of learning many different unknown distributions in the PDE system. The test distributions include

1. the Exponential distribution with a rate 1;
2. the F distribution with degrees of freedom $(5, 2)$;
3. the Arcsine distribution in $[0, 1]$;
4. the Beta distribution with shape parameters $(1, 3)$;
5. the Cauchy distribution with the location parameter $x_0 = 0$ and the scale parameter $b = 0.5$;
6. the raised Cosine distribution with parameters $\mu = 0.5$, $s = 0.5$.

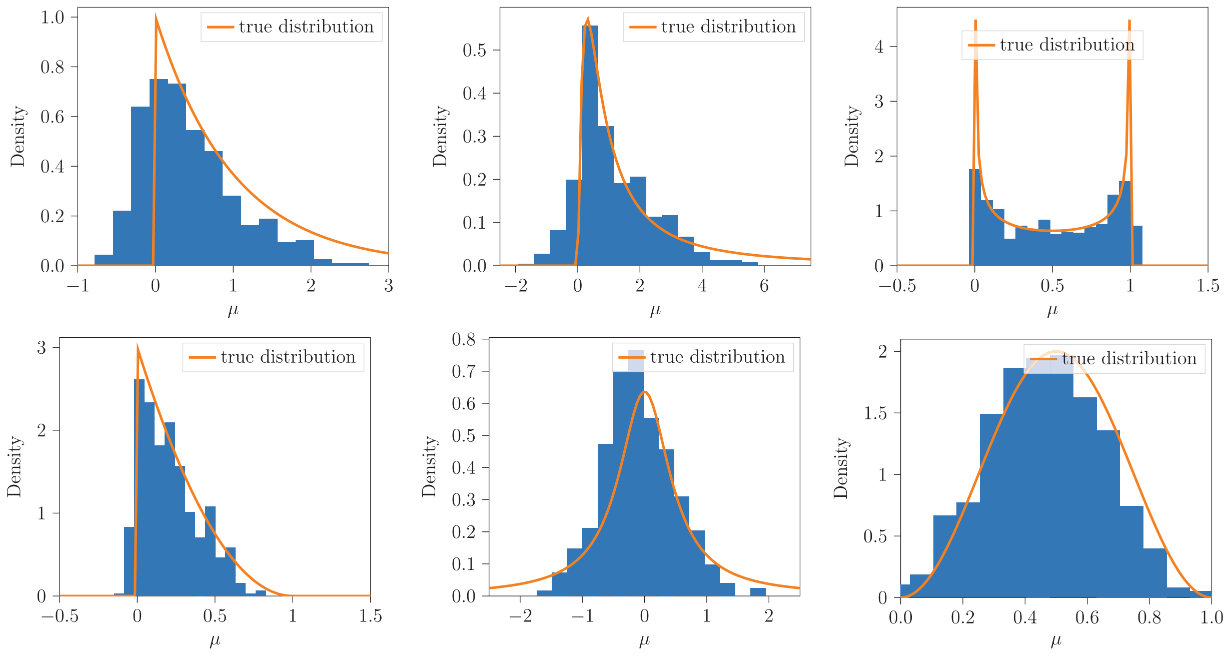

Fig. 6. AIM results for estimating the probability distribution of μ, where μ is subject to a different distribution for each case. The first row is (left to right) the Exponential distribution, the F distribution, and the Arcsine distribution; the second row is (left to right) the Beta distribution, the Cauchy distribution, and the raised Cosine distribution.

4.3. Two dimensional case

We also consider 2D distributions. In this case, we assume that μ, σ are both random variables and the joint distribution of (μ, σ) is unknown. We increase the number of the output layer neurons from 1 to 2 for generating σ in the previous case and perform adversarial inverse modeling. For generating synthetic data, we draw (μ, σ) from the following random distributions:

1. a 2D Gaussian distribution with mean (0.3, 1.0) and covariance matrix
$$\begin{pmatrix} 0.1 & -0.05 \\ -0.05 & 0.1 \end{pmatrix},$$
2. a Dirichlet distribution with parameters $\boldsymbol{\alpha} = (1.0, 1.0, 1.0)$,
3. a Dirichlet distribution with parameters $\boldsymbol{\alpha} = (1.0, 2.0, 3.0)$.

Because in the model σ appears as σ^2 in the coefficient function $a(x)$, we consider estimating the distribution of $(\mu, |\sigma|)$ instead of (μ, σ). Figs. 7 to 9 show the results at multiple iterations of the minimization.

4.4. Parameter calibration for CIR processes from sample paths

In this section, we consider the CIR process example considered in Sections 2.2 and 3.2 and numerically demonstrate the analysis in Section 3. We also compare different optimizers and conclude that `L-BFGS-B` converges the fastest and most stably.

For better accuracy, we use the weighted Milstein scheme [38] for the simulation Eq. (6)

$$u = \frac{x + \kappa(\tau - \alpha x)\Delta t + \sigma\sqrt{x}\sqrt{\Delta t}W + \frac{1}{4}\sigma^2\Delta t(W^2 - 1)}{1 + (1-\alpha)\kappa\Delta t}, \tag{23}$$

where $\alpha \in [0, 1]$ is the weight, x is the sample at last time step. In the numerical example, we let $\alpha = 0.5$, $\Delta t = 0.01$, $\sigma = 0.08$, $\kappa = 0.5$, and the exact $\tau^* = 0.06$. The length of the sample path is 4000. Fig. D.16 shows a

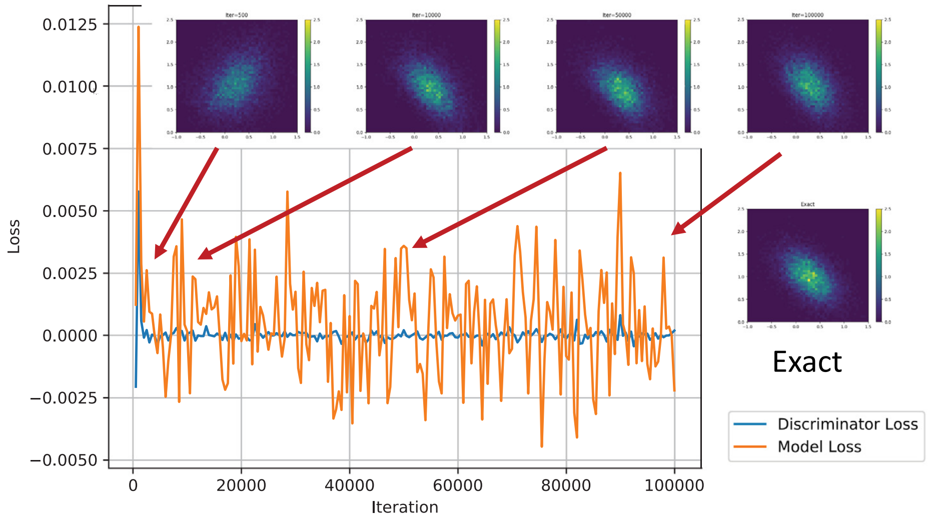

Fig. 7. Convergence of the $(\mu, |\sigma|)$ distribution using AIM; the PDF of (μ, σ) is a **2D Gaussian** distribution. The figure shows the loss function vs. iteration. The plots on the top show the $(\mu, |\sigma|)$ distribution for various iterations. The exact $(\mu, |\sigma|)$ distribution is shown to the right.

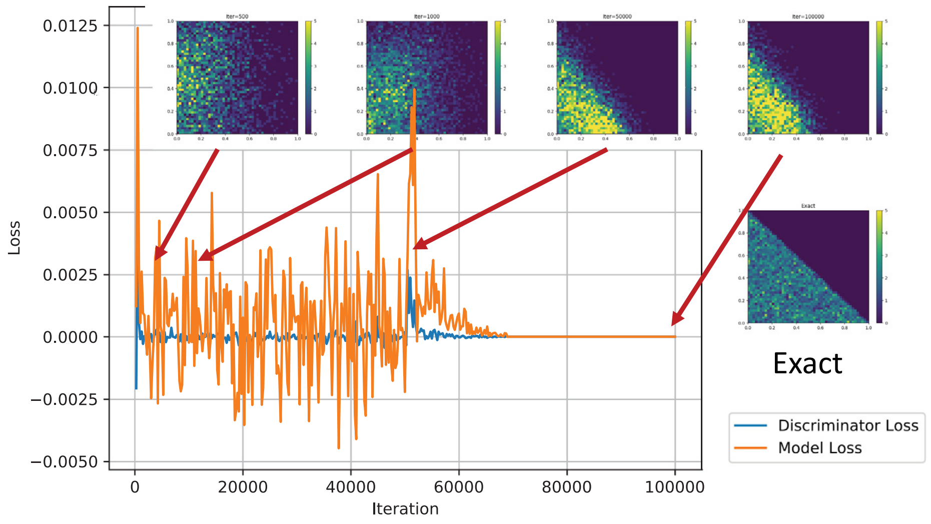

Fig. 8. Convergence of the $(\mu, |\sigma|)$ distribution using AIM; the PDF of (μ, σ) is a **Dirichlet** distribution with parameters $\boldsymbol{\alpha} = (1.0, 1.0, 1.0)$.

realization of the sample path. The KL GAN loss functions and three optimizers (`ADAM`, `RMSProp` and `L-BFGS-B`) with a full batch size are used.

Optimizers. Fig. 10 shows the result of AIM. The first two plots show the generator and discriminator losses, and, as we can see, they converge to 0 and $\log 4$ respectively. The third plot shows the convergence profile for τ. It is interesting to see the performance of the different optimizers. With a strong optimizer for the generator (`L-BFGS-B`), the convergence profile is much smoother than the others. `RMSProp` tends to produce kinks and oscillates around the true value $\tau^* = 0.06$ like `ADAM`.

Convergence. Next, we discuss the convergence of κ. As indicated in Theorem 2, the convergence of κ can be very slow if $X_0 = \tau$ and $X_{-1} = \frac{1}{\tau}$. Unfortunately, the equalities hold if (R_k, R_{k+1}) are sampled from a sufficient enough

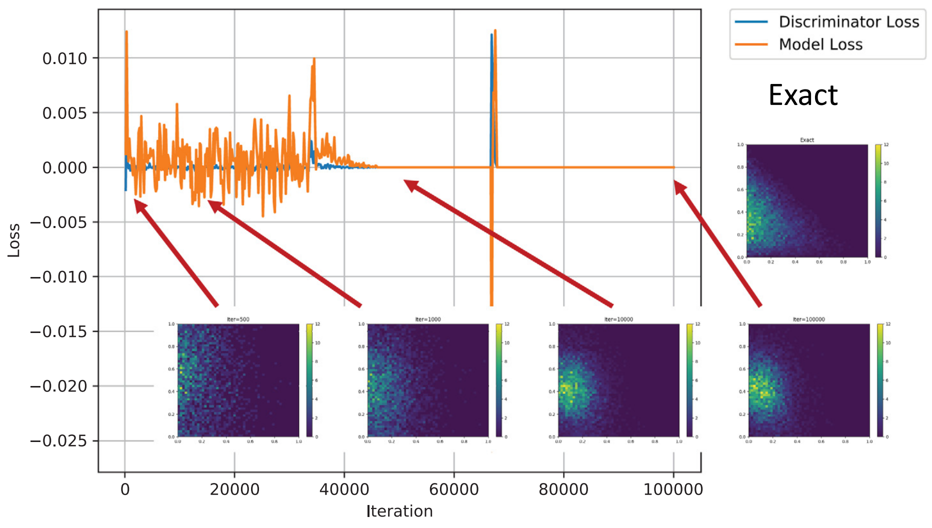

Fig. 9. Convergence of the $(\mu, |\sigma|)$ distribution using AIM; the PDF of (μ, σ) is a **Dirichlet** distribution with parameters $\boldsymbol{\alpha} = (1.0, 2.0, 3.0)$.

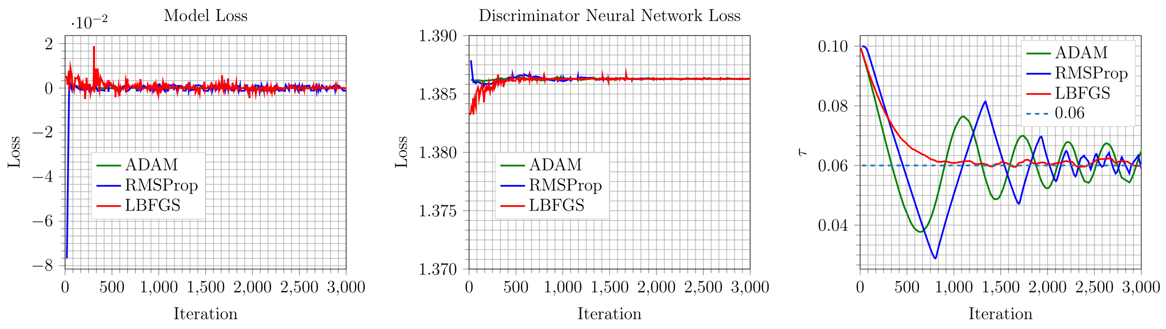

Fig. 10. AIM results for Eq. (23). The first two plots show the model loss and discriminator loss respectively. The third plot shows the convergence of the hidden parameter τ. In the last plot to the right, `L-BFGS-B` converges much faster than `ADAM` and `RMSProp`.

sample path. This is demonstrated in Fig. 11. We approximate the KL divergence with discrete KL divergence

$$L_\tau = \sum_i P_{\tau^*,i} \log\left(\frac{P_{\tau^*,i}}{P_{\tau,i}}\right), \quad L_\kappa = \sum_i P_{\kappa^*,i} \log\left(\frac{P_{\kappa^*,i}}{P_{\kappa,i}}\right),$$

where τ, κ are the parameters, τ^*, κ^* are their true values, and $P_{\tau,i}$ is the discrete density function. In the first plot, we let $\tau = 0.06$, $\sigma = 0.08$ and vary κ. The true value is $\kappa^* = 0.5$. In the second plot, we fix $\kappa = 0.5$, $\sigma = 0.08$ and vary τ, whose true value is $\tau^* = 0.06$. We carry out the simulation Eq. (23) from initial location $R_0 = 0.05$, $\Delta t = 0.001$ and path length 100,000. The result shows that near the true value, the L_κ curve is very flat, indicating small second order derivatives. Compared to L_τ, L_κ is much more oscillatory.

A possible fix to the problem is to modify the distribution R_k. In the left panel in Fig. 12, we artificially sample R_k from $\mathcal{U}(0.001, 0.03)$. We use $\Delta t = 0.001$, `RMSProp` with learning rate 10^{-3} for both κ and D_ξ. In each step, D_ξ is updated five times while F_θ is updated only once. We can see that the estimated κ oscillates around $\kappa^* = 0.5$ but the variance is still large. In Fig. 12-right, we further consider reducing the variance by increasing Δt. The observed variance reduction is consistent with Eq. (17).

Robustness. To assess the robustness of AIM, we add Gaussian noises to the sample data

$$\tilde{R}_i = R_i + v_i, \quad v_i \sim \mathcal{N}(0, \sigma_{\text{noise}}^2).$$

Here $\sigma_{\text{noise}} \geq 0$ is the standard deviation of the added noise. We let $\alpha = 0.5$, $\Delta t = 0.01$, $\sigma = 0.08$, $\kappa = 0.5$, and the exact $\tau^* = 0.06$. The initial guess for τ is 1.0. We use `RMSProp` to minimize both the model loss and discriminator

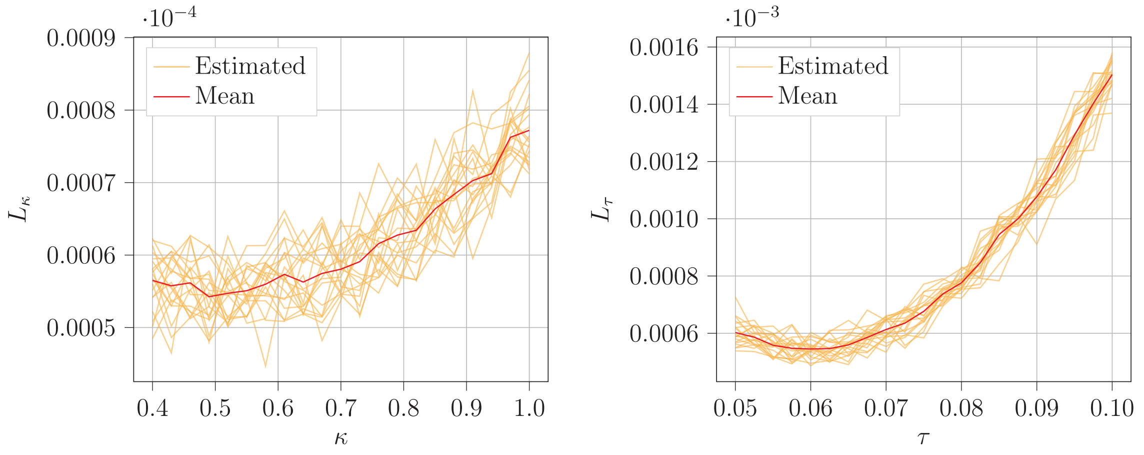

Fig. 11. Discrete KL divergence as a function of κ and τ. Each plot has 10 realizations of sample paths. We see that for the plot where κ is varied, the profile is much more oscillatory and the landscape at $\kappa^* = 0.5$ is nearly flat.

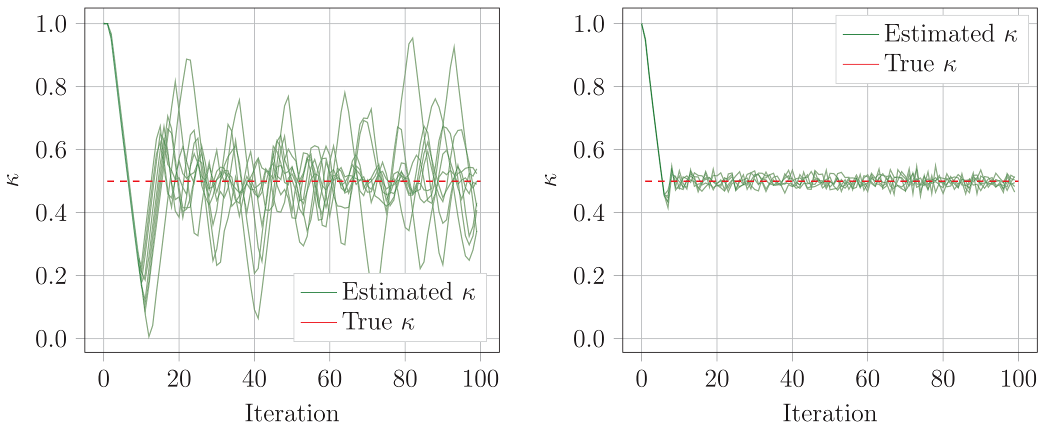

Fig. 12. Convergence of κ to the true value κ^*. In the right panel, we sample with interval $\Delta t = 0.1$ instead of $\Delta t = 0.001$. The variance of the κ estimation is reduced by about 100.

loss, with a learning rate 10^{-4}. In each iteration, we perform 5 steps of `RMSProp` for minimizing the discriminator loss while 1 step for the model loss. The result is shown in Fig. 13. We can see that AIM is robust to noises with moderate standard deviations ($\sigma_{\text{noise}} \leq 0.005$). This demonstrates the robustness of our AIM algorithm.

4.5. Volatility inference from option prices: Direct estimation from Monte Carlo simulation

The final example is an application of AIM to estimating volatility from European call option prices. In this case, σ is the unknown volatility parameter.

A European call option is a contract giving the holder the right to buy the underlying asset for a fixed strike price K at expiry time T. Let S_t be the price of the underlying such as the stock price at time t, then the payoff of a European call is a random variable [39]

$$P = \begin{cases} S_T - K, & \text{if } S_T > K, \\ 0, & \text{otherwise }. \end{cases}$$

The stock price can be modeled as a stochastic process such as geometric Brownian motion

$$dS_t = rS_t dt + \sigma S_t dW_t, \qquad S_0 = s, \tag{24}$$

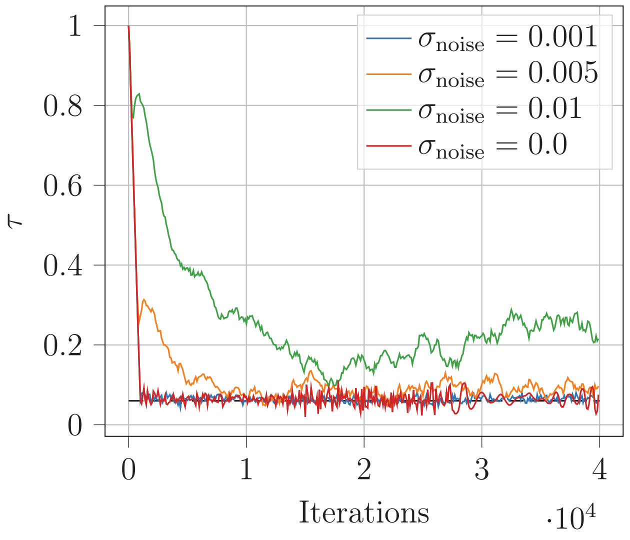

Fig. 13. Convergence of τ using data with different standard deviations of noises. We can see that AIM is robust to noises with moderate standard deviations ($\sigma_{\text{noise}} \le 0.005$). The exact $\tau^* = 0.06$ is shown with a dashed black line.

where W_t is a Gaussian random variable with zero mean, σ is the volatility, μ is the expected return, and s is the spot price of the stock.

The task is to estimate σ from many observations of option prices P_i, $i = 1, 2, \ldots, 100$. The forward model is composed of a Monte Carlo simulation for

$$S_T = s \exp\left[\left(r - \frac{\sigma^2}{2}\right)T + (\sigma\sqrt{T})W\right], \tag{25}$$

where W sampled from a standard normal distribution. Fig. D.17 shows samples for $s = 100$, $K = 100$, $r = 0.05$, $\sigma = 0.2$, and $T = 1$. The mathematical optimization problem can be formulated as

$$\begin{aligned} &\min_{\sigma} \mathbb{E}_{\hat{P}\sim p_{\text{data}}} \text{Discrepancy}(P, \hat{P}), \\ &\text{s.t. } dS_t = rS_t dt + \sigma S_t dW_t, \quad S_0 = s, \\ &\quad P = \max(S_T - K, 0), \end{aligned}$$

where p_{data} is the observed distribution of the option prices.

For AIM, we use the loss functions from the vanilla GAN. `RMSProp` with learning rate 10^{-4} and full batch size is used. The computation pipeline is shown in Fig. 14. The result is shown in Fig. 15. The left plot shows the convergence of the model loss and discriminator loss. We can see that the values converge to theoretical optimal values. The right plot shows the convergence of the estimated volatility.

5. Conclusion

We have demonstrated the viability of adversarial inverse modeling for solving inverse modeling problems of the type

$$F : (w, \theta) \mapsto u(\cdot, \cdot),$$

where w is a known stochastic process, θ is an unknown random variable, and the output u is also a random variable or process. The key ingredients for applying AIM are:

- Approximating the unknown distribution of θ by a neural network, $\hat{\theta} = G_\eta(\theta)$.
- Using a discriminator neural network D_ξ to measure the discrepancy between the actual random process x and estimated random process $\hat{x} = F(w, \hat{\theta})$.

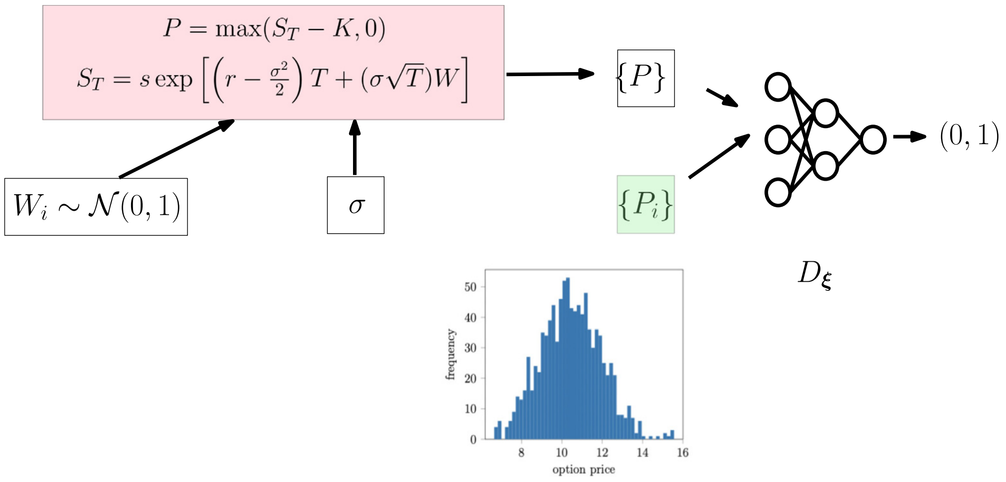
$P = \max(S_T - K, 0)$
$S_T = s\exp\left[\left(r - \frac{\sigma^2}{2}\right)T + (\sigma\sqrt{T})W\right]$
$\{P\}$
$\{P_i\}$
$(0, 1)$
$W_i \sim \mathcal{N}(0, 1)$
σ
D_ξ
frequency
option price

Fig. 14. AIM computation pipeline for the option price example. In this example, the unknown σ is a deterministic parameter. However, because W_i is stochastic, the output of the model is stochastic.

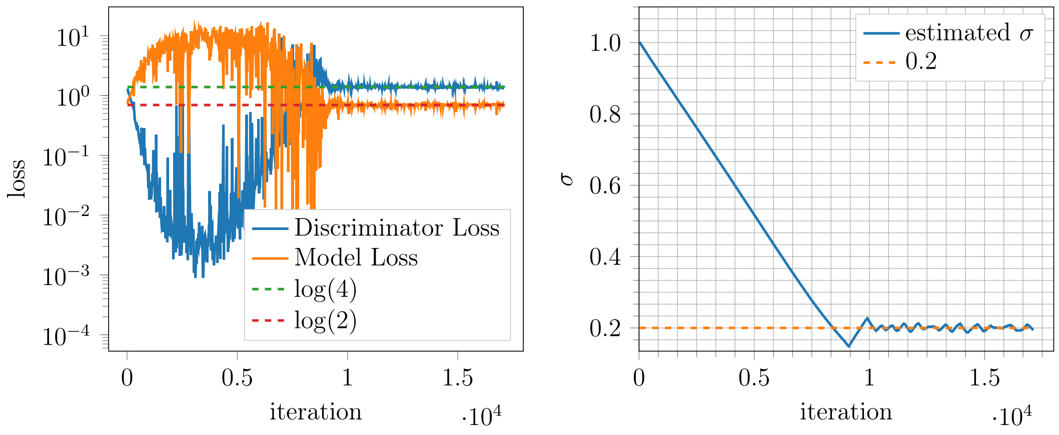

Fig. 15. AIM results for the option pricing example. The first plot shows the loss function while the second shows the convergence of the hidden parameter σ. The blue line for the estimated σ converges to the exact solution equal to 0.2. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

- Choosing a proper loss function for L^D and L^F.
- Adversarially training $\hat{\theta}$ and ξ. The gradients $\frac{\partial L^D}{\partial \xi}$ and $\frac{\partial L^F}{\partial \eta}$ can be computed with automatic differentiation.

We demonstrate numerically that the approach is able to learn underlying parameters and recover complex unknown distributions. Different optimizers were benchmarked and the results showed the promising behavior of full-batch optimizers such a `L-BFGS-B` for engineering applications.

Despite many strengths, AIM has some limitations. Firstly, due to the adversarial training approach in nature, the convergence of AIM may be unstable, as indicated by the oscillation of the loss functions around the equilibrium values. Although we have shown that using the `L-BFGS-B` optimizer improves the stability compared to other first-order optimizers, the applicability of the optimizer requires further investigation. Note that many other techniques, such as multiple random projections [40], regularization [41], and L_p-norm normalization [42], have been proposed to stabilize the training of GANs, and those techniques can also be used to improve the convergence of AIM. Secondly, the choice of neural network architectures is important but unexplored in the current work. The neural network architecture affects not only the training efficiency but also the accuracy of the approximation. Specifically, in the context of approximating random variables, it is interesting to investigate the approximation capability of the neural networks for random variables with different landscapes of probability density functions. Finally, the stopping criterion also forms an important topic for adversarial training. AIM does not yet provide a stopping criterion for evaluating the convergence *a priori*. However, these limitations do not prevent us from leveraging the power of neural networks and adversarial training for solving inverse problems in stochastic models, which are ubiquitous in physical sciences, statistical modeling, and mathematical finance. AIM provides an alternative approach to estimate unknown parameters and distributions when analytical tools are not available, or traditional methods are less accurate or too expensive.

In the context of inverse problems, AIM provides a powerful solution that pipelines neural networks, adversarial training, and automatic differentiation to learn parameters and distributions in a stochastic model. The synergy of deep learning techniques and inverse modeling has the potential to solve many challenging inverse problems where traditional methods are either inapplicable or intractable.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work is supported by the Applied Mathematics Program within the Department of Energy (DOE) Office of Advanced Scientific Computing Research (ASCR), USA, through the Collaboratory on Mathematics and Physics-Informed Learning Machines for Multiscale and Multiphysics Problems Research Center (DE-SC0019453).

Appendix A. Probability metrics for AIM

Here we discuss a few probability metrics that are related to AIM; we refer the readers to [43] for more details on this topic. Note that these metrics are not mutually exclusive.

Information-theoretical Metrics [44] provide metrics for probabilities from an information-theory point of view. The idea is to find the shortest description of the data. Assume that the base of the logarithm is 2. The entropy $H(P) = \int_{\mathcal{X}} P(x)\log P(dx)$ describes the optimal average asymptotical code length for a single distribution P according to asymptotic equipartition property [45]. For two distributions P and Q for the same measurable space $\mathcal{X}$, assume that P is the true distribution and Q is the proposed distribution for approximating P, the relative entropy, also called Kullback–Leibler divergence, $D(P \parallel Q) = \int_{\mathcal{X}} P(x)\log \frac{P(dx)}{Q(dx)}$ satisfies [46]

$$H(P) + D(P \parallel Q) \le \mathbb{E}L < H(P) + D(P \parallel Q) + 1, \tag{A.1}$$

where $\mathbb{E}L$ the expected code length for encoding P. According to Eq. (A.1), if we encode P optimally, $D(P \parallel Q)$ can be interpreted as the extra symbols we need to encode the Q. Consequently, minimizing $D(P \parallel Q)$ reduces the redundancy in coding. The maximum likelihood method minimizes $D(P \parallel Q)$ directly. There are many other variants for the divergence metrics. For example, the vanilla GAN proposed in [28] is equivalent to minimizing the Jensen–Shannon divergence [47]

$$d(P, Q) = \frac{1}{2}D(P \parallel M) + \frac{1}{2}D(Q \parallel M),\ M = \frac{P+Q}{2}.$$

The least square GAN [48] is proved to minimize the χ^2 divergence for certain parameters. There is also a class of GANs that minimizes f-divergence [21], which is also known as Ali–Silvey distances

$$d(P, Q) = \int_{\mathcal{X}} Q(x) f\left(\frac{P(dx)}{Q(dx)}\right),$$

where $f : \mathbb{R}_+ \to \mathbb{R}$, $f(1) = 0$ is a convex, lower semi-continuous function.

Integral probability metrics (IPM) [49] is a class of distance measures on probabilities that measures the largest discrepancy in expectation over a class of witness functions, which is defined as

$$d(P, Q) = \sup_{f\in\mathcal{F}} \left| \int f dP - \int f dQ \right|,$$

Table A.3
Examples of IPM [50].

$\mathcal{F}$	Description	Note
$\{f : \|f\|_L \le 1\}$	Kantorovich metric	The dual representation of the Wasserstein distance
$\{f : \|f\|_L + \|f\|_\infty \le 1\}$	Dudley metric	
$\{f : \|f\|_\infty \le 1\}$	L_1 distance total variation metric×2	Equivalent to $\int \|P(dx) - Q(dx)\|$. The only nontrivial metric that belongs to f-divergence and IPM
$\{\mathbf{1}_{(-\infty,t]} : t \in \mathbb{R}^d\}$	Kolmogorov distance	
$\{f : \|f\|_{\mathcal{H}} \le 1\}$	Kernel distance	$\mathcal{H}$ represents a reproducing kernel Hilbert space (RKHS)

where P, Q are two probabilities and $\mathcal{F}$ is a class of real-valued bounded measurable functions. Examples of such distances are shown in Table A.3; in the table, $\|f\|_L$ is the Lipschitz semi-norm of a bounded continuous real-valued function f

$$\|f\|_L := \sup\left\{\frac{|f(x)-f(y)|}{|x-y|} : x \neq y\right\}.$$

The maximum mean discrepancy belongs to the kernel distance,

$$d(P,Q) = \sup_{f\in\mathcal{F}} \left(\mathbb{E}_{x\sim P} f(x) - \mathbb{E}_{y\sim Q} f(y)\right). \tag{A.2}$$

When $\mathcal{F} = C^0(\mathcal{X})$, i.e., the set of all continuous, bounded functions on $\mathcal{X}$ [51], we have $d(P,Q) = 0$ if and only if $P = Q$ and thus it can be used as a metric for proximity of P and Q. Such functions can be approximated with functions in a universal reproducing kernel Hilbert space [52]. The latter metric is used to design MMD GAN [53].

Optimal transport (OT) [54] can also be used to derive probability metrics. It basically considers the cost of transforming one probability distribution to another. Let $\mathcal{X}$ and $\mathcal{Y}$ be two measurable space. P, Q are two probability distributions on $\mathcal{X}$ and $\mathcal{Y}$ respectively. The Kantorovich problem in optimal transport is

$$\begin{aligned}&\min_\pi \int c(x,y)d\pi(x,y),\\ &s.t.\ P_{\mathcal{X}\#}\pi = P, \quad P_{\mathcal{Y}\#}\pi = Q,\end{aligned} \tag{A.3}$$

where $P_{\mathcal{X}\#}\pi$ and $P_{\mathcal{Y}\#}\pi$ denote the marginals of π with respect to the first and second component. If $\mathcal{X} = \mathcal{Y}$, and let $d(x,y)$ be a distance on the space $\mathcal{X}$. If $c(x,y) = d(x,y)^p$, $p \ge 1$ in Eq. (A.3), the p-Wasserstein distance on $\mathcal{X}$ is defined as

$$d(P,Q) = \left(\min_{\mu\in\mathcal{L}(P,Q)} \int c(x,y)^p d\pi(x,y)\right)^{1/p},$$

where $\mathcal{L}(P,Q)$ is the set of all measures with marginals P and Q, and $\rho \ge 0$ is the cost function. For example, if $\mathcal{X} = \mathbb{R}^n$, we can pick $d(x,y) = |x-y|$. Different from many divergences, the Wasserstein distance is a true distance, i.e., $d(x,y) = d(y,x)$ and satisfies the triangle inequality. When $p = 1$, the 1-Wasserstein distance has a supremum form due to Kantorovich–Rubinstein duality

$$d(P,Q) = \sup_{\|f\|_L\le 1} \mathbb{E}_{x\sim P}[f(x)] - \mathbb{E}_{x\sim Q}[f(x)],$$

where $\|f\|_L \le 1$ indicates f is 1-Lipschitz. The 1-Wasserstein distance is adopted in Wasserstein GAN [29].

Good performance in one metric does not necessarily mean the same in another metric. One would pick an appropriate metric for her specific applications. Also the choice of metrics has significant impact on the optimization algorithm. As an illustration, Table A.4 shows the probability metric for two delta distribution and two Gaussian distributions. If we are comparing discrete distributions and we are using KL-divergence, JS-divergence or the total variation metric, we may get a constant or infinity number, and consequently the gradients will be meaningless.

Table A.4
Different probability metrics for two delta distribution. Here $\theta_1 \neq \theta_2$.

Metrics	$P = \delta_{\theta_1},\ Q = \delta_{\theta_2}$
KL-divergence	$+\infty$
JS-divergence	$\log 2$
Wasserstein-2 distance	$\|\theta_1 - \theta_2\|$
Total variation distance	1
IPM $\mathcal{F} = \{x, x^2\}$	$\max\{\|\theta_1, \theta_2\|, \|\theta_1^2 - \theta_2^2\|\}$

Appendix B. Proof of Theorem 1

Proof. Since $\mathbb{E}\left(\frac{1}{x}\right) < \infty$, by law of large numbers we have

$$\frac{1}{n}\sum_{i=1}^{n}\frac{1}{x_i} \to X_{-1},\ n \to \infty. \tag{B.1}$$

We first verify that $\mathbb{E}\left(\frac{y}{x}\right) < \infty$ so that the law of large numbers holds. For CIR processes, we have [55]

$$\mathbb{E}[y|x] = xe^{-\kappa\Delta t} + \tau^*(1 - e^{-\kappa\Delta t}),$$

thus

$$\mathbb{E}\left[\frac{y}{x}\right] = \mathbb{E}\left[\mathbb{E}\left[\frac{y}{x}\middle| x\right]\right] = e^{-\kappa\Delta t} + \tau^*(1 - e^{-\kappa\Delta t})X_{-1} < \infty.$$

Therefore

$$\frac{1}{n}\sum_{i=1}^{n}\frac{u_i}{x_i} \to e^{-\kappa\Delta t} + \tau^*(1 - e^{-\kappa\Delta t})X_{-1}, \quad n \to \infty. \tag{B.2}$$

Plug Eqs. (B.1) and (B.2) into Eq. (13) we obtained using the continuous mapping theorem [56]

$$\hat{\tau}_n \to \frac{\tau^*(1 - e^{-\kappa\Delta t})}{\kappa\Delta t} + \frac{e^{-\kappa\Delta t} + \kappa\Delta t - 1}{X_{-1}\kappa\Delta t} = \tau^* + \mathcal{O}(\Delta t). \quad \square$$

Appendix C. Proof of Theorem 2

Proof. The log-likelihood function for estimating κ is

$$l_\kappa(x, y) = -\frac{1}{2}\left(\frac{y - x - \kappa(\tau - x)\Delta t}{\sigma\sqrt{x\Delta t}}\right)^2 + \log f_1(x) - \log\left(\sigma\sqrt{2\pi x\Delta t}\right),$$

and therefore the empirical log likelihood function is

$$\sum_{i=1}^{n} l_\kappa(x_i, u_i) = -\frac{1}{2}\sum_{i=1}^{n}\left(\frac{u_i - x_i - \kappa(\tau - x_i)\Delta t}{\sigma\sqrt{x_i\Delta t}}\right)^2 + \sum_{i=1}^{n}\log f_1(x_i) - \sum_{i=1}^{n}\log\left(\sigma\sqrt{2\pi x_i\Delta t}\right).$$

The maximum likelihood estimator can be computed as

$$\sum_{i=1}^{n} l'_\kappa(x_i, u_i) = 0 \Rightarrow \hat{\kappa}_n = \frac{1}{\Delta t}\frac{\tau\frac{1}{n}\sum_{i=1}^{n}\frac{u_i}{x_i} - \frac{1}{n}\sum_{i=1}^{n}u_i - \tau + \frac{1}{n}\sum_{i=1}^{n}x_i}{\tau^2\frac{1}{n}\sum_{i=1}^{n}\frac{1}{x_i} - 2\tau + \frac{1}{n}\sum_{i=1}^{n}x_i}. \tag{C.1}$$

By the law of large numbers, we have

$$\frac{1}{n}\sum_{i=1}^{n}\frac{1}{x_i} \to X_{-1}, \qquad \frac{1}{n}\sum_{i=1}^{n}x_i \to X_0; \tag{C.2}$$

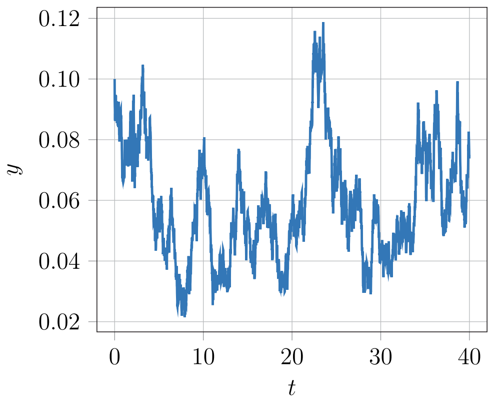

Fig. D.16. A sample path of the CIR process.

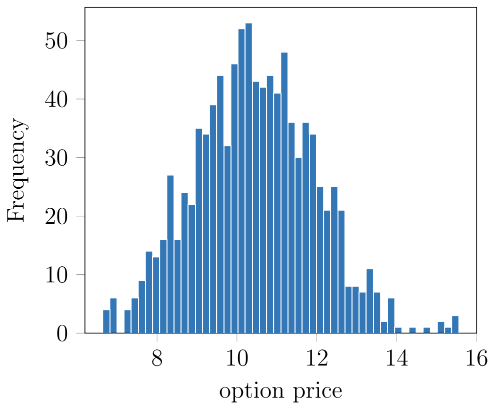

Fig. D.17. Observations of option prices. Each data point is generated using Eq. (25).

in addition, we have

$$\frac{1}{n}\sum_{i=1}^{n} u_i \to \mathbb{E}[\mathbb{E}[y|x]] = X_0 e^{-\kappa^* \Delta t} + \tau(1 - e^{-\kappa^* \Delta t}). \tag{C.3}$$

Plug Eqs. (C.2) and (C.3) into Eq. (C.1) and note that $e^{-\kappa^* \Delta t} \to 1 - \kappa^* \Delta t$, $\Delta t \to 0$. Then, we have

$$\begin{aligned}\hat{\kappa}_n &\to \frac{1}{\Delta t}\frac{\tau\left(e^{-\kappa^*\Delta t} + \tau(1-e^{-\kappa^*\Delta t})X_{-1}\right) - \left(X_0 e^{-\kappa^*\Delta t} + \tau(1-e^{-\kappa^*\Delta t})\right) - \tau + X_0}{\tau^2 X_{-1} - 2\tau + X_0} \\ &\to \kappa^* + \mathcal{O}(\Delta t), \quad \Delta t \to 0.\end{aligned}$$

Since

$$l''_\kappa(x) = -\frac{(\tau - x)^2 \Delta t}{\sigma^2 x},$$

we have

$$I(\kappa) = -\mathbb{E}(l''_\kappa(x)) = -\frac{\Delta t}{\sigma^2}\left(\tau^2 \mathbb{E}\left(\frac{1}{x}\right) - 2\tau + \mathbb{E}(x)\right) = \frac{\Delta t}{\sigma^2}(\tau^2 X_{-1} - 2\tau + X_0). \qquad \square$$

Appendix D. Sample data in Sections 4.4 and 4.5

Figs. D.16 and D.17 show sampled data used in Sections 4.4 and 4.5.

References

[1] Samuel H. Rudy, Steven L. Brunton, Joshua L. Proctor, J. Nathan Kutz, Data-driven discovery of partial differential equations, Sci. Adv. 3 (4) (2017) e1602614.

[2] Samuel Rudy, Alessandro Alla, Steven L. Brunton, J. Nathan Kutz, Data-driven identification of parametric partial differential equations, SIAM J. Appl. Dyn. Syst. 18 (2) (2019) 643–660.

[3] Dhruv Patel, Raghav Tibrewala, Adriana Vega, Li Dong, Nicholas Hugenberg, Assad A. Oberai, Circumventing the solution of inverse problems in mechanics through deep learning: Application to elasticity imaging, Comput. Methods Appl. Mech. Engrg. 353 (2019) 448–466.

[4] Ralph C. Smith, Uncertainty Quantification: Theory, Implementation, and Applications, Vol. 12, Siam, 2013.

[5] Yibo Yang, Paris Perdikaris, Adversarial uncertainty quantification in physics-informed neural networks, J. Comput. Phys. 394 (2019) 136–152.

[6] Anqi Shao, A Fast and Exact Simulation for CIR Process (PhD thesis), University of Florida Gainesville, FL, 2012.

[7] H.E. Daniels, The asymptotic efficiency of a maximum likelihood estimator, in: Fourth Berkeley Symposium on Mathematical Statistics and Probability, Vol. 1, University of California Press, Berkeley, 1961, pp. 151–163.

[8] Harald Cramér, Mathematical Methods of Statistics, Vol. 9, Princeton university press, 1999.

[9] Lucien Le Cam, et al., On the asymptotic theory of estimation and testing hypotheses, in: Proceedings of the Third Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Contributions to the Theory of Statistics, The Regents of the University of California, 1956.

[10] Thangarajah Akilan, Q.M. Jonathan Wu, Yimin Yang, Fusion-based foreground enhancement for background subtraction using multivariate multi-model Gaussian distribution, Inform. Sci. 430 (2018) 414–431.

[11] David M. Blei, Alp Kucukelbir, Jon D. McAuliffe, Variational inference: A review for statisticians, J. Amer. Statist. Assoc. 112 (518) (2017) 859–877.

[12] Masoumeh Dashti, Andrew M. Stuart, The Bayesian approach to inverse problems, in: Handbook of Uncertainty Quantification, Springer, 2016, pp. 1–118.

[13] George E.P. Box, George C. Tiao, Bayesian Inference in Statistical Analysis, Vol. 40, John Wiley & Sons, 2011.

[14] Arthur P. Dempster, A generalization of Bayesian inference, J. R. Stat. Soc. Ser. B Stat. Methodol. 30 (2) (1968) 205–232.

[15] David C. Knill, Whitman Richards, Perception as Bayesian Inference, Cambridge University Press, 1996.

[16] Matthew James Beal, et al., Variational Algorithms for Approximate Bayesian Inference, university of London London, 2003.

[17] Christophe Andrieu, Nando De Freitas, Arnaud Doucet, Michael I. Jordan, An introduction to MCMC for machine learning, Mach. Learn. 50 (1–2) (2003) 5–43.

[18] Radford M. Neal, et al., MCMC using hamiltonian dynamics, Handb. Markov Chain Monte Carlo 2 (11) (2011) 2.

[19] Michael Johannes, Nicholas Polson, MCMC methods for continuous-time financial econometrics, in: Handbook of Financial Econometrics: Applications, Elsevier, 2010, pp. 1–72.

[20] Don Van Ravenzwaaij, Pete Cassey, Scott D. Brown, A simple introduction to Markov Chain Monte Carlo sampling, Psychon. Bull. Rev. 25 (1) (2018) 143–154.

[21] Sebastian Nowozin, Botond Cseke, Ryota Tomioka, F-GAN: Training generative neural samplers using variational divergence minimization, in: Advances in Neural Information Processing Systems, 2016, pp. 271–279.

[22] Atilim Gunes Baydin, Barak A Pearlmutter, Alexey Andreyevich Radul, Jeffrey Mark Siskind, Automatic differentiation in machine learning: a survey, J. Mach. Learn. Res. 18 (2018) 1–43.

[23] Kailai Xu, Eric Darve, ADCME: Learning spatially-varying physical fields using deep neural networks, 2020, arxiv preprint arXiv:2011.11955.

[24] Dong C. Liu, Jorge Nocedal, On the limited memory BFGS method for large scale optimization, Math. Program. 45 (1–3) (1989) 503–528.

[25] Ciyou Zhu, Richard H. Byrd, Peihuang Lu, Jorge Nocedal, Algorithm 778: L-BFGS-b: Fortran subroutines for large-scale bound-constrained optimization, ACM Trans. Math. Softw. 23 (4) (1997) 550–560.

[26] Kamil Kladívko, Maximum likelihood estimation of the Cox-Ingersoll-Ross process: the Matlab implementation, Tech. Comput. Prague 7 (2007).

[27] Robert A. Jarrow, Modeling Fixed-Income Securities and Interest Rate Options, Stanford University Press, 2002.

[28] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio, Generative adversarial nets, in: Advances in Neural Information Processing Systems, 2014, pp. 2672–2680.

[29] Martin Arjovsky, Soumith Chintala, Léon Bottou, Wasserstein GAN, 2017, arxiv preprint arXiv:1701.07875.

[30] Diederik P. Kingma, Jimmy Ba, Adam: A method for stochastic optimization, 2014, arxiv preprint arXiv:1412.6980.

[31] Geoffrey Hinton, Nitish Srivastava, Kevin Swersky, Lecture slides, 2019, http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf. (Accessed on 04/01/2019).

[32] Anders Skajaa, Limited Memory BFGS for Nonsmooth Optimization (Master's Thesis), Citeseer, 2010.

[33] Alexia Jolicoeur-Martineau, GANs beyond divergence minimization, 2018, arxiv preprint arXiv:1809.02145.

[34] Ian Goodfellow, NIPS 2016 tutorial: Generative adversarial networks, 2016, arxiv preprint arXiv:1701.00160.

[35] In Jae Myung, Tutorial on maximum likelihood estimation, J. Math. Psych. 47 (1) (2003) 90–100.

[36] Alexander Ly, Maarten Marsman, Josine Verhagen, Raoul PPP Grasman, Eric-Jan Wagenmakers, A tutorial on Fisher information, J. Math. Psych. 80 (2017) 40–55.

[37] Gene H. Golub, Charles F. Van Loan, Matrix Computations, Vol. 3, JHU press, 2012.

[38] Paul Glasserman, Bin Yu, Large sample properties of weighted Monte Carlo estimators, Oper. Res. 53 (2) (2005) 298–312.

[39] David G. Luenberger, et al., Investment Science, OUP Catalogue, 1997.
[40] Behnam Neyshabur, Srinadh Bhojanapalli, Ayan Chakrabarti, Stabilizing GAN training with multiple random projections, 2017, arxiv preprint arXiv:1705.07831.
[41] Kevin Roth, Aurelien Lucchi, Sebastian Nowozin, Thomas Hofmann, Stabilizing training of generative adversarial networks through regularization, in: Advances in Neural Information Processing Systems, 2017, pp. 2018–2028.
[42] Changsheng Zhou, Jiangshe Zhang, Junmin Liu, Lp-WGAN: Using Lp-norm normalization to stabilize Wasserstein generative adversarial networks, Knowl.-Based Syst. 161 (2018) 415–424.
[43] Alison L. Gibbs, Francis Edward Su, On choosing and bounding probability metrics, Internat. Statist. Rev. 70 (3) (2002) 419–435.
[44] Gavin E. Crooks, On measures of entropy and information, Tech. Note 9 (2017) v4.
[45] Thomas M. Cover, Joy A. Thomas, Elements of Information Theory, John Wiley & Sons, 2012.
[46] Bin Yu, Tutorial: Information theory and statistics, 2008, https://www.icmla-conference.org/icmla08/slides1.pdf, (Accessed on 05/05/2019).
[47] Jianhua Lin, Divergence measures based on the Shannon entropy, IEEE Trans. Inform. Theory 37 (1) (1991) 145–151.
[48] Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, Stephen Paul Smolley, Least squares generative adversarial networks, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 2794–2802.
[49] Bharath K. Sriperumbudur, Kenji Fukumizu, Arthur Gretton, Bernhard Schölkopf, Gert R.G. Lanckriet, On integral probability metrics, ϕ-divergences and binary classification, 2009, arxiv preprint arXiv:0901.2698.
[50] Bharath K. Sriperumbudur, Kenji Fukumizu, Arthur Gretton, Bernhard Schölkopf, Gert R.G. Lanckriet, et al., On the empirical estimation of integral probability metrics, Electron. J. Stat. 6 (2012) 1550–1599.
[51] Richard M. Dudley, Real Analysis and Probability, Chapman and Hall/CRC, 2018.
[52] Ingo Steinwart, On the influence of the kernel on the consistency of support vector machines, J. Mach. Learn. Res. 2 (Nov) (2001) 67–93.
[53] Chun-Liang Li, Wei-Cheng Chang, Yu Cheng, Yiming Yang, Barnabás Póczos, MMD GAN: Towards deeper understanding of moment matching network, in: Advances in Neural Information Processing Systems, 2017, pp. 2203–2213.
[54] Gabriel Peyré, Marco Cuturi, et al., Computational optimal transport, Found. Trends Mach. Learn. 11 (5–6) (2019) 355–607.
[55] Ching-Sung Chou, Hsien-Jen Lin, Some properties of CIR processes, Stoch. Anal. Appl. 24 (4) (2006) 901–912.
[56] Alessandro Rinaldo, Lecture Notes. http://www.stat.cmu.edu/~arinaldo/Teaching/36752/S18/Scribed_Lectures/Apr5.pdf.