Novel and flexible parameter estimation methods for data-consistent inversion in mechanistic modelling

Timothy Rumbell, Jaimit Parikh, James Kozloski and Viatcheslav Gurev

IBM Research, Hybrid Biological-AI Modeling, Yorktown Heights, NY, USA

Predictions for physical systems often rely upon knowledge acquired from ensembles of entities, e.g. ensembles of cells in biological sciences. For qualitative and quantitative analysis, these ensembles are simulated with parametric families of mechanistic models (MMs). Two classes of methodologies, based on Bayesian inference and population of models, currently prevail in parameter estimation for physical systems. However, in Bayesian analysis, uninformative priors for MM parameters introduce undesirable bias. Here, we propose how to infer parameters within the framework of stochastic inverse problems (SIPs), also termed data-consistent inversion, wherein the prior targets only uncertainties that arise due to MM non-invertibility. To demonstrate, we introduce new methods to solve SIPs based on rejection sampling, Markov chain Monte Carlo, and generative adversarial networks (GANs). In addition, to overcome limitations of SIPs, we reformulate SIPs based on constrained optimization and present a novel GAN to solve the constrained optimization problem.

1. Introduction

In the era of machine learning (ML), mechanistic modelling is indispensable for performing prediction with datasets of small sizes, which are typical in biology, medicine and other physical sciences. Mechanistic modelling complements the power of pure ML by introducing formal model-based explanations of the data and additional prior information derived from known physical mechanisms. The first stage of generating predictions with mechanistic models (MMs) is fitting a model to reproduce known physical samples, solving an inverse problem. One widely used form of inverse problem estimates parameters in sets of observations acquired from single individuals. Another form of
inverse problems, that we discuss in this paper, is where each data sample represents a single observation of an individual within an ensemble, and the goal is to find a parametric family of MMs that recreates the distribution of observations from the entire ensemble.

The above inverse problem can be reduced to a stochastic inverse problem (SIP) [1]. In SIPs with deterministic MMs, the goal is to find a distribution $Q_X$ of inputs to a function $y = M(x)$, which, after ‘push-forward’ by the function $M$, produces a given target distribution $Q_Y$, such as $y = M(x) \sim Q_Y$, $x \sim Q_X$. Usually the function $M$ is not invertible, so multiple (i.e. an infinite number of) input distributions can produce the target. Therefore, the prior $P_X$ is introduced to restrict the solution using assumptions grounded in existing knowledge of parameter ranges and distributions.

A set of methods for $Q_X$ estimation have been developed [2], termed ‘consistent Bayesian inference’ [1] or ‘data-consistent inversion’ [3], which were initially developed for deterministic models [1] and have recently been extended to problems involving stochastic models $y = M(x, \epsilon)$ [3] with an additional vector of random variables, $\epsilon$, modelling physical noise. These methods are based on Monte Carlo sampling, importance sampling, rejection sampling and density estimation. However, several major drawbacks prevent such methods from being widely applied in SIPs, including diminishing accuracy as dimensionality increases and difficulty handling the type of complex inference scenarios and arbitrary data distributions that present themselves in practice. In the current work, we aim to overcome these challenges by reformulating SIP as an optimization problem. We present new methods to solve SIPs by adapting previous rejection sampling approaches, as well as introducing deep learning into SIPs for the first time to solve our optimization problem, opening new avenues for solving SIPs.

The first novel algorithm we introduce is a sequential rejection algorithm that performs data-consistent inversion [3] by employing a series of optimization steps. We refine this method using Markov chain Monte Carlo (MCMC) to initialize a proposal distribution prior to rejection. These approaches, described in Methods, form new benchmarks for SIP using conventional ML methods, which we can use for comparison and evaluation of our novel deep learning networks.

Although deep learning has not previously been used in SIPs, multiple deep learning methods have been recently developed for application to the related problem of model inversion for single observations, termed the parameter identification problem (PIP) or simulation-based inference (SBI) [4–6]. These include conditional networks such as conditional normalizing flows [7–9] and the conditional generative adversarial network (c-GAN) [10,11]. Therefore, we first tested the c-GAN [12] to estimate $Q_X$, while identifying some of the limitations that this approach shares with prior SIP methods. We then show that more complex SIPs and SIPs with stochastic MMs can be solved when formulated as constrained optimization problems using a novel generative model architecture, described in Methods, which we implement here using GANs.

We experimentally evaluate our methods on a range of example problems and demonstrate equivalent performance to the benchmark method of rejection sampling. Next, we tackle problems that are not accessible to existing methods and therefore analyse these deep learning results without comparison. For an intuitive example of a high-dimensional parameter inference problem involving highly structured data, we present super-resolution imaging on the MNIST dataset reformulated as a SIP. In the electronic supplementary material, we demonstrate the ability to extend our GAN architecture to solve a parameter inference problem in an intervention scenario, analogous to drugging a population of cells in a biological experiment. Our results establish GANs as a new instrument that can solve configurations of parameter inference problems for which there are currently no solutions.

2. Methods

2.1. Background and related work

In the inverse problem, observable vector-valued functions $t \rightarrow s_r(t)$, e.g. sets of experimental signal waveforms $\{s_r: r \in J\} \subseteq S$, are recorded from objects in an ensemble. The functions of time $s_r(t)$ are indexed by a discrete or continuous index $r \in J$. We are interested in finding solutions $\{f(t; x, \epsilon): x \in \mathbb{R}^m, \epsilon \in \mathbb{R}^d\} \subseteq S$ of differential equations for a given MM that approximate the experimental observations. Here $x$ is a vector of MM parameters, $\epsilon$ is a vector of random variables and $S$ is a functional space of continuous time signals. Feature vectors $L(s_r(\cdot))$ and $L(f(\cdot); x, \epsilon)$ (also referred to as quantities of interest) are extracted from experimental $s_r$ and simulated $f(\cdot; x, \epsilon)$ signals using some map $L: S \rightarrow \mathbb{R}^m$. In the SIP, the model is usually reduced to a function $y = M(x, \epsilon)$, defined as $M(x, \epsilon) = L(f(\cdot; x, \epsilon))$. If the objects are approximated by deterministic MMs, then $y = M(x)$. The goal is to find the distribution of MM parameters $Q_X$, which when passed through $M$, generates a distribution of outputs matching the distribution of features $Q_Y$ extracted from
experimental signals $L(s, \cdot)$. The model function $M$ could be in a closed form or represented by a surrogate trained on features from numerical solutions of MM differential equations. Therefore in analysis, we use the function $M$ rather than the original differential equation model and refer to $M$ as the ‘model’. Feature extraction functions $L$ are typically hand-crafted, but recent work has proposed automated feature extraction within the SIP context [13]. It should be emphasized that SIP is different from parameter identification problems that aim to estimate parameters for observed data from a single individual, which are often solved using classical Bayesian inference. These two different parameter inference objectives have recently been compared in detail [14]. In general, when sampling from the posterior using Bayesian inference, samples will be biased by the prior and, when passed through the model, will not generate samples from the target distribution (for empirical comparison, see [1,14]).

In the case of deterministic MMs, Poole & Raftery [2] showed that given random variables $X$ and $Y$, linked deterministically by $y = M(x)$, the density $q_Y(y)$ of function outputs can be mapped to the density of function inputs $q_X(x)$ coherent to the input data using the equation

$$q_X(x) = q_Y(y) \frac{p_X(x)}{p_Y(y)} \bigg|_{y=M(x)},$$

(2.1)

where $p_X(x)$ is the prior density on the input and $p_Y(y)$ is the model-induced prior density obtained upon sampling from $p_X(x)$ and applying the function $M$ to the samples (push-forward of the prior). For invertible functions, the ratio $p_X(x)/p_Y(M(x))$ is simply the Jacobian of the function $M$. Recent work [1], which rediscovered Poole & Raftery’s formulation, outlines direct parallels between (2.1) and classical Bayesian inference and provides measure theoretic proofs of important properties of (2.1), such as solution stability. Modifications [3] of (2.1) (termed ‘data-consistent inversion’) generalized this framework to stochastic models and extended its applicability to a wider variety of SIPs with stochastic $y = M(x, \epsilon)$:

$$q_{X,E}(x, \epsilon) = q_Y(y) \frac{p_{X,E}(x, \epsilon)}{p_Y(y)} \bigg|_{y=M(x,\epsilon)}.$$  

(2.2)

After sampling from $q_{X,E}$, the marginal samples are taken by dropping $\epsilon$. Note that there is no difference in treatment of model parameters $x$ and noise $\epsilon$ in (2.2). In real-world problems, the nature of noise is often assumed a priori (e.g. Gaussian), and noise is assumed to be independent of mechanistic model parameters. However, $x$ and $\epsilon$ can become correlated in samples from $q_X(x, \epsilon)$ [3]. In this paper, we present a method for configuring a parametric model of the noise during estimation of a data-consistent solution $q_X(x)$ to the general SIP with stochastic models.

Two methods are proposed in [1,3] to sample from $q_X$. First, the density $p_Y(y)$ can be estimated by some standard density estimator and parameters can be sampled by Monte Carlo methods from (2.1) and (2.2). The second option [1] is a rejection sampling algorithm (algorithm 1). For conciseness, we provide the version of the algorithm for deterministic models, since the methods to solve (2.1) and (2.2) are essentially the same. Another algorithm based on importance sampling was proposed in [2]. The main limitation of all above methods is high instability and variance due to inaccuracy in estimation of $p_Y(y)$, since this term appears in the denominators. For example, during rejection sampling, errors in estimation may lead to a large, possibly infinite, bound constant $B$ (step 7 in algorithm 1) and cause a high rejection rate.

**Algorithm 1.** Rejection sampling [1].

**Require:** Number of samples and iterations $N_S$; model $y = M(x)$; density $q_Y$

1: Sample $x^i_{1\ldots N_S} \sim p_X$
2: for $i=1\ldots N_S$
3: \hspace{1em} $y_i := M(x_i)$
4: end for
5: $P := \{x_i, y_i\}_{i=1}^{N_S}$
6: Use $\{y_i : (x_i, y_i) \in P\}$ to build density model $p_Y$
7: Compute $B := \max_i \frac{q_Y(y_i)}{p_Y(y_i)}$
8: for $\{x_i, y_i\} \in P$ do
9: \hspace{1em} $\rho \sim U(0,1)$
10: \hspace{1em} if $\rho > \frac{q_Y(y_i)}{B \times p_Y(y_i)}$ then
11: \hspace{2em} Reject $P := P \setminus (x_i, y_i)$
12: \hspace{1em} end if
13: end for
14: return $\{x_i : (x_i, y_i) \in P\}$
In parallel with the methods developed in [1] following [2], an empirically derived method to solve a similar problem was described in [15]. There, a sequential Monte Carlo stage was used to sample from \( q_Y(y)p_X(x) |_{y = \text{M}(x)} \) with uniform \( p_X(x) \), obtaining initial samples that are then refined using a rejection stage that reduces JS-divergence (JSD) between the samples and the target observations. This solution did not rely on the theoretical background of [1], but solves a similar problem using different methods.

Deep learning methods have not been applied for parameter inference in SIP. However, deep learning is widespread in the PIP/SBI problems mentioned above, wherein experimental data \( D = \{y_i\} \) are acquired from a single individual over several trials, and simulated data derive from a parametrizable, stochastic model with parameters \( x \) and outputs \( y \) simulating \( p(y|x) \). SBI is commonly solved using approximate Bayesian computation (ABC), or ‘likelihood-free’ methods, to bypass the intractable likelihood term [4]. The task is to infer the posterior distribution \( P(x|D) \) of model parameters \( x \) that likely generated the individual’s observed data. This scenario can be formulated for independent observations as

\[
p(x|D) = \frac{p(D|x) \cdot p(x)}{p(D)} \propto \prod_i p(y_i|x) \cdot p(x).
\]  

Recent advances in conditional neural density estimation have enabled a host of new approaches in the SBI domain to approximate the likelihood and posterior densities with networks trained on simulated data. Mixture density networks [5,16], emulator networks [17,18] and autoregressive flows [5,8] each approximate the likelihood \( p(y|x) \) or posterior \( p(x|y) \). Models of the posterior \( p(x|y) \) are trained on samples \( (x_i, y_i) \) by sampling \( x_i \sim P_X \) and running the MM \( y_i \sim p(y|x_i) \). The most advanced networks for parameter inference from the literature are conditional normalizing flows [19], which perform invertible transformations of random variables. Invertibility requires the determinant Jacobian to be cheaply computable, which limits the range of transformations that can be used. Advances such as masked autoregressive flows (MAFs) [7], inverse autoregressive flows (IAFs) [20] and neural autoregressive flows (NAFs) [21] have each improved the accuracy of inference achievable with normalizing flow models. These conditional density estimation networks provide explicit density models that can be used for fast approximation of density from multiple datasets \( D \) in SBI problems. Following the same trend, we tested the c-GAN as an implicit estimator of stochastic maps for MM parameter inference in SIPs. Other conditional distribution models, such as conditional diffusion models, could be used instead of c-GAN for this purpose.

Although conditional neural networks are suitable for parameter inference, they are essentially estimators of distributions from previously sampled sets of parameters, in contrast to other methods that actively explore parameter space and propose parameter samples for evaluation. These networks may require a large sample size for training and capacity to learn the conditional distribution, and inference accuracy may suffer in regions of \( y \) that have low density in \( P_Y \) (the model induced prior), necessitating sequential approaches [8]. In practice, however, conditional networks often show good performance even in high-dimensional parameter inference scenarios [11]. For SIPs with stochastic MMs c-GAN provides the same solution as rejection methods applied to (2.2). Here, we propose a reformulation of SIP as an optimization problem to facilitate the use of deep neural networks as a flexible approach to a broader class of SIP applications (see §3.2).

### 2.2. Parameter inference methods for stochastic inverse problems

We present four novel methods to solve SIPs. We use the first two, a modification of the rejection sampling (algorithm 1), and rejection sampling initialized with an MCMC stage, as numerical benchmarks for comparing the results of the subsequent GAN-based solutions.

#### 2.2.1. Modified rejection sampling

A substantial problem with algorithm 1 is the possibility of a large bound constant \( B \). To address this, we modified the algorithm to perform rejections in multiple iterations, selecting samples for rejection to incrementally minimize divergence between model outputs and observations at each iteration, as described in algorithm 2. The main difference from algorithm 1 is that the constant \( B \) does not have to be a bounding constant of the density ratio, but a parameter of the algorithm. If \( B \) is equal to the bounding constant, then \( N_I = 1 \) and algorithm 2 reduces to algorithm 1. We used \( B = 1 \) and \( N_I = 10 \) in all our examples, which was sufficient for \( q_Y(y) \) to converge to \( q_Y(y) \), resulting in distributions consistent with

\[
q_Y(y) \approx q_Y(y) = \sum_{i=1}^{N_I} \delta(y - y_i).
\]
of parameters that correspond to \( y \). In this case, \( p(x \mid y) \) is ill defined, since the distribution \( \pi(y) \) is an arbitrary function, \( \pi(y) \geq 0 \) with strict inequality in \( y \) where \( p(y) > 0 \). Examples of such proposal distributions could be simply the prior \( p_y(x) \), or the numerator of (2.1), \( q_y(y)p_x(x) \mid y = M(z) \). The latter is often closer to the solution, and we used it, sampled with MCMC, to boost the rejection method by providing a higher density of samples in the regions of interest from which to construct the initial density model in algorithm 2.

2.2.3. Conditional generative adversarial network for amortized inference

For deterministic models \( y = M(x) \), we are able to use the c-GAN [12] to find MM parameter distributions given experimental observations. In this case, \( p(x \mid y) \) is ill defined, since the distribution of parameters that correspond to \( y \) is degenerate. As [22] noted, one advantage of GANs is their ability to represent degenerate distributions, and we were able to apply c-GAN in SIPs with deterministic models. The generator network of c-GAN, \( x = G(y, z \mid \theta) \), with network parameters \( \theta \), which generates \( x \) from \( y \) and samples \( z \sim P_z \) of the base GAN distribution, is trained on the samples \( (x_i, y_i) \), where \( x_i \sim p(x) \) and \( y_i = M(x_i) \). At the inference stage, \( y_i \sim Q_y \) would be transferred to samples from estimated \( Q_x \) by \( z_i \sim P_z, x_i = G(y_i, z_i) \). The advantage of this approach is that it uses ‘amortized inference’ [4], providing solutions to any number of problems at the inference stage after being trained a single time using one simulated training dataset. In several examples, we demonstrate
Figure 1. Generative adversarial models for inference of model input parameters. Regularized GAN (r-GAN) solves the constrained optimization problem using the penalty method. The loss of the generator is the weighted sum of losses from r-GAN’s two discriminators. r-GAN enforces the equality of $Q_Y$ and $Q_{Y_t}$ and maximizes the overlap between $P_X$ and $Q_{X_t}$.

that results obtained using c-GAN are empirically consistent with results obtained from (2.1) and (2.2) using algorithm 2.

2.2.4. Regularized generative adversarial networks

In (2.1), the prior density $p_X(x)$, as in the Bayes formula, is used as the relative likelihood of model input parameter values. Here, we reformulate the problem as a constrained optimization, aiming to minimize divergence between the prior $P_X$ and the distribution $Q_{X_t}$ of parameters sampled by a generator in a GAN from some parametric family $G_\theta \in \{G_\theta(\cdot) | \theta \in \Theta\}$, given that the distribution $Q_{Y_t}$ of the pushforward of $Q_{X_t}$ through the model $M$ matches the target distribution of observations $Q_Y$ (the primary goal in SIPs). This places a regularizing constraint on parameters sampled by the generator in a model we term the regularized generative adversarial network (r-GAN) solution to SIP, shown in figure 1. Thus, the problem is formulated as

$$
\begin{align*}
\text{given} & \quad P_X, \quad Q_Y, \quad y = M(x) \\
\text{minimize} & \quad D_f(Q_{X_t} || P_X) \\
\text{subject to} & \quad \text{supp}(X_t) \subseteq \text{supp}(X), \quad D_f(Q_{Y_t} || Q_Y) = 0 \\
\text{where} & \quad y_s = M(x_s) \sim Q_{Y_t}, \quad x_s \sim Q_{X_t}.
\end{align*}
$$

In (2.4), $D_f(\cdot || \cdot)$ is an f-divergence measure such as Jensen–Shannon (JS) divergence. This reformulation of the problem provides another way to account for the prior. We are looking for not just any distribution of model input parameters that produces $Q_Y$, but the distribution with minimal divergence from the prior. The additional constraint $\text{supp}(X_t) \subseteq \text{supp}(X)$ ensures that the distribution of the generated input parameters $X_t$ is within the prior bounds. To solve (2.4), we first convert the constrained optimization problem to a non-constrained problem with minimization of $w_X \times D_f(P_X || Q_{X_t}) + w_Y \times D_f(Q_{Y_t} || Q_Y)$, solving (2.4) with the penalty method. In a series of training steps, the weights $w_Y$ and $w_X$ are changed such that the weight $w_X$ is much smaller than the weight $w_Y$ at the last step. For example, we used $w_Y = 0, w_X = 1$ and $w_Y = 1.0, w_X = 0.1$ for the first and last steps, respectively. Non-constrained optimization could be solved with different methods including methods that resemble sequential Monte Carlo sampling [23], perturbing and resampling particles that represent MM parameter distributions in a series of importance sampling iterations. Rather than optimization by stochastically perturbing samples, we instead sample using a neural network generator, minimizing the divergence $D_f(P_X || Q_{X_t})$ over $\theta$ in the generator: $z \sim P_Z, \quad x_s = G_\theta(z) \sim Q_{X_t}$, where $P_Z$ is a base distribution, usually Gaussian. We can then optimize with stochastic gradient descent, and employ GAN discriminators to calculate the divergence measures in the optimization problem. The r-GAN has two discriminators, and the generator loss $L_G$ is composed of a weighted sum of losses due to both discriminators, generator loss $L_{D_X}$ due to discriminator $D_X$ and loss $L_{D_Y}$ due to $D_Y$:

$$
L_G = w_Y L_{D_Y} + w_X L_{D_X}.
$$

Different f-divergence measures could be applied using different GAN loss functions [24]. This type of network is similar to that used in adversarial variational optimization (AVO) [25], where adversarial optimization minimizes $D_f(Q_{Y_t} || Q_Y)$. However, a major difference is the use of the second discriminator to place a regularization on the generated samples, whereas AVO uses a penalty term that instead decreases sample entropy. Another discriminator-based method for model inversion has been proposed [26] in the context of ABC, but no regularization according to the parameter distribution is used in this example, and SIP solutions are not addressed.
Our SIP reformulation using r-GAN provides several benefits over existing methods. r-GAN makes more accurate predictions for general stochastic SIPs, demonstrated by our stochastic MM experiment, enables sampling from distributions of high-dimensional data as demonstrated by our imaging experiment, and can be extended to complex simulation configurations as demonstrated in our intervention experiment and additional experiments in electronic supplementary material.

3. Results

To illustrate our method’s applicability to a variety of MMs, we used four test functions as the MM \( y = M(x) \) and solved SIP for each using rejection sampling, MCMC-boosted rejection sampling, c-GAN and r-GAN methods. The first example was used as a standard example of SIP in [1,27], and we use it here to compare our SIP methods and highlight the role of the prior discriminator. For the remaining three examples, r-GAN is the only existing solution, and we therefore demonstrate scenarios that could not previously be solved accurately. Second, we use SIP with a stochastic simulator, which was attempted in a previous publication using algorithm 1 [3], and for which we show that r-GAN improves upon biased solutions produced using the other methods. Third, we use r-GAN for a super-resolution imaging SIP; an example of a high-dimensional implicit sampling problem with no analogous solution among other methods. Fourth, in the electronic supplementary material, we demonstrate SIP for an intervention scenario using an extension to r-GAN, which highlights the potential usefulness of this extensible framework for tackling complex SIP configurations. To quantify performance, we estimated both \( \text{JSD}(P_X||Q_X) \) and \( \text{JSD}(Q_Y||Q_{Y_g}) \) using classifiers trained on samples from the distributions. We provide summary implementation details here and full details on the GAN implementation, GAN configuration, JS-divergence estimation from implicit samples and MCMC methods in the electronic supplementary material. Demonstration code for r-GAN in Pytorch is available at https://github.com/IBM/rgan-demo-pytorch. For additional deterministic function tests from the literature, see electronic supplementary material.

For a fair comparison between inference methods, we used a fixed computational budget of 2 000 000 simulations of the MM. For rejection and c-GAN, these simulations were performed in advance, generating samples used to initialize the algorithms. In MCMC-boosted rejection, we perform simulations throughout iterations of MCMC (algorithm specifics are provided in the electronic supplementary material). The r-GAN requires simulation of the MM during training for each sample from the generator network, and we trained the r-GAN for 200 epochs with 10 000 samples per epoch. We did not perform elaborate computational benchmarks between methods, as we envision the methods being used in combination according to their strengths and weaknesses for specific problems.

3.1. Nonlinear function and the role of the prior

In the first example, we represented the mechanistic model by a nonlinear system of equations with two input parameters \((x_1, x_2)\):

\[
\begin{align*}
x_1 y_1^2 + y_2^2 &= 1 \\
y_1^2 - x_2 y_2^2 &= 1,
\end{align*}
\]  

(3.1)

with \( y_2 \) as the model output (shown in figure 2a as \( y \)). The target observation distribution \( Q_Y \) was synthetic data with distribution \( N(0.3, 0.025^2) \), truncated to the interval \((0, 0.6)\) (figure 2b, black).

We used this problem to assess the influence of the parameter prior on generated samples using two different \( P_X \). First, a uniform prior was considered with \( x_1 \sim U(0.79, 0.99) \) and \( x_2 \sim U(1 - 4.5\sqrt{0.1}, 1 + 4.5\sqrt{0.1}) \) as in [1]. Figure 2b shows the distribution of \( y_2 \) (after push-forward with (3.1)) obtained using rejection, MCMC, c-GAN and r-GAN to sample \( x_1 \) and \( x_2 \), with histograms of parameter samples for each method shown in figures 2d–g, respectively. JSD between the prior \( P_X \) and the generated samples \( Q_X \), and between the push-forward of the generated samples through the model \( Q_{Y_g} \) and the target output distribution \( Q_Y \) (figure 2c) indicated that all four methods performed well in estimating model input parameters coherent to the target distribution. While the c-GAN approach has lower \( \text{JSD}(Q_Y||Q_{Y_g}) \) (figure 2), indicating slightly more accurate sampling in model output space, it also has more artificial structure in parameter samples (figure 2f) than other approaches, likely due to the generator network learning the full conditional distribution \( P_X, Y \) required for amortized inference.
The inferred parameter distribution is influenced by $P$ between $\theta$. Here, a square plate is wobbling around a pivot positioned near the origin, and the height of the plate $y$ at a location on the surface $(p_1, p_2)$ is given by

$$y = y_0 + x_1(p_1 + \epsilon_1) + x_2(p_2 + \epsilon_2),$$

(3.2)

To test SIP using GANs with a stochastic mechanistic model, we used a model of a tiltmeter, as in [3].
where \( y_0 = 3.0 \) is the height of the plate above the origin, \((x_1, x_2)\) are the slopes of the plate (the parameters to be inferred), and \((e_1, e_2)\) are stochastic noise embedded in the simulator, effectively perturbing the measurement location. In a deterministic scenario, the height of the plate is measured with complete accuracy \((e_1, e_2) = (0, 0)\) at two positions \(P_A = (0.6, 0.6)\) and \(P_B = (0.8, 0.6)\) at different times to form the target \(Q_Y\), comprising measurements \((y_A, y_B)\). The parameter prior was considered with \( x \sim \mathcal{U}(0, 2) \) for \(x_1\) and \(x_2\), and synthetic observation data were simulated using (3.2), with \((x_1, x_2)\) drawn from \((\mathcal{U}(0.85, 1.6), \mathcal{U}(1.45, 1.85))\), creating the target density shown in figure 3a,b. Figure 3a,b shows the distribution of \( y \) obtained by push-forward with (3.2) of \(x_1\) and \(x_2\) sampled using rejection, MCMC, c-GAN and r-GAN. Histograms of parameter samples for each method (shown in figure 3d–g) demonstrate that all inference methods sample precisely from the ‘true’ parameter distribution that was used to create the target data.

Adding stochasticity to the model, we created the synthetic target \(Q_Y\), shown in figure 3i,j, by calculating (3.2) with \((x_1, x_2)\) drawn from \((\mathcal{U}(0.85, 1.6), \mathcal{U}(1.45, 1.85))\) and \((e_1, e_2)\) drawn from \((N(0, 0.075^2), N(0, 0.075^2))\), embedding noise in the locations at which the measurements were taken, as in [3]. When sampling from \(Q_X\), the simulations were run with priors \((N(0, 0.0825^2), N(0, 0.0825^2))\) for \((e_1, e_2)\), reflecting a scenario where the true noise in the real system is not fully known. Figure 3i,j shows the distribution of \( y \) obtained by push-forward with (3.2) of \(x_1\) and \(x_2\) sampled using rejection, MCMC, c-GAN and r-GAN. In this example, \(Q_Y\) for the rejection, MCMC-boosted rejection and c-GAN methods are biased away from the target \(Q_Y\) (figure 3i,j). This is because \((e_1, e_2)\) are only sampled a single time for each \((x_1, x_2)\), which induces a correlation between \(x\) and \(e\) in \(Q_X\), and the samples of \(e\) diverge from the true noise distribution. When samples from \(Q_X\) are then pushed forward through the MM, \((e_1, e_2)\) are resampled and \(Q_Y\) diverges from \(Q_Y\). Thus, these methods do not provide a solution that accurately estimates parameter density.

In the r-GAN framework, we were able to implement a more flexible network configuration that samples a parametrized model of noise during training. We incorporated a trainable parameter \(\sigma\) (shown in figure 3h) for the variance of a noise distribution \((N(0, \sigma^2))\), constrained by an additional discriminator \(D_{N}\) (figure 3h) to produce samples close to the prior assumptions about the noise distribution. The noise parameter was able to gradually shift from the prior value of 0.0825\(^2\) towards the ground truth value of 0.075\(^2\) during training. This enabled model parameter samples from r-GAN to be highly concentrated in the ground truth region, indicating that the network was able to find a result more consistent with the observed data. A major benefit of framing SIP as a constrained optimization is the ability to incorporate additional constraints into the problem formulation, which is made straightforward by our use of a neural network graph as in r-GAN (figure 1).

### 3.3. Stochastic inverse problem for super-resolution imaging

Our next experiment aimed to demonstrate SIP in a high-dimensional setting. We chose a super-resolution imaging scenario with the MNIST dataset for intuitive visualization of the results. The model \(y = M(x)\) is average pooling over an input window of size \(7 \times 7\) with stride 1 and without padding, which operates on a high resolution (HR) image as input \(x\) and outputs a blurred, low-resolution (LR) image \(y\). Our dataset of images representing the prior distribution \(P_X\) was the MNIST dataset, and we passed MNIST images labelled with digit ‘5’ through the model \(y = M(x)\) to obtain the target dataset \(Q_Y\).

As an initial training stage in all r-GAN experiments, we trained the generator to reproduce the prior distribution by setting \(w_Y = 0\) in (2.5). For the super-resolution scenario, this amounts to training a standard GAN to produce MNIST images. Samples from \(Q_X\) after training are shown in figure 4a, top right, and span the distribution of images from the full MNIST prior \(P_X\) (figure 4a, top left). We added the loss of the second discriminator \(D_{Y}\) by setting \(w_Y = 1\) to train the complete r-GAN. Images generated by \(G\) were passed through \(y = M(x)\) and compared with target samples, resulting in \(Q_X\) (figure 4a, bottom right), which when passed through \(M\), approximates \(Q_Y\) (figure 4a, bottom left). The samples consist of a variety of images that resemble those labelled ‘5’, as well as occasional images that resemble those with different labels (most notably ‘3’s, ‘6’s and ‘8’s), but which resemble LR ‘5’s after applying \(M\). Note that these images resembling alternatively labelled images are appropriate for the generator to sample, since, when blurred by \(M\), they are indistinguishable from blurred ‘5’s.

In other super-resolution imaging solutions, we observe that typically only a single image corresponding to a given LR image is sampled, as in the SRGAN [28] and PULSE [29] models. In PULSE, a GAN is first trained on the prior images. Then, the following objective is minimized:
Figure 3. Parameter inference for stochastic simulator (3.2). (a,b) Marginal KDEs of \( (y_x, y_d) \) constituting \( \mathcal{Q}_Y \) and the generated (inferred) output distributions \( \mathcal{Q}_{y} \) using rejection, MCMC, c-GAN and r-GAN. (c) Left: estimated JS-divergence between \( \mathcal{P}_Y \) and \( \mathcal{Q}_Y \). Right: estimated JS-divergence between \( \mathcal{Q}_Y \) and \( \mathcal{Q}_{y} \). (d–g) Histograms of \( \mathcal{Q}_X \) for each method. (h) Schematic diagram of r-GAN for stochastic MMs. Note the additional generator and discriminator for noise parameters compared with the r-GAN in Figure 1. (i–o) As in (a–g), but for the stochastic model, with embedded noise using \( \epsilon_1 \) and \( \epsilon_2 \) in (3.2). In (d–g) and (i–o), the white dashed rectangle denotes bounds set by \( \mathcal{P}_X \), and the blue dashed rectangle denotes bounds of the uniform ‘true’ parameter distribution.
min_{z \in S^d} \| DS(G(z)) - I_{LR} \|_1$, thus finding the latent vector $z$ on the $(d - 1)$-sphere with radius $\sqrt{d}$, where $d$ is the dimension of the latent space of the generator $G(z)$. $DS$ is the down-sampling function (our model $M$), and $I_{LR}$ is an LR image. We also used the PULSE model to find specific generated samples that closely match each image in $Q_Y$. Using either the GAN generator (trained during the initial stage: $w_Y = 0$) or the r-GAN generator, PULSE found optimal latent space values that generated HR samples, which, when blurred, matched single LR samples from $Q_Y$ (figure 4b).

Evaluating generated samples in a high-dimensional space is challenging [30]. Here, we measured JS-divergence between both $P_X$ and $Q_X$, and between $P_Y$ and $Q_Y$, using a classifier trained to distinguish generated and real samples, shown in table 1. However, regularization of the classifier affects the calculated divergence metric, as the implicit density model defined by the classifier depends on its chosen regularization and metaparameters. Generated datasets (labelled in the left column of table 1) are compared against subsets of MNIST for different digits (0, 1, ..., 9) or the whole dataset (All), using either low (LR) or high (HR) resolutions. GAN and r-GAN labels indicate the generator used for samples. PULSE indicates sampling performed with PULSE model and a generator. We discuss the evaluations further, and provide details of the classifier, in the electronic supplementary material.

The low JS-divergence of 0.22 compared to other digits between the MNIST images labelled ‘5’ and the samples from the r-GAN after applying $M$ to both (r-GAN (LR)) indicates that $Q_Y$, samples are close to the target $Q_Y$. The JS-divergence of 0.61 between the MNIST images labelled ‘5’ and the samples from the r-GAN before applying $M$ (r-GAN (HR)) (i.e. $M$’s parameter space) can be compared with the values when sampling MNIST images with PULSE using either the GAN or r-GAN generator (0.73 and 0.71, respectively). This indicates that sampling from the r-GAN without attempting to select specific
samples for each output of $M$ in the target dataset (i.e. treating superresolution imaging as a SIP) produces a distribution of HR images that is closer to the true distribution of ‘5’s in MNIST than when sampling specific generator outputs that are optimal for each image in the dataset.

3.4. Additional stochastic inverse problem examples in electronic supplementary material

A fundamental benefit of the r-GAN formulation of SIPs, as demonstrated in §§3.2 and 3.3, is the flexibility to reconfigure the framework to tailor it to entirely new types of SIP. Finally, we demonstrate a solution to a complex parameter inference problem that appears routinely in scientific disciplines using an intervention scenario, reflecting situations where recordings are made from a biological ensemble before and after a perturbation, such as administration of a drug to a tissue. In previous work, we applied such an intervention r-GAN to evaluation of the mechanism of action of a cardiac inotrope drug using a cardiac mechanics cellular model [31]. Here, we include a test demonstration of this approach using the Rosenbrock function as the MM in the electronic supplementary material.

To further validate the reliability of the r-GAN approach, we tested SIP using GANs for several other test functions, described in detail in the electronic supplementary material. For example, we used the Rosenbrock function with a multi-modal target output distribution dependent on disjoint parameter regions. All methods (rejection, MCMC and r-GAN) were able to accurately infer parameter samples coherent with the target distribution for a low-dimensional Rosenbrock function. In a high-dimensional Rosenbrock function example, the c-GAN struggled to constrain parameter samples to precisely match $Q_Y$ and $Q_Y^G$, due to the requirement that c-GAN learn the full conditional distribution. The MCMC and r-GAN methods performed well for this challenging test case, however.

For certain mechanistic models, a surrogate regression model approximating $y = M(x)$ and trained on a dataset calculated from the original model is required to make an otherwise non-differentiable or costly simulation accessible to the r-GAN framework. Two additional SIPs, a piecewise smooth function (to test parameter inference in a discontinuous space) and an ODE model (a typical format for a mechanistic simulator) [1], were solved using surrogate models in place of a direct solution to the model equations in the r-GAN. Parameters coherent with target outputs were sampled accurately for both the piecewise and ODE test functions by rejection, MCMC and r-GAN.

4. Discussion

Parameter estimation for MMs to fit ensembles of experimental observations in the domain of model output is a common task in biology. A typical problem is fitting MMs to biomarkers obtained from a set of cells with random variation of cell characteristics within the set. In practice, parametrizing MMs is a critical task when attempting to combine results of multiple experimental protocols in inference problems, and also for putting additional constraints into machine learning models in the form of an inductive bias provided by MMs.

Probably the most difficult step in MM-based analysis is parameter estimation. In the literature, investigators usually apply Bayesian methods to fit MMs to individual entities one-by-one, and then
aggregate the found model parameters to build a density model in MM parameter space [32]. The major drawback of this approach is that the prior on model parameters is rarely informative, and a uniform distribution within somewhat arbitrary ranges is often used. Thus, Bayesian analysis introduces unnecessary bias when there is little or no information available about MM parameters. Similar bias is also introduced in an approach termed ‘population of models’ [33].

Data-consistent inversion in SIPs provides a viable alternative to Bayesian parameter inference. In this approach, a density model of experimental observations is constructed, and the goal is to find a distribution of MM parameters that produces the target density. Although the prior also introduces a bias in the solution, the prior does not affect the distribution of model outputs. In this paper, we have presented new algorithms to solve SIPs addressing limitations of the methods found in previous research. These new methods can be divided into two categories. The rejection and r-GAN methods fall into the first category, which, similar to variational inference, optimizes distributions of model parameters by minimizing divergence between MM output distributions and target distributions. The distributions of MM parameters are represented either by sets of samples, as in the rejection methods, or by parametric density estimators, e.g. generator networks, as in the r-GAN. In both cases, samples are scored using discriminators in optimization algorithms. In the second category, instead of optimization, distributions of MM parameters for individual target samples are memorized by density estimators. We presented c-GAN as an example of this class of solution.

In the rejection algorithm, we used Gaussian mixture models to estimate the densities of the sets of samples to compute a density ratio for the rejection step. As an alternative, we could employ the density ratio trick with a classifier, as in GANs, to minimize f-divergence between the target and MM output distributions. The MCMC-boosted rejection algorithm aims to initialize the proposal distribution with a distribution closer to the solution, similar to the algorithm from [15]. The main limitation of the rejection algorithm is the prohibitively large number of initial samples from the prior that are required as dimensionality or problem complexity increases. Therefore, we looked for other algorithms in which particle sets are evolved within perturbation and resampling steps, similar to how sequential Monte Carlo methods are applied in ABC. To construct such an algorithm, we reformulated the SIP into a constrained optimization problem and proposed to solve it in a series of unconstrained optimization steps. These steps could be implemented as a combination of perturbations according to a kernel, followed by application of the rejection algorithm with particle scores calculated from the sum of weighted discriminators (2.4). As one potential method to solve SIPs using stochastic gradient descent, we implemented r-GAN, a generative adversarial network with two discriminators.

A widely recognized limitation of GANs is that they are difficult to train, primarily caused by issues relating to mode collapse, thus limiting their adoption in general scientific applications. The stabilization method used here (see electronic supplementary material for details) enabled stable training across all examples without extensive hyperparameter search. Mode collapse may still occur if a solution to SIP is in a low density region of the prior distribution, but one potential solution to this is to combine an explicit density estimator (e.g. normalizing flow networks) with a classifier to apply boosting for training the prior discriminator [34]. We anticipate such limitations of GANs can in the future be mitigated by more advanced training techniques and GAN configurations.

One requirement of r-GAN is that the mechanistic model must be differentiable to allow backpropagation of \( L_{D_{G}} \) to \( G \). However, mechanistic simulators are often numerical solutions of differential equations, and cannot be directly incorporated into a deep network. In two of our test examples, a differentiable surrogate model was trained on samples from \( P_{X} \) paired with their outputs of \( M \) and used in place of the mechanistic model (see electronic supplementary material). This approach can introduce error if the target region of interest is undersampled during surrogate model training. Alternatively, instead of using a neural network for the GAN generator, distributions of model parameters can be represented by a set of particles, and the objective of (2.4) can be optimized by applying perturbations and resampling to the particle set. Parallels can be made to sequential Monte Carlo sampling methods [23,35] used in ABC [36], a direction we are currently actively exploring. Active learning approaches, such as those used for sequential refinement of conditional density model training data from SBI studies [5,17], can also be applied to iteratively refine a surrogate model.

Fast and accurate solutions to SIPs are critical in any physical science domain where simulations are routinely used to model populations of observations. In the domain of computational biology, specifically the field of cardiac mechanics, we have used the intervention methods presented here to perform an evaluation of the mechanism of action of various cardiac inotropes, applying our methods to unloaded contraction cell data and a cardiac mechanics cellular model [31]. The methods are also being actively explored for analysis of cardiac and neuronal electrophysiology data as in [37,38].
The r-GAN extends GANs’ applications beyond generative modelling and variational inference [39]. Our first experiment showed that this method solves SIPs for systems of equations with accuracy equalling rejection methods. We also formulated three SIP scenarios that cannot currently be solved using existing methods. First, we demonstrated an improvement in predictions for SIP with stochastic models by incorporating a network for parametrizing noise into the optimization. We showed how alternative methods fail at this task. Second, we formulated a super-resolution imaging problem as a SIP, demonstrating the use of r-GAN with convolutional networks. The rejection sampling algorithm would be prohibitively slow to converge when tackling high-dimensional problems of this type. Instead, using r-GAN we were able to initialize the generator to the prior by means of conventional rejection methods. We also formulated three SIP scenarios that solves SIP within an intervention experiment, as demonstrated previously for a cardiac inotrope [31]. The success of these test experiments leads us to propose that approaches based on our constrained optimization formulation are currently the only path to solving a broad family of related SIPs with equally broad applications.

Ethics. This work did not require ethical approval from a human subject or animal welfare committee.

References

1. Butler T, Jakeman J, Wldye T. 2018 Combining push-forward measures and Bayes’ rule to construct consistent solutions to stochastic inverse problems. SIAM J. Sci. Comput. 40, A984–A1011. doi:10.1137/16M1087229
2. Poole D, Warby AE. 2000 Inference for a Bayesian melding approach. J. Am. Stat. Assoc. 95, 1244–1255. doi:10.1080/01621459.2000.10893424
3. Butler T, Wldye T, Yen TH. 2020 Data-consistent inversion for deterministic simulation models: the Bayesian approach. Inverse Prob. 36, 085015. doi:10.1088/1361-6420/abd883
4. Grammer K, Brehmer J, Lopouge G. 2020 The frontier of simulation-based inference. Proc. Natl Acad. Sci. USA 117, 30 055–30 062. doi:10.1073/pnas.1912789117
5. Goncalves PJ et al. 2020 Training deep neural density estimators to identify mechanistic models of neural dynamics. eLife 9, e56261. doi:10.7554/eLife.56261
6. Lueckmann JM, Boelts J, Greenberg D, Goncalves P, Macke J. 2021 Benchmarking simulation-based inference. In Proc. 24th Int. Conf. on Artificial Intelligence and Statistics (AISTATS) 2021, San Diego, CA, USA, 13–15 April 2021. PMLR, vol. 130.
7. Papamakarios G, Pavlakou T, Murray J. 2017 Masked autoencoder flow for density estimation. In Proc. 31st Conf. on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA, 4–9 December 2017.
8. Papamakarios G, Griswold D, Murray J. 2019 Sequential neural likelihood: fast likelihood-free inference with autoregressive flows. In Proc. 22nd Int. Conf. on Artificial Intelligence and Statistics, Okinawa, Japan, 16–18 April 2019. PMLR, vol. 89.
9. Papamakarios G, Nalisnick E, Rezende DJ, Mohamed S, Lakshminarayanan B. 2021 Normalizing flows for probabilistic modeling and inference. J. Mach. Learn. Res. 22, 2617–2680.
10. Adler J, Öktem O. 2018 Deep Bayesian inversion. (https://arxiv.org/abs/1811.05910)
11. Ramesh P, Lueckmann JM, Boelts J, Goncalves P, Macke JH. 2022 GATSB: generative adversarial training for simulation-based inference. (https://arxiv.org/abs/2203.06481)
12. Micca M, Oudoven S. 2014 Conditional generative adversarial nets. (http://arxiv.org/abs/1411.1784)
13. Mattis S, Steffen K, Butler T, Dawson C, Estep D. 2022 Learning quantities of interest from dynamical systems for observation-consistent inversion. Comput. Methods Appl. Mech. Eng. 388, 114230. doi:10.1016/j.cma.2021.114230
14. Pilsow M, Yen TY, Butler T, Dawson C. 2023 Parameter estimation with maximal updated densities. Comput. Methods Appl. Mech. Eng. 407, 115906. doi:10.1016/j.cma.2023.115906
15. Lawson BA, Drowandi CC, Cusimano N, Burage P, Rodriguez B, Burage K. 2018 Unlocking data sets by calibrating populations of models to data density: a study in atrial electrophysiology. Sci. Adv. 4, e1701676. doi:10.1126/sciadv.1701676
16. Papamakarios G, Murray J. 2016 Fast e×-free inference of simulation models with Bayesian conditional density estimation. (https://arxiv.org/abs/1605.03676)
17. Lueckmann JM, Bassetto G, Karabatsos T, Macke JH. 2019 Likelihood-free inference with emulator networks. (https://arxiv.org/abs/1805.00924)
18. Fengler A, Govindarajan LN, Chen T, Frank MJ. 2020 Likelihood approximation networks (LANs) for fast inference of simulation models in cognitive neuroscience. bioRxiv. doi:10.1101/2020.11.20.392274
19. Radov R, Mertens UK, Voss A, Ardiszone L, Köthe U. 2020 BayesFlow: learning complex
stochastic models with invertible neural networks. (http://arxiv.org/abs/2003.06281)

20. Kingma DP, Salimans T, Jozefowicz R, Chen X, Sutskever I, Welling M. 2016 Improved variational inference with inverse autoregressive flow. In Proc. 30th Conf. on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain, 5–10 December 2016.

21. Huang CW, Dinh L, Courville A. 2020 Augmented normalizing flows: bridging the gap between generative flows and latent variable models. (http://arxiv.org/abs/2002.07101)

22. Goodfellow IJ, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y. 2014 Generative adversarial networks. (https://arxiv.org/abs/1406.2661)

23. Moral PD, Doucet A, Jasra A. 2006 Sequential Monte Carlo samplers. J. R. Stat. Soc. Ser. B 68, 411–436. (doi:10.1111/j.1467-9868.2006.00553.x)

24. Nowozin S, Cseke B, Tomioka R. 2016 f-GAN: training generative neural samplers using variational divergence minimization. In Proc. 30th Conf. on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain, 5–10 December 2016, pp. 271–279.

25. Louppe G, Hermans J, Cranmer K. 2019 Adversarial variational optimization of non-differentiable simulators. In Proc. 22nd Int. Conf. on Artificial Intelligence and Statistics, Okinawa, Japan, 16–18 April 2019. PMLR, vol. 89.

26. Jethava V, Dubhashi D. 2018 ABC-GAN: easy high-dimensional likelihood-free inference. (https://arxiv.org/abs/1711.11139)

27. Breidt J, Butler T, Estep D. 2011 A measure-theoretic computational method for inverse sensitivity problems I: method and analysis. SIAM J. Numer. Anal. 49, 1836–1859. (doi:10.1137/100785946)

28.Ledig C et al. 2017 Photo-realistic single image super-resolution using a generative adversarial network. In 2017 IEEE Conf. on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017, pp. 4681–4690. (doi:10.1109/CVPR.2017.19)

29. Menon S, Damian A, Hu S, Ravi N, Rudin C. 2020 PULSE: Self-supervised photo upsampling via latent space exploration of generative models. In 2020 IEEE/CVF Conf. on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020, pp. 2437–2445. (doi:10.1109/CVPR42600.2020.00251)

30. Borji A. 2019 Pros and cons of GAN evaluation measures. Comput. Vis. Image Underst. 179, 41–65. (doi:10.1016/j.cviu.2018.10.009)

31. Parikh J et al. 2022 Generative adversarial networks for construction of virtual populations of mechanistic model: simulations to study Omecamtiv Mecarbil action. J. Pharmacokinet. Pharmacodyn. 49, 51–64. (doi:10.1007/s10928-021-09787-4)

32. Hamrod KK, Rogers JL, Feinstein JA, Marudan AI, Schiavazzi DE. 2021 Predictive modeling of secondary pulmonary hypertension in left ventricular diastolic dysfunction. Front. Physiol. 12, 66915. (doi:10.3389/fphys.2021.66915)

33. Drovdani C, Cusimano N, Psaltis S, Lawson B, Pettitt A, Burrage P, Burrage K. 2016 Sampling methods for exploring between-subject variability in cardiac electrophysiology experiments. J. R. Soc. Interface 13, 20160214. (doi:10.1098/rsif.2016.0214)

34. Cranko Z, Nock R. 2019 Boosted density estimation remastered. Proc. Mach. Learn. Res. 97, 1416–1425.

35. Doucet A, De Freitas N, Gordon NJ. 2001 Sequential Monte Carlo methods in practice, vol. 1. Berlin, Germany: Springer.

36. Siisson SA, Fan Y, Beaumont MA. 2018 Handbook of approximate Bayesian computation, 1st edn. Boca Raton, FL: CRC Press.

37. Smirnov D, Pikunov A, Syunyaev R, Deviatiiarov R, Gusev O, Aras K, Gams A, Koppel A, Efimov IR. 2020 Genetic algorithm-based personalized models of human cardiac action potential. PLoS ONE 15, e0231695. (doi:10.1371/journal.pone.0231695)

38. Rumbell T, Kozloski J. 2019 Dimensions of control for subthreshold oscillations and spontaneous firing in dopamine neurons. PloS Comput. Biol. 15, e1007375. (doi:10.1371/journal.pcbi.1007375)

39. Hesz L. 2017 Variational inference using implicit distributions. (http://arxiv.org/abs/1702.08235)

40. Rumbell T, Parikh J, Kozloski J, Gurev V. 2023 Novel and flexible parameter estimation methods for data-consistent inversion in mechanistic modeling. Zenodo. (doi:10.5281/zenodo.8125978)

41. Rumbell T, Parikh J, Kozloski J, Gurev V. 2023 Novel and flexible parameter estimation methods for data-consistent inversion in mechanistic modelling. Figspace. (doi:10.6084/m9.figshare.c.6915820)