Inverse sensitivity analysis of mathematical models avoiding the curse of dimensionality

Ben Lambert[1,2]*†, David J. Gavaghan[3], Simon Tavener[4].

1 Department of Zoology, University of Oxford, Oxford, Oxfordshire, U.K.
2 MRC Centre for Outbreak Analysis and Modelling, Infectious Disease Epidemiology, Imperial College London, London W2 1PG, UK.
3 Department of Computer Science, University of Oxford, Oxford, U.K.
4 Department of Statistics, Colorado State University, Fort Collins, Colorado, U.S.A.

*ben.c.lambert@gmail.com. † author order TBC.

1 Abstract

Biological systems often have evolved to be robust to perturbations in their constituent processes. In a wide range of applications, ranging from therapeutics to conservation, it is important to understand which elements of these systems are integral to achieve a given outcome, and the set of conditions under which fluctuations in the system's components do not affect functioning. For the complex systems described by biology, mathematical models have proved key to understanding the mechanisms which underpin their function. Sensitivity analyses of such models provide insight into the responsiveness of the biological systems when experimental manipulation of such systems is difficult. Inverse sensitivity analysis is an emergent field in which a probability distribution over a model's parameter values is sought which results in a given output distribution. Up until now, the computational complexity of the methods used to conduct these analyses has limited their application to models with relatively few parameters. Here we describe a novel Markov Chain Monte Carlo method we call "Contour Monte Carlo" that involves approximating the Jacobian transformation by sampling, which can be used to invert systems with an arbitrary number of parameters. We demonstrate the utility of this method by inverting a number of frequently-used deterministic models of biological systems, including the logistic growth equation and an SIR model of disease transmission with nine input parameters. We argue that the simplicity of our approach means it is amenable to a large class of problems of practical significance and, more generally, provides a probabilistic framework for understanding inversion of deterministic models.

2 Author summary

Inversion is the process of going from a given distribution of model outputs (the "effects") to a distribution over inputs (their "cause"). The process of inversion is well-defined for systems involving randomness and is described by Bayesian inference. However for deterministic systems inversion cannot be handled by the standard Bayesian approach. Here we describe a conceptual framework that describes the inversion of deterministic systems where the dimensionality of inputs typically exceeds that of outputs. Like Bayesian inference our approach uses probability distributions to describe the uncertainty over inputs and outputs. To yield a unique "posterior"

probability distribution over inputs we must specify a prior distribution over input space, but in contrast to the Bayesian approach our method does not require a likelihood. We describe a computational Monte Carlo method that allows efficient sampling from the posterior distribution over inputs. This is a two-step process where we first independently-sample from the priors to estimate the "volume" of contours in input space for each output value; we then use this volume distribution to define a sampling algorithm that yields the requisite input distribution asymptotically. We apply our method to invert a number of deterministic systems, including some with high parameter dimensionality.

3 Introduction

3.1 Inverse sensitivity problems

Simon TBD. What are inverse sensitivity problems? Why are these of interest to (computational) biologists? What are the current approaches?

Forward sensitivity determines the probability distribution of the outputs of a map based on the probability distribution of the inputs of the map. Inverse sensitivity determines the probability distribution of the inputs of a map based on the probability distribution of the outputs of the map.

When considering the laminar flow of a Newtonian fluid, the governing equations (the Navier-Stokes equations) are a system of pdes based on fundamental physical principles and specific assumptions. They involve a small number of parameters that can be independently measured to high precision. Biological models in contrast are often poorly understood and highly parametrized, and further, their parameters cannot be independently measured, but must be inferred from the output of the model. Parameter estimation often seeks a single value through least squares estimation. Inverse sensitivity seeks a probability distribution for the input parameters.

Least squares and Tikhonov regularization for ill-posed problems are probably the most commonly used.

3.2 A one dimensional motivating example

Consider an input (λ) to output (Q) map of the form,

$$Q = \lambda^2, \tag{1}$$

where $\lambda \in [0, 1]$. Suppose that we seek a distribution over inputs $p(\lambda)$ such that the corresponding distribution over outputs follows $Q(\lambda) \sim beta(2, 2)$ (see Figure 1A. Output). A straightforward way to generate samples from $p(\lambda)$ is to first sample $Q_i \sim beta(2, 2)$ then use the inverse map to yield samples of $\lambda_i = Q_i^{1/2}$. For a large class of models of practical interest in biology, including ordinary differential equations (ODEs) and partial differential equations (PDEs), however, an inverse map either cannot be analytically derived or, when the dimension of the inputs exceeds the outputs, does not exist. As such, we aim to generate samples from $p(\lambda)$ using only the forward map.

It may appear that a valid approach to generate input samples from $p(\lambda)$ is to use a Markov chain Monte Carlo (MCMC) algorithm like random walk Metropolis [1], where proposed λ' are accepted as samples if,

$$\frac{p(Q(\lambda'))}{p(Q(\lambda))} > u \sim U(0, 1), \tag{2}$$

where $p(Q(\lambda))$ is the target density (here a $beta(2, 2)$ distribution) evaluated at $Q = \lambda^2$. This approach, however, results in an input distribution (Figure 1A. Input) which, when

transformed, does not recapture the target density (Figure 1A. Output). The reason for this bias is that it neglects to include a Jacobian term in the density which accounts for the nonlinear change of measure in going from input to output space. If this Jacobian term is included in the Metropolis accept-reject rule,

$$\frac{p(Q(\lambda'))}{p(Q(\lambda))}\frac{|J(Q(\lambda'))|}{|J(Q(\lambda))|} > u \sim U(0,1), \tag{3}$$

where, in this case, $J(Q) = \frac{\mathrm{d}Q}{\mathrm{d}\lambda} = 2\sqrt{Q}$, then this results in an input distribution which, when transformed, results in an output distribution that corresponds to the target (Figure 1B. Output).

Indeed, in this simple example, it is possible to exactly determine the input distribution that results in a $beta(2,2)$ target. This is given by a Jacobian transformation multiplied by the target density,

$$\begin{aligned} p(\lambda) &= p(Q(\lambda)) \times |J(Q(\lambda))| && (4)\\ &= 6\lambda^2(1-\lambda^2) \times 2\lambda && (5)\\ &= 12\lambda^3(1-\lambda^2), && (6) \end{aligned}$$

which is shown by the black lines in the left hand column of Figure 1.

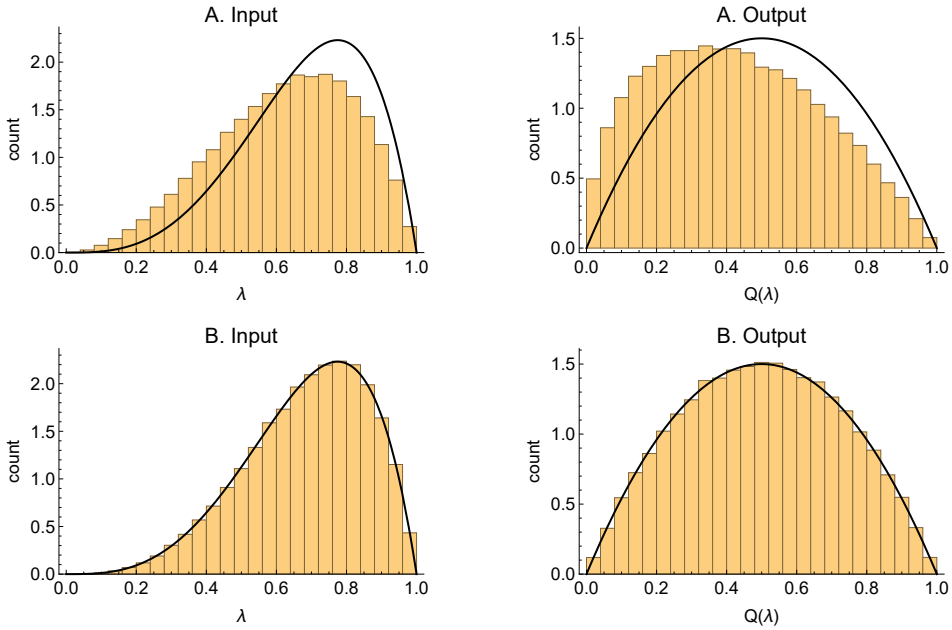

Fig 1. Top: Using the naive Metropolis algorithm as described in the text results in an input distribution (left; orange histogram) which when transformed (right; orange histogram) does not correspond to the target distribution (black line). Bottom: using the Metropolis algorithm including a Jacobian transformation results in an input distribution which maps to the target. In the left-hand column, the analytic input distribution which results in the target density is shown by black lines. In each of the top and bottom rows, 316,000 Metropolis samples across 8 Markov chains (40,000 on each chain, with the first 500 samples discarded as warm-up) were used to produce the histograms.

3.3 Estimating Jacobian transformations by sampling

Whilst the Jacobian may be calculated analytically and the marginalization performed for simple input to output maps, for maps of modest difficulty (for example, those defined by ODEs and PDEs), this term cannot typically be analytically derived. In this section, we introduce a way to approximate Jacobian transformations by sampling. This, hence, allows us to use the Jacobian-corrected Metropolis sampler defined by the acceptance ratio of eq. (3) to generate samples from the input distribution $p(\lambda)$.

The idea underlying our approximation to the Jacobian can be understood by first considering a linear input-to-output map. Here, sampling uniformly across input space results in uniform sampling in output space, because the Jacobian transformation is a constant. In the case of a nonlinear map, such as $Q = \lambda^2$, however, uniform sampling on input space induces nonlinear sampling on output space (Figure 2). What is the analytic form of the sampling distribution in output space? This exactly corresponds with the inverse Jacobian transformation, and is given by $p(Q) = \frac{1}{2\sqrt{Q}}$. In our approach, rather than analytically derive the Jacobian transformation directly, we estimate its density using sample-based estimates. In Figure 2B, we illustrate how Gaussian kernel density estimates of the inverse Jacobian transformation (dashed blue lines) can well approximate the the analytic density with a modest number of samples (grey line).

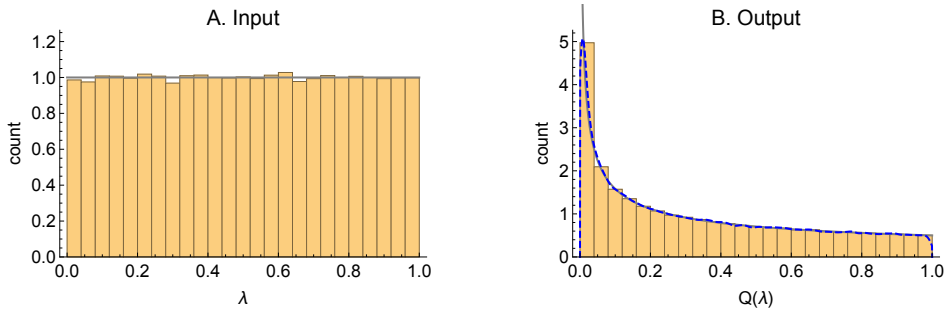

Fig 2. Uniform sampling over a bounded input space (A) induces skewed sampling on output space (B), whenever the input-to-output map is nonlinear, with the output sampling distribution corresponding to a density defined by the inverse Jacobian transformation (grey line). Here the input-to-output map is defined by $Q = \lambda^2$, and 100,000 random samples from a U(0,1) distribution were used to generate the histograms. The grey lines show the actual sampling distribution in each case, and the blue-dotted line in panel B shows Gaussian kernel density estimates of the sampling distribution, using a bandwidth of 0.01.

Once the inverse Jacobian transformation has been estimated, we simply replace the actual Jacobian in our Metropolis-based rule given by eq. (3), with the sample-based estimates $\hat{J}$,

$$\frac{p(Q(\lambda'))}{p(Q(\lambda))} \frac{|\hat{J}(Q(\lambda'))|}{|\hat{J}(Q(\lambda))|} > u \sim U(0,1). \tag{7}$$

Our approach, therefore, consists of two distinct steps: first, independent sampling from the prior space (which is currently assumed to be bounded but, in Section 3.6, this is relaxed) and transformation of these inputs to form a sample of outputs, followed by the fitting of an empirical density estimator to the output samples; second, Jacobian-modified Metropolis sampling using the Jacobian transformations estimated from the first step. When the sample size of the first step approaches infinity, the Markov chain in the second step converges asymptotically to the target density. For a finite sample size for the first step, the sampling distribution of the Markov chain in the

second step does not exactly converge to the target distribution due to Jensen's inequality. For even modest sample sizes, however, we have found this bias is negligible compared to other sources of uncertainty.

In Figure 3, we illustrate how using our two step method allows us to generate an input distribution which maps to the target.

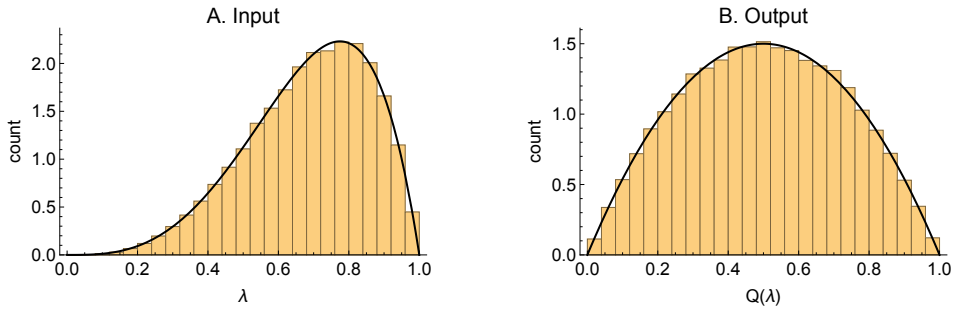

Fig 3. Using kernel density estimates of the Jacobian transformation in Metropolis sampling results in an input distribution (A) which maps to the target distribution (B). Here 316,000 Metropolis samples across 8 Markov chains (40,000 on each chain, with the first 500 samples discarded as warm-up) were used to produce the input distribution. The Jacobian transformations were estimated as indicated in the caption to Figure 2, and are shown as the blue-dashed line in Figure 2B.

3.4 Multi-dimensional input-to-output maps

We next consider a multi-dimensional input-to-output map, and illustrate how this problem can be framed in a Bayesian light. Note that whilst this shift may appear substantive, it amounts to an aesthetic difference compared to the Jacobian approach; the inherent method and motivation remains the same. The only difference is that whilst before we were estimating a one-to-one Jacobian, we must now approximate a many-to-many Jacobian transformation. We do, however, find that the Bayesian framework generalises better to the case when we consider unbounded prior spaces in Section 3.6, and so introduce it now.

Consider a deterministic function from a vector of inputs $\boldsymbol{\Lambda} = (\lambda_1, \lambda_2, ..., \lambda_K) \in \mathbb{R}^K$ to output quantities $\boldsymbol{Q} = (Q_1(\Lambda), Q_2(\Lambda), ..., Q_P(\Lambda)) \in \mathbb{R}^P$, where $P < K$. Since the dimensionality of the input parameters exceeds that of the output quantities, each feasible output value permits a set of possible input causes (Figure 4). This means that without further information it is not possible to determine a single cause for an output value, unless that output value is "singular". That which is true for individual output values is also true for output distributions. For any given output distribution there are a collection of possible input distributions that could yield it.

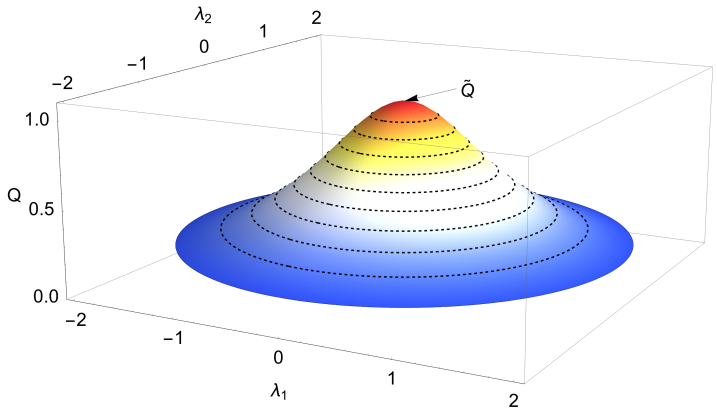

Fig 4. Each value of the function $Q = 1/(1 + \lambda_1^2 + \lambda_2^2)$ maps to any point along two dimensional contours (dashed lines) in input space (λ_1, λ_2), with the exception of singular point $\tilde{Q}$. The (λ_1, λ_2) domain consists of the region bounded by $\lambda_1^2 + \lambda_2^2 \leq 2^2$.

Like in Bayesian inference, to reduce the set of allowable input distributions to one, we specify a prior distribution over input parameters. In contrast to Bayesian inference, this prior is specified as a conditional probability distribution, where we condition on particular output values. A natural way to specify such a prior is to assume that the probability distribution is uniform within the set of possible inputs that generate a given output value. The prior distribution is then given by the probability density,

$$p(\boldsymbol{\Lambda}'|\overline{\boldsymbol{Q}}) = \begin{cases} \frac{1}{\mathcal{V}(\overline{\boldsymbol{Q}})}, & \text{if } Q(\boldsymbol{\Lambda}') = \overline{\boldsymbol{Q}} \\ 0, & \text{otherwise,} \end{cases} \tag{8}$$

where $\mathcal{V}(\overline{\boldsymbol{Q}}) = |\boldsymbol{\Lambda}'|$ is the volume of input space where $Q(\boldsymbol{\Lambda}') = \overline{\boldsymbol{Q}}$. We term the size of the region of input space that produces a particular output value, its "contour volume". Note that we use the term "volume" liberally here. For a two-dimension input space and one dimensional output space as shown in Figure 4 this corresponds to a length, although for higher dimensional examples the set can be an area, a volume or a hypervolume.

The type of uniform prior we have described thus far appears sensible since it allocates probability mass in exact accordance with the geometry of the input-output map. In the absence of further information it seems reasonable to suppose that all inputs which produce a given output value are equally likely causes of it. It is less clear how to analytically calculate such a prior density since, in general, the function $\boldsymbol{Q}$ is nonlinear, and is a function of a parameter vector $\boldsymbol{\Lambda} \in \mathbb{R}^K$ which exists in a high dimensional space. The high dimensionality of the inputs also means that exhaustively generating a numerical input to output map, which can be inverted to yield contour volumes, is prohibitive for all but the simplest problems.

Example 1 An alternative approach is to estimate the volume of contours by Monte Carlo sampling of the inputs. To explain how this process works we consider the case of two dimensional inputs (λ_1, λ_2) bounded by the region $\lambda_1^2 + \lambda_2^2 \leq 2^2$, and a unidimensional output,

$$Q = \frac{1}{1 + \lambda_1^2 + \lambda_2^2}. \tag{9}$$

For each value of Q we would like to determine the size of the set of inputs which map to it. To start, we idealise the problem and assume the function value Q is constant

within annuli of a given width and distance from the origin (Figure 5A). Since our function and input domain are well-behaved it is possible to analytically calculate the area of each annulus (Figure 5B). In most applied cases, however, analytic calculation of the contour volumes is not possible, and we are required to estimate them. To estimate the area of each annulus (the contour volume in this case) we independently sample the inputs uniformly within their bounded domain (Figure 5C). After relatively few samples the shape of the estimated contour volume distribution (Figure 5D) appears indistinguishable from the true one.

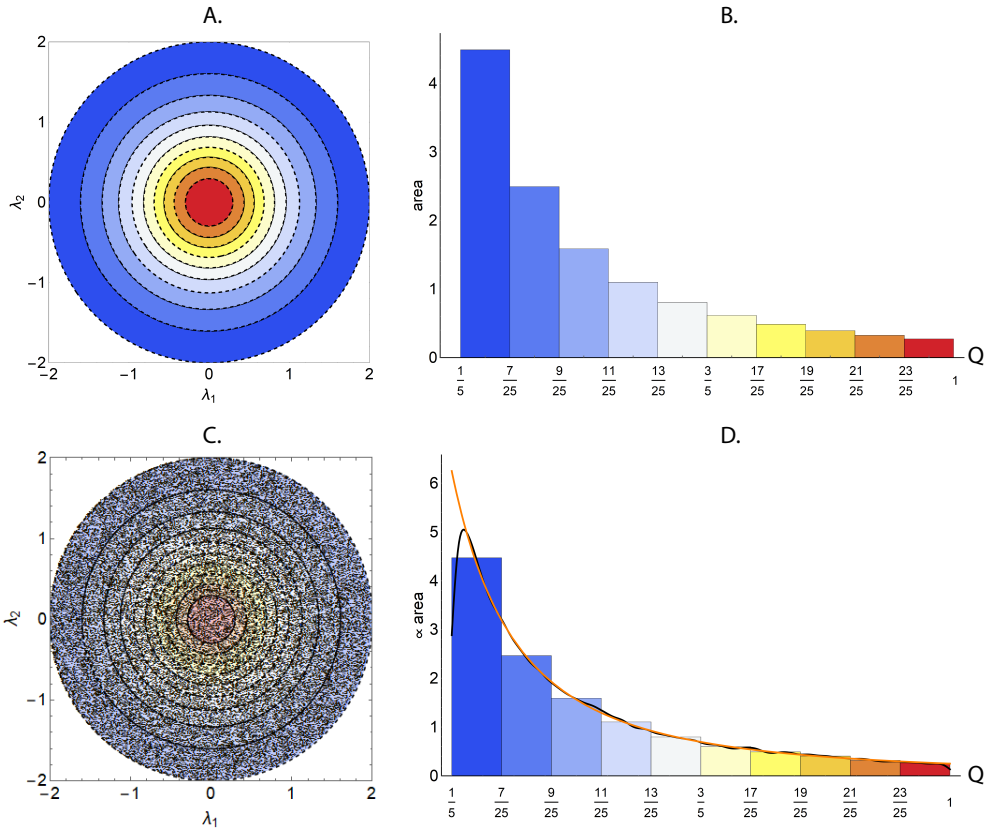

Fig 5. (A) Annuli indicating the mean value of the function $Q = 1/(1 + \lambda_1^2 + \lambda_2^2)$ within them, and the relative area of each annulus (B). Uniform sampling of the input space (C) is used to produce estimates of the contour volumes (D). In C, 50,000 independent samples of the input were used to estimate the contour volume in D. The dashed lines indicate contours where the function value is constant. The black line in D shows a kernel density model fit estimated using the un-binned output data with Gaussian kernel and a bandwidth of 0.01 (using Mathematica's "SmoothKernelDistribution" function [2]), and the orange line indicates the true contour volumes. The (λ_1, λ_2) domain consists of the region bounded by $\lambda_1^2 + \lambda_2^2 \leq 2^2$.

In the above explanation, we assumed that the function value was constant within a given annulus, however we recognise that the continuous nature of the function means that the contours are actually one dimensional lines (Figure 4). Given the simple nature of this output function we can actually analytically determine the length (which we call a contour volume) of each of these lines. The distribution which is uniform over inputs

$p(\lambda_1, \lambda_2) = \frac{1}{4\pi}$ can be transformed to polar coordinate form,

$$p(r, \theta) = \frac{r}{4\pi}, \tag{10}$$

where $r = \lambda_1^2 + \lambda_2^2$ and $\theta = \arctan(\lambda_2/\lambda_1)$. When marginalising over θ the above expression yields $p(r) = r/2$. Since we can rewrite $Q = 1/(1 + r^2)$, a one-dimensional Jacobian transformation yields the contour volume distribution pertaining to $p(r)$,

$$p(Q) = \frac{1}{4Q^2}. \tag{11}$$

In our algorithm we estimate the volume of contours by sampling uniformly from the input space, and for each set of inputs $(\lambda_{1i}, \lambda_{2i})$ we calculate a corresponding Q_i. To approximate $p(Q)$ in eq. (11) we then fit a kernel density model to $(Q_1, Q_2, ..., Q_N)$, where N is the number of input samples (the black line in Figure 5D). After relatively few input samples, the estimated contour volume distribution approximates well the true distribution of eq. (11) (the orange line in Figure 5D).

Figure 5D is analogous to Figure 2B, except for a two-dimensional input example. `SJT: This paragraph needs thought. I have commented out much of the earlier text.`

3.5 The posterior distribution over inputs

The distribution that we want to estimate is a conditional density of the form $p(\boldsymbol{\Lambda}|\boldsymbol{Q}(.), data)$, where $\boldsymbol{Q}(.)$ denotes the particular form of output function used, and *data* denotes a collection of output data points. In analogy to Bayesian inference, we term this conditional density over inputs the "posterior distribution". In what follows we implicitly assume a dependence on $\boldsymbol{Q}(.)$, and omit it for brevity. To sample from this distribution efficiently, we first consider the joint distribution,

$$p(\boldsymbol{\Lambda}, \boldsymbol{Q}|data) = \underbrace{p(\boldsymbol{\Lambda}|\boldsymbol{Q})}_{\text{prior}} \times \underbrace{p(\boldsymbol{Q}|data)}_{\text{target distribution}}, \tag{12}$$

where the *prior* term does not contain any dependence on the data since it solely depends on the geometry of the input-output map, which is determined by $\boldsymbol{Q}(.)$. We assume that the output data distribution has been fitted with an appropriate distribution, to yield a density value for any value of output, $p(\boldsymbol{Q}|data)$, which we term the "target distribution". An alternative way to decompose the joint distribution is,

$$p(\boldsymbol{\Lambda}, \boldsymbol{Q}|data) = p(\boldsymbol{Q}|\boldsymbol{\Lambda}, data) \times \underbrace{p(\boldsymbol{\Lambda}|data)}_{\text{posterior input distribution}}. \tag{13}$$

For a deterministic model $p(\boldsymbol{Q}|\boldsymbol{\Lambda}, data) = p(\boldsymbol{Q}|\boldsymbol{\Lambda})$, since $\boldsymbol{Q}$ is fully determined by the inputs $\boldsymbol{\Lambda}$. Because of the determinacy this term becomes,

$$p(\boldsymbol{Q}|\boldsymbol{\Lambda}) = \begin{cases} 1, & \text{if } \boldsymbol{Q} = \boldsymbol{Q}(\boldsymbol{\Lambda}), \\ 0, & \text{otherwise.} \end{cases} \tag{14}$$

Equating the right hand side of eqns. (12) and (13), we obtain an expression for the posterior input distribution,

$$p(\boldsymbol{\Lambda}|data) = \underbrace{p(\boldsymbol{\Lambda}|\boldsymbol{Q}(\boldsymbol{\Lambda}))}_{\text{prior}} \times \underbrace{p(\boldsymbol{Q}(\boldsymbol{\Lambda})|data)}_{\text{target distribution}}. \tag{15}$$

To sample from the posterior distribution specified by eqn. (15), we use our estimates of the prior that are obtained by independent sampling of the inputs. This results in a (potentially un-normalised) density of the form,

$$p(\boldsymbol{\Lambda}|data) \propto \frac{1}{\widehat{\mathcal{V}}(\boldsymbol{Q}(\boldsymbol{\Lambda}))} \times p(\boldsymbol{Q}(\boldsymbol{\Lambda})|data), \tag{16}$$

where $\widehat{\mathcal{V}}(\boldsymbol{Q})$ is the (potentially un-normalised) contour volume at an output value of $\boldsymbol{Q}$. Since the above probability distribution is non-standard and potentially un-normalised, we use MCMC to sample from it. As for the one-dimensional example introduced in Section 3.2, we use the Random Walk Metropolis sampling [3] for this process. However we recognise that since an approximate form of the un-normalised distribution, as well as its gradients, are known, more efficient forms of MCMC algorithms such as Hamiltonian Monte Carlo (HMC) [4] or No U-Turn Sampling (NUTS) [5] could also be used.

To summarise, our "Contour Monte Carlo" (CMC) algorithm is composed of two distinct phases (see Algorithm 1 for more detail): first sample the input parameters to yield estimates of the contour volume $\widehat{\mathcal{V}}(\boldsymbol{Q}(\boldsymbol{\Lambda}))$; second use $\widehat{\mathcal{V}}$ to specify a prior that distributes probability mass uniformly within a contour volume which, along with a target output distribution, determines a posterior input distribution. This distribution is then sampled from using an MCMC method.

Example 2 To illustrate the workings of the CMC algorithm we return to the function and input domain described in **Example 1**. We suppose that we want to produce an input distribution which results in a target output distribution which is uniform across allowed function values $Q \in [0.2, 1]$. For this example, we can analytically calculate the prior term in eq. (15) using eq. (8) and the normalizing constant equal to the area of the domain Ω $(= 4\pi)$,

$$\begin{aligned} p(\boldsymbol{\Lambda}|\boldsymbol{Q}(\boldsymbol{\Lambda})) &= \frac{1}{\mathcal{V}(Q(\boldsymbol{\Lambda}))}, & (17)\\ &= \frac{1}{4\pi p(Q(\boldsymbol{\Lambda}))}, & (18)\\ &= \frac{Q(\boldsymbol{\Lambda})^2}{\pi}, & (19)\\ &= \frac{1}{\pi(1+\lambda_1^2+\lambda_2^2)^2}. & (20) \end{aligned}$$

Then using eq. (15), and assuming $p(Q|data) = 1/(4/5) = 5/4$, i.e., assuming Q is uniformly distributed over its range, we calculate an analytic form of the posterior input distribution,

$$p(\boldsymbol{\Lambda}|data) = \frac{5}{4\pi(1+\lambda_1^2+\lambda_2^2)^2}, \text{ where } \lambda_1^2+\lambda_2^2 \leq 2^2. \tag{21}$$

This can be alternatively expressed in polar coordinates as,

$$p((r,\theta)|data) = \frac{5r}{4\pi(1+r^2)^2}, \text{ where } r \leq 2 \text{ and } \theta \in [0, 2\pi]. \tag{22}$$

If we marginalise θ from eq. (22), we obtain the marginal posterior distribution for r,

$$p(r|data) = \frac{5r}{2(1+r^2)^2}, \text{ where } r \leq 2. \tag{23}$$

Algorithm 1 Pseudocode for Contour Monte Carlo algorithm applied to bounded parameter spaces with uniform priors. Parameters that may be vectors are shown in bold.

procedure ContourMonteCarlo($p(\boldsymbol{Q}|data)$) ▷ Sample from posterior input distribution
  $\hat{f}(\boldsymbol{Q})$ = ContourVolumeEstimator()
  $(\boldsymbol{\lambda}_0, \boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, ..., \boldsymbol{\lambda}_{N_2})$ = JacobianModifiedMetropolis($\hat{f}, p$)
end procedure

procedure ContourVolumeEstimator( ) ▷ Estimate volume of contours
  for i in $1 : N_1$ **do**
    $\boldsymbol{\lambda}_i \sim U(low, high)^K$ ▷ Sample uniformly within boundaries of $\mathbb{R}^K$ input space
    $\boldsymbol{Q}_i = \boldsymbol{Q}(\boldsymbol{\lambda}_i)$ ▷ Calculate corresponding output value
  end for
  $(\boldsymbol{Q}_1, \boldsymbol{Q}_2, ..., \boldsymbol{Q}_{N_1}) \sim \hat{f}(\boldsymbol{Q})$ ▷ Fit kernel density estimator to output values to estimate Jacobian transformation
  return $\hat{f}(\boldsymbol{Q})$
end procedure

procedure JacobianModifiedMetropolis($\hat{f}, p$) ▷ Sample from posterior input distribution
  $\boldsymbol{\lambda}_0 \sim \pi(.)$ ▷ Sample from arbitrary initialisation distribution
  for i in $1 : N_2$ **do**
    $\boldsymbol{\lambda}'_i \sim \mathcal{N}(\boldsymbol{\lambda}_{i-1}, \boldsymbol{\Sigma})$ ▷ Propose new parameter values for inputs
    $r = \min\left(\frac{\hat{f}(\boldsymbol{Q}(\boldsymbol{\lambda}_i))}{\hat{f}(\boldsymbol{Q}(\boldsymbol{\lambda}'_i))} \frac{p(\boldsymbol{Q}(\boldsymbol{\lambda}'_i)|data)}{p(\boldsymbol{Q}(\boldsymbol{\lambda}_i)|data)}, 1\right)$ ▷ Calculate probability of accepting proposal using eq. (7)
    $u \sim U(0, 1)$ ▷ Sample from uniform distribution
    if $r > u$ **then**
      $\boldsymbol{\lambda}_i = \boldsymbol{\lambda}'_i$ ▷ Accept proposal
    else
      $\boldsymbol{\lambda}_i = \boldsymbol{\lambda}_{i-1}$ ▷ Reject proposal
    end if
  end for
  return $(\boldsymbol{\lambda}_0, \boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, ..., \boldsymbol{\lambda}_{N_2})$
end procedure

In what follows, we compare the estimated marginal posterior distribution for r that results from the CMC algorithm with the analytic result of eq. (23). In the first phase of CMC, we estimated the contour volume for any output value using a kernel density estimator fit to the function evaluations from uniformly sampled inputs (Figure 5D). We then suppose that we wish to sample uniformly from the output space. This probability distribution is specified as a $Q|data \sim u(0.2, 1)$ distribution, since the function is bounded by its extreme values at $\lambda_1^2 + \lambda_2^2 = 2^2$ (where $Q = 0.2$), and $(\lambda_1, \lambda_2) = (0, 0)$ (where $Q = 1$) respectively. In this case, our posterior density is of the simple form,

$$p(\mathbf{\Lambda}|data) \propto \frac{1}{\widehat{\mathcal{V}}(Q(\mathbf{\Lambda}))}, \tag{24}$$

in other words we step in inverse proportion to the estimated contour volumes. This results in a marginal posterior density for inputs which is concentrated towards the centre of the input domain (Figure 6A). A benefit of our algorithm is that we can check that it has worked by comparing the marginal output posterior with the target. Since in this case our output distribution looks approximately uniform (Figure 6B), it appears that the algorithm is working correctly.

Since we are reconstructing the posterior input distribution by MCMC sampling, it is crucial to determine when our sampling distribution has converged. To do so, we use the same convergence diagnostic criteria that are used in applied Bayesian inference. We run a number of Markov Chains in parallel and compare their within-chain variance to that between the chains by calculating $\hat{R}$ [6]. We use a convergence threshold of $\hat{R} <\sim 1.1$ on all input parameters, and use twelve Markov Chains that are initialised at random starting locations. Once convergence is achieved, we discard the first half of the samples as "warmup" and use the second half to form our posterior distribution.

It is important to explain why the naive approach of using MCMC to sample directly from $p(Q(\Lambda)|data)$ does not produce the target distribution. This approach fails because it neglects to account for the differential volume of contours (alternatively, this can be explained as being because a lack of accounting for a Jacobian transformation, by the logic of Section 3.2). In **Example 2**, a sampler using solely a $Q(\Lambda)|data \sim u(0.2, 1)$ distribution over output values will accept all input samples, and hence samples uniformly in input space (Figure 5C). The result is a sampling distribution of inputs with more points towards the boundary of the input domain. Those boundary values of the domain have a lower output value and hence the output distribution is skewed towards those function values (Figure 5D). To correct for the over-representation of points with low function values, we must correct for their longer lengths, which we do by estimating $\widehat{\mathcal{V}}$ and using the inverse of this as a prior.

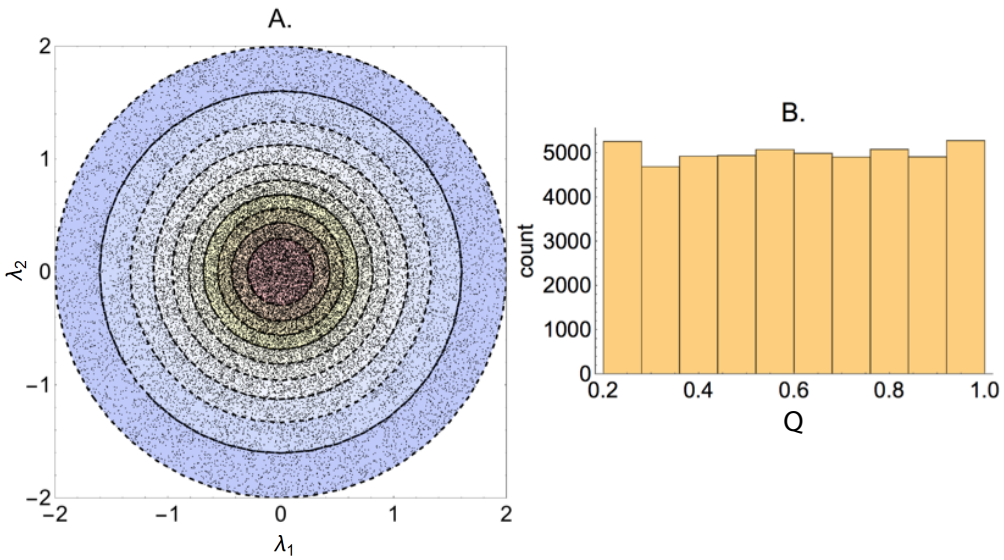

Fig 6. The marginal posterior distribution over inputs (A) and the marginal posterior distribution over outputs (B) for the case described in Example 2. The kernel density estimator for the volume of contours is shown in Figure 5D and resulted from 50,000 independent samples from a uniform distribution over input space. The MCMC was performed using the Metropolis algorithm with toroidal boundaries, and consisted of 50,000 samples. The (λ_1, λ_2) domain consists of the region bounded by $\lambda_1^2 + \lambda_2^2 \leq 2^2$.

Example 3 We use another example to further illustrate the importance of contour volume estimation in CMC. Here we use an input distribution of the form,

$$\begin{pmatrix}\lambda_1\\ \lambda_2\end{pmatrix} \sim \mathcal{N}\left[\begin{pmatrix}+0.75\\ 0\end{pmatrix}, \begin{pmatrix}0.1 & 0\\ 0 & 0.1\end{pmatrix}\right], \tag{25}$$

which we independently sample from to yield a sample of inputs $(\lambda_{1i}, z\lambda_{2i})$, $\forall i = 1, ..., N$ (Figure 7A). For each of the input samples we use eqn. (9) to determine a value, Q_i. We then fit a normal distribution to the resultant sample of outputs (Figure 7B). This output distribution is then used as the target distribution to compare "simple MCMC" (where no attempt is made to correct for contour volumes) and CMC. The simple MCMC approach is biased towards regions near the outside of the domain because of the longer contour volume of these regions (Figure 7C). The result is an output distribution that is skewed towards lower function values (Figure 8A). CMC results in more concentrated sampling towards the centre of the input domain (Figure 7D) and obtains an output distribution that is closer to the target (Figure 8B).

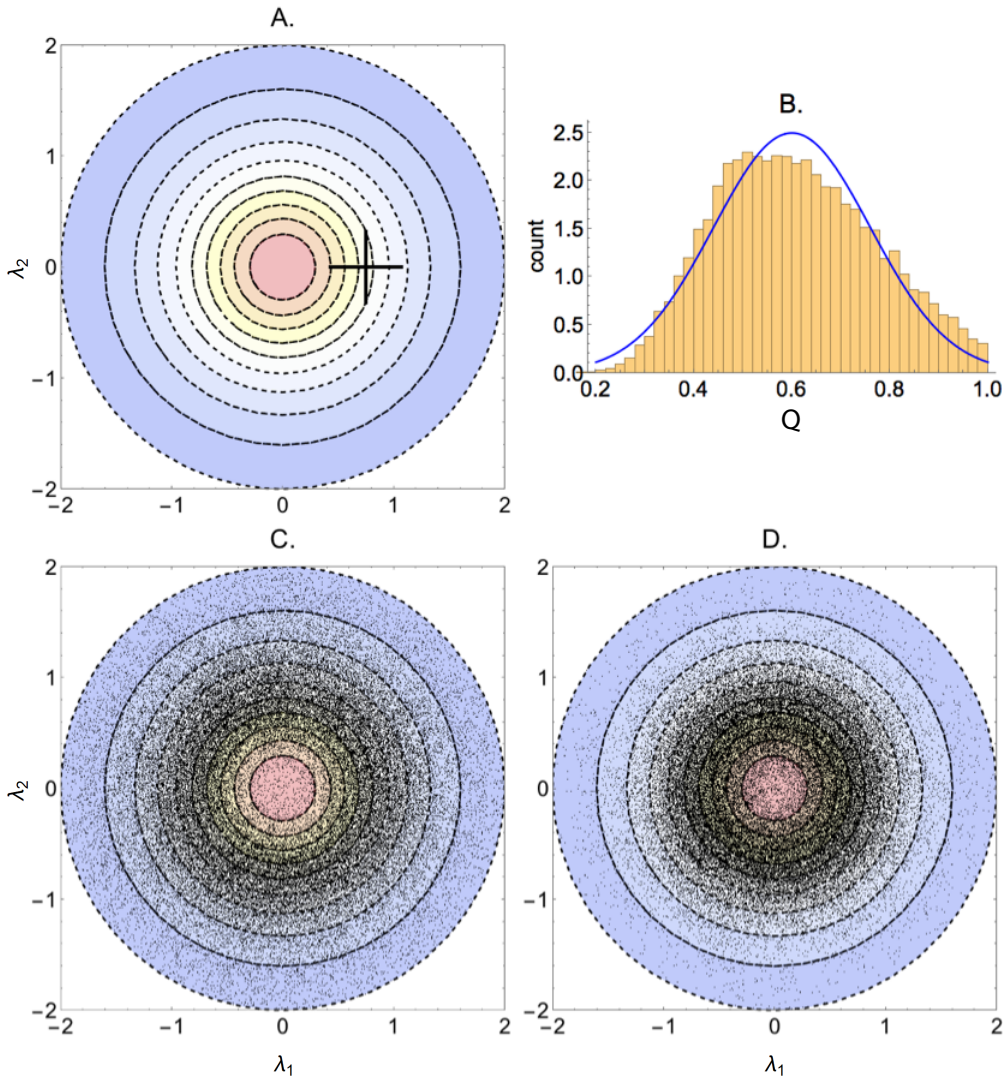

Fig 7. The standard deviations of a bivariate normal distribution (A) used to generate a specific output distribution (B; orange histogram) which is fit using a normal distribution (B; blue line). In C and D resultant posterior samples from simple MCMC and CMC are shown. The kernel density estimator for the volume of contours is shown in Figure 5D and resulted from 50,000 independent samples from a uniform distribution over input space. In both cases the MCMC was performed using the Metropolis algorithm with toroidal boundaries, and consisted of 50,000 samples. The (λ_1, λ_2) domain consists of the region bounded by $\lambda_1^2 + \lambda_2^2 \leq 2^2$.

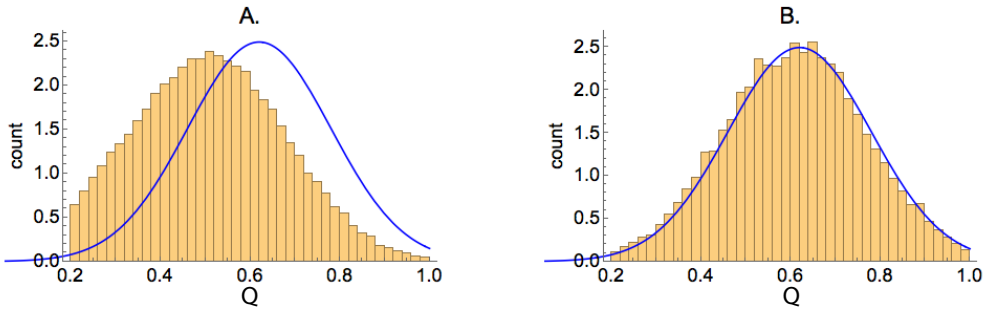

Fig 8. The target (blue lines) and obtained output (orange histogram) distribution when using (A) simple MCMC and (B) CMC. Note the output distribution is truncated at $Q = 0.2$ and $Q = 1$ since these are the minimum and maximum values of the function respectively. In both cases the MCMC was performed using the Metropolis algorithm with toroidal boundaries, and consisted of 50,000 samples.

Example 4 In the previous example we obtained a posterior distribution over inputs (Figure 7D) that was not equal to the original input distribution specified in eqn. (25) that was used to generate the target output distribution (Figure 7B). To illustrate why there is a bias away from the original input distribution we consider another example. Again we use a bivariate normal as an input distribution although this time with a different mean and covariance matrix (Figure 9A),

$$\begin{pmatrix}\lambda_1\\ \lambda_2\end{pmatrix} \sim \mathcal{N}\left[\begin{pmatrix}-1.1\\ 1.0\end{pmatrix}, \quad \begin{pmatrix}0.01 & 0\\ 0 & 0.01\end{pmatrix}\right], \tag{26}$$

As in **Example 3** we apply the function specified by eqn. (9) to a large number of independent samples from the above distribution, and we fit a normal distribution to the output distribution (Figure 9B). We then use this output distribution as a target for CMC. The resultant posterior distribution is concentrated among points that lie within the same annulus in which the bulk of probability mass for the original input distribution lies (Figure 9C). This is because our prior distribution gives equal weight to all points within a given contour. Since the function values in the annulus are similar to those obtained from using the input distribution, we obtain a posterior probability mass that is distributed uniformly around the annulus.

Note that the posterior distribution is *not* the same as the input distribution. This example demonstrates an important characteristic of deterministic models where the input dimension exceeds that of the outputs: a given input distribution induces an output distribution which, when inverted, produces an input distribution that is not necessarily the same as its cause. Information is lost when transforming from the inputs to the outputs that cannot necessarily be regained by inverting the output map.

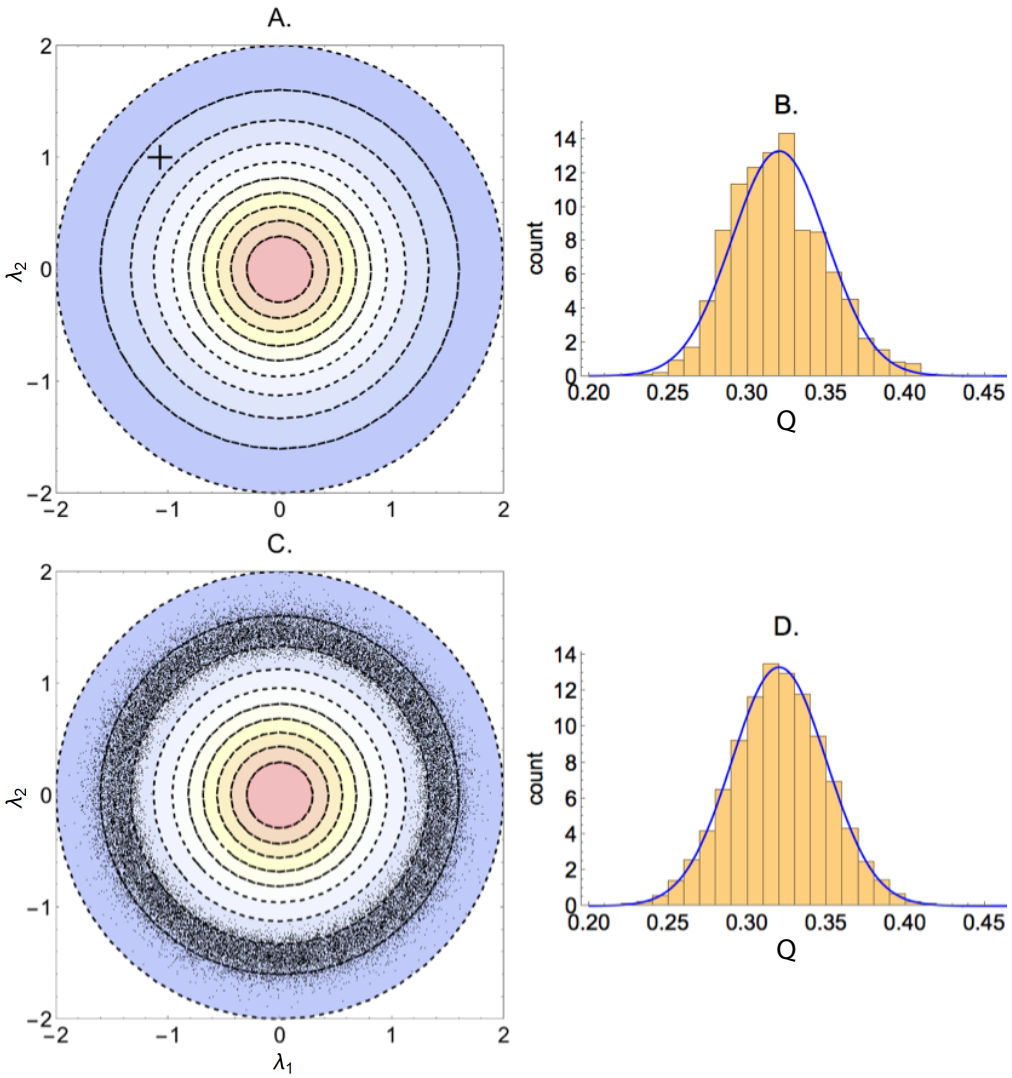

Fig 9. The standard deviations of a bivariate normal distribution (A) used to generate a specific output distribution (B; orange bars) which was fit with a normal distribution (B; blue line). The resultant MCMC samples from the posterior are shown in C along with (D) the output distribution (orange bars) and target distribution (blue lines). The kernel density estimator for the volume of contours is shown in Figure 5D and resulted from 50,000 independent samples from a uniform distribution over input space. The MCMC was performed using the Metropolis algorithm with toroidal boundaries, and consisted of 50,000 samples. The (λ_1, λ_2) domain consists of the region bounded by $\lambda_1^2 + \lambda_2^2 \leq 2^2$.

3.6 General prior distributions

We now describe how the model described in Section 3.4 can be extended to handle arbitrary prior distributions. We start with the expression for the posterior distribution,

$$p(\boldsymbol{\Lambda}|data) = p(\boldsymbol{\Lambda}|\boldsymbol{Q}(\boldsymbol{\Lambda})) \times p(\boldsymbol{Q}(\boldsymbol{\Lambda})|data) \tag{27}$$

$$= \frac{p(\boldsymbol{Q}(\boldsymbol{\Lambda})|\boldsymbol{\Lambda}) \times p(\boldsymbol{\Lambda})}{p(\boldsymbol{Q}(\boldsymbol{\Lambda}))} \times p(\boldsymbol{Q}(\boldsymbol{\Lambda})|data), \tag{28}$$

where we have used Bayes' rule to rewrite the conditional prior term. We then recognise that for a deterministic model $p(\boldsymbol{Q}(\boldsymbol{\Lambda})|\boldsymbol{\Lambda}) = 1$, resulting in the following general expression for the posterior distribution,

$$p(\boldsymbol{\Lambda}|data) = \frac{p(\boldsymbol{\Lambda})}{p(\boldsymbol{Q}(\boldsymbol{\Lambda}))} \times p(\boldsymbol{Q}|data), \tag{29}$$

If the parameter bounds are non-infinite and consist of a total volume V in input space, and we suppose that all values of $\boldsymbol{\Lambda}$ are equally likely *a priori*, then the above becomes,

$$p(\boldsymbol{\Lambda}|data) = \frac{1}{V \times p(\boldsymbol{Q}(\boldsymbol{\Lambda}))} \times p(\boldsymbol{Q}|data). \tag{30}$$

However if all parameter values are equally probable then $p(\boldsymbol{Q}(\boldsymbol{\Lambda}))$ is simply the proportion of input space defined by the set $\{\Lambda' : Q(\Lambda') = Q(\Lambda)\}$. Multiplying this by the overall volume of input space yields the volume of input space occupied by this set, $\mathcal{V}(\boldsymbol{Q}(\Lambda))$, and we recapitulate the expression for the posterior that we approximated in eqn. (16). We thus see that the assumption of a uniform prior within a contour volume is equivalent to the assumption of an uniform prior throughout the input domain.

More generally we can imagine setting informative priors on parameters that may or may not be finitely bounded. For parameters with infinite bounds a uniform prior is improper, and we therefore resort to using valid probability distributions that have support over a infinity of values. In general, the priors that we set will not be aligned with the shape of input contours, meaning that there can be a variation in the probability density within a given contour volume. How should we handle parameter inference in this new framework? As the next example highlights, eqn. (29) contains all the information we need.

Example 5 Suppose that we use the same unidimensional Q function as defined in eqn. (9), with an output distribution as described in **Example 4**, but change the priors now such that they coincide with the input distribution given in eqn. (26). We sample from this prior distribution (Figure 10A) and use each $(\lambda_{1i}, \lambda_{2i})$ pair as inputs to $Q(\lambda_{1i}, \lambda_{2i})$. We then obtain an approximate output distribution by fitting a kernel density estimator to the resultant output values (Figure 10B). Note that because our prior distribution has changed, our prior output distribution $p(\boldsymbol{Q}(\boldsymbol{\Lambda}))$ changes (compare Figure 10B with Figure 5B). We discuss the meaning of this new distribution below but, for now, continue to use it as an input to the Metropolis stage of CMC (see Algorithm 2), where we accept an input proposal with probability,

$$r = \min\left(\frac{\hat{f}(\boldsymbol{Q}(\boldsymbol{\Lambda}))}{\hat{f}(\boldsymbol{Q}(\boldsymbol{\Lambda}'))} \frac{p(\boldsymbol{\Lambda}')}{p(\boldsymbol{\Lambda})} \frac{p(\boldsymbol{Q}'|data)}{p(\boldsymbol{Q}|data)}, 1\right). \tag{31}$$

Here $\hat{f}(\boldsymbol{Q}(\boldsymbol{\Lambda}))$ indicate the prior probabilities of each output value which estimated by fitting kernel density estimators to samples from the prior output distribution (Figure 10B). The posterior distribution that results is located near to the prior distribution (Figure 10C). Comparing this with the distribution that resulted from using a uniform prior (Figure 9C), we see that more informative priors allow us to identify the input parameter distribution. However in both cases the output distribution matched the target (Figure 9D and Figure 10D).

When uniform priors are specified over inputs, the estimated output distribution $\hat{f}(\boldsymbol{Q})$ represents estimates of the proportion of total input volume that map to a value of $\boldsymbol{Q}$. However with non-uniform priors this is no longer true. In this case, we interpret this estimated distribution as representing our *a priori* output distribution that results

Algorithm 2 Pseudocode for Contour Monte Carlo algorithm applied to potentially unbounded parameter spaces with general prior distributions. Parameters that may be vectors are shown in bold.

procedure GeneralContourMonteCarloGeneralPrior($p(Q|data)$) ▷ Sample from posterior input distribution
  $\hat{f}(\boldsymbol{Q})$ = GeneralContourVolumeEstimator()
  $(\boldsymbol{\lambda}_0, \boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, ..., \boldsymbol{\lambda}_{N_2})$ = GeneralJacobianModifiedMetropolis($\hat{f}, p$)
end procedure

procedure GeneralContourVolumeEstimator( ) ▷ Estimate volume of contours
  for i in $1 : N_1$ **do**
    $\boldsymbol{\lambda}_i \sim p(\boldsymbol{\lambda})$ ▷ Sample from prior density
    $\boldsymbol{Q}_i = \boldsymbol{Q}(\boldsymbol{\lambda}_i)$ ▷ Calculate corresponding output value
  end for
  $(\boldsymbol{Q}_1, \boldsymbol{Q}_2, ..., \boldsymbol{Q}_{N_1}) \sim \hat{f}(\boldsymbol{Q})$ ▷ Fit kernel density estimator to output values
  return $\hat{f}(\boldsymbol{Q})$
end procedure

procedure GeneralJacobianModifiedMetropolis($\hat{f}, p$) ▷ Sample from posterior input distribution
  $\boldsymbol{\lambda}_0 \sim \pi(.)$ ▷ Sample from arbitrary initialisation distribution
  for i in $1 : N_2$ **do**
    $\boldsymbol{\lambda}'_i \sim \mathcal{N}(\boldsymbol{\lambda}_{i-1}, \boldsymbol{\Sigma})$ ▷ Propose new parameter values for inputs
    $r = \min\left(\frac{\hat{f}(\boldsymbol{Q}(\boldsymbol{\lambda}_i))}{\hat{f}(\boldsymbol{Q}(\boldsymbol{\lambda}'_i))} \frac{p(\boldsymbol{\lambda}')}{p(\boldsymbol{\lambda})} \frac{p(\boldsymbol{Q}(\boldsymbol{\lambda}'_i)|data)}{p(\boldsymbol{Q}(\boldsymbol{\lambda}_i)|data)}, 1\right)$ ▷ Calculate probability of accepting proposal using eq. (31)
    $u \sim U(0, 1)$ ▷ Sample from uniform distribution
    if $r > u$ **then**
      $\boldsymbol{\lambda}_i = \boldsymbol{\lambda}'_i$ ▷ Accept proposal
    else
      $\boldsymbol{\lambda}_i = \boldsymbol{\lambda}_{i-1}$ ▷ Reject proposal
    end if
  end for
  return $(\boldsymbol{\lambda}_0, \boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, ..., \boldsymbol{\lambda}_{N_2})$
end procedure

from our prior weighting over inputs. Here $\hat{f}(\boldsymbol{Q})$ is the estimated probability of obtaining an output $\boldsymbol{Q}$ given our input priors.

Why does using the distribution given in eqn. (29) ensure that the sampled output distribution matches the target? If we attempt to sample from a uniform target distribution we obtain,

$$p(\boldsymbol{\Lambda}|data) = \frac{p(\boldsymbol{\Lambda})}{p(\boldsymbol{Q}(\boldsymbol{\Lambda}))}. \tag{32}$$

Where, by construction, sampling from the above input distribution results in a uniform output distribution. However the terms on the right hand side of eqn. (32) are always present in the case of a more general target distribution. Since they induce a uniform output distribution when this term is multiplied this by an arbitrary target distribution this results in a distribution whose shape is the same as the target. In this sense we are constrained to always specify a prior output distribution that is uniform so that the posterior output distribution matches the target.

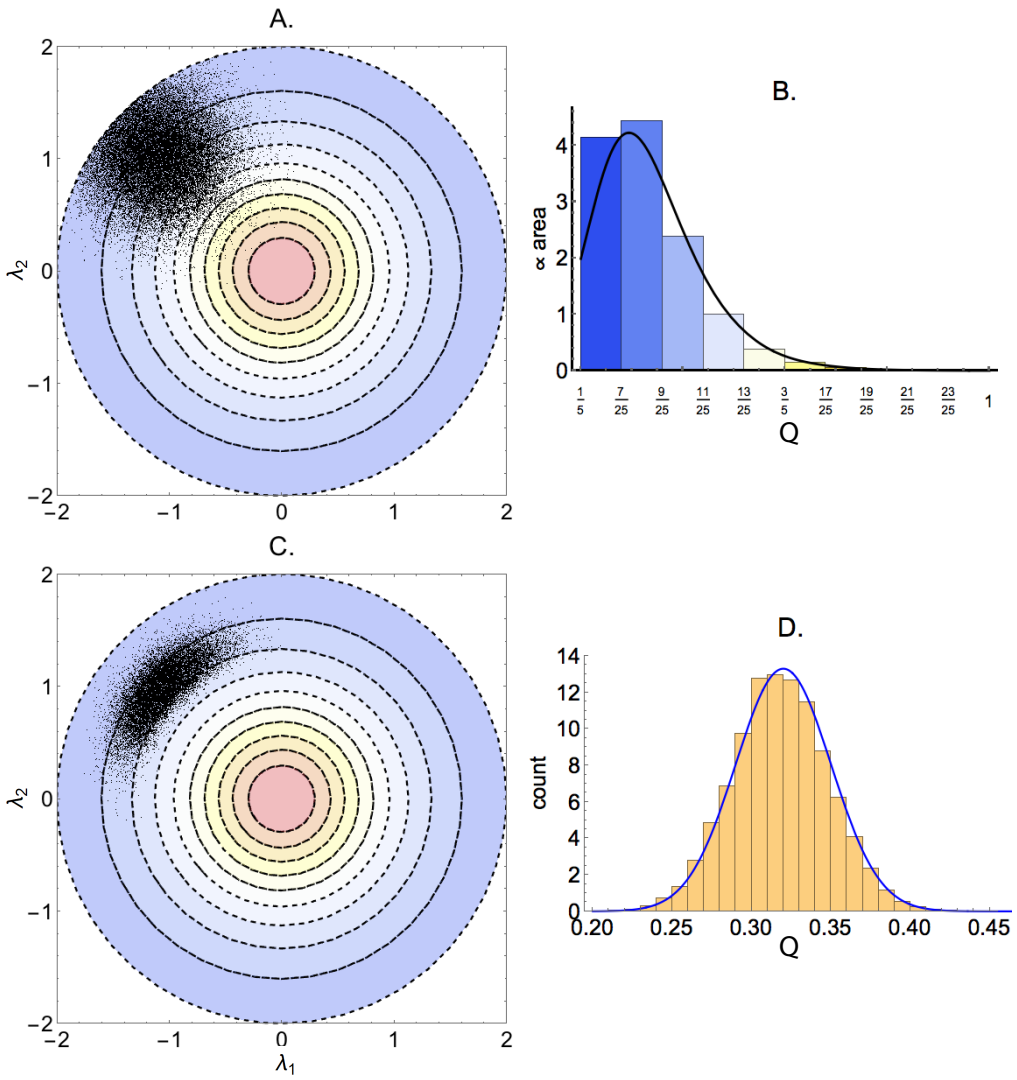

Fig 10. (A) Samples from the prior distribution specified by eqn. (26) with (B) the resultant output prior distribution (coloured bars) and kernel density estimates (black line). (C) shows the posterior distribution over inputs and (D) shows the output distribution (orange bars) along with the target (blue lines). A Gaussian kernel density estimator with bandwidth 0.05 was fit to the univariate output distribution resultant from 50,000 samples from the bivariate normal prior (using Mathematica's "SmoothKernelDistribution" function [2]). To obtain the posterior distribution another round of MCMC was performed using the Metropolis algorithm with toroidal boundaries, and consisted of 50,000 samples. The (λ_1, λ_2) domain consists of the region bounded by $\lambda_1^2 + \lambda_2^2 \leq 2^2$.

4 Results

4.1 A nonlinear map with a bivariate output distribution

We now demonstrate the efficacy of CMC by applying it to a number of examples. The first of these is the pair of nonlinear maps introduced in [7],

$$Q_1 = \lambda_1^2 + \lambda_2^2 + \lambda_3^2, \tag{33}$$

$$Q_2 = arccos(\lambda_3/\sqrt{\lambda_1^2 + \lambda_2^2 + \lambda_3^2}), \tag{34}$$

where there is a three dimensional input space parameterised by $(\lambda_1, \lambda_2, \lambda_3)$. We assume that each of the input parameters is uniformly distributed across the unit interval. In the first stage of CMC, we estimate the contour volumes by sampling uniformly in input space and using the input samples to calculate a corresponding output distribution. This output distribution is then fit with a kernel density estimator to yield estimates of the contour volume at each output value (Figure S1).

In the second stage of CMC, we target the following output distribution,

$$Q_1 \sim \mathcal{N}(0.75, 0.025)\,. \tag{35}$$

The resultant posterior distribution recapitulates the result of the original paper and consists of symmetric marginals across each pair of inputs (top row of Figure 11). The input parameters are not identified since contours of Q_1 correspond to the surface of a sphere centred at the origin. The posterior probability mass is, hence, distributed uniformly across shells of the sphere, and does not uniquely identify individual parameters. To check that CMC had converged on an input distribution that results in the target distribution, we calculated a value of Q_1 for each of the input samples, which resulted in a distribution that was in good agreement with the target (S2A).

We next supposed that we had a target distribution for Q_2,

$$Q_2 \sim \mathcal{N}(0.8, 0.01)\,. \tag{36}$$

Now the posterior distribution was in good agreement with the results of the original paper, and exhibits a symmetry between λ_1 and λ_2 (middle row Figure 11). This is because exchanging these two terms in the expression for Q_2 has no effect on the output. As for the previous example, the resultant output distribution was consistent with the target (Figure 6B).

We finally specify a joint distribution on (Q_1, Q_2) which is used to estimate a posterior input distribution,

$$\begin{pmatrix} Q_1 \\ Q_2 \end{pmatrix} \sim \mathcal{N}\left[\begin{pmatrix} 0.75 \\ 0.80 \end{pmatrix}, \quad \begin{pmatrix} 0.025 & 0 \\ 0 & 0.01 \end{pmatrix}\right], \tag{37}$$

where we note that the marginal distributions for each output remain the same as before. The posterior distribution in this case allows for identification of λ_3, although due to the exchangeability of λ_1 and λ_2 these parameters cannot be identified. The output distribution in this case was again in good correspondence with the target (Figure S2D).

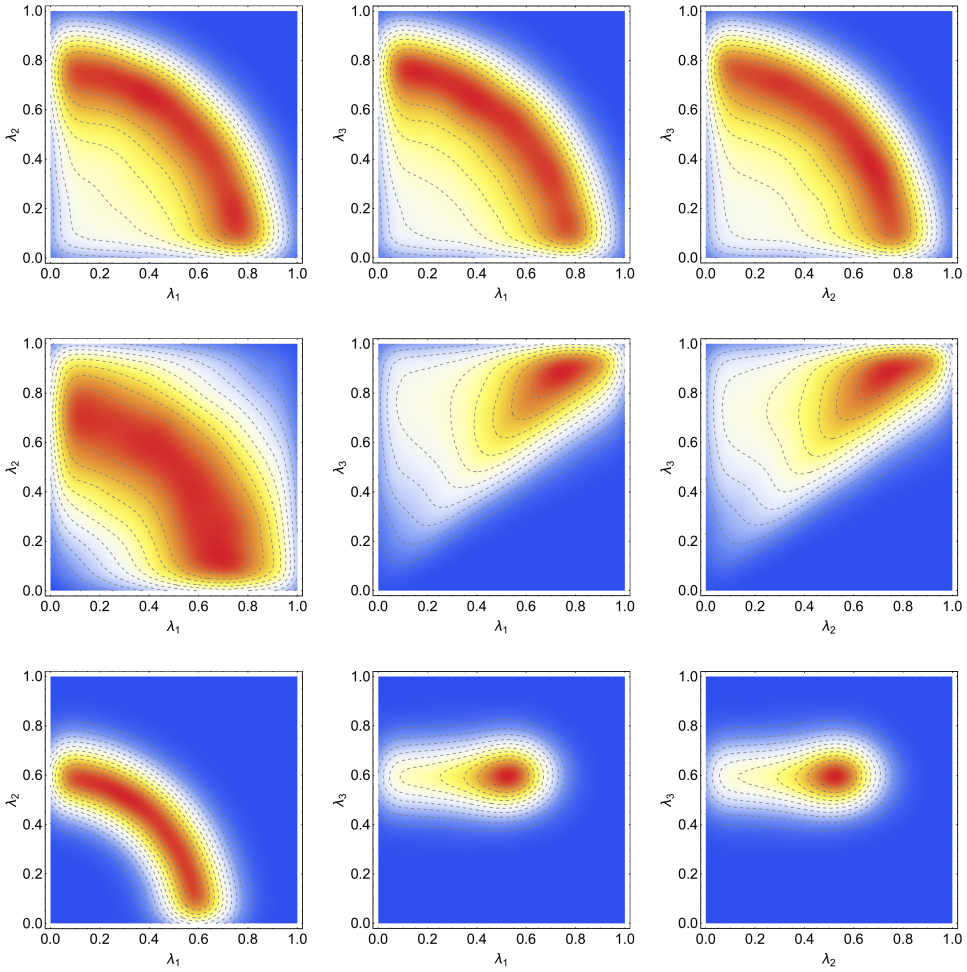

Fig 11. Marginal densities for $\lambda_1\lambda_2$ (left column), $\lambda_1\lambda_3$ (middle), and $\lambda_2\lambda_3$ (right) obtained by inverting using distributions for $Q_1 = \lambda_1^2 + \lambda_2^2 + \lambda_3^2$ (top), $Q_2 = \mathbf{arccos}(\lambda_3/\sqrt{\lambda_1^2 + \lambda_2^2 + \lambda_3^2})$ (middle) and both Q_1 & Q_2 (bottom). The distributions of Q_1 and Q_2 are described in the text. In all cases the priors specified were $\lambda_i \sim U(0, 1)$ for $i = 1, 2, 3$. The contour volumes were estimated using a two-dimensional Gaussian kernel density estimator with bandwidth 0.05 fit to 100,000 independent samples from the prior distributions (using Matlab's "mvksdensity" function [8]), and the posteriors were estimated using 1.2m MCMC iterations.

4.2 Logistic growth

We next consider the logistic growth equation,

$$\frac{dy(t)}{dt} = ry(t)(1 - \frac{y(t)}{\kappa}), \tag{38}$$

where $y(t)$ represents the density of individuals at time t in a population that experiences competition for resources, for example, a bacterial colony confined to a Petri dish. Here r represents the initial (exponential) growth rate of the colony and κ is the carrying capacity. We first suppose that our output function $Q_1 = y(8)$ is univariate, and corresponds to the solution to the logistic equation at time step $t = 8$. An output distribution was generated by calculating the output value for each pair of

samples from the following independent input distributions,

$$r \sim \mathcal{N}(0.5, 0.05)\,, \tag{39}$$
$$\kappa \sim \mathcal{N}(10, 0.5)\,, \tag{40}$$

where we assumed that the initial population size was $y_0 = 0.1$. The resultant output distribution was fit using a normal distribution and specified as the target. We assumed that the priors for the input parameters were of the form, $r \sim U(0, 2)$ and $\kappa \sim U(5, 15)$. The resultant posterior distribution in this case indicated that the growth rate r was well identified, whereas the carrying capacity was not (Figure 12A).

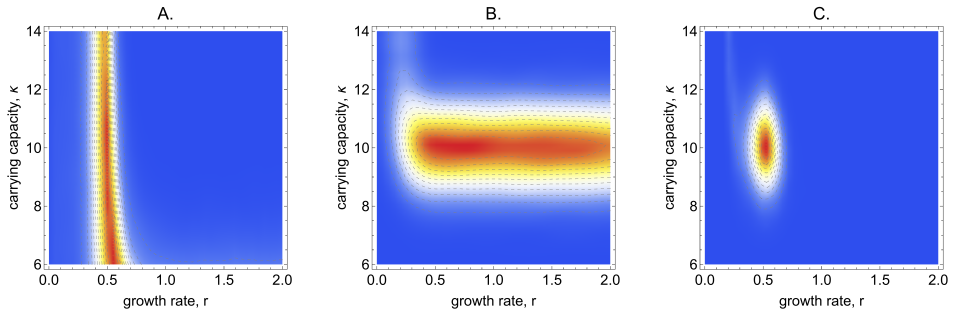

Fig 12. The joint distribution of the growth rate and carrying capacity as a result of sampling the population size at early (A), late (B) and both (C) times. The early & late times corresponded to time steps $t = 8$ & $t = 29$ respectively, with an initial population size $y_0 = 0.1$. The output distributions are described in the text and were chosen to correspond to a mean of $(r, \kappa) = (0.5, 10)$. In all cases the priors specified were $r \sim U(0, 2)$ and $\kappa \sim U(5, 15)$. The contour volumes were estimated using a two-dimensional Gaussian kernel density estimator with default bandwidth fit to 100,000 independent samples from the prior distributions (using Matlab's "mvksdensity" function [8]), and the posteriors were estimated using 1.2m MCMC iterations.

To explain this pattern of identification we calculate the elasticity of Q_1 with respect to each parameters (Figure 13) using the method described in XXXX (preprint of "Geometric assessment of parameter identiability in nonlinear biological models"). The resultant posterior distribution demonstrates identification for the growth rate parameter because during early times (for example, at $t = 8$) the population size is sensitive to the growth rate parameter. By contrast the population size at this time is insensitive to the carrying capacity since competition for resources has yet to predominate, and we are unable to identify this parameter.

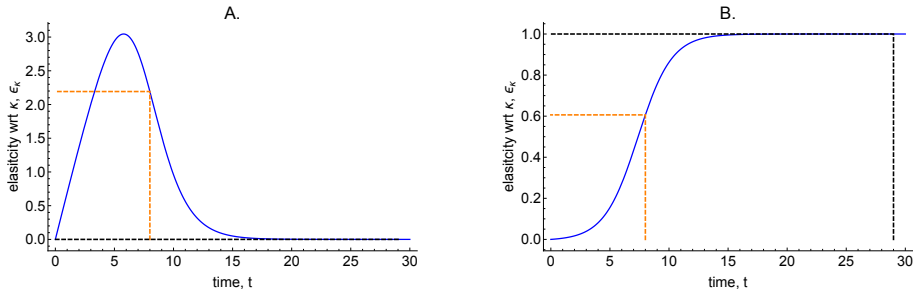

Fig 13. The elasticities of the solution to the logistic equation with respect to (A) the growth rate r, and (B) the carrying capacity, κ. The dashed orange and black lines indicate the elasticities for $Q_1 = y(8)$ and $Q_2 = y(29)$ respectively. These curves were generated by assuming $y_0 = 0.1$, $r = 0.5$ and $\kappa = 10$.

We next use $Q_2 = y(29)$ as the output summary statistic, and use the same input distribution to generate a target output distribution. We then use CMC to generate a posterior input distribution (Figure 12B). This distribution indicates that the carrying capacity is now identified, whereas the growth rate is no longer so. This is because at $t = 29$ the population size is elastic with respect to the carrying capacity but inelastic with respect to the growth rate.

We finally use the target distribution of (Q_1, Q_2) which was generated by fitting a bivariate normal to samples from the joint output distribution which was generated by the same input distribution. The posterior in this case indicated that both input parameters were well-identified (Figure 12C). This is because by combining the output summary statistics from the two previous cases we achieve an output distribution that is sensitive with respect to each of the parameters. Note that in this case the posterior distribution ascribes probability mass to a region of parameter space that is similar to the original (causative) input distribution. As we discussed in the Methods section this is a special case, and not typical with input-output maps. This is because in the univariate output cases the estimates of the individual parameters are not correlated with one another, and lead to unbiased identification of r (for Q_1) and κ (for Q_2) respectively. These constraints ensure that using the joint output distribution leads to unbiased estimation of both parameters.

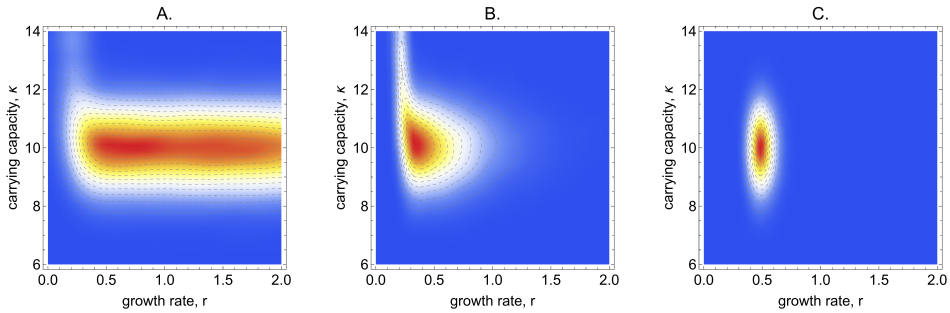

Fig 14. The effect of reducing prior variance for the growth rate ($A. \to C.$) on parameter identification in the logistic model. In all cases the population size was sampled at time step $t = 29$, with a priors on the growth rate given by $r \sim U(0, 2)$ (A), $r \sim \Gamma(2.5, 0.2)$ (B) and $r \sim \Gamma(40, 0.0125)$ (C). The output distributions are described in the text and were chosen to correspond to a mean of $(r, \kappa) = (0.5, 10)$. In all cases the prior specified on the carrying capacity was $\kappa \sim U(5, 15)$. The contour volumes were estimated using a two-dimensional Gaussian kernel density estimator with bandwidth 0.05 fit to 100,000 independent samples from the prior distributions (using Matlab's "mvksdensity" function [8]) , and the posteriors were estimated using 1.2m iterations of the Metropolis-Hastings algorithm.

We now use the logistic example to illustrate the use of informative priors over the input parameters. In this case we use $Q_2 = y(29)$ as our univariate output summary statistic. If we use uniform priors the posterior distribution (Figures 12C and 14A) indicates that although we can identify the carrying capacity we cannot identify the growth rate. If we instead use a $r \sim \Gamma(2.5, 0.2)$ prior distribution that allocates probability mass around $r = 0.5$ the resultant posterior narrows with respect to this parameter (Figure 14B). If we use the narrower still $r \sim \Gamma(40, 0.0125)$ prior distribution, we strongly identify both parameters (Figure 14D).

Like in Bayesian statistics, there are two ways to identify a parameter: either we collect more data (here by specifying another summary statistic), or we use prior information. In Bayesian statistics, it is the stochasticity of the input-to-output map that generates the uncertainty, meaning that the output distribution (the "data") can be generated by a collection of possible parameter distributions. In deterministic input-output maps, where the dimension of the inputs exceeds the outputs, it is the multiplicity of the mapping that introduces uncertainty. In both cases, a prior distribution must be specified to ensure that the resultant posterior is unique, and a valid probability distribution.

4.3 SIR model

The next example we use is an SIR model of disease transmission incorporating a carrying capacity for the population [9], which is described by the following system of ODEs,

$$\frac{dS}{dt} = rN(t)\left(1 - \frac{N(t)}{\kappa}\right) - \beta S(t)I(t) - \mu S(t) \tag{41}$$

$$\frac{dI}{dt} = \beta S(t)I(t) - \gamma I(t) - \mu I(t) - \nu I(t) \tag{42}$$

$$\frac{dR}{dt} = \gamma I(t) - \mu R(t), \tag{43}$$

where $N(t) = S(t) + I(t) + R(t)$ is the total population size at time t. In what follows, we also assume that we are uncertain about the initial conditions (S_0, I_0, R_0), meaning that there are nine uncertain parameters.

We start by assuming uniform prior distributions on each of the nine parameters in this system (bounds are those on the axes in Figure 15). We use a three dimensional output summary statistic in which the elements consists of sampling the infected population at $t = 10$ and the recovered population at times $t = 10, 80$. For our target distribution, we choose the following multivariate normal,

$$\begin{pmatrix} I(10) \\ R(10) \\ R(80) \end{pmatrix} \sim \mathcal{N}\left[\begin{pmatrix} 500 \\ 300 \\ 500 \end{pmatrix}, \begin{pmatrix} 400 & 0 & 0 \\ 0 & 100 & 0 \\ 0 & 0 & 200 \end{pmatrix} \right]. \tag{44}$$

The posterior distribution over inputs indicates identification of the parameter r, which dictates the rate of initial exponential growth for the susceptible population (Figure 15r). The posterior distribution of the carrying capacity term κ, the death rate μ and recovery rate γ were also constrained to subsets of input space (Figure 15κ, 15μ, 15γ). The remaining parameters of the model were unidentified using this summary statistic distribution (Figures 15 and 16).

For the posterior input distribution that resulted, we calculated the corresponding output distribution, and compared this with the marginal target distributions, indicating good agreement (Figure 17).

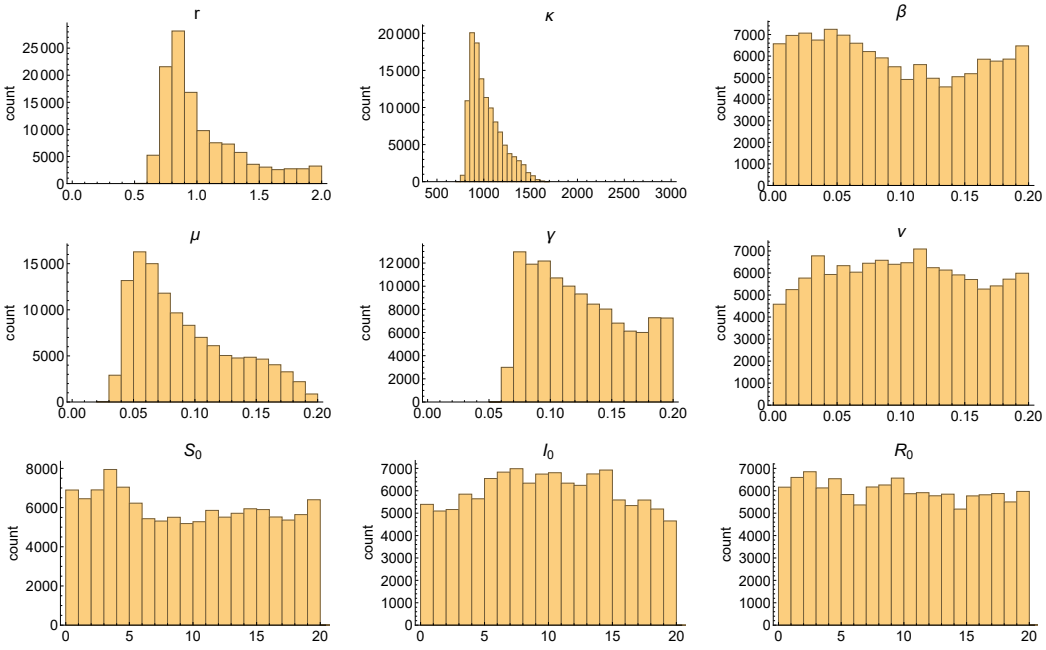

Fig 15. The marginal distribution of parameters for the SIR model. The output distributions are described in the text. The parameters were assigned uniform priors over the ranges shown in the axes. The contour volumes were estimated using a Gaussian kernel density estimator with default bandwidth fit to 100,000 iterations (using the "kde" function in the "ks" package with grid sizes of 300 and maxima of 3000 in each dimension [10]), and the posteriors were estimated using 1.2m MCMC iterations.

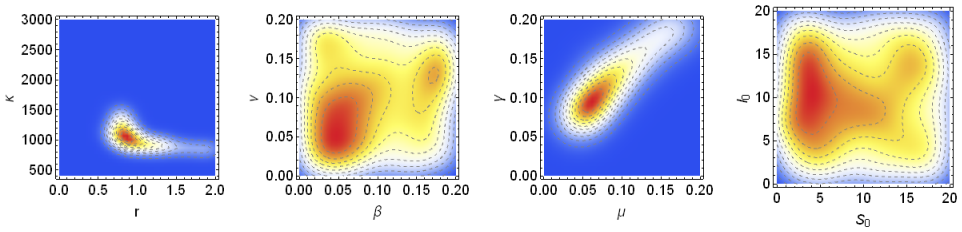

Fig 16. The joint distribution of pairs of parameters for the SIR model. The output distributions are described in the text. The parameters were assigned uniform priors over the ranges shown in the axes. The contour volumes were estimated using a Gaussian kernel density estimator with default bandwidth fit to 100,000 iterations (using the "kde" function in the "ks" package with grid sizes of 300 and maxima of 3000 in each dimension [10]), and the posteriors were estimated using 1.2m MCMC iterations.

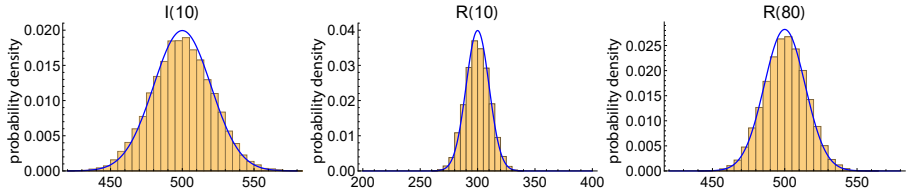

Fig 17. The estimated marginal output distribution (orange histogram) versus the marginal target distribution (blue lines) for $I(10)$ (A), $R(10)$(B) and $R(80)$(C). The target output distributions are described in the text. The parameters were assigned uniform priors over the ranges shown in the axes of Figure 15. The contour volumes were estimated using a Gaussian kernel density estimator with default bandwidth fit to 100,000 iterations (using the "kde" function in the "ks" package with grid sizes of 300 and maxima of 3000 in each dimension [10]), and the posteriors were estimated using 1.2m MCMC iterations.

5 Discussion

Do we need/want this? The only thing I would want to discuss would be potential kernel density estimation approaches for the output dimension. For example, Vine Copulas.

6 Conclusion

Inverse sensitivity analyses are an important element of the model building and validation process. Existing methods for undertaking this process rely on grid-based methods for inverting the input-output map and so experience the curse of dimensionality, meaning these approaches are restricted to models with few parameters. Here we introduce a Monte Carlo method which can be applied to models with an arbitrary number of input parameters, and illustrate its utility for a range of problems of interest to computational biology.

Supporting information

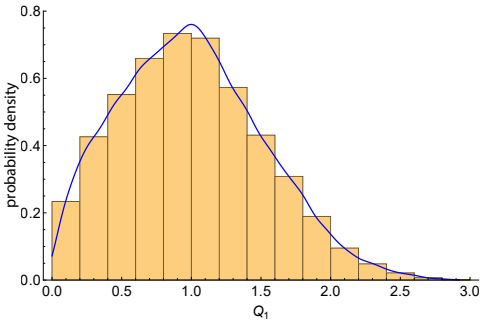

Fig 18. The estimated contour volume (blue line) fit to samples of Q_1 obtained from independent sampling from the input parameter priors. The priors specified were $\lambda_i \sim U(0,1)$ for $i = 1,2,3$. The contour volumes were estimated using a Gaussian kernel density estimator with bandwidth of 0.05 fit to 100,000 iterations.

S1

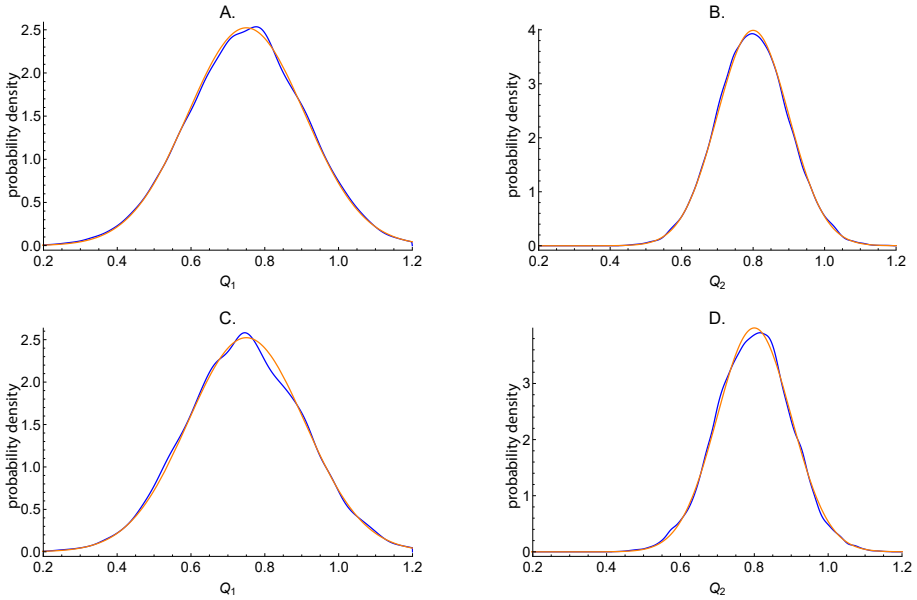

Fig 19. The target output distribution (orange) versus the estimated (blue) for the nonlinear map problem when inverting using Q_1 (A), Q_2 (B) and both (C&D). The distributions of Q_1 and Q_2 are described in the text.

S2

References

1. Metropolis N, Rosenbluth AW, Rosenbluth MN, Teller AH, Teller E. Equation of state calculations by fast computing machines. The Journal of Chemical Physics. 1953;21(6):1087–1092.

2. Wolfram Research, Inc . Mathematica 8.0;. Available from: `https://www.wolfram.com`.

3. Lambert B. A Student's Guide to Bayesian Statistics. Sage Publications Ltd.; 2018.

4. Neal RM, et al. MCMC using Hamiltonian dynamics. Handbook of Markov Chain Monte Carlo. 2011;2:113–162.

5. Hoffman MD, Gelman A. The No-U-turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo. Journal of Machine Learning Research. 2014;15(1):1593–1623.

6. Gelman A, Rubin DB. Inference from iterative simulation using multiple sequences. Statistical Science. 1992; p. 457–472.

7. Butler T, Estep D, Tavener S, Dawson C, Westerink J. A measure-theoretic computational method for inverse sensitivity problems III: Multiple quantities of interest. SIAM/ASA Journal on Uncertainty Quantification. 2014;2(1):174–202.

8. MATLAB version 9.0.0.341360 (R2016a); 2016.

9. Tavener S, Mikucki M, Field SG, Antolin MF. Transient sensitivity analysis for nonlinear population models. Methods in Ecology and Evolution. 2011;2(5):560–575.

10. Duong T, Duong MT. Package ks. 2018;.