# Robust Optimisation Monte Carlo

**Borislav Ikonomov**
School of Informatics, University of Edinburgh

**Michael U. Gutmann**
School of Informatics, University of Edinburgh

## Abstract

This paper is on Bayesian inference for parametric statistical models that are defined by a stochastic simulator which specifies how data is generated. Exact sampling is then possible but evaluating the likelihood function is typically prohibitively expensive. Approximate Bayesian Computation (ABC) is a framework to perform approximate inference in such situations. While basic ABC algorithms are widely applicable, they are notoriously slow and much research has focused on increasing their efficiency. Optimisation Monte Carlo (OMC) has recently been proposed as an efficient and embarrassingly parallel method that leverages optimisation to accelerate the inference. In this paper, we demonstrate an important previously unrecognised failure mode of OMC: It generates strongly overconfident approximations by collapsing regions of similar or near-constant likelihood into a single point. We propose an efficient, robust generalisation of OMC that corrects this. It makes fewer assumptions, retains the main benefits of OMC, and can be performed either as post-processing to OMC or as a stand-alone computation. We demonstrate the effectiveness of the proposed Robust OMC on toy examples and tasks in inverse-graphics where we perform Bayesian inference with a complex image renderer.

## 1 INTRODUCTION

Simulator-based models can describe many complex processes that occur in nature, such as the evolution of genomes (Marttinen et al., 2015) or the dynamics of gene regulation (Toni et al., 2009). Learning their parameters, in particular when done in a Bayesian framework, allows us to make predictions or take decisions based on incomplete information. However, learning the parameters or obtaining their posterior distribution is typically computationally very demanding as their likelihood functions are intractable. Likelihood-Free Inference (LFI) methods have thus emerged that perform inference when the likelihood function is not available in closed form but sampling from the model is possible.

A prominent instance of LFI is Approximate Bayesian Computation (ABC); for recent reviews, see for example (Sisson et al., 2018; Lintusaari et al., 2017). Other instances of LFI are the synthetic likelihood approach by Wood (2010) and its generalisations (Thomas et al., 2016; Price et al., 2017; Fasiolo et al., 2018). This paper focuses on ABC where the basic idea is to identify the parameter values which generate synthetic data that is close to the observed data under some chosen discrepancy measure. This measure can be the Euclidean distance between suitably chosen summary statistics, but other measures are possible too (e.g. Gutmann et al., 2014; Bernton et al., 2018). Generally, there are two main avenues of research for ABC — one focuses on improving the distance metric and/or the summary statistics used (e.g. Aeschbacher et al., 2012; Fearnhead and Prangle, 2012), while the other concentrates on computational efficiency (e.g. Beaumont et al., 2002; Blum et al., 2013; Meeds and Welling, 2015; Gutmann and Corander, 2016; Papamakarios and Murray, 2016). This paper focuses on the latter, and as such we assume the distance and summary statistics are given.

The primary focus of this paper is Optimisation Monte Carlo (OMC) — an ABC method developed by Meeds and Welling (2015) and also independently by Forneron and Ng (2016, 2018) under the name of "the reverse sampler". It uses optimisation to efficiently produce weighted posterior samples in a fully parallelisable manner, which makes it a desirable ABC method.

A weight produced by OMC represents the volume of the parameter region around a posterior sample which con-

Proceedings of the 23[rd]International Conference on Artificial Intelligence and Statistics (AISTATS) 2020, Palermo, Italy. PMLR: Volume 108. 

tains points that are as good as the sample. These points should thus be considered to be samples from the posterior too. However, if this region is particularly big, it is no longer appropriate to approximate the entire region with a single point and, as a result, OMC produces an overly con dent posterior. Figure 1 illustrates this failure case for a simple 1D scenario. We can see that OMC fails to characterise the posterior uncertainty and collapses regions of similar likelihood into a single point.

Figure 1: An example where OMC fails to approximate the true posterior, collapsing a region of similar likelihood into a single point. Heuristic OMC is a simple heuristic that (unsuccessfully) attempts to solve this issue. Robust OMC is the approach proposed in this paper. See Subsection 4.1 for details.

We propose Robust OMC (ROMC) which ef ciently identi es the regions of acceptable points themselves and samples from them directly. These regions are characterised by the set of all parameters which generate data whose distance to the observed data is less than a certain threshold. Instead of approximating a possibly very large region with a single point, we draw samples from the region which then replace the original OMC sample along with its weight. The main improvements of ROMC over OMC are:

- It handles likelihoods that are (nearly) at on signi cant regions in parameter space. All our experiments con rm this.
- It can be applied as post-processing to an original OMC run (see experiment 2), or as part of a standalone run (see experiment 3).
- OMC requires that the derivatives of the simulator can be computed or reasonably approximated, while ROMC does not (see experiment 3).

## 2 BACKGROUND

We here present the basics of Approximate Bayesian Computation (ABC), review the OMC algorithm by Meeds and Welling (2015), and discuss when it collapses regions of similar likelihood into a single point.

### 2.1 Rejection ABC

ABC methods produce samples from an approximate posterior (see e.g. Lintusaari et al., 2017). They generally make minimal assumptions about the model and only assume black-box access to the simulator $g(\theta, u)$. The target parameters $\theta$, which we wish to infer, are used as input to the simulator which then stochastically produces synthetic data by drawing $u$ using a random number generator. These $u$ correspond to nuisance variables since we do not aim to nd a distribution over them.

The simplest ABC method is Rejection ABC. In each iteration, data $x_i$ is simulated using the generative model $g(\theta, u_i)$ for some setting $u_i$ of the nuisance variables and with parameter values sampled from the prior. The distance between the simulated and observed data $d_i = d(x_i, x^0)$ is then computed and stored. After a sufciently large amount of samples has been generated, the algorithm accepts those having the lowest $d_i$ as samples from the approximate posterior. See Algorithm 4 in the appendix for pseudo-code.

While simple and robust, Rejection ABC is known to be computationally inef cient, especially when the prior space is large (e.g. Lintusaari et al., 2017). In most cases, more sophisticated methods are necessary.

### 2.2 Optimisation Monte Carlo

We start our brief review of Optimisation Monte Carlo (OMC) by noting that ABC algorithms in general implicitly approximate the likelihood function by the probability $\Pr(d(g(\theta, u), x^0) \leq \epsilon)$ that the generated data is within distance $\epsilon$ of the observed data (e.g. Lintusaari et al., 2017). ABC algorithms thus produce samples from the following approximate posterior

$$p(\theta | x^0) \propto p(\theta) \Pr(d(g(\theta, u), x^0) \leq \epsilon \mid \theta) \quad (1)$$

$$\propto p(\theta) \int p(u) 1_C(\theta, u)\, du, \quad (2)$$

where $p(u)$ is the density of the nuisance variables, $C = \{(\theta, u) : d(g(\theta, u), x^0) \leq \epsilon\}$ is the set of points $(\theta, u)$ for which the distance is below the threshold, and $1_C(\theta, u)$ is an indicator function that equals one only if $(\theta, u) \in C$. While this formulation uses the indicator function (boxcar kernel), more general kernels can be used as well.

The integral over $u$ corresponds to an expectation with respect to $p(u)$ and can thus be approximated as a sample average so that we obtain the approximation

$$p(\theta | x^0) \propto p(\theta) \frac{1}{n} \sum_{i=1}^{n} 1_C(\theta, u_i), \quad (3)$$

where the $u_i$ are sampled from $p(u)$. Importantly, this formulation essentially removes the randomness from the simulator: $g(\cdot\,; u_i)$, which occurs in each $1_C(\cdot\,; u_i)$ by definition of $C_\epsilon$, is a deterministic function of $\theta$ because $u_i$ is held fixed.

OMC exploits the fact that $g(\cdot\,; u_i)$ is a deterministic function in order to accelerate the sampling from the posterior in (3). For each $u_i$, OMC finds a value $\theta_i$ for which $g(\theta_i; u_i)$ and $x^0$ are within distance $\epsilon$. Importantly, this is done by minimising the deterministic cost function $d(g(\cdot\,; u_i), x^0)$ with respect to $\theta$. Note that settings of $u_i$ for which no $\theta_i$ has distance below $\epsilon$ are excluded from the sum and thus do not affect the posterior.

Meeds and Welling (2015) consider the case where the distance $d(g(\cdot\,; u), x^0))$ is the Euclidean distance between some summary statistics of the generated and observed data. We denote the summary statistics of the observed data by $\mathbf{y}^0 = \eta(x^0)$ and we further absorb the computation of the summary statistics into the simulator so that we obtain $f(\cdot\,; u) = \eta(g(\cdot\,; u))$, which can be regarded as a generative model on the level of the summary statistics. With this notation, OMC considers the distance $d(g(\cdot\,; u), x^0) = \|f(\cdot\,; u) - \mathbf{y}^0\|$.

OMC approximates the posterior in the limit of $\epsilon \to 0$ as a mixture of weighted point masses centred at the minimisers $\theta_i$: $p(\theta|x^0) \propto \sum_{i=1}^n w_i\, \delta(\theta - \theta_i)$. The value $w_i$ is a weight that reflects the local behaviour of the distance function and hence $d(\theta, u_i)$ around $\theta_i$. As shown by Meeds and Welling (2015), it equals $p(\theta_i)\det(J_i^\top J_i)^{-1/2}$ where $J_i$ is the Jacobian matrix with columns $\partial f(\theta_i; u_i)/\partial \theta_k$ with $\theta_k$ denoting the $k$-th element of $\theta$. Algorithm 1 summarises the OMC algorithm. For further details, we refer the reader to the original paper by Meeds and Welling (2015).

**Algorithm 1** Optimisation Monte Carlo. Generates $n$ independent samples $\theta_i$ with weights $w_i$ from the approximate posterior.

1: **for** $i = 1$ to $n$ **do**
2: $\quad u_i \sim p(u)$. Draw nuisance parameters.
3: $\quad \theta_i = \arg\min \|f(\theta; u_i) - \mathbf{y}^0\|$. Optimisation.
4: $\quad$ Compute $J_i$ with columns $\partial f(\theta_i; u_i)/\partial \theta_k$
5: $\quad$ Compute $w_i = p(\theta_i)\,(\det(J_i^\top J_i))^{-1/2}$
6: $\quad$ Accept $\theta_i$ as posterior sample with weight $w_i$.

As can readily be seen from Algorithm 1, ill-conditioned matrices $J_i^\top J_i$ produce very large weights for the corresponding $\theta_i$, possibly completely overshadowing the remaining samples and creating an approximate posterior density that is spiked at a single location (as in Figure 1). One may think that this issue can be easily fixed by regularising $J_i^\top J_i$ before computing the determinant. However, the issue goes deeper: ill-conditioned matrices $J_i^\top J_i$ occur when a large parameter region around the optimum $\theta_i$ produces data with small distances. These regions are poorly approximated by point masses or infinitesimally small ellipsoids, and amending the value of the weight cannot correct for this.[1] In other words, the OMC failure mode occurs when a large parameter region around $\theta_i$ is a solution to $\|f(\cdot\,; u_i) - \mathbf{y}^0\| < \epsilon$, which happens, for example, when the likelihood function is (nearly) constant around $\theta_i$.

## 3 ROBUST OPTIMISATION MONTE CARLO

We here develop a framework and concrete algorithms that have the benefits of OMC but do not collapse areas of similar likelihood into a point-mass.

### 3.1 The Robust OMC Framework

We start from the basic characterisation of the finite sample version of the ABC posterior in Equation (3) which holds irrespective of OMC. Under this approximation, the expectation of an arbitrary function $h(\theta)$ under the posterior $p(\theta|x^0)$ is

$$\mathrm{E}[h(\theta)|x^0] = \frac{\int h(\theta)p(\theta)\frac{1}{n}\sum_{i=1}^n 1_C(\theta; u_i)\,\mathrm{d}\theta}{\int p(\theta)\frac{1}{n}\sum_{i=1}^n 1_C(\theta; u_i)\,\mathrm{d}\theta} \quad (4)$$

$$= \frac{\sum_{i=1}^n \int h(\theta)p(\theta)1_{C^i}(\theta)\,\mathrm{d}\theta}{\sum_{i=1}^n \int p(\theta)1_{C^i}(\theta)\,\mathrm{d}\theta}; \quad (5)$$

where $C^i = \{\theta : d(g(\theta; u_i), x^0) < \epsilon\}$ is the set of parameters where, for a particular random seed or realisation of $u_i$, the simulated data is within distance $\epsilon$ from the observed data. Equation (5) features integrals $I_i$ in the numerator,

$$I_i = \int h(\theta)p(\theta)1_{C^i}(\theta)\,\mathrm{d}\theta, \quad (6)$$

and similar ones in the denominator. The integrals are generally intractable but since they correspond to an expectation with respect to the prior $p(\theta)$, they could be approximated by a sample-based average because sampling from the prior is typically possible in likelihood-free inference problems. However, this would be inefficient in the case of a broad prior as most samples would give $1_{C^i}(\theta) = 0$, i.e. they would essentially get rejected much like in rejection ABC. It is more efficient to sample from a proposal distribution $q(\theta)$ that only has support

[1] As pointed out in the original paper as a limitation of OMC, this happens if there are fewer summary statistics than parameters. But as shown here, this failure mode is more general. The proposed robust method provides a solution.

on the acceptance region $C^i$. We will discuss how to construct such $q_i(\theta)$ in Subsection 3.3. Assuming we have a suitable $q_i(\theta)$, the integrals $I_i$ can be approximated as

$$I_i = \int h(\theta) 1_{C^i}(\theta) \frac{p(\theta)}{q_i(\theta)} q_i(\theta)\, d\theta \quad (7)$$

$$\frac{1}{m} \sum_{j=1}^{m} h(\theta_{ij}) 1_{C^i}(\theta_{ij}) \frac{p(\theta_{ij})}{q_i(\theta_{ij})}; \quad (8)$$

where $\theta_{ij} \sim q_i(\theta)$, and equivalently for the integral in the denominator. Replacing the integrals in (5) with their sample-based approximations, we obtain

$$E[h(\theta)|x^0] \approx \frac{\sum_{i=1}^{n} \sum_{j=1}^{m} h(\theta_{ij}) 1_{C^i}(\theta_{ij}) \frac{p(\theta_{ij})}{q_i(\theta_{ij})}}{\sum_{i=1}^{n} \sum_{j=1}^{m} 1_{C^i}(\theta_{ij}) \frac{p(\theta_{ij})}{q_i(\theta_{ij})}}. \quad (9)$$

This expression corresponds to a weighted sample average of $h(\theta)$. Denoting $E[h(\theta)|x^0]$ by $h$, we have

$$h \approx \frac{\sum_{ij} w_{ij} h(\theta_{ij})}{\sum_{ij} w_{ij}}; \quad w_{ij} = 1_{C^i}(\theta_{ij}) \frac{p(\theta_{ij})}{q_i(\theta_{ij})}; \quad (10)$$

where the $w_{ij}$ are the (unnormalised) weights and the samples $\theta_{ij}$ are drawn from $q_i(\theta)$. Since our test function $h(\theta)$ has been arbitrary, this means that to obtain samples from the approximate posterior, we first draw samples $u_i$,[2] thus defining the acceptance region $C^i$, and then $m$ samples $\theta_{ij}$ from the corresponding proposal distribution $q_i(\theta)$. This process is what we refer to as the Robust OMC approach.

Before discussing the construction of the proposal distributions $q_i(\theta)$, we show how OMC is obtained from (9) by making additional assumptions. Some of the assumptions can be easily violated in practice, which then leads to the failure mode pointed out above and illustrated in Figure 1.

### 3.2 Connection to OMC

We present here the assumptions under which the proposed Robust OMC (ROMC) approach becomes standard OMC. It shows that ROMC is both more general and more robust than standard OMC.

**Theorem 3.1** Under the below assumptions, ROMC becomes equivalent to standard OMC as $\epsilon \to 0$.

**Assumption 1.** The distance $d(g(\theta, u), x^0)$ is given by the Euclidean distance between summary statistics $\|f(\theta, u) - y^0\|$.

**Assumption 2.** The proposal distribution $q_i(\theta)$ is the uniform distribution on $C^i$.

**Assumption 3.** The acceptance regions $C^i$ are approximated by the ellipsoid $C^i = \{\theta : (\theta - \theta_i)^\top J_i^\top J_i (\theta - \theta_i) \leq \epsilon - \delta\}$ where $J_i$ is the Jacobian matrix with columns $\partial f(\theta_i, u_i)/\partial_k$.

**Assumption 4.** The matrix square root $A_i$ of $J_i^\top J_i$ is full rank, i.e. $\text{rank}(A_i) = \dim(\theta)$.

**Assumption 5.** The prior is constant on the acceptance regions $C^i$, i.e. $p(\theta) = p(\theta_i)$ for $\theta \in C^i$.

The proof of Theorem 3.1 is given in the appendix.

Assumptions 1 and 2 are of technical nature, but Assumption 1 highlights that ROMC can also use distances other than Euclidean ones. Assumption 3 and 4 show that OMC relies on $C^i$ being well approximated by an ellipsoid of finite volume whose shape is determined by the local behaviour of $f(\theta, u_i)$ at $\theta_i$. The failure case described in Subsection 2.2 and illustrated in Figure 1 is caused by a violation of these two assumptions. Assumption 5 is also important because it shows that e.g. strong smoothing of the empirical distribution defined by the weighted samples in OMC would ignore that the prior distribution may not be constant on the corresponding finite-sized ellipsoid.

### 3.3 Robust OMC Algorithms

The Robust OMC (ROMC) framework has three key ingredients: the optimisation procedure as in OMC, the specification of the $\epsilon$-threshold as usual in ABC, and the proposal distribution. We here consider two sets of choices for these ingredients, resulting in Algorithms 2 and 3 detailed below. The former algorithm assumes access to (approximate) simulator gradients and can be run as post-processing to standard OMC, and the latter is gradient-free. The two algorithms show that the proposed ROMC framework is versatile and that it can be used to exploit specific properties of the model.

#### 3.3.1 Optimisation Step

To obtain the optimisation end point $\theta_i$, any optimisation algorithm can be used as long as it can minimise the distance with respect to $\theta$.

**Algorithm 2:** If gradients of the simulator are available, standard gradient-based optimisers are applicable.

**Algorithm 3:** If the simulator gradients are not available, we propose using Bayesian optimisation, which is a powerful optimisation scheme for objective functions that can be evaluated but whose gradients are not available (see e.g. Shahriari et al., 2016). In the simulations below, we use standard Bayesian optimisation (GPyOpt with default settings) that builds a Gaussian Process sur-

[2] In practice, this is done by fixing the seeds of the simulator.

Algorithm 2 Boxed Robust OMC. Requires simulator gradients, possible as post-processing to standard OMC.

1: for $i = 1$ to $n$ do
2: Obtain optimisation end point $\theta_i$.
3: Use curvature of $J_i^\top J_i$ to create a bounding box with volume $V_i$ as described in Subsection 3.3.3.
4: Define a uniform distribution $q_i(\theta)$ over the box.
5: for $j = 1$ to $m$ do
6: $\theta_{ij} \sim q_i(\theta)$
7: Accept $\theta_{ij}$ as posterior sample with weight $w_{ij} = 1_{C^i}(\theta_{ij}) \; p(\theta_{ij}) \; V_i$

Algorithm 3 Ellipsoidal Robust OMC. Does not require simulator gradients.

1: for $i = 1$ to $n$ do
2: Obtain optimisation end point $\theta_i$ and GP model distance $\hat{d}_i(\theta)$ using Bayesian optimisation.
3: Construct ellipse with volume $V_i$ using $\hat{d}_i(\theta)$ as described in Subsection 3.3.3.
4: Define a uniform distribution $q_i(\theta)$ over ellipse.
5: for $j = 1$ to $m$ do
6: $\theta_{ij} \sim q_i(\theta)$
7: Accept $\theta_{ij}$ as posterior sample with weight $w_{ij} = 1_{C^i}(\theta_{ij}) \; p(\theta_{ij}) \; V_i$

rogate model $\hat{d}_i$ for each distance $d(g(\theta, u_i), x^0)$ that needs to be minimised. The main purpose of the surrogate model in Bayesian optimisation is to decide at which $\theta$ to evaluate the distance next. This also applies to our situation but for ROMC, there are two further uses of the surrogate model: 1) it can be used to greatly speed up the acceptance check $1_{C^i}(\theta_{ij})$ in Equation (10) by using the surrogate distance rather than the true distance (see experiment 3); and 2) it can facilitate the construction of the proposal distribution $q_i(\theta)$ as discussed below.

### 3.3.2 Threshold

Algorithms 2 and 3: ROMC requires a value for the threshold $\epsilon$ that occurs in the term $1_{C^i}(\theta_{ij})$. This requirement to set a threshold is similar to most ABC algorithms where it is typically chosen as a small quantile of the observed distances (for other solutions see e.g. the work by Beaumont et al. (2002); Blum and Francois (2010); Papamakarios and Murray (2016); Chen and Gutmann (2019); Simola et al. (2019)). We take a similar approach but base the value of $\epsilon$ on the distances $d_i$ at the optimisation end points, $d_i = d(g(\theta_i, u_i), x^0)$. Since the $d_i$ are the minimal distances obtained in the optimisation step, their values are much smaller than the distances that one would see in other ABC algorithms, and we can choose a large quantile. In our simulations we chose the 90% quantile of the $d_i$ in order to be robust against bad optimisation instances. Since all $\theta_{ij}$ are saved, the exact value for $\epsilon$ can be changed later by the user without incurring any overheads.

### 3.3.3 Proposal Distribution

We here describe two methods to construct the proposal distributions $q_i(\theta)$. We assume that the optimisation step has given us a sample $\theta_i$ that is within $C^i$.

Algorithm 2: If simulator gradients are available, then it is possible to compute the matrix $J_i^\top J_i$. Its eigenvectors are orthogonal directions of highest curvature, along which we scan until we reach a point whose resulting distance no longer falls under the threshold. Doing so in each dimension specifies a box, and defining a uniform distribution on this box gives us the proposal distribution $q_i(\theta)$. Since $J_i^\top J_i$ is computed by standard OMC, this approach can be done entirely as post-processing to it.

Algorithm 3: Without simulator gradients, we construct a box around the optimisation end point as in Algorithm 2, except that we use the Hessian of the GP model instead of $J_i^\top J_i$. To robustify the approach against e.g. inaccuracies in the GP model and hence estimation of the curvature, we sample parameter values from inside the box and compute their distance to the observed data using the posterior mean of the GP model. This incurs practically no overhead and is considerably faster than using the true distances if the simulator is expensive. These parameter-distance pairs are then used to train a quadratic regression model of the distance. The contour where the distance is equal to the threshold defines an ellipsoid, and we use the uniform distribution on it as proposal distribution $q_i(\theta)$. Figure 2 visualises the construction.

The appendix has further details on both algorithms.

Figure 2: Algorithm 3, example proposal distribution $q_i(\theta)$. The contours show the GP model distance, the green dots visualise the true acceptance region $C^i$, and $q_i(\theta)$ is the uniform distribution on the red ellipse which well approximates $C^i$.

# 4 EXPERIMENTS

We assess Robust OMC (ROMC) on three tasks and compare its performance to standard OMC. As reference, we use posteriors obtained by expensive Rejection ABC runs. In the appendix we further show comparison results to the exact posteriors when tractable. The accuracy is measured using the Jensen-Shannon divergence. We also contrast effective samples sizes, which are given by $\text{ESS} = (\sum_{i=1}^{n} w_i)^2 / \sum_{i=1}^{n} w_i^2$. Note that we did not perform additional comparisons to other ABC methods because such comparisons were already performed by Meeds and Welling (2015). In their experiments, OMC showed clear improvements — about a factor of 10 fewer calls to the simulator per accepted sample. These advantages are inherited by ROMC.

We further compared the performance of ROMC to that of two simple heuristic fixes of OMC. In the first fix, we ignored a percentage of the smallest eigenvalues of the $J_i^\top J_i$ matrices when computing the determinants, hence reducing the magnitude of the biggest weights. This is similar to using pseudo-determinants. For the second fix, we stabilised the $J_i^\top J_i$ matrices by adding a constant value to the diagonals before computing the determinant. This stabiliser value was chosen by picking a given percentile of the magnitudes of the diagonal elements of all matrices.

## 4.1 Experiment 1: ROMC Resolves OMC Failure

We first consider a simple simulator to illustrate that ROMC resolves the identified issue of standard OMC, namely that it collapses regions of similar likelihood into a single point. The simulator is defined such that the likelihood function is flat in the area around $x^0$ and linear otherwise:

$$p(x \mid \theta) \propto \begin{cases} \theta^4 + u & \text{if } \theta \in [-0.5, 0.5] \\ c + u & \text{otherwise} \end{cases} \quad (11)$$

The parameter of interest is $\theta$ and $u \sim \mathcal{N}(0, 1)$ is a nuisance parameter and the only source of randomness. The term $c = (0.5 - 0.5^4)$ makes sure the function is continuous. Figure 8 in the appendix shows the simulator output for specific values of $u$. For posterior inference, we assume that the observed data is $x^0 = 0$. We used Algorithm 2 but exploited the fact that the box can be constructed analytically in this simple example.

Figure 1 in the introduction shows example posteriors. Despite generating samples which span the entire range of the prior, OMC assigns much higher weights to the samples in the middle of the flat region, resulting in a posterior that is overly confident at that point. Conversely, ROMC can nearly perfectly reproduce the reference posterior. For OMC, the effective sample size divided by the number of total samples is $\text{ESS} \approx 0.5$, while for ROMC, $\text{ESS} \approx 0.95$.

Figure 3: Experiment 1. Comparing OMC, two heuristic fixes to OMC, and Robust OMC. Smaller divergences are better.

Figure 3 shows how well OMC, ROMC, and the two discussed heuristic fixes of OMC can match the reference posterior at varying computational budgets. ROMC clearly outperforms OMC and both heuristics whatever the computation time. Additionally, we see that the two heuristic OMC methods perform similarly, with the pseudo-determinant version reaching a lower divergence. We will thus only consider that heuristic from now on.

## 4.2 Experiment 2: ROMC as Post-processing

This experiment showcases that Robust OMC can be performed as post-processing to standard OMC. We assume that we can compute simulator gradients (which is required in OMC), and hence we use Algorithm 2. We consider the case were the summary statistics are not completely informative about the parameters, which is a scenario that comes up often when using ABC in real-world problems (e.g. Aeschbacher et al., 2012). As a prototypical example of this scenario, we infer the mean and variance of a normal distribution with only the sample average available as a summary statistic (see appendix for details). Since there is no direct information on the variance, the optimisation surfaces will be completely flat in one direction.

Figure 4 compares the methods for different run times against the reference Rejection ABC posterior. Heuristic OMC refers to the version based on pseudo-determinants, and we use the hyper-parameter value that produced the best result. As before, ROMC outperforms the alternatives. Figures of the posteriors themselves are shown in Figure 11 in the appendix, and Heuristic

Figure 4: Experiment 2. Comparing OMC, heuristic OMC, and Robust OMC. Smaller divergences are better.

OMC performance as a function of its hyper-parameter is shown in Figure 13 in the appendix.

### 4.3 Experiment 3: Gradient-free ROMC

Here, we do not make use of the simulator gradients, and use Algorithm 3. This example is about inverse-graphics. It involves a considerably more complicated simulator that takes as input a set of 20 parameters and deterministically renders an image of a object (in this case, a teapot) on a uniform background. This is based on the generative model used by Moreno et al. (2016). We focus on the task of learning the posterior distribution of two colour parameters in a setting where there are two possible explanations for the observed image and thus the posterior is expected to be bi-modal. The remaining 18 parameters are used as nuisance parameters. They control the illumination, shape, pose and other aspects of the objects (see appendix).

The five illumination parameters are the most relevant ones among the nuisance variables for the task considered. The first four parameters specify the global illumination strength, the directional light strength, and the directional light angle. The fifth one allows the directional lighting to be in one of two modes: either white or red. This is what causes the bi-modality of the posterior — if the observed image depicts a red teapot, it is both possible that it could be a grey teapot with red lighting, or a red teapot with white lighting (see Figure 5). We use the former case as the observed image in our experiments.

The generative model was implemented using Open Differential Renderer (OpenDR, Loper and Black, 2014). While this renderer does allow us to compute the simulator gradients, we did not use them for any of our presented results. To compute the distance $d(x_i; x^0)$ between a simulated image $x_i = g(\ ; u_i)$ and the observed image $x^0$, we use a recognition model that predicts the target parameters for an input image $x$ and then compute the Euclidean distance between the two images' predicted parameters, so that

$$d(x_i; x^0) = \sqrt{\sum_{n=1}^{j} (r_n(x_i) - r_n(x^0))^2}; \qquad (12)$$

where $r_n(\ )$ is the prediction of the $n$-th parameter. The parameter estimates can thus be viewed as summary statistics. We implemented the recognition model as a neural network that was pre-trained on data generated from the simulator using a broad prior on the nuisance variables under daylight (white illumination). For details about the neural network's architecture, training procedure, and performance, as well as for examples of the training data, see the appendix.

Obtaining a single optimisation end point $\theta_i$, i.e. minimising the distance $d(x_i; x^0)$ with respect to the colour parameters, took approximately 2 minutes. We ran all simulations on a single computer only, and did not exploit the possibility to parallelise the inference. We based our posterior approximation on 250 optimisation end points. For ROMC, we generated 100 new samples per original optimisation end point.

The OMC and ROMC posteriors are shown in Figure 6, along with a Rejection ABC reference posterior and the predictions by the recognition network (see the appendix for results with the pseudo-determinant heuristic). First of all, we see that the recognition network prediction is off even though the network was well trained (see Figure 16 in the appendix). This is because red lightning conditions were not part of the training data and the recognition network does not well generalise towards this condition. Indeed, while still not accurate, the recognition network favours the solution in Figure 5(b), which is closer to images typically seen during training. Among the Bayesian methods, OMC is overly confident at a single location, with an effective sample size of 1.2 (out of 250 total samples). What is more, its posterior is not centred on a viable solution. On the other hand, ROMC produces two distinct posterior modes that contain the two possible solutions as we would want in this scenario, and

(a) Gray teapot with red light. (b) Red teapot with white light.

Figure 5: Example teapots (brightened for clarity). We use (a) as the observed data, but (b) is another possible explanation.

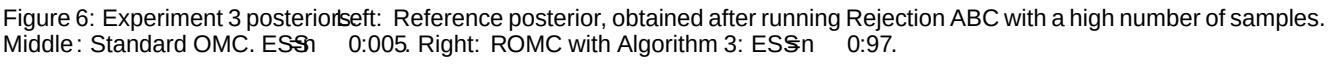


Figure 6: Experiment 3 posteriors. Left: Reference posterior, obtained after running Rejection ABC with a high number of samples. Middle: Standard OMC. ESS = 0:005. Right: ROMC with Algorithm 3: ESS = 0:97.

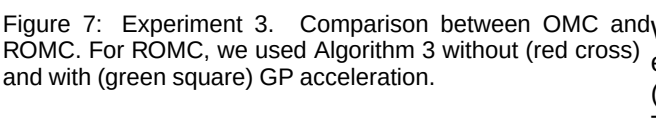


Figure 7: Experiment 3. Comparison between OMC and ROMC. For ROMC, we used Algorithm 3 without (red cross) and with (green square) GP acceleration.

it generally matches the reference posterior. Remarkably, these results were obtained despite using a biased recognition network, which points to a general ability of ABC in dealing with systematic biases in recognition networks.

Figure 7 compares the trade-off between accuracy and compute time for OMC (black) and ROMC (red and green). We see that ROMC as implemented in Algorithm 3 (red) is much more accurate than OMC but that it incurs an extra cost. This extra cost is due to the additional runs of the simulator that are needed for the acceptance check $\mathbb{1}_{C_i}(\theta_{ij})$ in step 7 of the algorithm. This cost could be reduced by parallelising the runs (which we did not do). Alternatively, we can use the GP model of the distance rather than the true distance in the acceptance check, which does not require additional runs of the simulator. We call this approach GP-ROMC. The figure shows that GP-ROMC (green) has almost the same accuracy as ROMC, but that it is much faster and only incurs a tiny overhead compared to OMC. The posterior for GP-ROMC is shown in Figure 12 in the appendix. In line with the numerical result, the posterior is very close to one obtained with exact acceptance checks.

## 5 CONCLUSIONS

This paper dealt with the task of performing Bayesian inference for parametric models when the likelihood is intractable but sampling from the model is possible. We considered Optimisation Monte Carlo (OMC) which has been shown to be a promising tool to efficiently sample from an approximate posterior. We showed that OMC, while efficient, has the shortcoming that it collapses regions of similar or near-constant likelihood into a single point. This matters because OMC samples might thus severely under-represent the uncertainty in the posterior and hence produce overly confident predictions.

We addressed this issue by introducing the more general framework of Robust Optimisation Monte Carlo (ROMC) and two concrete algorithms implementing it. The ROMC framework can be considered to be a form of ABC where we use optimisation to automatically construct suitable and localised proposal distributions. The first algorithm can be run as a form of post-processing after standard OMC to correct for the identified pathology. The second algorithm, unlike OMC, can be used when (approximate) gradients are not available. It uses a surrogate model of the distance and we have seen that this approach can be used to almost entirely eliminate the extra cost of ROMC compared to OMC. It is hence reasonable to also use a surrogate model in the first algorithm if reducing compute cost is necessary.

We tested the proposed framework and algorithms on both prototypical toy examples and complex inference tasks from inverse-graphics, and found that the proposed ROMC approach did accurately estimate the posteriors while OMC did not.

## Acknowledgements

This work was supported in part by the EPSRC Centre for Doctoral Training in Data Science, funded by the UK Engineering and Physical Sciences Research Council (grant EP/L016427/1) and the University of Edinburgh.

## References


Simon Aeschbacher, Mark A Beaumont, and Andreas Futschik. A novel approach for choosing summary statistics in approximate Bayesian computation. Genetics, 192(3):1027–1047, 2012.

Mark A Beaumont, Wenyang Zhang, and David J Balding. Approximate Bayesian computation in population genetics. Genetics, 162(4):2025–2035, 2002.

E. Bernton, P.E. Jacob, M. Gerber, and C.P. Robert. Approximate Bayesian computation with the Wasserstein distance. Journal of the Royal Statistical Society: Series B, 2018.

M. G. B. Blum, M. A. Nunes, D. Prangle, and S. A. Sisson. A comparative review of dimension reduction methods in approximate Bayesian computation. Statistical Science, 28(2):189–208, 2013.

Michael Blum and Olivier Francois. Non-linear regression models for Approximate Bayesian Computation. Statistics and Computing, 20(1):63–73, 2010.

Y. Chen and M.U. Gutmann. Adaptive Gaussian copula ABC. In Proceedings of the International Conference on Arti cial Intelligence and Statistics (AISTATS), 2019.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. Journal of Machine Learning Research, 12(Jul):2121–2159, 2011.

Matteo Fasiolo, Simon N. Wood, Florian Hartig, and Mark V. Bravington. An extended empirical saddlepoint approximation for intractable likelihoods. Electron. J. Statist., 12(1):1544–1578, 2018.

Paul Fearnhead and Dennis Prangle. Constructing summary statistics for approximate Bayesian computation: semi-automatic approximate Bayesian computation. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 74(3):419–474, 2012.

Jean-Jacques Forneron and Serena Ng. A likelihood-free reverse sampler of the posterior distribution. Essays in Honor of Aman Ullah, pages 389–415. Emerald Group Publishing Limited, 2016.

Jean-Jacques Forneron and Serena Ng. The ABC of simulation estimation with auxiliary statistics. Journal of Econometrics, 205(1):112–139, 2018.

Michael U Gutmann and Jukka Corander. Bayesian optimization for likelihood-free inference of simulator-based statistical models. The Journal of Machine Learning Research, 17(1):4256–4302, 2016.

Michael U Gutmann, Ritabrata Dutta, Samuel Kaski, and Jukka Corander. Likelihood-free inference via classification. arXiv preprint arXiv:1407.4981, 2014.

Jarno Lintusaari, Michael U Gutmann, Ritabrata Dutta, Samuel Kaski, and Jukka Corander. Fundamentals and recent developments in approximate Bayesian computation. Systematic biology, 66(1):e66–e82, 2017.

Matthew M Loper and Michael J Black. Opendr: An approximate differentiable renderer. In European Conference on Computer Vision, pages 154–169. Springer, 2014.

Pekka Marttinen, Nicholas J Croucher, Michael U Gutmann, Jukka Corander, and William P Hanage. Recombination produces coherent bacterial species clusters in both core and accessory genomes. Microbial Genomics, 1(5), 2015.

Ted Meeds and Max Welling. Optimization Monte Carlo: Ef cient and embarrassingly parallel likelihood-free inference. In Advances in Neural Information Processing Systems, pages 2080–2088, 2015.

Pol Moreno, Christopher KI Williams, Charlie Nash, and Pushmeet Kohli. Overcoming occlusion with inverse graphics. In Computer Vision–ECCV 2016 Workshops, pages 170–185. Springer, 2016.

George Papamakarios and Iain Murray. Fast ε-free inference of simulation models with Bayesian conditional density estimation. In Advances in Neural Information Processing Systems, pages 1028–1036, 2016.

L. F. Price, C. C. Drovandi, A. Lee, and D. J. Nott. Bayesian synthetic likelihood. Journal of Computational and Graphical Statistics, pages 1–11, March 2017.

Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P. Adams, and Nando de Freitas. Taking the human out of the loop: A review of Bayesian optimization. Proceedings of the IEEE, 104(1):148–175, 2016.

Umberto Simola, Jessica Cisewski-Kehe, Michael U. Gutmann, and Jukka Corander. Adaptive approximate Bayesian computation tolerance selection. arXiv:1907.01505, 2019.

S.A. Sisson, Y Fan, and M.A. Beaumont. Handbook of Approximate Bayesian Computation, chapter Overview of Approximate Bayesian Computation. Chapman and Hall/CRC Press, 2018.

O. Thomas, R. Dutta, J. Corander, S. Kaski, and M.U. Gutmann. Likelihood-free inference by ratio estimation. arXiv:1611.10242, 2016.

Tina Toni, David Welch, Natalja Strelkowa, Andreas Ipsen, and Michael PH Stumpf. Approximate Bayesian computation scheme for parameter inference and model selection in dynamical systems. Journal of the Royal Society Interface, 6(31):187–202, 2009.

Simon N. Wood. Statistical inference for noisy nonlinear ecological dynamic systems. *Nature*, 466(7310): 1102–1104, 2010.

# Appendix

## A Rejection ABC Algorithm

**Algorithm 4** Rejection ABC. Generates $n$ independent samples $\theta_i$ of the approximate posterior. Needs black-box simulator $g(\theta, u)$, observed data $x^0$, computational budget $N$, and number of accepted samples $n$.

1: **for** $i$ 1 to $N$ **do**
2: $\theta_i \sim p(\theta)$ . Draw parameters $\theta_i$ from prior.
3: $x_i \leftarrow g(\theta_i, u)$ . Simulate synthetic data using $\theta_i$.
4: $d_i = d(x_i, x^0)$ . Compute distance to $x^0$.
5: Accept the $n$ samples $\theta_i$ with the lowest distance $d_i$ as samples from the posterior.

A standard variant of Rejection ABC accepts only those samples whose distance falls under a threshold $\epsilon$ specified by the user.

## B Proof of Theorem 3.1

As a reminder, the assumptions under which ROMC becomes standard OMC as $\epsilon \to 0$ are:

**Assumption 1.** The distance $d(g(\theta, u), x^0)$ is given by the Euclidean distance between summary statistics $\|f(\theta, u) - y^0\|$.

**Assumption 2.** The proposal distribution $q_i(\theta)$ is the uniform distribution on $C^i$.

**Assumption 3.** The acceptance regions $C^i$ are approximated by the ellipsoid $C^i = \{\theta : (\theta - \theta_i)^\top J_i^\top J_i (\theta - \theta_i) \le \epsilon\}$ where $J_i$ is the Jacobian matrix with columns $\partial f(\theta_i, u_i)/\partial \theta_k$.

**Assumption 4.** The matrix square root $A_i$ of $J_i^\top J_i$ is full rank, i.e. $\text{rank}(A_i) = \dim(\theta)$.

**Assumption 5.** The prior is constant on the acceptance regions $C^i$, i.e. $p(\theta) = p(\theta_i)$ for $\theta \in C^i$.

Since settings of $u_i$ that result in empty sets $C^i$ (i.e. no $\theta_i$ exists such that the resulting distance is below $\epsilon$) are excluded from affecting the approximate posterior (both in ROMC and in standard OMC), we here consider only the case of non-empty sets $C^i$.

We start from Equation (9) using for $q_i(\theta)$ — as per Assumption 2 — the uniform distribution on $C^i$ with density $U^i(\theta)$,

$$U^i(\theta) = \frac{1}{\text{vol}(C^i)} 1_{C^i}(\theta). \tag{13}$$

Since the proposal distribution $q_i(\theta)$ is zero outside $C^i$, we have $1_{C^i}(\theta_{ij}) = 1$ for all $\theta_{ij}$ and hence

$$\mathbb{E}[h(\theta)|x^0] \approx \frac{\sum_{i=1}^n \sum_{j=1}^m h(\theta_{ij})\text{vol}(C^i)p(\theta_{ij})}{\sum_{i=1}^n \sum_{j=1}^m \text{vol}(C^i)p(\theta_{ij})}. \tag{14}$$

By Assumption 3, $C^i$ is an ellipsoid with volume determined by the matrix square root $A$ of $J_i^\top J_i$, as well as the value of $\epsilon$. It is possible to split the volume into a term determined by the shape of the ellipsoid and a term determined by $\epsilon$. With the change of variables $w = A_i(\theta - \theta_i)$, we have:

$$\text{vol}(C^i) = \int_{C^i} d\theta = \int_{w:\|w\|^2 \le \epsilon} |\det(A_i)|^{-1} dw \tag{15}$$

$$= |\det(A_i)|^{-1} \text{vol}(B_\epsilon); \tag{16}$$

where $B_\epsilon$ denotes an $\epsilon$-ball in Euclidean space. By Assumption 4, $|\det(A_i)|^{-1}$ is finite. Low-rank matrices $A_i$ would correspond to ellipsoids that extend without bound into one (or more) directions. We thus obtain

$$\mathbb{E}[h(\theta)|x^0] \approx \frac{\sum_{i=1}^n \sum_{j=1}^m h(\theta_{ij}) \frac{\text{vol}(B_\epsilon)}{|\det(A_i)|} p(\theta_{ij})}{\sum_{i=1}^n \sum_{j=1}^m \frac{\text{vol}(B_\epsilon)}{|\det(A_i)|} p(\theta_{ij})} \tag{17}$$

$$\frac{\sum_{i=1}^n |\det(A_i)|^{-1} \sum_{j=1}^m h(\theta_{ij})p(\theta_{ij})}{\sum_{i=1}^n |\det(A_i)|^{-1} \sum_{j=1}^m p(\theta_{ij})} \tag{18}$$

where we cancelled $\text{vol}(B_\epsilon)$ so that only the term $|\det(A_i)|^{-1}$ reflecting the geometry of the ellipsoid remains. Note that $|\det(A_i)|^{-1}$ can also be written as $|\det(A_i)|^{-1} = (\det J_i^\top J_i)^{-1/2}$. By Assumption 5, $p(\theta_{ij}) = p(\theta_i)$, so that we have

$$\mathbb{E}[h(\theta)|x^0] \approx \frac{\sum_{i=1}^n (\det J_i^\top J_i)^{-\frac{1}{2}} p(\theta_i) \sum_{j=1}^m h(\theta_{ij})}{\sum_{i=1}^n (\det J_i^\top J_i)^{-\frac{1}{2}} p(\theta_i)} \tag{19}$$

In this expression, the only dependency on $\epsilon$ remains in the samples $\theta_{ij} \sim U^i(\theta)$. In the limit of infinitely small $\epsilon$, $U^i(\theta)$ becomes a Dirac delta distribution $\delta(\theta - \theta_i)$ centred at $\theta_i$. This means that the only possible sample from that distribution is $\theta_i$ and hence that $h(\theta_{ij}) = h(\theta_i)$ for all $j$. In the limit of $\epsilon \to 0$, we thus obtain

$$\mathbb{E}[h(\theta)|x^0] \approx \frac{\sum_{i=1}^n (\det J_i^\top J_i)^{-\frac{1}{2}} p(\theta_i) h(\theta_i)}{\sum_{i=1}^n (\det J_i^\top J_i)^{-\frac{1}{2}} p(\theta_i)}. \tag{20}$$

This expression is a weighted average using samples $\theta_i$ and weights $w_i$ as defined in Algorithm 1. This means that the stated assumptions yield the weighted posterior samples of OMC and concludes the proof.

## C Constructing the Proposal Region for ROMC

Here we give more details on how we obtain the proposal distribution $q_i(\theta)$, using Figure 2 as a visual aid. We start with the optimisation end point $\theta_i$ (blue cross in Figure 2). We also have the curvature matrix, which is $J_i^\top J_i$ in Algorithm 2 or the Hessian of the GP model at $\theta_i$ in Algorithm 3. Its eigenvectors are used to determine what we call scan directions — one per dimension, along with its opposite (orange lines). We move along these directions until the resulting distance to $x^0$ is no longer under a certain threshold. The distance is calculated with the GP model if it is available in order to speed this process up. These end points along the scan directions are then used to create a *loose* rectangular box. This procedure is summarised in Algorithm 5. The loose box is then either the final proposal region (Algorithm 2), or is used to draw training data for the regression model to create an ellipse which is the final proposal region (Algorithm 3). In both cases, placing a uniform distribution on this region gives the final proposal distribution $q_i(\theta)$.

While we do use $\epsilon$ for the final $1_{C^l}(\theta_{ij})$ check, it would be reasonable to make the proposal region slightly bigger in order to ensure we capture as much of the actual acceptance region as possible at the cost of rejecting a few more samples. We achieve this by specifying the threshold for finding $q_i(\theta)$ to be bigger than the threshold used for $1_{C^l}(\theta_{ij})$. In Algorithm 2, we use $\epsilon_{prop}$ as defined by the 95% quantile of the optimisation end point distances to define the proposal region. In Algorithm 3, we use $\epsilon_{prop}$ based on the same 95% quantile on the ellipse, and a threshold based on a bigger 97.5% quantile to define the loose box from which the training data for the regression model is drawn. It is important to note that just using the single 90% quantile for all thresholds still produces a good final posterior, so the method is to some extent robust to that choice. The bigger thresholds we propose above produce a very slight performance improvement in practice (the divergence to the reference posterior with the bigger thresholds is about 1% smaller,[3] which indicates a better performance) and also make intuitive sense, which is why we use them in our final implementation.

For Algorithm 3, the regression model is quadratic and is trained via least squares. As it is quadratic, its contours are ellipsoidal. Thus, by finding the contour equal to $\epsilon_{prop}$, we obtain an ellipse that is suitable for being the proposal region.

Additionally, to show that Algorithm 3 is reasonably robust to the construction of the loose box, we compare the above proposal construction method with an alternative one. This additional method works as follows: begin with a tiny box centred at the optimisation end point, aligned with the scan directions obtained from the GP model's Hessian. Sample $\epsilon$ values uniformly from the box, compute their distance using the GP model, and check how many are under the acceptance threshold. Gradually expand this box until 50% of the sampled points are no longer within the threshold. That is the final loose region, which is then used to train the regression model and thus to produce an ellipse as before. The divergence between the resulting final posterior and the reference posterior is less than 3% bigger than when we use Algorithm 3 as described previously.[5] This difference is quite small, implying some robustness to the minute details of the construction of the proposal region.

**Algorithm 5** Box construction for one iteration $i$ of Algorithms 2 and 3. Needs $\theta_i$, $u_i$, step size $\delta$, number of refinements $K$, and curvature matrix $H_i$ ($J_i^\top J_i$ if Algorithm 2 or GP Hessian if Algorithm 3).

Compute eigenvectors $v_{dim}$ of $H_i$ $(dim = 1, \ldots, jj\ jj)$
**for** $dim$ 1 to $jj$ $jj$ **do**
  $\tilde{\theta} \leftarrow \theta_i$
  $k \leftarrow 0$
  **repeat**
    **repeat**
      $\tilde{\theta} \leftarrow \tilde{\theta} + v_{dim}\ \delta$ . Large step size $\delta$.
    **until** $d(g(\tilde{\theta}, u_i), x^0)$
    $\tilde{\theta} \leftarrow \tilde{\theta} - v_{dim}\ \delta/2$ . More accurate region boundary[4].
    $k \leftarrow k + 1$
  **until** $k = K$
  Set final $\tilde{\theta}$ as region end point.
  Repeat steps 3 - 13 for $v_{dim} = -v_{dim}$
Fit a rectangular box around the region end points.

## D Additional Information for Exp. 1

In this section, we further discuss Experiment 1 from Subsection 4.1. As a reminder, the likelihood function we use is defined by:

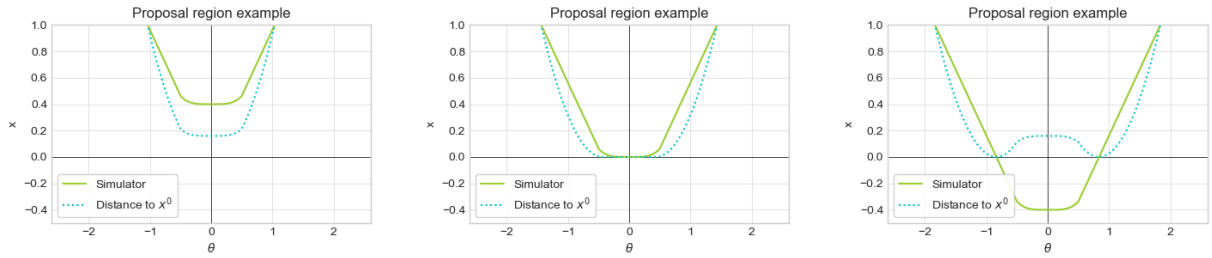
$$p(x|\theta) \propto \begin{cases} \theta^4 + u & \text{if } \theta \in [-0.5, 0.5] \\ c + u & \text{otherwise} \end{cases} \quad (21)$$

where $u \sim N(0, 1)$ is a nuisance parameter and the only source of randomness, and the term $c = (0.5^2 - 0.5^4)$ ensures the function is continuous. We assume that the observed data is $x^0 = 0$ and that the distance is Euclidean.

[3] The exact numbers for the Jensen-Shannon divergences are 0.290 and 0.293.

[4] This provides a minor benefit and is not necessary.

[5] The exact numbers for the Jensen-Shannon divergences are 0.290 and 0.298.




(a) $u = 0.4$ (b) $u = 0$ (c) $u = -0.4$

Figure 8: Experiment 1. Examples of the simulator's output and its distance to the observed data $x^0 = 0$ for three specific values of the nuisance parameter $u$.

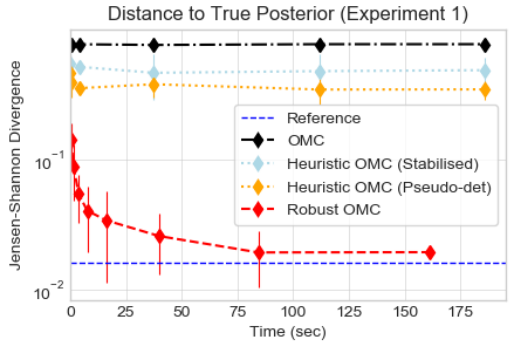



Figure 9: Experiment 1. Comparison of OMC, Robust OMC, and the two heuristic OMC methods against the true posterior. The reference Rejection ABC posterior's divergence is also shown.

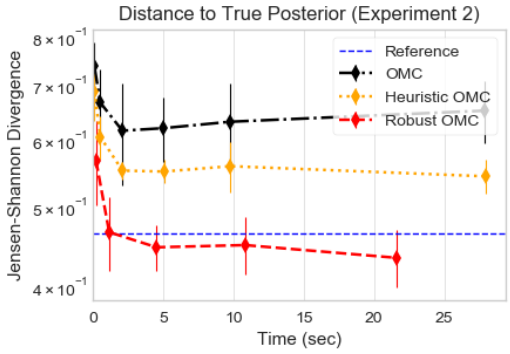



Figure 10: Experiment 2. Comparison of OMC and Robust OMC against the true posterior. The reference posterior's divergence is also shown.

Figure 8 shows the simulator output for specific values of $u$. In the case of $u > 0$, the simulator can never generate a data point that matches $x^0 = 0$ for any $\theta$, although for $u$ sufficiently close to 0, some $\theta$ may result in a data point within the distance threshold.

For $u < 0$, we enter the interesting situation where there are two values for which the distance is 0, and the distance is non-zero between them. In other words, there are two possible zero-solutions to the OMC optimisation objective $d(g(\theta; u_i); x^0)$. This would imply that there can be two disjointed acceptance regions $C^i$ for a single $u_i$ if the threshold $\epsilon$ is small enough. Currently, both OMC and our two Robust OMC implementations would not be able to capture the full disjointed region $C^i$ in such a scenario — OMC would at best approximate only the size of the region around $u_i$ with a weight, and Robust OMC would construct the box / ellipse and hence the proposal distribution only around $u_i$ as well. This problem does not manifest in the results we have presented as we computed the correct acceptance region analytically. In general, this is a difficult issue to solve, although a simple fix (i.e. a potential improvement on Robust OMC) would be to restart the optimiser at different initial values in order to find all possible solutions to the optimisation objective. It is also possible that, with enough samples, errors from the disjointed regions would average out and thus the final posterior would still be correct.

In the main text, Figure 3 compared OMC, Robust OMC, and two heuristic OMC methods against a reference posterior obtained from an expensive Rejection ABC run. Here we show the comparison made against the true posterior (which can be computed analytically) in Figure 9. As before, Robust OMC outperforms the other methods. It does not quite reach the level of the reference Rejection ABC method (blue dashed line) but this is to be expected as the reference was ran for a much longer time than the other methods.

## E Additional Information for Exp. 2

Here we present further details and results for Experiment 2 discussed in Subsection 4.2. Recall that the task was about inferring the parameters of a 2D Gaussian with a non-informative summary statistic, namely just the mean of a sample from the Gaussian, while also

having access to the simulator gradients.

We assume that the sample size is $M = 25$, and that the observed sample average is $^0 = 1$. We chose a Gaussian prior for the mean and Inverse-Gamma prior for the standard deviation: $p( ) = N(0;5)$, $p( ) =$ Inv-Gamma$(0{:}2;1)$. To compute the exact posterior, we used the fact that for a sample of size $M$ from a normal $N( ; ^2)$ the sample mean is distributed according to $N( ; ^2=M)$.

In Figure 10, we show the performance of the methods compared against the true posterior rather than the reference ABC one used in the main text. The Jensen-Shannon divergence between Robust OMC's posterior and the true posterior is much lower than for the other methods. The actual posteriors themselves are shown in Figure 11. OMC correctly identifies the marginal over but fails to do so for and does not match the reference posterior. On the other hand, Robust OMC reasonably matches the reference. Also note that OMC's effective sample size divided by the number of total samples is ESS$=n$ $0{:}01$, implying that the vast majority of the samples are ignored. Conversely, the corresponding value for Robust OMC is ESS$=n$ $0{:}55$, which is a significant improvement.

Figure 13 additionally compares the performance of our pseudo-determinant heuristic fix to OMC for different values of the hyper-parameter. This hyper-parameter represents the number of eigenvalues ignored when computing the determinants of the $\mathbf{J}_i^{>}\mathbf{J}_i$ matrices and hence the weights. While the divergence to the reference posterior does change, there is still a large gap between the best heuristic OMC and the robust OMC result.

## F Additional Information for Exp. 3

**Additional results.** Figure 12 qualitatively compares GP-ROMC — the approach where we speed up the final distance check in Algorithm 3 by using the GP model instead of the simulator — to the standard Robust OMC approach and the reference posterior. Additionally, Figure 14 compares overall Robust OMC performance to that of OMC and Heuristic OMC. Comparing Robust OMC to Heuristic OMC, we noticed that as more eigenvalues are ignored for Heuristic OMC, the weights become more similar and the accuracy of the posterior improves. This is because, in this particular example and unlike before, the unweighted samples $_i$ do reasonably represent the posterior so that setting all weights to a constant provides a reasonable solution. However, such tuning is not possible in practice where a reference posterior is not available.

**All renderer parameters.** The full list of parameters we use for the renderer in Experiment 3 in Subsection 4.3 is as follows:

> Ten shape parameters. The object's exact shape is based on a morphable mesh specified by Principal Component Analysis. The 10 dimensions used are the 10 highest principal components.
>
> Two rotation parameters, specifically the azimuth and elevation. The camera is always centred at the midpoint of the object.
>
> Three colour parameters — an RGB array which globally identifies the colour of the object. The first two (the red and green channels) are the target parameters over which we perform inference in Experiment 3.
>
> Five illumination parameters that characterise the lighting on the object. Unlike Moreno et al. (2016) who use spherical harmonics to model illumination, we use single-source directional lighting as it is more intuitive and natural.

**Recognition network details.** The network we used has 3 convolutional layers, each with 64 5x5 filters and 2x2 max pooling, followed by 2 linear layers with 256 and 64 hidden units respectively. Each layer uses ReLU activation functions except the final layer which uses an identity activation. The network parameters were learned with Adagrad (Duchi et al., 2011) as it showed the most robustness to the values of the hyper-parameters of the neural network training procedure (batch size, learning rate, and dropout probability), which in turn were chosen via hyper-parameter optimisation. Additionally, Figure 15 shows samples from the training set used to train the recognition network. Note that there is a reasonable amount of variability in shape, pose, illumination, and colour. Figure 16 shows that the learned recognition network is reasonably good at reconstructing the parameters for test images from the training data.

**Gaussian process model and Bayesian optimisation details.** For simplicity, we used the default options in GPyOpt when running our experiments — namely the Expected Improvement acquisition function, the Matern 5/2 kernel, 50 optimisation iterations, and 5 optimisation restarts. Importantly, experimentation with different settings did not lead to noticeable differences. We should also note that while using the GP model instead of the simulator in Experiment 3 incurred a significant speed up, the benefits of the GP model largely depend on the simulator cost. However, in the second experiment (where the simulator is very fast) using the GP model is still about as fast as using the simulator itself, and both approaches produce good results.

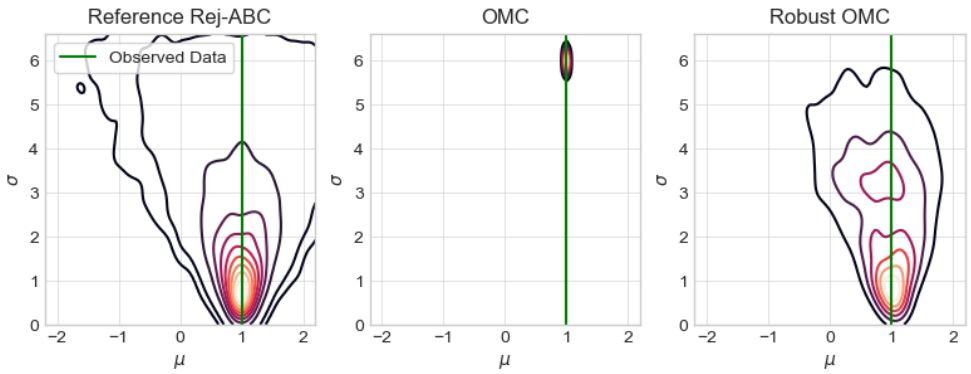



Figure 11: Experiment 2 posteriors. For OMC, ESS=$n$ 0.02, implying that the vast majority of the samples are ignored. For Robust OMC, ESS=$n$ 0.55, which is a significant improvement.

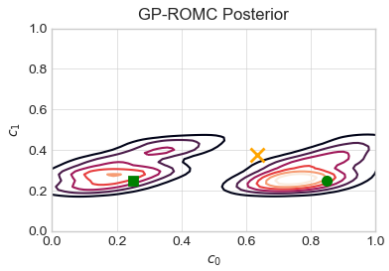



Figure 12: Experiment 3 posteriors. Similar to Figure 6, but with the GP-ROMC approach (middle) shown for comparison as well.

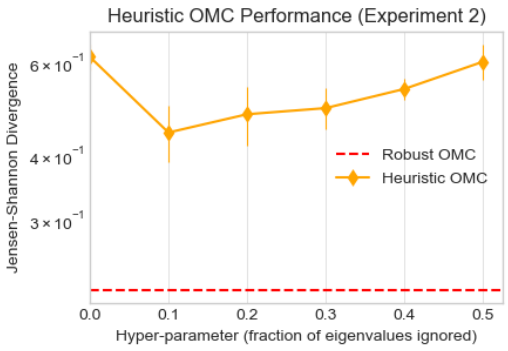



Figure 13: Performance of the heuristic pseudo-determinant fix to OMC as a function of its hyper-parameter in Experiment 2, compared against Robust OMC ran with roughly the same computational budget.

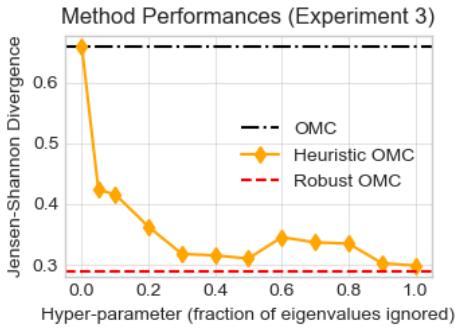



Figure 14: Performance of the heuristic pseudo-determinant fix to OMC in Experiment 3, compared against OMC and Robust OMC.