David J. Price, Nigel G. Bean, Joshua V. Ross, Jonathan Tuke
**An induced natural selection heuristic for finding optimal Bayesian experimental designs**
E Computational Statistics and Data Analysis, 2018; 126:112-124



Published at: http://dx.doi.org/10.1016/j.csda.2018.04.011

# An Induced Natural Selection Heuristic for Evaluating Optimal Bayesian Experimental Designs

David J. Price[1], Nigel G. Bean[2,3], Joshua V. Ross[2], Jonathan Tuke[2]

[1]Disease Dynamics Unit, Department of Veterinary Medicine, University of Cambridge, Cambridge CB3 0ES, U.K.,
[2]School of Mathematical Sciences, University of Adelaide, SA 5005, Australia
[3] ARC Centre of Excellence for Mathematical & Statistical Frontiers

**Abstract**

Bayesian optimal experimental design has immense potential to inform the collection of data, so as to subsequently enhance our understanding of a variety of processes. However, a major impediment is the difficulty in evaluating optimal designs for problems with large, or high-dimensional, design spaces. We propose an efficient search heuristic suitable for general optimisation problems, with a particular focus on optimal Bayesian experimental design problems. The heuristic evaluates the objective (utility) function at an initial, randomly generated set of input values. At each generation of the algorithm, input values are "accepted" if their corresponding objective (utility) function satisfies some acceptance criteria, and new inputs are sampled about these accepted points. We demonstrate the new algorithm by evaluating the optimal Bayesian experimental designs for two popular stochastic models: a Markovian death model, and a pharmacokinetic model. The designs from this new algorithm are compared to those evaluated by existing algorithms, and computation times are given as a demonstration of the computational efficiency. A comparison to the current "gold-standard" method are given, to demonstrate that INSH finds designs that contain a similar amount of information, but more computationally efficiently. We also consider a simple approach to the construction of sampling windows for the pharmacokinetic model using the output of the proposed algorithm.

**Keywords:** Bayesian optimal design; Optimisation heuristic; Stochastic models; Sampling windows

# 1 Introduction

Optimising the design of experiments is an important consideration in many areas of science, including, but not limited, to: biology (Faller et al. [2003]), clinical trials (Berry [2004]) and epidemiology (Pagendam and Pollett [2013]). The theory of optimal experimental design is a statistical framework that allows us to determine the optimal experimental protocol to gain the most information about model parameters, given constraints on resources.

In evaluating an optimal Bayesian design, there are two main components: the search across the design space, and the evaluation of the utility. There have been many approaches to improving the efficiency of both aspects, summarised by Ryan et al. [2015a]. Recently, Overstall and Woods [2016b] proposed the Approximate Coordinate Exchange (ACE) algorithm for finding optimal Bayesian experimental designs efficiently. The method utilises a coordinate exchange algorithm to update one dimension of the design at a time, coupled with a Gaussian process in order to search each dimension efficiently. Price et al. [2016] proposed an ABC-based approach to evaluating the utility in an efficient manner, embedded within an exhaustive search of the design space. The overall optimal design tool lends itself to evaluating multiple criteria simultaneously (*i.e.*, simultaneously evaluate different utilities for each design), albeit without any sensible search mechanism across the design space. It has been asserted that the future of optimal Bayesian experimental design lies in the ability to evaluate the optimal designs for large-scale problems (*i.e.*, large or high-dimensional design spaces), in a computationally-efficient manner (Ryan et al. [2015a]). In this paper, we address this by proposing a new, efficient search algorithm which is more efficient than the current "gold-standard" ACE algorithm.

The search heuristic we present performs targeted sampling of the design space to find high utility designs, without making any assumptions about the shape of the utility function. The algorithm determines similarly well-performing designs as determined by ACE (Overstall and Woods [2016b]), with greater computational efficiency. Our method borrows the idea of targeting regions of high utility, as per the MCMC approaches, by sampling new designs at each iteration around the "best" designs. The "best" designs are chosen according to some acceptance criteria; in this work we consider two alternative approaches. First, we consider accepting all points that are within some proportion of the current maximum objective (utility) function, and increase this proportion towards one as the algorithm progresses (*i.e.*,

such that only designs that perform as well as the optimal are retained). This choice of acceptance criteria is similar to a fully adaptive cross-entropy algorithm (De Boer et al. [2005]), in that the number of "elite" samples accepted at each iteration is changed. Alternatively, this increasing cut-off can be thought of in a similar way to an *annealing schedule* in a simulated-annealing algorithm (Rutenbar [1989]). The initial cutoff value, and the rate at which the cutoff increases towards one controls the trade-off between exploration and exploitation, as in existing algorithms (e.g., the annealing schedule in a simulated-annealing algorithm, Rutenbar [1989], or the proportion of "elite" samples retained in a cross-entropy algorithm, De Boer et al. [2005]). Second, we consider accepting a fixed number of samples at each iteration of the algorithm, in the same way as a standard cross-entropy algorithm (De Boer et al. [2005]). However, in both scenarios, rather than updating a sampling distribution at each iteration based on a combination of the "best" samples – as in a cross-entropy or a genetic algorithm – we propose new samples around each of these "best" samples independently, in a similar way to a sequential importance sampling algorithm. We describe this algorithm using the notion of "survival-of-the-fittest", as the "fittest" individuals – according to their objective (utility) function value – survive at each iteration (generation) based on a user-defined acceptance criteria, to produce offspring for the next generation. Hence, we refer to this new algorithm as an Induced Natural Selection Heuristic (INSH).

By independently sampling new designs around each accepted design, we aim to avoid the pitfalls associated with some other optimisation routines. For example, INSH is able to sample multiple regions of high utility at a time, thus exploring multiple optima simultaneously, rather than potentially being stuck at a local optima. Furthermore, by not combining the retained designs in any way, INSH avoids the potential to move to a region of low utility when there are multiple modes that is at the "centre" of the modes – as may occur in a cross-entropy or genetic algorithm. By taking a sampling approach, as opposed to trying to approximate the function, INSH makes no assumptions about the shape of the utility function – thus, it is not limited to utility functions that are, for example, smooth. Utilising (embarrassingly) parallel computation tools, the method can evaluate the utility for a large number of designs in each iteration, in an efficient manner.

The ACE algorithm of Overstall and Woods [2016b] has allowed the consideration of Bayesian optimal designs for a larger, more-complex class of statistical models and experiments than was possible with previous algo-

rithms. There are a number of drawbacks to the ACE algorithm, however. By searching in one-dimension at a time, the ACE algorithm risks missing the globally optimal design, and instead finding only local-optima. The authors approach to avoid this is to re-run the algorithm from a number (typically 20) of randomly generated initial designs. Similarly, as noted by the authors, by searching in one-dimension at a time, the algorithm will be inefficient in scenarios where there is a large correlation between the design variables – a problem which adds to the difficulty in choosing a suitable number of iterations for each phase of the algorithm. The algorithm requires a sufficiently good estimate of the utility when determining whether to accept the candidate design – spurious estimates may lead to sub-optimal candidate designs being accepted, and thus push the algorithm away from regions of high utility. Alternatively, a large improvement in the computation time comes about from the estimation of the utility surface in each dimension in the form of a Gaussian process based on a number of candidate points. This approximation to the utility surface based on noisy evaluations of the utility aims to provide a smooth approximation to the surface. When the surface is not smooth, or has a discontinuity (e.g., the utility surface corresponding to the death model in Figure 2a), this may cause significant problems for the ACE algorithm in finding optimal designs.

In the following, we present the INSH search algorithm in a general framework, and we note that efficient evaluation of the utility is another problem that needs to be addressed. We consider two existing approaches to evaluating the utility: the method used in ABCdE (Price et al. [2016]) in a scenario where the benefits of this approach are realised; and a nested Monte-Carlo approximation using code from the `acebayes` package (Overstall and Woods [2016a]).

We consider the problem of finding the optimal design for two stochastic models – the death model, and a pharmacokinetic (PK) model tracking the concentration of a drug or treatment in the blood. In the death and PK examples, a design $d$ consists of $n$ sampling times $(t_1, \ldots, t_n)$, subject to some problem-specific constraints. The first problem we address is when to observe the stochastic process in order to gain the most information about the model parameters. The stochastic death model has been considered previously in a Bayesian framework by Cook et al. [2008], Drovandi and Pettitt [2013], and Price et al. [2016]. We evaluate the optimal experimental designs for 1-4 observation times, in order to demonstrate the efficacy of the method, and the relative improvement in computation time. Second, we consider the

optimal Bayesian experimental design for a PK model – a process where the design space is a considerably higher-dimension – in order to demonstrate the efficiency of the INSH algorithm for larger design spaces. The optimal designs are compared to those evaluated using the "gold-standard" Approximate Coordinate Exchange algorithm of Overstall and Woods [2016b]. We also consider the idea of sampling windows for this example, which have been considered previously by Green and Duffull [2003], Chenel et al. [2005], Graham and Aarons [2006], McGree et al. [2012], and Duffull et al. [2012], for example. The idea of sampling windows allows those implementing an optimally-chosen design some flexibility in choosing the sampling times, such that the resulting design is more practically feasible. By defining sampling windows, we can dictate a set of near-optimal designs – which are practically feasible – which can be implemented more easily. This avoids the scenario where an inferior design is chosen preferentially by those that are implementing the design, having been supplied with an impractical optimal design.

# 2 Background

The aim of optimal experimental design is to determine the best experimental protocol in order to maximise some utility of the experiment. To achieve this aim, we specify a utility function $U(\boldsymbol{\theta}, \boldsymbol{x}, d)$ representing how we 'value' the experimental design $d$, chosen from the set of all designs $\mathcal{D}$, where $\boldsymbol{\theta}$ represents the model parameters and $\boldsymbol{x}$ is the data. We are interested in the expected utility of using design $d$, over the unknown model parameters and data. That is, we wish to evaluate,

$$\begin{aligned} u(d) &= E_{\boldsymbol{\theta},\boldsymbol{x}}[U(\boldsymbol{\theta}, \boldsymbol{x}, d)] \\ &= \int_{\boldsymbol{x}} \int_{\boldsymbol{\theta}} U(\boldsymbol{\theta}, \boldsymbol{x}, d) p(\boldsymbol{x} \mid \boldsymbol{\theta}, d) p(\boldsymbol{\theta}) d\boldsymbol{\theta} d\boldsymbol{x}, \end{aligned} \tag{1}$$

where $p(\boldsymbol{x} \mid \boldsymbol{\theta}, d)$ is the likelihood function of the unobserved data $\boldsymbol{x}$, under design $d$, and $p(\boldsymbol{\theta})$ is the prior distribution of the model parameters. The optimal design $d^*$ maximises the expected utility over the design space $\mathcal{D}$, that is, $d^* = \text{argmax}_{d \in \mathcal{D}} u(d)$.

The utility function we use throughout this work is the Kullback-Leibler divergence (Kullback and Leibler [1951]) from the prior distribution to the

posterior distribution (which is independent of $\boldsymbol{\theta}$),

$$U(\boldsymbol{x}, d) = \int_{\boldsymbol{\theta}} \log\left(\frac{p(\boldsymbol{\theta} \mid \boldsymbol{x}, d)}{p(\boldsymbol{\theta})}\right) p(\boldsymbol{\theta} \mid \boldsymbol{x}, d) d\boldsymbol{\theta},$$

which leads to an expected utility:

$$u(d) = \int_{\boldsymbol{x}} \int_{\boldsymbol{\theta}} \log\left(\frac{p(\boldsymbol{\theta} \mid \boldsymbol{x}, d)}{p(\boldsymbol{\theta})}\right) p(\boldsymbol{x} \mid \boldsymbol{\theta}, d) p(\boldsymbol{\theta}) d\boldsymbol{\theta} d\boldsymbol{x}. \tag{2}$$

See Price et al. [2016] for details of the derivation.

Alternatively, it is commonplace to consider the Shannon Information Gain (SIG), which can be written as:

$$\begin{aligned} U(\boldsymbol{\theta}, \boldsymbol{x}, d) &= \log p(\boldsymbol{\theta} \mid \boldsymbol{x}, d) - \log p(\boldsymbol{\theta}) \\ &= \log p(\boldsymbol{x} \mid \boldsymbol{\theta}, d) - \log p(\boldsymbol{x} \mid d), \end{aligned} \tag{3}$$

through the application of Bayes' theorem. Maximisation of the expected SIG is equivalent to maximisation of the expected Kullback-Leibler divergence above.

Analytic evaluation of the expected utility function $u(d)$ is typically not possible. In light of this limitation, Müller [1999] proposed an MCMC sampling scheme from the joint probability distribution, $h(\boldsymbol{\theta}, \boldsymbol{x}, d) \propto U(\boldsymbol{\theta}, \boldsymbol{x}, d) p(\boldsymbol{x} \mid \boldsymbol{\theta}, d) p(\boldsymbol{\theta})$. Sampling from $h(\boldsymbol{\theta}, \boldsymbol{x}, d)$ in this way allows us to obtain samples from a distribution that is proportional to $u(d)$ by considering the marginal distribution of $h(\boldsymbol{\theta}, \boldsymbol{x}, d)$ in $d$. The approximate optimal experimental design is thus obtained as the mode of the function proportional to $u(d)$, as determined by the samples from the MCMC sampling scheme. The MCMC sampling scheme defined by Müller [1999] is outlined in Online Resource A. For further details, extensions, and comments on the algorithm, see Müller [1999]. The difficulty in evaluating the mode of a high-dimensional utility surface is one of the limiting factors of the MCMC algorithms for evaluating optimal experimental designs for a large number of design variables (Ryan et al. [2015a]).

## 2.1 ACE Algorithm

The Approximate Coordinate Exchange algorithm of Overstall and Woods [2016b] directly addresses the need for a computationally efficient algorithm

for determining optimal Bayesian experimental designs in large-dimensional design spaces (Ryan et al. [2015a]). The reader is directed to Overstall and Woods [2016b] for full details of the algorithm. Briefly, the algorithm considers each dimension of the experimental design one-at-a-time (e.g., the first observation time in an observation schedule), and evaluates the utility at a number of new, candidate values in that dimension (e.g., consider the utility at each of $q$ equally spaced times across the feasible range of observation times, conditional on the other elements of the design). Having obtained these approximate utilities across the feasible range for the particular dimension of the design, a Gaussian process is fit to these candidate values to find an approximate "optimal" value as an update to this dimension of the design (accepted with some probability). The algorithm cycles through each design variable (probabilistically) updating them to the best value according to the Gaussian process approximation to the utility. The ACE algorithm is the first algorithm that is capable of dealing with designs in large dimensional design spaces, in a computationally feasible amount of time.

## 2.2 ABCdE Algorithm

In the following work, we utilise the efficient approach to evaluating the utility used in the ABCdE algorithm by Price et al. [2016], within the INSH algorithm. We give a brief description here, and detail the ABCdE algorithm in Online Resource A. The ABCdE algorithm evaluates the utility of a collection of designs (in Price et al. [2016], this collection was all designs across an exhaustive grid on the design space) simultaneously in an embarrassingly parallel fashion using, e.g., `parfor` in MATLAB, or the `doParallel` and `foreach` packages (Analytics and Weston [2015a,b]) in R (R Core Team [2016]). For each design, we pre-simulate $N_{pre}$ data sets from the model, corresponding to parameters independently sampled from the prior distribution. We use each set of the pre-simulated data as the "observed datum" one-by-one, and evaluate the utility using all the $N_{pre}$ data as "simulated data". This creates a set of posterior samples having observed every set of simulated data for a particular design. That is, for simulated data $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{N_{pre}}$ under design $d$, we determine Approximate Bayesian Computation (ABC) posterior distributions $[\hat{p}(\boldsymbol{\theta} \mid \boldsymbol{x}_1, d), \hat{p}(\boldsymbol{\theta} \mid \boldsymbol{x}_2, d), \ldots, \hat{p}(\boldsymbol{\theta} \mid \boldsymbol{x}_{N_{pre}}, d)]$ using a standard rejection ABC algorithm (detailed in Online Resource A). We pre-simulate data across all designs and thus we can pass pre-simulated data and corresponding parameter values to the ABC rejection-algorithm. This increases

memory requirements, but saves on simulation effort, since we do not simulate new parameter values and data each time, as would typically be done in an ABC rejection-algorithm. We evaluate the utility using each of these $N_{pre}$ posterior distributions under a particular design, and take the average of these $N_{pre}$ values to be our measure of the expected utility for that design.

One of the main advantages of this approach to evaluating the utility comes about for discrete data, and when the number of unique datum are small compared to the number of simulations considered, $N_{pre}$. We evaluate a single (approximate) posterior for each unique datum only, as the approximate posterior distributions for each of the data sets will be the same (ignoring the removal of the sampled parameter that was used to produce the data set we are currently considering). We note that it is often not the case that we have discrete data, or, we may wish to consider a large number of design points where the dimension of the data is increased. In these cases, this advantage is lost. Hence, we consider this approach to evaluate the utility for the death model only.

# 3 INSH Algorithm

In the following, we present a new algorithm to find optimal Bayesian experimental designs efficiently. Given the current advantages of parallel computing – which are rapidly improving as parallel-computing becomes more widely-available, more easy to implement, and more powerful – we wish to retain some of the parallel aspects of the ABCdE algorithm (or other optimisation routines, for example, Evans et al. [2007]). However, we embrace an advantageous aspect of the MCMC search algorithms implemented by Müller [1999], Cook et al. [2008], and Drovandi and Pettitt [2013], namely, we seek to spend less computational effort evaluating designs in low-utility regions. This forms the crux of the efficiency of an MCMC approach, and is achieved by sampling from a function proportional to the utility, and hence, more samples are taken from regions of high utility. However, MCMC is inherently sequential, in that the next design is chosen after evaluating the current design, and so it cannot be parallelised efficiently. The new algorithm we propose instead evaluates the utility of multiple designs simultaneously – in order to retain the parallel nature that was advantageous in the ABCdE algorithm – and samples new designs at each iteration of the algorithm around designs satisfying some acceptance criteria. The acceptance criteria for designs at

each iteration can be chosen in a number of different ways. In this paper, we demonstrate an increasing cut-off for the proportion of utility compared to the current maximum, similar to an annealing schedule in a simulated-annealing algorithm (Rutenbar [1989]), or alternatively, accepting a fixed number of the "best" designs, similar to the proportion of "elite" samples in a cross-entropy algorithm (De Boer et al. [2005]). However, in contrast to these existing optimisation algorithms, the algorithm presented here considers multiple designs at each iteration, allowing us to explore the design space more efficiently. The INSH algorithm is detailed in Algorithm 1. Note that in order to continue to explore the region near the current optimal design, the optimal design that has been considered thus far is re-introduced into the set of designs that are to be sampled around, at each iteration.

**Algorithm 1** INSH Algorithm

1: Choose an initial set of designs. $D$ (e.g., a coarse grid of design points across the design space, or randomly sample).
2: Specify the number of generations (iterations) of the algorithm $W$, a perturbation function $f(d \mid d')$, and the acceptance criteria.
3: **for** $w = 1$ to $W$ **do**
4: For each design $d^i \in D$, sample parameters $\boldsymbol{\theta} \sim p(\boldsymbol{\theta})$, and simulate data $\boldsymbol{x}^i$ from the model.
5: Evaluate utility $u(d^i)$, for each design $d^i \in D$.
6: Set $D'$ to be the designs which satisfy the acceptance criteria, and the current optimal design $d^*$ (even if it occurred in a previous generation).
7: Sample $m$ designs from $f(d \mid d')$, for each $d' \in D'$. Set $D$ to be these newly sampled designs.
8: **end for**

**Output:** Set of designs $d$, and corresponding utilities $u(d)$ (and hence, the optimal design $d^* = \underset{d \in \mathcal{D}}{\text{argmax}}(u(d))$).

## 3.1 Evaluation of the Utility

An efficient approach to evaluate the utility of a design in Step 5 of Algorithm 1 is the same approach used in the ABCdE algorithm (Price et al. [2016]). In particular, we use Steps 3 to 9 of Algorithm 2, in Online Resource A. Note that this approach is suitable for discrete data, and for low-dimensional

design spaces. This is due to the majority of the efficiency coming from having to evaluate a posterior distribution only once for each unique data set, as described previously. As the number of possible unique data sets increases – for example, either by observing the process more often (increasing the size of the design space), or having a larger population – this approach becomes less efficient. We use this approach to demonstrate the efficacy of the INSH algorithm for the Markovian death model.

For cases where the dimension of the data is too large (or continuous), we must consider an alternative approach to evaluating the utility for each design. As noted previously, this is one of the two main challenges when considering Bayesian optimal experimental design. A suitable and efficient method for evaluation of the utility for a design is often problem-specific, and a number of different approaches have been considered – a summary of these approaches can be found in Ryan et al. [2015a]. For the pharmacokinetic example we consider subsequently, we implement the utility function of Overstall and Woods [2016b] provided in the `acebayes` package in R. Briefly, the SIG utility in equation (3), is estimated by a nested Monte-Carlo approximation of the values $p(\boldsymbol{x} \mid \boldsymbol{\theta}, d)$ and $p(\boldsymbol{x} \mid d)$, within the Monte-Carlo approximation to the expected utility, $u(d)$. Borrowing the notation of Overstall and Woods [2016b], define $\boldsymbol{\psi} = (\boldsymbol{\theta}, \boldsymbol{\gamma})$ to be the combination of the parameters of interest, $\boldsymbol{\theta}$, and nuisance parameters, $\boldsymbol{\gamma}$. Then, we use $\tilde{B}$ simulations to approximate the inner Monte-Carlo estimates:

$$\tilde{p}(\boldsymbol{x} \mid \boldsymbol{\theta}, d) = \frac{1}{\tilde{B}} \sum_{b=1}^{\tilde{B}} p(\boldsymbol{x} \mid \boldsymbol{\theta}, \tilde{\boldsymbol{\gamma}}_b, d), \qquad \text{and} \qquad \tilde{p}(\boldsymbol{x} \mid d) = \frac{1}{\tilde{B}} \sum_{b=1}^{\tilde{B}} p(\boldsymbol{x} \mid \tilde{\boldsymbol{\theta}}_b, \tilde{\boldsymbol{\gamma}}_b, d),$$

where $(\tilde{\boldsymbol{\theta}}_b, \tilde{\boldsymbol{\gamma}}_b)$ are the $\tilde{B}$ parameters sampled from the prior distribution of $\boldsymbol{\psi}$. Similarly, $B$ simulations are used to evaluate the outer Monte-Carlo estimate,

$$\tilde{u}(d) = \frac{1}{B} \sum_{l=1}^{B} \left[ \log \tilde{p}(\boldsymbol{x}_l \mid \boldsymbol{\theta}_l, d) - \log \tilde{p}(\boldsymbol{x}_l \mid d) \right],$$

with $\{\boldsymbol{x}_l, \boldsymbol{\theta}_l\}$ parameters, and corresponding simulations, sampled from the prior and simulated from the model, respectively. In the work of Overstall and Woods [2016b], the authors use $\tilde{B} = B = 1{,}000$ to evaluate the candidate designs' utilities in the one-dimensional search (Step 1b of the ACE Algorithm in Overstall and Woods [2016b]), and $\tilde{B} = B = 20{,}000$ to evaluate the utility when determining whether to accept the candidate design

(Steps 1d and 3e of the ACE Algorithm in Overstall and Woods [2016b]; note that Step 3 is not implemented for the compartmental model). As the INSH method considers the utility of multiple designs simultaneously, and multiple designs in each region, rather than searching for the optimal design sequentially, we are able to use significantly less effort (*i.e.*, Monte Carlo simulations) to evaluate the utility of each design without detriment to the results.

## 3.2 Choice of Acceptance Criteria

There are a number of ways the "best" designs can be retained at each iteration of the INSH algorithm. We propose the following two approaches, which we implement for the Markovian death model and pharmacokinetic examples respectively.

First, we propose that those designs which correspond to a utility within some percentage of the current maximum utility are retained. That is, keep all $d^i$ such that $u(d^i) \geq \alpha_w \times u(d^*)$, $\alpha_w \in (0,1)$, $w = 1, \ldots, W$, where $d^*$ is the "best" design considered up to generation $i$ of the algorithm. The proportional cut-off $\alpha_w$ should increase steadily from some initial value in (0,1), up to the maximum of one. The rate at which the cut-off increases balances the trade-off between exploration and exploitation, as in other optimisation routines (e.g., the rate at which the annealing schedule converges to zero in simulated-annealing, or the proportion of "elite" samples retained in a cross-entropy algorithm). As the INSH algorithm progresses, later generations of candidate designs will (ideally) be located closer to the optimal design than many of the earlier generations. If the region containing the optimal design is retained throughout the algorithm, naturally, this region is explored better than regions of low utility. However, as the cut-off for the utility approaches one, there may be very few designs accepted/retained and thus very few new designs to be considered. Hence, it is sensible to replace the number of sampled designs at each generation, $m$, with a non-decreasing sequence of samples $m_w$, $w = 1, \ldots, W$, such that later generations (*i.e.*, those closer to the optimal) produce more offspring. Hence, we sample more designs around the optimal design, and thus get better resolution in this region.

One downside of this approach to the acceptance criteria is that all designs in the initial waves of the algorithm may be within the initial cutoff $\alpha_1$. Hence, the algorithm accepts *all* designs, and thus samples many designs in the following iteration. It is possible that this occurs for multiple

generations of the algorithm, resulting in a large increase in the computation time, as a larger portion of the design space is explored. Thus, one must first analyse the distribution of utilities across the design space in order to choose a sensible sequence for $\alpha_w$. In the name of producing an efficient algorithm which is more simple to implement "out-of-the-box", we propose the following alternative.

The second approach we propose is similar to the "elite" samples of a cross-entropy algorithm (De Boer et al. [2005]). That is, at each generation, the algorithm accepts the best $r$ designs, based on the utility. At the next generation of the algorithm, we sample $m$ designs from the perturbation kernel from each of these $r$ designs. Similar to the first approach, one may sensibly propose an increasing number of samples $m_w$ at each generation in order to provide better resolution around the optimal design. Similarly, the number of "elite" samples can be altered at each generation, as is done in a fully-adaptive cross-entropy algorithm (De Boer et al. [2005]) – increasing exploitation at later stages of the algorithm at the cost of exploration. Given one dictates the number of samples that are accepted at each generation, $r_w$, and the number of new samples at each generation, $m_w$, this ensures full control over the number of designs considered at each generation of the algorithm, allowing more control over the computational effort. Thus, one may reasonably evaluate the optimal (or near-optimal) Bayesian design in a computationally efficient time-frame.

## 3.3 Perturbation Kernel

The perturbation kernel is used to sample new designs at each generation of the INSH algorithm. In the two examples we consider in this work, we use a truncated, multivariate-Normal distribution (where the dimension is given by the dimension of the design space, and the truncation is to ensure constraints are satisfied). One could alternatively sample from any symmetric distribution, centred on the current design points. A standard cross-entropy algorithm uses the accepted samples to define the mean and (co-)variance structure of a (multivariate-)Normal distribution, and all new samples are generated by this distribution. We prefer to avoid this approach, rather, allowing the region surrounding each accepted point to be explored individually. Combining all accepted samples into a single distribution from which to sample, may result in new samples not being generated in regions of high utility (for example, when considering multi-modal utility surfaces), and re-

quires re-evaluation of the (co-)variance matrix at each generation.

### 3.4 Stopping Criteria

A common feature of optimisation tools is a criterion for stopping the algorithm. It would be straight-forward for the user to implement a stopping criteria based on the change in utility of newly sampled designs at each iteration of the algorithm, based on the desired level of accuracy. In the examples in this work, we choose to demonstrate the algorithm by running it for a fixed number of iterations.

## 4 Examples

### 4.1 Markovian Death Model

Consider the Markovian death model as defined by Cook et al. [2008]. There is a population of $N$ individuals which, independently, move to an infectious class $I$ at constant rate $b_1$ – for example, due to infection from an environmental source. The Markov chain models the number of infectious individuals at time $t$, $I(t)$ (where the number of susceptible individuals is $S(t) = N - I(t)$). The positive transition rates of the Markov chain are given by $q_{i,i+1} = b_1(N - i)$, for $i = 0, \ldots, N - 1$. The prior distribution we consider is $b_1 \sim \log\text{-}N(-0.005, 0.01)$, chosen such that the mean lifetime of individuals in the population is one, with an approximate variance of 0.01 (as per Cook et al. [2008]).

The optimal experimental design for the Markovian Death model has previously been considered in a Bayesian framework by Cook et al. [2008], Drovandi and Pettitt [2013], and Price et al. [2016]. Cook et al. [2008] utilised the MCMC approach of Müller [1999], and used an exact posterior, hence, the designs of Cook et al. [2008] provide a gold-standard with which to compare our results. Drovandi and Pettitt [2013] also utilised the MCMC approach of Müller [1999], however, coupled with an approximate posterior distribution evaluated via an ABC approach. We note however, that the MCMC approach struggles to evaluate the optimal design once the dimension of the design space is more than four. This is due to the increasing computational difficulty associated with the evaluation of the mode of the multi-dimensional utility surface. Price et al. [2016] provided an exhaustive-search across a grid on the

design space, where the posterior distributions, and hence the utility, were evaluated using an ABC method. The INSH code for the death model is implemented in MATLAB R2015b.

## 4.2 Pharmacokinetic Model

Consider the pharmacokinetic experiment considered by Ryan et al. [2014] and Overstall and Woods [2016b]. In these pharmacokinetic experiments, individuals are administered a fixed amount of a drug. Blood samples are taken in order to understand the behaviour of the drug within the body.

Let $y_t$ represent the observed concentration of the drug at time $t$. We model the concentration as $y_t = \mu(t)(1+\epsilon_{1t})+\epsilon_{2t}$, where, $\mu(t) = \frac{D\theta_2}{\theta_3(\theta_2-\theta_1)}\left(e^{-\theta_1 t} - e^{-\theta_2 t}\right)$ is the mean concentration at time $t$, and $\epsilon_{1t} \sim N(0, \sigma^2_{prop})$, $\epsilon_{2t} \sim N(0, \sigma^2_{add})$, $\sigma^2_{prop} = 0.01$ and $\sigma^2_{add} = 0.1$. That is,

$$y_t \sim N\left(\mu(t), \sigma^2_{add} + \sigma^2_{prop}\mu(t)^2\right).$$

The blood samples are taken within the first 24 hours after the drug is administered (that is, $t \in [0, 24]$), and it is not practical to take blood samples less than 10-15 minutes apart (hence, $t_{i+1} - t_i \geq 0.25$). We wish to obtain information about the model parameters $\boldsymbol{\theta} = (\theta_1, \theta_2, \theta_3)$, where $\theta_1$ represents the first-order elimination rate constant, $\theta_2$ represents the first-order absorption rate constant, and $\theta_3$ represents the *volume of distribution* – a theoretical volume that a drug would have to occupy in order to provide the same concentration as is currently present in the blood plasma, assuming the drug is uniformly distributed (Ryan et al. [2014]).

As per Ryan et al. [2014] and Overstall and Woods [2016b], the model parameters $\boldsymbol{\theta} = (\theta_1, \theta_2, \theta_3)$ are assumed *a priori* to be independently, normally distributed on the log-scale, with mean $\log(0.1)$, $\log(1)$, and $\log(20)$ respectively, and variance 0.05. Overstall and Woods [2016b] demonstrate that the designs evaluated using their method are clearly better than those obtained via the dimension reduction schemes implemented by Ryan et al. [2014], and so we only compare our results to those of Overstall and Woods [2016b].

Duffull et al. [2012], McGree et al. [2012], Ryan et al. [2014], and Ryan et al. [2015b] have previously evaluated optimal Bayesian experimental designs for pharmacokinetic models, either for a few sampling times (less than five), or consider more observations through dimension reduction schemes

(e.g., search across the two-parameters of a Beta distribution, where the quantiles are scaled to give the observation times). Overstall and Woods [2016b] are currently the only example of a method efficient enough to establish optimal Bayesian designs for a system of this magnitude without implementing a dimension reduction scheme in a feasible amount of computation time.

Furthermore, we show how the output of the INSH algorithm can be used simply to construct *sampling windows* – a range of values for each observation, rather than a fixed value for each observation time. The motivation for sampling windows, comes from the potential difficulty associated with implementing what is the "optimal design" in a practical setting. Sampling windows allow practitioners (or those implementing the design) to conduct each sample within a range of values, which may not be as informative as the optimal, but potentially more informative than the practical design that ends up being implemented. Sampling windows have been considered previously for similar types of models, for example, in Green and Duffull [2003], Chenel et al. [2005], Graham and Aarons [2006], Duffull et al. [2012], and McGree et al. [2012], to name a few. As the output of the INSH algorithm consists of a large number of designs sampled around regions of high utility – as opposed to a single design, as in ACE – the construction of sampling windows is a simple extension to the algorithm.

The INSH algorithm for the pharmacokinetic example is implemented in R (version 3.3.0).

# 5 Results

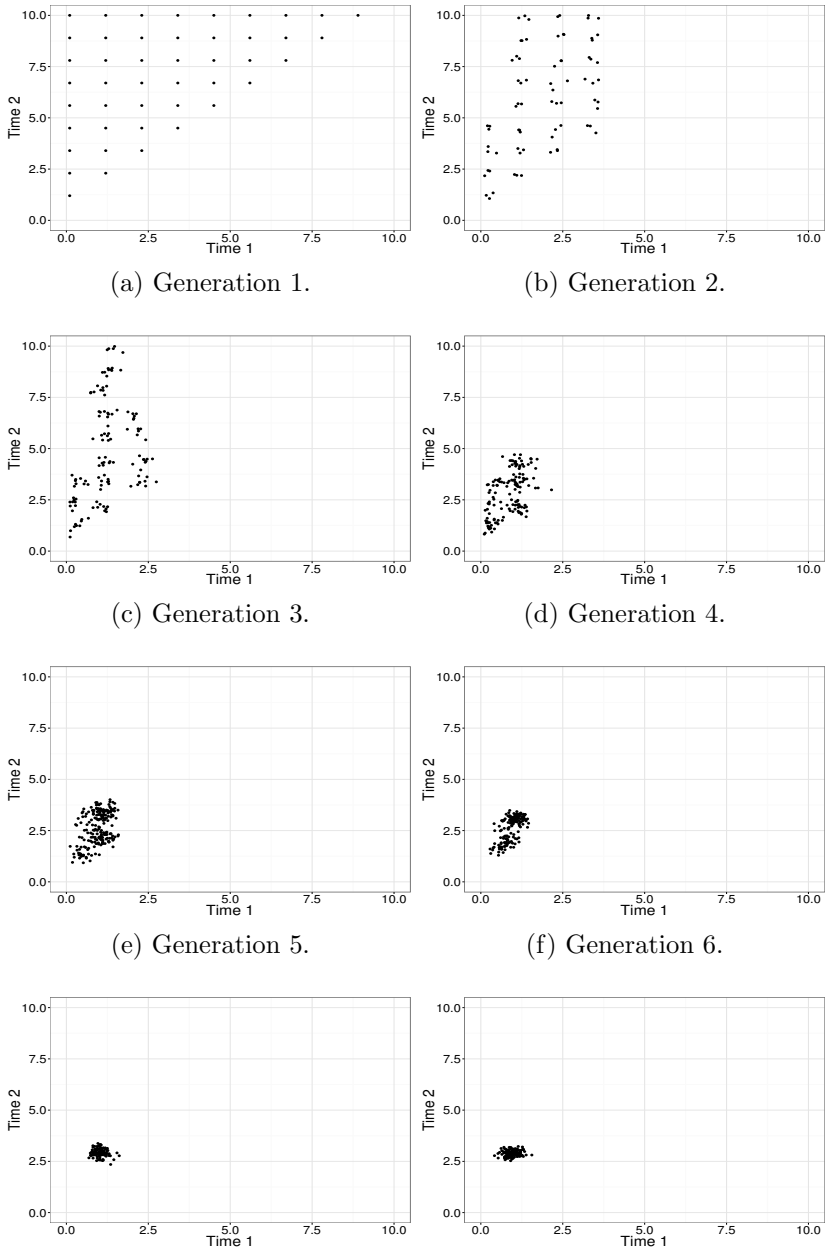
## 5.1 Markovian Death Model

We wish to determine the optimal observation times for the Markovian death model. We consider the optimal observation schedule when the number of observations permitted is $n = 1, 2, 3, 4, 6$ or 8. The designs and computation times for $n = 1, \ldots, 4$ observation times are compared to existing results. The observation schedules and computation times for 6 and 8 observations are also provided in order to demonstrate the efficiency of this algorithm for larger design spaces.

First, however, we demonstrate how the INSH algorithm works by considering two observation times for the death model. We choose to implement the

first approach to the acceptance criteria: that is, accept designs corresponding to a utility within an increasing proportional cut-off. We set $\alpha_w = 1 - a^w$, for $a = 0.6$, and $w = 1, \ldots, W$, where $W = 15$ (we define the sequence in this way solely for reproducibility – any values can be specified for $\alpha_w$). At each generation, $m = 3$ designs are sampled for each accepted design from the perturbation kernel – a Normal distribution centred on the accepted design, with fixed standard deviation 0.15 for each design parameter, and zero covariances (truncated subject to the design constraints, *i.e.*, $t_{i+1} - t_i > 0$, $i = 1, \ldots, n-1$).

Figure 1 shows the progression of the INSH algorithm at each of the first eight generations. For comparison, Figure 2a shows the full utility surface for the death model, evaluated using the ABCdE algorithm at all observation times across a grid with spacing 0.1, with $t_i \in [0.1, 10]$. We can clearly see the optimal design is on a ridge at approximately (0.9, 2.8). There is also a region of high utility around (0.7, 2.0). Regions of low utility exist for very early $t_1$ (and in particular, $t_2 > 4$), or where both $t_1$ and $t_2$ are high (e.g., both above 3.5). In Figure 1, Generation 2 (Figure 1b) clearly shows that regions of low utility are discarded early, and high utility regions are retained. Generations 2-6 (Figures 1b-1f) clearly demonstrate the convergence of the samples towards the region containing the optimal design. Generation 6 demonstrates the samples converging about the two "peaks" observed in Figure 2a – clearly demonstrating the ability to investigate multiple regions of high utility simultaneously. Figure 2b shows all design points considered throughout the INSH algorithm, with each point shaded by the utility value (darker corresponds to higher utility). The regions of high utility clearly have been sampled more thoroughly.



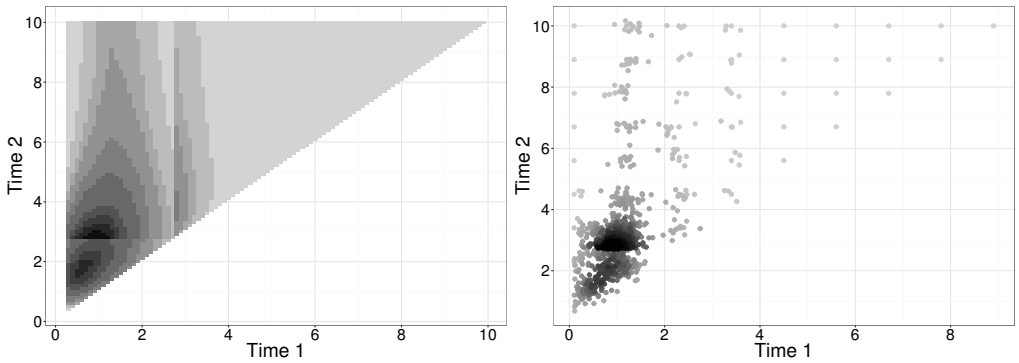

(a) Generation 1. (b) Generation 2.

(c) Generation 3. (d) Generation 4.

(e) Generation 5. (f) Generation 6.

(g) Generation 7. (h) Generation 8.

Figure 1: Demonstration of the design regions being considered by the INSH algorithm at each of the first eight generations.




(a) (b)

Figure 2: (a) Full utility surface for two observations of the death model evaluated using the ABCdE algorithm, where dark shaded regions indicate high utility. (b) Samples from every generation of the INSH algorithm for two observations of the death model, where darker points have higher utility values.

Online Resource B contains box-plots illustrating the convergence of the sampled observation times towards the optimal, and the corresponding utilities towards the maximum. Online Resource B also contains tables with the optimal experimental designs determined using the INSH algorithm compared to the existing methods (along with the corresponding INSH algorithm inputs), and the computation time compared to the ABCdE algorithm, respectively. These times are represented in Figure 3.

Figure 3 clearly depicts the improved efficiency of the INSH algorithm for large dimensional design spaces compared to the ABCdE algorithm. In particular, we note the increase in computation time is dependent only on the algorithm inputs, which can be controlled by the user, rather than the size of the design space. Note that the INSH algorithm is slower than the ABCdE algorithm for the same number of designs, as the simulations are created at each generation – in the ABCdE algorithm, all simulations are created in parallel, prior to running the algorithm. It is clear that the computation times of the INSH algorithm are much lower for more observation times compared to the ABCdE algorithm, as significantly less designs are considered.

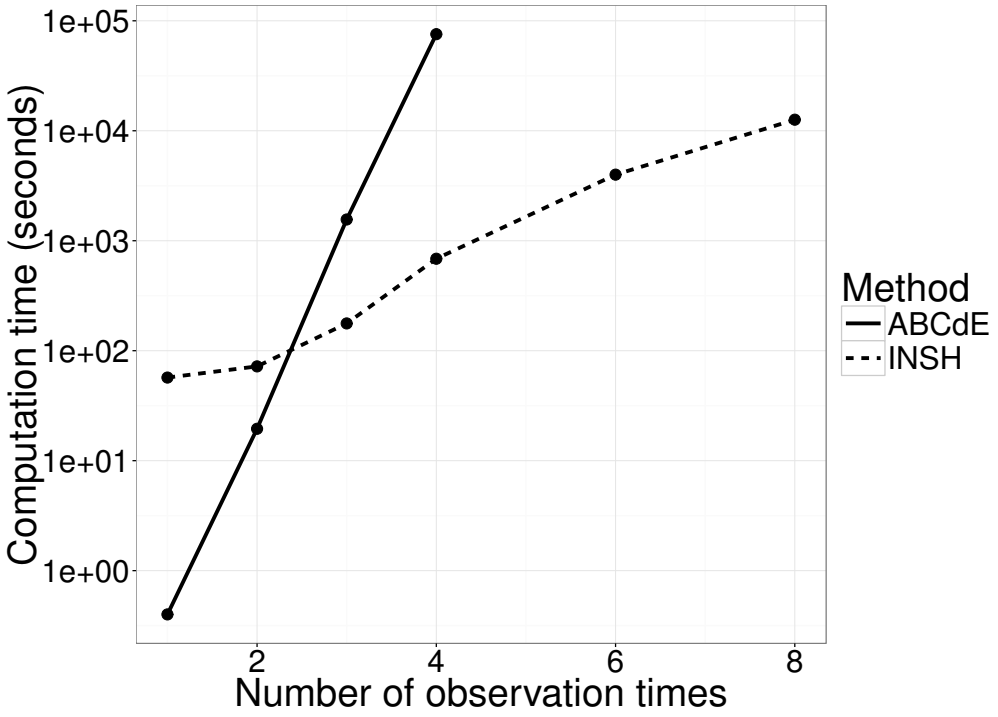



Figure 3: Comparison of the computation times for the ABCdE and INSH algorithms, to evaluate optimal designs for the Markovian death model.

## 5.2 Pharmacokinetic Model

We choose the acceptance criteria to be a fixed number of "elite" samples at each generation. That is, we accept at each generation of the algorithm the best $r_w$ designs, and sample $m_w$ designs from the perturbation kernel using these "best" $r_w$ designs. We step down the value of $r_w$, and increase the value of $m_w$ as $w$ increases, such that early iterations are geared towards exploration, while later iterations are focussed on exploitation. Due to the physical constraints on the frequency at which sampling can be performed (at least 15 minutes apart), we restrict the designs such that $t_{i+1} - t_i \geq 0.25$, $i = 1, \ldots, 14$. We sample designs from a multivariate-Normal perturbation kernel with fixed standard deviation 0.20, and zero covariance (truncated subject to the design constraints). The first generation of designs were sampled uniformly from the viable design space, [0, 24], such that the designs satisfied

the constraints.

In order to compare the run time of the ACE algorithm to the INSH algorithm, we implemented the ACE algorithm as detailed in Overstall and Woods [2016b], (*i.e.*, running 20 instances of the ACE algorithm from the `acebayes` package in (embarrassingly) parallel across four cores). On an iMac running OSX 10.11.4 with 4.0GHz Intel Core i7 processor and 32GB memory, this took 15.53 hours. We did not include the run time of the post-processing utility evaluation of the 20 candidate designs, 20 times each, in order to establish the overall optimal design, for reasons we state shortly.

The ACE algorithm for this example in Overstall and Woods [2016b] was performed 20 times from random initial conditions, each for a total of 20 iterations. Each iteration searches across each of the 15 dimensions of the design, and considers 20 candidate times to fit the Gaussian process. Thus, a total of 120,000 designs are considered (*i.e.*, utility evaluations) in the ACE algorithm, where 6,000 of these utility evaluations are completed using significantly more Monte Carlo simulations. Specifically, the utility for the 20 candidate times used to train the Gaussian process are evaluated using $\tilde{B} = B = 1{,}000$ Monte Carlo simulations, while the utility corresponding to the design with the proposed new observation time is evaluated using $\tilde{B} = B = 20{,}000$.

The advantage of the INSH algorithm is in the ability to consider a large number of designs in multiple regions, simultaneously. Hence, it is sufficient to use less effort to evaluate the utility of each design, as poorly-estimated utilities will have less influence on the output of the algorithm. Hence, we used $\tilde{B} = B = 5{,}000$ for the evaluation of the utility of each design, which was completed in parallel on four cores (using `foreach` and `doParallel` packages in R), on the same machine as stated above. We ran the INSH algorithm for $W = 40$ iterations, with 1200 randomly generated initial designs. At each iteration, we retained the "best" 150, 75, 50, 25, and 10 designs, and proposed two, four, six, 12 and 30 new designs around each accepted design, for 12 iterations of each combination (*i.e.*, $(r_1, r_2, \ldots, r_{12}, r_{13}, \ldots, r_{60}) = (150, 150, \ldots, 150, 75, \ldots, 10)$) – maintaining consideration of 300 designs at each iteration, while increasing the exploitation and reducing exploration at each iteration. This run of the INSH algorithm took approximately 2.23 hours (approximately 7 times faster than the ACE algorithm). Having obtained the designs and utility evaluations from the INSH algorithms, we perform the same post-processing utility evaluation on the 20 best considered designs, with 20 evaluations of the utility of each

design with $\tilde{B} = B = 20{,}000$, in order to identify the overall optimal.

The total number of designs considered by the INSH algorithm with this selection criteria is approximately: (No. initial designs)$+(W-1)\times r_w \times m_w = 1200 + (60-1) \times 300 = 18{,}900$. In practice, this number is often slightly higher, as the $r_w^{th}$ ranked design can be a tie, and the optimal design is re-introduced into the set of designs being considered if it occurred in a previous generation (this run of the INSH algorithm considered 19,428 designs). In contrast, if we were to attempt to evaluate the utility at every design across a grid with spacing 0.05 (e.g., as in the ABCdE algorithm of Price et al. [2016]), we would be required to evaluate $\binom{476}{15} \approx 8.93 \times 10^{27}$ designs – approximately $4.7 \times 10^{23}$ times as many designs as were considered by the INSH algorithm.

Figure 4 shows box-plots of the 20 utility evaluations for each of the 20 best designs that were considered by the INSH algorithm, compared to the same number of evaluations of the ACE optimal design reported in `optdescomp15sig()` in the `acebayes` package (each utility evaluation using $\tilde{B} = B = 20{,}000$). We can see from this figure that there are three designs that perform similarly well to the design found using the ACE algorithm. Online Resource C contains a table with summaries of the evaluated utilities for each design, a figure demonstrating convergence of the INSH algorithm to the optimal region, and a figure demonstrating how well each design performs with respect to inference (in particular, a comparison of the posterior variance, and the bias in posterior mode).

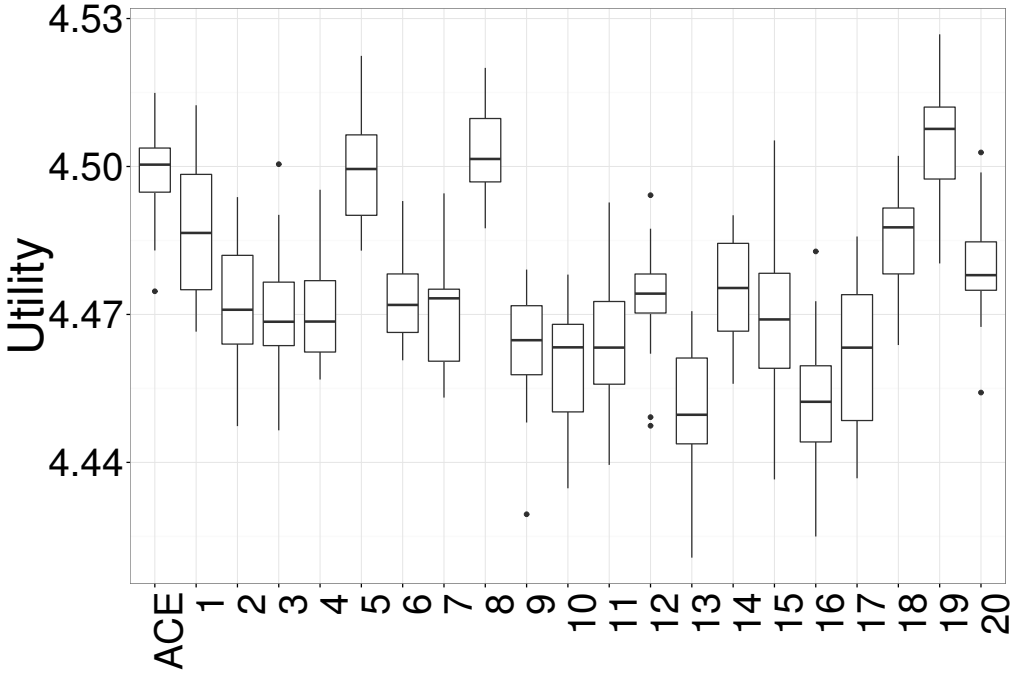



Figure 4: Box-plots of the utility for the optimal design found by the ACE algorithm, compared to the top 20 designs considered by the INSH algorithm. The utility of each design is evaluated 20 times, using $\tilde{B} = B = 20{,}000$ Monte Carlo simulations.

### 5.2.1 Sampling Windows

The nature of the INSH algorithm means that we retain a large number of designs with high utility. We use these "best" designs to construct the sampling windows for each sampling time, similar to the approach of McGree et al. [2012]. McGree et al. [2012] use percentiles of the designs evaluated once a stopping-criteria has been reached in their algorithm to form the sampling windows, whereas we just choose a fixed number of "best" designs to form the windows. Given the windows, the one implementing the design can then simply choose each observation time from these windows, ensuring that the physical constraint, $t_{i+1} - t_i \geq 0.25$, is satisfied.

As an example of this process, we arbitrarily consider the top 20 designs

from the output of the INSH algorithm for the pharmacokinetic example, and form sampling windows as the range of values considered at each observation time for these "best" designs. Alternatively, one could consider all designs that were within some percentage of the utility corresponding to the maximum, or, use a weighting based on the expected utility for each design to approximate a distribution for each sampling time which could subsequently be sampled. In order to construct the sampling window designs for the purpose of evaluating their utility, we "bootstrap" an observation schedule by randomly selecting each of the 15 sampling times (with equal probability), from the 20 candidate observation times, subject to the constraints. A new design is sampled for each of the 20 utility evaluations to demonstrate the range of potential outputs from this approach. Figures 5c and 5d show the sampling windows for each observation time, calculated using the output of the INSH algorithm. Figure 5a shows the optimal observation schedules evaluated using the ACE and INSH algorithms. Note that the optimal design returned from the INSH method, was that which corresponded to the $19^{th}$ highest utility value from the original output of the INSH algorithm (*i.e.*, using $\tilde{B} = B = 5{,}000$). It was deemed the optimal design as it corresponded to the highest mean utility, from 20 utliity evaluations using $\tilde{B} = B = 20{,}000$ (Figure 4). Figure 5b shows box-plots of 20 utility evaluations (using $\tilde{B} = B = 20{,}000$) for the ACE and INSH optimal designs, and the 20 randomly selected designs from the sampling windows. Note that the average efficiency of the sampling windows designs compared to the INSH optimal design is 99.07%.

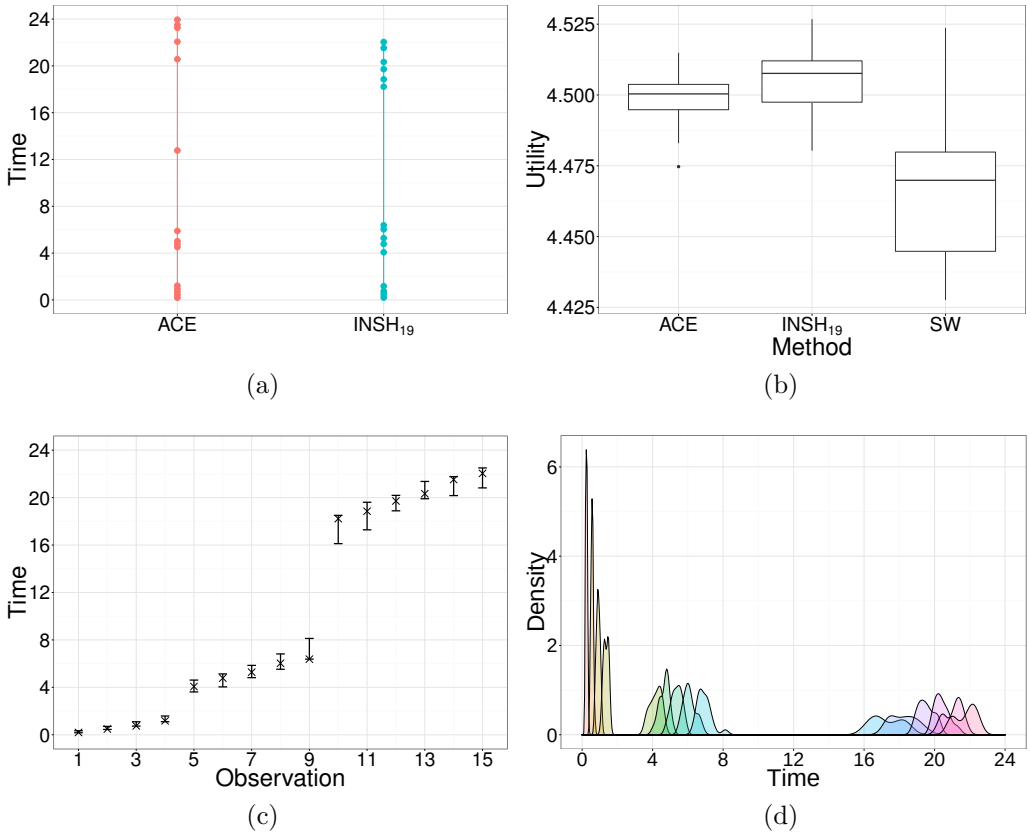



Figure 5: (a) Comparison of the resulting optimal designs from the ACE and INSH methods. (b) Boxplots of 20 utility evaluations for the ACE and INSH optimal designs and the sampling windows designs, using $\tilde{B} = B = 20000$. (c) Sampling windows for each observation time obtained from the 20 designs corresponding to the highest utilities found during the INSH algorithm. The error bars show the range for each observation time from the 20 designs, and the cross shows the corresponding observation from the optimal design. (d) Density plot of the sampling windows for each observation time.

# 6 Discussion

The results of the INSH algorithm applied to the death model indicates the suitability of this method, compared to existing methods. The resulting

optimal designs evaluated are near-identical to those evaluated by existing methods – especially ABCdE. This makes sense given the estimation of the utility at each design is performed using the same mechanism. We have omitted the comparison of designs with regards to inference, as the comparison between all existing methods was made in Price et al. [2016], and these new designs from INSH are not significantly different to those previously considered – the main purpose of this example was to demonstrate the computational efficiency. The computation time is clearly reliant on the choice of parameters for the algorithm, however the conservative choices made for the death model have clearly demonstrated a vast improvement in the computation time compared to the ABCdE method, while also allowing a simple approach to evaluated optimal experimental designs for more than four dimensional design spaces that are not practically feasible with the MCMC approaches. The larger computation time for more observation times for the death model is due to the evaluation of the utility. As previously stated, the evaluation of the utility is efficient due to the consideration of only unique data sets. As the number of observation times increases, the number of unique data sets increases, and hence evaluation of the utility for each design is more computationally demanding.

We used the INSH algorithm to evaluate the optimal Bayesian experimental designs for 15 observation times of the pharmacokinetic model. Previous work had successfully considered fewer observation times (e.g., Duffull et al. [2012], McGree et al. [2012]), or a re-parameterisation of the problem in order to obtain optimal designs for a large number of sampling times in a computationally feasible amount of time (Ryan et al. [2014]). Overstall and Woods [2016b] provided a significant step forward with their efficient approach to evaluating the optimal design for this model, with a high-dimension design space, that logically perform better than the re-parameterised designs considered previously. Here, we have used the INSH algorithm to evaluate Bayesian experimental designs for the pharmacokinetic model, which slightly outperform those evaluated by the ACE algorithm of Overstall and Woods [2016b]. Of particular significance is the small amount of computation time required to evaluate these designs – for the example we considered here, the INSH algorithm found a number of well-performing designs in approximately 14% of the computation time of the ACE algorithm.

For the pharmacokinetic example, we considered forming sampling windows for practical implementation of near-optimal designs. We showed that these sampling windows can be derived very simply from the output of the

INSH algorithm, with minimal post-processing. We showed that the sampling windows (chosen in a completely uninformed manner) provided a similar level of information to that provided by the optimal design evaluated from the INSH algorithm.

We note that in evaluating the optimal designs for the ACE algorithm, $\tilde{B} = B = 20{,}000$ was used. A more accurate estimate of the utility of each design is required in the ACE algorithm, as one design is considered at a time, and subsequent designs considered are dependent on whether or not the new design coordinate is accepted. An "extreme" utility evaluation could push the algorithm from a region of high-utility, and thus the algorithm could converge to, or conclude whilst still in, a sub-optimal region. The INSH algorithm considers a large number of designs simultaneously, and so a spurious utility evaluation has less of an impact on the overall output of the algorithm. By performing the post-processing of the "best" designs returned by the INSH algorithm – as is done for the ACE candidate designs as well – we can use a sufficiently large amount of effort to evaluate the utility of these designs and thus have greater confidence in their estimated utility.

We have not provided a proof that the INSH algorithm will converge to the optimal design, however, one can see that in the limit (*i.e.*, $W \to \infty$, $\alpha_0 = 0$ and $\alpha_w \to 1$ as $w \to \infty$, and sufficiently large $m$), the INSH algorithm will identify the optimal design. When searching for the design across a gridded design space, less restrictive conditions would be required to ensure convergence to the optimal solution. However, as with all optimisation routines, the aim of this algorithm is to find near-optimal designs in a computationally feasible amount of time. Thus, practical algorithm inputs must be chosen, which may not guarantee convergence to the optimal solution. However, this trade-off is apparent in a number of existing optimisation routines – for example, simulated-annealing, cross-entropy, and genetic algorithms all have the potential to converge to local, rather than global, optima. Furthermore, the authors believe that the choice of algorithm parameters, in particular those corresponding to the acceptance criteria (proportion of maximum utility $\alpha_w$, or number of "elite" samples $r$) and perturbation kernel (distribution $f(d \mid d')$, and number of new samples $m_w$), are quite intuitive, and thus, INSH is easier to implement than some existing stochastic optimisation methods. We also note that, as with other stochastic optimisation routines, some trial-and-error may be required in order to choose suitable values of these parameters for particular problem classes.

The INSH algorithm we have presented here is quite general, and there

exist many aspects of the algorithm which can be explored in order to improve the efficiency of the algorithm for different optimisation problems. For example, randomly incorporating a sample in a region of the design space that has either not been considered previously, or was dismissed earlier in the algorithm, may be a beneficial approach in order to increase exploration of the input space, and maximise the chances of obtaining the optimal solution. Another important consideration will be to provide some general rules regarding the choice of algorithm inputs for a particular utility surface. For example, the initial samples could be used to approximate some characteristics of the utility surface, and provide some insight into sensible choices of the inputs for the algorithm. While we did not consider it here, increasing the number of utility evaluations which form the approximate expected utility could also be increased as the algorithm progresses, *i.e.*, specify a sequence for $\tilde{B}$ and $B$ in the SIG utility evaluation – ensuring more effort is spent evaluating a more precise estimate of the utility in regions near to the optimal design. One could also incorporate the noise in the utility evaluation in the acceptance criteria – discarding only those samples that have an upper limit below the threshold value.

Bayesian optimal experimental design has immense potential to inform the collection of data, so as to subsequently enhance our understanding of a variety of processes. However, a major impediment is the difficulty in evaluating optimal designs for problems with large, or high-dimensional, design spaces; attempts at evaluating optimal experimental designs prior to the ACE algorithm of Overstall and Woods [2016b] have been restricted to considering no more than several design parameters, and typically, for relatively simple models. Here, we have proposed an algorithm which improves the efficiency of the search across the design space, even compared to that of the ACE algorithm. While it still requires identification of an efficient approach to evaluating the utility, this improvement in the search aspect of determining the optimal design is extremely beneficial. In particular, coupled with advances in parallel computing – both in power and accessibility – this algorithm should prove to be a significant step towards the wide spread use of Bayesian optimal experimental design.

## Acknowledgements

DJP acknowledges the support of the BBSRC (BB/M020193/1) awarded to Olivier Restif. JVR acknowledges the support of the ARC (Future Fellowship FT130100254; CoE ACEMS) and the NHMRC (CRE PRISM$^2$).

## References

Analytics, R., Weston, S., 2015a. doParallel: Foreach Parallel Adaptor for the 'parallel' package. R package version 1.0.10.
URL `https://CRAN.R-project.org/package=doParallel`

Analytics, R., Weston, S., 2015b. foreach: Provides Foreach Looping Construct for R. R package version 1.4.3.
URL `https://CRAN.R-project.org/package=foreach`

Berry, D. A., 2004. Bayesian statistics and the efficiency and ethics of clinical trials. Statistical Science 19, 175–187.

Chenel, M., Ogungbenro, K., Duval, V., Laveille, C., Jochemsen, R., Aarons, L., 2005. Optimal blood sampling time windows for parameter estimation using a population approach: Design of a phase ii clinical trial. Journal of Pharmacokinetics and Pharmacodynamics 32.

Cook, A. R., Gibson, G. J., Gilligan, C. A., 2008. Optimal observation times in experimental epidemic processes. Biometrics 64 (3), 860–868.

De Boer, P., Kroese, D. P., Mannor, S., Rubinstein, R. Y., 2005. A tutorial on the cross-entropy method. Annals of Operations Research 134, 19–67.

Drovandi, C., Pettitt, A., 2013. Bayesian experimental design for models with intractable likelihoods. Biometrics 69, 937–948.

Duffull, S. B., Graham, G., Mengersen, K., Eccleston, J., 2012. Evaluation of the pre-posterior distribution of optimized sampling times for the design of pharmacokinetic studies. Journal of Biopharmaceutical Statistics 22, 16–29.

Evans, G. E., Keith, J. M., Kroese, D. P., Dec. 2007. Parallel cross-entropy optimization. In: Simulation Conference, 2007 Winter. pp. 2196–2202.

Faller, D., Klingmüller, U., Timmer, J., 2003. Simulation methods for optimal experimental design in systems biology. Simulation 79, 717–725.

Graham, G., Aarons, L., 2006. Optimum blood sampling time windows for parameter estimation in population pharmacokinetic experiments. Statistics in Medicine 25, 4004–4019.

Green, B., Duffull, S. B., 2003. Prospective evaluation of a D-optimal designed population pharmacokinetic study. Journal of Pharmacokinetics and Pharmacodynamics 30.

Kullback, S., Leibler, R. A., 1951. On information and sufficiency. The Annals of Mathematical Statistics 22, 79–86.

McGree, J. M., Drovandi, C. C., Pettitt, A. N., 2012. A sequential Monte Carlo approach to derive sampling times and windows for population pharmacokinetic studies. Journal of Pharmacokinetics and Pharmacodynamics 39, 519–526.

Müller, P., 1999. Simulation based optimal design. In: Bernardo, J. (Ed.), Bayesian Statistics. Oxford University Press, pp. 459–474.

Overstall, A. M., Woods, D. C., 2016a. acebayes: Optimal Bayesian Experimental Design using the ACE algorithm. R package version 1.2.
URL `https://CRAN.R-project.org/package=acebayes`

Overstall, A. M., Woods, D. C., 2016b. Bayesian design of experiments using approximate coordinate exchange. Technometrics.
URL `http://dx.doi.org/10.1080/00401706.2016.1251495`

Pagendam, D. E., Pollett, P. K., 2013. Optimal design of experimental epidemics. Journal of Statistical Planning and Inference 143 (3), 563–572.

Price, D. J., Bean, N. G., Ross, J. V., Tuke, J., 2016. On the efficient determination of optimal Bayesian experimental designs using ABC: A case study in optimal observation of epidemics. Journal of Statistical Planning and Inference 172, 1–15.

R Core Team, 2016. R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria.
URL `https://www.R-project.org/`

Rutenbar, R. A., 1989. Simulated annealing algorithms: An overview. IEEE Circuits and Devices Magazine 5.

Ryan, E. G., Drovandi, C., McGree, J. M., Pettitt, A., 2015a. A review of modern computational algorithms for Bayesian optimal design. International Statistics Review.

Ryan, E. G., Drovandi, C. C., Pettitt, A. N., 2015b. Fully Bayesian experimental design for pharmacokinetic studies. Entropy 17, 1063–1089.

Ryan, E. G., Drovandi, C. C., Thompson, M. H., Pettitt, A. N., 2014. Towards Bayesian experimental design for nonlinear models that require a large number of sampling times. Computational Statistics & Data Analysis 70, 45–60.

# Supplementary Materials for An Induced Natural Selection Heuristic for Evaluating Optimal Bayesian Experimental Designs

David J. Price[1], Nigel G. Bean[2,3], Joshua V. Ross[2], Jonathan Tuke[2]

[1]Disease Dynamics Unit, Department of Veterinary Medicine, University of Cambridge, Cambridge CB3 0ES, U.K.,
[2]School of Mathematical Sciences, University of Adelaide, SA 5005, Australia
[3] ARC Centre of Excellence for Mathematical & Statistical Frontiers

Appendix A contains existing algorithms referenced throughout the main text. Appendix B contains the results, tables and figures corresponding to the Markovian death model, as well as some discussion of the choice of inputs to the INSH algorithm. Appendix C contains the results, tables and figures corresponding to the pharmacokinetic model. Appendix D contains figures and tables corresponding to the sampling windows for the pharmacokinetic model.

# 7 Appendix A

*Existing Algorithms*

Algorithm 2 describes the ABC algorithm used by ABCdE and the INSH algorithm (for the death model) to evaluate the posterior distribution.

**Algorithm 2** ABC Algorithm: Fixed tolerance

**Input:** Observed data $\boldsymbol{x}$, simulated data $\boldsymbol{y} = (\boldsymbol{y}^1, \ldots, \boldsymbol{y}^N)$, corresponding parameter values $\boldsymbol{\theta}^i, i = 1, \ldots, N$, and tolerance $\epsilon$.

1: Evaluate discrepancies $\rho^i = \rho(\boldsymbol{x}, \boldsymbol{y}^i)$, creating particles $\{\boldsymbol{\theta}^i, \rho^i\}$ for $i = 1, \ldots, N$.

2: Using the posterior sample of parameters $\boldsymbol{\theta}^i$ such that $\rho^i < \epsilon$, evaluate utility.

**Output:** Utility for current design, having observed $\boldsymbol{x}$, $U(d, \boldsymbol{x})$.

Algorithm 3 describes the ABCdE algorithm of Price et al. [2016], to evaluate the optimal Bayesian experimental design.

**Algorithm 3** ABCdE Algorithm

1: Choose grid over the parameter space for the discrete estimate of the utility, number of simulations $N_{pre}$, and tolerance $\epsilon$.
2: Sample $N_{pre}$ parameters $\boldsymbol{\theta}$ from $p(\boldsymbol{\theta})$.
3: For each of the $N_{pre}$ parameters, and under every design $d$ in the design space $\mathcal{D}$, simulate process and store $X_{N_{pre}\times|\mathcal{D}|}(\boldsymbol{\theta}, d)$.
4: **for** $i = 1$ to $|\mathcal{D}|$ **do**
5: Consider the unique rows of data $Y(\boldsymbol{\theta}, d^i) =$ unique$(X(\boldsymbol{\theta}, d^i))$. *Note: We let $K^i$ be the number of such unique data, and $n_{k^i}$ be the number of repetitions of the $k^{i^{th}}$ unique data, for $k^i = 1, \ldots, K^i$.*
6: **for** $k^i = 1$ to $K^i$ **do**
7: Pass 'observed data' $\boldsymbol{y}^{k^i} = [Y(\boldsymbol{\theta}, d^i)]_{k^i}$, 'simulated data' $X(\boldsymbol{\theta}, d^i)$, $N_{pre}$ sampled parameters, and tolerance $\epsilon$ to Algorithm 2, and return contribution $U(\boldsymbol{y}^{k^i}, d^i)$ to the expected utility, for $k^{i^{th}}$ unique datum ('observed data') and $i^{th}$ design.
8: **end for**
9: Store $u(d^i) = \frac{1}{N_{pre}} \sum_{k^i} n_{k^i} U(\boldsymbol{y}^{k^i}, d^i)$; the average utility over all parameters and data for design $d^i$.
10: **end for**
**Output:** The optimal design $d^* = \underset{d\in\mathcal{D}}{\text{argmax}}(u(d))$.

Algorithm 4 details the MCMC algorithm for determining Bayesian optimal designs proposed by Muller [1999].

# 8 Appendix B

*Markovian Death Model ABC Choices*

We provide the parameter choices for the ABC algorithm used to evaluate the approximate posterior distributions when evaluating the utility for the Markovian death model example. Prior to running the ABC algorithm (Algorithm 2), we sample $N = 100,000$ parameter values from the prior distribution, and simulate data corresponding to each under each design. For each of 1, 2, 3, 4, 6, and 8 observation times, we use a tolerance of 0.25, 0.50, 0.75, 1.00, 1.50, 1.50, respectively.

We note however, that these choices are problem specific, and suggest that researchers undertake a pilot-study in order to determine sensible parameter

**Algorithm 4** MCMC with stationary distribution $h(\boldsymbol{\theta}, \boldsymbol{x}, d)$, Muller [1999]

**Input:** Number of samples $m$, prior distribution of model parameters $p(\boldsymbol{\theta})$, and proposal density $q(\cdot)$.

Choose, or simulate an initial design, $d^1$.
Sample $\boldsymbol{\theta}^1 \sim p(\boldsymbol{\theta})$, simulate $\boldsymbol{x}^1 \sim p(\boldsymbol{x} \mid \boldsymbol{\theta}^1, d^1)$, and evaluate $u^1 = U(\boldsymbol{\theta}^1, \boldsymbol{x}^1, d^1)$.
**for** $i = 1 : m$ **do**
    Generate a candidate design, $\tilde{d}$, from a proposal density $q(\tilde{d} \mid d^i)$.
    Sample $\tilde{\boldsymbol{\theta}} \sim p(\boldsymbol{\theta})$, simulate $\tilde{\boldsymbol{x}} \sim p(\boldsymbol{x} \mid \tilde{\boldsymbol{\theta}}, \tilde{d})$, and evaluate $\tilde{u} = U(\tilde{\boldsymbol{\theta}}, \tilde{\boldsymbol{x}}, \tilde{d})$.
    Calculate,

$$\alpha = \min \left\{ 1, \frac{\tilde{u}\ q(d^i \mid \tilde{d})}{u^i\ q(\tilde{d} \mid d^i)} \right\}.$$

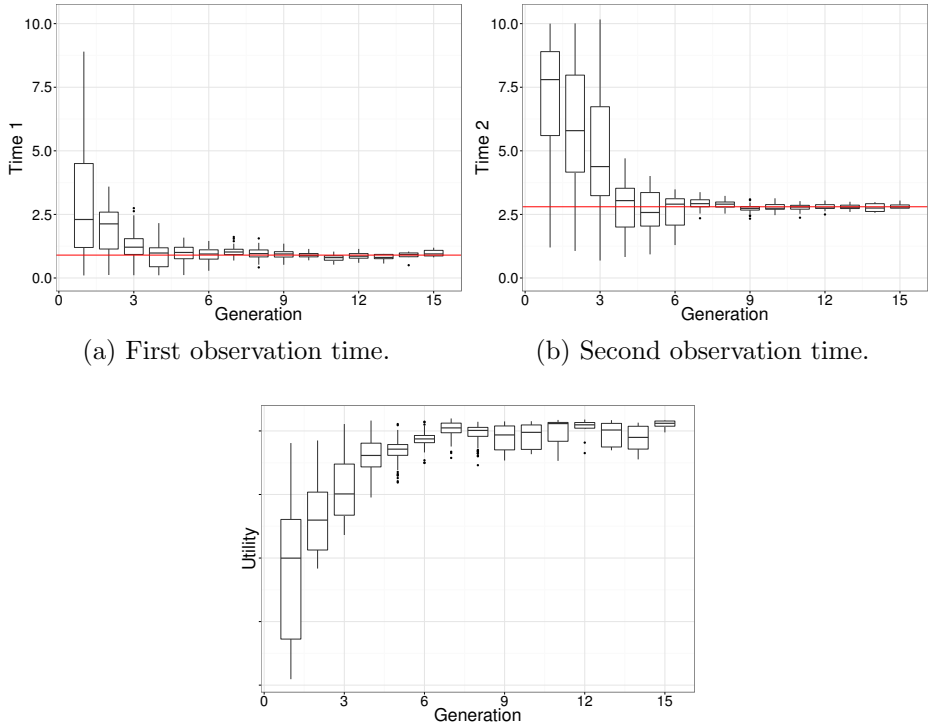
    Generate $a \sim U(0, 1)$
    **if** $a < \alpha$ **then**
        Set $(d^{i+1}, u^{i+1}) = (\tilde{d}, \tilde{u})$
    **else**
        Set $(d^{i+1}, u^{i+1}) = (d^i, u^i)$
    **end if**
**end for**

**Output:** Sample of $m$ designs, $d$.

choices, as one would do prior to using ABC for inference.

*Markovian Death Model Results*

Figure 6 demonstrates the convergence of the INSH algorithm to the optimal observation times, and the maximum utility.




(a) First observation time.

(b) Second observation time.

(c) Utility.

Figure 6: Boxplots of the two observation times, and the utility corresponding to the considered designs at each generation of the INSH algorithm. The horizontal lines in (a) and (b) correspond to the optimal observation times evaluated using the ABCdE method.

Table 1: Comparison of the optimal observation times for the death process, from Cook et al. [2008], Drovandi and Pettitt [2013], Price et al. [2016], and the INSH algorithm. $|t|$ is the pre-determined number of observation times, and $i$ is the $i^{th}$ time.

| | | Design Method | | | |
|---|---|---|---|---|---|
| $\|t\|$ | $i$ | Cook, *et al.* | Drovandi & Pettitt | ABCdE | INSH |
| 1 | 1 | 1.70 | 1.60 | 1.50 | 1.55 |
| 2 | 1 | 0.90 | 1.15 | 0.80 | 0.90 |
| - | 2 | 2.40 | 3.05 | 2.80 | 2.80 |
| 3 | 1 | 0.70 | 0.75 | 0.40 | 0.45 |
| - | 2 | 1.50 | 1.90 | 1.30 | 1.25 |
| - | 3 | 2.90 | 3.90 | 2.60 | 2.65 |
| 4 | 1 | 0.80 | 0.75 | 0.30 | 0.30 |
| - | 2 | 1.70 | 1.70 | 0.70 | 0.65 |
| - | 3 | 3.10 | 2.75 | 1.30 | 1.50 |
| - | 4 | 5.30 | 4.35 | 2.70 | 2.75 |
| 6 | (1,2) | - | - | - | (0.10,0.20) |
| - | (3,4) | - | - | - | (0.40,0.90) |
| - | (5,6) | - | - | - | (1.60,3.00) |
| 8 | (1,2) | - | - | - | (0.10,0.20) |
| - | (3,4) | - | - | - | (0.40,0.60) |
| - | (5,6) | - | - | - | (1.00,1.60) |
| - | (7,8) | - | - | - | (2.10,3.20) |

Table 1 contains the optimal experimental designs for different numbers of observations of the Markovian death model, evaluated by Cook et al. [2008], Drovandi and Pettitt [2013], Price et al. [2016], and the INSH algorithm (each where appropriate).

Table 2 gives a comparison of the computation times for the ABCdE algorithm of Price et al. [2016], compared to the INSH algorithm. These times are represented in Figure 3 in the main text.

Note, we define the number of offspring at each generation $m_w$, to be a non-decreasing function. As the algorithm converges towards the optimal design, it is desirable that we consider more designs in order to better explore the region surrounding the optimal design. As the algorithm converges

towards the optimal, there are typically fewer designs being accepted, and so producing more offspring at each design is no significant burden computationally. We choose to increase the number of designs at each generation according to the function $\lfloor m \times \exp(p \times w) \rfloor$, for $w = 1, \ldots, W$, where $m$ is the initial number generated at early generations, and $p$ governs the rate at which the number of offspring increases at each generation. The parameter $p$ is co-dependent on $W$, and so we reduce $p$ when $W$ is large in order to slow the rate at which $m_w$ increases. The floor is taken to ensure an integer number of points are generated. Similarly, we choose to slow the rate of convergence of the increasing cut-off for the utility at each generation, from $1 - a^w$, to $1 - a^{\sqrt{w}}, w = 1, \ldots, W$, to ensure the design space is not reduced too rapidly, and thus increase exploration of the design space in the example where eight observation times are considered. We note that there is no need for the cutoff values $\rho$, or number of new samples at each generation $m$, to be defined by a smooth function: one can simply input a set of discrete values. We have chosen to define these values via functions for the purposes of simplicity. The initial grid in each example is all combinations of times from 0.75 to 10, in steps of 0.75, subject to the constraint, $t_{i+1} > t_i$. The speed at which the cut-off $\rho_w$ converges to one and the size of the initial grid can both be adjusted in order to ensure that the optimal design is found — a fine initial grid means that a larger portion of the design space is explored initially, and hence the cut-off can increase faster; whereas a coarse grid means that there are potentially regions of high utility that are not observed and so the cut-off should increase slower so as to provide the opportunity for more points to be sampled about the existing points. A similar argument holds for the variance of the perturbation kernel – a smaller variance means more effort will be spent searching near to accepted designs, while a larger variance means that greater regions of the design space will be explored.

Table 2: Computation times for the ABCdE algorithm compared to the INSH algorithm for the Death model, with each observation time in [0.1,6] for ABCdE, and [0.1,10] for INSH. Computation times are from an iMac with 4GHz Quad-core Intel Core i7 Processor, 32GB 1867MHz DDR3 memory.

| | Computation Time (No. Designs Considered) | | INSH Parameters |
|---|---|---|---|
| $\|t\|$ | ABCdE | INSH | $(W, a, m_w, \sigma, \Delta t)$ |
| 1 | 0.4 secs (60) | 57.00 secs (152) | (10, 0.45, $\lfloor 2\exp(0.1w)\rfloor$, 0.1, 0.05) |
| 2 | 19.5 secs (1770) | 72.00 secs (464) | (10, 0.45, $\lfloor 3\exp(0.1w)\rfloor$, 0.1, 0.05) |
| 3 | 26.0 mins (34220) | 2.95 mins (1058) | (15, 0.50, $\lfloor 3\exp(0.1w)\rfloor$, 0.1, 0.05) |
| 4 | 21.0 hours (487635) | 0.19 hours (2170) | (20, 0.50, $\lfloor 3\exp(0.1w)\rfloor$, 0.1, 0.05) |
| 6 | - | 1.10 hours (4100) | (25, 0.50, $\lfloor 3\exp(0.1w)\rfloor$), 0.1, 0.05) |
| 8 | - | 3.50 hours (4131) | (50, 0.50, $\lfloor 3\exp(0.05w)\rfloor$), 0.1, 0.05) |

# 9 Appendix C

Figure 7 demonstrates the mean concentrations over time of the pharmacokinetic model evaluated for 50 parameter sets sampled from the prior distribution.

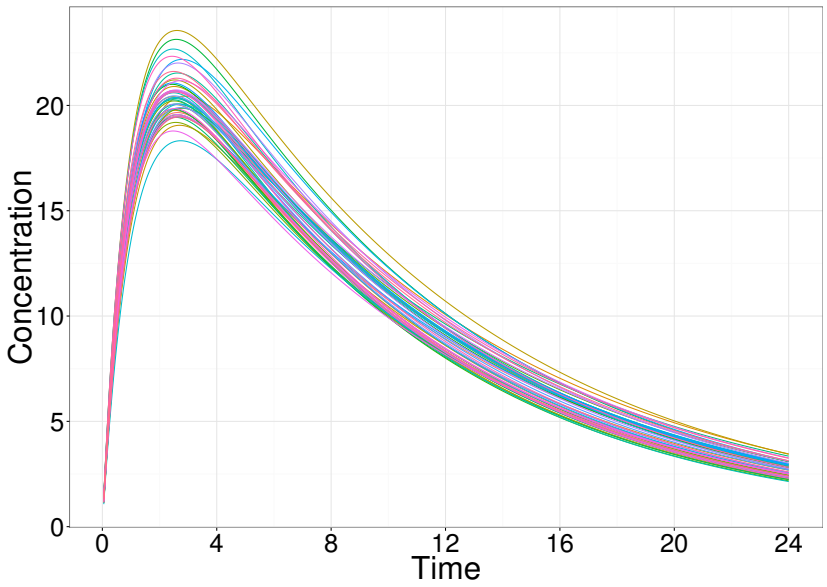



Figure 7: Plot of 50 mean concentrations over time of the pharmacokinetic model simulated using values sampled from the prior distribution.

Table 3 gives the 15 optimal observation times from the top three designs considered by the INSH algorithm. Each chosen optimal design shows the same pattern – four early observation times ($< 1.2$), followed by a cluster of observation times around 4-7, and the remaining observations grouped together towards the final permitted time.

Table 3: Three best sampling schedules evaluated from the INSH algorithm for the pharmacokinetic model.

| Original rank | Design |
|---|---|
| 19 | $(0.1961, 0.4840, 0.7506, 1.176, 4.069, 4.780, 5.281,$ $6.030, 6.377, 18.22, 18.85, 19.72, 20.33, 21.52, 22.04)$ |
| 2 | $(0.2460, 0.5054, 0.8017, 1.211, 4.035, 4.477, 5.173, 6.101,$ $6.632, 17.82, 18.63, 19.71, 20.32, 21.57, 21.98)$ |
| 3 | $(0.1989, 0.4801, 0.7778, 1.103, 4.465, 4.754, 5.776, 6.270,$ $6.754, 18.50, 18.99, 20.19, 20.87, 21.16, 21.87)$ |

Figure 8 shows box plots of the observation times, and the utility evaluations of the corresponding designs considered at each wave of the INSH algorithm. The figure for $t_9$, for example, depicts the ability of the INSH algorithm to search multiple regions simultaneously. In particular, iterations 17-27 are considering observation times in approximately three clusters – around times of 5, 12 and 15.

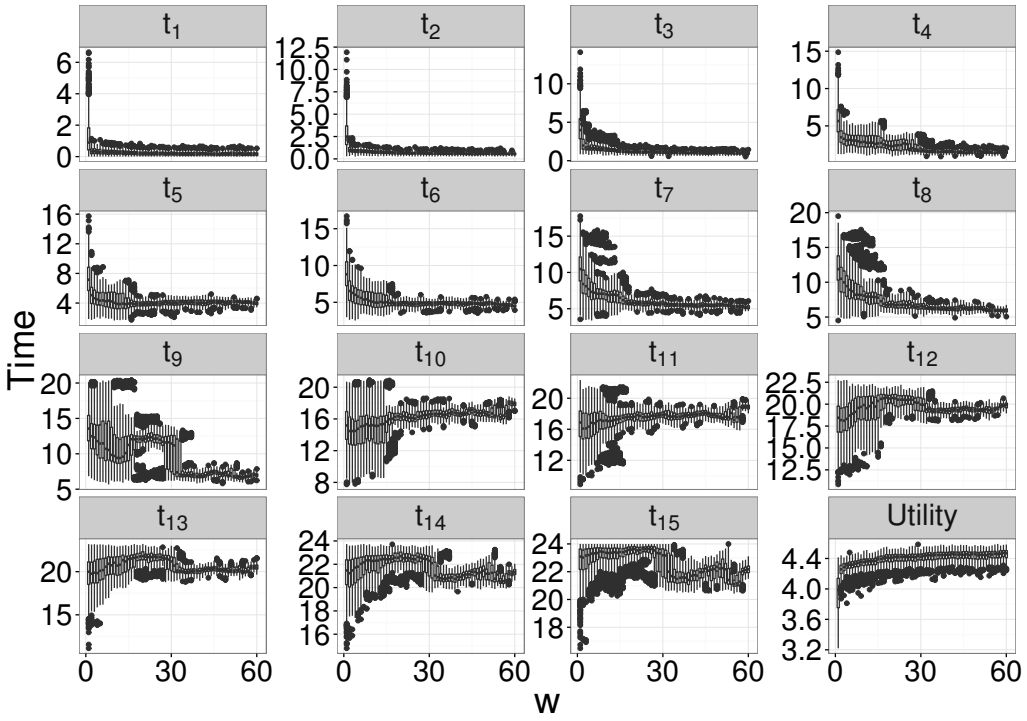



Figure 8: Figure showing the convergence of the sampled designs towards the region near the optimal design. Each panel represents an individual aspect of the sampled designs, the x-axis is the iteration of the INSH algorithm, and the y-axis is the value of the design aspect. The final panel shows the utilities corresponding to the sampled utilities.

Table 4 contains the estimated expected utility, median utility, and the $10^{th}$ and $90^{th}$ percentiles, corresponding to each of the top 20 designs considered by INSH, and the optimal returned by the ACE algorithm.

Table 4: Summary statistics of estimated utilities corresponding to the top 20 designs from the INSH algorithm, and the optimal design returned by the ACE algorithm. The design highlighted in bold is the design considered to be the optimal from the INSH algorithm.

| | Utility | | | |
|---|---|---|---|---|
| Design | Mean | Median | 10% | 90% |
| ACE | 4.4987 | 4.5004 | 4.4844 | 4.5102 |
| 1 | 4.4874 | 4.4865 | 4.4715 | 4.5040 |
| 2 | 4.4725 | 4.4710 | 4.4596 | 4.4894 |
| 3 | 4.4707 | 4.4685 | 4.4598 | 4.4864 |
| 4 | 4.4700 | 4.4686 | 4.4576 | 4.4835 |
| 5 | 4.4991 | 4.4995 | 4.4870 | 4.5111 |
| 6 | 4.4739 | 4.4719 | 4.4638 | 4.4900 |
| 7 | 4.4707 | 4.4733 | 4.4558 | 4.4866 |
| 8 | 4.5034 | 4.5015 | 4.4956 | 4.5156 |
| 9 | 4.4633 | 4.4648 | 4.4506 | 4.4758 |
| 10 | 4.4595 | 4.4633 | 4.4444 | 4.4736 |
| 11 | 4.4652 | 4.4633 | 4.4526 | 4.4803 |
| 12 | 4.4733 | 4.4742 | 4.4608 | 4.4868 |
| 13 | 4.4508 | 4.4497 | 4.4349 | 4.4654 |
| 14 | 4.4748 | 4.4754 | 4.4616 | 4.4878 |
| 15 | 4.4702 | 4.4690 | 4.4527 | 4.4941 |
| 16 | 4.4537 | 4.4523 | 4.4426 | 4.4725 |
| 17 | 4.4625 | 4.4633 | 4.4439 | 4.4846 |
| 18 | 4.4853 | 4.4877 | 4.4702 | 4.4991 |
| **19** | **4.5052** | **4.5076** | **4.4866** | **4.5204** |
| 20 | 4.4799 | 4.4780 | 4.4676 | 4.4975 |

Figure 9 provides a comparison of the inferential performance of the two optimal designs – corresponding to INSH and ACE – with regards to bias in a point estimate, and the posterior variance. We simulated 100 experiments from random parameters drawn from the prior distribution, and evaluated an approximate posterior distribution using a Metropolis-Hastings algorithm (retaining 100,000 samples from the posterior, following a burn-in of 10,000). The bias is estimated as the difference between the MAP (*maximum a posteriori*) estimate and the true parameter value that created the simulated data. Recall, the prior variance was 0.05 for each parameter.

It appears as though the design evaluated by the INSH algorithm performs marginally better with respect to the posterior variance – that is, the estimated variances are slightly lower for each parameter. The bias in the parameter estimates appears roughly equivalent between the two designs.

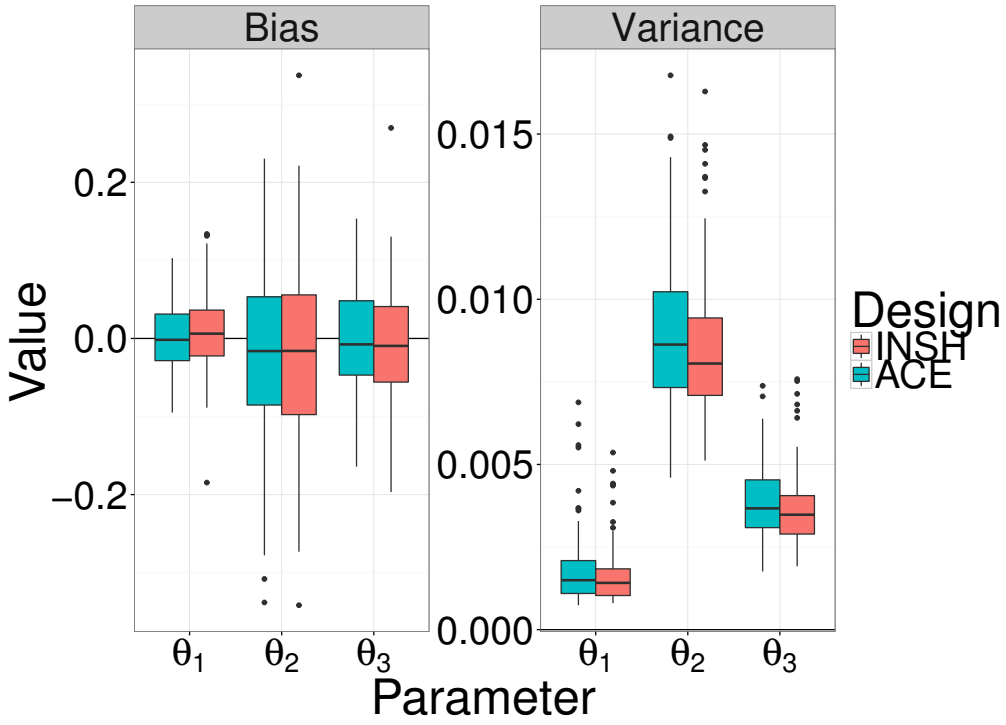



Figure 9: Comparison of the bias in MAP estimate, and posterior variance of each parameter in the pharmacokinetic model.

# References

A. R. Cook, G. J. Gibson, and C. A. Gilligan. Optimal observation times in experimental epidemic processes. *Biometrics*, 64(3):860–868, 2008.

C.C. Drovandi and A.N. Pettitt. Bayesian experimental design for models with intractable likelihoods. *Biometrics*, 69:937–948, 2013.

D. J. Price, N. G. Bean, J. V. Ross, and J. Tuke. On the efficient determination of optimal Bayesian experimental designs using ABC: A case study in optimal observation of epidemics. *Journal of Statistical Planning and Inference*, 172:1–15, 2016.