# Supplementary Materials for A Scalable Gradient Free Method for Bayesian Experimental Design with Implicit Models

## 1 Reference MI calculation using a nested Monte Carlo method

In this work, we use a nested Monte Carlo method to compute a reference mutual information (MI) value at the optimal design $\boldsymbol{\xi}^*$. The nested Monte Carlo method provides an average of MI value and an approximation to the likelihood $p(\boldsymbol{y} \mid \boldsymbol{\xi}^*, \boldsymbol{\theta})$.

Assuming $D$ independent measurement at $\boldsymbol{\xi}^* = [\xi_1^*, ..., \xi_D^*]^\top$, the likelihood can be written by:

$$p(\boldsymbol{y}|\boldsymbol{\xi}^*, \boldsymbol{\theta}) = \prod_{j=1}^{D} p(y_j \mid \xi_j^*, \boldsymbol{\theta}). \tag{1}$$

Therefore we can use Eq (1) and a sample-average of the marginal $p(\boldsymbol{y} \mid \boldsymbol{\xi}^*)$ to approximate the MI:

$$I_{\text{MI}}(\boldsymbol{\xi}^*) \approx \frac{1}{N_1} \sum_{i=1}^{N_1} \left[ \log \frac{\prod_{j=1}^{D} p(y_j^{(i)} \mid \xi_j^*, \boldsymbol{\theta}^{(i)})}{1/N_2 \sum_{k=1}^{N_2} \prod_{j=1}^{D} p(y_j^{(i)} \mid \xi_j^*, \boldsymbol{\theta}^{(k)})} \right] \tag{2}$$

where $y_j^{(i)}$ are data drawn from $p(y_j \mid \xi_j^*, \boldsymbol{\theta}^{(i)})$, $\boldsymbol{\theta}^{(i)}$ and $\boldsymbol{\theta}^{(k)}$ are prior samples drawn from $p(\boldsymbol{\theta})$.

### 1.1 Noisy linear model

For the noisy linear model, the source of noise are $\boldsymbol{\epsilon} \sim \mathcal{N}(0, 1)$ and $\boldsymbol{\nu} \sim \Gamma(2, 2)$, so the probability distribution of $\boldsymbol{\epsilon} + \boldsymbol{\nu}$ can be approximated via kernel density estimate (KDE). We generate 50,000 random samples of $\boldsymbol{\epsilon}$ and $\boldsymbol{\nu}$, and compute the noise distribution $p_{\text{noise}}(\boldsymbol{\epsilon} + \boldsymbol{\nu})$ using KDE with Gaussian kernels. Then we approximate the probability density $p(y_j \mid \xi_j^*, \boldsymbol{\theta})$ based on the sampling path for the linear model, which is given by: $\boldsymbol{y} = \theta_1 \mathbf{1} + \theta_2 \boldsymbol{\xi} + \boldsymbol{\epsilon} + \boldsymbol{\nu}$. The sampling path can be reformulated to $\boldsymbol{y} - (\theta_1 \mathbf{1} + \theta_2 \boldsymbol{\xi}) = \boldsymbol{\epsilon} + \boldsymbol{\nu}$. Thus we have $p(y_j \mid \xi_j, \boldsymbol{\theta}) = p_{\text{noise}}(y_j - (\theta_0 + \theta_1 \xi))$,which enables us to compute the reference MI using Eq. (2). To obtain an accurate approximation, we use $N_1 = 10,000$ and $N_2 = 1000$ samples for the nested Monte Carlo method.

### 1.2 Pharmacokinetic model

For the PK model, we aim to compute the reference MI value at $\boldsymbol{\xi}^* = [t_1^*, ..., t_D^*]^\top$. Since the noise source is Gaussian noise $\epsilon_{1t} \sim \mathcal{N}(0, 0.01)$ and $\epsilon_{2t} \sim \mathcal{N}(0, 0.1)$, we can obtain an explicit equation for the data-generating distribution $p(y_j \mid t_j, \boldsymbol{\theta})$,

$$p(y_j \mid t_j, \boldsymbol{\theta}) = \mathcal{N}(y_j; z(t_j, \boldsymbol{\theta}), z(t_j, \boldsymbol{\theta})^2 0.01^2 + 0.1^2) \tag{3}$$

where $\boldsymbol{\theta} = [V, k_a, k_e]^\top$ and the forward function $z(t_j, \boldsymbol{\theta})$ is

$$z(t_j, \boldsymbol{\theta}) = \frac{D_v}{V} \frac{k_a}{k_a - k_e} [e^{-k_e t_j} - e^{-k_a t_j}]. \tag{4}$$

We can therefore approximate the reference MI value in Eq. (1) using a sample average of $p(\boldsymbol{y} \mid \boldsymbol{\xi}^*)$ and data-generating distribution $p(y_j \mid t_j, \boldsymbol{\theta})$ in Eq. (3).

### 1.3 Tuning for quantum control

This example is an implicit model without pathwise gradients and no available sampling path is provided. It is infeasible to compute or approximate the reference MI value, even though the nest Monte Carlo works for the first two examples. The gradient-based BED method is not applicable for this case so we only focus on the comparison between our proposed `SAGABED` and two-stage method with Bayesian optimization.

## 2 Hyperparameter optimization

We conduct a hyperparameter optimization to search for an optimal neural network architecture and training parameters for maximizing the MI lower bound. We build a validation dataset by generating random samples from the prior distribution and data-generating distribution at the optimal design, then run five trails for every trained neural network model and estimate the mean and standard deviation. The optimal hyperparameters are obtained by means of grid search based on a set of candidate parameters.

### 2.1 Noisy linear model

For the low-dimensional cases $D = 1$ and $D = 10$, we consider to use one layer neural network model with different number of neurons, $\mathcal{H} = \{50, 100, 150, 200, 250, 300\}$. For the high-dimensional cases, $D = 50$ and $D = 100$, we use multiple layers from 1 to 6, and each layer has $\mathcal{H} = \{10, 30, 50, 100\}$. The learning rates is chosen properly from $\ell_\psi = \left\{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}\right\}$ and $\ell_\xi = \left\{10^{-3}, 10^{-2}, 10^{-1}\right\}$.

### 2.2 Pharmacokinetic model

In the PK model, we consider to use one layer neural network model with $\mathcal{H} = \{50, 100, 150, 200, 250, 300\}$ for low-dimensional case $D = 10$. For the high-dimensional cases, $D = 50$, $D = 100$ and $D = 500$. we use multiple layers from 1 to 6, and each layer has $\mathcal{H} = \{10, 30, 50, 100\}$. The candidate list of learning rates is $\ell_\psi = \left\{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}\right\}$ and $\ell_\xi = \left\{10^{-3}, 10^{-2}, 10^{-1}\right\}$.

### 2.3 Tuning for quantum control

We use to use one layer neural network model with $\mathcal{H} = \{50, 100, 150, 200, 250, 300\}$ for $D = 1$, $D = 5$ and $D = 10$. For the high-dimensional cases, $D = 50$, $D = 100$ and $D = 500$. we use multiple layers from 1 to 6, and each layer has $\mathcal{H} = \{10, 30, 50, 100\}$. The candidate list of learning rates is $\ell_\psi = \left\{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\right\}$ and $\ell_\xi = \left\{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\right\}$.

## 3 Additional details for the Guide ES algorithm

We use the implementation of the Guided ES (GES) algorithm published at https://github.com/brain-research/guided-evolutionary-strategies by the authors of the GES method. The hyperparameters of GES are listed as follows:

- $\alpha$: is a hyperparameter that trades off variance between the full parameter space and the subspace. If $\alpha = 1$, it is the vanilla ES estimator, but we choose $\alpha < 1$ that can result in significantly improved performance. We use $\alpha = 0.5$ for all examples.
- $\beta$: is a scale parameter that is an overall scale factor to estimate the gradient. We use $\beta = 2.0$ for all examples.
- $\sigma$: is the variance of the perturbations or called a smooth radius. We use $\sigma = 0.1$ for the linear model and PK examples, and $\sigma = 0.5$ for quantum control example.
- $P$: us the number of sample pairs or called population size. It depends on the dimensionality of the problems. So we use $P = 100$ for low-dimensional cases, such as $D = 1, 10$, and we use $P = 500$ instead for high-dimensional cases, for example, $D = 50, 100, 500$.