UW Biostatistics Working Paper Series

5-8-2015

Stochastic Optimization via Forward Slice

Bob A. Salim

University of Washington - Seattle Campus, bobs3@uw.edu

Lurdes Y. T. Inoue

University of Washington, linoue@u.washington.edu

Stochastic Optimization via Forward Slice

May 8, 2015

1. INTRODUCTION

Optimization consists of maximizing or minimizing a real-valued objective function. In chemical engineering, for example, optimization may be used to find a set of conditions under which the maximum yield may be obtained from a chemical process. The field of industrial engineering also deals with optimization of complex processes, which include people, information, materials, etc. In statistics, optimization has been used, for example, in estimating parameters of the linear regression model, where the goal is to minimize the square loss function. Another area in statistics for which optimization methods are highly useful is in the area of optimal design where the experimental samples are chosen to maximize the information that is attained from the experiment.

In many optimization problems, such as in estimating the linear regression parameters, the objective function is simple enough such that closed-form solutions may be obtained. However, in many problems encountered in practice, the objective function to be maximized may not yield closed-form solutions. Over many decades, optimization methods have been developed to solve these problems. The optimization methods can be characterized into two categories: deterministic and stochastic optimization methods.

Deterministic optimization is a branch in mathematics which encompasses classical methods for optimization. Deterministic optimization methods, in general, rely on the gradient or the Hessian of the objective function. Some common methods for deterministic optimization include the conjugate gradient (CG) method [6], BFGS method [1], and Nelder-Mead method [10].

Deterministic optimization methods are usually sensitive to the initial values and may not converge depending on the initial value and the shape of the objective function. Furthermore, the solution from the deterministic optimization methods may converge to a local extremum but not a global extremum.

Stochastic optimization methods attempt to address some of the above issues. Stochastic optimization methods refer to methods that utilize iterates that are 'random'. An example of stochastic optimization methods is simulated annealing, which is a local search algorithm used for optimization problems that is capable of escaping from a local optimum by allowing a chance to select inferior points at a given iteration [7]. The probability of selecting an inferior point depends on a "temperature" parameter at each iteration. However, some of the issues with the simulated annealing method are that it needs to be heavily calibrated, it is sensitive to initial value, and it does not guarantee convergence to the global maximum.

In this paper, we propose another stochastic optimization method, which we call "Forward Slice", which we describe in Section 2. In Section 3, the performance of the proposed method is evaluated in both optimization and in its application in optimal design. In Section 4, we discuss the performance of the method, as well as its advantages and some limitations.

2. METHOD: FORWARD SLICE

In this section we propose a method for stochastic optimization. We evaluate its performance and apply to design problems in Section 3. At its core, our method is based on the procedure that Neal (2003) called the 'slice sampling' procedure [9] [3] [8], which was originally developed as a Markov chain Monte Carlo sampling procedure to draw samples from a target distribution. The slice sampling method relies on an auxiliary variable which defines a level at which we slice the target density to obtain regions from which we draw samples of the target distribution.

Similar to Neal's method, our procedure uses an auxiliary variable for stochastic optimization that also defines the slices, but of an objective function to be maximized (or minimized). Moreover, unlike with Neal's method, the auxiliary variable in our approach is not sampled and takes on non-decreasing values in the sequential iterations of the procedure so that, for a given pre–specified tolerance, at the end of the procedure we attain the maxima and the argument of the maxima (or close values given the selected tolerance level).

2.1 Forward Slice: Unidimensional case

Let $f : \mathcal{X} \to \mathbb{R}$ be a function with a compact domain $\mathcal{X}$ and codomain $\mathbb{R}$. We note that we alternatively may refer to $\mathcal{X}$ as the design space. Our goal is to find values $x \in \mathcal{X}$ that maximize $f(\cdot)$. We build on the slice sampling idea to derive a procedure for stochastic optimization.

In our method we obtain slices with increasing function level y at each iteration. We call it the *forward slice* procedure. For a given $x_0 \in \mathcal{X}$, we take $y = f(x_0)$. Let S denote the horizontal slice as defined by Neal, which is the region defined as $S = \{x : f(x) \geq y\}$. The forward slice procedure selects the next point from the slice region S, more specifically, the slice region within the design space, which we denote as S^X (i.e. $S^X = S \cap \mathcal{X}$). The forward slice procedure guarantees that the next sample generated from the forward slice (x_1) follows $f(x_1) \geq f(x_0)$ because x_1 is sampled from S, a region where the function cannot take on decreasing values.

Note that the set S may not be completely known. Regardless, the sampling of x_1 from the set S can be done using a simple rejection sampling scheme. Specifically, first, we sample an initial proposal candidate (call it p_0) by uniformly sampling from the entire design space $\mathcal{X}$. If $f(p_0) \geq f(x_0)$, then we take $x_1 = p_0$. Otherwise, we reduce the space from which we sample a proposal based on p_0. Specifically, since $x_0 \in S$ (because $f(x_0) \geq f(x_0)$), then we can obtain the revised sampling space as $\mathcal{X}^{(1)} = \mathcal{X}^{(0)} \cap (-\infty, p_0]$ if $p_0 > x_0$ or $\mathcal{X}^{(1)} = \mathcal{X}^{(0)} \cap [p_0, \infty)$ if $p_0 < x_0$. Then, we sample a new proposal p_1 from $\mathcal{X}^{(1)}$. Similarly, if $f(p_1) \geq f(x_0)$, then we take $x_1 = p_1$. Otherwise, we reduce the sampling space again based on p_1 as $\mathcal{X}^{(2)} = \mathcal{X}^{(1)} \cap (-\infty, p_1]$ if $p_1 > x_0$ or $\mathcal{X}^{(2)} = \mathcal{X}^{(1)} \cap [p_1, \infty)$ if $p_1 < x_0$. We continue to sample proposal candidates and to reduce the sampling space until we sample p_j such that $f(p_j) \geq f(x_0)$, for which we take $x_1 = p_j$.

There is no guarantee that the next selected point x_1 maximizes the function $f(x)$. Thus, we define the forward slice as an iterative procedure, using the obtained x_1 from the previous iteration as the next input in the next iteration. We use the following notation: x_0 denotes the initial point input to the forward slice, while x_k and S_k denote, respectively, the point and the slice region after the k^{th} iteration where $S_k = \{x : f(x) \geq f(x_k)\}$, and S_k^X denotes the portion of the slice region S_k within the design space $\mathcal{X}$, i.e. $S_k^X = S_k \cap \mathcal{X}$. In other words, S_k^X is a set of points in $\mathcal{X}$ that is inside the slice region after the k^{th} iteration, i.e. $S_k^X = \{x : x \in \mathcal{X}, f(x) \geq f(x_k)\}$. Furthermore, we use the following notation for the proposal candidates for one slice iteration: p_0 denotes the initial proposal candidate, $\mathcal{X}^{(1)}$ denotes the space from which the next proposal (p_1, if needed) is drawn; and more generally p_j denotes the proposal candidate after j previously-rejected proposals, and $\mathcal{X}^{(j+1)}$ denotes the space from which the next proposal candidate (p_{j+1}) is to be drawn if p_j is also rejected. By convention, $\mathcal{X}^{(0)} = \mathcal{X}$. Given a user-specified tolerance for convergence ϵ, the forward slice procedure is given as follows in Algorithm 1.

Algorithm 1 Univariate forward slice

```
1: Set $\epsilon>0$ as convergence tolerance, initial point $x_0$, and design space $\mathcal{X}$.
2: Set $k = 0$
3: repeat
4:     Define $S_k = \{x : f(x) \geq f(x_k)\}$
5:     Set $j = 0$. Sample $x_{k+1}$ uniformly from $S_k^X = S_k \cap \mathcal{X}$ as follows:
6:     repeat
7:         Sample $p_j$ uniformly from $\mathcal{X}^{(j)}$
8:         if $f(p_j) \geq f(x_k)$ then
9:             $x_{k+1} \leftarrow p_j$
10:            break
11:        else
12:            if $p_j>x_k$ then
13:                $\mathcal{X}^{(j+1)} \leftarrow \mathcal{X}^{(j)} \cap (-\infty, p_j]$
14:            else
15:                $\mathcal{X}^{(j+1)} \leftarrow \mathcal{X}^{(j)} \cap [p_j, \infty)$
16:            end if
17:            $j \leftarrow j + 1$
18:        end if
19:    until $f(p_j) \geq f(x_k)$
20:    Calculate $D = |(f(x_{k+1}) - f(x_k))/f(x_k)|$
21:    $k \leftarrow k + 1$
22: until $D \leq \epsilon$
```

2.2 Forward Slice: Multidimensional case

We can extend the forward slice method to the multi–dimensional case in analogy to the adaptive slice sampling by Thompson and Neal [11] that utilizes sampled "crumbs" from a known auxiliary distribution. Neal [9] proposed these "crumbs" as a trail of points in the space $\mathcal{X}$ that guide the drawing of the next sample given the initial point. Assume the domain space $\mathcal{X}$ has dimension p. Let J denote the matrix whose nullspace represents the subspace of $\mathcal{X}$ from which the next crumb is to be drawn. This space is informed by the rejected samples obtained from the crumbs. Similarly to the unidimensional case, for a given $\mathbf{x}_0 \in \mathcal{X}$, we take $y = f(\mathbf{x}_0)$. Let S denote the horizontal slice as defined by Neal, which is the region defined as $S = \{\mathbf{x} : f(\mathbf{x}) \geq y\}$. The sampling from the slice S is done also by rejection sampling similarly to the unidimensional case. We start by obtaining a proposal sample $\mathbf{x}_p$ from the design space $\mathcal{X}$. If $f(\mathbf{x}_p)>f(\mathbf{x}_0)$, then we take $\mathbf{x}_1 = \mathbf{x}_p$. Otherwise, the proposal $\mathbf{x}_p$ is rejected, and the subspace from which the next proposal sample is taken from is guided by $\mathbf{x}_p$. For a rejected proposal $\mathbf{x}_p$, let g^* be the projection of $\nabla f(\mathbf{x}_p)$ onto the nullspace of J, where ∇f denotes the gradient of the function f. If g^* has a "large angle" with the gradient of the function at the rejected proposal, then the nullspace of

J and the gradient is "nearly" orthogonal. On the other hand, when the angle is small, then we adapt the sampling subspace by appending the column of J by the normalized g^*. Thompson and Neal defined θ to be the threshold angle for "non-orthogonality", and they proposed 60^o as the threshold [11]. Then, for a rejected proposal $\mathbf{x}_p$, a new proposal sample $\mathbf{x}_p$ is obtained by sampling from $\mathcal{N}(J) \cap \mathcal{X}$, where $\mathcal{N}(J)$ denotes the nullspace of J. This can be done by sampling $\mathbf{x}_p^0$ from $\mathcal{X}$ and taking its projection onto the nullspace of J, i.e. $\mathbf{x}_p = (I - JJ^T)\mathbf{x}_p^0$. This continues until a proposal $\mathbf{x}_p$ such that $f(\mathbf{x}_p) > f(\mathbf{x}_0)$ is obtained

Thus, adapting the shrinking rank approach by Thompson and Neal [11] for stochastic optimization via the iterative forward slice procedure we attain the following Algorithm 2.

Algorithm 2 Multivariate forward slice

```
1:  Set the the tolerance $\epsilon$, initial design point $x_0$, the maximum number of
    candidates per slice $n_c$, the shrinking multiplier $\phi \in (0,1)$, and the angle $\theta$.
2:  **repeat**
3:      Obtain $y \leftarrow f(x_0)$
4:      Define $S = \{x : f(x) \geq y\}$
5:      Initialize $J = [\ ]$ as an empty-matrix.
6:      Initialize $\mathcal{S} = \mathcal{X}$.
7:      Sample $x_1$ from $S^X = S \cap \mathcal{X}$ as follows
8:      **repeat**
9:          Obtain a random sample $z_k$ from $\mathcal{S}$
10:         **if** $J = [\ ]$ **then**
11:             $\tilde{x}_k \leftarrow x_0 + z_k$
12:         **else**
13:             $\tilde{x}_k \leftarrow x_0 + z_k - JJ^T z_k$
14:         **end if**
15:         **if** $f(\tilde{x}_k) \geq y$ **then**
16:             $x_1 \leftarrow \tilde{x}_k$
17:             **break**
18:         **else**
19:             **if** the number of columns of $J < p - 1$ **then**
20:                 Calculate $g^*$ as follows
21:                 **if** $J = [\ ]$ **then**
22:                     $g^* = \nabla f(\tilde{x}_k)$
23:                 **else**
24:                     $g^* = \nabla f(\tilde{x}_k) - JJ^T \nabla f(\tilde{x}_k)$
25:                 **end if**
26:                 **if** $\frac{g^{*T} \nabla f(\tilde{x}_k)}{\|g^{*T}\| \|\nabla f(\tilde{x}_k)\|} > \cos(\theta)$ **then**
27:                     Update $J = [J \quad g^*/\|g^*\|]$
28:                 **else**
29:                     Update $\mathcal{S} \leftarrow \mathcal{S} \times \phi$
30:                 **end if**
31:             **else**
32:                 Obtain $n_c$ candidate points from the sampling subspace.
33:                 Call this candidate set $\mathcal{C}$.
34:                 **if** $\exists x_c \in \mathcal{C}$ such that $f(x_c) \geq y$ **then**
35:                     $x_1 \leftarrow \underset{c \in \mathcal{C}}{\text{argmax}}[f(c)]$
36:                     **break**
37:                 **else**
38:                     Reset $J = [\ ]$
39:                 **end if**
40:             **end if**
41:         **end if**
42:     **until** $f(x_1) \geq y$
43:     Calculate $D = |(f(x_1) - f(x_0))/f(x_0)|$
44:     $x_0 \leftarrow x_1$
45: **until** $D \leq \epsilon$
```

2.3 Heuristics for Stochastic Optimization via Forward Slice

The forward slicing method should select a point that gives a global maximum of a function $f(\cdot)$ under the following conditions:

(1). The function f is continuous, smooth, and differentiable in the space $\mathcal{X}$.

(2). There is no *local plateau* in $\mathcal{X}$. We define a local plateau as follows: a local plateau exists in $\mathcal{X}$ if $\exists$ an interval $P \subset \mathcal{X}$ such that $f(x_i) = f(x_j)$ $\forall x_i, x_j \in P$ and $\exists x_k \in \mathcal{X}$ such that $f(x_k) > f(x)$ $\forall x \in P$.

(3). The design space $\mathcal{X}$ is compact.

Under the above conditions, in the Appendix we show that the iterates converge to a global maximum. For practical implementation of the algorithm, however, we set a tolerance $\epsilon > 0$ to restrict the number of iterations.

A graphical illustration of the forward slice procedure is shown in Figure 1. Consider the case where we have the initial point x_0 closer to one of the local modes that is not the global maximum. The first iteration of the forward slice procedure is shown below.

The upper left panel of Figure 1 shows that the next sampled point based on the first iteration is at the second mode or very close to it. Since the difference in the response between $f(x_1)$ and $f(x_0)$ is larger than the set tolerance, we perform another iteration of the forward slice, as shown in the upper right panel of Figure 1. Given the new slice level, the slice S contains two intervals: one interval for the first mode and another for the second mode. Since the first interval is much larger than the second one, we are more likely to sample the next x_1 from the slice portion containing the first mode, which is what happened in this case.

Given that we have not reached the tolerance for convergence, we perform another forward slice step, as shown in the lower left panel of the Figure 1. We observe that the slice level now excludes the second mode entirely, ensuring that we will obtain the maximum that is in the first mode if we continue running the forward slice until convergence. In this case, convergence is achieved after 17 iterations which is shown in the lower right panel of Figure 1, and we observe that the x obtained gives the global maximum for $f(x)$.

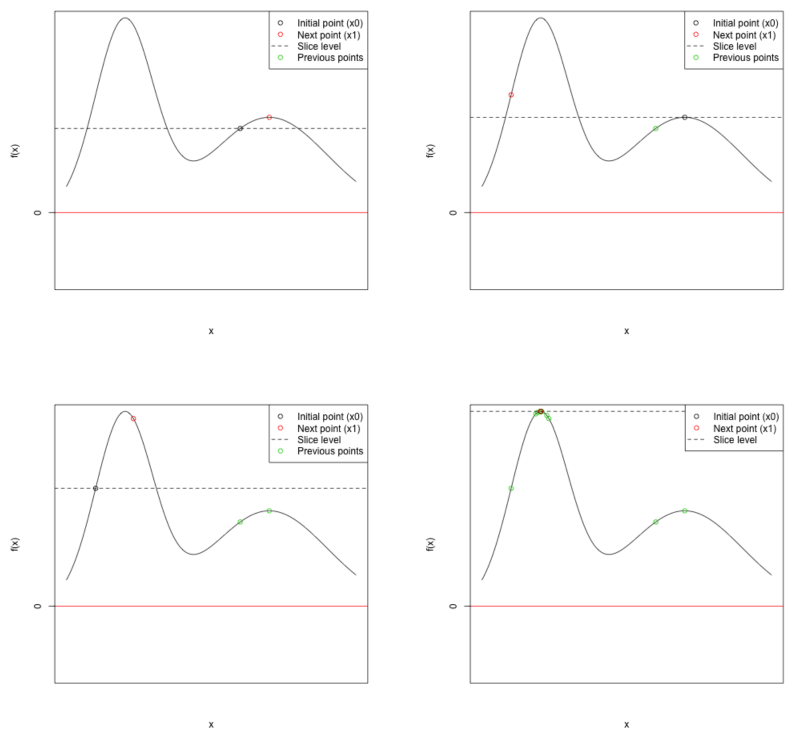


Figure 1: An example run of the forward slice with tolerance ϵ = 1E-8.

3. SIMULATION STUDIES

3.1 Forward Slice in Optimization

We utilized simulation studies to assess the performance of the forward slice method compared to alternative optimization methods, namely, the conjugate gradient (CG) method, the Nelder-Mead method (NM), the Broyden-Fletcher-Goldfarb-Shanno (BFGS) method, the bounded BFGS (L-BFGS-B) method, simulated annealing (SANN), and the Brent method, all of which available with the optim() function in R.

For the unidimensional case, we considered a multimodal response function as a mixture of three normal distributions and assessed the methods over 1000 replications by comparing their resulting points x that maximize the function $f(x)$. All methods utilized a common initial value.

Figure 2 shows that the NM, BFGS, L-BFGS-B, CG, and SANN methods correctly select the highest mode around 50% of the time. On the other hand, the forward slice method correctly selects the optimum at the highest mode 98% of the time.

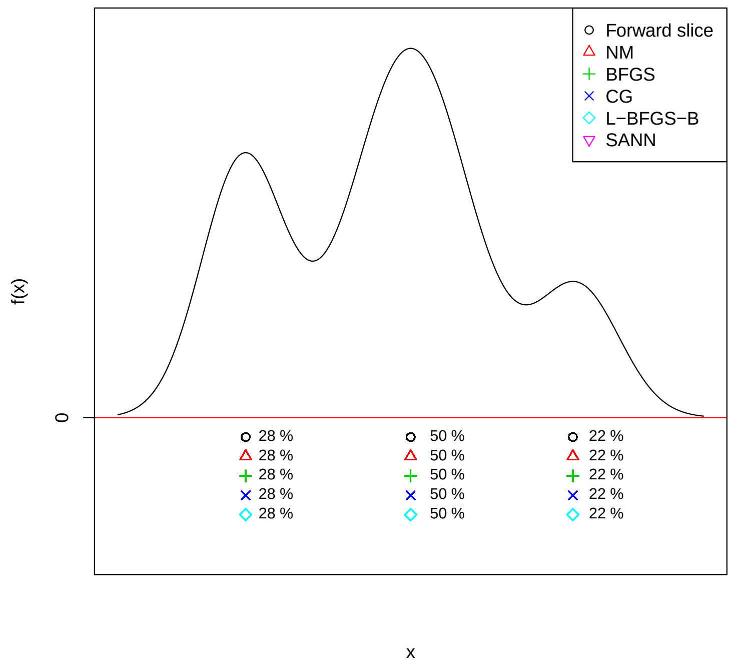


Figure 2: Comparison of methods in a univariate setting. The points represent where the optimum was selected for one replication. Reported percentages show how often the point was selected as the optimum for each method upon 1,000 replications

Similarly, we considered a scenario of the multivariate response function as a mixture of three bivariate normal distributions and assessed the methods over 1000 replications by comparing their resulting points x that maximize the function $f(x)$.

Figure 3 and Table 1 show that the multivariate forward slice procedure chooses the correct mode more frequently compared to the alternative methods. The average maxima is also higher and closer to the true maxima under the multivariate forward slice compared to the average from the alternative methods.

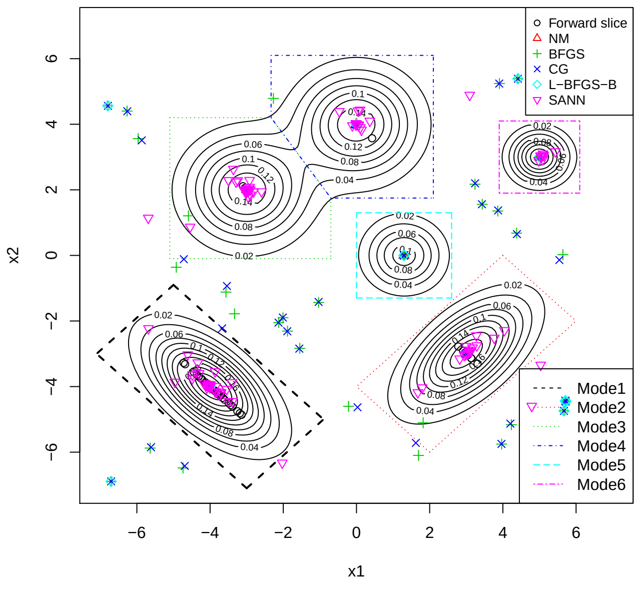


Figure 3: Comparison of methods in multivariate setting. The hollow round points represent where the optimum was selected for one replication.

Table 1: Comparison of methods in multivariate setting as shown also in Figure 3. The percentages are the frequency for each method selecting a point of that given mode. This table presents results for the highest two modes of the function, with the first mode higher than the second mode.

Method	Mode 1 (%)	Mode 2 (%)	Mean response
Forward slice	61	22	0.208
NM	17	24	0.179
BFGS	12	15	0.123
CG	12	15	0.128
L-BFGS-B	18	21	0.170
SANN	35	16	0.169
True Maximum			0.239

3.2 Forward Slice in Design Problems

In this section we consider the multi–dimensional design problem where the goal is to select design points to maximize the information obtained from an experiment, given that there is a constraint in the maximum sample size. Many optimality criteria, describing some measure of information from an experiment, have been developed over the years. The most commonly used criterion is the D-optimality criterion, which seeks to minimize the volume of the confidence ellipsoid of the parameters in a specific model.

Dror and Steinberg (2008) proposed a sequential method for obtaining a D-optimal multi-variable experiment under the generalized linear models (GLM) framework. Their method relies on approximations to the Bayesian D-optimality criterion to improve computational efficiency and time. The Bayesian D-optimality criterion is given by Chaloner and Larntz [2]:

$$\phi(d) = \int \log(|\mathbf{I}(\boldsymbol{\beta}; d)|) d\pi(\boldsymbol{\beta}).$$

Dror and Steinberg proposed two approximations to the Bayesian D-optimality criterion. First, their method is based on the discretization of the prior. Second, they use an approximation of the Bayesian D-optimality criterion (defined above) to obtain a candidate set of points for appending to the current design matrix with:

$$\phi_2(\tilde{\boldsymbol{\beta}}; d) \quad = \quad \log(|\mathbf{I}(\tilde{\boldsymbol{\beta}}; d)|)$$

where $\tilde{\boldsymbol{\beta}}$ denotes the posterior median of the parameter $\boldsymbol{\beta}$ for the GLM model. At each sequential step, the design point that maximizes the optimality criterion is obtained using Federov's exchange algorithm [4] [5]. At any given sequential step, a candidate set of points $\mathcal{C}$ is obtained from the design space $\mathcal{X}$, and the current design matrix is appended by a random subset of the candidate set. The Federov's exchange algorithm is interative. At any iteration of the exchange algorithm, a point from the design matrix is exchanged with a point from the candidate set as to maximize the increase in the determinant of the information matrix [5].

Alternatively, we propose to select the next design point using the forward slice procedure. We note that the approximate D-optimality criterion specifically define the objective function for the forward slice. Specifically, we utilize the function

$$f(x) \quad = \quad \phi_2(\tilde{\boldsymbol{\beta}}; d) \; = \; \phi_2\left(\tilde{\boldsymbol{\beta}}; \begin{pmatrix} X \\ x \end{pmatrix}\right) \; = \; \log(|\mathbf{I}(\tilde{\boldsymbol{\beta}}; d)|).$$

Thus, in the above, $d = \begin{pmatrix} X \\ x \end{pmatrix}$, where X is the current design matrix and x is the next (multivariate) design point.

For completeness, we re-state the forward slice procedure for the D-optimality criterion using the above function. We note that the initial design points are obtained using the algorithm by Dror and Steinberg to guarantee that the information matrix is of full rank, and to ensure comparability between methods.

Let n denote the current number of samples, and N denote the total sample size. Then, the forward slice procedure for selecting a local D-optimal design point is given by

Algorithm 3 Univariate forward slice

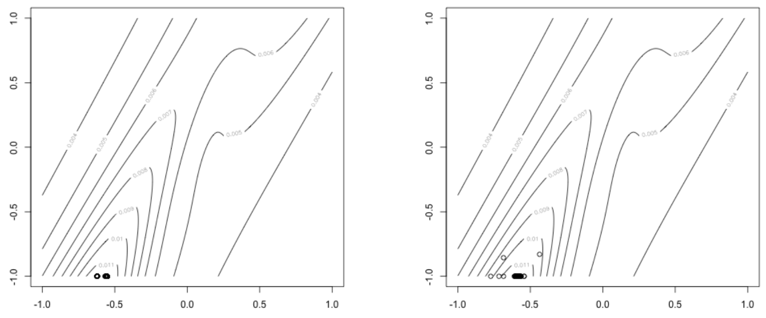
```
1: Define the tolerance $\epsilon$.
2: Obtain the first $n_0$ design points defining a full–rank current design matrix
   $X_0$ using the method by Dror and Steinberg using the Fedorov's exchange
   algorithm. (Note that $n = n_0$ after this step)
3: for $n<N$ do
4:     Define $x_{0n}$ to be the $n^{th}$ row of $X_0$
5:     Define $d_n = \begin{pmatrix} X_0 \\ x_{0n} \end{pmatrix}$
6:     Define $f(x) = \phi_2 \left( \tilde{\boldsymbol{\beta}}; \begin{pmatrix} X_0 \\ x \end{pmatrix} \right)$.
7:     $y \leftarrow f(x_{0n}) = \phi_2(\tilde{\boldsymbol{\beta}}; d_n)$
8:     Obtain $x_1$ using Algorithm 2 with $x_{0n}$ as initial point and $f(x)$ defined
   in step 6
9:     Append $X_0 \leftarrow \begin{pmatrix} X_0 \\ x_1 \end{pmatrix}$
10: end for
```

Simulation with 2 variables. We performed a simulation study with 1000 replications to assess the performance of the slice design procedure compared to the Fedorov's exchange algorithm for selecting a single next design point. We used a logistic regression model with 2 variables with true regression parameters $(0, 7, -3)^T$. The initial design matrix is given by $\mathbf{X} =$

$$\begin{pmatrix} 1 & -0.262 & -1 \\ 1 & 0.259 & 1 \\ 1 & 0.754 & 1 \\ 1 & 0.108 & -1 \end{pmatrix}$$

This initial design matrix $\mathbf{X}$ is based on running the algorithm by Dror and Steinberg up until the information matrix given the current design points is non-singular. Given that there are only 2 variables, we can obtain the contour plot of the D-optimality criterion upon addition of one design point (x_1, x_2) as a function of x_1 and x_2. With 1000 replications, the scatter of the next design point given the current design $\mathbf{X}$ for both methods is given below.

Figure 4(a) shows that the obtained design point is close to the optimal using the Fedorov's exchange algorithm, whereas in Figure 4(b), we observe that the obtained design point using the slice procedure also clusters around the optimal. The mean log D-optimality for Fedorov's is 0.01189, whereas the mean log D-optimality for the slice procedure is 0.01188. The two methods appear to be comparable in selecting the next design point in terms of the D-optimality criterion.



(a) (b)

Figure 4: Contour plot of the D-optimality criterion upon addition of one design point (x_1, x_2). Scatter of next optimal design points using the Federov's exchange algorithm (panel a) or the forward slice procedure (panel b).

Simulation studies are then performed to assess the performance of the forward slice method throughout a study instead of selecting a single next design point to sample. We assess the performance of the forward slice method in Dror and Steinberg's sequential D-optimal design with a binary response variable. Dror and Steinberg's sequential D-optimal design re-estimates the model parameters after the response under a new design point is observed. Based on the current parameter estimates, the next design point is chosen to maximize the estimate of the D-optimality criterion using the Fedorov Exchange Algorithm. This procedure is repeated until the maximum sample size. In our simulation studies, we only replace the Fedorov Exchange Algorithm with the forward slice procedure, but do not change the parameter estimation procedure or stopping criterion.

We performed a simulation study with 100 replications to assess the performance of the slice design procedure compared to the Fedorov's exchange algorithm using a logistic regression model with 4 variables with the maximum sample size of n=30.

Table 2 shows the true parameter values and the prior distribution for each parameter along with corresponding simulation results. We note that in this set of simulations the prior distributions are flat, but centered at the true parameter values.

Table 2: Simulation results comparing the performance of the forward slice to the Fedorov's exchange algorithm in the context of the method by Dror and Steinberg when the priors are flat, but centered around the true parameter value. We assume n=30.

			Mean parameter (Mean 95% range)	
Parameter	True value	Prior dist.	Fedorov	Forward-slice
β_0	0	Unif(-10,10)	-0.233 (2.246)	-0.049 (1.98)
β_1	7	Unif(-3,17)	11.84 (6.64)	12.20 (7.06)
β_2	8	Unif(-2,18)	13.56 (7.46)	13.30 (7.91)
β_3	-3	Unif(-13,7)	-5.33 (3.80)	-4.75 (2.87)
β_4	0.5	Unif(-9.5,10.5)	0.79 (2.61)	0.79 (2.19)
Median D-Efficiency			0.52	0.56

From Table 2 above, we see that the median D-efficiency is slightly higher using the forward-slice procedure compared to using the Fedorov Exchange Algorithm. The bias and uncertainty of the parameter estimates are comparable between the two methods.

We further explored the methods to assess sensitivity to the the assumed priors with a simulation with 100 replications, but this time assuming that the priors are flat and centered around 0 instead of centered around the true parameter values, and with maximum sample size of $n = 40$ per replication.

Table 3: Simulation results comparing the performance of the forward slice to the Fedorov's exchange algorithm in the context of the method by Dror and Steinberg when the priors are flat, but centered around 0. We assume $n = 40$.

			Mean parameter (Mean 95% range)	
Parameter	True value	Prior dist.	Fedorov	Forward-slice
β_0	0	Unif(-10,10)	0.124 (1.528)	0.404 (1.25)
β_1	7	Unif(-10,10)	6.77 (4.05)	7.88 (3.60)
β_2	8	Unif(-10,10)	7.92 (3.29)	8.28 (3.04)
β_3	-3	Unif(-10,10)	-3.37 (2.09)	-3.03 (1.70)
β_4	0.5	Unif(-10,10)	-0.23 (2.23)	-0.39 (2.46)
Median D-Efficiency			0.46	0.50

From Table 3 above, we see that the median D-efficiency is slightly higher using the forward-slice procedure compared to using Fedorov Exchange Algorithm. The bias and uncertainty of the parameter estimates are comparable between the two methods.

4. DISCUSSION

The forward slice procedure is a stochastic optimization procedure that selects points with response closer to the optimum at each iteration. The points obtained from the forward slice procedure theoretically converge to the global maximum within a constraint of the design space $\mathcal{X}$. The forward slice procedure allows escape from a local optimum by not restricting the search for the next point to the local search. Rather, the forward slice procedure samples from $S^X = S \cap \mathcal{X}$, that is, the slice within the design space.

The forward slice procedure selects the highest mode (or maxima) most of the times in univariate optimization, as shown in Figures 2, and 3 and outperforms the selection obtained with alternative optimization methods (conjugate gradient, BFGS, L-BFGS-B, Nelder-Mead, and simulated annealing). Similarly, the forward slice procedure outperforms the aforementioned alternative optimization methods in the multivariate setting.

We assessed the method in a design problem approached by Dror and Steinberg and utilized the forward slice procedure instead of the Federov Exchange Algorithm to obtain the design point that would maximize the D-optimality criterion when appended to the current design matrix. In this case, the parameters defining the true objective function are unknown. Thus, the objective function changes upon accrual of new information. As a consequence, the optimum design points are selected based on the current estimate of the objective function (by assuming a given model and based on the current estimates of the model parameters). For the multifactor experiments, we found that the forward slice procedure obtains design points with slightly higher D-efficiency compared to the design points obtained with the Fedorov exchange algorithm.

One limitation for the forward slice procedure is that the probability of sampling from the highest mode compared to a local lower mode depends on the relative size of the 'support' of the modes at each slice level. For example, for a function with narrow high mode and wider lower mode, as shown in Figure 5 below, we observe that the forward slice procedure only selects the highest mode around 57% of the times. We observe, however, that the forward slice procedure still outperforms the alternative methods which only selected the highest mode around 20% of the times.

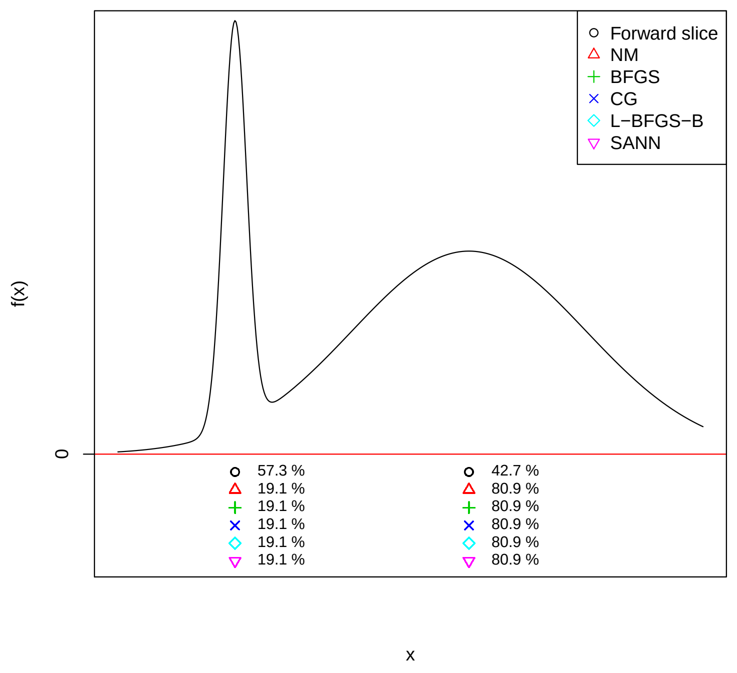


Figure 5: Comparison of methods. The points represent where the optimum was selected for one replication. Reported percentages show how often the point was selected as the optimum for each method upon 1,000 replications

We note that the forward slice procedure may have an increased time to convergence. In our simulations, the forward slice procedure took on average, around 5 times the computing time required for the simulated annealing method. To explain this increase in computation time, take for an example the one–dimensional optimization. The alternative methods may stop after finding a local mode. The forward slice procedure, however, may continue sampling if there is support for other modes in a given slice. In multi–dimensional setting the increase in computational time can further be explained by the adaptive procedure for sampling from the slice. Thus, we observe a performance–computing time trade–off with respect to our proposed method versus existing alternatives. However, the ability of the method to continue to explore other modes allows for its finding of the global maximum.

Appendix

Under the conditions stated in Section 2.3, in this section we show that the iterates of the forward slice procedure converge to a global maximum.

Lemma 2.1. For any given point $x_0 \in \mathcal{X}$, for which x_0 is not a global maximum, one iteration of the forward slice procedure gives a point $x_1 \in \mathcal{X}$ such that $f(x_1) > f(x_0)$ a.s.

Proof. Given the initial point x_0, we obtain the slice level $y = f(x_0)$ which defines the horizontal 'slice' $S_0 = \{x : f(x) \geq f(x_0)\}$. Based on our slice procedure, the next point x_1 is sampled from $S_0^X = S_0 \cap \mathcal{X} \subseteq S_0$. Note that $S_0^X \neq \emptyset$, since $x_0 \in S_0^X$. Since $\forall x \in S_0$, $f(x) \geq f(x_0)$ by definition of S_0, we have $f(x_1) \geq f(x_0)$. Let $A_0^X = \{x : f(x) = f(x_0)\} \cap \mathcal{X}$ and $B_0^X = \{x : f(x) > f(x_0)\} \cap \mathcal{X}$. Thus, clearly $S_0^X = A_0^X \cup B_0^X$, and $A_0^X \cap B_0^X = \emptyset$. Since x_1 is sampled from S_0^X, so either $x_1 \in A_0^X$ or $x_1 \in B_0^X$. However, given the continuity and absence of local plateau, A_0^X is a set of finitely many points, i.e. $l(A_0^X) = 0$, where l denotes the Lebesgue measure. Since x_1 is sampled uniformly from S_0^X, then x_1 is from A_0^X with probability $l(A_0^X)/l(S_0^X)$ or x_1 is from B_0^X with probability $l(B_0^X)/l(S_0^X)$. Since $l(A_0^X) = 0$, then $x_1 \in B_0^X$ a.s., which means $f(x_1) > f(x_0)$ a.s.

□

Recall the notation that for the k^{th} iteration of the forward slice procedure, x_k denotes the current iterate, x_{k+1} denotes the next iterate, and S_k denotes the 'slice' at the current iterate, i.e. $S_k = \{x : f(x_k) \geq f(x)\}$. Let $X^* = \{x_1^*, x_2^*, ..., x_N^*\}$ denote the points that give the global maxima, i.e. $f(x_i^*) \geq f(x)$, $\forall x \in \mathcal{X}$, $\forall i = 1, 2, ..., N$. Given condition (2) of no local plateau in the design space $\mathcal{X}$, X^* contains finitely many points.

Lemma 2.2. For a given point $x_0 \in \mathcal{X}$, for which x_0 is a global maximum, i.e. $x_0 \in X^*$, an iteration of the forward slice procedure gives a point $x_1 \in X^*$ a.s.

Proof. Given the initial point $x_0 \in X^*$, the slice region is given by $S^X = X^*$ since x_0 is already a global maximum. Hence, the next design point x_1 obtained from an iteration of the slice procedure would result in uniform sampling of the points from X^* and hence $x_1 \in X^*$.

□

For an arbitrary k^{th} iteration, define $S_k^X = S_k \cap \mathcal{X}$. Hence, $S_k^X \subseteq S_{k-1}^X$ because $f(x_k) \geq f(x_{k-1})$. So, given the initial input x_0, $S_0^X \supseteq S_1^X \supseteq \ldots \supseteq S_k^X \supseteq \ldots$. Then, we have that $\cap_{n=1}^{k} S_n^X = S_k^X$. Furthermore, S_k^X is compact for any k^{th} iteration. By Cantor's interesection theorem, $\cap_{n=1}^{\infty} S_n^X \neq \emptyset$. First, we prove that the limit of the slice regions exist, and is non-empty.

Lemma 2.3. The limit of S_k^X exists and is non-empty

Proof.

$$\liminf_{k\to\infty} S_k^X \quad = \quad \bigcup_{k\in\mathbb{N}} \bigcap_{i\geq k} S_k^X \quad = \quad \bigcup_{k\in\mathbb{N}} (S_k^X \cap S_{k+1}^X \cap \ldots)$$

Since $S_k^X \subseteq S_{k-1}^X$ for any k, then $(S_k^X \cap S_{k+1}^X \cap \ldots) \supseteq (S_{k+1}^X \cap S_{k+2}^X \cap \ldots) \supseteq \ldots$. Thus, we have

$$\liminf_{k\to\infty} S_k^X \quad = \quad \bigcup_{k\in\mathbb{N}} (S_k^X \cap S_{k+1}^X \cap \ldots) \quad = \quad (S_1^X \cap S_2^X \cap \ldots) \quad = \quad \bigcap_{k\in\mathbb{N}} S_k^X$$

and

$$\limsup_{k\to\infty} S_k^X \quad = \quad \bigcap_{k\in\mathbb{N}} \bigcup_{i\geq k} S_k^X \quad = \quad \bigcap_{k\in\mathbb{N}} S_k^X.$$

Hence, $\lim_{k\to\infty} S_k^X = \limsup_{k\to\infty} S_k^X = \liminf_{k\to\infty} S_k^X = \bigcap_{k\in\mathbb{N}} S_k^X$. By Cantor's intersection theorem, $\bigcap_{k\in\mathbb{N}} S_k^X \neq \emptyset$.

□

Moreover, $\forall x \in X^*, x \in S_k^X$ for any $k = 1, 2, ...$ and the smallest possible slice within the design space is $S^X = X^*$. We then prove that the limit of the slice regions S_k^X is X^*.

Theorem 2.4. $\lim_{k\to\infty} S_k^X = X^*$.

Proof. First, note that $\forall x^* \in X^*$, $x^* \in S_k^X$ for any $k \in \mathbb{N}$, since x^* is a global maximum. Hence, $X^* \subseteq \bigcap_{k\in\mathbb{N}} S_k^X$. What is left to be proven is that the limit of S_k^X is not a set strictly greater than X^*.

Suppose that $\tilde{S}$ is any set strictly greater than X^*, that is, $\tilde{S} \cap (X^*)^C \neq \emptyset$. Denote $S_C = \tilde{S} \cap (X^*)^C$. For any $k, S_k^X \subseteq S_{k-1}^X$, to have $\lim_{k\to\infty} S_k^X = \tilde{S}$, we need that $x_c \in S_k^X, \forall x_c \in S_c\ \forall k \in \mathbb{N}$. By definition of S_C, $f(x_c) < f(x^*)$ for any $x_c \in S_C, x^* \in X^*$.

Given any iteration k, for $\tilde{S}$ to be the limit of S_k^X, we need $\tilde{S} \subseteq S_k^X$. By the continuity of the function f and Lemma 2.1, then for any given $x_c \in S_C$, $\exists$ an interval I_c around x^* for any $x^* \in X^*$ such that $f(x_I) > f(x_c), \forall x_I \in I_c$. Let I_C denote the largest of such intervals, i.e. $I_C = \bigcup_c I_c$. Then, the given iteration k, there is a non-zero probability that the next iterate from the forward procedure selects the next iterate from I_C. Specifically, the probability of obtaining the next iterate from I_C is given by $P_k^C = l(I_C)/l(S_k^X)$, where $l()$ denotes the Lebesgue measure. Then:

- If the next iterate $x_{k+1} \in I_C$, then the next slice region S_{k+1}^X would exclude x_c entirely, since $f(x_c) < f(x_{k+1})$.

- If the next iterate $x_{k+1} \notin I_C$, then the next slice region S_{k+1}^X would still include x_c. However, note that since $S_k^X \supset S_{k+1}^X$, $l(S_k^X) > l(S_{k+1}^X)$, and hence $P_k^C < P_{k+1}^C$. Thus, for a sequence of iterates $x_k, x_{k+1}, ... \notin I_C$, $P_k^C < P_{k+1}^C < ...$, and hence $P_k^C \to 1$. Then, $\exists$ a number n such that $x_{k+n} \in I_C$ a.s.

Thus, $\exists$ a slice level S_{k+n}^X such that $x_c \notin S_{k+n}^X$ a.s., and so $\forall m \geq n, x_c \notin S_{k+m}^X$. Hence, $x_c \notin \bigcap_{k \in \mathbb{N}} S_k^X$. Since x_c is an arbitrary point from S_C, this is true for all points in S_C, and hence $\forall x_c \in S_C, x_c \notin \bigcap_{k \in \mathbb{N}} S_k^X$. So, we have $S_C = \emptyset$, which is a contradiction to the definition of S_C. This is true for an arbitrary $\tilde{S}$, and hence true for any $\tilde{S}$. Hence, there does not exist a set $\tilde{S}$ strictly larger than X^* such that $\lim_{k\to\infty} S_k^X = \tilde{S}$. Therefore, $\lim_{k\to\infty} S_k^X = X^*$.

$\square$

It follow from the above Theorem 2.4 that the iterates converge to a global maximum, i.e. $x_k \to x^*, x^* \in X^*$.

References

[1] Richard H Byrd, Peihuang Lu, Jorge Nocedal, and Ciyou Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16(5):1190–1208, 1995.

[2] Kathryn Chaloner and Kinley Larntz. Optimal bayesian design applied to logistic regression experiments. *Journal of Statistical Planning and Inference*, 21(2):191–208, 1989.

[3] P Damien, Jonathan Wakefield, and Stephen Walker. Gibbs sampling for bayesian non-conjugate and hierarchical models by using auxiliary variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 61(2):331–344, 1999.

[4] Hovav A Dror and David M Steinberg. Sequential experimental designs for generalized linear models. *Journal of the American Statistical Association*, 103(481):288–298, 2008.

[5] Valerii Vadimovich Fedorov. *Theory of optimal experiments*. Elsevier, 1972.

[6] Reeves Fletcher and Colin M Reeves. Function minimization by conjugate gradients. *The computer journal*, 7(2):149–154, 1964.

[7] Darrall Henderson, Sheldon H Jacobson, and Alan W Johnson. The theory and practice of simulated annealing. In *Handbook of metaheuristics*, pages 287–319. Springer, 2003.

[8] Radford M Neal. Markov chain monte carlo methods based onslicing'the density function. *Technical Report*, 1997.

[9] Radford M Neal. Slice sampling. *Annals of statistics*, pages 705–741, 2003.

[10] John A Nelder and Roger Mead. A simplex method for function minimization. *Computer journal*, 7(4):308–313, 1965.

[11] Madeleine B Thompson and Radford M Neal. Slice sampling with adaptive multivariate steps: The shrinking-rank method. *arXiv preprint arXiv:1011.4722*, 2010.