Efficient Optimization of the Likelihood Function in Gaussian Process Modelling

A. Butler[a], T.D. Humphries[a], P. Ranjan[b], R.D. Haynes[a,*]

[a]*Department of Mathematics and Statistics, Memorial University, St. John's, NL, Canada, A1C 5S7*
[b]*Department of Mathematics and Statistics, Acadia University, Wolfville, NS, Canada. B4P 2R6*

Abstract

Gaussian Process (GP) models are popular statistical surrogates used for emulating computationally expensive computer simulators. The quality of a GP model fit can be assessed by a goodness of fit measure based on optimized likelihood. Finding the global maximum of the likelihood function for a GP model is typically very challenging as the likelihood surface often has multiple local optima, and an explicit expression for the gradient of the likelihood function is typically unavailable. Previous methods for optimizing the likelihood function (e.g., MacDonald et al. (2013)) have proven to be robust and accurate, though relatively inefficient. We propose several likelihood optimization techniques, including two modified multi-start local search techniques, based on the method implemented by MacDonald et al. (2013), that are equally as reliable, and significantly more efficient. A hybridization of the global search algorithm Dividing Rectangles (DIRECT) with the local optimization algorithm BFGS provides a comparable GP model quality for a fraction of the computational cost, and is the preferred optimization technique when computational resources are limited. We use several test functions and a real application from an oil reservoir simulation to test and compare the performance of the proposed methods with the one implemented by MacDonald et al. (2013) in the R library **GPfit**. The proposed method is implemented in a Matlab package, **GPMfit**.

*Corresponding author
Email addresses: adsb85@mun.ca (A. Butler), thumphries@mun.ca (T.D. Humphries), pritam.ranjan@acadiau.ca (P. Ranjan), rhaynes@mun.ca (R.D. Haynes)

Keywords: BFGS, clustering, computer simulators, Dividing Rectangles, Implicit Filtering, ill-conditioning, nugget.

1. Introduction

Computer simulators are useful tools for modelling complex real world systems that are either impractical, expensive, or time consuming to physically observe. For example, the energy generated by the tides of large ocean basins (Greenberg, 1979), the estimation of the magnetic field generated near the Milky Way (Short et al., 2007), and the analysis of the flow of oil in a reservoir (Aziz and Settari, 1979) – the latter of which motivated this research – can be achieved through the use of computer simulators. That being said, realistic computer simulators can be computationally expensive to run, and as a result are often emulated using statistical models, such as Gaussian Process (GP) models (Sacks et al., 1989).

The maximum likelihood approach for fitting a GP model to deterministic simulator output requires the minimization of the negative log-likelihood, or deviance ($-2\log(L)$). Rasmussen and Williams (2006) proposed the use of either a randomized multi-start conjugate gradient method or Newton's method for this problem. Explicit information about the gradient of deviance, however, cannot easily be obtained, and the deviance surface often has multiple local optima, making the optimization problem challenging (MacDonald et al., 2013). Derivative-free optimization techniques, such as the genetic algorithm used by Ranjan et al. (2011), or the differential evolution algorithm used by Petelin et al. (2011), are robust, but can be computationally inefficient. Gradient approximation methods, such as the Broyden-Fletcher-Goldfarb-Shanno method (BFGS) (Broyden, 1970; Fletcher, 1970; Goldfarb, 1970; Shanno, 1970), are generally faster, but have the potential to converge only locally if poorly initialized. MacDonald et al. (2013) proposed a clustering-based multi-start BFGS algorithm, which allows for a more global search to be performed. Nonetheless, this method demands several executions of BFGS, which is also computationally expensive.

In this paper we investigate several optimization techniques in order to improve the efficiency of the likelihood optimization process. Each technique is a combination of the global and local search components. We propose using the Dividing Rectangles algorithm (DIRECT) (Finkel, 2003) as an alternative to the clustering-based approach for choosing the starting point(s) of BFGS.

In terms of the local search, we compare the efficiency of BFGS with Implicit Filtering (IF), a sophisticated pattern search algorithm developed by Kelley (2011) for multimodal noisy functions. We use several test functions and a real application from an oil reservoir simulation to compare the performance of different optimization techniques; measured by the prediction accuracy (optimized deviance and root mean squared prediction error) and number of deviance function evaluations (FEs) required to optimize the deviance. After an extensive case study we find that a hybrid approach of DIRECT-BFGS is the most efficient optimization technique for fitting such GP models.

The remainder of the paper is outlined as follows. Section 2 describes the GP model and the main components of the newly developed Matlab package **GPMfit**. In Section 3 we briefly outline the optimization techniques used for the deviance minimization. Section 4 provides the results and analysis for several test functions, followed by a real world example in Section 5 where the GP model is fit to an oil reservoir simulator using our proposed method. Concluding remarks are provided in Section 6.

2. The Gaussian Process Model

The GP model requires as input a set of design points, $x_i = (x_{i1}, ..., x_{id})'$, and the corresponding simulator outputs, $y_i = y(x_i)$, where $i = 1, ..., n$, and n is the number of user supplied design points. Here, the prime, $'$, denotes the transpose of vectors and matrices. We assume that the simulator provides a scalar valued output, y_i, for each design point x_i, and we denote $Y = (y_1, ..., y_n)'$ as the $n \times 1$ vector of simulator outputs. The simulator output is modelled as

$$y_i = \mu + z(x_i),$$

where μ is the overall mean, and $z(x_i)$ is a GP with $E[z(x_i)] = 0$, $\text{Var}[z(x_i)] = \sigma^2$, and $\text{Cov}[(z(x_i), z(x_j)] = \sigma^2 R_{ij}$.

The $n \times n$ spatial correlation matrix R defines the degree of dependency between design points, based on their observed simulator value. Following MacDonald et al. (2013), we use the Gaussian correlation matrix, R; a special case of the power exponential correlation family defined as

$$R_{ij} = \prod_{k=1}^{d} \exp\{-10^{\beta_k} |x_{ik} - x_{jk}|^{p_k}\} \quad \text{for all } i, j. \tag{1}$$

Here $p_k = 2$ is the smoothness parameter, and $\beta = (\beta_1, ..., \beta_d)$ is a $1 \times d$ vector of correlation hyper-parameters which measures the sensitivity of the response to the spatial distribution of $|x_{ik} - x_{jk}|^2$ for all $i, j \in \{1, ..., n\}$ and $k \in \{1, ..., d\}$ (Loeppky et al., 2009).

The formulation of correlation function in Equation (1) is slightly different than the popular form of Gaussian correlation, which replaces 10^{β_k} with θ_k (e.g., in Ranjan et al. (2011)). MacDonald et al. (2013) demonstrate that the deviance surface with β-parametrization shown in Equation (1) is much easier to optimize as compared to the commonly used θ-parametrization.

Sacks et al. (1989) show that the best linear unbiased predictor (BLUP) at a given x^* in the input space (typically normalized to $[0, 1]^d$) is

$$\begin{aligned}\hat{y}(x^*) &= \hat{\mu} + r' R^{-1}(Y - \mathbf{1_n}\hat{\mu}) \\ &= \left[\frac{(1 - r' R^{-1}\mathbf{1_n})}{\mathbf{1_n}' R^{-1}\mathbf{1_n}}\mathbf{1_n}' + r'\right] R^{-1} Y \\ &\equiv C' Y,\end{aligned}$$

where $r = [r_1(x^*), ..., r_n(x^*)]'$, and $r_i(x^*) = \text{corr}[z(x^*), z(x_i)]$ is the correlation between $z(x^*)$ and $z(x_i)$. The GP model also returns the associated uncertainty estimate, $s^2(x^*)$, as measured by the mean squared error (MSE),

$$s^2(x^*) = E\left[(\hat{y}(x^*) - y(x^*))^2\right] = \hat{\sigma}^2(1 - 2C'r + C'RC). \tag{2}$$

The model fitting process requires the estimation of μ, σ^2 and β. The closed form estimators of the mean and variance are given by

$$\hat{\mu}(\beta) = (\mathbf{1_n}' R^{-1}\mathbf{1_n})^{-1}(\mathbf{1_n}' R^{-1} Y)$$

and

$$\hat{\sigma}^2(\beta) = \frac{(Y - \mathbf{1_n}\hat{\mu}(\beta))' R^{-1}(Y - \mathbf{1_n}\hat{\mu}(\beta))}{n},$$

respectively, and are used to obtain the profiled negative log-likelihood or deviance (ignoring the unimportant terms like $\log(\sqrt{2\pi})$ and $\log(n)$):

$$\mathcal{L}_\beta = \log(|R|) + n \log\left[(Y - \mathbf{1_n}\hat{\mu}(\beta))' R^{-1}(Y - \mathbf{1_n}\hat{\mu}(\beta))\right], \tag{3}$$

where $\mathbf{1_n}$ is an $n \times 1$ vector of all ones. The most difficult part of fitting the

GP model is to find

$$\beta^* = \arg\min_{\beta \in \mathbb{R}^d} (\mathcal{L}_\beta).$$

Equation (3) shows that evaluating $\mathcal{L}_\beta$ requires computing both the action of the inverse, R^{-1}, and determinant, $|R|$, of the correlation matrix. If any pair of design points are sufficiently close to one another, the matrix R can become near-singular, resulting in unstable computation of R^{-1} and $|R|$. This can lead to an unreliable model fit.

To overcome this instability, we follow a popular technique developed by Sacks et al. (1989), Neal (1997), and Booker et al. (1998), which adds a small "nugget" parameter, δ, to the model fitting procedure. The inclusion of δ smoothes the model predictions, and consequently the GP model fit will no longer exactly interpolate the design points (O'Hagan and Kingman, 1978; Wahba, 1978). To avoid over-smoothing, Ranjan et al. (2011) introduce a lower bound on the nugget parameter,

$$\delta_{lb} = \max\left\{\frac{\lambda_n(\kappa(R) - e^a)}{\kappa(R)(e^a - 1)}, 0\right\},$$

where $\kappa(R) = \lambda_n/\lambda_1$ is the 2-norm condition number and $\lambda_1 \leq \lambda_2 \leq ... \leq \lambda_n$ are the eigenvalues of R. That is, R is simply replaced by $R_\delta = R + \delta_{lb} I$ in (3). The over-smoothness can further be reduced by using the iterative regularization technique proposed by Ranjan et al. (2011). In our simulations, we scale the input-space domain to $[0, 1]^d$, and the design points are generated via a space-filling maximin Latin hypercube design (LHD). The desired threshold for $\kappa(R)$ is e^a, where $a = 25$ is recommended for a space-filling LHD (McKay et al., 1979). Under this configuration, δ_{lb} values remain relatively small, if not zero, and therefore the iterative approach is not needed.

Although there are several choices for the correlation function, we focus on the Gaussian correlation, with $p_k = 2$, because of its smoothness property and popularity in other areas such as machine learning (radial basis kernels) and geostatistics (kriging). In practice, however, we can increase the stability of inverse and determinant computation by slightly lowering the smoothness parameter, p_k, of Equation (1), such that $p_k \lessapprox 2$ (e.g., $p_k = 1.99$). By setting $p_k = 1.99$, the smoothness is not affected significantly and the occurrence of near-singularity is substantially reduced; though not completely resolved as instability may still occur if the design points are extremely close to each other in input space. In this paper, we used $p_k = 2$ with δ_{lb} as given above

for all simulated examples in Section 4, and $p_k = 1.99$ with the same formula for δ_{lb} in the oil reservoir application in Section 5.

3. Optimization Methodology

Our objective function, $\mathcal{L}_\beta$, has a complicated dependency on β, namely in the form of the mean estimator, $\hat{\mu}(\beta)$, the correlation matrix, R, and the nugget parameter, δ. Thus, it is difficult to extract an explicit gradient, $\nabla\mathcal{L}_\beta$, and the optimization methods that require the user to provide an expression for $\nabla\mathcal{L}_\beta$ are not applicable here. We can, however, compute numerical approximations to $\nabla\mathcal{L}_\beta$, which is implicitly performed in both BFGS and IF algorithms. That said, $\mathcal{L}_\beta$ may contain several local optima and flat regions (see Figure 1). Thus, a strictly descent-based optimization approach may not be desirable.

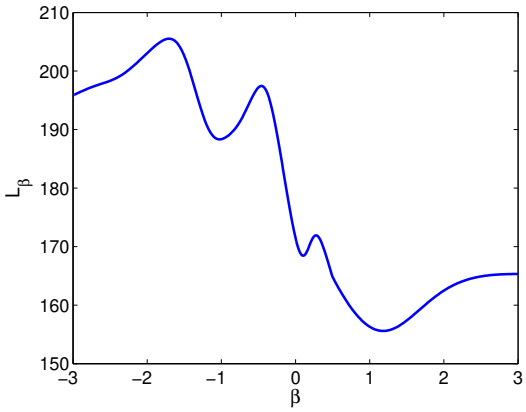


Figure 1: 1-D plot of $\mathcal{L}_\beta$ surface for the 1-D Hump function and a given set of design points. The Hump function is described in Appendix A.

In this section, we first present a brief description of the local optimization algorithms used herein, namely BFGS and IF. In Sections 3.2 and 3.3 we describe the global optimization techniques, namely multi-start clustering and DIRECT. We conclude with a brief discussion on the bound constraints imposed on the β search space.

3.1. BFGS and Implicit Filtering

The BFGS algorithm is a quasi-Newton optimization technique that uses a rank-two Hessian update formula to locate an optimal β (Coleman et al.,

1999). At iteration k, the algorithm obtains a descent direction, $q^{(k)}$, by approximating the Hessian matrix, $H^{(k)}$, and the gradient, $\nabla\mathcal{L}_\beta^{(k)}$, at a current point, $\beta^{(k)}$, and solving

$$H^{(k)}q^{(k)} = -\nabla\mathcal{L}_\beta^{(k)} \cdot \beta^{(k)} \quad \text{for} \quad k = 0, 1, ... \quad .$$

A line search procedure then determines a suitable step-size, $\alpha^{(k)}$, along $q^{(k)}$, in order to obtain an updated solution, $\beta^{(k+1)}$, given by

$$\beta^{(k+1)} = \beta^{(k)} + \alpha^{(k)}q^{(k)}.$$

We use Matlab's built-in unconstrained optimization routine `fminunc` for an implementation of BFGS. We have chosen to use the *medium-scale* implementation of `fminunc`, where the user must supply an initial value, $\beta^{(0)}$.

Implicit Filtering (IF) is a sophisticated, deterministic pattern search algorithm designed by Kelley (2011) for bound constrained optimization. Specifically, IF is a local optimization algorithm that hybridizes a general pattern search algorithm with a BFGS derivative-approximation algorithm. The pattern search is arranged on a stencil, which, given an incumbent solution $\beta^{(k)}$, will evaluate $\mathcal{L}_\beta$ at $\beta^{(k)} \pm hv_j$, in all coordinate directions $j = \{1, ..., d\}$. Here, $v_j = (L_j - U_j)e_j$, where L_j and U_j represent the components of the lower and upper bounds of the search space, respectively, and e_j is the unit coordinate vector. The scale, h, varies as the optimization progresses, according to the sequence $h = \{2^{-m}\}_{m=1}^{7}$, where m increases by 1 each time the current stencil fails to find a more optimal position than the incumbent point. As the pattern search progresses, IF constructs a linear least squares interpolant from previously sampled points. After each pattern search phase, the linear interpolant surface is optimized locally using the BFGS algorithm. This process repeats until an optimal β-parameter is located. The idea is that the pattern search phase, with a suitable step-size, could step over local minima, while the quasi-Newton phase of IF will establish efficient convergence in regions near the global optimum.

The BFGS algorithm, `fminunc`, does not require user specified bound constraints and works efficiently for smooth objective functions. With appropriate Hessian and gradient approximations and an efficient line search, the BFGS algorithm converges superlinearly to a locally optimal β-parameter (Griva et al., 2009) near the initial solution $\beta^{(0)}$. Conversely, IF does require user specified bound constraints and is designed for functions that are noisy,

with many local optima. The convergence of IF is somewhat slower than superlinear, or equivalently, BFGS (Kelley, 2011); however, IF is, in principle, more likely to find the global optimum within the bound constraints, due to its pattern search component. The quasi-Newton phase of both algorithms requires computation and storage of an approximate Hessian matrix and solving the resulting system of equations in order to obtain a suitable descent direction, which is generally computationally expensive.

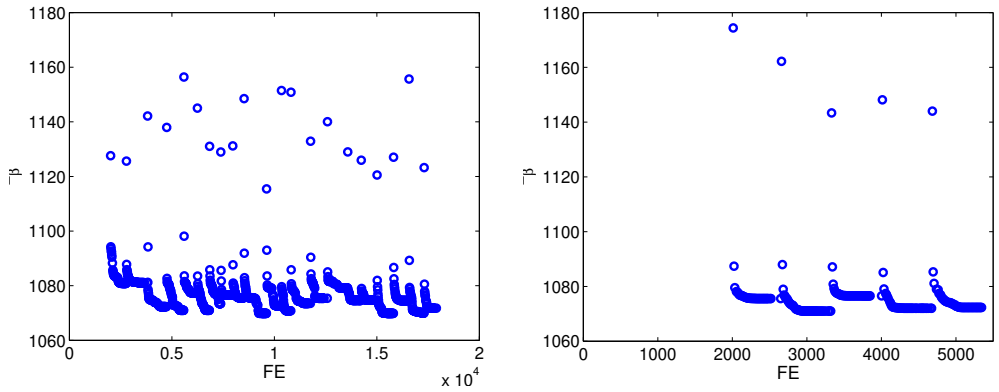
3.2. Clustering-based multistart technique

MacDonald et al. (2013) proposed using a clustering-based multistart BFGS to replace the computationally expensive genetic algorithm used by Ranjan et al. (2011). The BFGS algorithm converges more rapidly, but lacks robustness, in that the algorithm has the potential to get stuck in a local minimum, depending on the starting position, $\beta^{(0)}$. MacDonald et al. (2013) therefore proposed using $2d + 1$ starting points of BFGS to improve the chances of global convergence, where d is the dimension of the simulator input (and therefore of β as well). These points are determined through sampling and k-means clustering method (MacQueen, 1967), as described below.

1. Generate $200d$ β-vectors within the search space, $S_\beta \subset (-\infty, \infty)^d$ (as defined in Section 3.5), using a random maximin Latin hypercube design, and evaluate $\mathcal{L}_\beta$ for each β.
2. From the $200d$ evaluations of $\mathcal{L}_\beta$ obtained from Step 1, select the $80d$ β-vectors with the smallest $\mathcal{L}_\beta$ values.
3. Cluster these $80d$ points from Step 2 into $2d$ groups, using the best of 5 random restarts of k-means clustering method. The $2d$ cluster centers will serve as the $2d$ starting points of BFGS.
4. Evaluate $\mathcal{L}_\beta$ at three equidistant points along the main diagonal of the search space, S_β. The β-vector with the lowest $\mathcal{L}_\beta$ value is chosen as the $(2d + 1)$-th starting point.
5. Begin a run of BFGS from each of the $2d + 1$ starting points.

From thorough experimentation on several test functions, we have observed that executing $2d + 1$ starts of BFGS is excessive, and often results in several runs converging to comparable optima. As an example, Figure 2 shows the convergence of $2d + 1$ multistarts of BFGS (left plot) and $\lceil 0.5d \rceil$ multistarts of BFGS (right plot) as a function of the number of deviance

function evaluations (FE), for the problem of fitting a GP model to the 10-D Rastrigin test function (described in Appendix A). It is apparent that three of the five BFGS runs shown in the right plot converge to a value that is comparable to the best value found out of twenty-one BFGS runs in the left plot. We therefore propose reducing the number of cluster centers to $\lceil 0.5d \rceil$ and eliminating the additional starting point obtained from sampling along the main diagonal. We implement two clustering-based techniques: (i) $\lceil 0.5d \rceil$ multistarts of BFGS (the same method as in MacDonald et al. (2013)), and (ii) $\lceil 0.5d \rceil$ multistarts of IF.



(a) $2d + 1$ multi-start BFGS. (b) $\lceil 0.5d \rceil$ multi-start BFGS.

Figure 2: Plot showing the convergence of each BFGS start versus the cumulative number of FEs (includes the initial $200d = 2000$ FEs used for clustering), for fitting the 10-D Rastrigin function. The starting points are generated using a random maximin LHD.

The simulation results (in Table 1 and Figures 2 and 4) show that the $\mathcal{L}_\beta$ value returned by BFGS and IF do not usually change significantly after the first few iterations. Therefore, performing a full search (until the stopping criterion is satisfied) from each starting point $\beta^{(0)}$ may be excessive as well. We therefore investigate a two stage multistart IF method, denoted by IF-2. The process starts with a clustering-based, $\lceil 0.5d \rceil$ multistart IF approach, where each run of IF is limited by a budget of $20d$ FEs. In the second stage, the single run of IF that returns the lowest $\mathcal{L}_\beta$ value will then run to completion.

3.3. DIRECT Hybrid Techniques

Dividing Rectangles (DIRECT) is a derivative-free, block partitioning algorithm that sequentially samples points in the search space and partitions

the domain into hyper-rectangles based on the objective function value (here, $\mathcal{L}_\beta$) at the sampled points. Hyper-rectangles are then identified as being potentially optimal if they contain a sampled point whose function value is more optimal than the sampled points contained by all other hyper-rectangles of equal size. Each potentially optimal hyper-rectangle is then divided into thirds along its longest dimension, and the process repeats. Figure 3 provides a 2-dimensional visualization of how DIRECT samples and divides its search space. The left panel in Figure 3 shows the initial sampling phase and partitioning of the domain. The bold rectangles in the middle panel of Figure 3 are identified as being potentially optimal in the following iteration. The rectangles are then partitioned in turn (rightmost figure), and three new rectangles are identified as potentially optimal. The alternating process of partitioning and then identifying potentially optimal rectangles continues, until a stopping criterion is met. See Finkel (2003) for more details.

DIRECT is specifically designed as a global optimization approach, however, since it provides such a thorough exploration of search space it can be slow to converge locally. We therefore use DIRECT to provide a single, somewhat optimized starting point, $\beta^{(0)}$, from which to begin a run of either BFGS or IF, thereby eliminating the need for a multi-start approach. For ease of comparison with clustering-based approach, we provide a budget of $200d$ FEs to DIRECT.

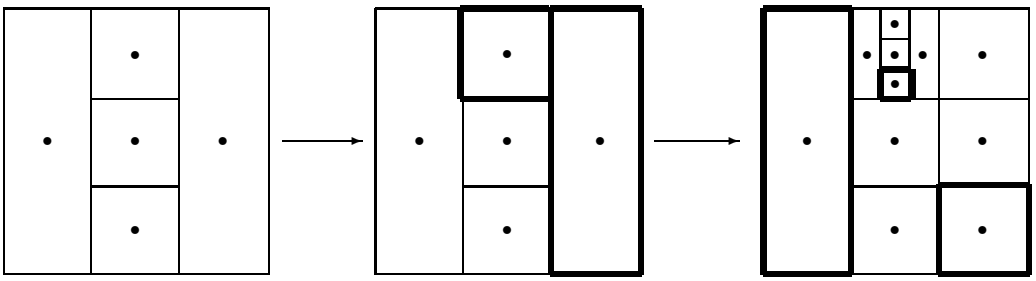


Figure 3: A few iterations of the Dividing Rectangles (DIRECT) algorithm. Bolded rectangles are identified as being potentially optimal and are divided in the following iteration. The •'s denote the sampled points.

3.4. Boundaries of the Search Space

Many optimization techniques, including IF and DIRECT, require user-supplied bound constraints on the optimization parameters, β. Although the exact position of the global optimum in β-space is unknown, we can determine a region, S_β, where the optimum is likely to be found. MacDonald et al. (2013) use the structural form of the correlation matrix R to provide an approximate bound for R_{ij}. Specifically

$$\exp(-5) \approx 0.0067 \leq R_{ij} \leq 0.9999 \approx \exp(-10^{-4}),$$

or equivalently,

$$10^{-4} \leq \sum_{k=1}^{d} 10^{\beta_k} |x_{ik} - x_{jk}|^2 \leq 5.$$

Since the assumed input-space is $[0,1]^d$, and the design points are generated via a maximin LHD, the approximate spatial distribution of the $10d$ design points that we use is at most $|x_{ik} - x_{jk}| \approx 1/10$ in any dimension k. Moreover, if we assume that the simulator function is equally smooth in all coordinate directions, i.e., $\beta_1 \approx \beta_2 \approx \cdots \beta_k$, then the set of β values that is likely to contain the global optimum is

$$S_\beta = \{(\beta_1, ..., \beta_d) : -2 - \log_{10}(d) \leq \beta_k \leq \log_{10}(500) - \log_{10}(d), k = 1, ..., d\}.$$

The starting points, $\beta^{(0)}$, determined using either clustering or DIRECT, will be confined to the region S_β.

BFGS is an unbounded optimization algorithm and does not require any user-specified bound constraints on the optimization variables, whereas IF does. The step-size calculated in the pattern search phase of IF is proportional to the physical size of the user supplied bound constraints, which we denote by S_β^{IF}. From our experimentation on the test functions, we have noticed that if the size of S_β^{IF} is too small, the optimization efficiency is compromised as IF requires a large number of FEs in order to converge to an optimal β-parametrization. Conversely, if S_β^{IF} is too large, IF has the tendency to "jump" around the potentially optimal regions of $\mathcal{L}_\beta$, resulting in convergence to a suboptimal β value.

MacDonald et al. (2013) note that the optimal β values are rarely large and positive, and hence, we impose bound constraints on IF in which the

negative β region occupies a larger portion of the domain, i.e.,

$$S_\beta^{IF} = \{(\beta_1, ..., \beta_d) : d(-2 - \log_{10}(d)) \leq \beta_k \leq \log_{10}(500), k = 1, ..., d\}.$$

We acknowledge that the β-value that globally minimizes $\mathcal{L}_\beta$ may occasionally be positioned outside the provided bounds, S_β and S_β^{IF}. Therefore, included in the **GPMfit** package is the option to multiplicatively expand or contract the default bound constraints.

4. Simulation Results

We use seven test functions, with input dimensions varying from $d = 1$ to $d = 12$ to compare the performance of the different optimization techniques discussed in Section 3. The formulae for all the test functions are provided in Appendix A. The performance of each optimization technique is averaged over 25 simulations. For each simulation, $10d$ training design points (x_i) and $100d$ validation prediction points (x_i^*) are chosen in $[0, 1]^d$. The initial sample points in β-space for the clustering procedure are randomly generated in S_β using the Latin hypercube design. All simulations were performed using 64-bit Matlab 2012(b) on a Gentoo Linux operating system with a Core 2 Quad Xeon processor.

4.1. Optimization Accuracy

Recall that our objective is to minimize $\mathcal{L}_\beta$. Typically, the parameter estimate that corresponds to the smallest $\mathcal{L}_\beta$ will provide the most accurate model fit, as measured by the average relative root mean square prediction error (RMSPE) between the GP model fit and the true simulator (test function) response. That is,

$$RMSPE = \sqrt{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \Big/ \sum_{i=1}^{N} y_i^2}, \quad \text{where } N = 100d. \tag{4}$$

We note that in most real applications, the true RMSPE values cannot be calculated, as the true simulator outputs are unknown at the validation points. One can, however, use the average MSE estimates (see Equation (2)) for performance comparison. The consistency of RMSPE values over 25

simulations is measured by the standard error in the RMSPE value, given by

$$\text{Std. Err.} = \sigma_{RMSPE}/\sqrt{25}, \tag{5}$$

where σ_{RMSPE} denotes the standard deviation of the RMSPE values. A standard error of one order of magnitude less than the corresponding RMSPE value indicates that our results are fairly consistent over all 25 simulations.

4.2. Convergence Efficiency

We measure the efficiency of an optimization technique by the number of likelihood function evaluations (FEs) required for optimization of $\mathcal{L}_\beta$. Using Matlab's profiler, we determined that evaluating $\mathcal{L}_\beta$ constituted the bulk of the computational load for all optimization techniques considered. In particular, we determined that computation of the correlation matrix, R_δ, demands anywhere from $60\%-90\%$ of the total computation time, depending on the simulator input dimension, d. Computation of R_δ is nested within the calculation of $\mathcal{L}_\beta$ and is evaluated once per FE. Therefore, the number of FEs, which is not affected by issues like server load, can be used in place of computation time as a fair measure of optimization efficiency.

4.3. Discussion

Table 1 summarizes the accuracy ($\mathcal{L}_\beta$ and RMSPE) and efficiency (FE) of each optimization technique for six of the seven test functions, namely the 2-D Goldstein-Price function, the 5-D Schwefel Function, the 6-D Hartmann function, the 10-D Rastrigin function, the 10-D Rosenbrock function and the 12-D Perm Function (see Appendix A for closed form expressions). Results for fitting the 1-D Hump function are not shown, as the performance of all the techniques was essentially the same for this simple test function. The $\%\Delta$ notation in Table 1 denotes the percent relative difference between value of the performance measure returned by a given technique and the best value found among all techniques. The standard errors are not included in this table; we found that for all cases the standard error was indeed at least one order of magnitude less than the corresponding RMSPE, suggesting that each technique is consistently able to provide the same GP model quality.

The results in Table 1 show that the $\lceil 0.5d \rceil$ multi-start techniques and DIRECT-based techniques provide efficient and reliable alternatives to the $2d+1$ BFGS technique. We observe that the $\lceil 0.5d \rceil$ multi-start and DIRECT-based techniques require anywhere from 20% to 90% fewer FEs than the

Algorithm	**Goldstein-Price (2-D)**			**Schwefel (5-D)**		
	$\%\Delta\mathcal{L}_\beta$	%ΔRMSPE	FE	$\%\Delta\mathcal{L}_\beta$	%ΔRMSPE	FE
$\lceil 0.5d \rceil$ BFGS	0.017	0.201	<u>439</u>	0.100	2.885	1859
$2d+1$ BFGS	–	3.213	653	–	2.885	4277
$\lceil 0.5d \rceil$ IF	0.007	–	518	0.206	0.962	2381
$2d+1$ IF	–	3.213	995	0.074	2.404	5735
$\lceil 0.5d \rceil$ IF-2	0.007	–	546	0.272	1.442	1677
DIRECT-BFGS	–	3.213	449	0.232	2.885	<u>1296</u>
DIRECT-IF	–	3.213	498	0.243	–	1304
	Hartmann (6-D)			**Rastrigin (10-D)**		
	$\%\Delta\mathcal{L}_\beta$	%ΔRMSPE	FE	$\%\Delta\mathcal{L}_\beta$	%ΔRMSPE	FE
$\lceil 0.5d \rceil$ BFGS	0.034	–	2068	0.071	3.865	5553
$2d+1$ BFGS	0.102	1.064	5526	–	3.865	17853
$\lceil 0.5d \rceil$ IF	1.703	6.991	2293	0.624	–	4346
$2d+1$ IF	–	0.304	7497	0.134	2.899	17284
$\lceil 0.5d \rceil$ IF-2	1.223	8.511	1866	0.545	0.483	4011
DIRECT-BFGS	0.207	1.216	<u>1526</u>	0.255	1.932	<u>2682</u>
DIRECT-IF	0.207	1.216	1533	0.287	1.932	2774
	Rosenbrock (10-D)			**Perm (12-D)**		
	$\%\Delta\mathcal{L}_\beta$	%ΔRMSPE	FE	$\%\Delta\mathcal{L}_\beta$	%ΔRMSPE	FE
$\lceil 0.5d \rceil$ BFGS	–	–	4562	0.003	1.370	8170
$2d+1$ BFGS	–	–	16245	–	1.643	27622
$\lceil 0.5d \rceil$ IF	–	–	6654	0.010	–	12411
$2d+1$ IF	–	–	24725	0.001	2.055	44375
$\lceil 0.5d \rceil$ IF-2	0.031	3.196	3775	0.025	–	4935
DIRECT-BFGS	0.003	0.457	<u>2332</u>	0.011	0.545	<u>3338</u>
DIRECT-IF	0.003	0.457	2654	0.012	0.685	3459

Table 1: Performance comparison of different likelihood optimization techniques on six test functions. Dashed values in the $\%\Delta\mathcal{L}_\beta$ and %ΔRMSPE columns indicate that the best overall value was found by this algorithm. Underlined values in the FE column indicate the smallest number of FEs required by any algorithm.

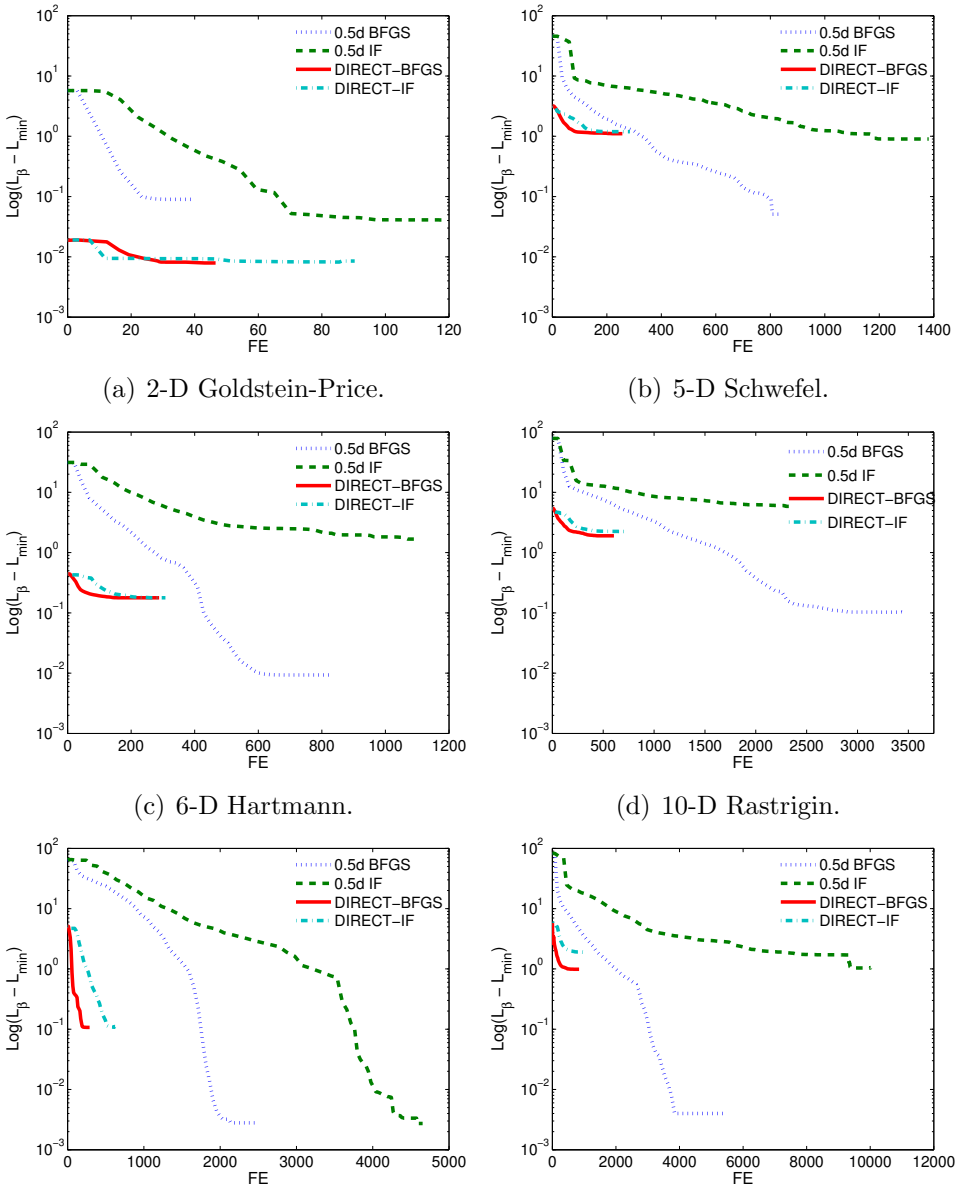
$2d+1$ BFGS technique (depending on the dimension of the test function), while maintaining a comparable level of optimization accuracy. The relative difference in the $\mathcal{L}_\beta$ value returned by each technique is typically less than 1%. As a result, each optimization technique provides comparable GP model quality, as measured by the RMSPE value. Table 1 shows, however, that for all test functions, the $\lceil 0.5d \rceil$ multi-start IF-2 technique returns a larger

average $\mathcal{L}_\beta$ value and requires anywhere from 10% to 50% more FEs than both of the DIRECT-based methods. As a result, IF-2 is not included in the **GPMfit** package, as more accurate and efficient techniques are clearly available.

The results in Table 1 suggest that a slightly suboptimal $\mathcal{L}_\beta$ value can result in an equal, or even slightly better GP model qualityas measured by the RMSPE. For example, for fitting the 10-D Rastrigin function, DIRECT-BFGS provides an RMSPE value that is 1.93% smaller than the RMSPE value returned by $2d+1$ BFGS, despite converging to a $\mathcal{L}_\beta$ value that is sub-optimal by 0.255%. This non-monotonic relationship between optimal $\mathcal{L}_\beta$ and GP model quality is presented in a recent paper by Kalaitzis and Lawrence (2011), who argue that one can maintain the quality of the GP model even with a slightly suboptimal $\mathcal{L}_\beta$ value. Furthermore, Nguyen et al. (2011) suggest that, due to the difficulties in finding optimal β-parameters, particularly when the training data (x_i, y_i) is sparse, GP models can be prone to overfitting, which can lead to larger than expected RMSPE values. Motivated by this non-monotonic relationship, our goal is to efficiently determine a sufficiently optimal $\mathcal{L}_\beta$ value, without compromising the resulting GP model quality.

Figure 4 shows the convergence performance of both the BFGS and IF algorithms after the initial $\beta^{(0)}$ point(s) are determined using either clustering or DIRECT. For the multi-start clustering-based techniques, the convergence plots are displayed as though each run of BFGS or IF were implemented in parallel, from which the best of the $\lceil 0.5d \rceil$ $\mathcal{L}_\beta$ values is plotted versus the cumulative number of FEs. The optimization performance for each technique has been averaged over all 25 simulations, and is plotted on a semi-log scale as the absolute difference between $\mathcal{L}_\beta$ and the minimum $\mathcal{L}_\beta$ value, $\mathcal{L}_{\min}$, (rounded down to the nearest 10^{-2}) determined by one of the four techniques.

We first observe that, for both the DIRECT and multi-start approaches, BFGS is more efficient than IF, and converges to a more optimal value of $\mathcal{L}_\beta$. This suggests that BFGS is generally better suited to this optimization problem than IF. Secondly, with the exception of the 2-D test case (Figure 4(a)), the $\lceil 0.5d \rceil$ BFGS technique converges to the best $\mathcal{L}_\beta$ value. In general, however, the difference between the $\mathcal{L}_\beta$ value returned by the $\lceil 0.5d \rceil$ BFGS technique and DIRECT-based methods is on the order of 10^{-1} to 10^0, which has only a small effect on the resulting quality of the GP model, as measured by the RMSPE. Moreover, Figure 4 shows that once the starting $\beta^{(0)}$ point(s) have been determined, the DIRECT-based methods require approximately



(a) 2-D Goldstein-Price.

(b) 5-D Schwefel.

(c) 6-D Hartmann.

(d) 10-D Rastrigin.

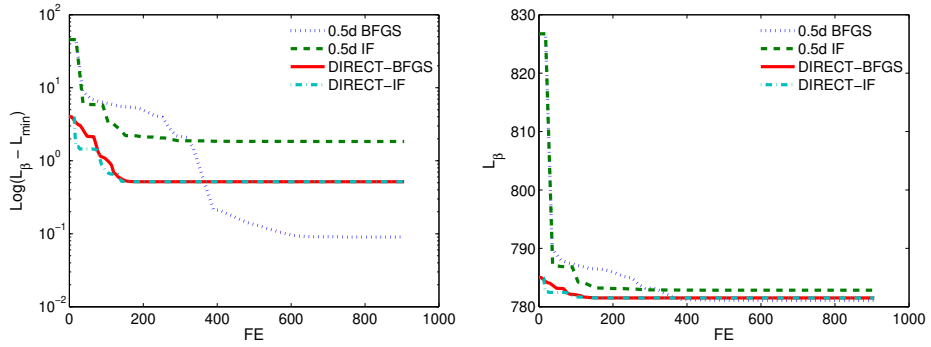
(e) 10-D Rosenbrock.

(f) 12-D Perm.

Figure 4: Semi-log plots comparing the $\mathcal{L}_\beta$ optimization performance of the single start DIRECT-BFGS and DIRECT-IF techniques and the multi-start $\lceil 0.5d \rceil$ BFGS and IF techniques, averaged over 25 simulations.

$\frac{1}{\lceil 0.5d \rceil}$ as many FEs as the multi-start clustering techniques; this represents a substantial increase in optimization efficiency, particularly as d increases. Thus, the DIRECT-based approaches are able to find an optimum that is only slightly worse than those found by the significantly more expensive clustering-based approaches.

We ran an additional experiment to determine whether providing additional FEs to the DIRECT-based methods would result in these methods converging to the optimal $\mathcal{L}_\beta$ value found by $\lceil 0.5d \rceil$ BFGS. Figure 5 compares the optimization performance of the DIRECT-based and $\lceil 0.5d \rceil$ multi-start clustering techniques for a single GP model realization of the 5-D Schwefel function. For this particular case, the local search techniques used after DIRECT have been allotted 900 FEs, which is the number of FEs that were required for $\lceil 0.5d \rceil$ BFGS to converge to the optimal $\mathcal{L}_\beta$. From the logarithmic plot (Figure 5(a)), we can see that there is no improvement in the solution found by the DIRECT-BFGS and DIRECT-IF methods after roughly 200 FEs, indicating that allotment of additional FEs provides no benefit to these approaches. We note again, however, that when plotted on regular axes, as in Figure 5(b), the discrepancy in the $\mathcal{L}_\beta$ values returned by the various methods is small.



(a) Semi-log Convergence Plot.

(b) Convergence plot.

Figure 5: Semi-log convergence plot and convergence plot comparing the $\mathcal{L}_\beta$ optimization performance of the DIRECT-based techniques and the multi-start clustering techniques.

Figures 4 and 5, show that the DIRECT algorithm is able to determine a more optimal starting position, $\beta^{(0)}$, for initialization of BFGS or IF. This enables us to implement a single run of BFGS or IF, with only a small loss of

optimization accuracy. These results were observed for all test functions and support a fundamental conclusion; if the user has significant computational resources at their disposal and if a highly accurate optimal $\mathcal{L}_\beta$ is desired, then the multi-start $\lceil 0.5d \rceil$ BFGS technique is preferred. If computational resources are limited, however, then one can use DIRECT-BFGS to obtain comparable GP model fit for a fraction of the computational cost. From these results, we are able to establish an overall performance ranking of each optimization technique, shown in Table 2.

Rank	**Algorithm**	**# of starts**
1	DIRECT-BFGS	1
2	DIRECT-IF	1
3	BFGS	$\lceil 0.5d \rceil$
4	IF	$\lceil 0.5d \rceil$
5	BFGS	$2d + 1$
6	IF	$2d + 1$
7	IF-2	$\lceil 0.5d \rceil$

Table 2: Ranks of the optimization techniques based on their overall average performance for the seven test functions.

5. Oil Reservoir Simulator Example

Determining optimal drilling locations for production and injection wells in an oil reservoir is a problem of considerable industrial interest (see for instance Yeten et al. (2003); Bangerth et al. (2006); Onwunalu and Durlofsky (2010)). The variables in this problem correspond to positional parameters for each well; in this example, we will consider only vertical wells, each of which can be parameterized by its (x_1, x_2) co-ordinates, representing a grid location in the discrete reservoir model. The well locations serve as input to a computationally expensive complex reservoir simulator – in our case, the MATLAB Reservoir Simulator (MRST) (Lie et al., 2011; SINTEF Applied Mathematics, 2012). The simulator output, along with various economic parameters, are then usually combined to provide the net present value (NPV) of the produced oil. The goal is to determine the configuration of wells that yields the best NPV.

We consider two problems using a simple 2-D reservoir model based on a 60×50 grid. For the first placement problem, we assume that two injection wells (×) and one production well (∘) have already been drilled at the positions shown in Figure 7(a), and the goal is to find the optimal location for the second production well. The NPV surface corresponding to this problem is shown in Figure 7(a). One could use an expected improvement based sequential design scheme (Jones et al., 1998) for finding this optimal location; the key component in such a sequential optimization is to efficiently emulate (i.e., fit a GP model to) the simulator response after every iteration of this sequential procedure. In this paper, we focus on this first step of fitting a GP model-based surrogate to the simulator output. For the second problem, we allow the positions of all four wells to be chosen freely, meaning that the NPV now depends on 8 variables. Both of these problems are variants of one considered in Humphries et al. (2013).

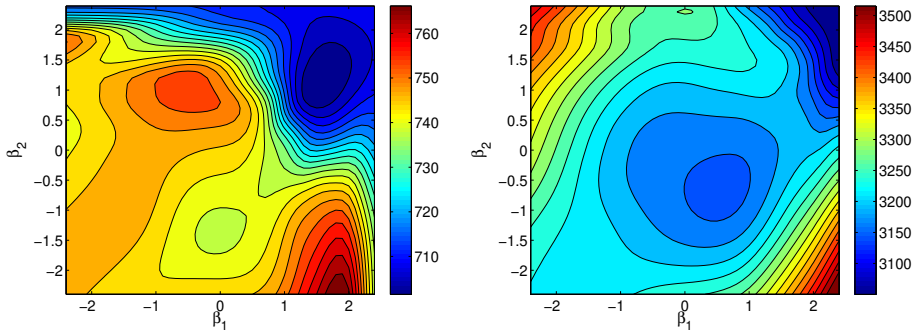
5.1. 2-D Reservoir Simulator

Table 3 compares the performance of three methods for fitting the GP model to the 2-D reservoir simulator: (i) $\lceil 0.5d \rceil$ BFGS, (ii) $2d+1$ multi-start BFGS, and (iii) DIRECT-BFGS. The number of training design points used, n, ranges from 20 to 100. The performance of each optimization technique is averaged over 25 simulations. For every value of n, the $2d+1$ BFGS technique returns the best $\mathcal{L}_\beta$ value. The relative percent difference of the $\mathcal{L}_\beta$ value returned by the DIRECT-BFGS technique, however, is always less than 0.1%. Moreover, in all cases, DIRECT-BFGS returns the smallest RMSPE value and requires anywhere from 37% to 45% fewer FEs than $2d+1$ BFGS, and is therefore the preferred technique.

A close look on one realization (see Figure 6) reveals that the likelihood surface changes significanly as the number of design points increases. In particular, when $n = 80$, a local optimum with a large basin of attraction is created near the center of the S^β domain. As a result, the $\lceil 0.5d \rceil$ multi-start BFGS technique, which is a single-start technique when $d = 2$, converges to this sub-optimal point when initialized in that region. DIRECT, on the other hand, is able to determine an starting point that is significantly closer to the global optimum (located in the top right corner) and thus avoids converging to the sub-optimal point. This explains why the $\mathcal{L}_\beta$ and corresponding RMPSE values determined by $\lceil 0.5d \rceil$ BFGS are significantly higher than those values determined by $2d+1$ BFGS and DIRECT-BFGS, when $n \geq 40$.

Algorithm	**n = 20**			**n = 40**		
	$\%\Delta\mathcal{L}_\beta$	%ΔRMSPE	FE	$\%\Delta\mathcal{L}_\beta$	%ΔRMSPE	FE
$\lceil 0.5d \rceil$ BFGS	0.110	2.461	<u>435</u>	2.153	41.602	452
$2d+1$ BFGS	–	2.461	691	–	3.359	721
DIRECT-BFGS	0.029	–	<u>435</u>	0.004	–	<u>440</u>
	n = 80			**n = 100**		
	$\%\Delta\mathcal{L}_\beta$	%ΔRMSPE	FE	$\%\Delta\mathcal{L}_\beta$	%ΔRMSPE	FE
$\lceil 0.5d \rceil$ BFGS	2.757	47.734	461	2.648	22.459	465
$2d+1$ BFGS	–	–	756	–	3.476	822
DIRECT-BFGS	–	–	<u>443</u>	0.072	–	<u>459</u>

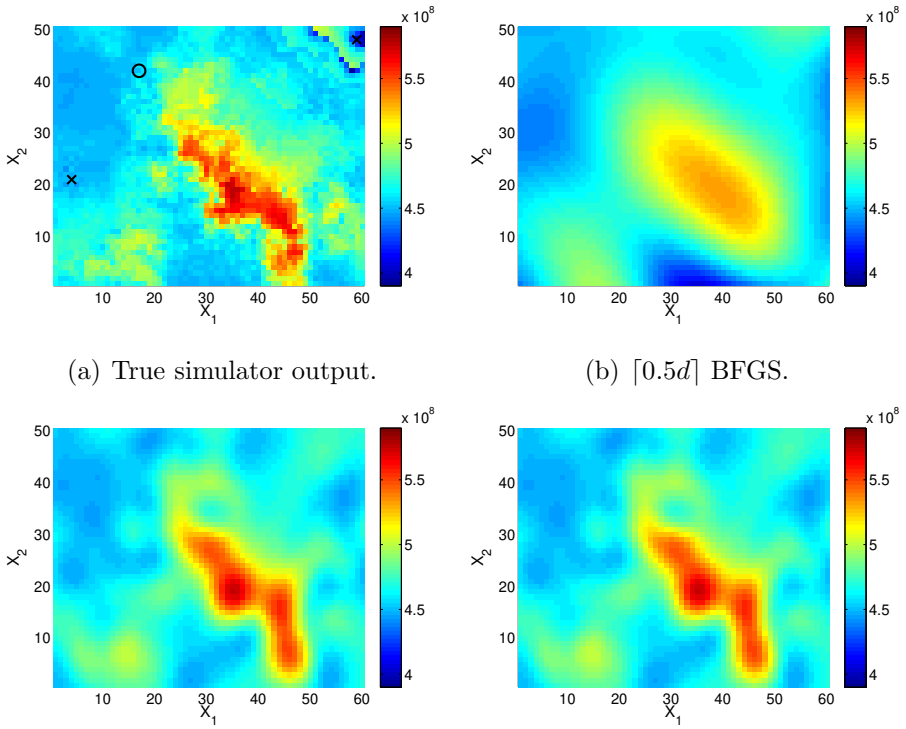
Table 3: Performance comparison of different $\mathcal{L}_\beta$ optimization methods for the 2-D reservoir simulator. Dashed values in the $\%\Delta\mathcal{L}_\beta$ and %ΔRMSPE columns indicate that the best overall value was found by this algorithm. Underlined values in the FE column indicate the smallest number of FEs required by any algorithm.



(a) $n = 20$ (b) $n = 80$

Figure 6: Comparing $\mathcal{L}_\beta$ surfaces of the 2-D reservoir simulator for varying design points.

Figure 7 shows the true simulator output and an example of the GP models that were found using each of the three optimization techniques, with $n = 100$ design points. The $2d + 1$ BFGS and DIRECT-BFGS approaches provide near identical GP model predictions, with DIRECT-BFGS requiring a fraction of the computational cost. As mentioned, the $\lceil 0.5d \rceil$ BFGS technique converges to a sub-optimal β-parameter, and as a result the GP model quality is significantly worse. In general we observe that for small dimensional simulators, DIRECT-BFGS outperforms the $\lceil 0.5d \rceil$ multi-start



(a) True simulator output.

(b) $\lceil 0.5d \rceil$ BFGS.

(c) $2d+1$ BFGS.

(d) DIRECT-BFGS.

Figure 7: True 2-D reservoir simulator output and GP fit surface obtained through $\mathcal{L}_\beta$ optimization using the $\lceil 0.5d \rceil$ BFGS, $2d+1$ BFGS and DIRECT-BFGS technique, with $n = 100$. The locations of the two existing injection ($\times$) wells and single production ($\circ$) well are shown in plot (a).

BFGS technique in terms of both optimization accuracy and efficiency (Tables 1 and 3, Figure 7). The $2d+1$ multi-start BFGS method often provides a more accurate GP model fit for any number of design points, but requires up to 80% more FEs than both DIRECT-BFGS and $\lceil 0.5d \rceil$ BFGS for a 2-D function. This example shows that DIRECT-BFGS provides an efficient alternative to the $2d+1$ multi-start BFGS approach, without sacrificing model accuracy.

5.2. 8-D Reservoir Simulator

In our second example, we fit a GP model to the oil reservoir simulator with 8 variables (the positions of all four wells), again using $\lceil 0.5d \rceil$ BFGS,

$2d+1$ BFGS and DIRECT-BFGS for $\mathcal{L}_\beta$ optimization. The GP model is initialized using both $10d$ and $20d$ design points, and predicts for $100d$ points of unknown function value. The performance of each of the three optimization techniques is averaged over 25 simulations.

Table 4 compares the $\mathcal{L}_\beta$ optimization and resulting GP model fitting performance for each of the three techniques. Again, we observe that the $2d+1$ BFGS technique converges to the best $\mathcal{L}_\beta$ value on average. Nonetheless, this method requires roughly 12000 FEs, which is more than 3 times the number of FEs required by $\lceil 0.5d \rceil$ BFGS and almost 6 times the number of FEs required by DIRECT-BFGS. Moreover, DIRECT-BFGS, on average, provides the highest quality GP model, as measured by the RMSPE value, despite converging to a slightly sub-optimal $\mathcal{L}_\beta$ value. The results obtained in fitting the GP model to a true reservoir simulator provide evidence that, in practice, one can employ the single start DIRECT-BFGS optimization technique to greatly increase the efficiency of the GP model fitting procedure, without compromising the quality of the model.

Algorithm	**n = 80**			**n = 160**		
	$\%\Delta\mathcal{L}_\beta$	%ΔRMSPE	FE	$\%\Delta\mathcal{L}_\beta$	%ΔRMSPE	FE
$\lceil 0.5d \rceil$ BFGS	0.013	0.711	3706	0.007	1.139	3567
$2d+1$ BFGS	–	2.370	11896	–	1.635	11893
DIRECT-BFGS	0.032	–	<u>2070</u>	0.026	–	<u>2005</u>

Table 4: Performance comparison of different $\mathcal{L}_\beta$ optimization methods for the 8-D reservoir simulator. Dashed values in the $\%\Delta\mathcal{L}_\beta$ and %ΔRMSPE columns indicate that the best overall value was found by this algorithm. Underlined values in the FE column indicate the smallest number of FEs required by any algorithm.

6. Conclusion

In this paper we have investigated several techniques for efficient optimization of the deviance function, $\mathcal{L}_\beta$, in GP modelling. These techniques provide the foundation for an improved Matlab package, **GPMfit**. The results obtained from simulated examples and real applications show that using $2d+1$ multi-starts of BFGS is computationally expensive, and that we are able to significantly improve the optimization efficiency by reducing the num-

ber of multi-starts to $\lceil 0.5d \rceil$, while maintaining the quality of the GP model in most cases.

Implicit Filtering proves to be slightly less accurate and efficient than BFGS, and therefore is included in the **GPMfit** solely as a secondary option. The modified multi-start technique IF-2 is generally unreliable; specifically, the slight reduction in computational cost that is gained does not outweigh the reduced accuracy, particularly when more efficient and robust optimization techniques exist.

Replacing the $\lceil 0.5d \rceil$ multi-start technique with the DIRECT optimization algorithm enables us to further reduce the number of starts of BFGS or IF from $\lceil 0.5d \rceil$ to 1. After an initial β_0 value has been determined, the DIRECT-based hybrid techniques require approximately $\frac{1}{\lceil 0.5d \rceil}$ as many function evaluations as the multi-start techniques and provide an almost equally accurate model fit. As a result, the DIRECT-BFGS hybrid technique is the default $\mathcal{L}_\beta$ optimization algorithm used in the **GPMfit** package. The slightly less efficient DIRECT-IF hybrid technique is also included in the **GPMfit** package as an additional optimization option, along with the more computationally expensive $2d + 1$ and $\lceil 0.5d \rceil$ multi-start BFGS and IF techniques.

Future Work: Training a GP model by expected improvement (EI) is a popular technique for optimization of computationally expensive simulators, in which repeatedly evaluating the simulator by use of traditional optimization routines is inefficient (Jones et al., 1998). We are currently exploring the use of GP models trained by EI sequential design as an efficient alternative to traditional optimization routines for global optimization of computationally expensive simulators.

Supplementary Material

The open source Matlab package **GPMfit** is available for download on `SourceForge.net`. See `Readme.txt` for detailed instruction. The main functions are `model_fit.m` and `predictor_iterative.m`.

Acknowledgments

Haynes, Ranjan and Butler would like to acknowledge the support of the Natural Sciences and Engineering Research Council of Canada's Discovery Grant and USRA programs. Humphries was supported by the Research Development Corporation of Newfoundland and the Atlantic Canada Opportunities Agency.

References

Aziz, K., Settari, A., 1979. Petroleum Reservoir Simulation. Applied Science Publishers, London.

Bangerth, W., Klie, H., Wheeler, M., Stoffa, P., Sen, M., 2006. On Optimization Algorithms for the Reservoir Oil Well Placement Problem. Computational Geosciences 10, 303–319.

Booker, A.J., Jr., J.E.D., Frank, P.D., Serafini, D.B., Torczon, V., Trosset, M.W., 1998. A Rigorous Framework for Optimization of Expensive Functions by Surrogates. Structural Optimization 17, 1–13.

Broyden, C., 1970. The Convergence of a Class of Double-Rank Minimization Algorithms. Journal of Applied Mathematics 6, 76–90.

Coleman, T., Branch, M.A., Grace, A., 1999. Optimization Toolbox: For Use with Matlab. The Math Works Inc.

Finkel, D.E., 2003. DIRECT Optimization Algorithm User Guide. Center for Research in Scientific Computation.

Fletcher, R., 1970. A New Approach to Variable Metric Algorithms. The Computer Journal 13, 317–322.

Goldfarb, D., 1970. A Family of Variable Metric Updates Derived by Variational Means. Mathematics of Computation 24, 23–26.

Greenberg, D., 1979. A Numerical Model Investigation of Tidal Phenomena in the Bay of Fundy and Gulf of Maine. Marine Geodesy 2, 161–187.

Griva, I., Nash, S., Sofer, A., 2009. Linear and Nonlinear Optimization. Society for Industrial and Applied Mathematics. pp. 355–448.

Humphries, T., Haynes, R., James, L., 2013. Simultaneous and Sequential Approaches to Joint Optimization of Well Placement and Control. Computational Geosciences Submitted.

Jones, D.R., Schonlau, M., Welch, W., 1998. Efficient Global Optimization of Expensive Black-Box Functions. Journal of Global Optimization 13, 445–492.

Kalaitzis, A.A., Lawrence, N.D., 2011. A Simple Approach to Ranking Differentially Expressed Gene Expression Time Courses Through Gaussian Process Regression. BMC Bioinformatics 12.

Kelley, C., 2011. Implicit Filtering. Society for Industrial and Applied Mathematics.

Lie, K.A., Krogstad, S., Ligaarden, I., Natvig, J., Nilsen, H., Skaflestad, B., 2011. Open-Source MATLAB Implementation of Consistent Discretisations on Complex Grids. Computational Geosciences , 1–26.

Loeppky, J.L., Sacks, J., Welch, W.J., 2009. Choosing the Sample Size of a Computer Experiment: A Practical Guide. Technometrics 51, 366–376.

MacDonald, B., Ranjan, P., Chipman, H., 2013. GPfit: An R package for Gaussian Process Model Fitting Using a New Optimization Algorithm. `arXiv:1305.0759`.

MacQueen, J.B., 1967. Some Methods for classification and Analysis of Multivariate Observations, in: Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability, University of California Press. pp. 281–297.

McKay, M.D., Beckman, R.J., Conover, W.J., 1979. A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code. Technometrics 21, 239–245.

Neal, M.R., 1997. Monte Carlo Implementation of Gaussian Process Models for Bayesian Regression and Classification.

Nguyen, H.M., Couckuyt, I., Knockaert, L., Dhaene, T., Gorissen, D., Saeys, Y., 2011. An Alternative Approach to Avoid Overfitting for Surrogate Models, in: Winter Simulation Conference, IEEE. pp. 2760–2771.

O'Hagan, A., Kingman, J.F.C., 1978. Curve Fitting and Optimal Design for Prediction. Journal of the Royal Statistical Society B 40, 1–42.

Onwunalu, J., Durlofsky, L., 2010. Application of a Particle Swarm Optimization Algorithm for Determining Optimum Well Location and Type. Comput. Geosci. 14, 183–198.

Petelin, D., Filipič, B., Kocijan, J., 2011. Optimization of Gaussian Process Models with Evolutionary Algorithms. Springer Berlin Heidelberg. volume 6593 of *Lecture Notes in Computer Science*. pp. 420–429.

Ranjan, P., Haynes, R., Karsten, R., 2011. A Computationally Stable Approach to Gaussian Process Interpolation of Deterministic Computer Simulation Data. Technomectrics 52, 366–378.

Rasmussen, C.E., Williams, C.K.I., 2006. Gaussian Processes for Machine Learning.

Sacks, J., Welch, W., Mitchell, T., Wynn, H., 1989. Design and Analysis of Computer Experiments. Statistical Science 4, 409–435.

Shanno, D., 1970. Conditioning of Quasi-Newton Methods for Function Minimization. Mathematics of Computation 24, 647–656.

Short, B., Higdon, D.M., Kronberg, P.P., 2007. Estimation of Faraday Rotation Measures of the Near Galactic Sky Using Gaussian Process Models. International Society for Bayesian Analysis 2, 665–680.

SINTEF Applied Mathematics, 2012. Matlab Reservoir Simulator Toolbox v. 2012a. `http://www.sintef.no/Projectweb/MRST/`.

Wahba, G., 1978. Improper Priors, Spline Smoothing and the Problem of Guarding Against Model Errors in Regression. Journal of the Royal Statistical Society B 40, 364–372.

Yeten, B., Durlofsky, L., Aziz, K., 2003. Optimization of Nonconventional Well Type, Location and Trajectory. SPE Journal 8, 200–210.

Appendix A. Test Functions

Test Function (d)	**Formula y=f(x)**
Hump (1)	$y = 1.0316285 + 4x^2 - 2.1x^4 + \frac{1}{3}x^6$
Goldstein-Price (2)	$y =$ $[1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)]\times$ $[30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$
Schwefel (5)	$y = 2094.9 - \sum_{i=1}^{5} x_i \sin\left(\sqrt{\lvert x_i \rvert}\right)$
Hartmann (6)	$y = -\sum_{i=1}^{6} \alpha \cdot \exp[-\sum_{j=1}^{6} B_{ij}(x_j - Q_{ij})^2]$ $\alpha = [1, 1.2, 3, 3.2], \quad B = \begin{bmatrix} 10 & 3 & 17 & 3.5 & 1.7 & 8 \\ 0.02 & 10 & 17 & 0.1 & 8. & 14 \\ 3 & 3.5 & 1.7 & 10 & 17 & 8 \\ 17 & 8 & 0.05 & 10 & 0.1 & 14 \end{bmatrix},$ $Q = \begin{bmatrix} 0.1312 & 0.1696 & 0.5569 & 0.0124 & 0.8283 & 0.588 \\ 0.2329 & 0.4135 & 0.8307 & 0.3736 & 0.1004 & 0.9991 \\ 0.2348 & 0.1451 & 0.3522 & 0.2883 & 0.3047 & 0.6650 \\ 0.4047 & 0.8828 & 0.8732 & 0.5743 & 0.1091 & 0.0381 \end{bmatrix}$
Rastrigin (10)	$y = 10n + \sum_{i=1}^{10}(x_i^2 - 10\cos(2\pi x_i))$
Rosenbrock (10)	$y = \sum_{i=1}^{9}[100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2]$
Perm (12)	$y = \sum_{i=1}^{12}\left[\sum_{j=1}^{12}\left[(j^i + 0.5)\left(\frac{x_j}{j}\right)^{i-1}\right]^2\right]$

Table A.5: Test functions and corresponding formula used for evaluating the performance of the $\mathcal{L}_\beta$ optimization process in GP modelling.