On the construction of probabilistic Newton-type algorithms

Adrian G. Wills and Thomas B. Schö

Abstract—It has recently been shown that many of the existing quasi-Newton algorithms can be formulated as learning algorithms, capable of learning local models of the cost functions. Importantly, this understanding allows us to safely start assembling probabilistic Newton-type algorithms, applicable in situations where we only have access to noisy observations of the cost function and its derivatives. This is where our interest lies. We make contributions to the use of the non-parametric and probabilistic Gaussian process models in solving these stochastic optimisation problems. Specifically, we present a new algorithm that unites these approximations together with recent probabilistic line search routines to deliver a probabilistic quasi-Newton approach. We also show that the probabilistic optimisation algorithms deliver promising results on challenging nonlinear system identification problems where the very nature of the problem is such that we can only access the cost function and its derivative via noisy observations, since there are no closed-form expressions available.

I. INTRODUCTION

We are in this paper concerned with the unconstrained nonlinear optimisation problem

$$\hat{x} = \arg \min_x f(x),$$

in situations when we only have access to noisy evaluations of the cost function $f(x)$ and its derivatives. In the noise-free case, solving this problem has attracted enormous research attention for many decades which has resulted in many variants of optimisation algorithms. Among these are the much celebrated class of quasi-Newton methods that are still—almost half a century after their inception—the state of the art methods when it comes to numerical solution of the unconstrained optimisation problem (1). This is true across most—if not all—branches of science where optimisation problems of the type (1) needs to be solved.

When $f(x)$ and its derivatives are noisy, then these existing methods suffer from a fundamental problem. Specifically, the algorithms may fail to converge since they rely on knowledge of descent directions and line-search (or related) methods that are not geared towards noisy cost functions and gradient evaluations. Such stochastic optimisation problems are commonly occurring for example in the following situations; 1) If the dataset is very large it is not possible to evaluate the cost function on the entire dataset and instead it is divided into so-called mini-batches. This situation arises quite often in Machine Learning and in particular in deep learning applications. 2) When we employ numerical algorithms to compute the cost function and its derivatives. This occurs for example in nonlinear system identification using the maximum likelihood method when various particle filters are used to compute the intractable cost function and its gradients. We use nonlinear system identification as a case study in this paper.

There has recently been some very relevant and encouraging developments for dealing with these stochastic optimisation problems. More specifically it has been shown that standard quasi-Newton methods like the BFGS method [3], [4], [5], [6], Broyden's method [7] and the DFP formula [8], [9] all can be interpreted as particular instances of Bayesian linear regression or as Gaussian process regression. This line of research has shown that we can reinterpret the quasi-Newton algorithms as learning algorithms that estimate a local quadratic model to the cost function [1], [10]. Most of this recent development has taken place within the relatively new and vibrant direction of research commonly referred to as probabilistic numerics, see probabilistic-numerics.org and [11].

Perhaps most importantly this line of research has opened up for genuinely new probabilistic optimisation algorithms, which is necessary in order to solve the problems we are facing when we only have noisy observations of the cost function and its derivatives. We will in this work explore these ideas and present a new algorithm that combines a new approach to modelling the Hessian matrix together with recent results for probabilistic line search routines. Importantly, we are able to ensure that the model of the Hessian is symmetric by making use of the half-vector operator [12]. Another technical contribution is that we propagate the uncertainty in the Hessian approximation between iterations of the algorithm. As a final contribution we have the application of these new algorithms to the challenging nonlinear system identification problem.

The key construction is provided by the Bayesian non-parametric Gaussian process (GP) [13], [14], which is very briefly introduced in Section II. In that section we also outline the two main directions of development that exist when it comes to modeling optimisation problems using the GP. The first of these two directions is then described and developed further in Section III, while the second approach is detailed in an extended version of this paper in [15]. The resulting algorithms are then profiled on the nontrivial nonlinear system identification application in Section IV. Finally, we state our conclusions and ideas for future direction of this research in Section V.

II. MODELING OPTIMISATION PROBLEMS USING GPs

A. Background on the Gaussian process

The Gaussian process [13], [14] is by now an established model for nonlinear functions. The representation that is
used by the GP in modeling a nonlinear function is non-parametric (meaning that it does not rely on any parametric functional form) and probabilistic (meaning that uncertainty is taken into account throughout the entire model).

The Gaussian process is formally defined as a (potentially infinite) collection of random variables such that any finite subset of it has a joint Gaussian distribution. Let us assume that we want to model some nonlinear function $f(x)$ as a realisation from a Gaussian process. We then assign a prior distribution over the function $f(x)$ given by the GP, which we denote by

$$f(x) \sim GP(\mu(x), k(x, x')),$$  \hspace{1cm} (2)

where $\mu(x)$ is some suitable mean function (for example, a strictly convex function centred on prior knowledge of the parameter values). The covariance function (also referred to as the kernel) $k(x, x')$ represents the correlation between function values based on the two evaluation points $x$ and $x'$.

This prior can then be updated using observations of the function via the standard results on partitioned Gaussian distributions, see e.g. [14] for details.

**B. Two existing directions**

The idea of using the GP for optimisation is rather natural, especially in situations where we only have access to noisy observations of the cost function and its derivatives. The approaches available so far can very broadly be divided into two directions.

The first direction starts by assuming that the Hessian is distributed according to a GP. This Hessian is then updated via (potentially noisy) observations of the Hessian, the gradient and the cost function. The observations of the gradients and the cost function take the form of line integral observations of the GP, which can readily be incorporated. Hennig recently outlined some promising and highly interesting developments along these lines [10], [11]. We follow this direction in Section III.

The second direction instead tries to build a global model of the cost function and possibly also of its derivatives. Here the crucial observation is that the derivative of a GP is another GP [14]. This straightforwardly opens up for the possibility of modelling the cost function and possibly also its derivatives as a joint GP. This global GP model is then updated using the (possibly noisy) observations of the cost function and its gradients. Developments along this line started in the global optimisation literature [16] under names such as Gaussian Process Optimisation (GPO) [17] and Bayesian optimisation [18]. This direction is detailed in an extended version of the current paper in [15].

Interestingly, the algorithms resulting from these two directions are also highly suitable for standard nonlinear deterministic problems. We see potential in new optimisation algorithms being created not only for the stochastic situation, but also for the classic deterministic problem.

**III. NON-PARAMETRIC QUASI-NEWTON METHODS**

**A. A non-standard take on the quasi-Newton methods**

The idea underlying the Newton and quasi-Newton methods is that they learn a local quadratic model $q(x_k, \delta)$ of the cost function $f(x)$ around the current iterate $x_k$

$$q(x_k, \delta) \equiv f(x_k) + g(x_k)^T \delta + \frac{1}{2} \delta^T H(x_k) \delta,$$  \hspace{1cm} (3)

where $\delta = x - x_k$, $g(x_k) = \nabla f(x)|_{x=x_k}$ denotes the gradient and $H(x_k) = \nabla^2 f(x)|_{x=x_k}$ denotes the Hessian. Note that (3) is a second order Taylor expansion of $f(x)$, i.e. $f(x) \approx q(x_k, \delta)$ in a close vicinity around $x_k$. The quasi-Newton methods compute an estimate of the Hessian based on zero and first order information (function values and their gradients). More specifically these methods are designed to represent the cost function according to the following model

$$f_q(x_k + \delta) = f(x_k) + g(x_k)^T \delta + \frac{1}{2} \delta^T B_k \delta,$$  \hspace{1cm} (4)

for some positive definite matrix $B_k$. Note that

$$\nabla_{\delta} f_q(x_k + \delta) = g(x_k) + B_k \delta.$$  \hspace{1cm} (5)

Quasi-Newton methods make a standing assumption that

$$\nabla_{\delta} f_q(x_k + \delta)|_{\delta=x_k-1-x_k} = \nabla_x f(x_{k-1}) = g(x_{k-1}).$$  \hspace{1cm} (6)

Equations (5) and (6) combined result in $g(x_{k-1}) = g(x_k) + B_k(x_{k-1} - x_k)$, so that if we define

$$y_k \triangleq g(x_k) - g(x_{k-1}), \hspace{0.5cm} s_k \triangleq x_k - x_{k-1},$$  \hspace{1cm} (7)

then we obtain the secant condition (quasi-Newton equation),

$$B_k s_k = y_k.$$  \hspace{1cm} (8)

This equation is not enough to define the elements of the Hessian approximation $B_k$, we also know that by construction it has to be symmetric. From a learning point of view this is very helpful since it halves the number of unknown parameters to be estimated, but some extra care will have to be taken to ensure this. That said, the existing quasi-Newton algorithms can all be interpreted as employing some particular form of regularisation on $B_k$, for example, one that minimises changes from a previous Hessian approximation $B_{k-1}$. As such, we can solve the following optimisation problem to find a suitable $B_k$ as the solution to

$$\min_B \|B - B_{k-1}\|_W, \hspace{0.5cm} \text{s.t.} \hspace{0.5cm} B = B^T, \hspace{0.5cm} B s_k = y_k,$$  \hspace{1cm} (9)

where $W$ is a positive definite weighting matrix. It was Henning [1], [2] who recently showed this enlightening unifying interpretation of the quasi-Newton algorithms. As pointed out in [2] this opens up for some flexibility in finding new algorithms which we will continue exploring below.

**B. Integral formulation of the quasi-Newton equation**

In the previous section we formulated the key quasi-Newton equation (8) using derivatives. We can represent the same information using a line integral, which comes about by noting that if we define the line segment $r_k(\tau)$ between the current iterate $x_k$ and the previous iterate $x_{k-1}$ as

$$r_k(\tau) \triangleq x_k + \tau (x_k - x_{k-1}), \hspace{0.5cm} \tau \in [0,1],$$  \hspace{1cm} (10)

then

$$\int_0^1 \frac{\partial}{\partial \tau} \nabla f(r_k(\tau)) d\tau = \nabla f(r_k(1)) - \nabla f(r_k(0)) = \nabla f(x_k) - \nabla f(x_{k-1}) \triangleq y_k.$$  \hspace{1cm} (11)
Now also note that by the chain rule
\[
\frac{\partial}{\partial \tau} \nabla f(r_k(\tau)) = \nabla^2 f(r_k(\tau)) \frac{\partial r_k(\tau)}{\partial \tau} = \nabla^2 f(r_k(\tau))(x_k - x_{k-1}).
\] (12)

Therefore,
\[
\int_0^1 \frac{\partial}{\partial \tau} \nabla f(r_k(\tau))d\tau = \int_0^1 \nabla^2 f(r_k(\tau))(x_k - x_{k-1})d\tau.
\] (13)

That is
\[
y_k = \int_0^1 \nabla^2 f(r_k(\tau))(x_k - x_{k-1})d\tau.
\] (14)

This means that the difference between gradients (i.e. \(y_k\)) can be considered a line integral observation of the Hessian matrix. Therefore, in theory we could update an estimate of the Hessian based on this and other such observations. However, since the Hessian is unknown we do not have any functional form for it. Hennig [10] introduced the idea of using a Gaussian process to represent the true Hessian. This is the approach we will take here as well. There are two key problems in building a Bayesian non-parametric model of the Hessian using a GP. Firstly, we have to be able to make use of the line integral observations (14) when learning the GP. This has been solved and used in other settings before, see e.g. [1], [19]. Secondly, how do we ensure that the resulting GP represents a Hessian, i.e. that its realisations are at least symmetric matrices? We will in the subsequent section develop a solution based on the so-called half-vector operator [12] to ensure that the GP employed in representing the Hessian is symmetric. Importantly, the fact that the gradient observations are potentially noisy does in fact not cause any problems at all, since this fits within the standard Gaussian process regression formulation.

**C. Modelling the Hessian as a GP**

The equivalent integral version of the quasi-Newton equation (8) was in the previous section shown to be
\[
y_k = \int_0^1 B(r_k(\tau))s_k d\tau,
\] (15)

where \(B(\cdot)\) denotes the model of the Hessian and \(s_k = x_k - x_{k-1}\). Note that since \(B(r_k(\tau))s_k\) is a column vector we can straightforwardly apply the vectorisation operator inside the integral in (15) without changing the result,
\[
y_k = \int_0^1 \text{vec}(B(r_k(\tau))s_k) d\tau = \int_0^1 (s_k^T \otimes I) \text{vec}(B(r_k(\tau))) d\tau
\]
\[
= (s_k^T \otimes I) \int_0^1 \text{vec}(B(r_k(\tau))) d\tau,
\] (16)

where \(\otimes\) denotes the Kronecker product. The whole point of this exercise is that we have now isolated the vectorised Hessian estimate \(\text{vec}(B(r_k(\tau)))\) inside the integral. One option would now be to place a GP prior on \(\text{vec}(B(r_k(\tau)))\), but that would not enforce the symmetry requirement we have on the Hessian estimate. We can solve this problem using the half-vectorisation operator\(^1\) \text{vech}(\cdot) [12]. Hence, we now assume that we have a GP prior on the unique elements in the Hessian estimate
\[
\tilde{B}(\tau) = \text{vech}(B(r_k(\tau))),
\] (17)

that is given by
\[
p\left(\tilde{B}(\tau)\right) = \mathcal{GP}(\mu_k(\tau), \kappa_k(\tau, t)).
\] (18)

We can then retrieve the full Hessian estimate using the so-called duplication matrix \(D\), which is a matrix such that
\[
\text{vec}(B(r_k(\tau))) = D \tilde{B}(\tau).
\] (19)

More details and some useful results on the duplication matrix, the associated elimination matrix (vech(A) = \(L \text{vec}(A)\)) and their use are provided by [12]. It is now straightforward to also generalise the measurement (16) by adding some noise
\[
y_k = (s_k^T \otimes I) \int_0^1 D \tilde{B}(\tau)d\tau + e_k, \quad e_k \sim \mathcal{N}(0, R).
\] (20)

It can be shown that the joint GP for \(\tilde{B}(\tau)\) and \(y_k\) is given by
\[
p(\tilde{B}(\tau), y_k) = \mathcal{GP}(m_j, K_j),
\] (21a)

\[
m_j = \begin{pmatrix} \mu_k(\tau) \\ (s_k \otimes I) \int_0^1 \mu_k(\tau)d\tau \end{pmatrix},
K_j = \begin{pmatrix} \kappa_k(\tau, t) & \gamma_k(\tau, t) \\ \gamma_k^T(\tau, t) & \pi_k(\tau, t) \end{pmatrix},
\]

where \(\gamma_k(\tau, t)\) and \(\pi_k(\tau, t)\) are given by
\[
\gamma_k(\tau, t) = \left(\int_0^1 \kappa_k(\tau, t)d\tau\right) D(s_k \otimes I),
\] (22)

and
\[
\pi_k(\tau, t) = (s_k^T \otimes I)D^T \left(\int_0^1 \kappa_k(\tau, t)d\tau dt\right) D(s_k \otimes I) + R.
\]

Employing the standard results for conditioning of partitioned Gaussians we obtain the posterior distribution \(p(\tilde{B}(\tau) | y_k)\) from which we can then retrieve the full Hessian estimate,
\[
p(\tilde{B}(\tau) | y_k) = \mathcal{GP}(m, K),
\] (23a)

where
\[
m = \mu_k(\tau) + \gamma_k(\tau, t)\pi^{-1}(\tau, t)(y_k - (s_k \otimes I) \int_0^1 \mu_k(\tau)d\tau),
\] (23b)

\[
K = \kappa_k(\tau, t) - \gamma_k(\tau, t)\pi^{-1}(\tau, t)\gamma_k^T(\tau, t).
\] (23c)

Finally, the Hessian estimate is according to (19) given by
\[
p(\text{vec}(B(r_k(\tau))) | y_k) = \mathcal{GP}(Dm, DKD^T).
\] (24)

---

1For a symmetric \(n \times n\) matrix \(A\) the vector \(\text{vec}(A)\) contains redundant information. More specifically, we do not need to keep the \(n(n-1)/2\) entries above the main diagonal. The half-vectorisation \(\text{vech}(A)\) of a symmetric matrix \(A\) is obtained by vectorising only the lower triangular part of \(A\).
D. Resulting optimisation algorithm

The above ideas are collected here in the form of an algorithm statement where the main theme is akin to quite standard gradient-based search algorithms. In particular, we compute a search direction based on gradient information and the Hessian approximation, and perform a line search along this direction using the cost function $f(x)$ to regulate a potential decrease in the cost. Importantly, care must be taken when performing a line search in this setting since $f(x)$ is stochastic. Here we employ the recent work in [20] that delivers a line search algorithm that handles noisy function and gradient evaluations and also satisfies Wolfe-like conditions on the calculated step length.

It is important to be specific about the covariance function employed below. Here we have opted to use a multi-variate version of the squared exponential covariance given by

\[ k_h(r, t) = \sigma^2 C_k e^{-\frac{1}{2} r^3(t) V_{r_h}(t)} \]  

(25)

where the matrix $C_k$ describes the covariance effect on each element of $B(\cdot)$, the matrix $V$ acts as an inverse length scale, and $\sigma^2$ scales the entire covariance.

Algorithm 1 GP Hessian Approximation optimisation

Require: An initial estimate $x_1$ and a mean estimate of the Hessian matrix $\mu_k(\cdot) = \text{vec}(B_1)$, and a covariance matrix $C_1$, and a positive integer $k_{\text{max}} > 0$ that determines the maximum number of iterations.

1: Set $k = 1$ and perform the following.
2: while $k < k_{\text{max}}$ do
3: Calculate a descent direction $p_k$ based on the current Hessian approximation $\hat{B}_k$ and gradient $g(x_k)$ (care should be taken to ensure that this is a descent direction since $\hat{B}_k$ is not guaranteed to be positive definite).
4: Calculate a suitable step length $\alpha_k$ along the direction $p_k$ according to [20] and set $x_{k+1} = x_k + \alpha_k p_k$.
5: Set $k \rightarrow k + 1$.
6: Update the Hessian approximation mean $\hat{B}_k = m$ and set the covariance matrix $\tilde{C}_k = K$ according to (23).
7: end while

IV. SYSTEM IDENTIFICATION EXPERIMENTS

As a testing ground for the probabilistic optimisation algorithm developed above we have chosen to study the problem of identifying a nonlinear state-space model of the form

\[ x_{t+1} = f(x_t, \theta) + w_t, \quad y_t = g(x_t, \theta) + e_t, \]  

(26a)

Note that we will in this section switch to the standard notation used within system identification. Here $x_t \in \mathbb{X} \subseteq \mathbb{R}^{n_x}$ and $y_t \in \mathbb{Y} \subseteq \mathbb{R}^{n_y}$ denotes the state and the measurement, respectively. The dynamics and the measurements are modeled by the nonlinear functions $f(\cdot)$ and $g(\cdot)$ parameterised by the unknown parameters $\theta \in \Theta \subseteq \mathbb{R}^{n_\theta}$. Finally, $w_t$ and $e_t$ denotes the process noise and measurement noise, respectively.

More specifically we will study the maximum likelihood formulation of the nonlinear system identification problem, which amounts to finding a point estimate of the unknown parameter $\theta$ in (26a) by solving the following optimisation problem

\[ \hat{\theta} = \arg \max_{\theta \in \Theta} p_\theta(y_{1:N}), \]  

(27)

where $y_{1:N} = \{y_1, \ldots, y_N\}$. The likelihood function $L(\theta) = p_\theta(y_{1:N})$ is not available in closed form, however using sequential Monte Carlo methods (a.k.a. particle filters) [21], [22] we can compute unbiased estimates of the likelihood, by solving the following integral

\[ L(\theta) = p_\theta(y_{1:N}) = \int p_\theta(y_{1:N}, x_{1:N}) dx_{1:N}, \]  

(28)

where the accuracy depends on the computational power we have available. For recent overviews and links into the rapidly expanding literature on the use of particle filters for nonlinear system identification we refer to [23], [24]. The idea of using a global GP model for the cost function in the nonlinear system identification problem has previously been explored in [25], but only using noisy observations of the likelihood, not its gradients.

We will in Section IV-A show the performance on a simple and controlled example where we can compute true cost function and the true optimal solution using alternative methods. This is to instill confidence in that the method does indeed perform as we expect it to do on a simple example. In Section IV-B we will then study a significantly harder nonlinear example.

A. Simple linear example

In order to gain some confidence in the probabilistic optimisation methods, here we present the results of applying Algorithm 1 and Algorithm 2 from [15] to a standard linear state-space model identification problem. Specifically, we are interested in estimating the parameters $\theta = \{a, c, q, r\}$ for the following system

\[ x_{t+1} = ax_t + w_t, \quad w_t \sim N(0, q), \]  

(29a)

\[ y_t = cx_t + e_t, \quad e_t \sim N(0, r). \]  

(29b)

The true values for the system are $a^* = 0.9, c^* = 1.0, q^* = 0.1$ and $r^* = 0.5$. The initial state is given by $x_1 \sim N(0, 1)$.

For a given set of measurements $y_{1:N}$ it is possible to calculate the likelihood $L(\theta)$ and its gradient $\nabla_{\theta} L(\theta)$ via standard Kalman filter equations and then employ standard gradient-based search algorithms to obtain $\hat{\theta}$ that maximises the likelihood. In this regard, the problem does not suffer from noisy likelihood and gradient calculations, which serves the purpose of profiling Algorithm 1 in the noise-free case.

To that end, we generated a Monte–Carlo simulation with 100 runs, where each run involves the generation of a new dataset $y_{1:N}$ according to system (29). Furthermore, the initial parameter vector $\theta_0$ was selected at random via moving each element within a range of 50% of the true value. A standard gradient-based search algorithm and Algorithm 1 were provided with the same initial conditions and dataset for each Monte–Carlo run.

For this simulation study, the GP hyperparameters used in Algorithm 1 were chosen as $B_1 = 100I$, $C_1 = I$, $V = 10^{-3}I$ and $\sigma^2 = 1$.

The top–left plot in Figure 1 shows the results for the Monte–Carlo runs. Perhaps not surprisingly, all algorithms produced identical transfer function estimates for the noise-free case, so we have shown only one plot.

Based on these positive results, we conducted a further Monte–Carlo simulation, again comprising 100 runs, where
noise was deliberately added to both the likelihood and gradient. Specifically,
\[
\tilde{L}(\theta) = L(\theta) + v_c, \quad \quad v_c \sim (0, 10^4), \quad (30a)
\]
\[
\nabla_{\theta} L(\theta) = \nabla_{\theta} L(\theta) + v_g, \quad \quad v_g \sim (0, 25I). \quad (30b)
\]
Again, each run involved the generation of a new dataset and this time the initial parameters were chosen as \(\theta_0 = \{a^*/10, c^*/10, q^*/10, r^*/10\}\), in order to ensure that the results were not just a function of randomly chosen initial parameters. Again, we ran both a standard gradient-based search algorithm and Algorithm 1 for each run.

The right-hand column of plots in Figure 1 shows the Bode responses for each estimated system. As possibly expected, the standard gradient-search algorithm often fails to converge due to the presence of noisy cost and gradient evaluations hence resulting in a large variation of estimated transfer functions. In many cases it is impossible to know if the search direction is actually a descent direction, and at the same time a line-search algorithm often fails to find a suitable scaling parameter since it is based on noisy function evaluations.

Contrasting this, Algorithm 1 (and Algorithm 2 from [15]) appear to generate estimates that have a similar distribution to the noise-free case. It is difficult to discern which of these two algorithms that performs best.

While we recognise that it is dangerous to draw definitive conclusions from this limited study, it is nevertheless very encouraging results.

\[
\begin{align*}
\min \quad & 0.5x_t + b + \frac{x_t}{1 + x_t^2} + 8 \cos(1.2t) + qw_t, \\
\text{s.t.} \quad & y_t = 0.05x_t^2 + \epsilon_t,
\end{align*}
\]
and the true parameters are \(b^* = 25\) and \(q^* = 0.1^{1/2}\). This example has previously been investigated by the current authors [26] and is profiled again here due to it being acknowledged as a challenging problem [27], [28].

Algorithm 1 and Algorithm 2 from [15] were employed to estimate \(b\) and \(q\) based on 100 Monte–Carlo runs using \(N = 100\) data points for each run as generated by (31) with the true parameter values. The initial parameter values were chosen randomly in each simulation where the value was chosen uniformly within a 50% range of the true value. The algorithms were allowed to iterate for no more than 100 iterations.

In this case, Algorithm 1 was employed with exactly the same hyperparameter choices as for the linear example. The specifics for Algorithm 2 from [15] are detailed in that paper.

The likelihood and its gradient cannot be calculated exactly in this case and we therefore employed sequential Monte Carlo methods and Fisher’s identity [29], [30] to provide noisy estimates of both. The number of particles used to calculate these terms was 500 in all cases. Note that each simulation required no more than 8 seconds of computation time on a MacBook Pro 2.8GHz Intel i7.

The results of this Monte–Carlo simulation can be observed in Figure 2. For Algorithm 2 from [15] we have removed 19 of the 100 simulation results due to convergence to a local minima, which resulted in a final parameter value that was greater than 5% in error relative true parameter value. For Algorithm 1 we removed only 1 simulation result due to the same criteria.

Again, it is dangerous to draw too many conclusions from these results. At the same time, the performance of Algorithm 1 appears to be slightly better than Algorithm 2 from [15], which may be related to the choice of hyperparameters for the latter method.

V. DISCUSSION

Minimising a nonlinear cost function \(f(x)\) is a challenging problem in general, and as verified again here, is made even more difficult if the cost function and its derivatives cannot be evaluated without unknown errors. These latter types of stochastic problems have been considered for some time now, and yet very recent results in [31] show that this is still an active area of research. The main thrust of current activities is to capture the curvature information available from noisy gradient measurements. In the current paper, we have developed a new approach and reviewed an existing approach for capturing this curvature information that both rely on Bayesian non-parametric estimates of the unknown functions. The first treats the Hessian matrix as an unknown function and employs integral observations of the gradient vector in order to form the curvature estimate. The second
Computational load by employing GP approximations. Stochastic cost functions, and the problem of reducing the computational load by employing GP approximations. Suitable stopping criteria for problems involving spurious large errors in the function or gradient. The question of adaptively tuning the hyperparameters as these algorithms progress also deserves more attention. Other areas to explore include suitable stopping criteria for problems involving stochastic cost functions, and the problem of reducing the computational load by employing GP approximations.

Both approaches appear to have merit and we believe that these approaches deserve further attention. Specifically, to the best of our knowledge, the choice of covariance functions and the corresponding selection of hyperparameters has not been explored in a rigorous manner. For example, Student-t processes [32] might be a natural way to reduce the effect of spurious large errors in the function or gradient. The question of adaptively tuning the hyperparameters as these algorithms progress also deserves more attention. Other areas to explore include suitable stopping criteria for problems involving stochastic cost functions, and the problem of reducing the computational load by employing GP approximations.

Fig. 2: Iterations of parameter values using the GP Hessian approximation in Algorithm 1 (left column) and the global GP of Algorithm 2 from [15] (right column) with estimates in red and true value shown as solid blue.