Research paper

Struct Multidisc Optim 27, 371–383 (2004)
DOI 10.1007/s00158-004-0397-9

A parallel updating scheme for approximating and optimizing high fidelity computer simulations[*]

A. Sóbester, S.J. Leary and A.J. Keane

Abstract Approximation methods are often used to construct surrogate models, which can replace expensive computer simulations for the purposes of optimization. One of the most important aspects of such optimization techniques is the choice of model updating strategy. In this paper we employ parallel updates by searching an expected improvement surface generated from a radial basis function model. We look at optimization based on standard and gradient-enhanced models. Given N_p processors, the best N_p local maxima of the expected improvement surface are highlighted and further runs are performed on these designs. To test these ideas, simple analytic functions and a finite element model of a simple structure are analysed and various approaches compared.

Key words gradient-enhanced approximations, parallel optimization, radial basis functions

1 Introduction

The problem of optimization using high fidelity computer simulations is common to many engineering design problems. These simulations are based on mathematical models of some system of interest. Examples include finite element (FE) analysis for structural engineering problems or Navier-Stokes models in computational fluid dynamics (CFD). Optimization is a highly repetitive process requiring many analyses of the model under consideration. If this model itself is computationally expensive, direct optimization algorithms can rarely be employed, as a highly repetitive analysis of a high fidelity model becomes too time consuming to be practical.

To overcome this problem cheap approximating models (often termed "surrogate models") are sought. These are based on a limited number of calls to the high fidelity model. Once constructed, the surrogate model can replace the original high fidelity model for the purposes of optimization.

The first strategic decision that needs to be made relates to the design of experiments (DoE) to be used to provide training data for the approximation method. There are many such methods available to the designer – we will briefly review some of them in the next section. Their common feature is that they try to fill the design space in some sense, as it is commonly recognized that in the absence of any *a priori* knowledge on the problem under consideration, uniformity of the design points throughout the region of interest is favourable.

Regarding the approximation methods, these can be applied locally or globally. Local approximations, such as polynomial response surfaces (see, e.g. Myers and Montgomery 1995), are defined over a specific region of interest, namely about the current best design. Optimization proceeds using a move limit or trust region strategy: we optimize only over the region where the model is valid. Successive local approximations are used to guide the search to a stationary point. Convergence is guaranteed, although only to a local optimum.

Global approximations on the other hand try to capture the behaviour of the overall design space. It may be possible to use polynomial response surfaces, but only if the underlying global response is of low modality. Typically, more global approximators are required. Examples include artificial neural networks (White *et al.* 1992), radial basis functions (Powell 1987) and kriging (Sacks *et al.* 1989). In this paper we are particularly concerned with global approximations.

Traditionally, the information available to construct the approximation is in terms of the response of interest only. Nowadays, however, gradient information (i.e. derivatives of the response with respect to the independent variables or inputs) is often available at little added

Received: 13 November 2002
Revised manuscript received: 14 October 2003
Published online: 29 June 2004
© Springer-Verlag 2004

A. Sóbester[✉], S.J. Leary and A.J. Keane

C.E.D.G., School of Engineering Sciences, University of Southampton, Southampton, SO17 1BJ, U.K.
e-mail: A.Sobester@soton.ac.uk

[*] Revised version of the paper presented at the Third ISSMO/AIAA Internet Conference on Approximations in Optimization October 14–25, 2002

cost. For example, a perturbation analysis of a finite element model may yield these derivatives at very little cost; in an adjoint CFD model, all the components of the objective function gradient are usually available at a cost of around one function evaluation. This information can be included in many approximation models. For example Chung and Alonso (2001) and van Keulen and Vervenne (2002) have considered gradient-enhanced polynomial response surfaces and Morris *et al.* (1993), Chung and Alonso (2002) and Leary *et al.* (to appear) use gradient-enhanced kriging models. In this paper we concentrate on the radial basis function approach.

Once a particular approximation has been built, some updating strategy needs to be defined. The simplest is to optimize the approximation, re-evaluate the high fidelity model at the predicted optimum and repeat. This procedure may stall early due to, for instance, the presence of local minima in the underlying function, depending on the global accuracy of the initial surrogate model. If the goal is to improve the quality of the prediction, then a search over prediction error (if available) may be considered. Such an approach can be found in Martin and Simpson (2002). Given both a prediction and some estimate of the error in the prediction, elegant global optimization strategies can be devised. An example of this is the notion of global search based on expected improvement (see, e.g. Jones *et al.* 1998), which balances the need for a surrogate objective value together with the uncertainty of the model. This idea can be amended to search more globally, as in Sasena *et al.* (2002). Another method would be to weight the objective and uncertainty components of the expected improvement function in order to drive the search more towards optimization or approximation, depending upon the designer's goals. An expected improvement function for gradient-enhanced kriging models is presented by Leary *et al.* (to appear).

Considering that in today's industrial setting it is commonplace for parallel computing architectures to be available to the design engineer, we advocate the use of parallel computing throughout the design process based on the methods described above. We will assume N_p processors are available; first these will be used to run our design of experiments in parallel. This information will then be used to construct a surrogate model. Both traditional and gradient-enhanced models will be considered. We will compare the approaches assuming gradients are free and alternatively that they are available at the cost of one function evaluation (which we sometimes term the "adjoint cost") as both these situations can arise in practice. We note that the use of gradient-enhanced models typically depends on the cost of the gradient: it does not make sense to use gradient enhanced models if the gradients are evaluated using finite differencing.

The updates to the model are also performed in parallel: we can either search values of the surrogate objective in order to attempt to quickly optimize the model, search areas of high error in order to globally reduce the model's error, or search some expected improvement or weighted expected improvement model. In all these cases either a local optimizer with multiple restarts or a genetic algorithm with clustering and sharing could be considered. The idea is either to locate the N_p best optima of the response surface, the N_p designs with the highest prediction errors or the N_p designs with the highest expected improvements.

The latter two goals tend to be extremely multimodal. However, if N_p unique minima are not available, we could add some extra points using either of the other two criteria. In this paper we consider an expected improvement approach alone as in all the examples tested we always found more local maxima of the expected improvement surface than available processors. We mention the others for completeness.

The rest of this paper is set out as follows. Section 2 considers design of experiments and approximation methods; in particular radial basis function approximation methods will be described. Section 3 introduces our parallel updating scheme for surrogate optimization. In Sect. 4 we show some results from our approach and in the final section conclusions are drawn and areas of further research highlighted.

2 DoE and global model building

Before we build an approximation we require a systematic means of selecting the set of inputs (called a design of experiments, or DoE) at which to perform a computational analysis. In k dimensions the 2^k vertices formed by the upper and lower bounds of the design space usually provide the bounding box within which the experimental design is created. The idea with DoE is to, in some sense, evenly fill this design space with a limited number of points. As a result many of the algorithms employed are referred to as "space filling" designs.[1]

Simple experimental designs include 2^k full factorial designs which are created by specifying each design variable at two levels, the upper and lower bounds on each variable (this design considers every vertex of the design bounding box). 3^k full factorial designs additionally include the midpoint of each input. These experimental designs prove expensive for large k so they are often replaced with fractional factorial designs. Several other types of designs can be encountered in the polynomial response surface literature – the interested reader may wish to consult, e.g. Myers and Montgomery (1995) for further details.

A popular choice for generating an experimental design for deterministic computer experiments is the *Latin*

[1] Note that if we know that the optimum we seek is likely to be defined by constraints then we may wish to bias the DoE to sample the constraint boundaries more thoroughly than regions away from these boundaries. Such ideas are not considered further here, however.

hypercube (Mackay *et al.* 1979). This has the advantage that each of the variables has all portions of its range represented equally. The downside of Latin hypercube designs is that they are not guaranteed to have good space-filling properties. A possible solution is to use *optimal Latin hypercube* designs. These are Latin hypercubes that achieve optimality in some space filling sense – for example, a maximin distance criterion (Morris and Mitchell 1995). In this paper our experimental designs are formed from optimal Latin hypercube designs using this criterion. An algorithm for generating such designs can be found in Morris and Mitchell (1995). Examples of a standard and a Morris-optimal Latin hypercube using 16 points with two independent variables are shown in Fig. 1. The optimal Latin hypercube DoE is used here to construct *training* data, upon which the surrogate models are built.

In this work, as we are dealing with relatively simple problems only, we can afford to create a set of *testing* data, which is used to assess the accuracy of our approximation methods – the results are presented in Tables 1 and 2, Sect. 4.2 (of course, in general, this would not be practical for expensive computer simulations). The testing data is created using LP$_\tau$ sequences (Sobol 1979), another space-filling DoE formulation.

Once we have chosen a suitable DoE and evaluated the high fidelity model at this set of inputs, we can construct an approximation. In a typical approximation model the relationship between observations (responses) and independent variables on a k-dimensional domain D is expressed as

$$y = f(\mathbf{x}) \tag{1}$$

where y is the observed response, $\mathbf{x}$ is a vector of k independent variables

$$\mathbf{x} = (x_1, x_2, \dots, x_k) \tag{2}$$

and $f(\mathbf{x})$ is some unknown function. An approximation to this response

$$\hat{y} = \hat{f}(\mathbf{x}) \tag{3}$$

is sought. As it will become apparent in the next section, it is also important from the optimization point of view to be able to obtain an error estimate for this approximation. To this end, here we employ a stochastic process-based modelling framework. This may seem counterintuitive, as physics-based numerical computer experiments, the pillars of the majority of computer-aided design optimization procedures, are usually deterministic. That is, unlike physical experiments, repeated runs of such simulations on the same design return the same figure of merit (objective value) each time. Nevertheless, it can be argued that stochastic process approximation techniques can be employed to model this type of output. The rationale is that, although the physics-based simulation process itself is deterministic, its output can be viewed as *a realization of a stochastic process*[2].

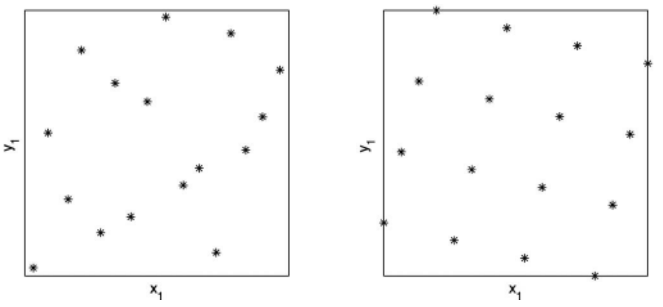

Fig. 1 A Latin hypercube and an optimal Latin hypercube design

There are several approaches for building such models – here we choose to work with radial basis functions (RBF) on the grounds that their training is inexpensive, yet, as we will see, they are sufficiently accurate for optimization purposes. RBF models attempt to express a complicated landscape as the weighted sum of several simple functions – in the following we describe the model building procedure in more detail.

Assuming that we can afford to run the analysis code N times, we sample the objective for N designs (denoted by $[\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(N)}]$), at which we obtain the responses $\mathbf{y} = [y^{(1)}, y^{(2)}, \dots, y^{(N)}]$. Radial basis functions can be used to make a prediction $\hat{y} = \hat{f}(\mathbf{x})$ at any point $\mathbf{x}$ in the design space and the first step towards this is to choose the basis function centers. To obtain an interpolating model we need at least N bases. The common choice here is the set of N points where we know the objective function values (i.e. $[\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(N)}]$). The basis functions take the form $\phi(\|\mathbf{x} - \mathbf{x}^{(i)}\|)$, where $\phi(\cdot)$ is some (usually) nonlinear function, the i^{th} such function depending on the Euclidean distance between $\mathbf{x}$ and $\mathbf{x}^{(i)}$. As we mentioned earlier, the predictor is a linear combination of these basis functions, that is,

$$\hat{y} = \hat{f}(\mathbf{x}) = \sum_{i=1}^{N} w_i \phi\left(\left\|\mathbf{x} - \mathbf{x}^{(i)}\right\|\right) . \tag{4}$$

The coefficients w_i have to be found such that the predictor interpolates the data. To do this, we are required to satisfy for $j = 1, \dots, N$:

$$\hat{f}\left(\mathbf{x}^{(j)}\right) = \sum_{i=1}^{N} w_i \phi\left(\left\|\mathbf{x}^{(j)} - \mathbf{x}^{(i)}\right\|\right) = y^{(j)} . \tag{5}$$

[2] We note here that there is some debate concerning the validity of this fiction and statisticians are sometimes reluctant to interpret predictors and error measures derived from it as more than practically useful figures, suggesting that no pretence should be made about the rigorousness of their mathematical foundation. The idea is nonetheless a powerful one, as it suggests plausible ways of constructing useful models of deterministic outputs (Trosset and Torczon 1997) and experience shows that the predictions obtained with them are adequate for practical purposes.

Defining the coefficient vector $\mathbf{w} = [w_1, w_2, \ldots, w_N]^T$ and the matrix $\mathbf{\Phi}_{i,j} = \phi(\|\mathbf{x}^{(i)} - \mathbf{x}^{(j)}\|)$, where $i = 1, \ldots, N$ and $j = 1, \ldots, N$, this can be written as $\mathbf{\Phi w} = \mathbf{y}^T$. Then, provided the inverse of $\mathbf{\Phi}$ exists, the coefficients can be determined by computing $\mathbf{w} = \mathbf{\Phi}^{-1}\mathbf{y}^T$ and a prediction $\hat{y}$ can be made in any point $\mathbf{x}^{(N+1)} \in D$ by computing:

$$\hat{y}^{(N+1)} = \boldsymbol{\phi}\mathbf{w} = \boldsymbol{\phi}\mathbf{\Phi}^{-1}\mathbf{y}^T \tag{6}$$

where

$$\boldsymbol{\phi} = \left[\phi\left(\left\|\mathbf{x}^{(N+1)} - \mathbf{x}^{(1)}\right\|\right), \phi\left(\left\|\mathbf{x}^{(N+1)} - \mathbf{x}^{(2)}\right\|\right), \ldots \right.$$
$$\left. \ldots, \phi\left(\left\|\mathbf{x}^{(N+1)} - \mathbf{x}^{(N)}\right\|\right)\right]. \tag{7}$$

Many different basis functions $\phi(\cdot)$ could be considered. Throughout this work we have used exponentially decaying Gaussian basis functions

$$\phi(r) = \exp\left(-r^2/2\sigma^2\right) \tag{8}$$

as these are twice differentiable and facilitate the derivation of an expected improvement measure (we will return later to the reasons why these two features are important). The choice of the hyperparameter σ, which governs the regions of influence of each kernel, is important and can affect prediction accuracy. In the work presented in this paper we use a *leave-one-out cross validation* procedure, searching for the optimum σ over the domain $[10^{-2}, 10^1]$. This means that for each value of σ we build N RBF models leaving out one of the training points in each case (as though we only had $N-1$ points), we compute the difference between the true objective value of the currently left out point and the objective predicted by the partial model (which uses the remaining $N-1$ points) in the same point. The final model is constructed using the σ that minimizes the sum of the squares of these residuals. Of course, the larger the number of training points, the more reliable the training will be (i.e. it is more difficult to find the best σ for, say, $N = 5$ by building the five four-point models, than for $N = 50$, when the predictions can be based on 49-point leave-one-out models). We consider 15 values of σ logarithmically spread over the range indicated above. More thorough searches (i.e. a full optimization of the hyperparameter σ) could be considered, but this would increase the computational cost of the training and in the authors' experience the gain in model accuracy thus achieved is not significant. We also note here that the optimum σ is related to the distances between the kernels – the range $[10^{-2}, 10^1]$ is suitable for the case when the problem domain is normalized to $D = [0,1]^k$ (as in the experiments described here).

Clearly, more accurate models could be considered; for example, we could allow the selection of a different σ for each independent variable (as is often done, for example, in kriging). However, the computational cost of model training then becomes an issue (particularly when we move to gradient-enhanced predictions where considerably larger matrices need to be inverted).

Supposing that the sensitivities of the objective function f with respect to the design variables x_i are cheaply available, this information can be used to enhance the radial basis function model. In this case we are seeking an *osculating interpolant* of the set of known objective value points, i.e. a model determined not only by a set of nodal points (the objective function values), but also by the required slopes in those points (the objective function gradients). Assuming we now define an $N(k+1)$ vector of responses as

$$\mathbf{y} = \left(y^{(1)}, \nabla y^{(1)}, y^{(2)}, \nabla y^{(2)}, \ldots, y^{(N)}, \nabla y^{(N)}\right)^T, \tag{9}$$

where

$$\nabla y^{(i)} = \left(\frac{\partial f}{\partial x_1}(\mathbf{x}_i), \frac{\partial f}{\partial x_2}(\mathbf{x}_i), \ldots, \frac{\partial f}{\partial x_k}(\mathbf{x}_i)\right), \tag{10}$$

then we can extend our matrix Φ to include the first and second derivatives of ϕ in much the same way as is done in kriging (see, e.g. Morris *et al.* 1993). This assumes our basis function is at least twice differentiable, another reason for using Gaussian basis functions. A gradient-enhanced radial basis function (GERBF) approximation can then be constructed.

3 Optimization

As we have already mentioned, a common approach to reducing the cost of a global optimization procedure is to make use of cheaper surrogate models, such as the stochastic process model-based global RBF approximation described in the previous section.

The cornerstone of any optimization strategy based on such low-cost surrogates is the choice of the updating method, i.e. given an initial global model how do we select the next site(s) where the expensive objective will be sampled? Perhaps the most obvious strategy is to re-sample in areas that appear promising in terms of the objective function value $\hat{y}$ predicted by the surrogate model. The success of this approach depends on the quality of the initial approximation. If the initial approximation is accurate, it is likely to lead the designer quickly to the global minimum or at least to a very good solution.

A second approach may be to search areas of high estimated approximation error, i.e. in our case, to choose the design that maximizes the estimated error of the RBF predictor. In order to calculate this, we make use of the fiction discussed earlier, namely that each deterministic response $y(\mathbf{x})$ is in fact the realization of some stochastic process $Y(\mathbf{x})$ (taken here to be a Gaussian random variable). Using the (Gaussian) distributions of the N responses $\mathbf{y} = [y^{(1)}, y^{(2)}, \ldots, y^{(N)}]$ collected so far, it can be shown that the mean and the variance of the assumed stochastic process at $\mathbf{x}^{(N+1)}$ are:

$$\hat{y}^{(N+1)} = \boldsymbol{\phi}\Phi^{-1}\mathbf{y}^T \tag{11}$$

$$\sigma^2_{\hat{y}^{(N+1)}} = 1 - \boldsymbol{\phi}\boldsymbol{\Phi}^{-1}\boldsymbol{\phi}^T \tag{12}$$

respectively (for a detailed demonstration see, e.g. Gibbs 1997). As expected, the mean of the assumed Gaussian distribution that we drew $\hat{y}^{(N+1)}$ from (11) is, in fact, the RBF predictor obtained earlier (6). We will use the variance of this Gaussian distribution (12) as a measure of the likely prediction error at untested sites.

Placing the new design to be evaluated at the global maximum of (12) may not make much sense if the goal is to locate a good design quickly, but it does allow global improvement of the approximating model. One might even consider a hybrid method, where in the early stages we search the errors in order to enhance the accuracy during the global exploration phase and then refine locally. Nevertheless, it is by no means obvious whether this approach would be better than simply starting with a larger design of experiments and then updating the model locally. Ultimately, this would be problem dependent.

From a global optimization perspective, the technique outlined above amounts to *exploration* of the search space, whereas searching the predictor itself (as mentioned earlier in this section) is equivalent to *exploiting* currently known promising basins of attraction. Clearly, we need a point selection criterion that balances these two approaches.

As we mentioned earlier, the stochastic process $Y(\mathbf{x})$ models our uncertainty about the response $y(\mathbf{x})$ in the point $\mathbf{x}$. Denoting the best objective value from the sample evaluated so far by $y_{\min} = \min\left[y^{(1)}, y^{(2)}, \dots, y^{(N)}\right]$, a further quantity can be defined: the *improvement*

$$I(\mathbf{x}) = \max\left[y_{\min} - Y(\mathbf{x}), 0\right] \tag{13}$$

(Jones *et al.* 1998). This is, of course, also a random variable – it models our uncertainty about the amount by which $y(\mathbf{x})$, the objective function value in the next evaluated sample point, will improve on the current best objective.

Given a prediction $\hat{y}$ and an error estimate $s = \sigma_{\hat{y}^{(N+1)}}$ (in a point $\mathbf{x}$, as per equations (11) and (12)), using Gaussian kernels, the expectation of the improvement (or, as it is often termed in the literature, the *expected improvement*) can be calculated (see, e.g. Schonlau 1997) as:

$$E(I) = \mathrm{EIF}(\mathbf{x}) =$$

$$\begin{cases} (y_{\min} - \hat{y})\Psi\left(\dfrac{y_{\min} - \hat{y}}{s}\right) + s\psi\left(\dfrac{y_{\min} - \hat{y}}{s}\right) & \text{if } s > 0 \\ 0 & \text{if } s = 0 \end{cases} \tag{14}$$

where $\Psi(\cdot)$ is the standard normal distribution function and $\psi(\cdot)$ is the standard normal density function.

The first term of (14) is the predicted difference between the current minimum and the prediction $\hat{y}$ in $\mathbf{x}$, penalized by the probability of improvement. Hence it is large where $\hat{y}$ is small (or it is likely to be smaller then $y_{\min}$). The second term is large when the error s is large, i.e. when there is much uncertainty about whether y will be better than $y_{\min}$. Thus, as Schonlau (1997) points out, the expected improvement will tend to be large at a point with predicted value smaller than $y_{\min}$ and/or there is much uncertainty associated with the prediction. Therefore, expected improvement can be considered as a balance between seeking promising areas of the design space (according to our approximation) and the uncertainty in the model. The global search strategy based on it (i.e. evaluation of initial DoE set, followed by updates in the maxima of the expected improvement surface) has the advantage that it is much less likely to stall than a search over the approximation only (although there are certain pathological cases when it does, see, e.g. Jones 2001). The disadvantage is that it usually takes longer to converge, which could be a drawback if the initial model did turn out to be an accurate one.

When we consider constrained optimization, whether the constraints are approximated or evaluated exactly, the expected improvement function needs to be modified as follows:

$$\mathrm{EIF}(\mathbf{x}) =$$

$$\begin{cases} (y_{\min} - \hat{y})\Psi\left(\dfrac{y_{\min} - \hat{y}}{s}\right) + s\psi\left(\dfrac{y_{\min} - \hat{y}}{s}\right) \\ \text{if } s > 0 \text{ and the (approx. or exact) constraints are} \\ \text{satisfied} \\ \\ 0 \\ \text{if } s = 0 \text{ or if the (approx. or exact) constraints are} \\ \text{violated} \end{cases} \tag{15}$$

Furthermore, in this case $y_{\min}$ should be taken as the minimum *feasible* response. To understand this, consider the simple one-variable demonstrative example shown at the top of Fig. 2.

Assuming without loss of generality that both our objective and constraint are expensive (if our constraint was cheap we would just evaluate it directly), the approximation to these responses based on three sampled values (at $x = 0.1$, $x = 0.5$ and $x = 0.9$) is shown in the second subplot of Fig. 2. Supposing that we are required to minimize this objective subject to the constraint remaining below the horizontal dashed line (shown on the top subplot), the minimum will occur at $x = 0.6$. If we take $y_{\min}$ to be the minimum response, as in the unconstrained case (14), it is possible that the expected improvement will be zero everywhere inside the feasible (predicted or exact) region. That is, we do not know where to (feasibly) sample in order to improve our model. This is illustrated in the third section of Fig. 2. If, however, we choose the minimum feasible response, then any predicted response below this value will have non-zero expected improvement. In this way, potentially better solutions can be highlighted. This is demonstrated in the bottom subplot of the figure. In this case our first update is close to the true solution – once this is added to the database of runs and a new approximation is constructed, we quickly home in on this solution.

The optimization strategies outlined above involve reconstructing the surrogate model after each new expensive evaluation – therefore the procedures, in their basic form, are sequential. Nevertheless, any of the three strategies can also be implemented in parallel – here we choose to work with expected improvement.

Given N_p processors, the idea is to locate the N_p best local maxima of the expected improvement surface (which is usually extremely multimodal) using, for instance, either a gradient-based optimizer with multiple restarts or a genetic algorithm (GA) with clustering and sharing. Once these N_p local maxima are identified, those locations are evaluated in parallel using the high fidelity model and the process is repeated until we converge or run out of time. A flowchart of the approach is shown in Fig. 3.

We could always consider a parallel version of the other model update criteria as well by replacing the expected improvement step in the flowchart with either optimizing the approximation or its error. As we have mentioned earlier, if there are not as many optima as there are processors (which is more likely to happen when optimizing the approximation) we could add some points using either of the other two approaches.

Furthermore, we could weight the two terms in the expression of the expected improvement (14). As discussed earlier, the first term relates to the approximation and the second to the error. A suitable weighting could be used to bias the search towards one of these, depending on the designer's goals. Once more, this can be done in parallel and the weighting could vary as the search progressed, biasing it towards error reduction to begin with and then moving towards finding an optimal design at the end.

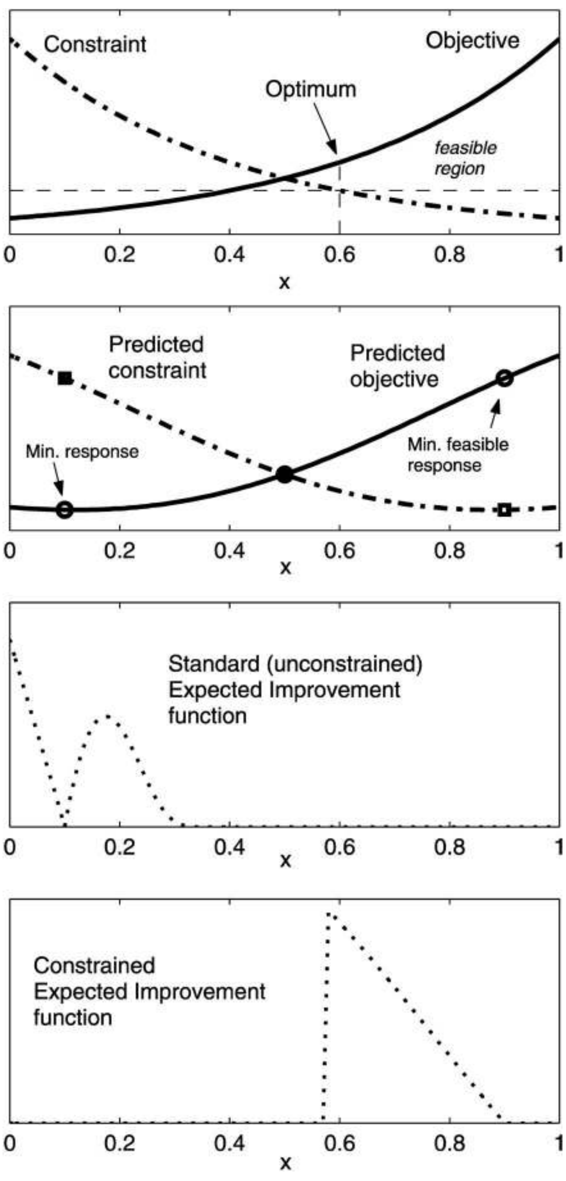

Fig. 2 One-variable demonstrative example of constructing the expected improvement surface in the case of a constrained problem

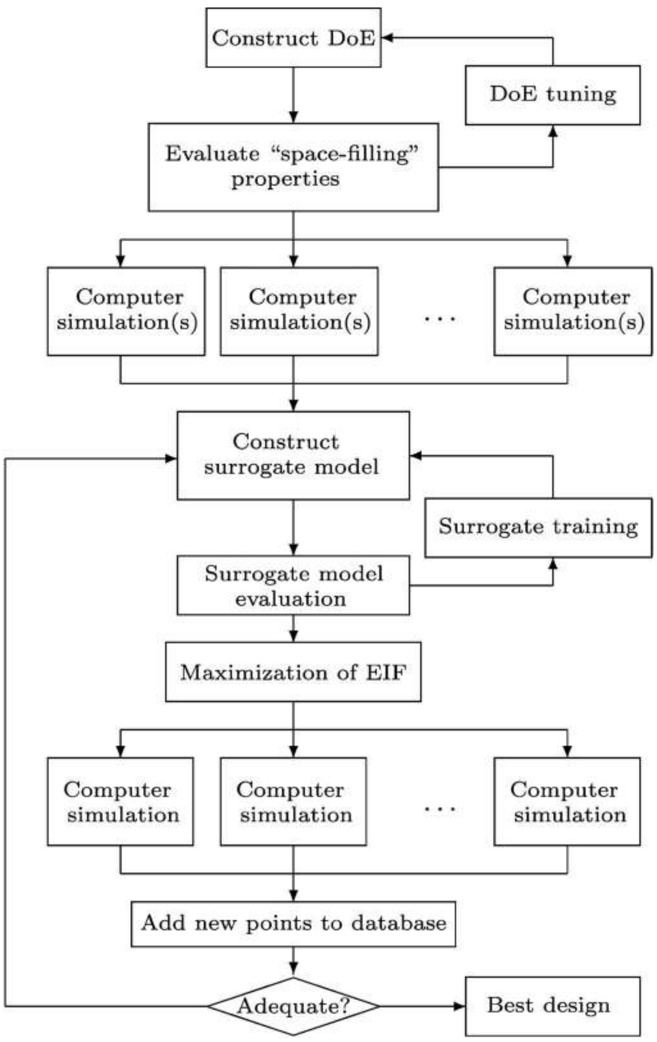

Fig. 3 Flowchart of the parallel RBF/GERBF-based optimization scheme

4
Results

4.1
Demonstrative example

In this section we compare results for the various approaches. To start with we demonstrate our parallel updating scheme on the two-variable Branin function. This function is described in the Appendix. It is defined on the domain $[-5, 10] \times [0, 15]$, however we scale these inputs to $[0, 1]^2$, as highlighted earlier. A contour plot of the Branin function is shown in Fig. 4.

First, using our radial basis function approximation without gradients, assuming four parallel processors, we start with a DoE consisting of eight computer experiments (two parallel runs). This is shown in Fig. 5 (top). We then locate the four best maxima of the expected improvement surface, as shown in Fig. 5 (bottom), and re-evaluate the model at these points. We continue until we have 24 points (six parallel runs) – see Fig. 6. The best minimum at this stage is 0.774.

We now consider a gradient-enhanced approximation, assuming that the derivatives are available for free. Again, we assume an initial DoE consisting of eight points (two parallel runs) and perform parallel updates using EIF maximization to locate the next design points. After 24 evaluations (six parallel runs) the approximation is as shown in Fig. 7 and the best minimum is 0.518.

Finally, we assume gradients are available at the cost of one function evaluation. Therefore, our initial DoE here consists of four points. These four function and gradient evaluations require two parallel runs. Four new points are found by maximizing the expected improvement. The objective function values and the gradients in these points are again evaluated (two parallel runs) and the optimization process is repeated once. The total computational cost is equivalent to those above and the best minimum here is 3.589 (Fig. 8). Whilst this is larger than before, it is still a very good objective value considering the range of values the Branin function takes over the domain.

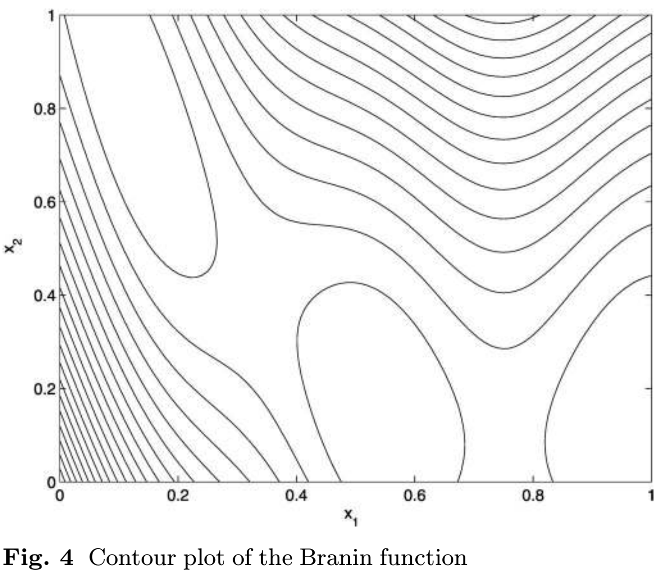

Fig. 4 Contour plot of the Branin function

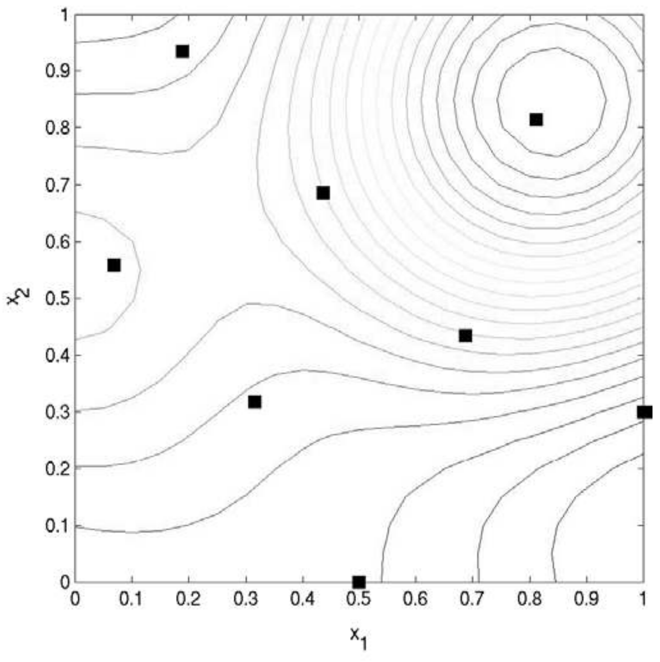
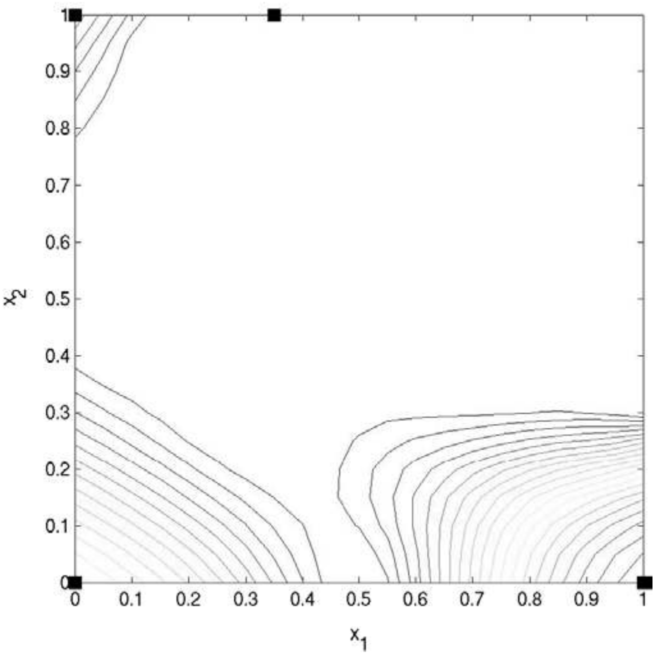

Fig. 5 Initial approximation (top) and first EIF maximization (bottom)

4.2
Results on two five-dimensional test functions

In this section we look at two five design variable test cases, a detailed description of which can be found in the Appendix. The first, the modified Rosenbrock function, is

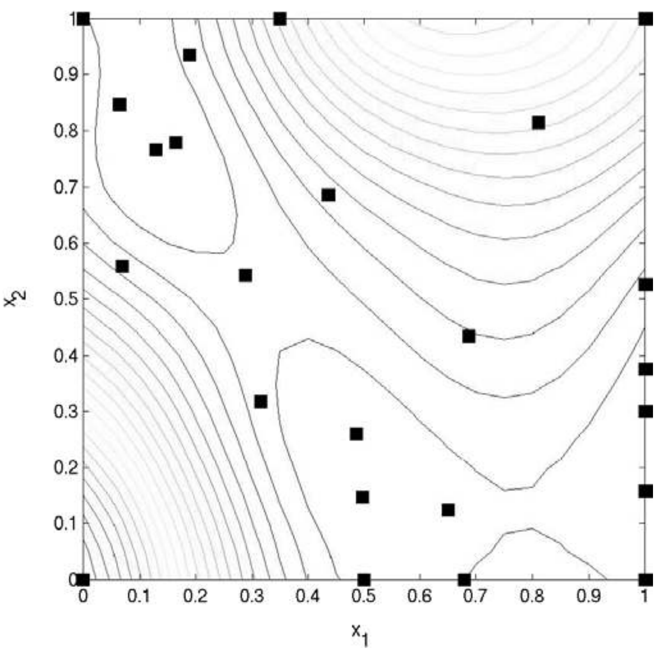

Fig. 6 Approximation to the Branin function after six parallel runs

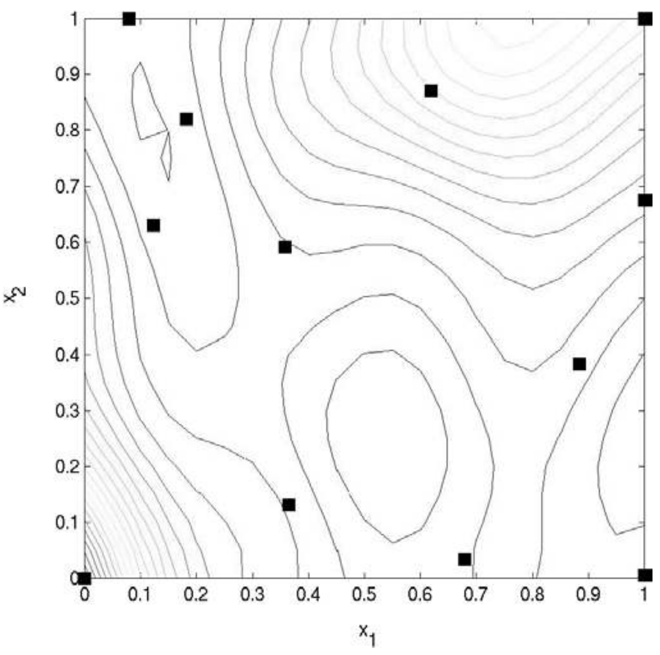

Fig. 8 Approximation to the Branin function assuming "adjoint" gradients after six parallel runs

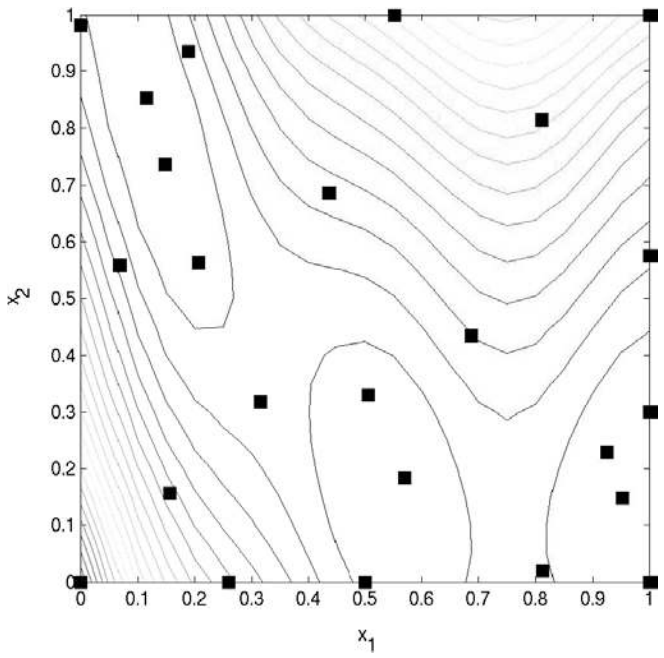

Fig. 7 Approximation to the Branin function assuming free gradients after six parallel runs

defined on the domain $[-1,1]^5$ and the second, the Ackley function, on $[-2.048, 2.048]^5$. As before, both functions are scaled to $[0,1]^5$.

First we compare the accuracy of the approximating models on one of the test cases. We arbitrarily choose to work with the modified Rosenbrock function. Table 1 shows the accuracy of our radial basis function approximation model without gradient information included. The first two columns of the table show the average error and the correlation coefficient ("r^2-value") between the model and the true Rosenbrock function, evaluated at a set of 500 points arranged in an LP_τ design. The table also includes σ, the optimum kernel width hyperparameter found for each case (see (8)). This data is shown for optimal Latin hypercube designs of several different sizes N. Table 2 contains the same information using the gradient-enhanced radial basis function (GERBF) approximation.

From Tables 1 and 2 it can be seen that as N increases our model generally becomes more accurate (the error reduces and the correlation increases), as expected. What is really interesting is to compare the accuracy of the GERBF model with that of the standard RBF model.

Table 1 Accuracy of the approximation without using gradients

N	Average error	Correlation	σ
16	0.7184	0.1412	0.32
32	0.6844	0.2594	0.19
48	0.5925	0.4870	0.19
64	0.5057	0.5540	0.32
80	0.4653	0.6173	0.32

Table 2 Accuracy of the approximation using gradients

N	Average error	Correlation	σ
8	0.6514	0.3625	0.32
16	0.6010	0.4781	0.19
32	0.5536	0.5708	0.19
48	0.3797	0.7355	0.32
64	0.3336	0.7825	0.32
80	0.2004	0.9140	0.32

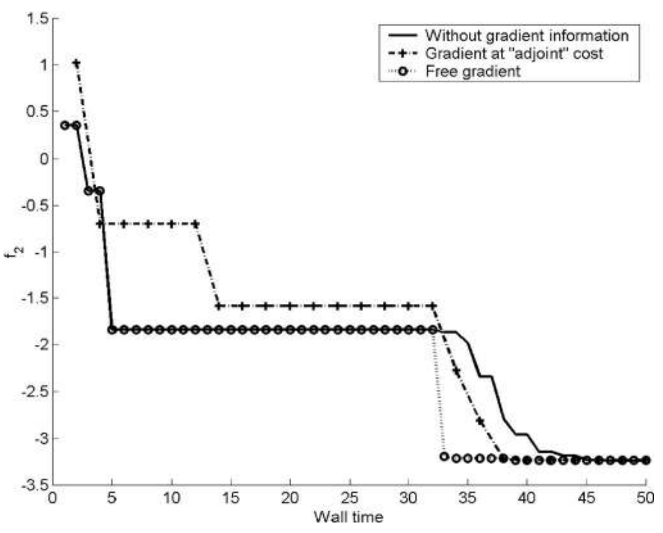

Fig. 9 Comparison of optimization strategies based on RBF and GERBF models for the modified Rosenbrock function, using one processor

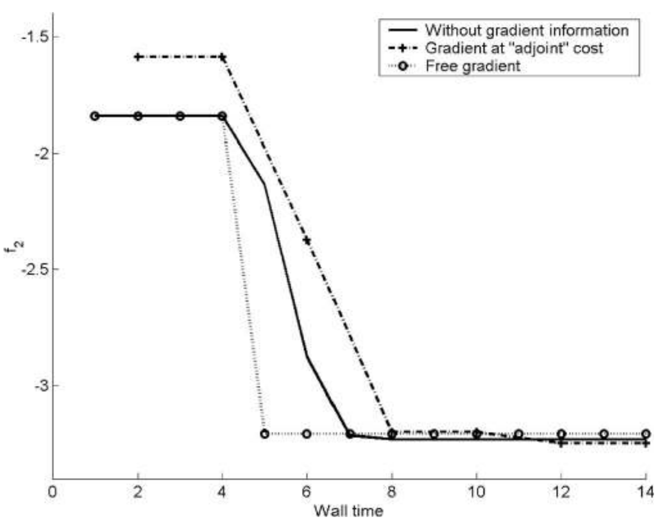

Fig. 11 Comparison of optimization strategies based on RBF and GERBF models for the modified Rosenbrock function, using eight processors

We note that when all the gradients are available at a cost of one function evaluation, utilizing gradient information increases the global accuracy of the model (c.f. Table 1 with $N = 16, 32$ and Table 2 using $N = 8, 16$). We have observed this to be true for other choices of N, k and other objective functions. When gradients are available virtually for free, the GERBF is far superior.

We now consider how the various approaches perform in terms of optimization. First, we compare the RBF, together with the GERBF, using $N_p = 1,\ 4$ and 8 processors on both five design variable objective functions. We look at the performance of the optimization strategy based on these models, assuming gradients are both free and available at a cost of one function evaluation. We choose the size of the DoE so that its computational cost will be the same in each case. We start from a 32-point Latin hypercube design in the case of the standard RBF model and the GERBF model when we assume free gradients and a 16-point Latin hypercube design when the gradients are assumed to have "adjoint" cost. Figures 9, 10 and 11 show the results of optimizing the modified Rosenbrock function, using $Np = 1,\ 4$ and 8 processors respectively. Figures. 12, 13 and 14 show the same information when using the Ackley function.

We observe that when we assume gradients are available at adjoint cost, the advantage of using the GERBF for optimization is less obvious than from the approximation point of view (as demonstrated earlier in Tables 1 and 2). If gradients are available for free then their use in optimization unquestionably makes sense.

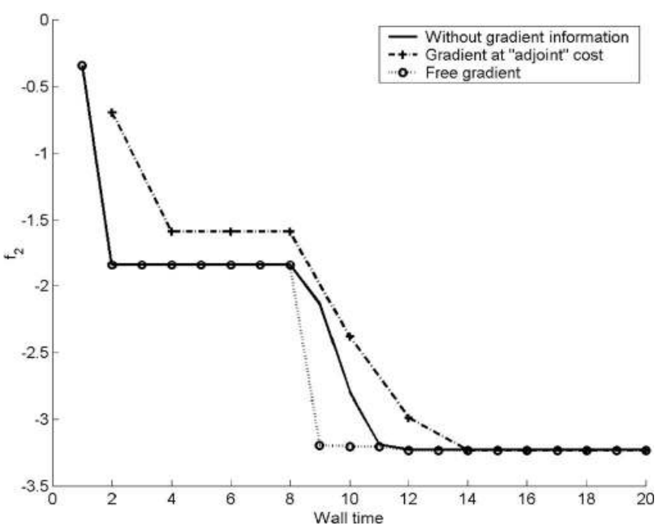

Fig. 10 Comparison of optimization strategies based on RBF and GERBF models for the modified Rosenbrock function, using four processors

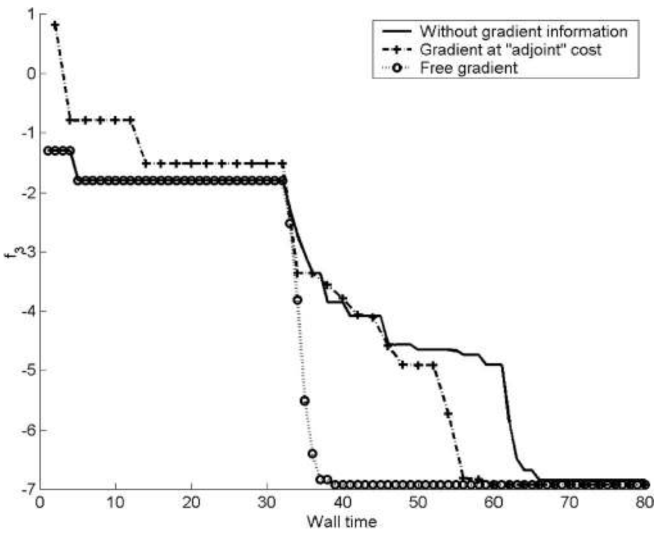

Fig. 12 Comparison of optimization strategies based on RBF and GERBF models for the Ackley function, using one processor

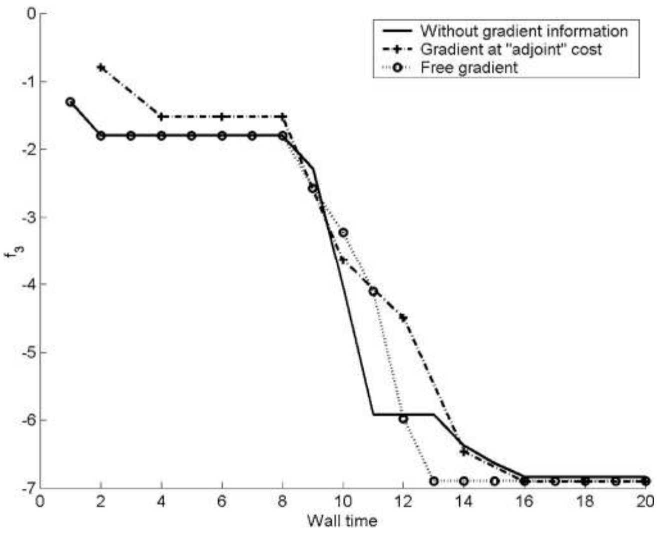

Fig. 13 Comparison of optimization strategies based on RBF and GERBF models for the Ackley function, using four processors

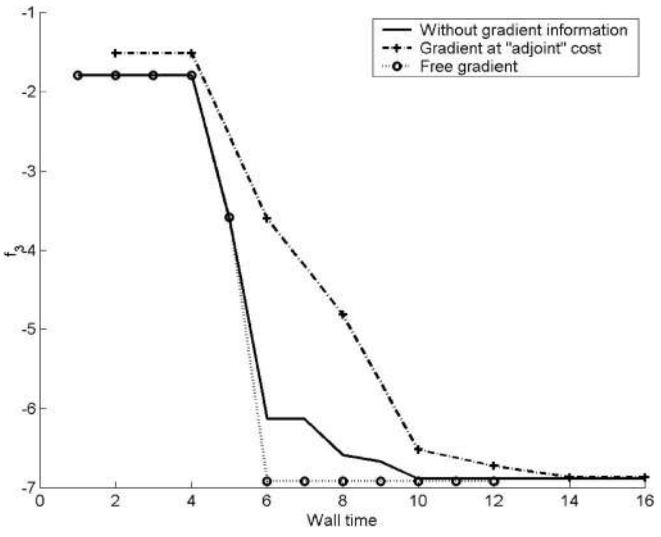

Fig. 14 Comparison of optimization strategies based on RBF and GERBF models for the Ackley function, using eight processors

One of the most important aspects of such optimization schemes is the efficiency of the parallelization. As shown in Tables 3 and 4, this varies from case to case. For example, when optimizing the Rosenbrock function using the free gradient-enhanced model, the speedup is nearly linear, whereas in the "adjoint gradient" model-based optimization of the Ackley function, the results on eight processors indicate a sharp drop in efficiency.

Finally, we consider how these surrogate-based optimization strategies perform against some frequently used optimization procedures. Here we assume the gradients are free and consider a GA, a gradient-based method (BFGS) and our optimization scheme utilizing GERBF predictions. Results for both the modified Rosenbrock and Ackley functions are shown in Figs. 15 and 16 respectively. Here we arbitrarily use $N_p = 4$.

Table 3 Comparison of number of evaluations (i.e. wall time × N_p) needed to reach convergence for the modified Rosenbrock function

N_p	Without grad.	"Adjoint" grad.	Free grad.
1	47	40	39
4	48	56	48
8	64	80	40

Table 4 Comparison of number of evaluations (i.e. wall time × N_p) needed to reach convergence for the Ackley function

N_p	Without grad.	"Adjoint" grad.	Free grad.
1	66	60	39
4	64	64	52
8	80	112	48

The GA and BFGS convergence histories are averaged over 50 optimization runs. When we consider the GA, new members of the population are spread over the N_p processors to make use of the parallel computing architecture. When considering BFGS we perform one optimization run on each processor, therefore we are using four random restarts. Multiple restarts are generally required when using gradient-based optimizers, as they become trapped in local minima. The convergence histories show the best objective function value found so far after each parallel run. Finally, we compare these approaches to the GERBF-based optimization approach (again, using four processors).

The results can be summarized as follows. In these cases the GA optimization performs relatively poorly; it is at an obvious disadvantage, as it makes no use of the free gradient information. The BFGS method performs much better: it is almost competitive with our GERBF

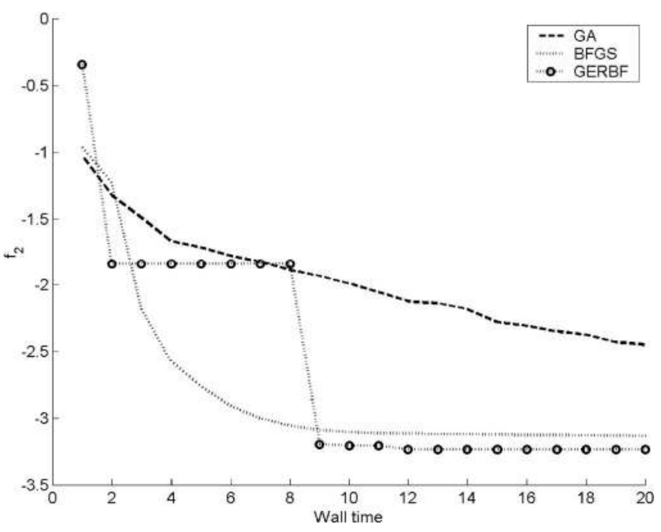

Fig. 15 Comparison of optimization techniques on the modified Rosenbrock function with free gradients

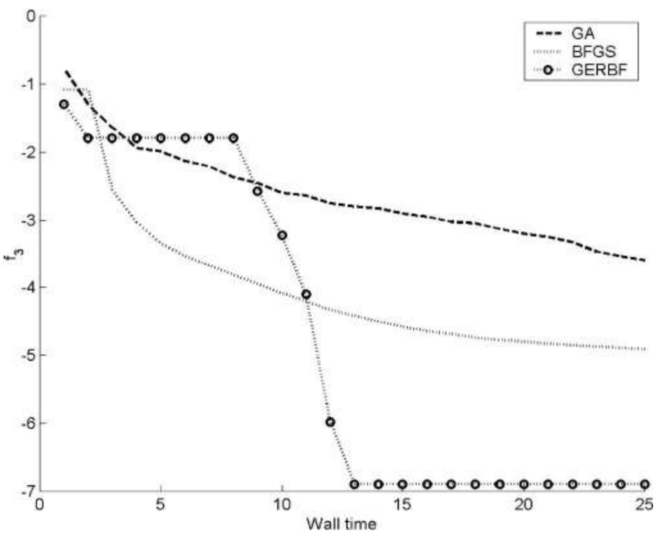

Fig. 16 Comparison of optimization techniques on the Ackley function with free gradients

based optimization approach on the modified Rosenbrock function, however it does often become trapped in one of the local minima, hence it usually converges to a higher value. On the more strongly multimodal Ackley function the BFGS method performs much worse, often becoming trapped in local minima and also converging to these minima at a slower rate. Our expected improvement approach using GERBF-based optimization consistently finds the global minimum very quickly. Our conclusions were similar when we considered gradients at "adjoint" cost.

4.3 Structural optimization example

With many finite element models the derivatives with respect to the design parameters can be available very cheaply. For example Haftka (1993) presents a semi-analytic method, whereby sensitivities can be obtained at a fraction of the cost of the analysis itself.

In this final example we consider optimization of the spoked structure shown in Fig. 17 (left). This model is made up entirely of beam elements the thickness of which can be altered in a variety of ways. The part of the structure being optimized is shown on the right of the figure. Six design parameters define the geometry, five of which describe the ring cross section whilst the sixth describes the spoke sections. The rest of the model simply enforces suitable boundary conditions. Our industrial collaborators provided realistic loadings on the structure. We note that with this parameterization the derivatives are available at negligible cost.

The goal here is to minimize the maximum von Mises stress within the structure such that the weight does not exceed a predefined value, here taken as 16.0 kg. The evaluation of the stress involves solving the finite element

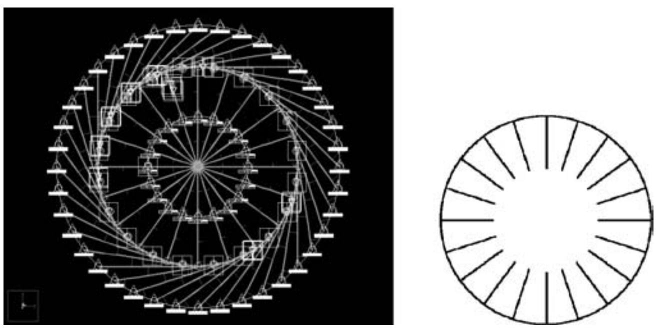

Fig. 17 Full FE model (left) and part of structure to be optimized (right)

problem at some cost. The weight is a straightforward function and can be evaluated using a simple analytic expression. We compare the gradient-enhanced approximation based approach to standard approaches that do not include sensitivity information.

We construct our original approximation using only 16 finite element evaluations for designs distributed evenly throughout the design space using an optimal Latin hypercube design. The accuracy of the models was checked against a database of 1000 previous runs. Without incorporating gradients there was an average error of 7279.4 and a correlation of 0.3868. Utilizing cheap gradient information led to a design with an average error of 4309.1 and an average correlation of 0.7903.[3] The latter approach thus turns out to be much more accurate on this problem. Secondly, we perform surrogate-based optimization using these two approximation models employing the strategy shown in Fig. 3. Figure 18 shows the re-

[3] Usually, this would of course not be possible (or indeed, necessary, for the purposes of optimization). Here we have only calculated these figures to gain an insight into the effect of gradient-enhancement on the global accuracy of the model.

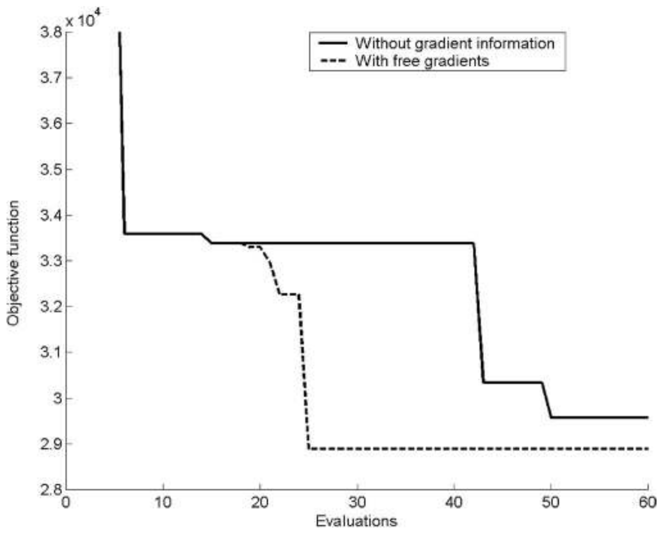

Fig. 18 Comparison of optimization techniques on the structural example assuming one processor

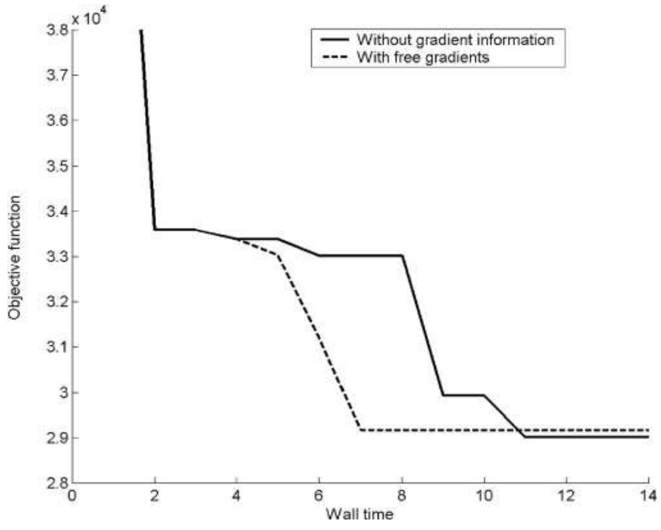

Fig. 19 Comparison of optimization techniques on the structural example assuming four processors. One wall time unit is the cost of one FE evaluation

sults assuming one processor is available, whereas Fig. 19 shows a similar result when four processors are available. As can be seen, utilizing cheap gradient information allows near optimal solutions to be found in significantly less wall time than methods that take no account of this information.

5 Conclusions

In this paper we discuss a parallel surrogate modelling-based approach to optimization. The parallel updates here come from an expected improvement measure, however other potential updating schemes are also highlighted. Surrogate models with and without gradient information have been considered. When the gradients are included the cost is assumed first to be free and then requiring the cost of one objective function evaluation. In terms of the model's accuracy it appears to make sense to incorporate this gradient information, even if the derivatives are available at adjoint cost. From the point of view of optimization the situation is less clear-cut. However, if gradients are available virtually for free, as is often the case in finite element models, then their use unquestionably makes sense. As the gradient-enhanced approximations are much more accurate than models that do not utilize gradient information, we could also start optimization using these models from a smaller DoE. Our approaches have been compared to more traditional optimization strategies and are both more efficient and find better optima in the cases considered. We plan to investigate other test cases, both analytic and using more sophisticated CFD or FE models in the near future. We also plan to apply the above techniques to problems that suffer from numerical noise using methods available in the RBF regression literature.

Acknowledgements This work has been supported by the University Technology Partnership, a collaboration between Rolls-Royce, BAE Systems and the Universities of Sheffield, Cambridge and Southampton. We would also like to thank Rolls-Royce for providing the industrial case study used in Sect. 4.3.

References

Chung, H.S.; Alonso, J.J. 2001: Using gradients to construct response surface models for high-dimensional design optimization problems. *AIAA 39th Aerosp. Sci. Meeting and Exhibit*, Reno, NV, January 8–11, 2001, AIAA-2001-0922

Chung, H.S.; Alonso, J.J. 2002: Design of a low-boom supersonic business jet using cokriging approximation models. *9th AIAA/ISSMO Symp. Multidisc. Anal. Optim.*, Atlanta, GA, September 4–6, 2002, AIAA-2002-5598

Gibbs, M.N. 1997: Bayesian Gaussian Processes for Regression and Classification. PhD Thesis, University of Cambridge

Haftka, R. 1993: Semi-analytical static non-linear structural sensitivity analysis. *AIAA J* **31**, 1307–1312

Jones, D.R. 2001: A taxonomy of global optimization methods based on response surfaces. *J Global Optim* **21**, 345–383

Jones, D.R.; Schonlau, M.; Welch, W.J. 1998: Efficient global optimization of expensive black-box functions. *J Global Optim* **13**, 455–492

Leary, S.J.; Bhaskar, A.; Keane, A.J. (to appear): A derivative based surrogate model for approximating and optimizing the output of an expensive computer simulation. *J Global Optim* (to appear)

Mackay, 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

Martin, J.D.; Simpson, T.W. 2002: Use of adaptive metamodelling for design optimization. *9th AIAA/ISSMO Symp. Multidisc. Anal. Optim.*, Atlanta, GA, September 4–6, 2002, AIAA-2002-5631

Morris, M.D.; Mitchell, T.J. 1995: Exploratory designs for computer experiments. *J Stat Plan Infer* **43**, 381–402

Morris, M.D.; Mitchell, T.J.; Ylvisaker, D. 1993: Bayesian design and analysis of computer experiments: Use of derivatives in surface prediction. *Technometrics* **35**, 243–255

Myers, R.H.; Montgomery, D.C. 1995: *Response surface methodology: Process and product optimization using designed experiments* New York: Wiley

Powell, M.J.D. 1987: Radial basis functions for multivariable interpolation: a review, *In Algorithms for Approximation* Oxford: Clarendon, 143–167

Sacks, J.; Welch, W.J.; Mitchell, T.J.; Wynn, H.P. 1989: Design and analysis of computer experiments (with discussion). *Stat Sci* **4**, 409–435

Sasena, M.J.; Papalambros, P.; Goovaerts, P. 2002: Exploration of metamodelling sampling criteria for constrained global optimization. *Eng Optim* **34**, 263–278

Schonlau, M. 1997: Computer Experiments and Global Optimization. *PhD Thesis, University of Waterloo, Canada*

Sobol, I.M. 1979: On the systematic search in a hypercube. *SIAM J Numer Anal* **16**, 790–793

Trosset, M.W.; Torczon, V. 1997: Numerical Optimization Using Computer Experiments. *ICASE Report TR-97-38, NASA Langley Research Center, Hampton, Virginia*

van Keulen, F.; Vervenne, K. 2002: Gradient-enhanced response surface building. *9th AIAA/ISSMO Symp. Multidisc. Anal. Optimiz.,* Atlanta, GA, September 4–6, 2002, AIAA-2002-5455

White, H.; Gallant, A.R.; Kornik, K.; Stinchcombe, M.; Wooldridge, J. 1992: *Artificial neural networks: Approximation and learning theory* Blackwell publishers

Appendix:

Here we present details of the test functions used in Sects. 4.1 and 4.2: the two-dimensional Branin function and the k-dimensional Modified Rosenbrock and Ackley functions (their five-dimensional variants have been used for the experiments described in this paper). They have been chosen to cover a range of complexities, with the Branin function being the simplest, the Modified Rosenbrock function having moderate multimodality, while the Ackley function featuring a high number of local optima.

Branin function

$$f_1(x_1,x_2) = a(x_2 - bx_1^2 + cx_1 - d)^2 + e(1-f)\cos(x_1) + e \tag{A.1}$$

where $a=1,\ b=\dfrac{5.1}{4\pi^2},\ c=\dfrac{5}{\pi}, d=6,\ e=10,\ f=\dfrac{1}{8\pi}$ and $(x_1,x_2)\in[-5,10]\times[0,15]$.

Modified Rosenbrock function

$$f_2(\mathbf{x}) = \frac{1}{a}\left[\sum_{i=1}^{k-1} 100(x_{i+1}-x_i^2)^2 + (1-x_i)^2 + \sum_{i=1}^{k} 75\sin(5(1-x_i)) - b\right] \tag{A.2}$$

where $a=206,\ b=300$ and $\mathbf{x}\in[-1,1]^k$.

Ackley's function

$$f_3(\mathbf{x}) = \frac{1}{f}\left\{-a\exp\left[-b\sqrt{\frac{1}{n}\sum_{i=1}^{k}x_i^2}\right] - \exp\left[\frac{1}{n}\sum_{i=1}^{k}\cos(cx_i)\right] + a + \exp(1) + d\right\} \tag{A.3}$$

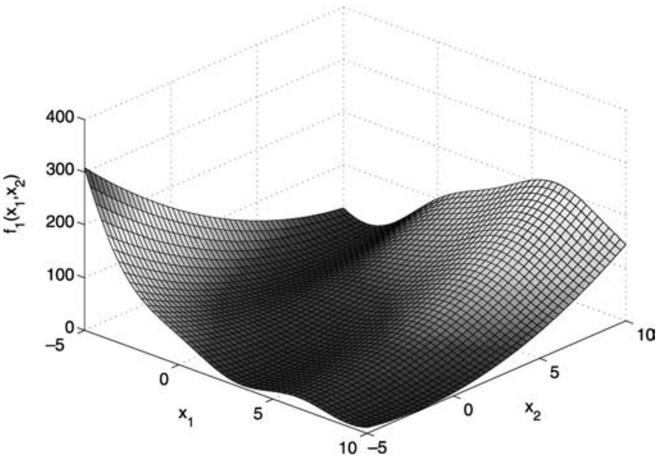

Fig. 20 Branin function

where $a=20,\ b=0.2,\ c=2\pi,\ d=5.7,\ f=0.8$ and $\mathbf{x}\in[-2.048,2.048]^k$.

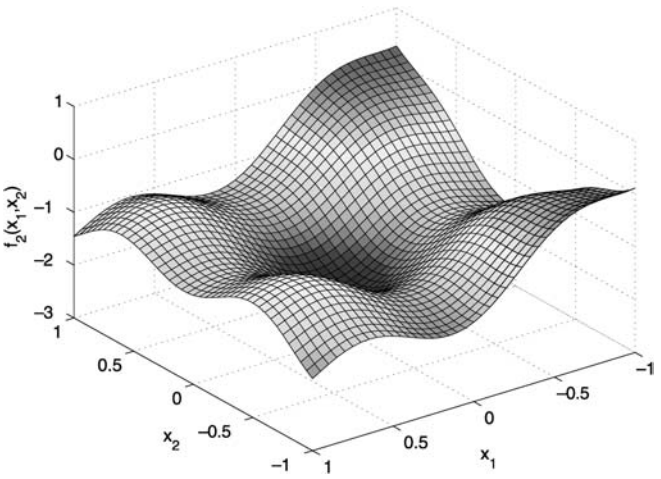

Fig. 21 Two-variable version of Modified Rosenbrock's function

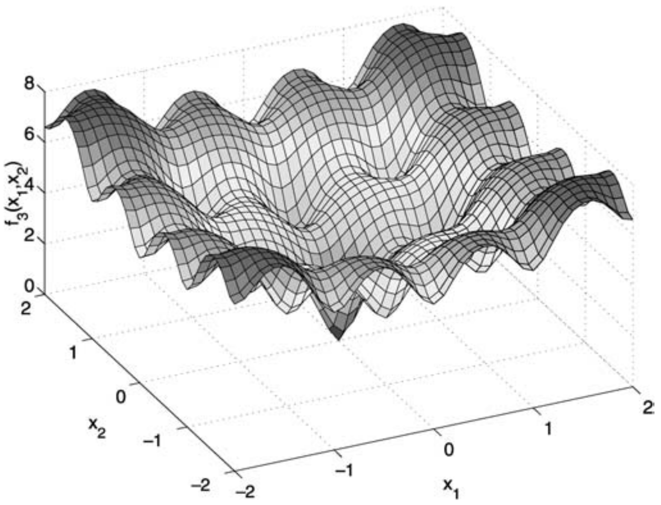

Fig. 22 Two-variable version of Ackley's function