Engineering Optimization
Vol. 43, No. 6, June 2011, 1–28

RESEARCH ARTICLE

The Development of a Hybridized Particle Swarm for Kriging Hyperparameter Tuning

D.J.J. Toal,[a]*N. W. Bressloff,[a] A.J. Keane[a] and C.M.E. Holden[b]

[a] *University of Southampton, Southampton, SO17 1BJ, United Kingdom*; [b] *Airbus Operations Ltd., New Filton House, Filton, Bristol BS99 7AR , United Kingdom*
(*Received 19 February 2010; final version received 3 June 2010*)

Optimizations involving high fidelity simulations can become prohibitively expensive when an exhaustive search is employed. To remove this expense a surrogate model is often constructed. One of the most popular techniques for the construction of such a surrogate model is that of kriging. However, the construction of a kriging model requires the optimization of a multi-model likelihood function, the cost of which, can approach that of the high fidelity simulations upon which the model is based. The following paper describes the development of a hybridized particle swarm algorithm which aims to reduce the cost of this likelihood optimization by drawing on an efficient adjoint of the likelihood. This hybridized tuning strategy is compared to a number of other strategies with respect to the inverse design of an airfoil as well as the optimization of an airfoil for minimum drag at a fixed lift.

Keywords: kriging; particle swarm optimization; hyperparameter tuning;

Nomenclature

$\boldsymbol{R}$	Correlation matrix
$\boldsymbol{V}$	Particle velocity vector
$\boldsymbol{x}_{Gbest}$	Current global best position found by swarm
$\boldsymbol{x}_{Pbest}$	Individual best position found by a particle
c_1	Particle swarm cognitive parameter
c_2	Particle swarm social parameter
d	No. of dimensions
K	Particle swarm constriction factor
n	No. of sample points

*Corresponding author. Email: djjt@soton.ac.uk

ISSN: 0305-215X print/ISSN 1029-0273 online

DOI: 10.1080/0305215X.2010.508524
http://www.informaworld.com

p	Hyperparameter determining smoothness
w	Inertial weight
x	Design variable
y	Objective function value
λ	Regression constant
μ	Mean
ϕ	Concentrated log-likelihood
σ	Standard deviation
θ	Hyperparameter determining correlation decrease

1. Introduction

Kriging was first used by Krige (1951) to estimate mineral concentrations within a particular region, and has since been adapted for use in the creation of surrogate models of deterministic computational experiments; a process popularized by Sacks *et al.* (1989). Of the numerous types of response surface models, kriging is perhaps one of the most effective due to its ability to model complicated responses through interpolation or regression whilst also providing an error estimate of the predictor. Since its initial application to surrogate modelling, kriging has been applied to a variety of engineering problems including, aerodynamic (Hoyle *et al.* (2006) and Forrester *et al.* (2006)), structural, Sakata *et al.* (2003), and multi-objective design problems by D'Angelo and Minisci (2005) and Keane (2006).

A typical response surface based optimization commences with an initial sampling of the design space using an expensive simulation. A surrogate model, in this case a kriging model, is then constructed which models the response of the objective function to changes in the magnitude of the design variables. This model can then be searched using a suitable global optimizer and updated in regions of interest indicated by the models prediction of the objective function or some other metric such as the expected improvement, Jones (2001). This process is repeated until an appropriate termination criterion is met, for example, a fixed budget of full simulations has been exhausted. Comprehensive reviews of the state of the art in surrogate modelling can be found in Simpson *et al.* (2001), Queipo *et al.* (2005) and Wang and Shan (2007).

The construction of a kriging model requires the solution to a complex multi-modal optimization problem in order to find a set of kriging hyperparameters which best relate the krig to the sample data. The optimization of these hyperparameters via maximization of the likelihood function requires an $O(n^3)$ factorization for each evaluation of the likelihood, where n is the number of sample points upon which the model is constructed. The cost of this optimization therefore increases dramatically as the number of sample points increases. As the number of sample points is typically linked to the number of dimensions, d, in the underlying problem, for example $10d$ in the case of Jones *et al.* (1998) or $3d$ in the case of Jin *et al.* (2001), the cost of evaluating the likelihood can quickly become an issue at high dimensions. The number of hyperparameters is typically $2d$ and this leads to a more complex hyperparameter optimization as the number of dimensions increases.

Figure 1 helps to illustrate this increase in tuning cost by demonstrating the total

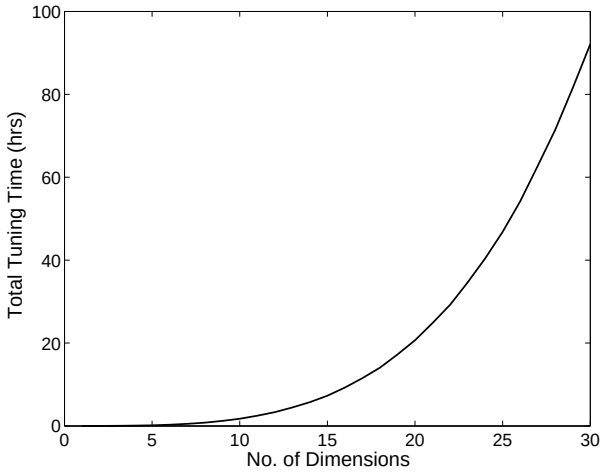

Figure 1. An illustration of the total time spent tuning the modelling parameters of a krig over the course of an optimization as problem dimensionality increases, Toal *et al.* (2009)

time spent tuning the hyperparameters over an entire optimization[2]. In this case a total simulation budget of $15d$ is assumed with one third employed in the initial sampling plan as per Sóbester *et al.* (2005) with the remainder used to update the model in batches of ten. The hyperparameters are assumed to be tuned after every set of updates via a 5,000 evaluation genetic algorithm followed by a 5,000 evaluation dynamic hill climber, Toal *et al.* (2008b). It can be observed from Figure 1 that the application of such a scheme can result in the total hyperparameter tuning cost quickly exceeding the cost of the high fidelity simulations which may have been used in the model's construction.

To attempt to reduce the cost associated with hyperparameter tuning a number of different strategies have been proposed in the literature. Toal *et al.* (2008b) demonstrated that although the retuning of the hyperparameters throughout an optimization is a necessity for the overall optimization to be effective, it may be possible to reduce the cost by retuning the hyperparameters after every other set of updates with minimal impact. Toal *et al.* (2008a) also demonstrated that by reparameterizing the original problem to exclude poorly performing designs, the size of the kriging correlation matrix can be reduced thereby significantly reducing the overall tuning cost.

The likelihood function used in the tuning process is a smooth differentiable function and therefore lends itself to the application of gradient information in its optimization. The literature includes a number of examples of the application of gradient information in likelihood optimization. Park and Baek (2001) derived an analytical gradient of the likelihood for use in a local quasi-Newton optimization. Zhang and Leithead (2005) took this a step further and derived an analytical Hessian of the likelihood and employed this in conjunction with a trust region search. Leithead and Zhang (2007) reduced the cost of approximate likelihoods and derivatives to an $O(n^2)$ operation through an approximation to the inverse of the covariance matrix via the BFGS, Broyden (1970), updating formula.

Each of these methods is a local optimization of the likelihood and as such will only locate the global optimum if initialized in the region of that optimum or if an appropriate restart procedure is adopted. It should be noted however, that there are cases when such a local optimizer may be effective in finding the optimal hyperparameters.

[2]Total tuning time on an Intel Core 2 processor running at 2.4Ghz with 1Gb of RAM

Zhang and Leithead (2005) note that given a sufficiently large dataset upon which to build the surrogate model the modality of the likelihood space is greatly reduced. Such densely populated design spaces are, however, very rare in engineering design optimizations, especially when each objective function evaluation involves a costly high fidelity simulation.

Whereas the methods of Park et al. and Zhang et al. exploit an exact analytical gradient or Hessian, the approximation method of Leithead et al. still requires an initial exact inverse of the correlation matrix and may require additional inversions as the optimization progresses due to corruption of the approximation. When used in conjunction with an engineering optimization problem, where there are few sample points and the likelihood is multi-modal in nature, the computational effort spent in carrying out the initial starting inversion and subsequent restart inversions, may be better spent in performing a global exploration of the likelihood. To this end we consider here the development of a hybridized particle swarm algorithm for the purposes of likelihood optimization which employs an efficient adjoint[1] of the likelihood. The resulting optimization algorithm aims to facilitate both a global exploration of the likelihood space and rapid convergence to an optimum through the utilization of the available cheap gradient information.

The following section first briefly describes both kriging and an efficient adjoint of the likelihood derived using the linear algebra results collated by Giles (2008). The paper then moves on to describe particle swarm optimization algorithms and reviews a number of different hybridized swarms from the literature. The framework for a new hybrid particle swarm is defined in Section 4 and then optimized specifically for the kriging hyperparameter tuning problem in Section 5. The performance of the final swarm is then compared to a number of other tuning strategies with regard to an airfoil inverse design problem, Section 6, and the optimization of an airfoil for minimum drag at a fixed lift, Section 7.

2. An Overview of Kriging

Given a pair of objective function values, $y(\boldsymbol{x}_i)$ and $y(\boldsymbol{x}_j)$ which are given by a function of the vectors of design variables $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, of length d, the objective function values will generally be similar if $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are close together within a design space. This can be modelled statistically by assuming that the correlation between two sets of random variables $Y(\boldsymbol{x}_i)$ and $Y(\boldsymbol{x}_j)$ is given by,

$$\mathrm{Corr}\left[Y(\boldsymbol{x}_i), Y(\boldsymbol{x}_j)\right] = \exp\left(-\sum_{l=1}^{d} 10^{\boldsymbol{\theta}_l} \|\boldsymbol{x}_{i_l} - \boldsymbol{x}_{j_l}\|^{\boldsymbol{p}_l}\right). \tag{1}$$

Here the hyperparameters $\boldsymbol{\theta}_l$ and $\boldsymbol{p}_l$ determine the rate at which the correlation decreases and the degree of smoothness in the l^{th} coordinate direction, respectively. Consider now a vector $\boldsymbol{y}$ consisting of n objective function values,

$$\boldsymbol{y} = \left[y(\boldsymbol{x}_1), \ldots, y(\boldsymbol{x}_n)\right]^T, \tag{2}$$

[1]An adjoint model computes the sensitivities of an output with respect to the output's intermediate variables. Such a model can compute the partial derivatives of outputs with respect to thousands of inputs at a cost of no more than a few function evaluations, Griewank (2000).

where the mean is $\mathbf{1}\hat{\mu}$, and $\mathbf{1}$ is an n by 1 vector of ones. The covariance of $\boldsymbol{y}$ may then be written as

$$\mathrm{Cov}(\boldsymbol{y}) = \sigma^2 \boldsymbol{R}, \tag{3}$$

where σ^2 is the variance and the elements of the matrix $\boldsymbol{R}$, the correlation matrix, are given by Equation 1. Values of $\boldsymbol{\theta}_l$ and $\boldsymbol{p}_l$ (the hyperparameters), are then chosen to maximize the likelihood on the observed data set $\boldsymbol{y}$. This maximum likelihood function is defined as:

$$\frac{1}{(2n)^{\frac{n}{2}}(\sigma^2)^{\frac{n}{2}}|\boldsymbol{R}|^{\frac{1}{2}}} \exp\left[\frac{-(\boldsymbol{y}-\mathbf{1}\hat{\mu})^T \boldsymbol{R}^{-1}(\boldsymbol{y}-\mathbf{1}\hat{\mu})}{2\sigma^2}\right], \tag{4}$$

or after taking natural logarithms of the function,

$$-\frac{n}{2}\ln(\sigma^2) - \frac{1}{2}\ln(|\boldsymbol{R}|) - \frac{(\boldsymbol{y}-\mathbf{1}\hat{\mu})^T \boldsymbol{R}^{-1}(\boldsymbol{y}-\mathbf{1}\hat{\mu})}{2\sigma^2}. \tag{5}$$

Expressions for the optimal values of the mean

$$\hat{\mu} = \frac{\mathbf{1}^T \boldsymbol{R}^{-1} \boldsymbol{y}}{\mathbf{1}^T \boldsymbol{R}^{-1} \mathbf{1}} \tag{6}$$

and variance

$$\hat{\sigma}^2 = \frac{1}{n}(\boldsymbol{y}-\mathbf{1}\hat{\mu})^T \boldsymbol{R}^{-1}(\boldsymbol{y}-\mathbf{1}\hat{\mu}) \tag{7}$$

can be found by taking partial derivatives of the log likelihood and equating them to zero. A concentrated likelihood function, Jones (2001), can then be derived by substituting the expressions for the optimal mean and variance into the log likelihood function and neglecting the constant $-n/2$,

$$\phi = -\frac{n}{2}\ln(\hat{\sigma}^2) - \frac{1}{2}\ln(|\boldsymbol{R}|). \tag{8}$$

The concentrated likelihood function is dependant only on the correlation matrix and hence on the hyperparameters which are tuned in an attempt to maximize the function.

As previously demonstrated the selection, or tuning, of these hyperparameters is an optimization problem in its own right and can be expensive and complex, with the overall cost dependent on both the dimensionality of the optimization problem and the number of sample points.

Recently Toal *et al.* (2009) demonstrated the application of algorithmic differentiation to the calculation of derivatives of the concentrated likelihood function. The resulting gradient calculations were demonstrated to be more efficient than the traditional analytical method presented by Park and Baek (2001). The method was less sensitive to both increases in the number of variables and in the number of sample points. However, this method relied upon the application of a reversely differentiated Cholesky factorization and reverse differentiations of both the forward and backward substitution algorithms. The adjoint can be made both more efficient and easier to code through the consideration

of the linear algebra results presented by Giles (2008) which draws on the previous work of Dwyer and M.S. (1948).

Using the notation of Griewank (2000), given an initial seeding for the adjoint of the concentrated likelihood of $\bar{\phi} = 1$ and using reverse algorithmic differentiation, from Equation 8 the adjoint of both the variance and the determinant of the correlation matrix can be found to be,

$$\bar{\sigma}^2 = -\frac{n}{2\hat{\sigma}^2} \tag{9}$$

and

$$\overline{|\boldsymbol{R}|} = -\frac{1}{2|\boldsymbol{R}|}, \tag{10}$$

where the bar denotes the adjoint of the intermediate variable. From the results presented by Giles (2008), the adjoint of the second quadratic matrix product,

$$\boldsymbol{C} = \boldsymbol{B}^T \boldsymbol{A}^{-1} \boldsymbol{B} \tag{11}$$

is given by,

$$\bar{\boldsymbol{A}} = -\boldsymbol{A}^{-T} \boldsymbol{B} \bar{\boldsymbol{C}} \boldsymbol{B}^T \boldsymbol{A}^{-T}, \tag{12}$$

where $^{-T}$ denotes the transpose of the inverse matrix. Using this result in conjunction with the equation for the variance, which is of a similar form, and substituting $\boldsymbol{A}$ for $\boldsymbol{R}$, $\boldsymbol{B}$ for $(\boldsymbol{y} - \mathbf{1}\hat{\mu})$ and $\bar{\boldsymbol{C}}$ for Equation 9, the component of the adjoint of the correlation matrix due to the variance can be calculated to be,

$$\frac{1}{2\hat{\sigma}^2} \boldsymbol{R}^{-T} (\boldsymbol{y} - \mathbf{1}\hat{\mu})^T (\boldsymbol{y} - \mathbf{1}\hat{\mu})^T \boldsymbol{R}^{-T}. \tag{13}$$

Giles (2008) also notes that if,

$$C = \det \boldsymbol{A} \tag{14}$$

then the adjoint of the matrix $\boldsymbol{A}$ is,

$$\bar{\boldsymbol{A}} = \bar{C} C \boldsymbol{A}^{-T}. \tag{15}$$

Using this result and replacing $\bar{C}$ with Equation 10, C with $|\boldsymbol{R}|$ and $\boldsymbol{A}$ with $\boldsymbol{R}$ the component of the adjoint of the correlation matrix due to the log of the determinant of the correlation matrix is,

$$-\frac{1}{2} \boldsymbol{R}^{-T}. \tag{16}$$

Combining both Equations 13 and 16 results in an analytical expression for the adjoint of the correlation matrix,

$$\bar{\boldsymbol{R}} = \frac{1}{2\hat{\sigma}^2} \boldsymbol{R}^{-T} (\boldsymbol{y} - \mathbf{1}\hat{\mu})^T (\boldsymbol{y} - \mathbf{1}\hat{\mu})^T \boldsymbol{R}^{-T} - \frac{1}{2} \boldsymbol{R}^{-T}. \tag{17}$$

As mentioned previously this formulation has no need for a bespoke reverse differentiation of the Choleksy factorization, or the back and forward substitution algorithms (Toal *et al.* (2009)) and can therefore be implemented with ease using the standard libraries for matrix and vector operations. Using this formulation the calculation of $\bar{\boldsymbol{R}}$ can also take advantage of the fact that the correlation matrix is symmetric and therefore $\boldsymbol{R}^{-T}(\boldsymbol{y} - \mathbf{1}\hat{\mu})^T$ and $(\boldsymbol{y} - \mathbf{1}\hat{\mu})^T\boldsymbol{R}^{-T}$ have already been calculated during the initial calculation of the variance, Equation 7. This reduces the calculation of $\bar{\boldsymbol{R}}$ to essentially a vector-vector multiplication followed by a matrix addition, which is more efficient than the reversely differentiated formulation presented previously by Toal *et al.* (2009).

The final calculation of the derivatives of the likelihood with respect to each of the hyperparameters follows the remainder of the formulation of Toal *et al.* (2009). The derivative of the likelihood with respect to the l^{th}, θ hyperparameter is therefore,

$$\frac{\partial \phi}{\partial \theta_l} = \ln 10 \sum_{ij} -10^{\theta_l} ||\boldsymbol{x}_{i_l} - \boldsymbol{x}_{j_l}||^{p_l} \boldsymbol{R}_{ij} \bar{\boldsymbol{R}}_{ij} \tag{18}$$

and the derivative with respect to the l^{th}, p hyperparameter is

$$\frac{\partial \phi}{\partial p_l} = \sum_{ij} -10^{\theta_l} ||\boldsymbol{x}_{i_l} - \boldsymbol{x}_{j_l}||^{p_l} \ln ||\boldsymbol{x}_{i_l} - \boldsymbol{x}_{j_l}|| \boldsymbol{R}_{ij} \bar{\boldsymbol{R}}_{ij}. \tag{19}$$

These derivative calculations can be made more efficient by noting that the calculation of $\boldsymbol{R}_{ij}\bar{\boldsymbol{R}}_{ij}$ is common to all cases and therefore needs to be calculated only once. The derivative of the likelihood with respect to a regression constant λ can be easily calculated from $\bar{\boldsymbol{R}}$,

$$\frac{\partial \phi}{\partial \lambda} = 10^{\lambda} \ln 10 \sum_{i} \bar{\boldsymbol{R}}_{ii}, \tag{20}$$

assuming that 10^{λ} has been added to the diagonal of the correlation matrix.

The interested reader may wish to consult Giles (2008) for the complete details of the derivation of the linear algebra results employed in the derivation of the likelihood adjoint and the excellant book by Griewank (2000) for more information on the process of algorithmic differentiation and its applications.

Assuming that an appropriate set of hyperparameters have been found it is necessary to 'search' the resulting surrogate for a set of update points which are to be evaluated using the true objective function evaluation. To make a prediction at a new point, $\boldsymbol{x}^*$, this point is added to the existing data as the $(n+1)^{\text{th}}$ observation. Using the previously defined hyperparameters an augmented likelihood, Jones (2001), can be calculated. The surrogates prediction of $y(\boldsymbol{x}^*)$ is therefore the value which maximises this likelihood giving,

$$y(\boldsymbol{x}^*) = \hat{\mu} + \boldsymbol{r}^T\boldsymbol{R}^{-1}(\boldsymbol{y} - \mathbf{1}\hat{\mu}). \tag{21}$$

More information on the derivation of the predictor and the other infill criterion such as expected improvement can be found in Jones (2001).

3. Particle Swarm Optimization

3.1. *The Basic Particle Swarm*

The particle swarm is one of the most recent stochastic optimization methods, and also one of the simplest. It was originally developed by Eberhart and Kennedy (1995) after they adopted simulations of simple social behavior for use in optimization.

Like genetic algorithms the particle swarm is inspired by nature, though instead of modelling evolutionary processes, the particle swarm endeavors to model the social behavior of a population of animals. Initial simulations of social behavior by Eberhart et al. drew on the previous work of Heppner and Grenander (1990), where a flock of birds flew around searching for a cornfield before landing. In these simulations every member of the population could remember its previous best position, in this case how far it was from the cornfield, and each member could 'see' the global best position that another member of the population had found. By permitting each bird to remember a previous best position, birds that overflew a good position were pulled back to that position. Allowing the birds to see the current global best position, in social terms, provides the group with a standard which the other group members attempt to attain.

The extension of this social behavior model to the field of optimization resulted in a very simple equation which updates the velocity of each swarm member. Given a velocity, $\boldsymbol{V}_i$, at the i^{th} iteration, and information on the current global best position, $\boldsymbol{x}_{Gbest}$ and the best position that an individual has found, $\boldsymbol{x}_{Pbest}$, the velocity of the particle at the next iteration can be found by,

$$\boldsymbol{V}_{i+1} = w\boldsymbol{V}_i + c_1\boldsymbol{r}_1(\boldsymbol{x}_{Pbest} - \boldsymbol{x}_i) + c_2\boldsymbol{r}_2(\boldsymbol{x}_{Gbest} - \boldsymbol{x}_i), \tag{22}$$

where $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$ are vectors of random numbers in the range [0,1] and w, c_1, and c_2 are the inertial weight, the cognitive parameter and social parameter, respectively, Blasi and Del Core (2007). With the new velocity of a particle calculated, the position of a particle at the next iteration can be easily found,

$$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \boldsymbol{V}_{i+1}. \tag{23}$$

The updated velocity given by Equation 22 has three main components. The first component is the velocity that the particle is currently travelling at; this is termed the inertial component. The second component of the velocity calculation tends to return the particle to a previous best point, this is sometimes referred to as 'simple nostalgia', Kennedy and Eberhart (1995). The final component of the velocity calculation tends to move the particle towards the location of the best point found so far in the optimization.

The two learning parameters, c_1 and c_2, in Equation 22 balance the local exploitation and global exploration of the algorithm. A large c_1 relative to c_2 forces each particle to return to the best positions that they, themselves, have found, largely ignoring the global best point. A large c_2 relative to c_1 results in each particle moving rapidly towards the global best position with little exploitation of any local minima found. A balance is therefore required between these two factors to ensure the algorithm moves to a global optimum, but in doing so exploits any locally optimal regions it encounters. For this reason in Eberhart and Kennedy's original algorithm, both c_1 and c_2, equal 2.

The inertial weight was an addition to the original formulation by Shi and Eberhart (1998) which improved the performance of the algorithm. Typically this inertial weight decreases linearly with each iteration of the algorithm. Eberhart and Shi (2000) found

however that modifying Equation 22 to use a constriction factor, K and restricting the velocity of each particle to $\boldsymbol{V}_{max}$ negates the need for a gradual reduction in inertial weight, and results in a performance increase. Equation 22 therefore becomes,

$$\boldsymbol{V}_{i+1} = K[\boldsymbol{V}_i + c_1\boldsymbol{r}_1(\boldsymbol{x}_{Pbest} - \boldsymbol{x}_i) + c_2\boldsymbol{r}_2(\boldsymbol{x}_{Gbest} - \boldsymbol{x}_i)] \tag{24}$$

subject to

$$\boldsymbol{V}_{i+1} \leq \boldsymbol{V}_{max} \tag{25}$$

where

$$K = \frac{2}{|2 - \gamma - \sqrt{\gamma^2 - 4\gamma}|}, \qquad \gamma = c_1 + c_2 \quad \text{and} \quad \gamma > 4. \tag{26}$$

The constriction factor is in effect a special implementation of the traditional particle swarm equation, only instead of the inertial weight reducing with each iteration it remains constant and is a function of the learning factors. The starting magnitude of the inertial weight and the rate at which it decays no longer need to be specified a priori.

The restriction in the velocity applied to each particle tends to prevent them from leaving the design space and also simulates an incremental learning process. The selection of this parameter has an impact on the performance of the optimization. Reducing this parameter prevents the particles from overshooting interesting regions but reduces the convergence rate of the algorithm. Setting this parameter too small however prevents the particle escaping from local minima. Typically $\boldsymbol{V}_{max}$ is defined as a fraction of the difference between the upper and lower bounds of the design space, Liu *et al.* (2006).

A typical particle swarm algorithm begins with the sampling of the design space at a number of points to construct the initial population. Each member of the swarm has its corresponding objective function calculated and is given a random initial velocity within the limits of $\boldsymbol{V}_{max}$. The global best point is recorded and the previous best point for each particle is initialized to the current position. The velocity is then updated using either Equations 22 or 24 and from this velocity the particle's new position is calculated. The objective function is then calculated at this new position and the global and previous best points are updated as necessary. The process then continues until a specified stopping criterion is reached.

Particle swarms have grown in popularity since their inception by Eberhart and Kennedy. They have been applied in the conceptual design of aircraft, Blasi and Del Core (2007), and compared favorably to genetic algorithms and simulated annealing in the optimization of airfoils, Ray and Tsai (2004), and analytical functions, see for example, Angeline (1998), Brandstatter and Baumgartner (2002) and Venter and Sobieszcaznski-Sobieski (2003).

Particle swarms do however suffer from the same exploitative deficiencies as genetic algorithms, that is, they are capable of finding the region of an optimal design but not the precise answer. As with genetic algorithms some effort has gone into improving the performance of particle swarms in this area, with the introduction of both hybrid particle swarms which employ a local search, Shu-Kai *et al.* (2004), or those which make specific use of gradient information in the velocity update equation, Noewl and Jannett (2004) and Ninomiya and Zhang (2008).

3.2. *Existing Hybridized Particle Swarms*

The utilization of a local search within a particle swarm aims to improve the overall convergence of the swarm to an optimum. Without this addition to the algorithm the particles tend to oscillate around the region of an optimum but make relatively little headway towards finding an exact solution. Local search algorithms typically involve some form of gradient descent and proceed quickly from an initial starting point to a precise minimum. However they cannot escape from a local minimum unless a restart procedure is employed. Combining the global exploration abilities of the particle swarm with the exploitation abilities of a local search therefore makes perfect sense.

There exists within the literature a number of different methods of combining a local search within a particle swarm, each with their own advantages and disadvantages. Local searches can be commenced from the $\boldsymbol{x}_{Gbest}$ point of a generation and carried out until convergence, Victoire and Jayakumar (2004) and Guo *et al.* (2006), but this can hamper the swarm's global exploration as particles are drawn to a $\boldsymbol{x}_{Gbest}$ point which may not change for a number of generations. This can be countered by the introduction of a diversity metric and particle repulsion, Riget and Vesterstrø (2002), or through the generation of random particles to increase diversity.

After completing a local search Liu *et al.* (2005) retains a subset of the population in the next generation with the remaining population members randomly generated. Where the random generation of particles maintains population diversity, the method proposed by Liu et al. has no control over where these points are generated and as such new points may be generated in regions of the design space previously visited and subsequently discarded.

An alternative approach is to split the total population into two subsets one utilizing a pure particle swarm approach while the other has each point optimized locally until convergence, Izui *et al.* (2007). Whilst this strategy effectively uses sensitivity information one could argue that given a fixed evaluation budget for each generation the convergence of a large number of such local searches reduces the number of available evaluations for any global exploration.

Rather than use a local search strategy, Noewl and Jannett (2004) modified the particle swarm update equation to make use of any available gradient information by adopting a gradient descent term instead of a nostalgia term. The components of the updated velocity were therefore due to the inertia of each particle, the location of the global best point and a step in the steepest descent direction.

While this approach appears attractive by simultaneously exploiting a particle's local gradient and the global best point, its effectiveness is closely dependant on the cost of the gradient. If the cost per optimization is to be maintained when using this strategy, the population size or number of generations must be reduced accordingly. This reduces either the exploration ability of the algorithm or the convergence. One could also argue that just taking a step in the direction of steepest descent is not as efficient as a quasi-Newton based optimization when it comes to an effective terminal search.

Rather than using raw gradient information in the velocity update, Ninomiya and Zhang (2008), employed a step in the direction of a BFGS search. This step then influenced the velocity of a particle's six neighbors through a modified fully informed particle swarm (FIPS) equation, Mendes *et al.* (2004). Ninomiya and Zhang realized that calculating the gradient and subsequent BFGS step for each particle at every iteration would be prohibitively expensive and therefore restricted the local step to only the global best point. Employing a FIPS based swarm with a reduced neighborhood maintains population diversity to a better extent than the simpler 'gbest' neighborhood topology of

Table 1. Pseudo code of the proposed hybrid particle swarm optimization algorithm.

Step	
1	Initialize the population members using a random Latin Hypercube
2	Initialize the particle velocities
3	Evaluate the objective function (concentrated likelihood)
4	Select a population member for refinement via a local optimization
5	Initialize $\boldsymbol{x}_{Pbest}$ and $\boldsymbol{x}_{Gbest}$
6	Update each particle's velocity and position
7	Reinitialize a proportion of the population to unexplored regions if required
8	Evaluate the objective function
9	Select a population member for refinement via a local search
10	Update $\boldsymbol{x}_{Pbest}$ and $\boldsymbol{x}_{Gbest}$
11	Return to step 6 unless the required number of generations has been reached
12	Terminal local search commencing from $\boldsymbol{x}_{Gbest}$

Kennedy and Eberhart's original swarm algorithm. Although the hybrid swarm of Ninomiya and Zhang does not suffer the same premature convergence problems of other hybrid swarms the absence of a terminal local search and the application of a random neighborhood topology, where the neighbors influenced by the global best point are in completely different areas of the design space, could be considered questionable.

4. The Proposed Hybridized Particle Swarm

Reviewing the advantages and disadvantages of a number of hybrid swarms from the literature provides a solid basis upon which to develop an algorithm for the purposes of hyperparameter optimization. From the above review a number of important features of a final strategy are apparent. A local search must be incorporated into the swarm which uses gradient information in an efficient and effective manner. Precautions must also be taken to prevent premature convergence by maintaining diversity in the population when the results of any local optimization are fed back into the swarm.

To this end a hybrid particle swarm has been developed which uses features from the literature and introduces new features similar to those used in the creation of sampling plans for computational experiments. The proposed algorithm, the pseudo code of which is presented in Table 1, begins with the initialization of every particle's position and velocity. Unlike the majority of particle swarms the initial position of each particle is defined by sampling the space using a random latin hypercube. Latin hypercubes, proposed by McKay *et al.* (1979) as an alternative to Monte Carlo sampling, partition each dimension of the design space into a series of n 'bins' of equal probability. A sample plan is then generated via a random permutation such that no two points lie within the same bin. A random perturbation is then applied to each point thus preventing it lying at the center of each bin. Utilizing a random Latin hypercube distributes the particles more evenly throughout the space when compared to a more traditional random initialization of points. With particle positions initialized each particle is then given a random velocity within the limits of $\boldsymbol{V}_{max}$.

The objective function of each particle is next evaluated as normal but based on this objective function a member of the population is selected for refinement via a local optimization, in this case Sequential Quadratic Programming (SQP). Unlike other hybrid swarms the $\boldsymbol{x}_{Gbest}$ member is not automatically selected for local improvement, instead the previously calculated objective function is used in a rank selection scheme, similar to that used by a genetic algorithm for the purposes of generating a mating pool.

The use of such a scheme results in a chance that some measure of local improvement is applied to any member of the swarm and not just the global best point. This adds an additional measure of local exploration to the population without the expense of a large number of local searches in each generation. Moreover, this local optimization is also not carried out to convergence. Instead a small proportion of the total budget of function evaluations available for each generation is reserved for local improvement. Depending on the magnitude of this proportion the local improvement may be anything from a single to multiple steps towards a local optimum. A similar scheme was employed by Fan *et al.* (2004) where a Nelder-Mead simplex update was used to replace a single swarm member.

The combination of the rank based selection and partial convergence of a local optimization also helps to preserve a level of diversity within the population as the optimization progresses. As previously mentioned complete convergence of the global best point after the initial generation is counter-productive, as subsequent swarm generations tend to move towards that point rather than exploring the space effectively. Here, the view is taken that the effort of a completely converged local optimization should be spread throughout each swarm generation to boost the local exploitation of the whole population and not just the best point.

Once local improvement steps have been taken, the $\boldsymbol{x}_{Pbest}$ for each particle and overall $\boldsymbol{x}_{Gbest}$ can be calculated and used to update the velocity of each swarm member. Here, the constriction factor method of Eberhart and Shi (2000) is utilized with the resulting velocity restricted to $\boldsymbol{V}_{max}$. The position of each particle can then be updated as normal using Equation 23.

Provision is made to prevent particles moving beyond the permitted variable bounds. If the updated position of the particle falls outside the bounds of a variable then the position of that particle is adjusted so the particle lies on the boundary. The corresponding velocity of the particle is then reversed so the inertial component of the velocity update in the next iteration tends to move the particle back from the boundary. A similar approach was used by Huang and Mohan (2005).

A reinitialization of a proportion of the swarm is carried out before the calculation of the objective function of each particle. This reinitialization attempts to increase the diversity of the swarm population but unlike the reinitialization procedure of Wang *et al.* (2006) the position of any new particle is not random but selected in order to explore regions of the space not previously visited by the swarm or embedded local search. This is achieved through a maximization of the minimum distance of a proposed restart point to those points previously evaluated, a similar metric to that used by Morris and Mitchell (1995) to calculate optimal space filling Latin hypercubes.

A 2,000 point Latin hypercube sampling plan of potential new swarm members is generated upon each reinitialization and compared to the whole optimization history. The Euclidian distance between each point is calculated and the point with the maximum minimum distance selected as the new population member. It should be noted that it is not the goal of this reinitialization procedure to find a globally optimal point which maximises the minimum Euclidean distance but rather to locate an approximate region within the design space which has not been previously explored. An actual sub-optimization could be utilized in the reinitialization process but dependant on the method employed this may incur an additional overhead. Although not considered here, the size of the Latin hypercube could be adjusted to reflect the dimensionality of the problem considered, thereby increasing efficiency at lower dimensions.

The alternative option to a particle reinitialization procedure is a particle repulsion scheme such as that used by Riget and Vesterstrø (2002). In such a strategy when the

diversity of the population reaches a predefined minimum the velocity update equation is altered and particles move away from each other in subsequent iterations until diversity has been improved. However, such a repulsion technique offers no guarantees that previously unexplored regions will be searched by the swarm, rather the swarm may expand and contract again within a region of the space which may have been explored during the initial contraction. The repulsion process itself can take a number of generations and may therefore waste objective function evaluations. The strategy proposed here forces the exploration of unexplored regions of the space immediately.

After reinitialization the objective function of each particle is evaluated and a particle is selected for local improvement as before. The global and personal best locations can then be updated to calculate a particles velocity in the next generation. This process is repeated until the total number of generations is reached after which the global best point is used as the starting point for a terminal local search. The final local optimization is carried out with the aim of exploiting fully the best point found throughout all of the previous generations of the swarm.

While the proposed hybrid particle swarm can make effective use of available gradient information a capacity to deal with constraints and multi-objective problems has not been explicitly developed. Although the embedded local SQP optimization can effectively deal with linear and nonlinear equalities and inequalities, there is no provision for such problems within the explorative swarm. Penalty functions could be employed to aid with constraints or indeed an information sharing strategy relying on Pareto methods to handle constraints could be employed, Ray and Saini (2002). Such a method can be easily expanded to deal with multi-objective optimisation problems, Ray and Tsai (2004). The apparent drawbacks of the proposed algorithm are not a major issue as the likelihood optimization problem to which it is applied is a single objective optimisation problem with no such complex constraints.

5. Optimization of the Swarm Parameters

Having defined the basic hybrid hyperparameter optimization strategy, questions still remain concerning the appropriate settings for the hybrid swarm parameters in order to achieve optimal performance. The total number of generations, the size of the population, the point at which the local search should start to improve members of the population, the increase in the degree of local improvement with subsequent generations, the number of evaluations reserved for a terminal local search and the number of points being reinitialized, all require consideration.

One way of addressing all of these issues is to cast them in the form of an optimization problem. To this end the basic hybrid particle swarm strategy described previously has been parameterized using ten variables which adjust the overall structure of the swarm. These variables can then be adjusted over the course of an optimization with the objective of improving the hyperparameter tuning performance of the algorithm. A similar procedure was carried out by Keane (1995) when the optimization of the control parameters of a genetic algorithm were considered in order to improve performance on multi-modal problems.

Each of the ten swarm control parameters considered, (shown in Table 2) varies greatly in its effect on the particle swarm; the first five control the complete nature of the optimizer while the rest control more subtle features.

Given a predefined budget of likelihood evaluations, the first control parameter governs

Table 2. Hybrid particle swarm control parameters to be optimized.

No.	Parameter description	Lower limit	Upper limit
1	Fraction of total evaluation budget for terminal local optimization	0	1
2	No. of evaluations per generation	10	100
3	Generation swarm becomes hybridized	1^{st}	Final
4	Initial no. of local search evaluations (fraction of total evaluations)	0	0.5
5	Final swarm size (fraction of evaluations reserved for swarm)	0.1	1
6	Magnitude of $\boldsymbol{V}_{max}$	0	1
7	Initial probability of particle reinitialization	0	1
8	Final probability of particle reinitialization	0	1
9	Initial fraction of population reinitialized	0	1
10	Final fraction of population reinitialized	0	1

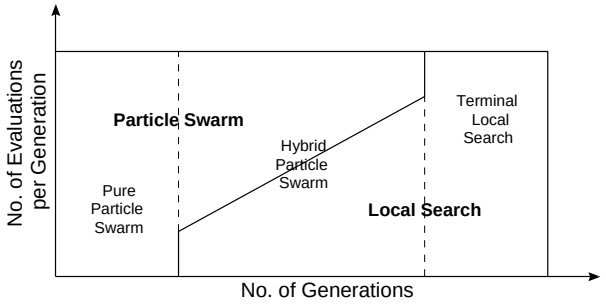

Figure 2. A graphical representation of the allocation of resources between the particle swarm and local search within the proposed hybrid optimization strategy

the fraction of this budget reserved for the terminal search. The second control parameter defines the size of the swarm population and therefore the total number of generations. The third control parameter governs the generation at which the swarm is hybridized, in essence, the generation at which a local search is first employed to improve a member of the population. The fourth and fifth parameters define how the number of evaluations per generation used in this local improvement changes as the optimization progresses.

The effect of these five parameters can be visualized using Figure 2. One can observe that adjusting the generation at which local improvement is utilized within the swarm converts the optimizer from the 'pure' swarm of Eberhart and Shi (2000) to a hybridized swarm. Likewise, adjusting the proportion of the total evaluation budget used in the terminal local search changes the optimizer from a purely global search to a purely local search from the best point of an initial design of experiments.

The fraction of a population's evaluation budget utilized in a local improvement alters linearly as the optimization progresses according to parameters four and five. The hybrid optimization can therefore move from one employing a small local improvement at each generation to one employing an increasing degree of local improvement as the optimization progresses. This therefore moves the optimization from a global exploration to a more localized exploitation.

The remaining five parameters control the particle swarm itself. The magnitude of $\boldsymbol{V}_{max}$ controls the incremental nature of the swarm and a particle's ability to escape from local optima. A large $\boldsymbol{V}_{max}$ causes all of the particles to move quickly around the space but perhaps overshooting regions of interest. A small $\boldsymbol{V}_{max}$ helps each particle to exploit regions of interest but may prevent them escaping local minima.

Table 3. Controlling parameters of the proposed hybrid particle swarm.

Pop. Size	No. Gens	V_{max}	Local Search Gen.	Min. Local Evals.	Min. Swarm Size	Prob of Reinitialization (Initial)	Prob of Reinitialization (Final)	No. of Points Reinitialized (Initial)	No. of Points Reinitialized (Final)
20	80	0.075	30	6	3	0.7	0.2	0.75	0.1

Parameters seven and eight control the probability that a particle reinitialization will occur as the optimization progresses. A random number is generated in each generation and compared to this probability: if this number is less than the probability then a reinitialization occurs. A large initial probability relative to final probability will therefore result in an intense exploration of the space, moving towards a more focused exploitation. A small initial probability on the other hand will result in the swarm progressing towards a more intense exploration in the final generations.

The remaining swarm control parameters control the fraction of the swarm population reinitialized as the optimization progresses. An optimization can therefore have no points reinitialized or indeed the whole population. Such an optimization strategy is an interesting prospect, as when coupled with a terminal local search, this essentially amounts to a space filling sampling of the likelihood space followed by a local optimization commenced from the best point found.

Before commencing any optimization of the swarm parameters it is necessary to first define a metric by which the performance of each potential hybrid swarm can be evaluated. The mean improvement in likelihood is here based on the difference in likelihood attained by the swarm tuning strategy to that attained by the Options MATLAB genetic algorithm, Keane (2003). For a given dimensionality the airfoil inverse design problem[1] outlined in Toal *et al.* (2008b) is sampled using a total of 50 different Latin hypercubes. Each sampling of the objective function is then used to construct a kriging model via an optimization of the likelihood. The likelihood achieved by a potential strategy is subtracted by that achieved by the baseline genetic algorithm, (given the same sampling plan and a budget of 5,000 likelihood evaluations), and the mean taken,

$$\frac{1}{50}\sum_{i=1}^{50}(\phi_i - \phi_{GA_i}). \tag{27}$$

As a likelihood optimization typically involves the minimization of the negative of the concentrated likelihood function, a negative value from the above equation indicates a better performing tuning strategy.

Using the mean improvement in likelihood as the objective function, a series of optimizations of the particle swarm parameters were carried out for a variety of problem complexities and for a range of likelihood evaluation budgets. Details of these optimizations are presented in Appendix A. Analysis of the results of the optimizations lead to the parameters presented in Table 3 being selected for the proposed hybridized particle swarm employing an equivalent of 2,000 likelihood evaluations.

The presented swarm control parameters result in a particle swarm which is skewed

[1]A NACA 0012 airfoil is modified through the addition of a series of Hicks-Henne bump functions to the upper and lower surface with the aim of recreating the pressure profile of the RAE-2822 airfoil at Mach 0.725 and at an angle of attack of 2°

Table 4. Description of the six hyperparameter optimizers used for comparison.

	Description
Heavy	A 5000 evaluation genetic algorithm followed by a 5000 evaluation dynamic hill climb
GA-DHC	A 2500 evaluation genetic algorithm followed by a 2500 evaluation dynamic hill climb
SA	A 5000 evaluation simulated annealing
DHC	A 5000 evaluation dynamic hill climb
PSO	A 5000 evaluation particle swarm optimization
SQP	A single sequential quadratic programming optimization commencing from a random starting point and running to convergence

towards an initial extensive exploration of the likelihood space with a high probability of reinitialization and high fraction of the population reinitialized to regions not previously explored. The local optimization commences once this exploration has reduced, after 30 generations, and begins with an initial effort equivalent to six likelihood function evaluations. A substantial fraction of the overall number of evaluations, 20%, is reserved for a terminal local search.

Essentially this results in a strategy somewhat similar to the dynamic hill climber of Yuret and Maza (1993). A local search is used to exploit regions of interest with exploration simultaneously occurring of previously unexplored regions via the reinitialization procedure and the particle swarm update process.

6. Performance of the Proposed Tuning Strategy

We next compare the performance of the optimized hybrid particle swarm algorithm to a selection of other methods, again using the likelihood metric and test problem described above. To do this we consider the performance of six other hyperparameter tuning strategies defined in Table 4. Three basic global optimizers, a basic particle swarm (PSO), a dynamic hill climber (DHC) and simulated annealing (SA), Kirkpatrick *et al.* (1983), are considered along with the 'Heavy' tuning strategy employed so successfully in the literature, (see Hoyle *et al.* (2006), Keane (2006) and Toal *et al.* (2008b)), a reduced cost variant of this and a simple sequential quadratic programming (SQP) search commenced from a random starting point.

The baseline genetic algorithm, basic particle swarm and simulated annealing algorithms all employ a population of 50 members for 100 generations. As the baseline genetic algorithm is restricted to a total of 5,000 likelihood evaluations so too are the particle swarm, dynamic hill climber and simulated annealing. The Heavy strategy however uses a total of 10,000 function evaluations, split evenly between the genetic algorithm and hill climber, and as such enjoys somewhat of an advantage over the other algorithms being tested. Hence, a second version of this strategy, denoted as 'GA-DHC' in Table 4, employing a total of 5,000 likelihood evaluations is also evaluated. The final tuning strategy considered involves a simple local search using a SQP algorithm commenced from a random starting point. This strategy is included as a contrast to the global optimizers with the intention of demonstrating the importance of the global optimization of the hyperparameters.

The mean improvement in likelihood of each of the methods is presented in Table 5 for each tuning strategy with the standard deviations of the improvement in likelihood presented in Table 6. In terms of the mean improvement in likelihood, the hybridized

Table 5. Comparison of the mean improvement in likelihood function for the hybridized swarm and six different hyperparameter optimizers over a range of problem sizes.

No. of Variables	Heavy	GA-DHC	SA	DHC	PSO	SQP	Hybrid PSO
2	-0.785	-0.509	-0.056	-0.929	-0.037	1.452	-0.0404
5	-1.734	-1.442	-0.347	-1.804	-0.789	5.319	-0.787
10	-3.685	-2.549	-0.860	-2.963	-0.656	5.448	-2.675
15	-3.169	-2.515	-0.090	-3.716	-1.558	4.925	-3.667
25	-5.452	-2.885	2.731	-2.639	-0.465	9.941	-6.911

Table 6. Comparison of the standard deviation of the improvement in likelihood function for the hybridized swarm and six different hyperparameter optimizers over a range of problem sizes.

No. of Variables	Heavy	GA-DHC	SA	DHC	PSO	SQP	Hybrid PSO
2	0.860	0.518	0.232	1.096	0.0953	1.388	0.087
5	1.848	1.777	1.113	2.087	1.992	4.084	1.933
10	3.329	2.905	3.750	3.558	3.784	4.350	3.366
15	2.641	2.639	3.228	3.321	4.160	4.440	4.031
25	3.388	2.478	4.655	5.763	5.316	6.960	4.497

particle swarm outperforms simulated annealing on the 5, 10, 15 and 25 variable problems and the particle swarm and GA-DHC strategy on the 10, 15 and 25 variable problem. The strategy even outperforms the Heavy strategy on the 15 and 25 variable problems which is particularly significant given that the hybridized particle swarm uses the equivalent of only 2,000 likelihood evaluations.

In terms of the standard deviation of the improvement in likelihood, the hybridized particle swarm is more consistent than the basic particle swarm upon which it is based, especially at higher dimensions. However, on the problems considered the hybrid swarm does not approach the consistency in likelihood achieved by the 'Heavy' strategy. This is perhaps rather unsurprising given the 'Heavy' strategy's much larger evaluation budget. Compared to the other global tuning strategies the hybridized particle swarm performs very well at high dimensions but for a significant reduction in the number of likelihood evaluations, 40% of those used by the GA-DHC, PSO and SA strategies and 20% of those used by the 'Heavy' strategy.

Note also that the simple SQP hyperparameter optimization performs consistently badly on all cases, with the performance reducing rapidly as problem dimensionality increases. This provides a clear demonstration of the advantages of a global strategy when attempting to optimize the likelihood.

Although Table 5 compares the actual minimum likelihood found by each algorithm it is the relationship of the corresponding hyperparameters to the quality of an overall optimization process which is of most interest. In other words, does a better likelihood equate to a better final design? We now consider a second metric by which to measure the performance of the hybrid particle swarm tuning strategy. The inverse design optimization of Toal *et al.* (2008b) is adopted once again but with the optimization now completed using a predefined budget of design function evaluations.

Unlike the similar comparisons of Toal *et al.* (2008b), where a fixed budget was applied to a series of problems of increasing dimensionality, here each optimization employs a total of $15 \times d$ evaluations. An inverse design problem of 15 variables will therefore have a total budget of 225 evaluations. This removes the problems associated with a fixed

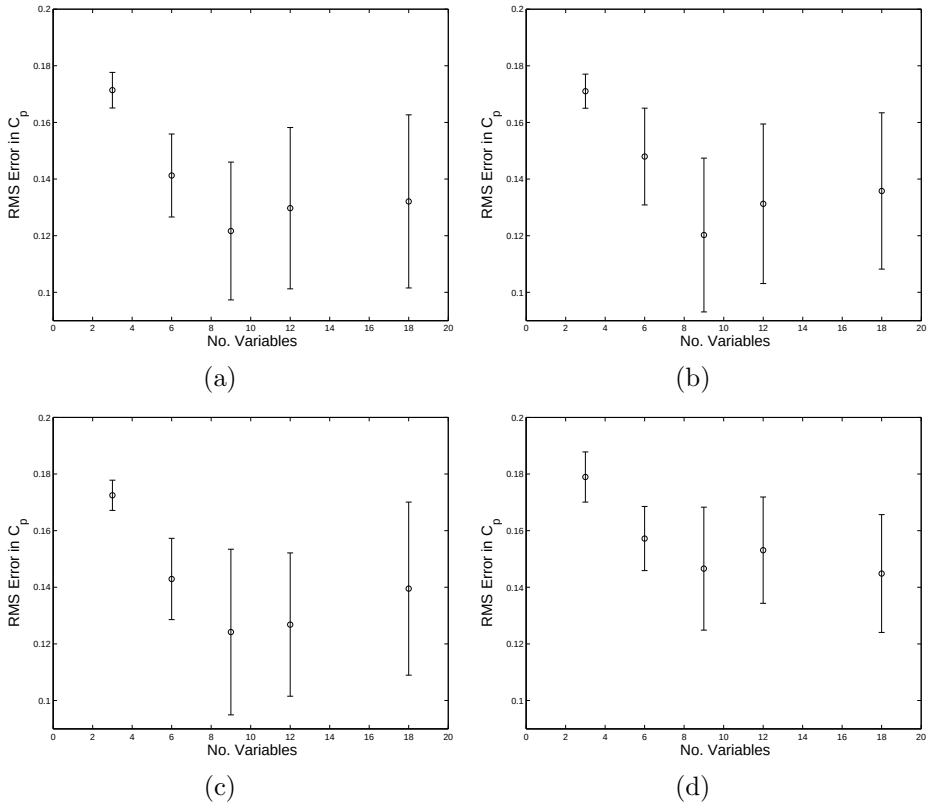

Figure 3. The effect of the 'Heavy' (a), Hybrid PSO (b), 2000 evaluation GA-DHC (c) and SQP (d) tuning strategies on a complete kriging based optimization.

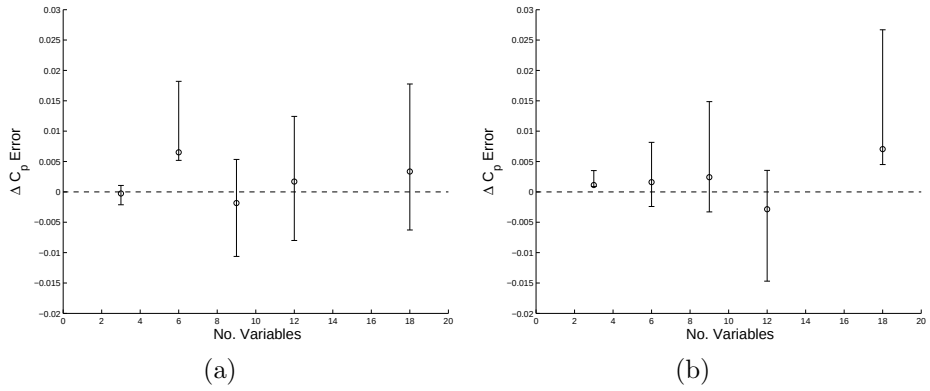

Figure 4. Bootstrapped mean and 95% confidence intervals for the Hybrid PSO (a) and 2000 evaluation GA-DHC (b) showing the improvement in root mean squared error obtained by a kriging based optimization to that obtained when employing the baseline 'Heavy' strategy.

simulation budget and an increasing dimensionality and provides a clearer indication of the effect of the tuning strategy. Of the total evaluation budget, one third is used in the initial design of experiments.

Figure 3 displays the average RMS error in pressure obtained by the final airfoil of the inverse design test problem when the hyperparameters are tuned via one of four strategies, the 'Heavy' strategy, the hybridized particle swarm a reduced cost GA-DHC

strategy employing 2,000 likelihood evaluations and the SQP strategy. As with the calculation of the mean improvement in likelihood, each optimization is carried out a total of 50 times and the average taken.

Each of the optimizations of Figure 3 commence from a series of identical design of experiments. The results of each optimization can therefore be directly compared to the corresponding optimization employing the 'Heavy' tuning strategy. Figure 4(a), for example, presents the mean difference between the best RMS error found using Hybrid PSO and the 'Heavy' strategies. Each of the presented means and 95% confidence intervals have been calculated via bootstrapping. Negative values indicate that a strategy obtains better airfoil designs as the best RMS error found by the 'Heavy' strategy is subtracted from that obtained by the other strategies. The data used in the construction of Figures 3 and 4 is presented in Appendix B.

The results presented in Figures 3 and 4 indicate that the performance of the hybridized particle swarm tuning strategy is comparable to that of the 'Heavy' tuning strategy and is capable of achieving similar final designs but for a considerable reduction in overall tuning cost. The 2,000 evaluation GA-DHC strategy is included in Figures 3(c) and 4(b) as a direct comparison to the hybridized particle swarm in terms of the total number of likelihood evaluations. This strategy is the equivalent of the 'Light' tuning strategy presented by Toal *et al.* (2008b) and employs an initial 1,000 evaluation genetic algorithm followed by a 1,000 evaluation dynamic hill climber. As supported by the results of Toal *et al.* (2008b) this strategy results in optimizations of broadly similar performance to that of the Heavy strategy until higher dimensional problems are encountered. The results for the 18 variable problem indicate a reduction in the quality of the final designs. The hybridized swarm on the other hand performs slightly better at higher dimensions as indicated in 4(a). Based on the presented results the hybrid swarm could be considered to make more efficient use of the available budget of concentrated likelihood evaluations than the equivalent cost GA-DHC strategy.

The results presented in Figure 3(d) for the SQP strategy indicate that a poorly optimized set of hyperparameters translates to a drop in overall optimization performance. Commencing a local search from a random point in the likelihood space can lead to the optimization being trapped in a plateau or converging to unrealistic hyperparameters. Often the 'optimum' hyperparameters returned by the SQP search contain a vector of p values equal to one, a vector of θ values equal to 3 or a regression constant of 3. Such values lead to completely unrealistic response surfaces and the poor optimization performance observed in Figure 3(d).

7. Airfoil Optimization

Finally, we consider the application of the proposed hybrid tuning strategy to an optimization completely unrelated to the inverse design problem considered so far in this paper. The RAE-2822 airfoil is optimized for minimum drag at a fixed lift coefficient of 0.3 at Mach 0.65 and a Reynolds number of 6×10^6. The airfoil is parameterized via two non-uniform rational B-splines (NURBS) curves, one representing the upper and one the lower surface, as shown in Figure 5. Fourteen control points are permitted to vary in both the x and y directions while the control points on the leading edge are fixed in the x direction resulting in a total of 30 variables.

The optimization is permitted a total of 450 objective function evaluations, of which 150 form the initial design of experiments (DOE). A typical function evaluation requires

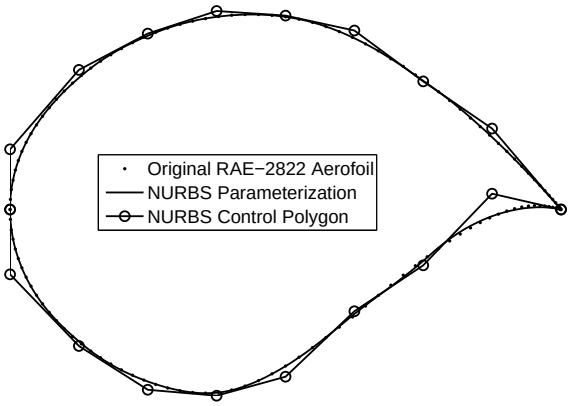

Figure 5. Thirty variable NURBS parameterization of the RAE-2822 airfoil.

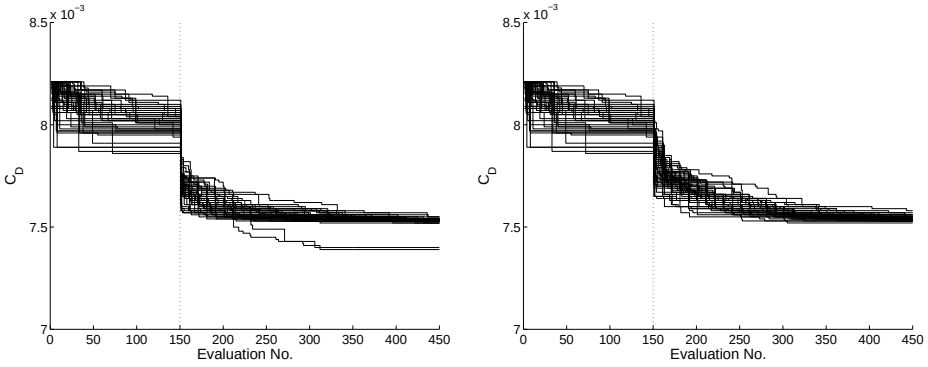

Figure 6. Optimization search histories resulting from a minimization of drag for fixed lift using the Hybrid (a) and the reduced cost GA-DHC (b) tuning strategies.

Table 7. Comparison of the quality of the designs resulting from the optimization of the RAE-2822 using two different tuning strategies.

Tuning Strategy	Average C_D	Standard Deviation C_D	% Improvement Over RAE-2822
Hybrid	7.53×10^{-3}	2.88×10^{-5}	8.29%
GA-DHC	7.54×10^{-3}	2.04×10^{-5}	8.13%

between three and four Viscous Garabedian and Korn (VGK) simulations at different angles of attack to achieve the required coefficient of lift, usually to within ± 0.005. A total of 30 batches of 10 updates are added to the model as the optimization progresses with the hyperparameters tuned after every update using either the hybrid particle swarm or the reduced cost GA-DHC strategy. An identical DOE is used in each case thereby providing a meaningful comparison between the final results. A total of 50 optimizations have been carried out for each case with the DOE varying each time, the optimization histories for which are presented in Figure 6 with the average drag of the final best designs presented in Table 7.

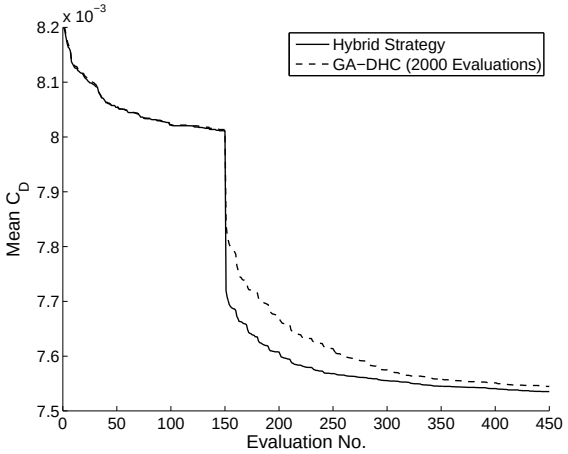

Figure 7. Average convergence of the kriging based optimizations based on the hybrid and reduced cost GA-DHC tuning strategies.

The results of Table 7 indicate that there is little to choose between the two strategies in terms of the quality of the final airfoil designs. Both the GA-DHC and the hybrid strategy achieve designs with an average drag coefficient of just over 7.5×10^{-3}. The results indicate that the hybrid strategy performs slightly better but this could be due to the effect of the two outlying designs observed in Figure 6(a) on the overall averages. Likewise the standard deviation associated with this strategy is elevated due to these designs. Both outlying designs exhibit a rapid and non-physical pressure oscillation close to the leading edge on the upper surface. This suggests that the simulation may be inaccurate and that the resulting drag coefficient cannot be relied upon. These optimizations have therefore exploited a weakness in the computational simulation and in doing so have achieved a false optimum. This represents a significant danger when any complex computer code is blindly relied upon during an optimization.

Removing these outlying designs leads to the hybrid strategy achieving designs with an average C_D of 7.535×10^{-4} and a standard deviation of 7.71×10^{-6}. The reduced cost GA-DHC obtains a slightly higher average drag of 7.545×10^{-4} and standard deviation of 1.22×10^{-5}. Onc e again both optimizations obtain broadly similar designs with the hybrid strategy achieving very slightly better and more consistent designs than the reduced cost GA-DHC strategy.

This can be explained somewhat upon comparison of the optimization histories of Figure 6 and the average convergence histories of Figure 7. Here one can observe a more rapid reduction in the magnitude of the objective function following the DOE when the hyperparameters are tuned via the hybrid strategy. The optimization therefore converges more quickly to the approximate region of an optimal design. Subsequent updates to the model are therefore spent fine tuning the optimal solution resulting in the reduction in the variance between final designs. For this particular optimization problem the hybrid strategy achieves the average drag coefficient of the GA-DHC strategy after approximately 350 objective function evaluations. Although this optimization utilized relatively fast VGK simulations in the evaluation of the objective function, a reduction of approximately 100 simulations is still quite significant.

While demonstrated in isolation within this paper, the presented hybridised particle swarm algorithm could be easily combined with other tuning strategies, such as tuning after alternative updates to the model and geometric filtration, Toal (2009).

8. Conclusions

A hybrid particle swarm optimization algorithm has been developed which effectively utilizes an efficient adjoint of the kriging likelihood function. The final algorithm combines a basic particle swarm with an SQP search and a particle reinitialization procedure.

The final structure of the hybridized swarm was defined through a series of optimizations of the swarm's control parameters with the aim of maximizing the algorithm's performance with respect to likelihood optimization. The swarm parameters were optimized for problems of differing complexity and for different budgets of likelihood evaluations. The results of these optimizations demonstrate, not only how the performance of the algorithm changes as more effort is applied, but also how the very nature of the algorithm alters in order to make effective use of the available budget.

These optimizations lead to an algorithm which intensely explores the likelihood space at the beginning of the optimization and then commences short local exploitations of promising regions after a number of generations. This local exploitation increases with each generation and concludes with a terminal search. In essence the algorithm is similar in its operation to dynamic hill climbing but makes effective use of the available gradient information.

The hybrid strategy was demonstrated to perform well with respect to likelihood optimization compared to a genetic algorithm, simulated annealing, a traditional particle swarm optimization algorithm and two GA-DHC based strategies even though the hybrid swarm employed considerably fewer function evaluations.

A final 30 variable optimization, unrelated to the inverse design problem used in the development of the strategy, demonstrated both an improvement in the quality and consistency of the final designs obtained over a GA-DHC strategy of equivalent cost. The hybrid strategy also demonstrated a useful acceleration in the convergence of the optimization, achieving the final design of the reduced cost GA-DHC strategy with 100 function evaluations to spare.

Acknowledgements

The presented work was undertaken as part of an Airbus funded activity. The authors would like to thank Dr. G. Endicott, Dr. A. Forrester, Dr. A. Sóbester and Dr. I. Voutchkov of the University of Southampton for their advice and input.

References

Angeline, P., 1998. Evolutionary Optimization Versus Particle Swarm Optimization: Philosophy and Performance Differences. *In*: *Proceedings of the 7th International Conference on Evolutionary Programming VII*, 601–610.

Blasi, L. and Del Core, G., 2007. Particle Swarm Approach in Finding Optimum Aircraft. *Journal of Aircraft*, 44 (2), 679–682.

Brandstatter, B. and Baumgartner, U., 2002. Particle Swarm Optimization - Mass-Spring System Analogon. *IEEE Transactions on Magnetics*, 38 (2), 997–1000.

Broyden, C., 1970. The Convergence of a Class of Double-rank Minimization Algorithms. *Journal of the Institute of Mathematics and its Applications*, 6 (1), 76–90.

D'Angelo, S. and Minisci, E., 2005. Mulit-Objective Evolutionary Optimization of Sub-

sonic Airfoils by Kriging Approximation and Evolution Control. *In*: *2005 IEEE Congress on Evolutionary Computation*, Vol. 2, 1262–1267.

Dwyer, P. and M.S., M., 1948. Symbolic Matrix Derivatives. *The Annals of Mathematical Statistics*, 19 (4), 517–534.

Eberhart, R. and Kennedy, J., 1995. A New Optimizer Using Particle Swarm Theory. *In*: 6^{th} *International Symposium on Micro Machine and Human Science*, 39–43.

Eberhart, R. and Shi, Y., 2000. Comparing Inertia Weights and Constriction Factors in Particle Swarm Optimization. *In*: *IEEE Conference on Evolutionary Computation*, 84–88.

Fan, S., Liang, Y., and Zahara, E., 2004. Hybrid Simplex Search and Part Swarm Optimization for Global Optimization of Multimodal Functions. *Engineering Optimization*, 36 (4), 401–418.

Forrester, A., Bressloff, N., and Keane, A., 2006. Optimization Using Surrogate Models and Partially Converged Computational Fluid Dynamics Simulations. *Proceedings of the Royal Society A*, 462 (2071), 2177–2204.

Giles, M., 2008. Collected matrix derivative results for forward and reverse mode algorithmic differentiation. *Lecture Notes in Computational Science and Engineering*, 64, 35–44.

Griewank, A., 2000. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. Society for Industrial and Applied Mathematics.

Guo, Q., Yu, H., and Xu, A., 2006. A Hybrid PSO-GD Based Intelligent Method for Machine Diagnosis. *Digital Signal Processing*, 16 (4), 402–418.

Heppner, F. and Grenander, U., 1990. *A Stochastic Nonlinear Model for Coordinated Bird Flocks*. AAAS Publications.

Hoyle, N., Bressloff, N., and Keane, A., 2006. Design Optimization of a Two-Dimensional Subsonic Engine Air Intake. *AIAA Journal*, 44 (11), 2672–2681.

Huang, T. and Mohan, A., 2005. Significance of neighborhood topologies for the reconstruction of microwave images using particle swarm optimization. *In*: *2005 Asia-Pacific Microwave Conference*, Suzhou, China.

Izui, K., Nishiwaki, S., and Yoshimura, M., 2007. Swarm Algorithms for Single- and Mulit-Objective Optimization Problems Incorporating Sensitivity Analysis. *Engineering Optimization*, 39 (8), 981–998.

Jin, R., Chen, W., and Simpson, T., 2001. Comparitive Studies of Metamodelling Techniques Under Multiple Modelling Criteria. *Structural and Multidisciplinary Optimization*, 23 (1), 1–13.

Jones, D., 2001. A Taxonomy of Global Optimization Methods Based on Response Surfaces. *Journal of Global Optimization*, 21 (4), 345–383.

Jones, D., Schonlau, M., and Welch, W., 1998. Efficient Global Optimization of Expensive Black-Box Functions. *Journal of Global Optimization*, 13 (4), 455–492.

Keane, A., 1995. Genetic algorithm optimization of multi-peak problems: Studies in convergence and robustness. *Artificial Intelligence in Engineering*, 9 (2), 75–83.

Keane, A., 2003. *The Options Design Exploration System: Reference Manual and User Guide (Version B3.1)*. http://www.soton.ac.uk/ ajk/options.ps.

Keane, A., 2006. Statistical Improvement Criteria for Use in Mulitobjective Design Optimization. *AIAA Journal*, 44 (4), 879–891.

Kennedy, J. and Eberhart, R., 1995. Particle Swarm Optimization. *In*: *IEEE Internation Conference on Neural Networks*.

Kirkpatrick, S., Gelatt, C., and Vecchi, M., 1983. Optimization by Simulated Annealing. *Science*, 220, 671–680.

Krige, D., 1951. A Statistical Approach to Some Basic Mine Valuation Problems on the Witwatersrand. *Journal of the Chemical, Metallurigical and Mining Engineering Society of South Affrica*, 52 (6), 119–139.

Leithead, W. and Zhang, Y., 2007. $O(N^2)$-Operation Approximation of Covariance Matrix Inverse in Gaussian Process Regression Based on Quasi-Newton BFGS Method. *Communications in Statistics - Simulation and Computation*, 36 (2), 367–380.

Liu, B., *et al.*, 2005. Improved Particle Swarm Optimization Combined with Chaos. *Chaos, Solitions and Fractals*, 25 (5), 1261–1271.

Liu, W., *et al.*, 2006. A Hybrid Particle Swarm Optimization Algorithm for Predicting the Chaotic Time Series. *In*: *Proceedings of the 2006 IEEE International Conference on Mechatronics and Automation*, 2452–2458.

McKay, M., Conover, W., and Beckman, R., 1979. A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code. *Technometrics*, 21 (2), 239–245.

Mendes, R., Kennedy, J., and Neves, J., 2004. The fully informed particle swarm: Simpler, maybe better. *IEEE Transactions on Evolutionary Computation*, 8 (3), 204–210.

Morris, M. and Mitchell, T., 1995. Exploratory Designs for Computational Experiments. *Journal of Statistical Planning and Inference*, 43 (3), 381–402.

Ninomiya, H. and Zhang, Q., 2008. Particle with ability of local search swarm optimization: PALSO for training of feedforward neural networks. *In*: *2008 IEEE International Joint Conference on Neural Networks*, 3009–14.

Noewl, M. and Jannett, T., 2004. Simulation of a New Hybrid Particle Swarm Optimization Algorithm. *In*: *Proceedings of the* 36^{th} *Southeastern Symposium on System Theory*, 150–153.

Park, J. and Baek, J., 2001. Efficient Computation of Maximum Likelihood Estimators in a Spatial Linear Model with Power Exponential Covariogram. *Computers & Geosciences*, 27 (1), 1–7.

Queipo, N., *et al.*, 2005. Surrogate-based analysis and optimization. *Progress in Aerospace Sciences*, 41, 1–28.

Ray, T. and Saini, P., 2002. Engineering design optimization Using a Swarm with an Intelligent Information Sharing Among Individuals. *Engineering Optimization*, 33 (6), 735–748.

Ray, T. and Tsai, H., 2004. Swarm Algorithm for Single and Mulit-Objective Airfoil Design Optimization. *AIAA Journal*, 42 (2), 366–373.

Riget, J. and Vesterstrø, M.J., A Diversity-Guided Particle Swarm Optimizer - the ARPSO. , 2002. , Technical report, EVALife Project Group.

Sacks, J., *et al.*, 1989. Design and Analysis of Computer Experiments. *Statistical Science*, 4 (4), 409–435.

Sakata, S., Ashida, F., and Zako, M., 2003. Structural Optimization Using Kriging Approximation. *Computer Methods in Applied Mechanics and Engineering*, 192 (7-9), 923–939.

Shi, Y. and Eberhart, R., 1998. Parameter Selection in Particle Swarm Optimization. *In*: *Evolutionary Programming VII*.

Shu-Kai, S., Yun-Chia, L., and Zahara, E., 2004. Hybrid Simplex Search and Particle Swarm Optimization for the Global Optimization of Multimodal Functions. *Engineering Optimization*, 36 (4), 401–418.

Simpson, T., Peplinski, J., and Kock, P., 2001. Metamodels for Computer-based Engineering Design: Survey and Recommendations. *Engineering with Computers*, 17 (2), 129–150.

Sóbester, A., Leary, S., and Keane, A., 2005. On the Design of Optimization Strategies Based on Global Response Surface Approximation Models. *Journal of Global Optimization*, 33 (1), 31–59.

Toal, D., 2009. *Proper Orthogonal Decomposition and Kriging Strategies for Design*. Ph.D Thesis.

Toal, D., Bressloff, N., and Keane, A., 2008a. Geometric Filtration Using POD for Aerodynamic Design Optimization. *In*: 26^{th} *AIAA Applied Aerodynamics Conference*.

Toal, D., Bressloff, N., and Keane, A., 2008b. Kriging Hyperparameter Tuning Strategies. *AIAA Journal*, 46 (5), 1240–1252.

Toal, D., *et al.*, 2009. An Adjoint For Likelihood Maximization. *Proceedings of the Royal Society A*, 465 (2111), 3267–3287.

Venter, G. and Sobieszczaznski-Sobieski, J., 2003. Particle Swarm Optimization. *AIAA Journal*, 41 (8), 1583–1589.

Victoire, T. and Jayakumar, A., 2004. Hybrid PSO-SQP for Economic Dispatch with Valve-Point Effect. *Electric Power Systems Research*, 71 (1), 51–59.

Wang, G. and Shan, S., 2007. Review of metamodeling techniques in support of engineering design optimization. *ASME Journal of Mechanical Design*, 129, 370–380.

Wang, Y., Zhang, J., and Zhang, Y., 2006. An Effective and Efficient Two Stage Algorithm for Global Optimization. *In*: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), v 3930 LNAI, Advances in Machine Learning and Cybernetics - 4^{th} International Conference, ICMLC 2005*, 487–496.

Yuret, D. and Maza, M., 1993. Dynamic Hill Climbing: Overcoming the Limitations of Optimization Techniques. *In*: *Proceedings of the 2^{nd} Turkish Symposium on AI and ANN*.

Zhang, Y. and Leithead, W., 2005. Exploiting Hessian Matrix and Trust-Region Algorithm in Hyperparameters Estimation of Gaussian Process. *Applied Mathematics and Computation*, 171 (2), 1264–1281.

Appendix A. Swarm Parameter Optimization

Each of the swarm control parameters, previously described in Table 2, were optimized using the classic response surface method. A kriging model was constructed from an initial 100 point DOE of the ten parameters. Twenty batches of up to twenty updates were then evaluated and added to the model. Throughout the course of the optimization the Heavy tuning strategy was employed in tuning the hyperparameters with the updates based on a search of the kriging models prediction of the performance of the swarm using a genetic algorithm.

A series of optimizations were carried out for different likelihood evaluation budgets and for underlying optimization problems of different complexities. A total of five different levels of complexity of the inverse design problem were considered from two to 25 variable problems. Naturally the number of hyperparameters that each potential swarm evaluates is double this. For each problem a number of different likelihood evaluation budgets were considered from 100 up to 2000 evaluations. These optimizations produce a Pareto front, Figure A1, demonstrating the performance of the hybrid strategy as more effort is applied to each optimization. Analyzing the results of each optimization therefore provides an interesting indication as to how the nature of the optimization should change as one moves from a quick to a slower more exhaustive search.

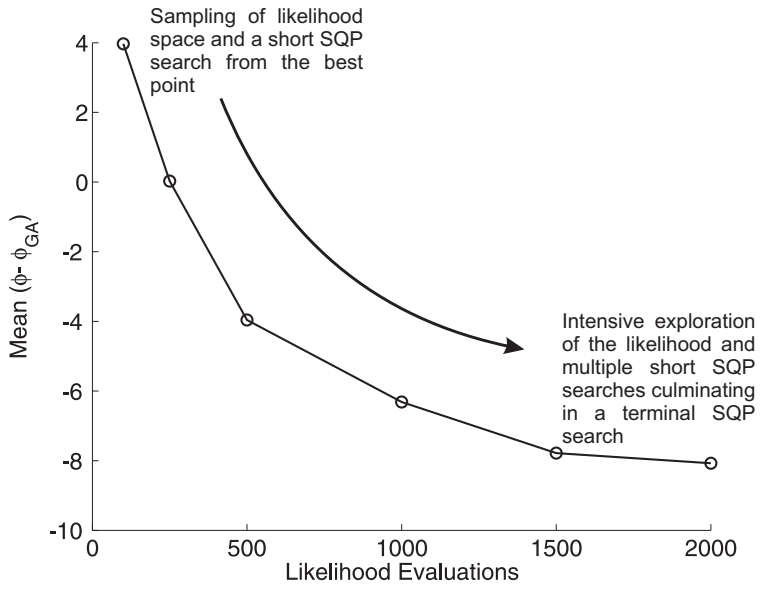

Figure A1. A Pareto front showing the change in performance and the nature of the optimization as the magnitude of the likelihood evaluation budget increases for the 25 variable optimization

Table A1. Parameters for the best performing hybrid swarm given a range of problem dimensionalities and a fixed budget of 100 evaluations.

No. of Variables	Population Size	No. of Generations	Objective Function
2	90	1	0.048
5	98	1	0.126
10	89	1	0.402
15	98	1	0.491
25	89	1	3.968

We consider first the results obtained by the swarm control parameter optimization when the likelihood evaluation budget is small. Table A1 presents the swarm control parameters for the best performing swarms on the five optimization problems considered. On all problems the optimization favors only a single generation of the swarm followed by a single SQP search, all of the remaining swarm parameters are irrelevant in such a case and are therefore not presented.

The results of Table A1 indicate that the optimization favors an initial sampling of the likelihood space using quite a large proportion of the available evaluation budget. This leaves a relatively small number of evaluations for a terminal SQP search commencing from the best point found in this initial sampling. The performance results presented in Table A1 do not compare favorably to those of the baseline genetic algorithm but fare much better than the SQP strategy of Table 5. This indicates that, in terms of likelihood optimization, the starting point of the local search is much more important than the effort applied to that local search.

Table A2 defines the best swarm parameters when a total of 2000 likelihood evaluations are employed in the hyperparameter optimization. Figure A1 demonstrates that after 1500 evaluations there is only a slight improvement in the magnitude of the likelihood compared to when 2000 evaluations are employed, we therefore assume that there

Table A2. Parameters for the best performing hybrid swarm given a range of problem dimensionalities and a fixed budget of 2000 evaluations.

No. of Vars.	Pop. Size	No. Gens	V_{max}	Local Search Gen.	Min. Local Evals.	Min. Swarm Size	Prob of Reinitialization (Initial)	Prob of Reinitialization (Final)	No. of Points Reinitialized (Initial)	No. of Points Reinitialized (Final)	Obj. Func.
2	32	50	0.810	39	3	17	0.05	0.50	0.80	0.52	-0.090
5	61	29	0.092	29	6	7	0.58	0.59	0.10	0.36	-1.115
10	63	25	0.045	15	1	50	0.44	0.63	0.84	0.005	-3.149
15	20	87	0.073	56	6	3	0.52	0.19	0.86	0.56	-4.038
25	15	87	0.064	22	6	3	0.82	0.27	0.43	0.11	-8.075

is no advantage to employing more than 2000 evaluations. The results of the these optimizations will therefore form the basis of a more general hybrid strategy.

Increasing the budget of available likelihood evaluations results in a significantly different optimization involving considerably more global exploration of the likelihood space. Table A2 demonstrates a much more effective use of the particle swarm with a considerable number of generations employed throughout the optimization.

Observe that the SQP search is not employed within the first generation but rather at some point in the middle of the optimization. This suggests that an initial search of the likelihood space via a particle swarm before hybridization is much more effective than employing a hybridized search from the start. The local search is therefore wasted if applied too soon before the swarm has found a promising region. This corresponds with the results of the SQP search of Figure 3(d) where the optimization frequently stalled on plateaus in the likelihood space. As a particle is selected for local improvement in the hybrid strategy via a rank based scheme, a poor particle may be selected and the same stalling may be observed at the beginning of the optimization. Commencing local searches part way through the optimization therefore reduces the chances of the optimization stalling and wasting valuable likelihood evaluations.

The results of the parameter optimizations indicate that an initial local search of approximately six likelihood evaluations is favorable. This makes perfect sense when one considers the cost of calculating the likelihood and its gradients is approximately twice that of a single likelihood evaluation. A minimum budget of six evaluations for the SQP search should therefore result in at least one step towards a local optimum.

The presented results also indicate a gradual increase in the level of effort expended in the local improvement of particles as the optimization progresses. Each optimization also concludes with a significant terminal search from the best point found so far by the optimization.

Reinitialization of points takes place throughout the course of all of the best performing optimizations. The results of Table A2 indicate that reinitialization tends to occur towards the start of the optimization with more points and the probability of reinitialization generally higher. This corresponds with the commencement of the local searches midway through the optimization. Once again the optimizer is attempting to rapidly explore the likelihood space before applying a local search. One must also note that 100% of the population is never reinitialized, this leaves some of the particles to gravitate towards the best current design point in the manner of a traditional particle swarm.

The results of Table A2, while indicating the general form that the final hybrid strategy should take, are quite different depending on the dimensionality of the underlying problem. The optimum hybrid swarm for the two variable problem is different to that for the 25 variable problem. A set of controlling swarm parameters which is optimal over all cases is therefore difficult to achieve. Instead emphasis is given to the more difficult cases,

namely the 15 and 25 variable problems. Based on the top ten performing strategies for both of these cases, the data for which can be found in Toal (2009), the hybrid strategy defined in Table 3 was selected.

Appendix B. Comparison of Optimization Results

Table B1. Details of the quality of the final designs resulting from the standard 'Heavy' tuning strategy and the SQP tuning strategy.

	Heavy Tuning Strategy			SQP Tuning Strategy		
No. of Variables	Mean Obj.	Standard Deviation	Tuning Time (hrs)	Mean Obj.	Standard Deviation	Tuning Time (hrs)
3	0.171	0.006	0.028	0.179	0.009	1.5×10^{-4}
6	0.141	0.015	0.298	0.157	0.011	0.003
9	0.122	0.024	1.246	0.147	0.022	0.031
12	0.130	0.029	3.366	0.153	0.019	0.11
18	0.132	0.031	14.054	0.145	0.021	0.69

Table B2. Details of the quality of the final designs resulting from the developed hybrid particle swarm tuning strategy.

	GA-DHC (2000 Evaluations)			Hybrid Swarm Tuning Strategy		
No. of Variables	Mean Obj.	Standard Deviation	Tuning Time (hrs)	Mean Obj.	Standard Deviation	Tuning Time (hrs)
3	0.1725	0.0050	0.005	0.1710	0.0060	0.005
6	0.1429	0.0144	0.060	0.1480	0.0171	0.060
9	0.1242	0.0293	0.249	0.1202	0.0271	0.249
12	0.1268	0.0253	0.673	0.1313	0.0281	0.673
18	0.1395	0.0306	2.811	0.1358	0.0276	2.811

Table B3. Bootstrapped means and 95% confidence limits for the improvement in design over the baseline 'Heavy' tuning strategy.

	GA-DHC (2000 Evaluations)			Hybrid Swarm Tuning Strategy		
No. of Vars	Mean ΔC_p	Upper 95% Limit	Lower 95% Limit	Mean ΔC_p	Upper 95% Limit	Lower 95% Limit
3	1.13×10^{-3}	2.37×10^{-3}	-2.40×10^{-4}	-2.60×10^{-4}	1.32×10^{-3}	-1.86×10^{-3}
6	1.61×10^{-3}	6.55×10^{-3}	-4.02×10^{-3}	6.51×10^{-3}	1.17×10^{-2}	1.32×10^{-3}
9	2.42×10^{-3}	1.24×10^{-2}	-5.71×10^{-3}	-1.83×10^{-3}	7.18×10^{-3}	-8.79×10^{-3}
12	-2.85×10^{-3}	6.40×10^{-3}	-1.18×10^{-2}	1.70×10^{-3}	1.07×10^{-2}	-9.70×10^{-3}
18	7.05×10^{-3}	1.96×10^{-2}	-2.54×10^{-3}	3.36×10^{-3}	1.44×10^{-2}	-9.63×10^{-3}