Engineering with Computers (2016) 32:377–404
DOI 10.1007/s00366-015-0421-2

ORIGINAL ARTICLE

A study into the potential of GPUs for the efficient construction and evaluation of Kriging models

David J. J. Toal[1]

Received: 11 February 2015 / Accepted: 8 September 2015 / Published online: 19 September 2015

Abstract The surrogate modelling technique known as Kriging, and its various derivatives, requires an optimization process to effectively determine the model's defining parameters. This optimization typically involves the maximisation of a likelihood function which requires the construction and inversion of a correlation matrix dependent on the selected modelling parameters. The construction of such models in high dimensions and with a large numbers of sample points can, therefore, be considerably expensive. Similarly, once such a model has been constructed the evaluation of the predictor, error and other related design and model improvement criteria can also be costly. The following paper investigates the potential for graphical processing units to be used to accelerate the evaluation of the Kriging likelihood, predictor and error functions. Five different Kriging formulations are considered including, ordinary, universal, non-stationary, gradient-enhanced and multi-fidelity Kriging. Other key contributions include the derivation of the adjoint of the likelihood function for a fully and partially gradient-enhanced Kriging model as well as the presentation of novel schemes to accelerate the likelihood optimization via a mixture of single and double precision calculations and by automatically selecting the best hardware to perform the evaluations on.

Keywords Kriging · Surrogate modelling · Meta-models · GPU

✉ David J. J. Toal
djjt@soton.ac.uk

[1] Faculty of Engineering and the Environment, University of Southampton, Southampton SO16 7QF, UK

1 Introduction

The application of surrogate models, meta-models or response surfaces to the prediction of the response of an expensive black box function has grown in popularity over the past 15 years [15, 49, 56, 59]. By emulating the output of a costly simulation or experiment these techniques have the potential to reduce the cost of design optimizations and sensitivity studies and provide fast and accurate conceptual design tools.

Of the many methods which can be employed to construct these surrogates, Kriging [32, 46] is one of the most popular due to the accuracy of the response and the useful update metrics based on the model's error prediction. However, the construction of a Kriging model requires the selection of a number of modelling parameters controlling the degree of regression, rate of correlation change and the smoothness of the response. Typically the selection of these parameters involves some form of maximisation of the Kriging likelihood function which can be costly for a number of reasons. The construction and inversion of the necessary correlation matrix can be expensive while the presence of multiple minima necessitates the use of a global optimization algorithm thereby requiring many evaluations of the likelihood function to ensure that a global optimum is attained.

This issue is exacerbated further as the number of sample points used to construct the Kriging model increases. This causes the size of the correlation matrix to increase which increases the expense of both the correlation matrix construction and inversion. Extensions of ordinary Kriging to cope with non-stationary responses [60], multi-fidelity data [31] and gradients [34], further increase the cost of calculating the likelihood. The construction of a non-stationary Kriging model requires the remapping of the original design space to one which a stationary Kriging model can better represent. Multi-fidelity Kriging models can include large amounts of data if the low-fidelity black box function is particularly cheap to evaluate while the size of the correlation matrix for a gradient-enhanced model increases rapidly with the number of dimensions.

Of course, once a Kriging model has been constructed evaluating the model's predictor or error functions can also be costly if used repeatedly within a global optimization to search for potentially good designs, as part of a sensitivity analysis where a Monte Carlo analysis is performed on the Kriging model or as part of some visualisation routine. Combining a Monte Carlo analysis within a global optimization, as would be the case in a robust design optimization [12], can be quite time consuming. As with the likelihood function, the cost of evaluating both the predictor and error function increases as the number of sample points or the complexity of the model increases. For all Kriging models both the predictor and error function require the calculation of the correlation of the unknown point and all of the points used to construct the model. The cost of constructing this vector of correlations increases with the number of sample points but also with the complexity of the model. Non-stationary Kriging, for example, requires the same non-linear mapping to be performed to the unknown point as to the points defining the model prior to the calculation of the correlation. Once the vector of correlations is constructed then a vector–vector multiplication is required to calculate the prediction or, in the case of the error, a matrix–vector followed by vector–vector multiplication is required both of which grow in cost with increasing sample size. In the case of a gradient-enhanced model, the cost also grows with increasing dimensions.

Previous work within the literature has attempted to tackle the cost of constructing Kriging models mainly by addressing the optimization problem used to select an appropriate set of modelling parameters. Toal et al., for example, first investigated the impact of varying optimization effort on the construction of Kriging models [50] and then developed an adjoint of the Kriging likelihood function [52] and an efficient hybridised particle swarm algorithm to employ this adjoint [51]. Zhang and Leithead developed an analytical Hessian of the likelihood function [63] and then a reduced cost approximation of the inverse of the covariance matrix [35] both of which were employed within local optimization algorithms.

The following paper approaches the problem of costly evaluations of the likelihood, prediction and error functions from the point of view of the hardware used to evaluate these functions. In particular, the following paper investigates the potential benefits of evaluating these functions on a graphical processing unit (GPU) for a variety of different Kriging models.

GPUs with their parallel processing prowess have become increasingly used to help accelerate the solution of a variety of problems in a number of fields such as engineering, physics and finance. Even modestly priced personal computers now come equipped with some form of GPU which, although typically used for playing games or other graphics intensive activities such as image processing or computer aided design, could easily be harnessed to accelerate traditional CPU-based activities. GPUs have already been used to great effect to accelerate computational fluid dynamics [3, 29], finite element [1, 40] and reaction [48] simulations. They have been used to accelerate topology optimizations [7, 62], uncertainty analyses [2, 45] and perform sensitivity studies [26]. Within the field of optimization algorithm development, genetic algorithms [24], simulated annealing [13, 61], ant colony searches [55], particle swarms [58] and tabu searches [10] have all been demonstrated to benefit from being run on a GPU. Within the field of surrogate modelling, artificial neural networks [39], self-organising maps [64], support vector machines [36, 37] and radial basis functions [4] have also benefited from the parallel processing capability offered by GPUs.

Recently GPUs have been applied to accelerate Kriging interpolation by Cheng [8], Demir and Westermann [11] and Gutiérrez de Ravé et al. [43]. While their work demonstrates very effectively the performance enhancements that GPUs can offer with respect to matrix multiplication, inversion and summation, there are a number of significant differences to the current work. The work of Cheng, Demir and Westermann and Gutiérrez de Ravé et al. employed a different formulation of Kriging compared to that used in the current paper. Here the method used previously in the literature by Sacks et al. [46] and Jones [27, 28] is employed which requires an optimization of the Kriging log-likelihood function in order to define the model parameters. As the likelihood function is not considered within the work of Cheng, Demir and Westermann and Gutiérrez de Ravé et al. the expense of computing its derivative on a GPU has also not been considered. While employing large sample sizes Cheng and Gutiérrez de Ravé et al. also tended to consider problems with relatively few dimensions whereas the optimization literature regularly applies Kriging to problems with over 10 variables. Cheng, Demir and Westermann and Gutiérrez de Ravé et al. also concentrated on the two most common forms of Kriging and neglected its non-stationary, multi-fidelity and gradient-enhanced variants as well as the model's predicted error.

The following paper, therefore, investigates the efficiency of evaluating the log-likelihood, predictor and error function on a GPU for each of the Kriging variants noted

Table 1 Hardware overview

Name	Information
i7-2860QM	4 cores, processor core clock 2.5 GHz & 16 GB DDR3 at 1600 MHz
Nvidia Quadro 2000M	192 CUDA cores, processor core clock 550 MHz and 2 GB GDDR3 RAM at 900 MHz
Nvidia Tesla K20C	2496 CUDA cores, processor core clock 706 MHz and 5 GB GDDR5 RAM at 2.6 GHz

above. For each case CPU and GPU versions of the functions are presented and compared for a variety of sampling plan sizes and problem dimensionalities. In addition to this the manner in which the functions are coded to take as much advantage as possible of efficient matrix and vector operations is presented and compared to two freely available toolboxes within the literature. These results provide an indication of the level of performance improvement offered by a GPU implementation of each Kriging model and useful coding tips for the development of similar functions. With the performance advantage established the paper then investigates two novel ways in which the general process of Kriging parameter optimization can be accelerated further, through the application of a mixture of single and double precision calculations and by automatically switching between hardware.

Both the CPU and GPU versions of all of the functions presented are coded using Matlab and its inbuilt GPU toolbox. Matlab is used in this case as it provides a rapid means of prototyping all of the functions and offers seamless integration with the Rolls-Royce proprietary optimization suite OPTIMATv2 [30, 51, 52, 54, 57] which is itself written in Matlab. Using Matlab also allows the programs to make use of the simple and efficient way Matlab has of handling the transfer of data between main memory and that of the GPU. It should be noted there are other languages that offer an interface to a GPU other than Matlab and indeed there are a variety of different libraries available for linear algebra operations such as, cuBLAS[1], MAGMA[2], CULA[3] and LibSciACC. While further gains in performance may be obtained over those presented if these libraries were employed the comparison of each of these different libraries and combinations of individual functions from separate libraries is deemed beyond the scope of the current investigation. Similarly, those functions running on the CPU could be written wholly or partially in a variety of languages and rather than comparing the efficiency of all of these only Matlab implementations will be considered.

The following paper commences by assessing the application of a GPU to some of the fundamental mathematical operations involved in the subsequent calculations of the likelihood, predictor and error functions. The paper then moves on to investigate the most basic form of Kriging, that of ordinary Kriging. The formulation of the likelihood function along with its adjoint and its corresponding predictor and error functions are presented. The efficient coding of each of these functions in Matlab is then discussed and the efficiency of CPU and GPU versions of these functions compared. This process of presenting the mathematics of the likelihood, it's adjoint and the predictor and error followed by a comparison of CPU and GPU implementations is repeated for universal Kriging, non-stationary Kriging, multi-fidelity Kriging and finally gradient-enhanced Kriging. The adjoint of a fully and partially gradient-enhanced Kriging likelihood function is presented here for the first time. The paper then proceeds to investigate efficiency of single precision calculations of the ordinary Kriging likelihood function and presents a novel mixed precision optimization strategy to reduce the cost of the hyperparameter optimization. Finally, the automated switching between CPU and GPU evaluations of the likelihood function prior to a hyperparameter optimization is considered.

2 Basic mathematical operations

As noted above, GPUs have been demonstrated to offer considerable performance improvements over traditional CPUs due to their parallel processing prowess. However, as with CPUs, GPUs come in a variety of different flavours. As such all of the calculations within this paper will be assessed using three different pieces of computational hardware, a mobile quad core CPU, a mobile GPU and a high end GPU details of which are presented in Table 1. Comparing the efficiency of all of the various operations using these three pieces of hardware provides an effective contrast of what can be achieved with a GPU on both limited and unlimited budgets. The Quadro 2000M, for example, is a very basic graphics card bundled with a laptop whereas the Tesla K20C is an extremely high end card developed especially for GPU-based supercomputers and is representative of the current cutting edge. The Tesla card, for example, has a much higher core clock speed, many more CUDA cores as well as more memory clocked at a higher

[1] https://developer.nvidia.com/cuBLAS.

[2] http://icl.cs.utk.edu/magma/.

[3] http://www.culatools.com/.

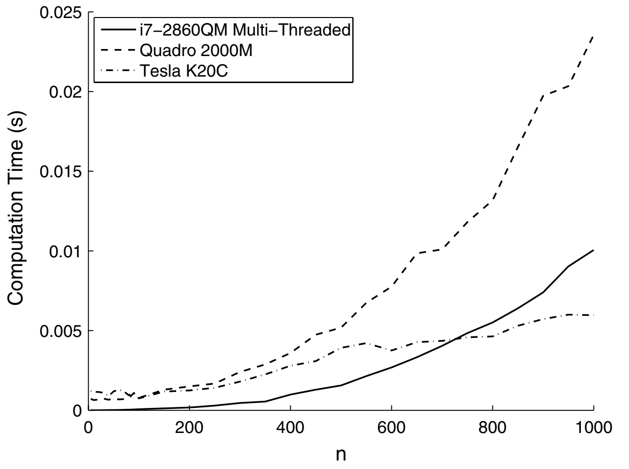

Fig. 1 Cost of performing a Cholesky decomposition with varying matrix size

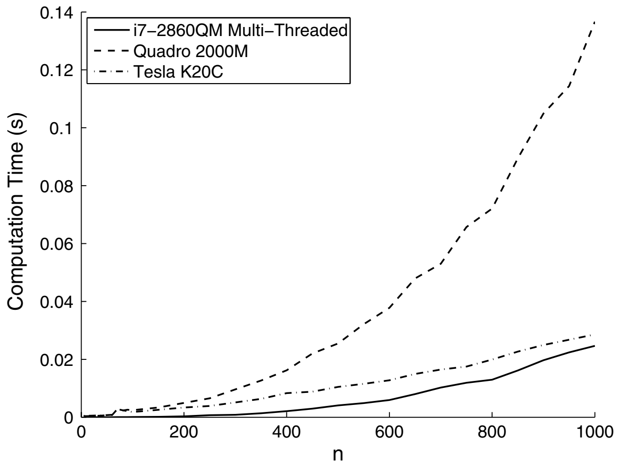

Fig. 2 Cost of performing a back substitution with varying matrix size

speed. Both the i7-2860QM CPU and Quadro 2000M are in the same laptop.

Before comparing the performance of CPU and GPU calculations of the likelihood, predictor and error functions let us compare the performance of both on a number of standard operations which are employed within the subsequent Kriging functions.

As noted above one of the major costs of evaluating the likelihood function is due to the inversion of the correlation matrix. In the following paper all such matrix inversions employ a Cholesky decomposition followed by a series of two back substitutions. Before considering the cost of a complete inversion let us consider each of these sub-operations in turn. Figure 1 presents a comparison of the cost of performing the Cholesky decomposition of a symmetric matrix of size n as n varies using the CPU and both GPUs. As can be observed, performing a Cholesky decomposition on the Quadro 2000M is considerably less efficient than the CPU. Likewise, over the majority of the range of n the CPU is considerably more efficient than the Tesla card. Only for large matrices, when $n > 700$ does the Tesla GPU offer any advantage.

Figure 2 compares the cost of performing a single back substitution. As before the Quadro card performs badly relative to the CPU. The Tesla card performs better but unlike with the Cholesky decomposition it never outperforms the CPU at any point over the range of n tested.

Combining the Cholesky decomposition and two back substitution operations together to calculate the inverse of a matrix, as illustrated in Fig. 3, it can be observed that the CPU is generally much more efficient although the Tesla card does begin to approach the performance of the CPU when $n = 1000$. Clearly the operations involved in the inversion of a matrix cannot be scaled effectively over the graphics card's multiple cores.

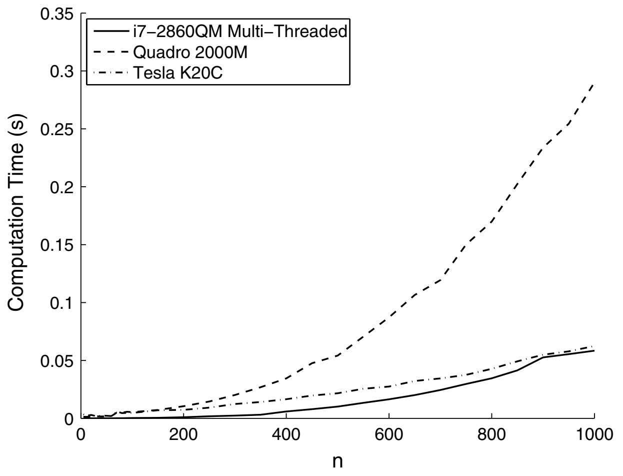

Fig. 3 Cost of performing a matrix inversion with varying matrix size

Of course, the inversion of a matrix is not the only operation performed regularly in the calculation of the likelihood, predictor or error functions. Figures 4, 5 and 6 compare the cost of, respectively, a matrix–matrix multiplication, a pointwise matrix–matrix multiplication and the pointwise multiplication of two three dimensional matrices. Unlike the matrix inversion the sub-operations involved in these three calculations are much more amenable to parallelization and the results of Figs. 4, 5 and 6 begin to illustrate the advantages of the many compute cores of a GPU. Matrix–matrix operations, are considerably faster on the Tesla card when $n > 150$ and the performance of the Tesla card scales much better with increasing n than the performance of the CPU. A similar trend is true for the Tesla card when performing pointwise matrix multiplications between both two and three dimensional matrices.

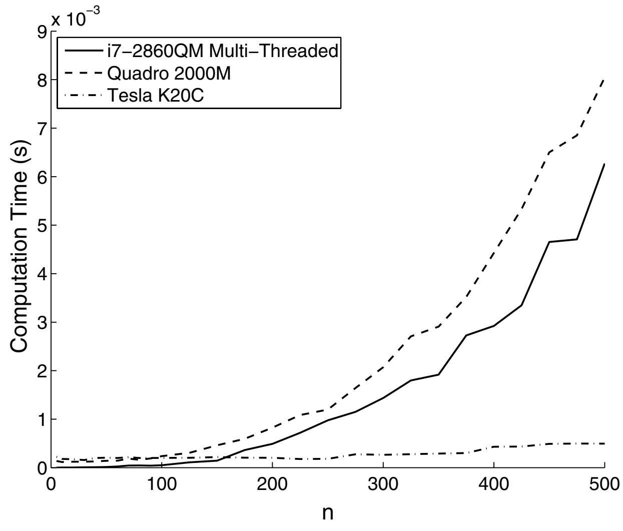

Fig. 4 Cost of performing a matrix–matrix multiplication with varying matrix size

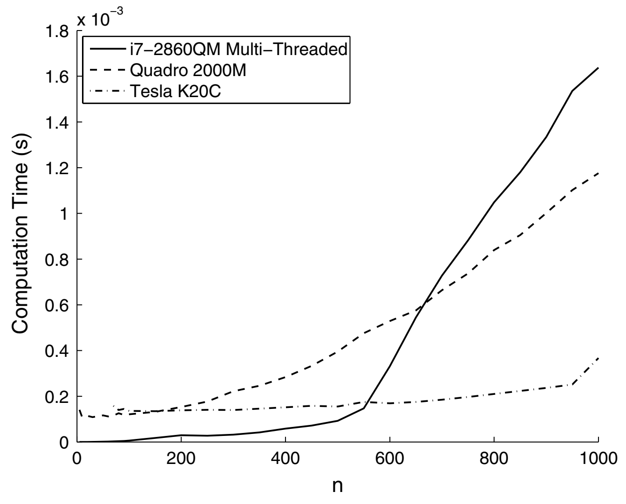

Fig. 5 Cost of performing a pointwise matrix–matrix multiplication with varying matrix size

Even with its additional cores the Quadro card does not perform as well as the CPU when performing matrix multiplications, however, it is much more efficient when performing pointwise multiplications between 3D matrices and outperforms the CPU when performing pointwise multiplications between 2D matrices when $n > 700$.

The above results, while illustrating both the advantages and disadvantages of a GPU, offer an insight into the results which can be expected when we compare the cost of calculating the likelihood, predictor and error functions. As will be presented in the following sections the calculation of the likelihood generally involves a series of operations to construct a matrix, invert the matrix and then calculate

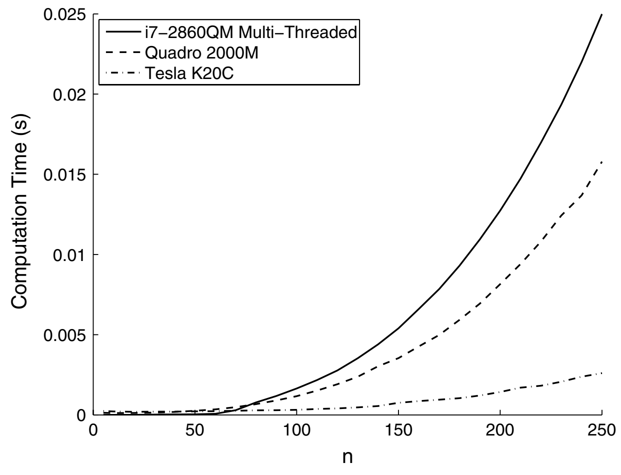

Fig. 6 Cost of performing a 3D pointwise matrix–matrix multiplication with varying matrix size

the adjoint and subsequent partial derivatives. Given that the matrix inversion when $n < 1000$ is faster on the CPU one would, therefore, expect the GPU to begin to out perform the CPU in the calculation of the complete likelihood when the other operations offer a greater cost saving than the cost penalty of inverting the matrix on the GPU. Any performance gain should also improve with increasing problem dimensionality as the size and, therefore, cost of the matrix inversion will stay constant but the cost of the other operations, such as the pointwise matrix multiplications, will scale better if performed on a GPU. With this in mind let us now consider the application of both GPUs to ordinary Kriging.

3 Ordinary Kriging

3.1 Ordinary Kriging formulation

Of the five different formulations of Kriging considered within the current paper ordinary Kriging is perhaps the simplest and forms the basis upon which all of the other models are derived. Popularised by Sacks et al. [46] for the prediction of deterministic computer experiments, ordinary Kriging has been applied in a wide variety of engineering design and optimization problems.

The construction of a Kriging model assumes that when two points are close together in the design space their objective function values will be similar. This is modelled by assuming that the correlation between two points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ is given by,

$$\boldsymbol{R}_{ij} = \exp\left(-\sum_{l=1}^{d} 10^{\boldsymbol{\theta}^{(l)}} \|\boldsymbol{x}_i^{(l)} - \boldsymbol{x}_j^{(l)}\|^{\boldsymbol{p}^{(l)}}\right), \tag{1}$$

where $\boldsymbol{\theta}$ and $\boldsymbol{p}$ represent vectors of length d of the Kriging modelling parameters selected via a maximisation of the likelihood on the observed dataset, $\boldsymbol{y}$, which is given by [27],

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

with the maximum likelihood variance, $\hat{\sigma}^2$, and mean, $\hat{\mu}$, given by,

$$\hat{\sigma}^2 = \frac{1}{n}(\boldsymbol{y} - \mathbf{1}\hat{\mu})^T\boldsymbol{R}^{-1}(\boldsymbol{y} - \mathbf{1}\hat{\mu}), \tag{3}$$

and

$$\hat{\mu} = \frac{\mathbf{1}^T\boldsymbol{R}^{-1}\boldsymbol{y}}{\mathbf{1}^T\boldsymbol{R}^{-1}\mathbf{1}}, \tag{4}$$

respectively where $\mathbf{1}$ denotes a vector of ones equal in length to the number of sample points, n. As previously noted the modelling parameters $\boldsymbol{\theta}$, $\boldsymbol{p}$ and, if necessary, a regression constant [16], λ, are selected via a maximisation of the likelihood function (Eq. 2). In order to accelerate this optimization Toal et al. [51] developed an adjoint of the likelihood function based on the linear algebra results of Giles [20]. Throughout this paper the notation of Griewank [22] is employed to denote the adjoint of a variable using the bar symbol, the adjoint of correlation matrix $\boldsymbol{R}$, for example, is, therefore, denoted by $\bar{\boldsymbol{R}}$.

Commencing from an initial seeding for the adjoint of the concentrated log-likelihood of $\bar{\phi} = 1$ it can be shown that the adjoint of the correlation matrix is given by,

$$\bar{\boldsymbol{R}} = \frac{1}{2\hat{\sigma}^2}\boldsymbol{R}^{-T}(\boldsymbol{y} - \mathbf{1}\hat{\mu})^T(\boldsymbol{y} - \mathbf{1}\hat{\mu})^T\boldsymbol{R}^{-T} - \frac{1}{2}\boldsymbol{R}^{-T}, \tag{5}$$

with the derivative of the concentrated log-likelihood with respect to the modelling parameters θ and p then given by,

$$\frac{\partial\phi}{\partial\theta_k} = \ln 10\sum_{ij} -10^{\theta_k}|\boldsymbol{x}_k^{(i)} - \boldsymbol{x}_k^{(j)}|^{p_k}\boldsymbol{R}^{(i,j)}\bar{\boldsymbol{R}}^{(i,j)} \tag{6}$$

and

$$\frac{\partial\phi}{\partial p_k} = -\sum_{ij} 10^{\theta_k}|\boldsymbol{x}_k^{(i)} - \boldsymbol{x}_k^{(j)}|^{p_k} \times \ln|\boldsymbol{x}_k^{(i)} - \boldsymbol{x}_k^{(j)}|\boldsymbol{R}^{(i,j)}\bar{\boldsymbol{R}}^{(i,j)}, \tag{7}$$

respectively. If a regression constant, 10^{λ} has been added to the diagonal of the correlation matrix the derivative of the likelihood with respect to this constant is given by,

$$\frac{\partial\phi}{\partial\lambda} = 10^{\lambda}\sum_{i}\bar{\boldsymbol{R}}^{(i,i)}. \tag{8}$$

Employing this efficient formulation for the derivatives of the likelihood function the optimization of the modelling parameters can be accelerated. With an optimised set of parameters obtained the corresponding correlation matrix for the sample set and the vector of correlations, $\boldsymbol{r}$, between an unknown point, $\boldsymbol{x}^*$ and the known sample points can be constructed and used to calculate the prediction of the Kriging model [27],

$$y(\boldsymbol{x}^*) = \hat{\mu} + \boldsymbol{r}^T\boldsymbol{R}^{-1}(\boldsymbol{y} - \mathbf{1}\hat{\mu}). \tag{9}$$

As noted above a Kriging model provides a very useful prediction of the error in the model at an unsampled point,

$$s^2(\boldsymbol{x}^*) = \hat{\sigma}^2\left[1 - \boldsymbol{r}^T\boldsymbol{R}^{-1}\boldsymbol{r}\right], \tag{10}$$

which can be used to define regions of the space to include additional data in order to improve the global accuracy of the model. Both the Kriging predictor and the error functions play an important role in the calculation of a number of other very useful metrics. The probability of improvement at an unknown point, $P[I(\boldsymbol{x}^*)]$, which is calculated as,

$$P[I(\boldsymbol{x}^*)] = \frac{1}{2}\left[1 + \mathrm{erf}\left(\frac{y_{\min} - y(\boldsymbol{x}^*)}{s\sqrt{2}}\right)\right], \tag{11}$$

provides a measure of the probability that an unknown point will attain an objective function value lower than the current minimum $y_{\min}$. A slight modification to this formula also provides a metric which can be used to determine the probability of a point exceeding a constraint if the surrogate model is constructed from a sampling plan of constraint values. While the probability of improvement indicates where improvement in the objective function can be obtained it does not provide a measure of how big that improvement will be. Another popular metric, the expected improvement, $E[I(\boldsymbol{x}^*)]$ does just that and calculates the amount of improvement over the current best value that is expected and is given by,

$$E[I(\boldsymbol{x}^*)] = (y_{\min} - y(\boldsymbol{x}^*))\left[\frac{1}{2} + \frac{1}{2}\mathrm{erf}\left(\frac{y_{\min} - y(\boldsymbol{x}^*)}{s\sqrt{2}}\right)\right] + \frac{s}{\sqrt{2\pi}}\exp\left[\frac{-(y_{\min} - y(\boldsymbol{x}^*))^2}{2s^2}\right]. \tag{12}$$

It can be observed from Eqs. 11 and 12 that the predictor, Eq. 9, and error function, Eq. 10, are employed repeatedly in the calculation of the more “exotic” Kriging update criteria. The efficiency of the calculation of both the prediction and the error is, therefore, central to the efficiency of the calculation of the probability of improvement, probability of feasibility and expected improvement. The current paper will only investigate the application of a GPU with respect to improving the efficiency of the likelihood, prediction and error calculations as these functions with their many matrix multiplications, inversions and summations are far more costly relative to the few additional mathematical operations to calculate $P[I(\boldsymbol{x}^*)]$ or $E[I(\boldsymbol{x}^*)]$.

3.2 Likelihood evaluation comparison

The first operation in the calculation of the concentrated log-likelihood function for an ordinary Kriging model is the construction of the correlation matrix. There are a number of ways in which this matrix can be constructed. In their freely available Matlab surrogate modelling toolbox Forrester et al. [17], for example, employ a nested for loop to construct the upper triangular portion of $\boldsymbol{R}$ and then reflect this in the diagonal to form the lower part of the matrix. The Matlab DACE toolbox of Lophaven et al. [38] takes a slightly different approach with the correlation matrix being calculated in one operation from a predefined matrix of distances between the sample points. However, neither of these toolboxes calculate the adjoint of the likelihood function which requires storage of some of the intermediate values used in the calculation of $\boldsymbol{R}$ in order to improve the efficiency of the derivative calculation.

In a similar manner to DACE, the algorithm employed here pre-computes an $n \times n \times d$ 3D matrix of distances between all of the sample points where n is the number of sample points and d is the number of dimensions. In the case of ordinary Kriging these distances are independent of the modelling parameters and remain constant throughout the likelihood optimization. A similar process is used within all of the Kriging routines wherever possible to reduce the number of unnecessary repeated calculations. Within the likelihood calculation this matrix of differences is then combined with two further 3D matrices of repeated θ and p values to calculate the 3D matrix of $10^{\theta_k}|\boldsymbol{x}_k^{(i)} - \boldsymbol{x}_k^{(j)}|^{p_k}$ values which is necessary for the calculation of $\frac{\partial \phi}{\partial \theta}$ and $\frac{\partial \phi}{\partial p}$.

With the $10^{\theta_k}|\boldsymbol{x}_k^{(i)} - \boldsymbol{x}_k^{(j)}|^{p_k}$ values calculated $\boldsymbol{R}$ is simply a summation across the third dimension of the matrix followed by the exponent. A diagonal matrix of 10^{λ} values is then added to $\boldsymbol{R}$ to regress the model if required.

As with the toolboxes of Forrester et al. [17] and Lophaven et al. [38] the calculation of $\boldsymbol{R}$ is followed by a Cholesky decomposition. However, whereas these toolboxes use the resulting triangular matrix in all subsequent calculations the present algorithm uses this matrix to calculate and store $\boldsymbol{R}^{-1}$. As $\boldsymbol{R}^{-1}$ is required in the calculation of $\bar{\boldsymbol{R}}$ it is much more efficient to compute it once, store and reuse it to calculate the mean and variance than to use the Cholesky decomposition and then calculate the inverse anyway within the calculation of $\bar{\boldsymbol{R}}$.

The calculation of the mean, $\hat{\mu}$ is performed directly using the inverse of the correlation matrix and the same could be done for the variance, $\hat{\sigma}^2$. However, as indicated in Eq. 3 the variance calculation includes the calculation of $\boldsymbol{R}^{-1}(\boldsymbol{y} - \boldsymbol{1}\hat{\mu})$ which is also required in the calculation of $\bar{\boldsymbol{R}}$. Rather than calculating the variance in a single line the operation is split into two parts with $\boldsymbol{R}^{-1}(\boldsymbol{y} - \boldsymbol{1}\hat{\mu})$ calculated separately and stored for use in the adjoint calculation.

With all of the major components of Eq. 2 now defined the concentrated log-likelihood can be calculated and returned to the optimization algorithm if necessary. If, however, the gradients are also required the algorithm now proceeds with their calculation which is a considerable departure from the work of Forrester et al. and Lophaven et al.

The derivative calculation commences with the calculation of the adjoint of the correlation matrix $\bar{\boldsymbol{R}}$. Given that $\boldsymbol{R}^{-1}$ and $\boldsymbol{R}^{-1}(\boldsymbol{y} - \boldsymbol{1}\hat{\mu})$ have already been calculated and stored this reduces to a matrix multiplication, addition and pointwise division by $2\hat{\sigma}^2$. With $\bar{\boldsymbol{R}}$ calculated the derivative of the likelihood with respect to the regression constant, λ, can be easily calculated in a single operation. $\frac{\partial \phi}{\partial \theta_k}$ is also easily calculated using the stored 3D matrix of $10^{\theta_k} \mid \boldsymbol{x}_k^{(i)} - \boldsymbol{x}_k^{(j)}|^{p_k}$ values and a pointwise multiplication to the stored $\boldsymbol{R}\bar{\boldsymbol{R}}$ matrix which has been repeated out d times to form a 3D matrix. Each of the d layers of this 3D matrix can then be summated to define the vector of derivatives. The derivative with respect to p, $\frac{\partial \phi}{\partial p}$ involves a very similar calculation but with the addition of a 3D matrix of $\ln|\boldsymbol{x}_k^{(i)} - \boldsymbol{x}_k^{(j)}|$ values precomputed along with the matrix of differences.

The calculation of ϕ and its derivatives, therefore, has a number of operations amenable to the parallel capabilities of a GPU, for example, the calculation of $10^{\theta_k} \mid \boldsymbol{x}_k^{(i)} - \boldsymbol{x}_k^{(j)} \mid^{p_k}$, $\boldsymbol{R}$, $\boldsymbol{R}^{-1}$, $\hat{\mu}$, $\boldsymbol{R}^{-1}(\boldsymbol{y} - \boldsymbol{1}\hat{\mu})$, $\hat{\sigma}^2$, $\bar{\boldsymbol{R}}$, $\frac{\partial \phi}{\partial \theta}$, $\frac{\partial \phi}{\partial p}$ and $\frac{\partial \phi}{\partial \lambda}$. These operations mainly consist of matrix and vector operations a lot of which involve pointwise operations which can be easily spanned across the GPU.

The computation times for the likelihood function are compared on a 5-, 10- and 15-dimensional analytical test problem for sampling plans of varying sizes. The simple sphere function is used to provide the objective function values with the sampling plan defined using a random Latin hypercube. The same sampling plan is used on every machine and varies from 10 to 1000 points for the 5 and 10 variable case and between 10 and 500 points for the 15 variable case. All timings are averaged over 100 evaluations of the likelihood with 100 sets of randomised modelling parameters used. The same modelling parameters are used when testing each piece of hardware. Both the CPU and GPU versions of the code have constant values, such as the distances between sample points, calculated and stored in memory prior to timing the 100 evaluations.

In the case of the GPU the transfer of these constants over to the GPU memory is not included in the timing as it would only occur once during a likelihood optimization

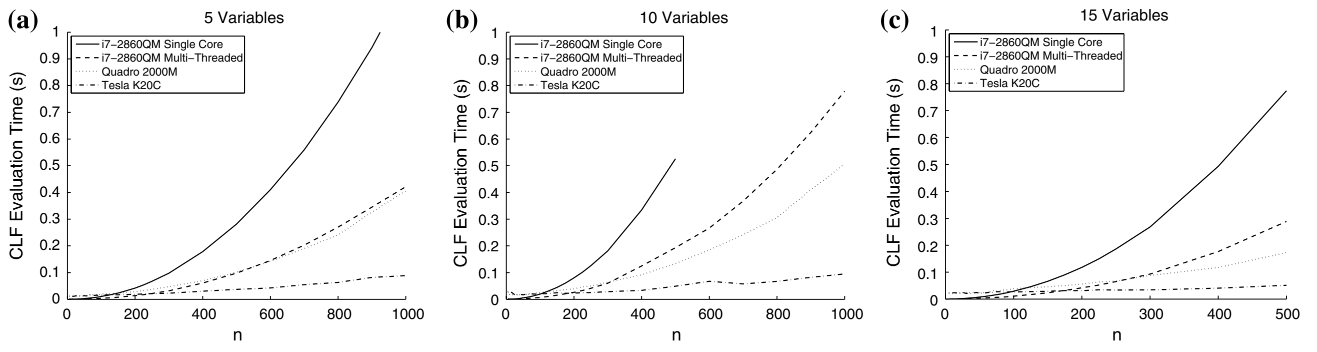

Fig. 7 Comparison of Kriging CLF evaluation costs for varying model size and computation method for **a** 5, **b** 10 and **c** 15-dimensional problems

and would, therefore, have very little bearing on the performance of a GPU-based likelihood optimization. However, the transfer of the Kriging modelling parameters to the GPU and the transfer of the likelihood and its derivatives back from the GPU is included in all of the GPU timings as these would not remain constant during an optimization. All comparisons of likelihood computation times include the calculation of both the likelihood and its adjoint.

Figure 7 presents a comparison of the costs of calculating the Kriging concentrated log-likelihood function using the three pieces of hardware. Also included in Fig. 7 is a line representing the cost of calculating the likelihood using the CPU but with it restricted to a single computational thread. By default Matlab will use all four computational threads on the test machine and the impact of this can be clearly observed in these plots. Given the clear advantage of Matlab using multiple CPU cores, single core computations will be discounted from the comparisons for the remainder of the present article.

Figure 7 illustrates that no matter the dimensionality of the problem there is always some form of overhead associated with the use of either GPU due to the inefficiencies of some processes, such as matrix inversions, when the matrices are small. However, once overcome there is a clear advantage to calculating the likelihood function on a GPU. The Tesla card, in particular, shows a considerable speed improvement over the CPU even for a modest number of design variables. Evaluating the likelihood function for a 5 dimensional problem with a 1000 point sampling plan, for example, is almost one fifth the cost of that of the CPU.

The less powerful Quadro 2000M GPU also offers an improvement in likelihood calculation times over the CPU as the number of sample points increases but the improvement is much less than that of the Tesla card. Nevertheless for high-dimensional problems with a large number of sample points there is a clear advantage to having a GPU evaluate the likelihood function even if that GPU is relatively low end.

3.3 Predictor evaluation comparison

Having considered the evaluation of the likelihood function and, therefore, the construction of a Kriging model let us now consider the model's predictor given by Eq. 9.

It is clear from this equation that both the $\hat{\mu}$ and the $\boldsymbol{R}^{-1}(\boldsymbol{y} - \boldsymbol{1}\hat{\mu})$ terms are independent of the unknown point at which a prediction is to be made. These terms can, therefore, be calculated and stored as soon as the Kriging model parameters have been determined for use in any future Kriging predictions. The DACE toolbox of Lophaven et al. [38] does exactly this.

The only unknown in Eq. 9 is, therefore, the correlation between the unknown point and the sample points, $\boldsymbol{r}$, the calculation of which closely follows that of $\boldsymbol{R}$ in the computation of the likelihood function. Here the distance between the unknown point(s), $\boldsymbol{x}^*$, and the sample points defining the Kriging model are calculated simultaneously in a single matrix operation. The distances are then all taken to the power of p, multiplied by 10^{θ} and summed to produce the vector $\boldsymbol{r}$ (or matrix if more than one sample point is required).

With $\boldsymbol{r}$ obtained the remaining operation is a simple matrix–vector or vector–vector multiplication between $\boldsymbol{r}$ and the stored $\boldsymbol{R}^{-1}(\boldsymbol{y} - \boldsymbol{1}\hat{\mu})$ to which the mean is added. The calculation of both $\boldsymbol{r}$ and $\boldsymbol{r}^T\boldsymbol{R}^{-1}(\boldsymbol{y} - \boldsymbol{1}\hat{\mu})$ are, therefore, prime candidates to benefit from being performed on a GPU.

Figure 8 illustrates the cost of 1000 simultaneous predictions using either a 5, 10 or 15 dimension Kriging model constructed from a variety of different sample sizes. All timings are once again averaged over 100 evaluations. It should be noted that the cost of the GPU predictions also includes the movement of the $\boldsymbol{R}^{-1}(\boldsymbol{y} - \boldsymbol{1}\hat{\mu})$ vector and matrix of differences to the GPU's memory.

As would be expected the cost for all cases scales linearly as the number of sample points increases. However, unlike the evaluation of the likelihood function the GPU

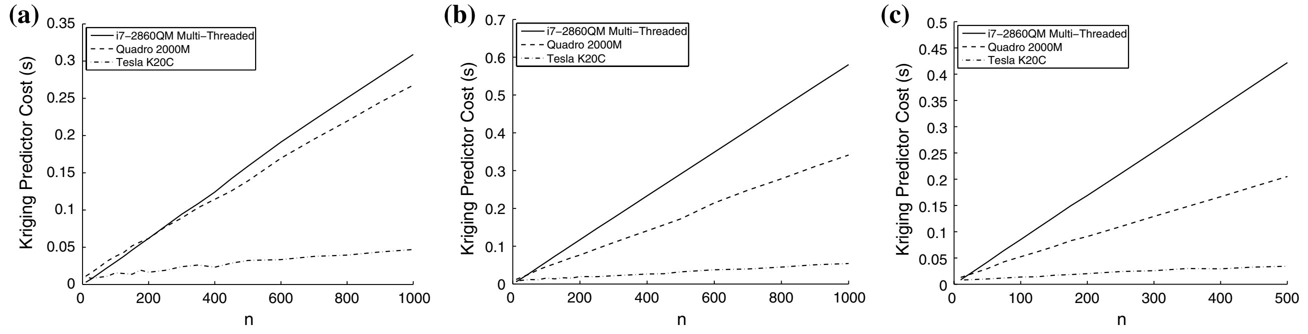

Fig. 8 Comparison of Kriging prediction evaluation costs for 1000 points in parallel for varying model size and computation method for **a** 5, **b** 10, **c** 15-dimensional problems

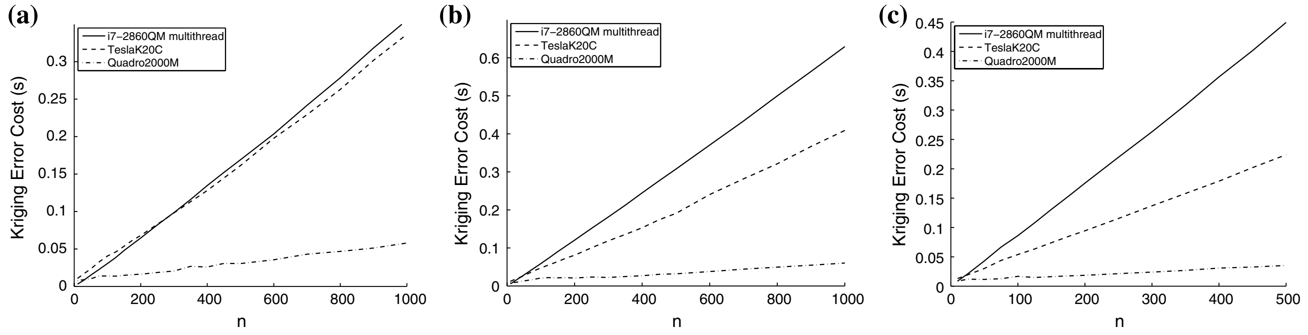

Fig. 9 Comparison of Kriging error evaluation costs for 1000 points in parallel for varying model size and computation method for **a** 5, **b** 10, **c** 15-dimensional problems

offers better evaluation times even with relatively small sample sizes. This is despite the slight overhead in transferring the constant data to the GPU each time. The Quadro 2000M also performs much better across the board than it did when calculating the likelihood function.

The above results, therefore, indicate that even if the likelihood evaluation and, therefore, the model parameter optimization is more efficient on a CPU when the sample size is small it is almost always more efficient to evaluate the predictor of a large number of points on a GPU.

3.4 Error evaluation comparison

Calculating the error has a number of commonalities with the predictor. It too requires the correlation between the sample points and the unknown point(s), $\boldsymbol{r}$, to be calculated. The approach described above for the predictor can, therefore, be repeated here. The major difference is, therefore, the calculation of $\boldsymbol{r}^T\boldsymbol{R}^{-1}\boldsymbol{r}$. The inverse of the correlation matrix can be calculated and stored once the model parameters have been determined as it is independent of the unknown point which leaves a matrix–vector and vector–vector multiplication, if only one point is required or a matrix–matrix followed by a matrix–matrix pointwise multiplication and summation, if the error at more than one point is required. These operations are once again ripe for parallelization on a GPU.

Figure 9 illustrates the cost of calculating the error at 1000 points simultaneously for 5-, 10- and 15-dimensional Kriging models constructed using varying numbers of sample points. As with the predictor the cost increases linearly with the number of sample points and once again the GPU offers a considerable advantage even when there are relatively few sample points in the underlying Kriging model. As with the predictor, even the low end GPU quickly outperforms the CPU.

4 Universal Kriging

4.1 Universal Kriging formulation

A natural extension of ordinary Kriging is to replace the assumption of a constant mean, $\hat{\mu}$, with a mean of a known functional form. This technique is known as universal Kriging [9] where the mean throughout the design space

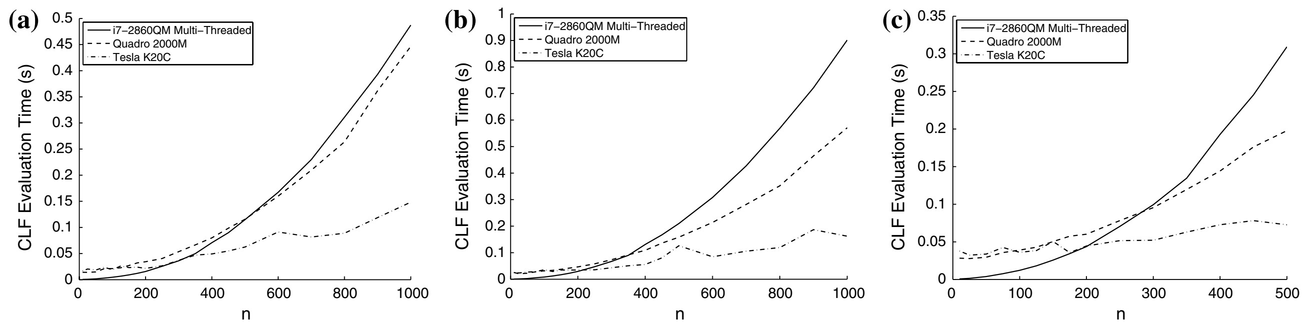

Fig. 10 Comparison of universal Kriging prediction evaluation costs for 1000 points in parallel for varying model size and computation method for **a** 5, **b** 10, **c** 15-dimensional problems

is given by $\mu(\boldsymbol{x}) = \boldsymbol{f}(\boldsymbol{x})^T\boldsymbol{\beta}$ where the vector $\boldsymbol{f}$ is made up of a set of known functions, $\boldsymbol{f}(\boldsymbol{x}) = [1, f_1(\boldsymbol{x}), \ldots, f_m(\boldsymbol{x})]$. Equation 4 in an ordinary Kriging model is, therefore, replaced by,

$$\hat{\boldsymbol{\beta}} = (\boldsymbol{F}^T\boldsymbol{R}^{-1}\boldsymbol{F})^{-1}\boldsymbol{F}^T\boldsymbol{R}^{-1}\boldsymbol{y}, \tag{13}$$

to calculate the vector of unknown parameters, $\boldsymbol{\beta}$. The correlation matrix, $\boldsymbol{R}$, is once again given by Eq. 1 and the only other difference in the calculation of the likelihood function is the replacing of $\boldsymbol{y} - \mathbf{1}\hat{\mu}$ in the above equations with $\boldsymbol{y} - \boldsymbol{F}\hat{\boldsymbol{\beta}}$. This pattern extends to the Kriging predictor which now becomes,

$$y(\boldsymbol{x}^*) = \boldsymbol{f}(\boldsymbol{x}^*)^T\hat{\boldsymbol{\beta}} + \boldsymbol{r}(\boldsymbol{x}^*)^T\boldsymbol{R}^{-1}(\boldsymbol{y} - \boldsymbol{F}\hat{\boldsymbol{\beta}}), \tag{14}$$

with the constant mean, $\hat{\mu}$, replaced by the mean at the unknown point given by the defined functional form. The calculation of the variance for the ordinary Kriging model does not employ a mean term, the formulation, therefore, remains exactly the same for a universal Kriging model. The adjoint formulation of the universal Kriging likelihood function is also identical to that of ordinary Kriging.

4.2 Likelihood evaluation comparison

With the exception of a mean term which varies throughout the design space the calculation of the universal Kriging likelihood function and its adjoint is very similar to that of ordinary Kriging. The calculation of the correlation matrix from a pre-calculated set of distances between the sample points, the inversion of this matrix, the calculation of the variance and the likelihood itself are all identical.

The main difference in the likelihood evaluation is, therefore, the calculation of the unknown parameters of the mean, $\hat{\boldsymbol{\beta}}$, via Eq. 13. While this is of a similar form to the calculation of the mean in ordinary Kriging it now involves matrix–matrix and matrix–vector multiplications and an inversion. While this is more costly than the equivalent operation during an evaluation of the likelihood for an ordinary Kriging model it can certainly benefit from parallelization on a GPU. The matrix $\boldsymbol{F}$ constructed from the known functional form is independent of the modelling parameters θ and p and can, therefore, be computed once, stored and reused in any subsequent likelihood evaluation. A similar process is used in the DACE toolbox of Lophaven et al. [38].

In a similar manner to the ordinary Kriging likelihood, Fig. 10 illustrates the cost of evaluating the universal Kriging likelihood on the CPU and GPUs for problems of 5, 10 and 15 dimension with a variety of different sample sizes, n. Once again the GPU timings do not include the transfer of the constants, such as the matrix of distances between the sample points for each evaluation of the likelihood.

As observed with the ordinary Kriging results there is an overhead associated with using the GPU when there are less than approximately 200 sample points in the model. However, as both the dimensions and number of sample points increases the GPU outperforms the CPU considerably. Once again the Tesla card offers the most performance gain but even the low end Quadro card offers a worthwhile speed-up at high dimensions.

4.3 Predictor evaluation comparison

As noted above the universal Kriging predictor is quite similar in form to that of the ordinary Kriging predictor. The only difference is that the constant mean has been replaced by $\boldsymbol{f}(\boldsymbol{x}^*)^T\hat{\boldsymbol{\beta}}$ and $\boldsymbol{R}^{-1}(\boldsymbol{y} - \mathbf{1}\hat{\mu})$ has been replaced by $\boldsymbol{R}^{-1}(\boldsymbol{y} - \boldsymbol{F}\hat{\boldsymbol{\beta}})$. As with the ordinary Kriging predictor this last term is independent of the unknown point at which a prediction is to be made and can therefore, be calculated, stored and reused indefinitely once the model parameters have been defined. The correlation between the unknown point and the sample points in the model can be calculated in exactly the same way as for the ordinary Kriging predictor. Only the mean term remains to be calculated and can be done so using the known functional form and the

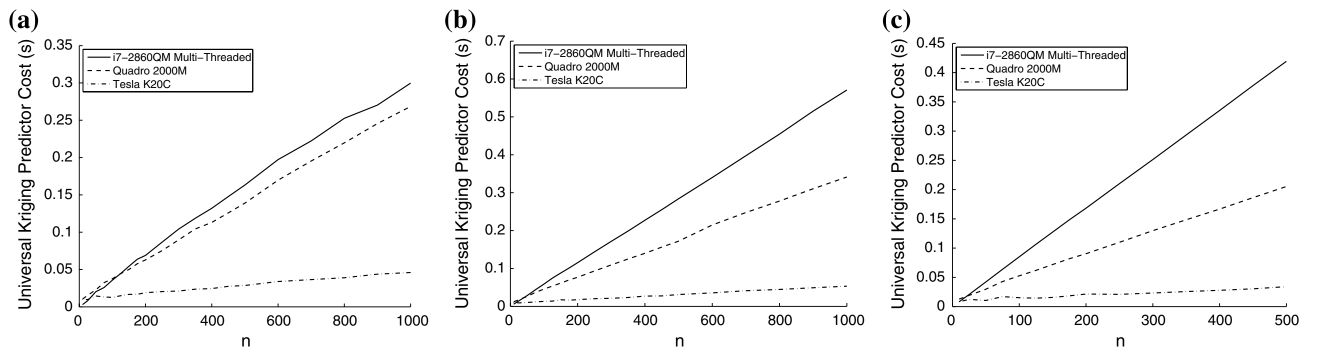

Fig. 11 Comparison of universal Kriging error evaluation costs for 1000 points in parallel for varying model size and computation method for **a** 5, **b** 10, **c** 15-dimensional problems

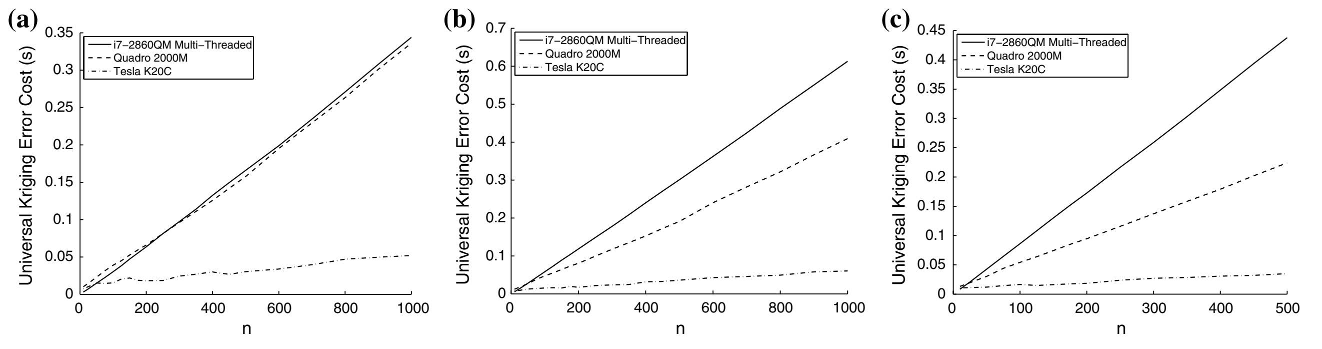

Fig. 12 Comparison of universal Kriging error evaluation costs for 1000 points in parallel for varying model size and computation method for **a** 5, **b** 10, **c** 15-dimensional problems

coordinates of the unknown point and the stored vector of coefficients, $\hat{\boldsymbol{\beta}}$.

In addition to the calculation of $\boldsymbol{r}(\boldsymbol{x}^*)^T\boldsymbol{R}^{-1}(\boldsymbol{y}-\boldsymbol{F}\hat{\boldsymbol{\beta}})$ via a matrix–vector multiplication the universal Kriging predictor therefore, requires an additional matrix–vector multiplication in the calculation of $\boldsymbol{f}(\boldsymbol{x}^*)^T\hat{\boldsymbol{\beta}}$. This assumes, of course, that the prediction at more than one point is required simultaneously, with the above calculations reducing to two vector–vector multiplications if a single point was required. As with the ordinary Kriging model the operations involved in the prediction calculation should, therefore, benefit from parallelization on a GPU.

Figure 11 illustrates the cost of calculating the universal Kriging prediction at 1000 points simultaneously for models with 5, 10 and 15 dimensions with a variety of different sample sizes.

As with the ordinary Kriging predictor there is a linear increase in cost for all cases as the sample size increases. Once again both GPUs perform considerably better than the CPU even when the sample size is relatively small with the Tesla card giving a considerable reduction in computational effort. As with ordinary Kriging, even if the likelihood optimization is more efficiently performed on the CPU it is almost always more efficient to evaluate the predictor for a large number of points on a GPU.

4.4 Error evaluation comparison

As noted above there is no difference in the formulation of the ordinary Kriging error and that of universal Kriging. This is reinforced by the timings presented in Fig. 12 which, when compared to those in Fig. 9 show little discernible difference.

5 Non-stationary Kriging

5.1 Non-stationary Kriging formulation

While both ordinary and universal Kriging have been shown to be effective at design space prediction and design optimization both techniques can struggle to represent non-stationary responses. In this instance, by non-stationarity we mean changes in the smoothness throughout the design space which can only be accurately represented by a corresponding variation in the covariance function, see the solid black line in Fig. 13a for a simple example.

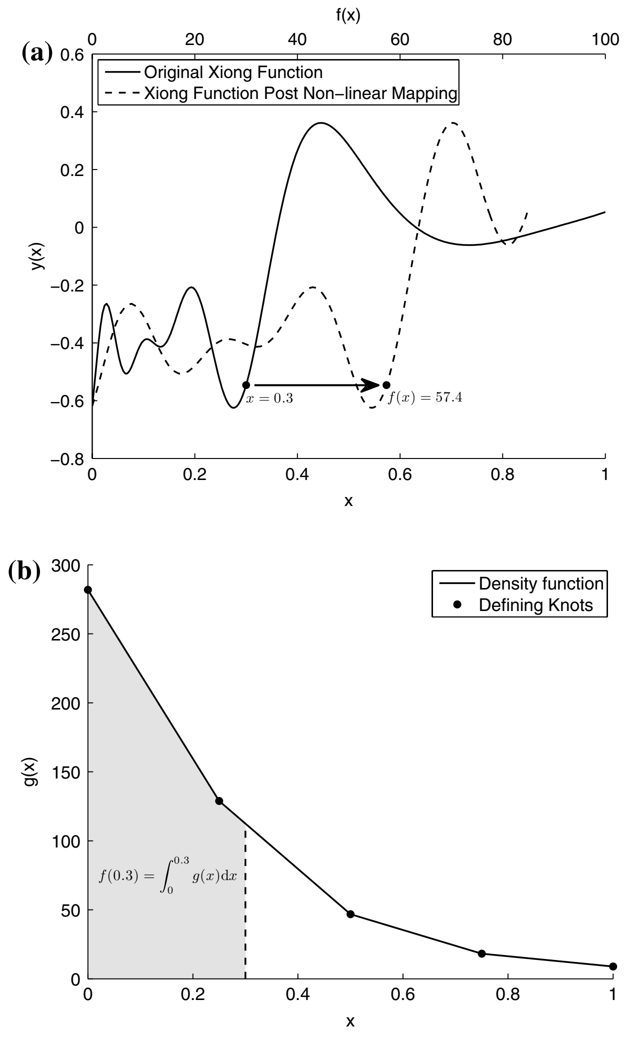

Fig. 13 Illustration of the piecewise linear representation of the density function (**a**) and the resulting non-linear mapping of the Xiong function (**b**) recreated from Toal and Keane [54]

There are a number of different schemes within the literature for dealing with such responses including the direct formulation of non-stationary covariance functions [19, 41, 42], moving window approaches [23], mixtures of experts [18] and nonlinear mappings [47, 60]. Nonlinear mapping schemes, which remap the original function into a space whereby it can be more easily represented by a stationary covariance function, are attractive due to their simplicity, the fact they result in a continuous function and are more suited to small sample sizes.

Nonlinear mapping schemes can, however, suffer from an over parameterization if not formulated correctly and can require a costly multivariate integration. The method used here and proposed by Xiong et al. [60] employs a univariate piecewise linear representation of the density function, Fig. 13b, used to perform the nonlinear mapping. Defining the nonlinear mapping in this manner simultaneously reduces the number of parameters to be optimised while the integration reduces to an analytical function.

The formulation of the non-stationary Kriging model of Xiong et al. is similar to that of an ordinary Kriging model except that instead of a direct correlation between the sample points, $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, the correlation between their nonlinear mappings is used,

$$\boldsymbol{R}_{ij} = \exp\left(-\sum_{l=1}^{d} 10^{\theta}\, \|f(\boldsymbol{x}_i^{(l)}) - f(\boldsymbol{x}_j^{(l)})\|^{\boldsymbol{p}^{(l)}}\right), \tag{15}$$

where the nonlinear mapping is given by,

$$f(\boldsymbol{x}^{(l)}) = \int_0^{\boldsymbol{x}^{(l)}} g(x')\mathrm{d}x', \tag{16}$$

with $g(x')$ defined by a piecewise linear function of K pieces defined using $K+1$ knots of density value 10^{η_k} and position ζ_k. In the following formulation it is assumed that the knots defining this function are evenly spaced along each design variable and that there is a single common θ parameter across all dimensions.

Figure 13, recreated from Toal and Keane [54] illustrates an example of the density function and its impact when used to map a non-stationary function. In this example the nonlinear mapping procedure for the point $x = 0.3$ simply involves the calculation of the shaded area under the piecewise line illustrated in Fig. 13b. When applied to the whole function, as illustrated in Fig. 13a, the nonlinear mapping expands out the rapidly oscillating region found towards the left of the design space while collapsing in the smoother region on the right thereby producing a new function with the same degree of smoothness throughout that can be better represented by a stationary covariance function.

The general integral under the piecewise linear function reduces to a series of constant areas under each section,

$$A_i = \frac{1}{2}(\zeta_{i+1} - \zeta_i)(10^{\eta_{i+1}} + 10^{\eta_i}), \tag{17}$$

where,

$$b_i = \frac{10^{\eta_{i+1}} - 10^{\eta_i}}{\zeta_{i+1} - \zeta_i} \quad \text{and} \quad a_i = 10^{\eta_{i+1}} - b_i\zeta_{i+1}, \tag{18}$$

which can be combined along with the position of the sample point to calculate the integral,

$$f(x) = \frac{1}{2}(x - \zeta_j)(10^{\eta_x} + 10^{\eta_j}) + \sum_{l=1}^{j-1} A_l, \tag{19}$$

where j denotes the piecewise section that x falls within and 10^{η_x} is,

$$10^{\eta_x} = b_i x + a_i. \tag{20}$$

As per any Kriging model the parameters defining the model need to be optimised. Once again the concentrated log-likelihood can be used to perform this optimization and with the correlation matrix defined using the remapped sample points the calculation of the likelihood function proceeds in an identical fashion as that for ordinary Kriging. The major difference here is that not only do θ, p and the regression constant λ need to be optimised but so to do the density values, η at each knot location.

As per ordinary Kriging an adjoint of the modelling parameters can also be derived for non-stationary Kriging thereby accelerating the optimization process. The calculation of $\bar{\boldsymbol{R}}$ remains identical to Eq. 5 while the partial derivatives of the model parameters become [54],

$$\frac{\partial \phi}{\partial \theta} = -10^{\theta} \ln 10 \sum_{l=1}^{d} \left(\sum_{ij} \| \Delta f_{ij}^{(l)} \|^{p^{(l)}} \boldsymbol{R}_{ij} \bar{\boldsymbol{R}}_{ij} \right), \tag{21}$$

where,

$$\Delta f_{ij}^{(l)} = f(\boldsymbol{x}_i^{(l)}) - f(\boldsymbol{x}_j^{(l)}), \tag{22}$$

with

$$\frac{\partial \phi}{\partial p^{(l)}} = \sum_{ij} -10^{\theta} \| \Delta f_{ij}^{(l)} \|^{p^{(l)}} \ln \| \Delta f_{ij}^{(l)} \| \boldsymbol{R}_{ij} \bar{\boldsymbol{R}}_{ij}, \tag{23}$$

and

$$\begin{aligned} \frac{\partial \phi}{\partial \eta_k^{(l)}} = \sum_{ij} & -10^{\theta} p^{(l)} \, \| \Delta f_{ij}^{(l)} \|^{(p^{(l)}-2)} \\ & \times \Delta f_{ij}^{(l)} \left[\frac{\partial f(\boldsymbol{x}_i^{(l)})}{\partial \eta_k^{(l)}} - \frac{\partial f(\boldsymbol{x}_j^{(l)})}{\partial \eta_k^{(l)}} \right] \boldsymbol{R}_{ij} \bar{\boldsymbol{R}}_{ij}, \end{aligned} \tag{24}$$

where

$$\frac{\partial f(\boldsymbol{x}^{(l)})}{\partial \eta_k} = \frac{\partial A_1}{\partial \eta_k} \quad \text{if} \quad k = 1 \tag{25}$$

$$\frac{\partial f(\boldsymbol{x}^{(l)})}{\partial \eta_k} = \frac{\partial A_k}{\partial \eta_k} + \frac{\partial A_{k-1}}{\partial \eta_k} \quad \text{if} \quad k \le L - 1 \tag{26}$$

$$\begin{aligned} \frac{\partial f(\boldsymbol{x}^{(l)})}{\partial \eta_k} = & \frac{1}{2}(\boldsymbol{x}^{(l)} - \zeta_L) \left(10^{\eta_L} \ln 10 + \boldsymbol{x}^{(l)} \frac{\partial b_L}{\partial \eta_k^{(l)}} \right. \\ & \left. + \frac{\partial a_L}{\partial \eta_k^{(l)}} \right) + \frac{\partial A_{L-1}}{\partial \eta_k^{(l)}} \quad \text{if} \quad k = L \end{aligned} \tag{27}$$

$$\frac{\partial f(\boldsymbol{x}^{(l)})}{\partial \eta_k^{(l)}} = \frac{1}{2}(\boldsymbol{x}^{(l)} - \zeta_L) \left(\boldsymbol{x}^{(l)} \frac{\partial b_L}{\partial \eta_k^{(l)}} + \frac{\partial a_L}{\partial \eta_k^{(l)}} \right) \quad \text{if} \quad k = L + 1 \tag{28}$$

$$\frac{\partial f(\boldsymbol{x}^{(l)})}{\partial \eta_k^{(l)}} = 0 \quad \text{if} \quad k \ge L + 2 \tag{29}$$

and where L refers to the Lth piecewise section that $\boldsymbol{x}^{(l)}$ falls within. Given that the density function is represented by a series of K straight lines of intercept a, gradient b and integral A then,

$$\frac{\partial a_L}{\partial \eta_L} = -\zeta_{L+1} \frac{\partial b_L}{\partial \eta_L} \tag{30}$$

$$\frac{\partial a_L}{\partial \eta_{L+1}} = 10^{\eta_{L+1}} \ln 10 - \zeta_{L+1} \frac{\partial b_L}{\partial \eta_{L+1}} \tag{31}$$

$$\frac{\partial b_L}{\partial \eta_L} = \frac{10^{\eta_L} \ln 10}{\zeta_{L+1} - \zeta_L} \tag{32}$$

$$\frac{\partial b_L}{\partial \eta_{L+1}} = \frac{10^{\eta_{L+1}} \ln 10}{\zeta_{L+1} - \zeta_L} \tag{33}$$

$$\frac{\partial A_{k-1}}{\partial \eta_k} = \frac{1}{2}(\zeta_k - \zeta_{k-1}) 10^{\eta_L} \ln 10 \tag{34}$$

$$\frac{\partial A_{k-1}}{\partial \eta_{k-1}} = \frac{1}{2}(\zeta_k - \zeta_{k-1}) 10^{\eta_{L+1}} \ln 10 \tag{35}$$

The formulation of both the non-stationary Kriging predictor and error are identical to those for ordinary Kriging with the exception that the unknown point(s) must undergo the same nonlinear mapping as the points defining the non-stationary Kriging model prior to calculation of the correlations.

5.2 Likelihood evaluation comparison

As described above, once the nonlinear mapping of the sample points has been performed the calculation of the correlation matrix and hence the calculation of the likelihood function for a non-stationary model will proceed in an identical fashion to that of an ordinary Kriging model. The nonlinear mapping, therefore, introduces an additional cost over that of an ordinary Kriging model. As the parameters defining the nonlinear mapping are subject to change during a likelihood optimization the distances between the sample points will no longer be constant. Therefore, not only does the nonlinear mapping have to be performed with every likelihood evaluation but so to does the calculation of distances between the sample points.

The calculation of the non-stationary likelihood function proceeds as follows. Firstly the provided density function values at each knot are used to calculate the gradient, b, and intercept, a of each piecewise line along with the $\frac{\partial a}{\partial \eta}$ and $\frac{\partial b}{\partial \eta}$ values if required. The constant areas, A, under each section

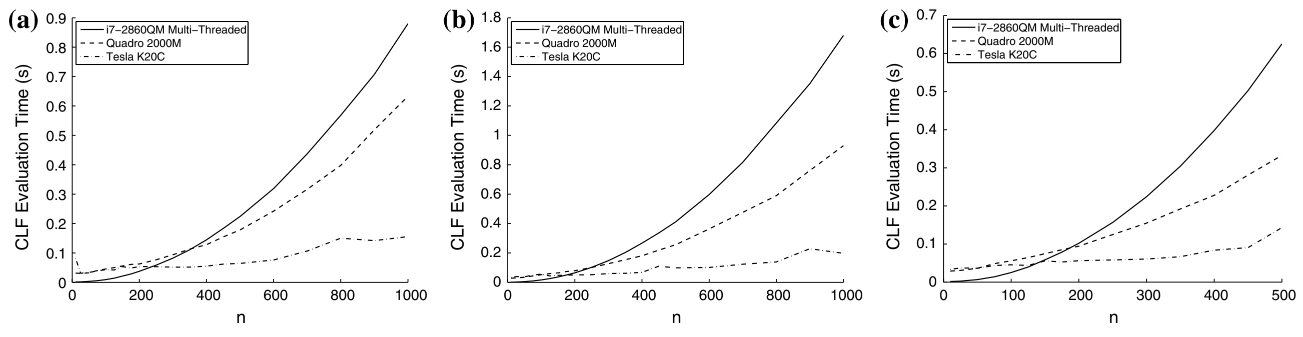

Fig. 14 Comparison of non-stationary Kriging likelihood evaluation costs for varying model size and computation method for **a** 5, **b** 10 and **c** 15-dimensional problems

are then calculated along with their derivatives. The bounds of the piecewise section that each of the sample points falls within is then determined in parallel and used to calculate the cumulative sum of the areas under the preceding sections which in turn is used to initialise the mapped values. The density function value at the current point is then calculated and used to calculate the integral under the current piecewise section. Upon adding this to the cumulative sum of the areas under the preceding sections the nonlinear mapping is completed. During this computation of the nonlinear mapping the derivatives, $\frac{\partial f(\boldsymbol{x})}{\partial \eta}$, can also be calculated.

With the nonlinear mapping performed the distances between every sample point in each axis can be calculated. As per the ordinary Kriging likelihood calculation, this enables the storage of the $n \times n \times d$ matrix of $10^{\theta_k} \mid \boldsymbol{x}_k^{(i)} - \boldsymbol{x}_k^{(j)}|^{p_k}$ values necessary for the subsequent adjoint calculation. With these distances determined the calculation of the correlation matrix and the remainder of the likelihood calculation can proceed as per ordinary Kriging.

With the likelihood calculated the derivative of the likelihood with respect to the modelling parameters still remains to be determined. The majority of this process is the same as that for ordinary Kriging. First the adjoint of the correlation matrix, $\bar{\boldsymbol{R}}$, is calculated using Eq. 5 and used to calculate $\frac{\partial \phi}{\partial \lambda}$ and the pointwise multiplication of $\bar{\boldsymbol{R}}$ and $\boldsymbol{R}$. The calculation of $\frac{\partial \phi}{\partial p}$ proceeds in an identical manner to that for an ordinary Kriging model whereas the calculation of $\frac{\partial \phi}{\partial \theta}$ is the same with the exception that the derivatives for every dimension are summed together as in the above model there is a single θ term. Given that $\frac{\partial f(\boldsymbol{x})}{\partial \eta}$ has been calculated for each sample point and each dimension it only remains to calculate $\frac{\partial \phi}{\partial \eta}$ for each knot density using Eq. 24.

Figure 14 presents a comparison of non-stationary likelihood evaluation times for 5, 10 and 15-dimensional problems with a variety of different sample sizes. Once again all of the results are averaged over 100 evaluations.

As observed with the ordinary and universal Kriging results the evaluation of the non-stationary likelihood function suffers from an initial overhead compared to the CPU-based evaluation. However, for all cases the point at which the GPUs become more efficient requires a smaller sample size. For the 15-dimensional problem the Telsa card becomes more efficient when the sample size is greater than approximately 150 points but for the universal and ordinary Kriging models it's only more efficient when a sample size greater than approximately 200 points is used. Similarly, for the same problem, the Quadro card becomes more efficient after a sample size of 200 with a non-stationary model compared to almost 300 for a universal or ordinary Kriging model.

The above results, therefore, illustrate that as the dimensionality and number of sample points in a non-stationary Kriging model increase it becomes more and more efficient to perform the calculation on a GPU as opposed to a CPU.

5.3 Predictor evaluation comparison

As with the universal Kriging predictor, the predictor for a non-stationary Kriging model is quite similar to that of an ordinary Kriging model. The only substantial difference is the need for the unknown point or points to undergo the same nonlinear mapping as the sample points defining the model. Once this has been carried out the correlation matrix can be constructed as normal with the stored $\boldsymbol{R}^{-1}(\boldsymbol{y} - \boldsymbol{1}\hat{\mu})$ vector and mean reducing the number of calculations required. With the modelling parameters defined the nonlinear mappings of the sample points can also be stored for use in the calculation of the correlation between the unknown and known points.

The nonlinear mapping of the coordinates of the unknown points can be performed using exactly the same algorithm as employed in the likelihood computation. The only difference is that the derivatives, $\frac{\partial f(\boldsymbol{x})}{\partial \eta}$, no longer need

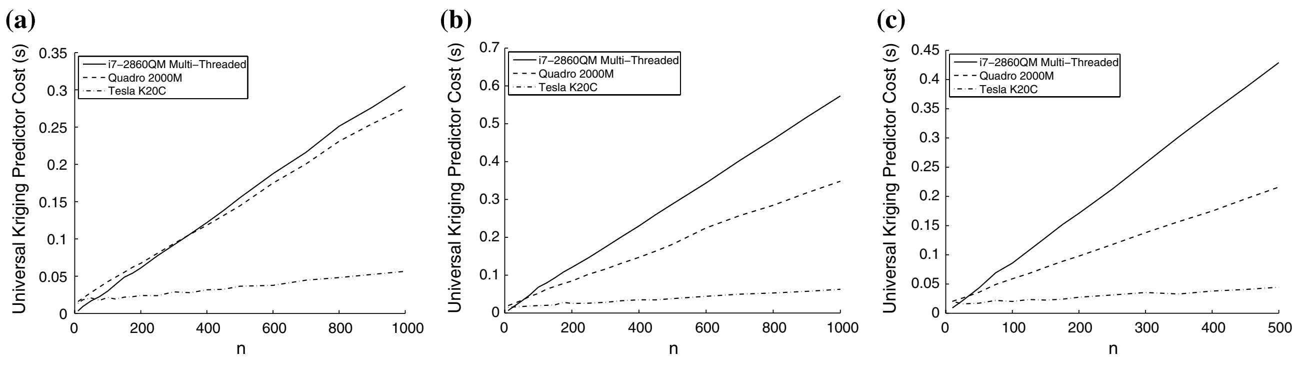

Fig. 15 Comparison of non-stationary Kriging prediction evaluation costs for 1000 points in parallel for varying model size and computation method for **a** 5, **b** 10, **c** 15-dimensional problems

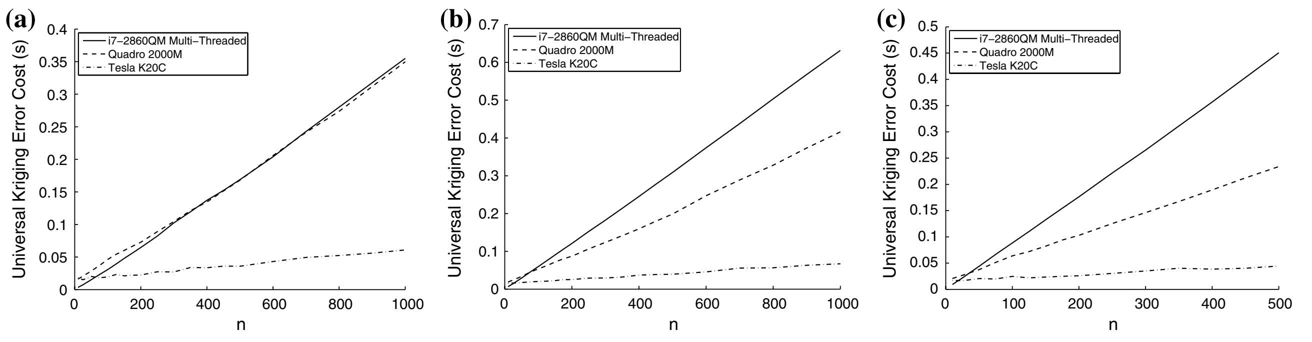

Fig. 16 Comparison of non-stationary Kriging Likelihood evaluation costs for varying model size and computation method for **a** 5, **b** 10 and **c** 15-dimensional problems

to be determined. Other constants such as the values of a, b and A can be stored and reused in any prediction.

Figure 15 presents the cost of 1000 simultaneous evaluations of the non-stationary Kriging predictor using the CPU and GPUs on 5, 10 and 15-dimensional problems for a variety of sample sizes. As with the preceding cases the cost of evaluating the prediction increases linearly with increasing sample size and as with these cases there is a clear advantage to using a GPU even at relatively low sample sizes. Both the low and high end GPUs out perform the CPU in the majority of cases.

5.4 Error evaluation comparison

The calculation of the predicted error of a non-stationary Kriging model is identical to that of an ordinary Kriging model except that, as with the predictor, a nonlinear mapping of the unknown points is required to construct the correlation.

Figure 16 illustrates the cost of making 1000 simultaneous evaluations of the error function of a non-stationary Kriging model constructed in 5, 10 and 15 dimensions using a variety of sampling plan sizes. Once again the linear increase in cost can be observed in all cases with the GPU quickly offering a substantial improvement in calculation time with even the low end GPU providing a considerable performance gain on high-dimensional problems.

6 Multi-fidelity Kriging

6.1 Multi-fidelity Kriging formulation

Multi-fidelity surrogate modelling techniques have grown in popularity over the past few years as they can be used to combine many cheap low-fidelity simulations with a smaller number of more accurate high-fidelity simulations to create a surrogate model much more accurate than a model created with only the high-fidelity simulations [5, 14, 53]. Figure 17 illustrates the potential of constructing such multi-fidelity models. Here two analytical functions are used to represent high-fidelity (expensive) and low-fidelity (cheap) functions. A surrogate model through only the expensive function at the indicated sample points produces an inaccurate prediction but if these points are augmented with 11 cheap sample points the resulting

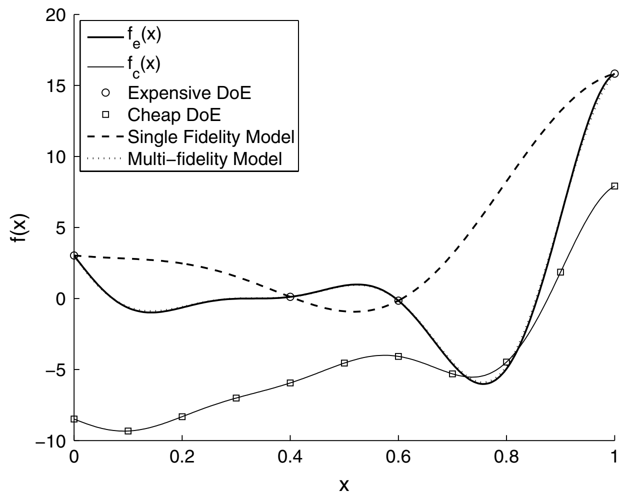

Fig. 17 A multi-fidelity Kriging example

multi-fidelity model almost exactly represents the true response of the expensive function.

Kriging has been extended to make use of multiple levels of simulation data by Kennedy and O'Hagan [31]. In such a model the output of a high-fidelity simulation is approximated by multiplying a surrogate model of the output of a cheap simulation by a scaling factor, ρ, and adding this to a second surrogate model of the difference between the low and high-fidelity simulation outputs,

$$Z_e(\boldsymbol{x}) = \rho Z_c(\boldsymbol{x}) + Z_d(\boldsymbol{x}). \tag{36}$$

If we denote two matrices of sample points, $\boldsymbol{X}_c$ and $\boldsymbol{X}_e$ as representing the cheap and expensive sampling plans, respectively, where we assume that a cheap, low-fidelity simulation has been carried out at every expensive, high-fidelity sample point then the covariance matrix, $\boldsymbol{C}$, is,

$$\boldsymbol{C} = \begin{pmatrix} \sigma_c^2 \boldsymbol{R}_c(\boldsymbol{X}_c, \boldsymbol{X}_c) & \rho\sigma_c^2 \boldsymbol{R}_c(\boldsymbol{X}_c, \boldsymbol{X}_e) \\ \rho\sigma_c^2 \boldsymbol{R}_c(\boldsymbol{X}_e, \boldsymbol{X}_c) & \rho^2\sigma_c^2 \boldsymbol{R}_c(\boldsymbol{X}_e, \boldsymbol{X}_e) + \sigma_d^2 \boldsymbol{R}_d(\boldsymbol{X}_e, \boldsymbol{X}_e) \end{pmatrix}. \tag{37}$$

These correlations are of the same form as Kriging, Eq. 1, except that there are now two correlations and, therefore, twice the number of modelling parameters to determine, one for each surrogate model. In addition to these parameters the scaling factor, ρ must also be determined.

In Kennedy and O'Hagan's approach the cheap data is assumed to be independent of the expensive data which means that the set of modelling parameters for the model of the cheap data can be determined in a completely identical manner to that of an ordinary Kriging model. The remaining set of modelling parameters, including the scaling factor, are determined through a very similar likelihood optimization but where the vector of objective function values, $\boldsymbol{y}$, in the above equations for the ordinary Kriging maximum likelihood estimators of mean and variance, have been replaced by the difference between the high-fidelity output and the low-fidelity output multiplied by the scaling factor,

$$\boldsymbol{d} = \boldsymbol{y}_e - \rho \boldsymbol{y}_c(\boldsymbol{X}_e). \tag{38}$$

The derivatives of the likelihood of the difference model with respect to the θ, p and λ modelling parameters can be calculated in an identical manner to those for an ordinary Kriging model. The derivative with respect to the scaling factor is calculated by first defining the adjoint of the vector of differences [53],

$$\bar{\boldsymbol{d}} = -\boldsymbol{R}_d^{-1}\left(\boldsymbol{d} - \boldsymbol{1}\hat{\mu}_d\right)\left(\frac{1}{\hat{\sigma}_d^2}\right), \tag{39}$$

which can then be used to calculate the derivative as,

$$\frac{\partial \phi}{\partial \rho} = -\sum_{i=1}^{n_e} \boldsymbol{y}_{c_i} \bar{\boldsymbol{d}}_i. \tag{40}$$

With the hyperparameters determined the multi-fidelity Kriging predictor is defined as,

$$y_e(\boldsymbol{x}^*) = \hat{\mu} + \boldsymbol{c}^T \boldsymbol{C}^{-1}(\boldsymbol{y} - \boldsymbol{1}\hat{\mu}), \tag{41}$$

where,

$$\hat{\mu} = \frac{\boldsymbol{1}^T \boldsymbol{C}^{-1} \boldsymbol{Y}}{\boldsymbol{1}^T \boldsymbol{C}^{-1} \boldsymbol{1}}, \tag{42}$$

and

$$\boldsymbol{c} = \begin{bmatrix} \rho\hat{\sigma}_c^2 \boldsymbol{R}_c(\boldsymbol{X}_c, \boldsymbol{x}^*) \\ \rho^2\hat{\sigma}_c^2 \boldsymbol{R}_c(\boldsymbol{X}_e, \boldsymbol{x}^*) + \hat{\sigma}_d^2 \boldsymbol{R}_d(\boldsymbol{X}_e, \boldsymbol{x}^*) \end{bmatrix} \tag{43}$$

and the multi-fidelity error prediction is defined as,

$$s_e^2(\boldsymbol{x}^*) = \rho^2\hat{\sigma}_c^2 + \hat{\sigma}_d^2 - \boldsymbol{c}^T \boldsymbol{C}^{-1} \boldsymbol{c}. \tag{44}$$

With the error and prediction determined the predicted improvement and expected improvement can be calculated as normal.

6.2 Likelihood evaluation comparison

As the optimization of the modelling parameters for a multi-fidelity Kriging model are performed in series the evaluation of the likelihood for the low-fidelity surrogate model is, therefore, identical to that of an ordinary Kriging model. The following section will, therefore, focus on the evaluation of the likelihood for the difference model under the assumption that an appropriate set of parameters for the cheap model has already be obtained.

The calculation of the likelihood function for the difference model is almost identical to that for ordinary Kriging. The correlation matrix is constructed in exactly the same

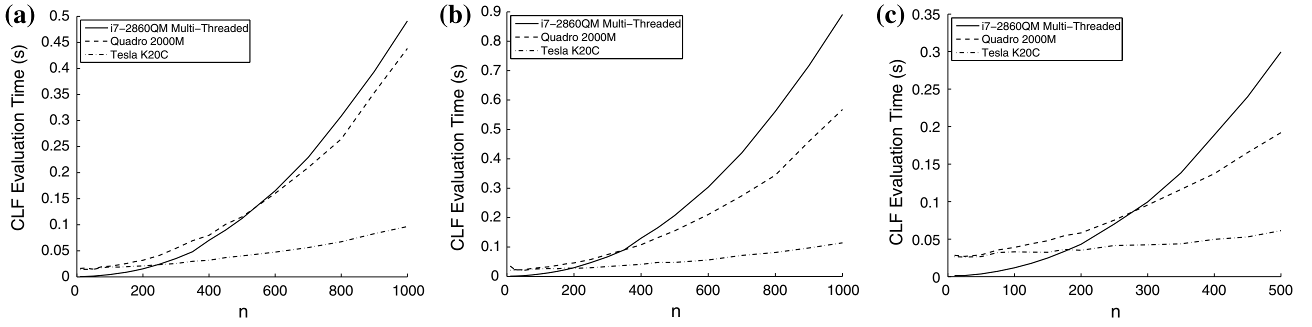

Fig. 18 Comparison of Co-Kriging likelihood evaluation costs for varying model size and computation method for **a** 5, **b** 10 and **c** 15-dimensional problems

fashion using a precomputed matrix of differences between the points. As per the other models in the current paper the inverse of the correlation matrix is explicitly calculated as it is required in the adjoint calculation. As noted above the only major difference is the calculation of $\boldsymbol{d}$ via Eq. 38 which is a very simple operation.

With the likelihood calculated, the calculation of the derivatives with respect to θ, p and λ is identical to that of the ordinary Kriging approach. The calculation of the derivative with respect to ρ is also very simple. The $\boldsymbol{R}_d^{-1}(\boldsymbol{d} - \mathbf{1}\hat{\mu}_d)$ term is used in the calculation of the variance as part of the likelihood calculation and is kept in memory for the calculation of $\bar{\boldsymbol{R}}$ so the calculation of $\bar{\boldsymbol{d}}$ involves a simple division of each element of this vector by the variance. The derivative is then the summation of the pointwise multiplication of $\bar{\boldsymbol{d}}$ and the vector of low-fidelity objective function values at the high-fidelity sample points.

Figure 18 presents the computational costs of evaluating the likelihood of a difference model for a 5, 10 and 15-dimensional model with a varying number of sample points. In each case it is assumed that the difference model is constructed on top of a low-fidelity model defined using a sampling plan with 1000 points. This is analogous to the construction of a multi-fidelity Kriging model from an extremely cheap simulation and an expensive simulation of varying cost thereby enabling different sampling plan sizes.

In the case of the likelihood evaluation of the difference model, the size of the underlying cheap surrogate has no impact as it is only the objective function values of the cheap function corresponding to the sampling plan of the expensive function that is of importance. The size of the matrix and vector multiplications are, therefore, only dependent on the size of the expensive sampling plan. Once again the costs of the evaluating the likelihood are averaged over 100 evaluations with the constant differences between sample points precomputed and passed to the GPUs memory not included.

Overall the results of Fig. 18 are extremely close to that for the ordinary Kriging model. The same overhead when using the GPU on small sampling plans and the same performance advantage on large sampling plans is observed.

6.3 Predictor evaluation comparison

The multi-fidelity Kriging predictor, given by Eq. 41, while similar in form to the ordinary Kriging predictor does require slightly more effort to calculate. As per ordinary Kriging a number of the terms can be precalculated and stored for use once the model parameters have all been determined. The mean, $\hat{\mu}$ and the $\boldsymbol{C}^{-1}(\boldsymbol{y} - \mathbf{1}\hat{\mu})$ terms, for example, can be calculated once and stored as can the matrices of differences between the two sampling plans. The only term which must be calculated is, therefore, the correlation, $\boldsymbol{c}$, between the unknown points and the sampling plans. This operation can be split into two parts, the first to calculate $\rho\hat{\sigma}_c^2\boldsymbol{R}_c(\boldsymbol{X}_c, \boldsymbol{x}^*)$ and the second to calculate $\rho^2\hat{\sigma}_c^2\boldsymbol{R}_c(\boldsymbol{X}_e, \boldsymbol{x}^*) + \hat{\sigma}_d^2\boldsymbol{R}_d(\boldsymbol{X}_e, \boldsymbol{x}^*)$. Both operations involve the calculation of the distances between the unknown and known points as per ordinary Kriging and then the calculation of the correlations. The calculation of $\boldsymbol{R}_c(\boldsymbol{X}_e, \boldsymbol{x}^*)$ and $\boldsymbol{R}_d(\boldsymbol{X}_e, \boldsymbol{x}^*)$ use the same set of differences which can be calculated once and reused. The calculation of the differences and the final correlations both involve a lot of matrix operations which should be amenable to parallelization on a GPU.

Figure 19 illustrates the cost of evaluating the multi-fidelity Kriging prediction at 1000 points simultaneously for 5, 10 and 15-dimensional problems. Unlike previous cases where the number of sample points increased in a linear manner the sampling plans used in this comparison aim to mimic that of a real world multi-fidelity surrogate modelling process. The size of the sampling plan for the underlying cheap function is constant throughout at 25 times the number of dimensions in the problem. The 5 dimension cases, therefore, all have 125 cheap function evaluations while the 15 dimension cases have 375. The number of points in the expensive sampling plan is also a factor of the number of dimensions but this time the factor is permitted

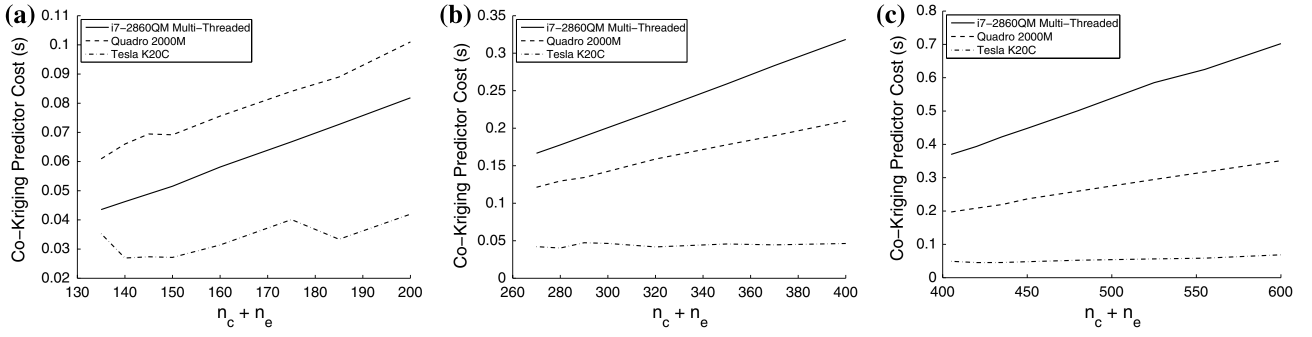

Fig. 19 Comparison of multi-fidelity Kriging prediction evaluation costs for 1000 points in parallel for varying model size and computation method for **a** 5, **b** 10, **c** 15-dimensional problems

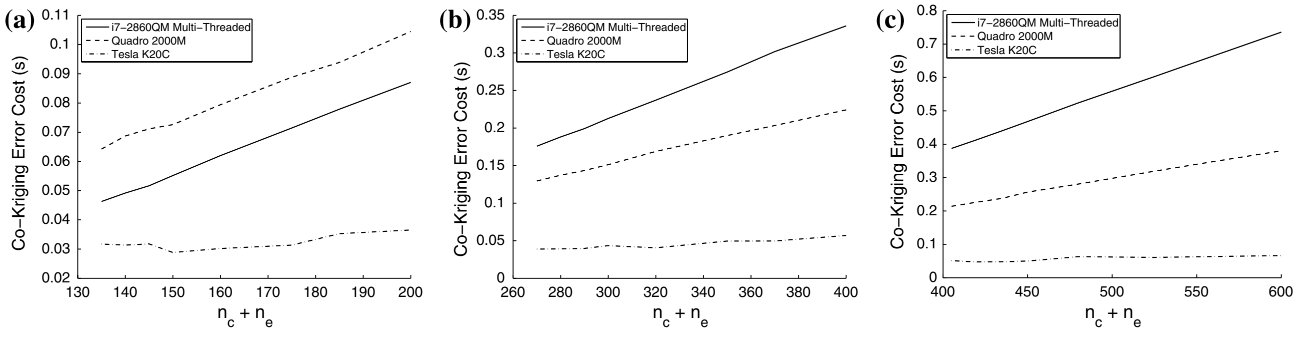

Fig. 20 Comparison of Co-Kriging error evaluation costs for 1000 points in parallel for varying model size and computation method for **a** 5, **b** 10, **c** 15-dimensional problems

to vary from 2 to 15 times the number of dimensions. The 5 dimensional case, therefore, varies from a case with 125 cheap points and 10 expensive points to a case with 125 cheap points and 75 expensive points. Likewise the 15-dimensional case varies from 375 cheap points and 30 expensive to 375 and 225 expensive points.

The results presented in Fig. 19 are similar to those of the previous Kriging predictors. When the total number of points is less than 200 and the problem is of low dimensions the low end GPU is less efficient than the CPU but this quickly reverses as the dimensionality and the number of sample points increases. As with the other predictors the high end Tesla card proves to be much more efficient than the CPU on all of the cases tested.

6.4 Error evaluation comparison

The calculation of the error function of a multi-fidelity Kriging model commences in the same manner as the predictor with the calculation of the correlation between the unknown point or points. With this calculated the multiplications with the stored inverse of the combined correlation matrix, $\boldsymbol{C}^{-1}$, can be carried out and the other terms added. The calculation of $\boldsymbol{c}^T\boldsymbol{C}^{-1}\boldsymbol{c}$ should, therefore, benefit from parallelization.

Figure 20 demonstrates the cost of evaluating the error for 5, 10 and 15-dimensional problems. As with the prediction comparisons a $25d$ low-fidelity sampling plan is used throughout with the high-fidelity sampling plan varying from $2d$ to $15d$ in size.

The results of Fig. 20 reinforce those observed with the previous cases. The Tesla card offers a huge advantage over the CPU in all of the cases considered whereas the Quadro card requires a larger number of sample points and a higher number of dimensions in order to become more efficient.

7 Gradient-enhanced Kriging

7.1 Gradient-enhanced Kriging formulation

The application of derivative information has long been a corner stone of local optimization methodologies [6] and with the development of automatic differentiation [22] and adjoint enabled simulations [21, 25] the derivatives

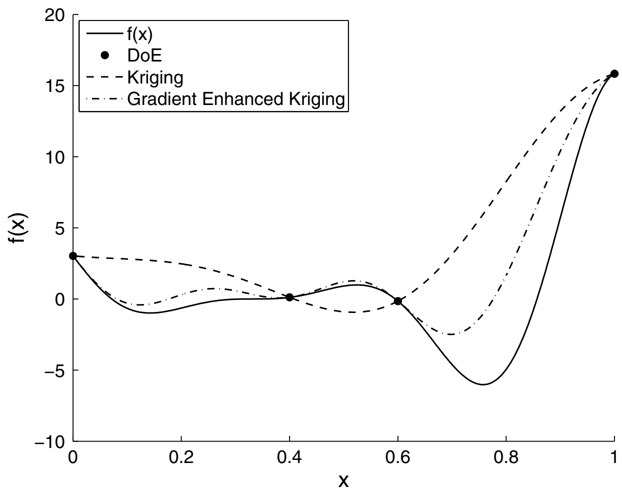

Fig. 21 Gradient-enhanced Kriging example

of even high-dimensional problems can be computed with very little additional cost. All of the variants of Kriging considered up until this point have assumed, however, that only the objective function value is known at each sampling point. Gradient-enhanced Kriging models [33, 34, 44] are an extension of Kriging to include both the objective function value and the derivative of the objective function with respect to each design variable at each sample point. Gradient-enhanced Kriging can, therefore, be used to fully exploit any derivative information available from an adjoint or an automatically differentiated computer code.

Figure 21 illustrates the impact that the inclusion of gradient information can have on a surrogate model. Here, the same analytical function used to illustrate multi-fidelity Kriging is employed but the analytical gradient information at each of the four sample points has also been included in the surrogate model. The resulting model is clearly more accurate throughout the design space than the model without gradient information.

In a gradient-enhanced Kriging model the vector of observed data $\boldsymbol{y}$ containing n observations is now extended to include derivative information at every sample point with respect to every variable,

$$\dot{\boldsymbol{y}} = \left(\boldsymbol{y}^T, \frac{\partial \boldsymbol{y}^T}{\partial x_1}, \frac{\partial \boldsymbol{y}^T}{\partial x_2} \cdots \frac{\partial \boldsymbol{y}^T}{\partial x_k} \right)^T, \tag{45}$$

and is now $n(d+1)$ long. The original correlation matrix, $\boldsymbol{R}$ must, therefore, be expanded to include the correlations between the data points and the derivatives and the derivatives and themselves. The resulting correlation matrix is $n(d+1) \times n(d+1)$ in size and defined as,

$$\dot{\boldsymbol{R}} = \begin{pmatrix} \boldsymbol{R} & \frac{\partial \boldsymbol{R}}{\partial x_1} & \frac{\partial \boldsymbol{R}}{\partial x_2} & \cdots & \frac{\partial \boldsymbol{R}}{\partial x_d} \\ \left(\frac{\partial \boldsymbol{R}}{\partial x_1}\right)^T & \frac{\partial^2 \boldsymbol{R}}{\partial x_1 \partial x_1} & \frac{\partial^2 \boldsymbol{R}}{\partial x_2 \partial x_1} & \cdots & \frac{\partial^2 \boldsymbol{R}}{\partial x_d \partial x_1} \\ \left(\frac{\partial \boldsymbol{R}}{\partial x_2}\right)^T & \frac{\partial^2 \boldsymbol{R}}{\partial x_1 \partial x_2} & \frac{\partial^2 \boldsymbol{R}}{\partial x_2 \partial x_2} & \cdots & \frac{\partial^2 \boldsymbol{R}}{\partial x_d \partial x_2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \left(\frac{\partial \boldsymbol{R}}{\partial x_d}\right)^T & \frac{\partial^2 \boldsymbol{R}}{\partial x_1 \partial x_d} & \frac{\partial^2 \boldsymbol{R}}{\partial x_2 \partial x_d} & \cdots & \frac{\partial^2 \boldsymbol{R}}{\partial x_d \partial x_d} \end{pmatrix}, \tag{46}$$

where $\boldsymbol{R}$ is defined by Eq. 1, the first derivative of the correlation matrix is defined as,

$$\frac{\partial \boldsymbol{R}^{(i,j)}}{\partial x_k} = -2(10^{\theta_k})(x_k^{(i)} - x_k^{(j)})\boldsymbol{R}^{(i,j)} \tag{47}$$

with the corresponding component of $\dot{\boldsymbol{R}}$ on the opposite side defined as its transpose. The diagonal components of $\dot{\boldsymbol{R}}$ defining the correlation between derivatives of the same dimensions are defined as,

$$\frac{\partial^2 \boldsymbol{R}^{(i,j)}}{\partial x_k^2} = \left[2(10^{\theta_k}) - 4(10^{2\theta_k})(x_k^{(i)} - x_k^{(j)})^2\right]\boldsymbol{R}^{(i,j)}, \tag{48}$$

with the off diagonal components, the correlation between derivatives of different dimensions, defined as,

$$\frac{\partial^2 \boldsymbol{R}^{(i,j)}}{\partial x_k \partial x_l} = -4(10^{\theta_k + \theta_l})(x_k^{(i)} - x_k^{(j)})(x_l^{(i)} - x_l^{(j)})\boldsymbol{R}^{(i,j)}. \tag{49}$$

As well as the inclusion of derivative information in such a Kriging model it should be noted that the p modelling parameter has been fixed at 2 in order for the model to be differentiable. While this makes the resulting parameter optimization simpler by reducing the number of variables the size of the correlation matrix, $\dot{\boldsymbol{R}}$, can grow considerably for high-dimensional problems with even a modest number of sample points thereby making the calculation of the likelihood function expensive. With the correlation matrix defined as above the calculation of the likelihood is identical to that for ordinary Kriging with $\boldsymbol{y}$ replaced by $\dot{\boldsymbol{y}}$ in the equations for the $\hat{\mu}$ and $\hat{\sigma}^2$ and the vector of ones, $\mathbf{1}$, now containing n ones followed by dn zeros and denoted as $\dot{\mathbf{1}}$.

As with the previous Kriging models it is possible to define an adjoint of the likelihood function with which to efficiently calculate the derivatives of the likelihood with respect to the θ parameters. Using the same linear algebra results [20] as before the adjoint of the gradient-enhanced Kriging correlation matrix becomes,

$$\bar{\dot{\boldsymbol{R}}} = \frac{1}{2\hat{\sigma}^2}\dot{\boldsymbol{R}}^{-T}(\dot{\boldsymbol{y}} - \dot{\mathbf{1}}\hat{\mu})^T(\dot{\boldsymbol{y}} - \dot{\mathbf{1}}\hat{\mu})^T\dot{\boldsymbol{R}}^{-T} - \frac{1}{2}\dot{\boldsymbol{R}}^{-T}. \tag{50}$$

As can be observed from Eq. 46 the gradient-enhanced Kriging correlation matrix is constructed from four sets of terms, the basic correlation matrix, the correlation of the data points with the derivatives and the correlation of the derivatives with themselves for the same and different dimensions. Equations 47, 48 and 49 which define these correlations all employ both the modelling parameter θ and the original correlation matrix, $\boldsymbol{R}$. As θ is also used to calculate $\boldsymbol{R}$ the calculation of the adjoint of the likelihood with respect to θ therefore, first requires the calculation of the adjoint of the original correlation matrix. As the correlation matrix appears in four different equations defining $\dot{\boldsymbol{R}}$ then its adjoint is itself constructed from four different terms,

$$\bar{\boldsymbol{R}} = \bar{\dot{\boldsymbol{R}}}_{(1:n,1:n)} + \bar{\boldsymbol{R}}_1 + \bar{\boldsymbol{R}}_2 + \bar{\boldsymbol{R}}_3, \tag{51}$$

where $\bar{\dot{\boldsymbol{R}}}_{(1:n,1:n)}$ denotes the top left corner of $\bar{\dot{\boldsymbol{R}}}$ corresponding to the correlation between the data point objective functions,

$$\bar{\boldsymbol{R}}_1 = 2 \times \sum_{k=1}^{d} 2(10^{\boldsymbol{\theta}_k})\Delta \boldsymbol{x}_k \overline{\frac{\partial \boldsymbol{R}}{\partial x_k}} \tag{52}$$

defines the component of $\bar{\boldsymbol{R}}$ due to the correlation between the data points and the gradients,

$$\bar{\boldsymbol{R}}_2 = \sum_{k=1}^{d} \left[2(10^{\boldsymbol{\theta}_k}) - 4(10^{2\boldsymbol{\theta}_k})\Delta \boldsymbol{x}_k^2\right] \overline{\frac{\partial^2 \boldsymbol{R}}{\partial x_k^2}}, \tag{53}$$

defines the component of $\bar{\boldsymbol{R}}$ due to the correlation between gradients of the same dimension and

$$\bar{\boldsymbol{R}}_3^{(k,l)} = -4(10^{\boldsymbol{\theta}_k})(10^{\boldsymbol{\theta}_l})\Delta \boldsymbol{x}_k \Delta \boldsymbol{x}_l \overline{\frac{\partial^2 \boldsymbol{R}}{\partial x_k \partial x_l}} \times 2, \tag{54}$$

defines the component due to the correlation between gradients of different dimensions with $\bar{\boldsymbol{R}}_3$ defined as the sum over all of the off diagonal components $\bar{\boldsymbol{R}}_3^{(k,l)}$. In all of these equations $\Delta \boldsymbol{x}_k$ is defined as a matrix of distances between the sample points of the kth dimension.

As with the correlation matrix, the θ parameter is used in four different correlation calculations, therefore, the derivative of the likelihood with respect to each of the θ parameters is dependent on four components,

$$\frac{\partial \phi}{\partial \theta_k} = \bar{\theta}_1 + \bar{\theta}_2 + \bar{\theta}_3 + \bar{\theta}_4 \tag{55}$$

where $\bar{\theta}_1$ is equal to Eq. 6 with $\bar{\boldsymbol{R}}$ replaced by that due to Eq. 51 and $p = 2$, $\bar{\theta}_2$ is due to the correlation between objective functions and gradients and is given by,

$$\bar{\theta}_2 = 2 \ln 10 \sum_{ij} \frac{\partial \boldsymbol{R}}{\partial x_k} \overline{\frac{\partial \boldsymbol{R}}{\partial x_k}}, \tag{56}$$

$\bar{\theta}_3$ is due to the correlation between gradients of the same dimension,

$$\bar{\theta}_3 = 2 \ln 10(10^{\boldsymbol{\theta}_k}) \sum_{ij} \left[1 - 4(10^{\boldsymbol{\theta}_k})\Delta \boldsymbol{x}_k^2\right] \frac{\partial^2 \boldsymbol{R}}{\partial x^2} \overline{\frac{\partial^2 \boldsymbol{R}}{\partial x_k^2}} \tag{57}$$

and $\bar{\theta}_4$ is due to the correlation between gradients of different dimensions,

$$\bar{\theta}_4 = \ln 10 \sum_{ij} \frac{\partial^2 \boldsymbol{R}}{\partial x_k \partial x_l} \overline{\frac{\partial^2 \boldsymbol{R}}{\partial x_k \partial x_l}}. \tag{58}$$

In the above equations the $\overline{\frac{\partial \boldsymbol{R}}{\partial x_k}}$, $\overline{\frac{\partial^2 \boldsymbol{R}}{\partial x_k^2}}$ and $\overline{\frac{\partial^2 \boldsymbol{R}}{\partial x_k \partial x_l}}$ terms refer to locations in $\bar{\dot{\boldsymbol{R}}}$ corresponding to the locations of $\frac{\partial \boldsymbol{R}}{\partial x_k}$, $\frac{\partial^2 \boldsymbol{R}}{\partial x_k^2}$ and $\frac{\partial^2 \boldsymbol{R}}{\partial x_k \partial x_l}$ respectively, in $\dot{\boldsymbol{R}}$.

With the θ modelling parameters determined the model's predictor and predicted error as well as the probability of improvement and expected improvement can be calculated as with any Kriging model. The equation for the predictor is of a similar form to that for ordinary Kriging,

$$y(\boldsymbol{x}^*) = \dot{\mu} + \dot{\boldsymbol{r}}^T \dot{\boldsymbol{R}}^{-1}(\dot{\boldsymbol{y}} - \dot{\mathbf{1}}\dot{\mu}) \tag{59}$$

with the extended vector of correlations, $\dot{\boldsymbol{r}}$, defined as,

$$\dot{\boldsymbol{r}} = \left(\boldsymbol{r}, \frac{\partial \boldsymbol{r}}{\partial x_1}, \frac{\partial \boldsymbol{r}}{\partial x_2}, \ldots, \frac{\partial \boldsymbol{r}}{\partial x_k}\right)^T \tag{60}$$

and the mean defined as,

$$\dot{\mu} = \frac{\dot{\mathbf{1}}^T \dot{\boldsymbol{R}}^{-1} \dot{\boldsymbol{y}}}{\dot{\mathbf{1}}^T \dot{\boldsymbol{R}}^{-1} \dot{\mathbf{1}}}. \tag{61}$$

The predicted error is once again of a similar form to that of ordinary Kriging,

$$s^2(\boldsymbol{x}^*) = \dot{\sigma}^2 \left[1 - \dot{\boldsymbol{r}}^T \boldsymbol{R}^{-1} \dot{\boldsymbol{r}}\right], \tag{62}$$

with $\boldsymbol{r}$ replaced by $\dot{\boldsymbol{r}}$ and with the variance defined as,

$$\dot{\sigma}^2 = \frac{1}{n}(\dot{\boldsymbol{y}} - \dot{\mathbf{1}}\dot{\mu})^T \dot{\boldsymbol{R}}^{-1}(\dot{\boldsymbol{y}} - \dot{\mathbf{1}}\dot{\mu}). \tag{63}$$

7.2 Constructing a model with missing information

The above formulation assumes, of course, that derivatives are available for all of the variables in the model and that every point has a complete set of derivatives associated with it. This may, of course, not be the case, in real-life design optimization problems there may be a mixture of sampling points with and without gradients and some design variables may preclude the calculation of derivatives completely. To cope with such scenarios a masking vector can be used to exclude columns and rows of the correlation matrix when calculating $\dot{\boldsymbol{R}}^{-1}$. In such an approach inputs to the algorithm include predefined "dummy" values for the derivatives at those points or for those dimensions for which gradients have not been provided. These dummy values can then be used to create a binary masking vector with a one indicating that information is known and a zero

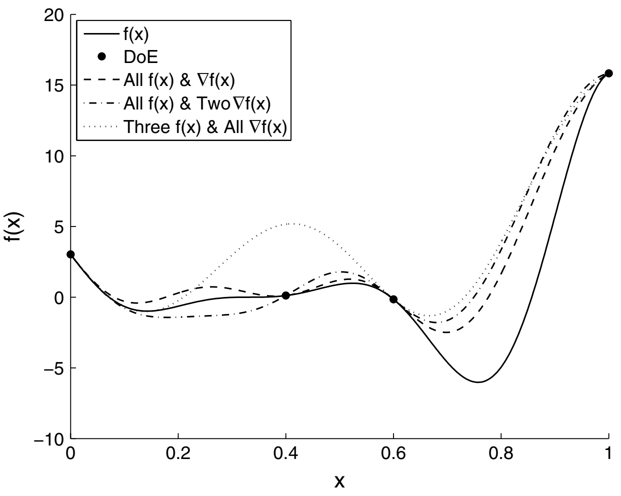

Fig. 22 Gradient-enhanced Kriging example with missing gradient and objective function values

indicating that no information is known. The correlation matrix, $\dot{\boldsymbol{R}}$, can be computed as normal with the masking vector then used to exclude unnecessary columns and rows prior to the calculation of its inverse, mean, variance and likelihood.

When employing such a masking the adjoint calculation commences as before but with the calculation of $\bar{\dot{\boldsymbol{R}}}$ resulting in a smaller matrix. The masking vector is then used to place these values into the rows and columns of the full size, $(d+1)n \times (d+1)n$, $\bar{\dot{\boldsymbol{R}}}$ matrix with the remaining values set to equal 0. Once this matrix has been defined the remaining adjoint calculation can proceed as normal.

Figure 22 illustrates this masking procedure in action using the same analytical example and sampling plan as used in Fig. 21. In this figure three different gradient-enhanced Kriging models have been produced from different amounts of information. In the first model all of the objective function values and all of the gradients have been used to construct the model, i.e. this is the same model as in Fig. 21. In the second model once again all of the objective function values have been used but the gradients at $x = 0.4$ and $x = 1.0$ have not been provided. Applying the above masking procedure clearly results in a model which matches the gradients at those sample points where gradients have been provided and which doesn't at the points where no gradients are known.

As the above masking procedure is completely generic it is also possible to construct a model where gradient information is provided at a point where no objective function value is provided. It would, of course, be unlikely that this would be the case in a real-life design optimization but it may be the case that derivatives of a problem are successfully calculated and returned while the objective function calculation subsequently fails. The third model presented in Fig. 22 illustrates such a case where the derivative information has been provided for all sample points but the objective function information is missing at the point $x = 0.4$. This model clearly interpolates the three remaining sample points and matches the gradients at these points but it also matches the gradient at $x = 0.4$ without interpolating the point as the objective function at this point has not been provided.

7.3 Likelihood evaluation comparison

Having addressed the mathematics behind the calculation of both the likelihood and its adjoint consider now the efficient programming of this calculation. As with the preceding Kriging models the three dimensional matrix of differences between the locations of each sampling point in each of the d dimensions can be evaluated, stored and used in all subsequent likelihood calculations as it is independent of the model parameter values.

With these values stored the calculation of the likelihood commences with the calculation of the correlation matrix $\boldsymbol{R}$. This process is identical to that used for ordinary Kriging with the same intermediate step taken to calculate and store the values of $10^{\theta_k}\,|\boldsymbol{x}_k^{(i)} - \boldsymbol{x}_k^{(j)}|^2$ which are necessary in the calculation of $\bar{\theta}_1$.

The matrix $\boldsymbol{R}$ then forms the top left corner of $\dot{\boldsymbol{R}}$ and is used to calculate the remaining terms. The complete set of $d\ \frac{\partial \boldsymbol{R}}{\partial x}$ terms can be calculated in one operation by carrying out a pointwise multiplication of $\boldsymbol{R}$, which has been repeated out d times, with the stored 3D matrix of differences and a 3D matrix of repeated θ values. This $n \times nd$ matrix can then be placed into the first row of $\dot{\boldsymbol{R}}$ and transposed and placed into the first column of $\dot{\boldsymbol{R}}$.

The calculation of the remaining components can be simplified considerably by considering the $\frac{\partial^2 \boldsymbol{R}}{\partial x_k^2}$ terms as modifications of the $\frac{\partial^2 \boldsymbol{R}}{\partial x_k \partial x_l}$ terms. By rearranging Eq. 48 to give,

$$\frac{\partial^2 \boldsymbol{R}^{(i,j)}}{\partial x_k^2} = 2(10^{\theta_k})\boldsymbol{R}^{(i,j)} - 4(10^{\theta_k+\theta_k})(x_k^{(i)} - x_k^{(j)})(x_k^{(i)} - x_k^{(j)})\boldsymbol{R}^{(i,j)}, \tag{64}$$

it can be observed that Eq. 49 can be calculated simultaneously everywhere with $2(10^{\theta})\boldsymbol{R}$ then added to only the diagonal components to complete $\dot{\boldsymbol{R}}$.

With the correlation matrix defined the calculation of the likelihood function proceeds in an identical manner as for ordinary Kriging. Once again a Cholesky factorisation is performed followed by a calculation of $\dot{\boldsymbol{R}}^{-1}$ as this is used directly in the calculation of $\bar{\dot{\boldsymbol{R}}}$, Eq. 50. Similarly the $\dot{\boldsymbol{R}}^{-1}(\dot{\boldsymbol{y}} - \dot{\boldsymbol{1}}\dot{\mu})$ term used in the calculation of the variance is also stored for use in this calculation.

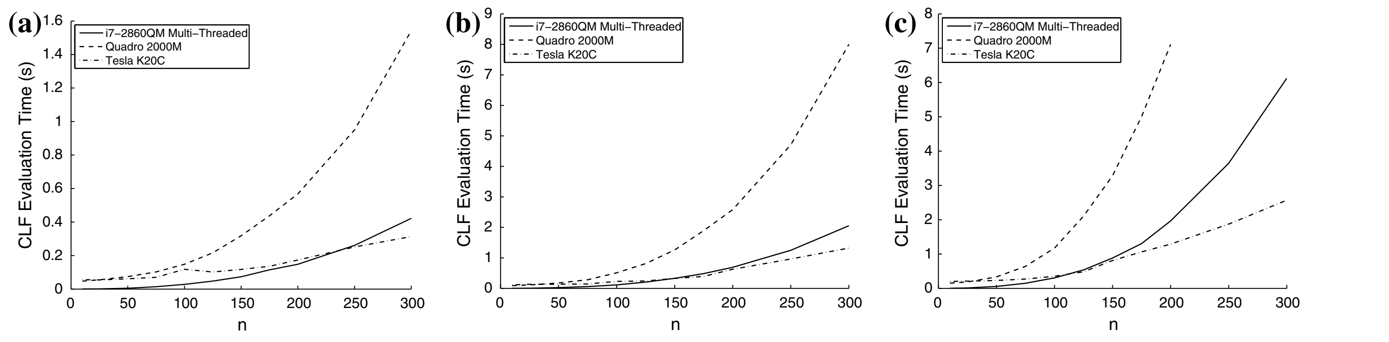

Fig. 23 Comparison of gradient-enhanced Kriging likelihood evaluation costs for varying model size and computation method for **a** 5, **b** 10 and **c** 15-dimensional problems

The calculation of the derivatives commences in a similar manner to that for an ordinary Kriging model with $\dot{\bar{\boldsymbol{R}}}$ calculated using Eq. 50 and the stored matrices. $\bar{\boldsymbol{R}}$ is then initialized using the top left $n \times n$ portion of $\dot{\bar{\boldsymbol{R}}}$. The $\bar{\boldsymbol{R}}_1$ component of $\bar{\boldsymbol{R}}$ can be calculated in one operation and added to the initialized terms. As with the calculation of the second derivative terms of the correlation matrix the calculation of the $\bar{\boldsymbol{R}}_2$ and $\bar{\boldsymbol{R}}_3$ components can be simplified by once again recognising the similarities between Eqs. 53 and 54. In this case $\bar{\boldsymbol{R}}_3$ can be calculated for all cases with additional component due to the diagonal, $\bar{\boldsymbol{R}}_2 = \sum_{k=1}^{d} 2(10^{\theta_k})\frac{\partial^2 \boldsymbol{R}}{\partial x_k^2}$, included separately.

With the calculation of $\bar{\boldsymbol{R}}$ completed $\frac{\partial\phi}{\partial\theta}$ can be initialized using Eq. 6. The $\bar{\theta}_2$ components which are the result of a simple pointwise multiplication between the relevant sections of $\dot{\boldsymbol{R}}$ and $\dot{\bar{\boldsymbol{R}}}$ can then the added to these initialized values. As with the calculation of $\bar{\boldsymbol{R}}$ the calculation of the remaining components, $\bar{\theta}_3$ and $\bar{\theta}_4$ can be simplified by using Eq. 58 for the diagonal terms as well as the off diagonal terms and then adding in the additional term due to the diagonals.

Figure 23 presents a comparison of the likelihood evaluation costs using the CPU and the two GPUs for 5, 10 and 15 variable problems with a variety of sampling plan sizes. In all cases the sampling plan size is limited to a maximum of 300 points due to the rapid increase in the size of the correlation matrix with increasing dimensionality compared to the previous Kriging models. The 15-dimensional gradient-enhanced Kriging model with a 300 point sampling plan, for example, requires a 4800 × 4800 correlation matrix. As with the other models the presented times for the GPU do not include the transfer of constant values to GPU memory which would occur only once at the beginning of any likelihood optimization.

As with the other Kriging models the results of Fig. 23 illustrate an initial overhead for the GPU when there are fewer sampling points in the model. Unlike the previous cases it appears that the complexity of the likelihood calculation makes it much less efficient for it to be performed on a GPU. The low end Quadro GPU, for example, is less efficient than the CPU across all of the cases considered. The Tesla card, on the other hand, does eventually evaluate the likelihood more efficiently than the CPU but is much more sensitive to an increase in sample size than observed with the previous Kriging models. The optimization of the likelihood of a gradient-enhanced Kriging model could, therefore, be said to benefit from having the likelihood evaluated on a GPU but only for high-dimensional problems with large sampling plans and only when evaluated on a high end GPU.

7.4 Predictor evaluation comparison

The calculation of the gradient-enhanced Kriging prediction can be simplified somewhat by the evaluation and storage of the constant values for $\dot{\mu}$ and $\dot{\boldsymbol{R}}^{-1}(\dot{\boldsymbol{y}} - \dot{\boldsymbol{1}}\dot{\mu})$ once the modelling parameters have been defined. As with the other Kriging models it is only the correlation between the unknown points and the existing sample points, $\dot{\bar{\boldsymbol{R}}}$, which needs to be calculated.

The calculation of this correlation vector, or matrix if more than one unknown point is needed, proceeds in a manner similar to the calculation of $\dot{\bar{\boldsymbol{R}}}$. The vector $\boldsymbol{r}$ is first calculated in exactly the same manner as for ordinary Kriging. This vector is then used in the calculation of all of the $\frac{\partial \boldsymbol{r}}{\partial x}$ terms which can be performed in a single operation using the calculated 3D matrix of point distances, a repeated $\boldsymbol{r}$ vector and the θ values.

With $\dot{\bar{\boldsymbol{R}}}$ calculated the prediction can be determined through a vector–vector multiplication with $\dot{\boldsymbol{R}}^{-1}(\dot{\boldsymbol{y}} - \dot{\boldsymbol{1}}\dot{\mu})$ or, if more than one prediction is required, through a matrix–vector multiplication.

Figure 24 illustrates the average cost of evaluating the predictor at 1000 points for problems of 5, 10 and 15 dimensions with varying sampling plan sizes. As with the

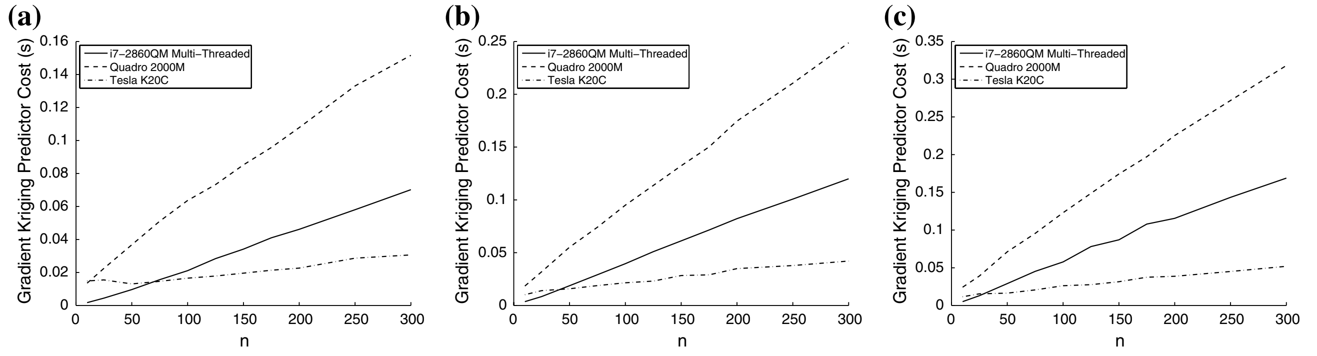

Fig. 24 Comparison of gradient-enhanced Kriging prediction evaluation costs for 1000 points in parallel for varying model size and computation method for **a** 5, **b** 10, **c** 15-dimensional problems

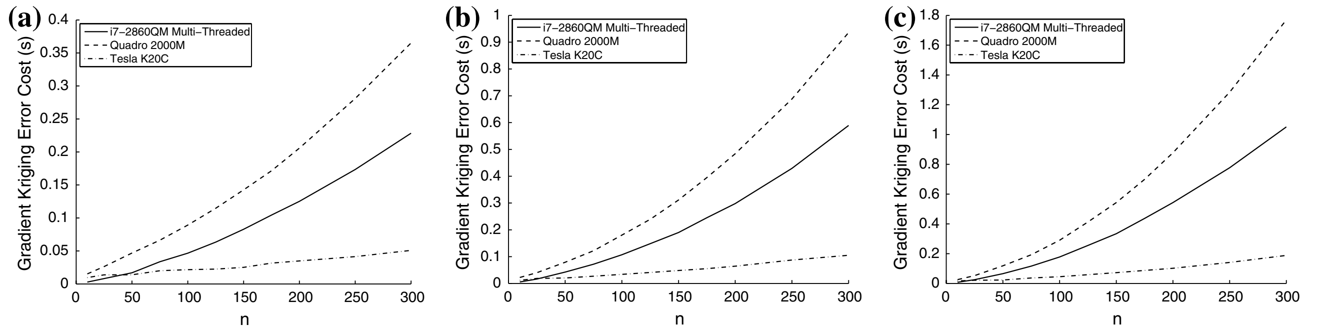

Fig. 25 Comparison of gradient-enhanced Kriging error evaluation costs for 1000 points in parallel for varying model size and computation method for **a** 5, **b** 10, **c** 15-dimensional problems

likelihood evaluation results, the Quadro GPU struggles compared to the CPU and at no point does it perform better whereas the high end GPU performs better in the majority of cases and is less sensitive than the CPU to increasing sampling plan size. Unlike the results for the previous Kriging models those of Fig. 24 indicate an initial overhead when evaluating the predictor on a GPU when the sampling size is small.

7.5 Error evaluation comparison

The calculation of the predicted error, Eq. 62, begins with the same calculation of $\dot{\bar{\boldsymbol{R}}}$ used in the predictor calculation. This is then used along with a precomputed and stored $\boldsymbol{R}^{-1}$ matrix to calculate $\dot{\boldsymbol{r}}^T\boldsymbol{R}^{-1}\dot{\boldsymbol{r}}$.

Figure 25 presents plots comparing the cost of calculating the error on the three pieces of hardware for 5, 10 and 15-dimensional problems with varying sample size. As with the prediction there is no benefit to evaluating the error on the low end Quadro GPU which never performs better than the CPU. The Tesla GPU on the other hand performs consistently better than the CPU with the exception of those cases where the sampling plan is relatively small. The Tesla GPU is also much less sensitive to increasing sampling size.

8 Single precision calculations

The above results clearly illustrate that the calculation of the likelihood, predictor and error functions for a variety of Kriging models can be improved when performed using a GPU, although the level of any performance improvement is dependent on the GPU hardware used. In all cases the lower end Quadro card is outperformed by the high end Tesla card.

However, all of the above comparisons use double precision calculations and GPUs are generally acknowledged to offer greater performance increases over CPUs when performing single precision operations. Can, therefore, the optimization of the Kriging model parameters, the most expensive of the above operations, be accelerated by switching to single precision calculations and performing these calculations on a GPU?

To give an indication of the potential performance improvement when switching to single precision calculations, Fig. 26 presents the compute time when performing

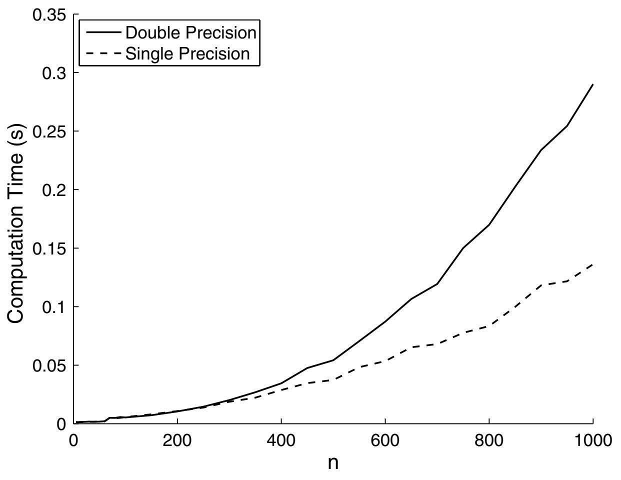

Fig. 26 Single and double precision matrix inversion costs with varying matrix size when using the Quadro 2000M

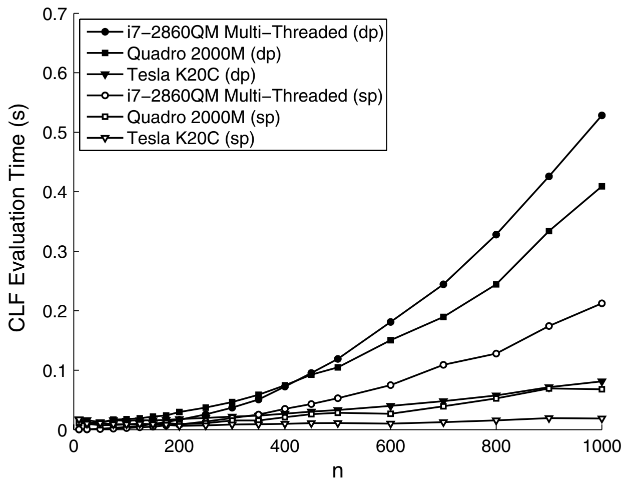

Fig. 27 Comparison of ordinary Kriging likelihood function evaluation cost when using single and double precision

single and double precision matrix inversions using the Quadro card. In this instance given a large matrix, $n = 1000$, there is the potential for the cost to reduce by over 50 % when performed using single precision.

Figure 27 presents the variation in computation time when calculating the likelihood function for an ordinary Kriging model on the CPU and both GPUs when performed on a five-dimensional problem in either single or double precision. This figure clearly illustrates that switching to single precision calculations offers a considerable performance advantage. When $n = 1000$, for example, the cost of calculating the likelihood function using the CPU drops by almost 60 %, the Quadro card by over 83 % and the Tesla card by over 76 %.

Of course, this reduction in calculation time is not without issues and resorting to single precision calculations can result in a loss of accuracy and in an increase in the number of calculations which fail due to the attempted inversion of a non-positive definite correlation matrix. Sampling the Kriging parameter space using a 1000 point Latin Hypercube for a five-dimensional problem with a sampling size of 1000 points i.e. $n = 1000$ in Fig. 27 and comparing the results of double and single precision likelihood evaluations gives an indication of the level of risk involved in employing single precision calculations. Of the 1000 likelihood evaluations approximately 12 % fail due to an attempted inversion of a non-positive definite matrix and of those cases that didn't fail over 3 % resulted in a relative error of more than 1 % compared to the double precision calculation. Of course issues with matrix inversions can also occur when using double precision but these are generally rare thanks to a small level of regression which is always introduced into the model.

While single precision calculations of the likelihood have the potential to be more efficient the greater potential for failed evaluations and errors in successful evaluations means that they need to be employed with caution within any optimization of the Kriging parameters.

The proprietary OPTIMATv2 software, of which all of the presented CPU and GPU functions are part, employs a hybridised particle swarm with periodic particle reinitialization [51] to perform all likelihood optimizations. As with any hybridised algorithm, the stochastic search, the particle swarm, attempts to locate the region of an optimal solution which is then refined using a terminal local optimization, in this case sequential quadratic programming. Clearly, relying solely on single precision calculations while faster offer less chance that the same optimum set of parameters will be obtained as with an optimization employing solely double precision calculations. The larger errors in the likelihood calculation and the high proportion of failures would particularly impact the terminal local search and the derivatives that it relies upon.

The particle swarm optimization, however, offers the basis for a mixture of single and double precision calculations to be performed within one optimization and, therefore, a potential reduction in overall tuning cost with very little loss in reliability. To achieve this the terminal local search is modified to employ only double precision calculations while the particle swarm employs single precision calculations, with a double precision calculation being performed if a single precision calculation fails. This means that the optimization is only really affected by the errors in the magnitude of the likelihood. We, therefore, assume that while these errors exist they are smaller than the general trend in the likelihood function. In other words it is assumed that even with mostly single precision calculations the particle swarm will be able to locate the correct region of the optimal parameter set which can then be converged

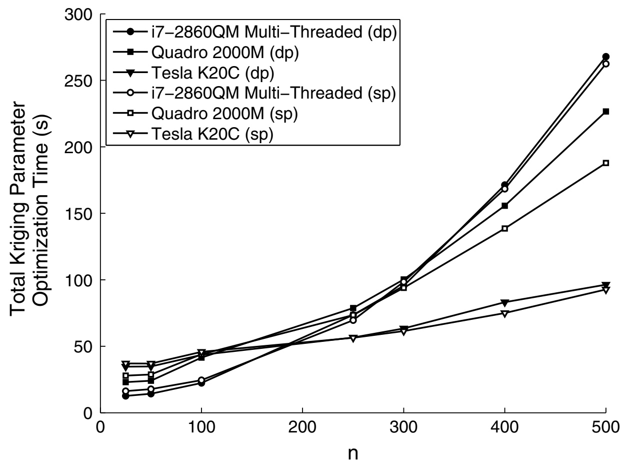

Fig. 28 Comparison of total hyperparameter optimization costs when employing a solely double precision and a single and double precision approach

to precisely using double precision calculations within the local search.

Figure 28 above, illustrates the total time taken to optimize the parameters for a five-dimensional Kriging model using solely double precision calculations within the hybridised search and with a mixture of double and single precision calculations as outlined above. In the majority of cases the strategy employing a mixture of precisions is either of a similar or reduced cost compared to the wholly double precision calculation. The results for the Quadro card, in particular, demonstrate a significant benefit when employing the mixed precision approach with the cost to optimize the parameters of a Kriging model with a sample size of $n = 1000$ reducing by over 17 %. In all cases the mixed precision approach attains the same set of optimal Kriging parameters that were found by the purely double precision approach. However, based on the results of Fig. 27 the reduction in the cost of the optimization is not as great as expected due to the additional cost of recalculating the likelihood for those single precision cases which fail and compensating for errors in the starting position of the terminal local search which may result in more double precision calculations within this part of the search. Of course any improvements in performance may be problem dependent with more single precision failures and, therefore, double precision recalculations as the optimization moves towards the lower bound of the regression parameter (i.e. towards a completely interpolating model) where matrix inversion failures can be more prevalent.

9 Automated selection of CPU/GPU

As already noted the above results illustrate that switching to evaluating the likelihood function on a GPU can offer a reduction in computation time. However, the results also illustrated that, in particular, for low dimensional problems with low sample sizes, the CPU can be more efficient. Kriging models and their different variants can be employed as part of an automated design optimization process where an initial small sampling plan of a design space is augmented with infill points based on some criteria, for example, the predictor or the expected improvement. As such, as a surrogate based optimization progresses the evaluation of the likelihood function involved in constructing the model may move from a region where it is more efficient to do so on the CPU to one where it is more efficient on the GPU. Similarly, an optimization may involve a number of surrogate models constructed from different sampling plans. An engineering design optimization, for example, may involve surrogate models constructed from a few expensive CFD or FEA simulations and other models of constraints derived from large numbers of cheap to evaluate geometrical quantities. In either case an automated approach to switch between CPU and GPU evaluations of the likelihood function would be considerably advantageous.

To that end consider a simple methodology whereby prior to proceeding with the Kriging parameter optimization a check is performed to determine whether evaluating the likelihood is more efficient on the CPU or GPU. In this instance 10 evaluations of the likelihood are performed in series using both the CPU and GPU on 10 different sets of Kriging model parameters. The same set of parameters are used to test both the CPU and GPU. The costs of these 10 evaluations are then used to determine if the CPU or GPU should be used. While this adds a small initial overhead to the total cost of the optimization, correctly determining which approach to use may offer considerable time savings further on.

An alternative to this approach would be to employ a series of mappings based on performance tests such as those carried out previously in this paper. Grids of CPU and GPU performance with varying problem dimensionality and sample size could be interpolated and used to predict the cost of each likelihood evaluation and, therefore, which piece of hardware should be used. While theoretically feasible the practicalities of employing such an approach in the real world are an issue. As can be observed from the previous results there can be a considerable difference in performance between GPUs. To make such an approach accurate over the vast range of hardware available would be impractical and calculating and comparing evaluation costs each time is a more robust approach. Of course if a comparison is performed once on a particular machine it could be stored and reused with the necessary performance mappings, therefore, being generated gradually as the software is used over time.

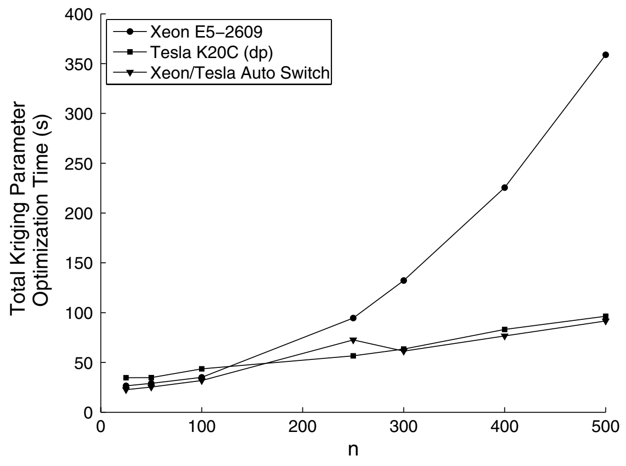

Fig. 29 Comparison of total hyperparameter optimization costs when employing only a CPU or GPU and when this is automatically determined

Figure 29 above illustrates the total parameter tuning cost for an ordinary Kriging model on a five-dimensional problem with varying sample size. This figure presents the total costs when solely using the Tesla card or the CPU and when automatically switching between the the CPU and GPU according to the above strategy. It should be noted here that the CPU used in this instance is a Xeon E5-2609, the CPU in the same machine as the Tesla card. The results in Fig. 29, therefore, represent a live test of the automatic switching and not a theoretical test based on the performance results presented previously.

As per the previous results the Tesla card is much less sensitive to increases in the sampling plan size than the CPU but the results in Fig. 29 still illustrate a cost penalty when the sampling plan size is low. The results for the automatic switching, however, clearly demonstrate that for the cases where n equals 25, 50 and 100 the CPU is being used whereas when $n \geq 150$ the GPU is being correctly used.

Whilst being a relatively simple approach, determining which hardware to evaluate the likelihood function on prior to performing any subsequent optimization is clearly very effective at reducing the overall parameter optimization cost.

10 Conclusions

The present article has investigated the potential of GPUs in the calculation of the concentrated log-likelihood function, prediction and error functions for five variations of Kriging. Ordinary Kriging, universal Kriging, non-stationary Kriging, multi-fidelity Kriging and gradient-enhanced Kriging were all considered with evaluation times for the above functions presented for a quad core CPU and two GPUs, one a low end GPU built into a laptop and the other a high end GPU specially design for GPU-based supercomputers. It should be noted that all likelihood evaluations performed included the adjoint of the likelihood function with the adjoint of the gradient-enhanced Kriging likelihood function presented within this article for the first time. In addition to these comparisons, two strategies for reducing the cost of Kriging parameter optimizations were presented, the first employing a mix of single and double precision calculations, the second employing a scheme for automatically switching between the most efficient hardware.

In evaluating the likelihood for the ordinary, universal, non-stationary and multi-fidelity Kriging models both GPUs showed a performance advantage over the tested CPU with the high end GPU, in particular, being considerably more efficient than the CPU on high-dimensional problems with large sampling plans. The low end GPU was less efficient but still outperformed the CPU when both the sampling plan and the underlying dimensionality of the problem were large enough. Both GPUs, however, demonstrated an inefficiency when calculating the likelihood function for problems with relatively small sampling plans. The cost of evaluating the likelihood for a gradient-enhanced Kriging model was the only case which demonstrated the need for a high end GPU in order to obtain any benefit over the CPU due to the size of the matrix inversions involved. The low end GPU in this case was always less efficient than the CPU.

The time taken to calculate predictions of each Kriging model in parallel was compared using the three hardware configurations. In all cases the predictor was demonstrated to benefit considerably from being evaluated on the GPU. The GPU was less sensitive to increasing sample sizes than the CPU. The only exception to this was the application of the low end GPU to the evaluation of the gradient-enhanced Kriging predictor which was demonstrated to never be more efficient than the CPU.

The efficiency of the error calculations was also shown to improve when performed on a GPU with the GPU again shown to be less sensitive to increasing sampling plan sizes and dimensionality. Once again only the high end GPU was more efficient than the CPU when calculating the predicted error for a gradient-enhanced Kriging model.

Switching from double to single precision calculations were demonstrated to offer a considerable reduction in compute time for the likelihood function but at the risk of introducing errors into the calculation and considerably more failures due to non-positive definite correlation matrices. However, if utilised effectively within an appropriate optimization strategy, such as the mixed single/double precision hybdridized particle swarm presented here, single precision calculations have been demonstrated to offer a reduction in the cost of the total Kriging parameter optimization. The level of this reduction is, of course, dependent on the underlying problem the Kriging model is attempting to represent and, therefore, the number of likelihood evaluation failures and hence double precision recalculations which result.

The performance comparisons for the likelihood functions all demonstrated a region where the CPU was more efficient than either GPU. Typically this was when the sample size was small and the number of dimensions low. In other words, when the loss in performance due to the inefficiency of the GPU-based matrix inversion could not be overcome by any increase in efficiency in the other operations. Performing a brief test of the efficiency of the CPU and GPU likelihood calculation prior to proceeding with the optimization of the likelihood was demonstrated to offer a simple way in which the optimum hardware could be automatically selected. This has the potential to offer significant time savings when the construction of a Kriging model is central to a surrogate based optimization and the sampling plan grows with each iteration.

In conclusion, the above series of studies indicate that GPUs offer considerable potential to reduce the cost of evaluating the Kriging likelihood, predictor and error functions which can reduce the time taken to construct Kriging models, perform Monte Carlo analyses for robust design and to create visualisations of the design space and update metrics. Switching to single precision calculations also has the potential to further reduce calculation times. However, the presented results also demonstrate that these reductions in cost are not without issue. Any performance benefit from a GPU is dependent on hardware configuration and the level of parallelization within the sub-operations making up each function while the use of single precision calculations leads to inaccuracies and a significant increase in the number of failed matrix inversions. To get the most efficient Kriging operations, therefore, requires careful management of when a GPU should and should not be used.

Acknowledgments The support of Rolls-Royce in carrying out this work is greatly appreciated.

References

1. Akbariyeh A, Carrigan T, Dennis B, Chan W, Wang B, Lawrence K (2012) Application of gpu-based computing to large scale finite element analysis of three-dimensional structures. In: Proceedings of the 8th international conference on engineering computational technology
2. Angelikopoulos P, Papadimitriou C (2012) Bayesian uncertainty quantification and propagation in molecular dynamics simulations. In: ECCOMAS 2012—European congress on computational methods in applied sciences and engineering
3. Appleyard J, Drikakis D (2011) Higher-order cfd and interface tracking methods on highly-parallel mpi and gpu systems. Comput Fluids 46(1):101–105. doi:10.1016/j.compfluid.2010.10.019
4. Brandstetter A, Artusi A (2008) Radial basis function networks gpu-based implementation. IEEE Trans Neural Netw 19(12):2150–2154. doi:10.1109/TNN.2008.2003284
5. Brooks C, Forrester A, Keane A, Shahpar S (2011) Multi-fidelity design optimisation of a transonic compressor rotor. In: 9th European turbomachinery conference. Istanbul
6. Broyden C (1970) The convergence of a class of double-rank minimization algorithms. J Inst Math Appl 6(1):76–90
7. Challis V, Roberts A, Grotowski J (2014) High resolution topology optimization using graphics processing units (gpus). Struct Multidiscip Optim 49:315–325. doi:10.1007/s00158-013-0980-z
8. Cheng T (2013) Accelerating universal Kriging interpolation algorithm using cuda-enable gpu. Comput Geosci 54:178–183. doi:10.1016/j.cageo.2012.11.013
9. Cressie N (1993) Statistics for spatial data, probability and mathematical statistics. Wiley, New York
10. Czapinski M, Barnes S (2011) Tabu search with two approaches to parallel flowshop evaluation on cuda platform. J Parallel Distrib Comput 71(6):802–811. doi:10.1016/j.jpdc.2011.02.006
11. Demir I, Westermann R (2013) Progressive high-quality response surfaces for visually guided sensitivity analysis. Comput Graph Forum 32(3):2130. doi:10.1111/cgf.12089
12. Dwight R, Han Z (2009) Efficient uncertainty quantification using gradient-enhanced kriging. In: 50th AIAA/ASME/ASCE/AHS/ASC Structures, structural dynamics, and materials conference. doi: Y10.2514/6.2009-2276
13. Ferreiro AM, Garcfa JA, Lpez-Salas JG, Vzquez C (2013) An efficient implementation of parallel simulated annealing algorithm in gpus. J Glob Optim 57(3):863–890. doi:10.1007/s10898-012-9979-z
14. Forrester A, Bressloff N, Keane A (2006) Optimization using surrogate models and partially converged computational fluid dynamics simulations. Proc R Soc A 462(2071):2177–2204. doi:10.1098/rspa.2006.1679
15. Forrester A, Keane A (2009) Recent advances in surrogate-based optimization. Prog Aerosp Sci 45(1–3):50–79. doi:10.1016/j.paerosci.2008.11.001
16. Forrester A, Keane A, Bressloff N (2006) Design and analysis of "noisy" computer experiments. AIAA Journal 44(10):2331–2339
17. Forrester A, Sóbester A, Keane A (2008) Engineering design via surrogate modelling. Wiley, London
18. Fuentes M (2001) A high frequency Kriging approach for non-stationary environmental processes. Environmetrics 12:469–483
19. Gibbs M (1997) Bayesian Gaussian processes for regression and classficiation. Ph.D. Dissertation, University of Cambridge
20. Giles M (2008) Collected matrix derivative results for forward and reverse mode algorithmic differentiation. Lect Notes Comput Sci Eng 64:35–44. doi:10.1007/978-3-540-68942-3-4
21. Giles M, Pierce N (2000) An introduction to the adjoint approach to design. Flow Turbul Combust 65(3–4):393–415
22. Griewank A (2000) Evaluating derivatives: principles and techniques of algorithmic differentiation. SIAM, Philadelphia
23. Haas T (1990) Lognormal and moving window methods of estimating acid deposition. J Am Stat Assoc 85(412):950–963
24. Hofmann J, Limmer S, Fey D (2013) Performance investigations of genetic algorithms on graphics cards. Swarm Evol Comput 12:33–47. doi:10.1016/j.swevo.2013.04.003
25. Jameson A (1988) Aerodynamic design via control theory. J Sci Comput 3(3):233–260. doi:10.1007/BF01061285
26. Jia X, Gu X, Graves YJ, Folkerts M, Jiang S (2011) Gpu-based fast Monte Carlo simulation for radiotherapy dose calculation. Phys Med Biol 56(22):7017–7031. doi:10.1088/0031-9155/56/22/002

27. Jones D (2001) A taxonomy of global optimization methods based on response surfaces. J Glob Optim 21(4):345–383. doi:10.1023/A:1012771025575
28. Jones D, Schonlau M, Welch W (1998) Efficient global optimization of expensive black-box functions. J Glob Optim 13(4):455–492. doi:10.1023/A:1008306431147
29. Kampolis I, Trompoukis X, Asouti V, Giannakoglou K (2010) Cfd-based analysis and two-level aerodynamic optimization on graphics processing units. Comput Methods Appl Mech Eng 199(9–12):712–722. doi:10.1016/j.cma.2009.11.001
30. Keane A (2006) Statistical improvement criteria for use in mulitobjective design optimization. AIAA J 44(4):879–891
31. Kennedy M, O'Hagan A (2000) Predicting the output from a complex computer code when fast approximations are available. Biometrika 87(1):1–13. doi:10.1093/biomet/87.1.1
32. Krige D (1951) A statistical approach to some basic mine valuation problems on the witwatersrand. J Chem Metallurigical Min Eng Soc S Afr 52(6):119–139. doi:10.2307/3006914
33. Laurenceau J, Sagaut P (2008) Building efficient response surfaces of aerodynamic functions with Kriging and cokriging. AIAA J 46(2):498–507. doi:10.2514/1.32308
34. Leary S, Bhaskar A, Keane A (2004) Global approximation and optimisation using adjoint computational fluid dynamics codes. AIAA J 42(3):631–641
35. Leithead W, Zhang Y (2007) $O(N^2)$-operation approximation of covariance matrix inverse in gaussian process regression based on quasi-newton BFGS method. Commun Stat Simul Comput 36(2):367–380
36. Li Q, Salman R, Kecman V (2010) An intelligent system for accelerating parallel SVM classification problems on large datasets using GPU. In: Proceedings of the 2010 10th international conference on intelligent systems design and applications, pp 1131–1135. doi:10.1109/ISDA.2010.5687033
37. Liao Q, Wang J, Webster Y, Watson I (2009) Gpu accelerated support vector machines for mining high-throughput screening data. J Chem Inf Model 49(12):2718–2725
38. Lophaven S, Nielsen H, Søndergaard J (2002) Dace: A matlab Kriging toolbox, imm-tr-2002-12. Tech. rep., Informatics and Mathematical Modelling, Technical University of Denmark
39. Luo Z, Liu H (2005) Artificial neural network computation on graphic process unit. In: Proceedings of the international joint conference on neural networks. doi:10.1109/IJCNN.2005.1555903
40. Mishra V, Suresh K (2011) GPU-friendly preconditioners for efficient 3-d finite element analysis of thin structures. In: Proceedings of the ASME 2011 international design engineering technical conferences and computers and information in engineering conference
41. Paciorek C, Schervish M (2004) Nonstationary covariance functions for Gaussian process regression. Adv Neural Inf Process Syst 16:273–280
42. Pintore A, Holmes C (2004) Spatially adaptive non-stationary covariance functions via spatially adaptive spectra. Technical Report, University of Oxford, U.K
43. Gutiérrez de Ravé E, Jiménez-Hornero F, Ariza-Villaverde A, Gómez-Lpez J (2014) Using general-purpose computing on graphics processing units (GPGPU) to accelerate the ordinary Kriging algorithm. Comput Geosci 64:1–6. doi:10.1016/j.cageo.2012.11.013
44. Rumpfkeil M (2013) Optimization under uncertainty using gradients, hessians and surrogate models. AIAA J 51(2):444–451. doi:10.2514/1.J051847
45. Rupesh S, Deb K (2013) An evolutionary based bayesian design optimization approach under incomplete information. Eng Optim 45(2). doi:10.1080/0305215X.2012.661730
46. Sacks J, Welch W, Mitchell T, Wynn H (1989) Design and analysis of computer experiments. Stat Sci 4(4):409–435. doi:10.2307/2245858
47. Sampson P, Guttorp P (1992) Nonparametric estimation of nonstationary spatial covariance structure. J Am Stat Assoc 87(417):108–119
48. Sankaran R, Grout R (2012) Accelerating the computation of detailed chemical reaction kinetics for simulating combustion of complex fuels. In: 50th AIAA aerospace sciences meeting including the new horizons forum and aerospace exposition. doi:Y10.2514/6.2012-720
49. Simpson T, Peplinski J, Kock P, Allen J (2001) Metamodels for computer-based engineering design: survey and recommendations. Eng Comput 17(2):129–150. doi:10.1007/PL00007198
50. Toal D, Bressloff N, Keane A (2008) Kriging hyperparameter tuning strategies. AIAA J 46(5):1240–1252. doi:10.2514/1.34822
51. Toal D, Bressloff N, Keane A, Holden C (2011) The development of a hybridized particle swarm for Kriging hyperparameter tuning. Eng Optim. doi:10.1080/0305215X.2010.508524. (Accepted for Publication)
52. Toal D, Forrester A, Bressloff N, Keane A, Holden C (2009) An adjoint for likelihood maximization. Proc R Soc A 465(2111):3267–3287. doi:10.1098/rspa.2009.0096
53. Toal D, Keane A (2011) Efficient multi-point aerodynamic design optimization via co-Kriging. J Aircr 48(5):1685–1695. doi:10.2514/1.C031342
54. Toal D, Keane A (2011) Non-stationary Kriging for design optimization. Eng Optim. doi:10.1080/0305215X.2011.607816
55. Uchida A, Ito Y, Nakano K (2013) Accelerating ant colony optimisation for the travelling salesman problem on the GPU. Int J Parallel Emergent Distrib Syst. doi:10.1080/17445760.2013.842568
56. Viana F, Simpson T, Balabanov V, Toropov V (2014) Metamodeling in multidisciplinary design optimization: how far have we come. AIAA J. doi:10.2514/1.J052375
57. Voutchkov I, Keane A, Fox R (2006) Robust structural design of a simplified jet engine model using multiobjective optimization. In: 11th AIAA/ISSMO Multidisciplinary analysis and optimization conference. Portsmouth
58. Wachowiak M, Lambe Foster AE (2012) GPU-based asynchronous global optimization with particle swarm. In: High performance computing symposium 2012
59. Wang G, Shan S (2007) Review of metamodeling techniques in support of engineering design optimization. ASME J Mech Des 129:370–380. doi:10.1115/1.2429697
60. Xiong Y, Chen W, Apley D, Ding X (2007) A non-stationary covariance-based Kriging method for metamodelling in engineering design. Int J Numer Methods Eng 71(6):733–756
61. Zbierski M (2011) A simulated annealing algorithm for GPU clusters. In: 9th International conference on parallel processing and applied mathematics
62. Zegard T, Paulino G (2013) Toward GPU accelerated topology optimization on unstructured meshes. Struct Multidiscip Optim 48:473–485. doi:10.1007/s00158-013-0920-y
63. Zhang Y, Leithead W (2005) Exploiting hessian matrix and trust-region algorithm in hyperparameters estimation of Gaussian process. Appl Math Comput 171(2):1264–1281
64. Zhongwen L, Hongzhi L, Zhengping Y, Xincai W (2005) Self-organizing maps computing on graphic process unit. In: Proceedings—13th European symposium on artificial neural networks