10th World Congress on Structural and Multidisciplinary Optimization
May 19 - 24, 2013, Orlando, Florida, USA

Computational cost of implementing a surrogate-based exploration or optimization technique

Simon W Miller[1a], Timothy W Simpson[1b], Michael A Yukish[2]

[1]Pennsylvania State University, University Park, Pennsylvania, 16802 USA
[a] swm154@psu.edu, [b] tws8@psu.edu
[2]Applied Research Laboratory at Penn State, State College, Pennsylvania, 16801 USA
may106@psu.edu

1. Abstract
Surrogate models, or metamodels, are compact analytical models that approximate the multivariate input/output behavior of complex systems based on a limited set of computationally expensive simulations. The idea of the surrogate is that it mimics the complex behavior of the underlying simulation model, but it is computationally inexpensive by comparison. It is useful for parametric studies, trade space exploration, and sensitivity analysis as it can be exercised rapidly and, in general, has second-order continuous derivatives. One concern with surrogate modeling is that it is being used indiscriminately. It is often heard that a researcher "sequentially built a meta-model to optimize the objective functions", but there is little said about whether this was necessary or efficient. This paper discusses the computational cost of developing metamodels. A radial basis function is used as an example kernel for estimating the cost of: (1) surrogate regression, (2) surrogate evaluation, (3) expectation of improvement estimator, and (4) accrued cost of sequentially building the regression. It is shown that the surrogate-regression is linear in the dimension of the input space and of order 3 with respect to the number of training points.

2. Keywords: surrogate modeling, computational cost, optimization, trade space exploration

3. Introduction and Motivation
Surrogate models, or metamodels, are compact analytical models that approximate the multivariate input/output behavior of complex systems based on a limited set of computationally expensive simulations [1]. Metamodels require a set of training data so that a regression can be made that estimates the behavior of the underlying or 'true' model under some assumptions. A question then emerges is whether metamodels are worth the computational effort required to construct and use them, let alone if they are a necessary or useful element in the optimization sequence. In fact, all surrogate modeling techniques are interpolating schema; so, to optimize one is trivial. What is optimized is the uncertainty or maximum-likelihood improvement estimator to match the metamodel's representation of the objective space to the true topology. Optimizing the metamodel alone places too much emphasis on exploiting the predictor rather than exploring regions of the space that are uncertain.

The generation of a metamodel has five main phases: (1) generating an initial training set, (2) calculating coefficients for a regression, (3) validating the regression, (4) optimizing the regression for some metric, (5) evaluating the optimized regressions argument in the expensive model, and (6) then going back to step (2) or stopping, as shown in Alg.(1).

Sacks et al. [1] devised a method for looking at complex computer models or codes and determining a fit as a 'cheaper' predictor of the output. This was useful in introducing a new problem of using statistical learning on deterministic outputs; deterministic outputs could be analyzed as a stochastic process under some assumptions. Sacks et al.'s work resulted in the development called *Design and Analysis of Computer Experiments* or DACE. This seminal work led to improvements in exploring a trade space under finite sampling by allowing 'global' knowledge to be gained from finite sample set. Jones et al. [2] state that there is a fundamental problem with the DACE method in optimization in that:

> *The problem with simply finding the minimum of the DACE surface is that this procedure does not acknowledge our uncertainty about that surface. It puts too much emphasis on exploiting the predictor and no emphasis on exploring points where we are uncertain. To eliminate this problem, we must put some emphasis on sampling where we are uncertain, as measured by the standard error of the predictor.*

Generate: generate training points
Evaluate: evaluate training points
while *more training points to be added* **do**
  perform regression analysis
  validate the regression
  optimize a regression metric to determine potential new point
  evaluate new point using *expensive* model and append to set
  if *convergence* **then**
    validate the regression
    break
  else
    continue
  end
end

Algorithm 1: General framework for using metamodels

The question then becomes, how does one choose points to sample and how many cycles are enough for good convergence? Chaudhuri et al. [3] developed a stopping criteria stating that "... continuing with one more cycle is justified only if it yields at least a specified improvement in the objective function". This may lead to the stopping criteria, but is the use surrogates necessary or computationally attractive? This paper seeks to understand the computational cost of generating and optimizing a metamodel as a means of determining its applicability and warrant in the optimization process. In this way, we assume that we are using a 'dumb' computer with limited memory, and so evaluations are counted as soon as they are encountered, i.e., if the same linear solve is shown in an equation twice, it will be counted twice.

4. Surrogate Models

Surrogate models come in many varieties such as response surfaces, kriging, and support vector machines (SVM). These models can be treated as statistical learning tools [4], but each has a set of advantages and disadvantages. Response surfaces are typically easy to implement polynomial regression and can lose much of the 'finer' details of the response by making powerful assumptions about the underlying model's structure [4]. Kriging is a stochastic surrogate that has flexibility to approximate a variety of complex and highly dimensional response functions [5]. Kriging models allow for underlying models (so called Universal kriging) to be used to describe the form of the response and a stochastic formulation to adjust for variations in the response. Radial basis functions (RBFs) are a deterministic form of a kriging model that uses static fitting parameters.

SVMs are used in regression analysis as a form of statistical learning tool [4] that adapts to the sparseness of available input/output data by generating a complete mapping of the input to the output space based on intensity and correlations between points in the space. When SVMs are used for regression analysis, with the proper alternative loss functions such as quadratic [6], they are then called Support Vector Regressions (SVRs). The quadratic loss function is the canonical least squares error and will be used in this paper as the basis for derivations and analysis for the cost of using an SVR. RBF are an application of SVR for metamodeling.

What does it cost to develop the SVR? First, some definitions and mathematical background are in order. An SVR is a nonlinear, statistical learning, data fitting tool. It is used to construct a mapping into a high-dimensional feature space by using reproducing kernels [7]. The kernel is a function that enables numerical operations to be performed on the input space rather than the high dimensional feature space of a support vector classification problem. The use of a kernel maps the inner product from the feature space into an equivalent kernel in the input space [8]. Essentially, each point has 'influence' on the other points in the space, or rather each point supports the space having a pairwise 'intensity' measure.

An inner product in the Reproducing Kernel Hilbert Spaces [7] has some special properties. The following theory is based on [8–11]. The kernel, $\mathcal{K}$, is defined as the inner product of two points in the input space as in Eq.(1) if the following conditions hold.

$$\mathcal{K}(x, x') = \langle \psi(x), \psi(x') \rangle \tag{1}$$

First, $\mathcal{K}$ is a symmetric positive definite function if it satisfies Mercer's Conditions for regularity shown in Eq.(2) and Eq.(3). Any $\mathcal{K}$ that satisfies these conditions represents a valid inner product in the input space is therefore applicable as an SVR kernel with $x \in \mathbb{R}$. L_2 is the Euclidean norm, a special case of the p-norm with $p = 2$ as in Eq.(4).

$$\mathcal{K}(x, x') = \sum_{m}^{\infty} a_m \psi_m(x) \psi(x'), \quad a_m \geq 0 \tag{2}$$

$$\int \int \mathcal{K}(x, x') g(x) g(x') dx dx' > 0, \ g \in L_2 \tag{3}$$

$$L_p = \left(\sum_{i=1}^{n} \left| x_i^{(1)} - x_i^{(2)} \right|^p \right)^{1/p} \tag{4}$$

There are many different kernels that can be used to develop an SVR. Radial basis functions relate a distance metric between two points as their influence using Eq.(4). Some common RBFs include the Gaussian, Multiquadric, Inverse Multiquadric, Polynomial, Wavelet, and Fourier (see [8] for formulations and specifications). The question is now, how much does one of these cost to use? That is, how many operations per kernel evaluation? In this paper, RBFs of the form Eq.(5) are used to assess computational costs. These RBF kernels are (infinitely) differentiable, have continuous derivatives, and have a 'smoothness' defined by p [12]. When $p = 1$, the formula allows for discontinuities, and when $p = 2$ there is considerable smoothness, as seen in Fig.(1) and Eq.(5) is then the Euclidean inner product.

$$\mathcal{K}_{\text{rbf}}(x, x') = e^{-\varepsilon L_p^p} \tag{5}$$

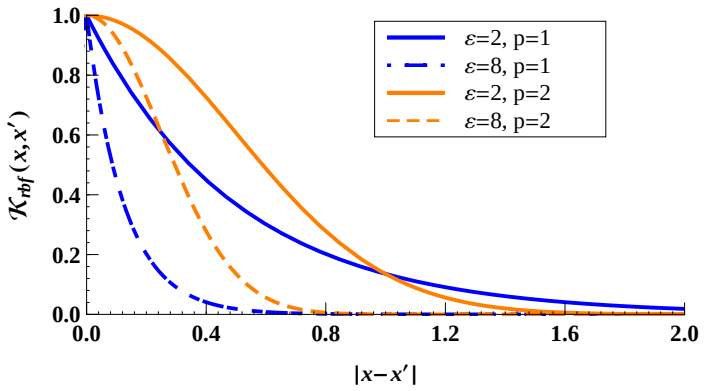


Figure 1: Effect of p and ε on the radial basis function

The Gaussian kernel uses $p = 2$ in Eq.(5) and has one fitting parameter ε. The fitting parameter ε is assumed to be given throughout this paper, but it could be estimated using maximum-likelihood estimator (MLE) techniques that are common for kriging formulations [2]. The computational cost of the Gaussian kernel (C_{gauss}) is shown in Eq.(6), which comprises the cost of taking the squared Euclidean inner product ($C_{L_2^2}$), multiplying it by ε, and then computing the exponential (C_e).

$$C_{\text{gauss}} = C_{L_2^2} + 1 + C_e = 3n + 1 + C_e \tag{6}$$

In this paper, we use RBFs as the basis for determining computational costs because of their flexibility in response and deterministic formulation.

5. Formulating the Problem

Surrogate models have some questions that must be addressed when considering their utility in an application such as quality, purpose, and cost. The quality metric can be answered through a validation technique, such as cross validation. The purpose aspect leads to several more questions. Surrogates require a set of training points and so a decision must be made on how many and in what manner these

should be generated. The training points incur a fixed cost based on the computational cost of the 'true' model that is being represented by the surrogate. The sampling also affects the purpose of the surrogate; is the surrogate being used to exploit or explore? That is, is the surrogate being used to find a local minimum or is it being used to obtain a representation of the feature space? In the former, a sampling and searching technique can be used to rapidly find a local minimum. For the latter, a measure of where to explore needs to be defined. All of these lead to questions of cost; this section seeks to address these questions by analyzing the formulation and structure of the surrogate and determining its cost.

Much of the difficulty in ensuring the quality of an SVR is in the sampling of the data. The problem of obtaining enough information to predict the surfaces from an SVR is affected by the increasing size of the hypercube which bounds the input space — an effect of the *curse of dimensionality*; however, a proper sampling schema allows for good results to be obtained. A design of experiments should be used to generate the training points for the SVR, be it a *maximin* Latin hypercube, random sampling, stratified random sampling, or something else [13].

Other questions about the process are also of concern such as: "should I scale the data?" and "how do I validate the metamodel?" Scaling of the problem can be very beneficial as it redistributes the interactions between dimensions and training points. This is especially true for RBF's as there is, in general, only one scaling coefficient which in and of itself is an art form to calculate [13]. It is many times convenient to work with normalized variables by projecting the training points into the unit hypercube — a process that requires finding the min and max of each dimension ($2n(m-1)$ comparisons) and then projecting the training points into the unit hypercube at a cost of $3nm$ operations for a total of $n(5m-2)$.

Validation of the model is a more difficult question. Leave-one-out strategies are a form of cross validation that is used frequently [4]. The computational cost is to build at most q metamodels and calculate an error term as Eq.(7), where q is a random subset of the training data. The validation comes from removing a subset of training points, fitting a new SVR, and then comparing the fit versus the subset's realized values. When the number of subsets is equal to the number of observations ($q = m$) a nearly unbiased error estimate will be obtained but with high variance [13]. Hastie et al. [4] suggest a smaller subset requirement, namely $q = 5$ or 10, to determine the validity.

$$\epsilon_{\text{cv}} = \frac{1}{q}\sum_{i=1}^{q} \left(y^{(i)} - \hat{y}^{(i)}\right)^2 \tag{7}$$

In this paper, it is assumed that scaling is not an issue, either computationally or necessary, when solving the regression problem. The inclusion of the scaling increases computational cost some but not too much as compared to the more expensive computations required in further analysis. Also, validation can be done by using the methods presented here repeatedly, and so the issue of validation is not addressed formally. In all matters, though, the metamodel needs to be constructed, and so its formulation, solution, and costs are presented in the following sections.

5.1. Solving for the Regression Coefficients

The problem of regression is to find a function that approximates the mapping from the input domain to the output domain based on a finite sample of training points. A quadratic loss function (as is typical in least squares regression) can written as Eq.(8) to assess the quality of the regression fit where $\hat{y}(x)$ is a real-valued function on the domain $X, x \in X \subset \mathbb{R}^n$ and $y \in \mathbb{R}$ are the 'true' or realized values of the function [2, 12–14].

$$L_{\text{quad}}(x, y, \hat{y}) = |y - \hat{y}(x)| = (\hat{y}(x) - y)^2 \tag{8}$$

The notion of the SVR is to have some nonlinear mapping ψ by which the vector x is mapped into the feature or output space. In a typical way, the regressor is written as Eq.(9).

$$y \approx \hat{y}(x, w) = \langle w, \psi(x) + b \rangle \tag{9}$$

The problem, then, is to estimate the function $\hat{y}$ based on a sampling of the space via training points $\{\{x_1, y_1\}, \{x_2, y_2\}, \ldots, \{x_m, y_m\}\}$. A set of weights, w, can be found to minimize the loss of the regressor using the criteria in [14], shown in Eq.(10) where C is a constant parameter.

$$C\sum_{i=1}^{m} L_{\text{quad}}(x_i, y_i, \hat{y}) + \frac{1}{2}\|w\|^2 \longrightarrow \min \tag{10}$$

A synonymous representation of the linear regressor, $\hat{y}(x)$ can be then expressed as Eq.(11) [15].

$$\hat{y}(x) = \sum_{i=1}^{m} \beta_i y_i \langle \psi(x_i), \psi(x_j) \rangle + b \tag{11}$$

This representation allows the regressor to be formulated as a linear combination of the training points. Also, as the kernel being used is an admissible kernel satisfying Mercer's condition, $\hat{y}(x)$ satisfies Eq.(12).

$$\hat{y}(x) = \sum_{i=1}^{m} \gamma_i y_i \mathcal{K}(x_i, x) + b = \sum_{i=1}^{m} \alpha_i y_i \mathcal{K}(x_i, x) + b \tag{12}$$

Then the solution to the regressor's minimization problem (i.e., Eq.(10)) becomes the optimization problem Eq.(13) [8, 12, 15] where α_i are the regression coefficients and * indicates the optimal value.

$$\begin{aligned} \max_{\alpha,\alpha^\star} \left(-\frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} (\alpha_i - \alpha_i^\star)\left(\alpha_j - \alpha_j^\star\right) \mathcal{K}(x_i, x_j) + \sum_{i=1}^{m} (\alpha_i - \alpha_i^\star) y_i - \frac{1}{2C} \sum_{i=1}^{m} \left(\alpha_i^2 + (\alpha_i^\star)^2 \right) \right) \\ \text{subject to:} \quad \sum_{i=1}^{m} (\alpha_i - \alpha_i^\star) = 0, \ i = 1 \dots m \\ 0 \leq \alpha_i, \alpha_i^\star \leq C, \ i = 1 \dots m \end{aligned} \tag{13}$$

We can use the Karush-Kuhn Tucker conditions ($\alpha_i \alpha_i^\star = 0, i = 1 \dots m$) and let $\beta_i = \alpha_i - \alpha_i^\star$; note that $\beta_i^\star = |\beta_i|$. This new optimization problem can then be written as in Eq.(14).

$$\begin{aligned} \min_{\beta} \left(\frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \beta_i \beta_j \mathcal{K}(x_i, x_j) - \sum_{i=1}^{m} \beta_i y_i + \frac{1}{2C} \sum_{i=1}^{m} \beta_i^2 \right) \\ \text{subject to:} \quad \sum_{i=1}^{m} \beta_i = 0, \ i = 1 \dots m \end{aligned} \tag{14}$$

The regressor function is then given by Eq.(9) and where the w and b are determined in Eq.(15) and where x_r, x_s are any two support vectors.

$$\overline{w} = \sum_{i=1}^{m} \beta_i x_i \qquad \overline{b} = -\frac{1}{2} \langle \overline{w}, (x_r + x_s) \rangle \tag{15}$$

The choice of a special kernel (one satisfying Mercer's Conditions of Eq.(2), Eq.(3)) makes this problem much easier to solve. In fact, this problem has a unique solution (assuming no duplicate training points), and the matrix of kernel values is positive definite [16]. Lagrange multipliers can be used to transform the optimization problem into a linear algebra problem as the constraint is affine. The final problem to solve is a quadratic program shown in its matrix form in Eq.(16).

$$\mathcal{L}(\boldsymbol{\beta}, \lambda) = \frac{1}{2} \boldsymbol{\beta}^\intercal \left(\mathbf{M} + \frac{1}{C} \mathbf{I}_m \right) \boldsymbol{\beta} - \boldsymbol{\beta}^\intercal \mathbf{y} + \lambda \left(\boldsymbol{\beta}^\intercal \mathbf{1} \right) \tag{16}$$

$\mathbf{M}$ is the matrix of the kernel evaluations (symmetric where $M_{ij} = \mathcal{K}(x_i, x_j)$) with cost Eq.(17), $\boldsymbol{\beta}$ is a column vector of length m of regression coefficients, $\mathbf{1}$ is a column vector of length m, $\mathbf{I}_m$ is the identity matrix of size m, and λ is the Lagrange multiplier. For convenience, let $\boldsymbol{\mathcal{M}} = \mathbf{M} + \mathbf{I}_m/\mathrm{C}$ to indicate the conditioned matrix of kernel evaluations. The solution to this problem can be found by taking the gradient, setting it equal to zero, and solving the resultant set of equations as shown in Eq.(18).

$$C_{\boldsymbol{\mathcal{M}}} = \frac{m(m+1)}{2} C_{\text{kernel}} + m + 1 \tag{17}$$

$$\nabla_{\boldsymbol{\beta},\lambda} \mathcal{L}(\boldsymbol{\beta}, \lambda) = \begin{bmatrix} \boldsymbol{\mathcal{M}} & \mathbf{1} \\ \mathbf{1}^\intercal & 0 \end{bmatrix} \begin{pmatrix} \boldsymbol{\beta} \\ \lambda \end{pmatrix} = \begin{pmatrix} \mathbf{y} \\ 0 \end{pmatrix} \tag{18}$$

This is a linear algebra problem of the form $\mathbf{Ax} = \mathbf{b}$. The choice of C is somewhat arbitrary, but the larger C is, the less 'error' in each training point. Numerically, C is useful to condition the matrix $\mathbf{M}$ — it can be set equal to ∞. In a computer, it suffices to condition $\mathbf{M}$ by adding any 'small' amount to its diagonal (such as 10^{-6}). The only requirement of the matrix $\mathbf{A}$ is that it has full rank (no linearly dependent rows/columns). The linear algebra solution is therefore valid if the matrix $\boldsymbol{\mathcal{M}}$ is positive definite (duplicate entries can be removed to restore the rank of the matrix $\mathbf{A}$). Note that the matrix $\mathbf{M}$ will, under choice selection of kernels, be positive semi-definite; the addition of $\mathbf{I}_m/C$ makes it positive definite. The addition of the single (affine) equality constraint does not impact the rank of the matrix because of the choice in the kernel as positive semi-definite.

The solution of the system of equations provides all the β_i's and λ so that the SVR can be evaluated for any point $x \in X'$, where X' is the set of x in the interior of the space defined by the training points. The solution of the gradient can easily be found using Gaussian elimination or any other linear equation solving technique. For example, Gaussian elimination has a cost for a matrix of size $\mathbb{R}^{d\times d}$ equal to Eq.(19) and so the resulting cost is Eq.(19) with $d = m + 1$, the number of training points plus the single affine constraint.

$$C_{\text{GE}} = \frac{1}{6}d(-7 + 9d + 4d^2) \tag{19}$$

5.2. Evaluating the Regression

Now that the regression coefficients have been found, how much does it cost to use it? Eq.(11) can be written as Eq.(20). In this equation the β_i's are the regression coefficients; x_i's are the training points — the input values to the regression; $\xi(\cdot)$ is a transforming function that scales the each vector into the unit hypercube; $\mathcal{K}$ is the kernel; m is the number of training points; λ is the Lagrange multiplier (the bias of the regression); and $\mathbf{r}$ is the kernel evaluation of a new point relative to the training points. The vector $\mathbf{r} \in \mathbb{R}^m$ can be written in indicial notation as in Eq.(21) with cost Eq.(22). The cost (of Eq.(20)) is then written in the sum of an inner product and the subtraction of the bias for a total of Eq.(23).

$$\text{SVR}(x) = \left(\sum_{i=1}^{m}\beta_i\mathcal{K}\left(\xi(x_i),\xi(x)\right)\right) + \lambda = \boldsymbol{\beta}^{\intercal}\mathbf{r} + \lambda \tag{20}$$

$$\xi(\cdot): \quad \text{Neglected in Analysis}$$

$$\boldsymbol{\beta}^{\intercal}\mathbf{r} + \lambda: \quad m \text{ multiplications}, m \text{ additions}$$

$$\mathbf{r} = \mathcal{K}\left(\xi(x_i),\xi(x)\right) \;\; i = 1\ldots m \tag{21}$$

$$C_{\mathbf{r}}: \quad mC_{\text{kernel}} \tag{22}$$

$$C_{\text{per use}}(x): \quad m(2 + C_{\text{kernel}}) \tag{23}$$

5.3. Cost of the Regression Process

The total cost to build and solve ($C_{b/s}$) the regression is shown in Eq.(24). This includes the evaluation of the $\mathbf{A}$ matrix (with its internal $\boldsymbol{\mathcal{M}}$ matrix which costs Eq.(17)), and then solving the linear equations.

$$C_{\text{b/s}}(m,n) = \frac{1}{6}(1+m)\left(12 + 17m + 4m^2 + 3mC_{\text{gauss}}\right) \sim \mathcal{O}(m^3 + m^2C_{\text{kernel}}) \tag{24}$$

The Gaussian kernel has a cost of $C_{\text{gauss}} = 3n + 1 + C_e$ where C_e is the cost of the exponential. If it is assumed that $C_e \approx 1.2^*$, then $C_{\text{gauss}} \approx 3n + 2.2$. Plots can be generated of the cost as a function or number of variables (n) and the number of training points (m) shown in plots of building the regression is shown in Fig.(2).

Fig.(2) shows that to build a regression with 50 training points and 100 variables is ≈478,000 operations. This method has shown the cost of solving for the regression coefficients. In some instances, the coefficients may need to be found using a maximum likelihood estimator (MLE) technique, and so an optimization routine must be written to get the best fitting parameters [1, 2, 5]. The next section discusses the cost of setting up this method for use.

*Computational cost is assessed by averaging the cost of computing the MATLAB functions (exp, sqrt, normpdf, normcdf) 10^6 times relative to addition on a PC running Windows 7 with 2.66GHz dual core processor and 4GB RAM.

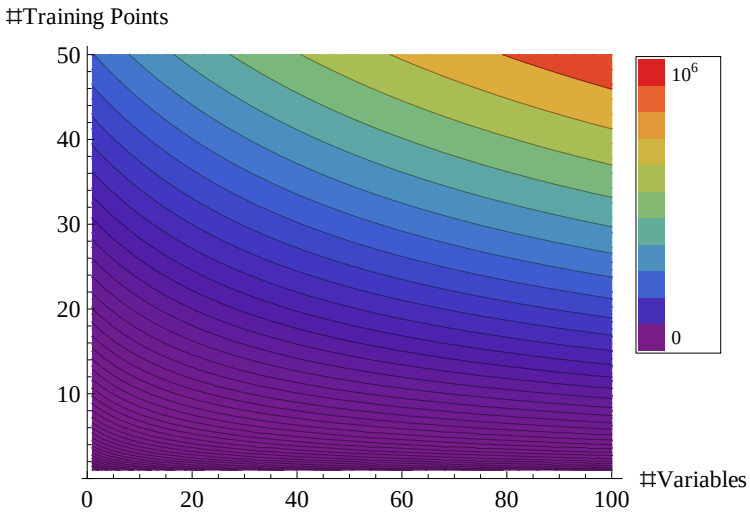


Figure 2: Contour plot of the computational cost of building a regression based on the number of variables in the problem and the number of training points used to generate the regression

5.4. Determining the Cost for Optimizing for the Coefficients

Following the work of [1] and [2], the regressor can be expressed as in Eq.(25) where $\hat{\mu}$ (i.e., Eq.(26)) is the bias of the regressor (as was λ before), and the other variables are as defined earlier.

$$\hat{y}(x) = \hat{\mu} + \mathbf{r}^\intercal \boldsymbol{\mathcal{M}}^{-1} (\mathbf{y} - \mathbf{1}\hat{\mu}) \tag{25}$$

$$\hat{\mu} = \frac{\mathbf{1}^\intercal \boldsymbol{\mathcal{M}}^{-1} \mathbf{y}}{\mathbf{1}^\intercal \boldsymbol{\mathcal{M}}^{-1} \mathbf{1}} \tag{26}$$

Under these conditions, the required number of computations is three Gaussian eliminations (i.e., Eq.(19)), construction of $\boldsymbol{\mathcal{M}}$ (i.e., Eq.(17)), $\mathbf{r}$ (i.e., Eq.(22)), three inner products ($3(2m-1)$), one vector subtraction (m), one addition (1), and one division (1). The total number of operations is then given in Eq.(27).

$$C_{\text{use}_2} = \frac{1}{2} m \left(13 + 9m + 4m^2 + (3+m)\, C_{\text{kernel}}\right) \tag{27}$$

This method is more expensive (i.e., Eq.(27)) than building (i.e., Eq.(24)) and exercising (i.e., Eq.(23)) the regression due to the number of matrix inversions noted by the amount in Eq.(28). Also, in the long run when sequentially building and applying this method, the number of inversions will become increasingly high to find the next point.

$$C_{\text{diff}} = \frac{1}{3} \left(-3 - 7m + 3m^2 + 4m^3\right) \tag{28}$$

Additionally, this does not count subsequent uses of the regression; it only affords one optimization iteration. If subsequent uses are taken into account (assuming 1 of Eq.(24) and $\mathcal{N}(m,n)$ of Eq.(23) minus Eq.(27)), then the difference is the cost is noted in Eq.(29) where $\mathcal{N}(m,n)$ is the number of uses to find the optimal coefficients.

$$\begin{aligned} C_{\text{seq cost diff}} = \frac{1}{6} \Big(&-12 - 41m - 21m^2 - 4m^3 + \mathcal{N}(m,n) \left(6 + 15m + 27m^2 + 12m^3\right) \\ &+ C_{\text{kernel}} \left(-9m - 3m^2 + 3m\mathcal{N}(m,n) + 3m^2\mathcal{N}(m,n)\right)\Big) \end{aligned} \tag{29}$$

5.5. Finding the Next Training Point for Update

By considering the regression $\hat{y}(x)$ as a realization of a random number, then the probability of an improvement $I = y_{\min} - Y(\mathbf{x})$ (where $Y(\mathbf{x})$ is the *actual* function and the objective is assumed to be minimizing this function) against the current best solution can be written as Eq.(30) [13].

$$P[I(x)] = \frac{1}{s\sqrt{2\pi}} \int_{\infty}^{0} e^{\left(-(I-\hat{y}(x))^2\right)/2s^2} \mathrm{d}I \tag{30}$$

This probability of improvement is commonly substituted by calculating its expectation (i.e., Eq.(31)) given that the mean regression is $\hat{y}(x)$, its variance as $s^2(x)$ from Eq.(32) with its calculated component σ^2 in Eq.(33) [2, 13] where $\Phi(\cdot)$ and $\phi(\cdot)$ are the normal cumulative distribution and probability density function, respectively.

$$E[I(x)] = \begin{cases} (y_{\min} - \hat{y}(x))\Phi\left(\frac{y_{\min} - \hat{y}(x)}{s(x)}\right) + s\phi\left(\frac{y_{\min} - \hat{y}(x)}{s(x)}\right) & \text{if } s > 0 \\ 0 & \text{if } s = 0 \end{cases} \tag{31}$$

$$s(x)^2 = \sigma^2 \left(\left(1 - \mathbf{r}^\intercal \boldsymbol{\mathcal{M}}^{-1} \mathbf{r}\right) + \frac{\left(1 - \mathbf{1}^\intercal \boldsymbol{\mathcal{M}}^{-1} \mathbf{r}\right)^2}{\mathbf{1}^\intercal \boldsymbol{\mathcal{M}}^{-1} \mathbf{1}} \right) \tag{32}$$

$$\sigma^2 = \frac{(\mathbf{y} - \mathbf{1}\hat{\mu})^\intercal \boldsymbol{\mathcal{M}}^{-1} (\mathbf{y} - \mathbf{1}\hat{\mu})}{m} \tag{33}$$

The reasoning for using expectation is to escape local minima by forcing some exploration during the modeling in the form of an infill criteria [13]. The point with the highest value would then be realized using the actual function. That is to say, this function should be maximized, and the resulting point appended to the training set for the metamodel and then the process should repeat.

The cost of doing this is calculated from the costs of the individual steps, including: $\boldsymbol{\mathcal{M}}$ from Eq.(17), $\mathbf{r}$ from Eq.(22), four Gaussian eliminations from Eq.(19), five inner products costing $5(2m-1)$, two vector subtraction costing $2m$, five subtractions, two additions, four multiplications, four divisions, one set of comparisons costing $m-1$, and computing the cdf, pdf, and two square roots. The total cost is then given by Eq.(34).

$$C_{\text{MLE}} = \frac{1}{6}\left(60 + 68m + 36m^2 + 16m^3 + 3m(3+m)C_{\text{kernel}}\right) + C_\Phi + C_\phi + 2C_{\sqrt{}} \tag{34}$$

Of course, optimizing Eq.(31) is not trivial, but it does have a lot of good properties as the function is smooth and continuous, but possibly multimodal. However, almost any optimization algorithm can find a local maximum in this space relatively easily. The only constraints in the formulation are that the point lies within the bounds of the sampled space, which makes the constraints simple to enforce via penalty methods. Random Search is a simple and attractive method, but so is something like a differential evolution or a gradient-based method with multistart.

5.6. Cost of Sequentially Solving the Regression

Sequentially building an SVR as new training points are introduced can accrue large costs. The cost can be minimized with proper storage techniques since the addition of each training point adds one element to the input and output vectors and expands the matrix $\mathbf{M}$ by 1 as well. However, if it is assumed that this is not done, then the computation of the influence matrix must be performed many times at the cost specified by Eq.(24).

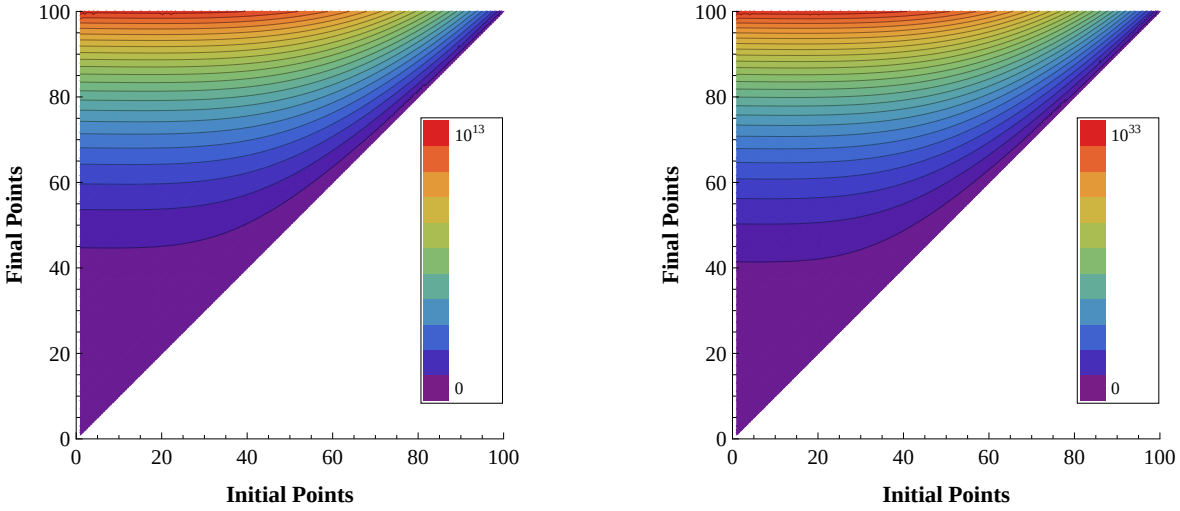
A question then becomes, how much does this cost for each additional training point? This is important if refinements are to be made to the metamodel to better approximate the true functional mappings. The optimization described in Section 5.5 has an estimator for the cost of finding an expected improvement, but not the number of times this must be computed to find the new training point. It can be assumed that this process takes a finite number of iterations to converge reasonably, which we designate as $\mathcal{N}(m,n)$ iterations. Using the same kernel configuration as in Eq.(5), the cost of the regression analysis as from a (the initial number) to b (the final number) training points as Eq.(35).

$$C_{cum}(a,b,n,\mathcal{N}) = \sum_{m=a}^{b} \left(C_{\text{b/s}} + \mathcal{N}(m,n)C_{\text{MLE}}\right) \tag{35}$$

If it is assumed that $\mathcal{N}(m,n) \to \mathcal{N}$, a constant, then the cumulative cost can be reduced. The constant $\mathcal{N}$ can be approximated from a grid sampling having a number of iterations taken to be constant based on the dimensionality and coarseness of a sampling grid. An example would be to have a Δ_i grid spacing for each dimension of the input space and to sample a uniform grid. The number of points is then given by Eq.(36).

$$\mathcal{N}(m,n) = \mathcal{N}(n) = \prod_{i=1}^{n} \frac{x_i^{\text{high}} - x_i^{\text{low}}}{\Delta_i} \tag{36}$$

If we assume that Δ_i is constant as well (i.e., each variable is sampled from the unit hypercube with density of 10, then $\mathcal{N}(n) = 10^n$. Fig.(3) shows a contourplot of the computational costs of Eq.(35) assuming that $C_e = 1.2, C_{\sqrt{}} = 1.4, C_\Phi = 117.7, C_\phi = 42.7$. Fig.(3) shows that to build a regression, find the next training point, and repeat (excluding the cost of running the true model) from, for example, 20 training points to 30 final points in 25-dimensions costs $\approx 8.3 \times 10^{30}$ operations, and at 1 gigaflop per second, this would take nearly 9.6×10^{16} days. Granted almost of this comes from the cost in finding the next training point with the expensive random search of 10^n per new point. Without the effort to find the new point, the computational cost is $\approx 4.3 \times 10^5$ which is less than $5 \times 10^{-24}\%$ of the total. This implies that an efficient means of finding the next point should be used in this method.



(a) 5D case. Note the color legend from $0 \to 10^{13}$

(b) 25D case. Note the color legend from $0 \to 10^{33}$

Figure 3: Notional computational cost for sequentially building, solving, and finding the next point for two different problem sizes based on random search optimization. These plots show the cost of the surrogate modeling but do not include the cost of running the 'true' model.

6. Discussion

The concern with surrogate modeling is that it is being used indiscriminately, and this paper has attempted to convey the computational cost of using these metamodels. The optimization problem was presented as a quadratic program with affine constraints that could be solved using linear algebra. Matrix inversions are the major components of solving for the coefficients as well as finding the next potential point to add to the training set. Inversion of a matrix is of order 3 with respect to dimensions, and the matrix in this problem is of size one plus the number of training points, which is not too expensive. Radial basis functions were used to assess the dimensionality of the problem. Radial basis functions require a distance metric that is linear with respect to the dimension of the data. The total computational cost is then of order 3 with respect to the number of training points and linear in the dimension of the problem.

This paper has only touched on the cost of finding the next point by estimating the cost of using a maximum expected improvement metric. The question of validation was addressed by supporting the cost of the regression against subsets of the problem. These two problems are fundamental to the generation and assessment of the regression but are left as smaller pieces of the art in using metamodels to explore and exploit data.

There are questions about how to find the next training point. Obviously the grid search is intractable but it remains to be determined what the most effective means of finding this point, be it from a better sampling schema, or a different objective formulation. In either case, a cost study needs to be taken to quantify each method's impact on the total cost of the process.

Future work should consider the formulation and optimization of the metamodel as well as its integration into the optimization paradigm. These metamodels can allow for expansive searches on a finite sample, but their utility is only as good as their training points and update schema. Some topics include: (1)

cost studies of various metamodel strategies including kriging and response surface, (2) 'smart' sampling schemes, (3) efficient optimization programs, and (4) the tractability of metamodling in the optimization and/or trade space exploration paradigm.

While this paper may show that it is relatively inexpensive to use metamodels (with efficient optimization routines), it is our goal to get scientists and engineers to think about the processes that they are using and to ask the question of whether it is necessary or tractable. There are more questions about using a stochastic metamodel as its parameters are, in general, not known *a priori* or via a simple heuristic that is solved through another optimization routine. The cost of these should be explored and compared to the deterministic radial basis method presented here in terms of cost and information generated.

7. Acknowledgements

The authors wish to acknowledge the support provided by the Defense Advanced Research Projects Agency (DARPA/TTO) under contract HR0011-12-C-0075 iFAB Foundry for making this work feasible. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of DARPA or the U.S. Government.

8. References

[**1**] Sacks, J., Welch, W.J., Mitchell, T. J., & H.P. Wynn. "Design and analysis of computer experiments." *Statistical Science* 4.4 (1989): 409-23.

[**2**] Jones, D.R., Shonlau, M. & W.J. Welch. "Efficient Global Optimization of Expensive Black-Box Functions." *Journal of Global Optimization* 13.4 (1998): 455-92.

[**3**] Chaudhuri, A., Haftka, R.T., & L.T. Watson. "How to Decide Whether to Run One More Cycle in Efficient Global Optimization." 14$^{\text{th}}$ AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference. Indianapolis, IN, USA. AIAA 2012-5668. 2012.

[**4**] Hastie, T.J., Tibshirani, R.J., & J.H. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* New York, NY, USA. Springer, 2009.

[**5**] Martin, J.D. & T.W. Simpson. "A Study on the Use of Kriging Models to Approximate Deterministic Computer Models." *2003 ASME Design Engineering Technical Conference.* Chicago, IL, USA. DETC2003/DAC48762. 2003.

[**6**] Smola, Alex J., "Regression estimation with support vector learning machines", Master's Thesis, *Technische Universitä München.* 1996.

[**7**] Aronszajn, Nachman. "Theory of Reproducing Kernels." *Transactions of the American Mathematical Society* 68.3 (1950): 337-404.

[**8**] Gunn, Steve R. "Support Vector Machines for Classification and Regression." Image Speech and Intelligent Systems Research Group, University of Southampton. `http://ce.sharif.ir/courses/85-86/2/ce725/resources/root/LECTURES/SVM.pdf/`. 10 May 1998. Accessed 02 April 2013.

[**9**] Wahba, Grace. *Spline Models for Observational Data.* Series in Applied Mathematics. Vol. 59. Society for Industrial and Applied Mathematics. Philadelphia, PA, USA. 1990.

[**10**] Girosi, Federico. "An Equivalence between Sparse Approximation and Support Vector Machines." *Neural Computation* 10.6 (1998): 1455-1480.

[**11**] Heckman, Nancy. "The Theory and Application of Penalized Least Squares Methods Or Reproducing Kernel Hilbert Spaces Made Easy." *Statistics Surveys* 6 (2012): 113-41.

[**12**] Paláncz, B., Völgyesi, L. & G. Popper, "Support Vector Classifier via *Mathematica*", Wolfram Research *Mathematica* Information Center, 2005, `http://library.wolfram.com/infocenter/MathSource/5270/`

[**13**] Forrester, A.I.J., & A.J. Keane. "Recent Advances in Surrogate-Based Optimization." *Progress in Aerospace Sciences* 45.1 (2009): 50-79.

[**14**] Smola, A.J., & B. Schölkopf. "A Tutorial on Support Vector Regression." *Statistics and Computing* 14.3 (2004): 199-222.

[**15**] Cristianini, N. & J. Shawe-Taylor. *An Introduction to Support Vector Machines: And Other Kernel-Based Learning Methods.* New York: Cambridge, University Press, 2000.

[**16**] Vapnik, Vladimir. *Statistical Learning Theory.* New York: Wiley, 1998.