**This is a preprint of the following article, which is available from** `http://mdolab.engin.umich.edu`
John T. Hwang and J. R. R. A. Martins. A fast-prediction surrogate model for large datasets. *Aerospace Science and Technology*, 2018. (In press). doi:10.1016/j.ast.2017.12.030
**The published article may differ from this preprint, and is available by following the DOI above.**

# A fast-prediction surrogate model for large datasets

John T. Hwang[1] and Joaquim R. R. A. Martins[2]

**Abstract** Surrogate models approximate a function based on a set of training points and can then predict the function at new points. In engineering, kriging is widely used because it is fast to train and is generally more accurate than other types of surrogate models. However, the prediction time of kriging increases with the size of the dataset, and the training can fail if the dataset is too large or poorly spaced, which limits the accuracy that is attainable. We develop a new surrogate modeling technique—regularized minimal-energy tensor-product splines (RMTS)—that is not susceptible to training failure, and whose prediction time does not increase with the number of training points. The improved scalability with the number of training points is due to the use of tensor-product splines, where energy minimization is used to handle under-constrained problems in which there are more spline coefficients than training points. RMTS scales up to four dimensions with 10–15 spline coefficients per dimension, but scaling beyond that requires coarsening of the spline in some of the dimensions because of the computational cost of the energy minimization step. Benchmarking using a suite of one- to four-dimensional problems shows that while kriging is the most accurate option for a small number of training points, RMTS is the best alternative when a large set of data points is available or a low prediction time is desired. The best-case average root-mean-square error for the 4-D problems is close to 1% for RMTS and just under 10% for kriging.

**Keywords:** Surrogate modeling; response surfaces; metamodels; regression; multidisciplinary design optimization; Krylov methods

## 1 Introduction

A surrogate model is an approximation that is cheaper or more convenient to evaluate than the underlying model it approximates. The most common use of surrogate models is to replace a known expensive computational model when a large number of repeated evaluations is required, e.g., for optimization or uncertainty quantification. Another common application is when we want to obtain a continuous function from a fixed dataset, e.g., when the data is obtained experimentally or from legacy code. A third application is smoothing an underlying model with a lower order of continuity, perhaps to achieve differentiability for gradient-based optimization (Hwang et al., 2014).

In discussions of surrogate models, it is beneficial to separate the construction and evaluation of the model, because most such models have parameters that are precomputed during the construction stage. Here, we refer to the evaluation of the model as *prediction*. Given $n_x$ inputs and $n_w$ parameters, the prediction is the evaluation of

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

where $\mathbf{x} \in \mathbb{R}^{n_x}$ is an input vector, $y \in \mathbb{R}$ is the output variable, and $\mathbf{w} \in \mathbb{R}^{n_w}$ is the vector of model parameters. We refer to the construction of the model as *training*, and thus to the dataset as *training points*. The objective of training is to compute the model parameters $\mathbf{w}$ that satisfy or approximate

$$\bar{y}_i \approx f(\bar{\mathbf{x}}_i, \mathbf{w}), \qquad \forall 1 \leq i \leq n_t \tag{2}$$

where $(\bar{\mathbf{x}}_1, \bar{y}_1), \ldots, (\bar{\mathbf{x}}_{n_t}, \bar{y}_{n_t})$ are the $n_t$ training points.

[1]Peerless Technologies Corporation, Beavercreek, OH 45324.
[2]Department of Aerospace Engineering, University of Michigan, Ann Arbor, MI 48109.

Because of its wide applicability and usefulness, surrogate modeling has been a topic of active research for decades Simpson et al. (2001a). In engineering, kriging is one of the most commonly used methods for several reasons Simpson et al. (2001b); Kleijnen (2009). First, it is the most accurate method overall for small or moderate numbers of training points ($n_t < 10^3$), as we confirm in Section 4. Second, kriging training and prediction times scale well with the number of dimensions, $n_x$, enabling its use in high-dimensional problems where $n_x$ can be as high as $\mathcal{O}(10^2)$. Third, its stochastic interpretation provides an estimate of the prediction error via the variance of the prediction point. However, the disadvantages of kriging include the increase in prediction time with the number of training points, and the propensity of the training to fail when the training points are too close to each other. These disadvantages limit the maximum number of training points that kriging can handle, which in turn limits the accuracy that can be achieved when many training points are available.

In this paper, we are primarily motivated by applications in which the surrogate model is a part of a larger model. For example, the surrogate model might approximate aircraft aerodynamic performance with respect to the flight conditions, where the surrogate is a part of a multidisciplinary model that includes other disciplines represented by other models that may or may not be surrogates. If the set of training points is fixed, then the surrogate model can be trained once in advance and used repeatedly each time the multidisciplinary model is run. This is the case in many problems in multidisciplinary design optimization (MDO), and it encourages emphasizing prediction time more than training time. Predictions are made repeatedly to converge the multidisciplinary system, which in turn is done once per optimization iteration. In some problems, this can lead to millions of predictions for a surrogate model trained once (Hwang and Martins, 2016). Other applications, such as surrogate-based optimization, place more weight on a lower training time because the training occurs at every optimization iteration.

We develop a new surrogate modeling method for low-dimensional problems ($n_x \leq 4$) that we call *regularized minimal-energy tensor-product splines* (RMTS). RMTS is generally slower to train than kriging, but it has a fast prediction time that does not increase with the number of training points. Moreover, it can work with much larger numbers of training points, meaning that when large datasets are available, e.g., when the data source is a fast but nondifferentiable model, the accuracy that can be achieved with RMTS is expected to be higher than with kriging, as we show in Section 4. Interest in tensor-product splines has declined in the last few decades because of their poor scaling with $n_x$; however, modern computing hardware mitigates these scaling limitations and enables RMTS to scale up to four-dimensional problems. Moreover, tensor-product splines enable prediction that is orders of magnitude faster than kriging when the number of training points is large ($n_t > 10^4$). RMTS uses energy minimization and regularization to improve accuracy with small datasets and to handle unstructured datasets, i.e., training points not arranged in a structured grid.

RMTS is available under an open-source license as part of the *surrogate modeling toolbox* (SMT)[1]. All the benchmarking problems, as well as the other surrogate modeling approaches considered in this paper, are included in the SMT repository, so our results are fully reproducible.

The paper is organized as follows. In Section 2, we review some of the surrogate modeling methods that are commonly used in engineering: polynomials, splines, artificial neural networks, support-vector regression, inverse-distance weighting, radial basis functions, and kriging. In Section 3, we present the equations and solution algorithms of RMTS. In Section 4, we use a benchmarking suite to evaluate RMTS and to compare the surrogate modeling methods in terms of training time, prediction time, and accuracy. We also discuss the use of RMTS in a practical MDO context dealing with aircraft mission optimization.

## 2 Review of surrogate modeling methods

In engineering, a surrogate model is also known as a *response surface* in some contexts, or as a *metamodel*, reflecting the idea that it is a model of an underlying model. In this paper, we use *surrogate model* throughout to remain consistent, while noting that different terms are used in other contexts.

Surrogate modeling approaches can be classified as *interpolation* (if the surrogate model matches the true function value at each point in the training dataset) or *regression* (if it does not). Regression methods smoothly approximate noisy data, and they include polynomials, splines, artificial neural networks (ANN), and support vector regression (SVR). Interpolation methods attempt to smoothly and accurately fit non-noisy data, and they include inverse distance weighting (IDW), radial basis functions (RBFs), and kriging.

These methods are extensively discussed in the literature Wang and Shan (2007); Simpson et al. (2008); Forrester et al. (2008).

Since RMTS is classified as an interpolation method, we review the regression methods briefly and explain the interpolation methods in more detail. Section 4 presents results comparing RMTS to IDW, RBFs, and kriging, so we also present the equations for each, in the form they are implemented for the benchmarking.

## 2.1 Regression methods

### 2.1.1 Polynomial regression

Polynomial regression uses low-order global polynomials in multiple variables to approximate the training data. Polynomial response surfaces were originally introduced by Box and Wilson (1951). They have the advantage of simplicity, making them fast and easy to work with. However, they lack flexibility, and therefore for many types of problems they are less accurate than other methods.

### 2.1.2 Splines

The most successful surrogate modeling method using splines is multivariate adaptive regression splines (MARS), developed by Friedman (1991). MARS uses basis functions that are piecewise linear in each dimension and adaptively splits the basis functions using a greedy algorithm. MARS scales well with problem dimension ($n_x$), but the downside is that both the training and prediction times increase with the number of knots, which is tied to accuracy.

### 2.1.3 Artificial neural networks

ANNs work with an interconnected set of nodes that compute an activation signal based on inputs, just as neurons in the brain fire based on impulses. These nodes are arranged in layers, where one layer consists of the $n_x$ inputs, another layer consists of the $n_y$ outputs, and the remaining layers are known as *hidden layers*. Compared to the typical surrogate modeling techniques in engineering, neural networks display slower convergence of error versus the number of training points. On the other hand, they are capable of dealing with significantly higher-dimensional problems, such as speech and character recognition.

### 2.1.4 Support vector regression

SVR Cortes and Vapnik (1995) also has its roots in machine learning, but it has been successful as a method for surrogate modeling in engineering applications. It is typically derived first as an optimization problem that finds the most "flat" linear approximation with a prescribed precision. The dual problem yields an equivalent form with a dot product between the input vectors in the objective function, and replacing this dot product with another function leads to the general SVR method. Choosing the Gaussian function turns out to be similar to RBFs with a Gaussian kernel, except that it performs regression with a prescribed tolerance rather than interpolation.

## 2.2 Interpolation methods

### 2.2.1 Inverse distance weighting

IDW, also known as Shepard's method (Shepard, 1968), uses a linear combination of the training outputs, where the coefficients are computed from the inverse of the distance from the prediction point to each training point. It exactly interpolates unstructured data in $n$-dimensions while being continuous and differentiable everywhere.

The original IDW interpolant is given by

$$f(\mathbf{x}, \mathbf{w}) = \begin{cases} \bar{y}_i, & \mathbf{x} = \bar{\mathbf{x}}_i \text{ for some } i \\ \frac{\sum_{i=1}^{n_t} d(\mathbf{x}, \bar{\mathbf{x}}_i)^{-p} \bar{y}_i}{\sum_{i=1}^{n_t} d(\mathbf{x}, \bar{\mathbf{x}}_i)^{-p}}, & \text{otherwise} \end{cases} \tag{IDW}$$

where $p > 1$ is an option. Although the distance function is not defined when the evaluation point is one of the training points, the overall IDW function is still continuous with this definition, and so are the derivatives as long as $p > 1$. This definition has been modified to use sparse functions (Renka, 1988) and spatially varying adaptive functions (Lu and Wong, 2008).

IDW is unique because it does not require any training. Therefore, its advantage is that it can handle very large numbers of training points (we give results for up to $n_t = 10^5$). Another advantage is that the interpolant stays within the minimum and maximum of the training points on the entire input space (Gordon and Wixom, 1978). However, IDW is slow to predict because it computes the distances from the evaluation point to all the training points. A more serious disadvantage is that the derivatives of the interpolant vanish at each training point, so IDW yields an undesirable interpolant even in the simple 1-D case when the training points are on a line.

### 2.2.2 Radial basis functions

RBFs are defined as linear combinations of basis functions, where each basis function depends on the distance from the prediction point to each training point. The coefficients in the linear combination are determined by solving a linear system that is typically dense. The radially varying basis functions are usually augmented with polynomial functions to capture the general trends.

In presenting the RBF equations, we use $p_i : \mathbb{R}^{n_x} \to \mathbb{R}$ to denote the $i$th of $n_p$ polynomial functions and $\phi_i : \mathbb{R}^{n_x} \to \mathbb{R}$ to denote the $i$th of $n_t$ basis functions. Prediction using RBFs is given by

$$f(\mathbf{x}, \mathbf{w}) = \sum_{i=1}^{n_p} p_i(\mathbf{x}) a_i + \sum_{i=1}^{n_t} \phi_i(\mathbf{x}) b_i \tag{RBF}$$

where $a_i \in \mathbb{R}$ is the weight for the $i$th polynomial trend function and $b_i \in \mathbb{R}$ is the weight for the $i$th basis function. Therefore, $\mathbf{w}$ consists of the coefficients $a_1, \ldots, a_{n_p}$ and $b_1, \ldots, b_{n_t}$.

In matrix notation, the prediction equation is

$$\mathbf{y} = \mathbf{P}\mathbf{a} + \mathbf{\Phi}\mathbf{b}, \tag{3}$$

where $\mathbf{y} \in \mathbb{R}^n$, $\mathbf{P} \in \mathbb{R}^{n \times n_p}$, $\mathbf{a} \in \mathbb{R}^{n_p}$, $\mathbf{\Phi} \in \mathbb{R}^{n \times n_t}$, and $\mathbf{b} \in \mathbb{R}^{n_t}$.

The training for RBFs consists in solving the linear system

$$\begin{bmatrix} \bar{\mathbf{\Phi}} & \bar{\mathbf{P}} \\ \bar{\mathbf{P}}^T & 0 \end{bmatrix} \begin{bmatrix} \mathbf{b} \\ \mathbf{a} \end{bmatrix} = \begin{bmatrix} \bar{\mathbf{y}} \\ 0 \end{bmatrix}, \tag{4}$$

where $\bar{\mathbf{\Phi}} \in \mathbb{R}^{n_t \times n_t}$ is the evaluation of the basis functions at $\bar{\mathbf{x}}_1, \ldots, \bar{\mathbf{x}}_{n_t}$, $\bar{\mathbf{P}} \in \mathbb{R}^{n_t \times n_p}$ is the evaluation of the polynomial functions at $\bar{\mathbf{x}}_1, \ldots, \bar{\mathbf{x}}_{n_t}$, and $\bar{\mathbf{y}} \in \mathbb{R}^{n_t}$ is the training point output values in matrix form. The linear system (4) is derived by performing an initial weighted least-squares approximation using the polynomials to capture the general trends before applying the basis functions to the resulting error. The least-squares system is $\bar{\mathbf{P}}\mathbf{a} = \bar{\mathbf{y}}$, and the associated normal equations are $\bar{\mathbf{P}}^T \bar{\mathbf{\Phi}}^{-1} \bar{\mathbf{P}}\mathbf{a} = \bar{\mathbf{P}}^T \bar{\mathbf{\Phi}}^{-1} \bar{\mathbf{y}}$. The second step solves the linear system $\bar{\mathbf{\Phi}}\mathbf{b} = \bar{\mathbf{y}} - \bar{\mathbf{P}}\mathbf{a}$.

There are many choices for the basis functions, but the most common ones are the Gaussian,

$$\phi_i(\mathbf{x}) = \exp\left(-\frac{||\mathbf{x} - \bar{\mathbf{x}}_i||^2}{r_0^2}\right), \tag{5}$$

thin plate splines (Duchon, 1977),

$$\phi_i(\mathbf{x}) = ||\mathbf{x} - \bar{\mathbf{x}}_i||^2 \ln ||\mathbf{x} - \bar{\mathbf{x}}_i||, \tag{6}$$

and the multiquadric (Hardy, 1971),

$$\phi_i(\mathbf{x}) = \sqrt{||\mathbf{x} - \bar{\mathbf{x}}_i||^2 + r_0^2}, \tag{7}$$

where $r_0$ is the tuning parameter. Thin plate splines have the advantage that they have no parameters that need to be tuned, and they are derived by analytically solving for the deformation of a thin elastic plate under bending. For multiquadrics, there have been studies on optimally choosing the $r_0$ parameter to improve the interpolant's accuracy (Carlson and Foley, 1991; Kansa and Carlson, 1992).

RBFs are very accurate for many problems, and they have a low training time for small datasets. However, one disadvantage of RBFs is that they are highly susceptible to spurious oscillations with data that is rapidly

changing or nonuniformly distributed in the input space. As the number of training points increases, the prediction time also increases, and the matrix in Eq. (4) can become ill-conditioned and cause training failures. Another disadvantage is the presence of problem-dependent tuning parameters. As we see in the next section, kriging uses the same underlying equations as RBFs, but kriging's equivalents of the basis functions are tuned in an automated way.

To improve accuracy, anisotropic basis functions—ones that do not only depend radially—have been studied extensively in the literature (Mitášová and Mitáš, 1993; Dinh et al., 2001; Casciola et al., 2007, 2010; Beatson et al., 2010; Yu and Turk, 2013). Another modification to RBFs is the use of compactly supported basis functions to achieve sparsity for better efficiency (Wendland, 1995; Wu, 1995; Buhmann, 2000). Compact basis functions improve the evaluation efficiency somewhat, and optimally chosen basis functions improve accuracy. However, the primary disadvantage of RBFs remains: they are not as robust as interpolants based on energy minimization, especially with data points that vary significantly in spacing or in output value.

### 2.2.3 Kriging

Kriging, also known as Gaussian process regression (GPR), can be seen as a generalization of the RBF approach, as we will show. However, kriging interprets the interpolation function as a random process, where the function value at each point in the domain is treated as a separate random variable that is correlated to all the others. Kriging was named after Krige (1951) by Matheron (1963). It was initially developed independently and in various forms in many fields (Cressie, 1990), most notably in geostatistics, but after Sacks et al. (1989) it became widely used in computational modeling and design.

Kriging incorporates general trend models in a manner similar to the global polynomials that are used for RBFs. *Simple kriging* uses a known mean; *ordinary kriging* uses a constant but unknown mean; and *universal kriging* assumes a general polynomial mean. Universal kriging is used to improve accuracy at the expense of a slight increase in computational complexity, but numerical experiments show that this is not universally the case (Zimmerman et al., 1999).

We use the notation of RBFs for the polynomial trend functions, $p_i : \mathbb{R}^{n_x} \to \mathbb{R}$, and we denote the correlation between the prediction point and the $i$th training point as $\psi_i : \mathbb{R}^{n_x} \to \mathbb{R}$. Then, the prediction equations for universal kriging are

$$f(\mathbf{x}, \mathbf{w}) = \sum_{i=1}^{n_p} p_i(\mathbf{x}) a_i + \sum_{i=1}^{n_t} \psi_i(\mathbf{x}) b_i \tag{8}$$

where $a_i \in \mathbb{R}$ is the weight for the $i$th polynomial trend function and $b_i \in \mathbb{R}$ is the weight for the $i$th training point. Therefore, $\mathbf{w}$ consists of the weights $a_1, \ldots, a_{n_p}$ and $b_1, \ldots, b_{n_t}$.

In matrix notation, the prediction equation is

$$\mathbf{y} = \mathbf{P}\mathbf{a} + \mathbf{\Psi}\mathbf{b}, \tag{9}$$

where $\mathbf{y} \in \mathbb{R}^n$, $\mathbf{P} \in \mathbb{R}^{n \times n_p}$, $\mathbf{a} \in \mathbb{R}^{n_p}$, $\mathbf{\Psi} \in \mathbb{R}^{n \times n_t}$, and $\mathbf{b} \in \mathbb{R}^{n_t}$.

The weights are computed by solving the linear system

$$\begin{bmatrix} \bar{\mathbf{\Psi}} & \bar{\mathbf{P}} \\ \bar{\mathbf{P}}^T & 0 \end{bmatrix} \begin{bmatrix} \mathbf{b} \\ \mathbf{a} \end{bmatrix} = \begin{bmatrix} \bar{\mathbf{y}} \\ 0 \end{bmatrix}, \tag{10}$$

which is identical to that for RBFs except that we have correlation functions $\psi$ instead of radial basis functions $\phi$.

Similarly to RBFs, we can derive Eq. (10) by applying a two-step process. The first step and the associated computation of $\mathbf{a}$ are identical to the RBF case—the coefficients of the polynomial functions can be found via a weighted least-squares solution. The second step and the computation of $\mathbf{b}$ follow from first deriving $\mathbf{y} - \mathbf{P}\mathbf{a} = \mathbf{\Psi}\bar{\mathbf{\Psi}}^{-1}(\bar{\mathbf{y}} - \bar{\mathbf{P}}\mathbf{a})$ (see A), which can easily be shown to solve Eq. (10).

Where kriging departs from RBFs is in the selection of the covariance function $\psi$. Kriging assumes a general form for $\psi$ and computes the parameters via optimization of the likelihood function. The most commonly used form, as expressed by Toal et al. (2008), is

$$\psi_i(\mathbf{x}) = \sigma^2 \exp\left( -\sum_{k=1}^{n_x} \theta_k ||x_k - \bar{x}_{i,k}||^{p_k} \right), \tag{11}$$

where $\sigma^2$ is the variance, and $\theta_1, \ldots, \theta_{n_x}$ and $p_1, \ldots, p_{n_x}$ are called *hyperparameters*.

The optimal hyperparameters for a given set of training data can be found by computing the *maximum likelihood estimator* (MLE) through optimization. In this context, the likelihood function takes as inputs the values of the hyperparameters and returns the probability that the random variables take on the actual output variable values at the training points. Therefore, we tune the values of the hyperparameters to maximize this likelihood. More details on the MLE for hyperparameter tuning can be found in (Toal et al., 2008).

The automation and optimization aspects of the MLE approach are attractive, but the correlation matrix can be susceptible to ill-conditioning, and the log of the likelihood function, which is used as the objective, can be multi-modal (Martin and Simpson, 2005). Two alternative approaches to MLE exist: fitting the covariance function to an empirically determined variogram, and "leave-one-out" cross-validation (Mitchell and Morris, 1992). Cross-validation avoids the numerical issues of maximizing the log-likelihood function, but it can be inconsistent and sometimes performs much worse (Martin and Simpson, 2005).

In summary, kriging has many advantages, but there are numerical issues for certain types of problems that limit its robustness. In terms of advantages, kriging scales well with dimensionality ($n_x$), and it has the attractive property that training points that are part of a dense cluster of points have lower weighting in general than those that are located in a sparse area of the input space. In terms of disadvantages, the size of the linear systems that kriging solves is tied to the number of training points, and for large problems this can be a bottleneck. This has motivated research into kriging using high-performance computing and parallelism (Kerry and Hawick, 1998), fixed-rank approximations using the Sherman–Morrison–Woodbury formula (Cressie and Johannesson, 2008), and approximation of the inverse of the covariance matrix Leithead and Zhang (2007) using the BFGS update formula. Another disadvantage is that computing the kriging hyperparameters can be a challenge. Determining the empirical variogram is a manual process that is not always robust, and MLE can fail to converge, especially with problems with more than $\mathcal{O}(100)$; we observed this in numerical experiments. The alternative is to manually select and tune the covariance function, in which case the method is no different from RBF interpolation.

## 3 Method

We now develop the RMTS surrogate model. It is similar to IDW, RBFs, and kriging in its representation of the prediction output as a linear combination of the training outputs. Like that for RBFs and kriging, the prediction output for RMTS can also be interpreted as a linear combination of basis functions, where the coefficients are precomputed during training. However, unlike RBFs and kriging, RMTS does not require the number of basis functions to be tied to the number of training points.

### 3.1 General form

The prediction equations for RMTS are given by

$$f(\mathbf{x}, \mathbf{w}) = \sum_{i_1,\ldots,i_{n_x} \in \mathcal{I}} b_{1,i_1}(x_1) \cdots b_{n_x,i_{n_x}}(x_{n_x}) w_{i_1,\ldots,i_{n_x}} \tag{12}$$

where $b_{k,i} : \mathbb{R} \to \mathbb{R}$ is the $i$th basis function for the $k$th input variable and $w_{i_1,\ldots,i_{n_x}}$ is a coefficient of the multivariate spline.

In matrix notation, the prediction equation is

$$\mathbf{y} = \mathbf{F}\mathbf{w}, \tag{13}$$

where $\mathbf{F} \in \mathbb{R}^{n \times n_w}$ and $\mathbf{w} \in \mathbb{R}^{n_w}$. Since splines are piecewise polynomials, each coefficient affects only part of the domain; therefore, each basis function has finite support and $\mathbf{F}$ is sparse.

The RMTS methodology permits any type of splines to be used. However, splines with uniform knot spacing have a lower prediction time. If the spline knot spacing is not uniform, it is necessary to perform a search for each prediction point and each of the $n_x$ dimensions to locate the knot interval containing the input value, which increases the prediction time. For this paper and in the SMT package, two types of splines have been implemented: B-splines with uniform knot vectors and cubic Hermite splines.

We call the B-spline implementation *RMTB*, where the *B* stands for *B-splines*. In the SMT implementation of RMTB, the B-spline order and number of control points can be arbitrarily chosen, as long as

the number of control points is not less than the order. With B-splines of higher order, better accuracy is typically achieved but the training and prediction times are much higher because $\mathbf{F}$ is less sparse. Typically, third- or fourth-order B-splines are used. B-splines have higher smoothness than cubic Hermite splines; the highest order of derivatives that are continuous at the knots is two less than the order of the B-spline.

We call the cubic Hermite spline implementation *RMTC*, where the *C* stands for *cubic Hermite splines*. In the SMT implementation of RMTC, the only option is the number of cubic elements in each dimension. Each cubic element is an $n_x$-dimensional hypercube. Its $4^{n_x}$ multivariate polynomial coefficients are uniquely defined by the $2^{n_x}$ interpolant values and derivatives at its $2^{n_x}$ nodes. At each node, the same values and derivatives are shared between all elements that share the node, so we are guaranteed $C^1$ continuity.

### 3.2 Constrained energy minimization

Conventional tensor-product spline interpolants require the number of coefficients, i.e., the size of $\mathbf{w}$, to be equal to the number of training points and for the training data to be arranged in a structured grid. Otherwise, the problem of computing the coefficients, $\mathbf{w}$, can be over- or under-constrained. RMTS uses more coefficients than necessary (more control points for RMTB or elements for RMTC) and minimizes the integral of the sum of squares of the second derivatives, i.e., the bending energy of the function. This idea draws inspiration from the behavior of an elastic plate or surface that deforms to pass through a set of points. Since the output variable is interpreted as the transverse displacement, the second derivatives represent the curvature of the plate, and the squares of the second derivatives quantify the bending energy stored in the material when loads are applied to achieve the desired deformation.

The idea of minimizing energy has been used extensively in the field of computer graphics for at least half a century (Schweikert, 1966; Lee and Forsythe, 1973). In these contexts, the term *minimal-energy splines* is used interchangeably with variational splines, faired splines, thin plate splines, and splines in tension, although there are small differences in meaning. Depending on the application, the kernel function can be 1-D cubic splines Brunnett and Kiefer (1994), 2-D bicubic splines Zhang et al. (2001), Bezier surfaces Moreton and Séquin (1992), B-spline surfaces Welch and Witkin (1992), triangular meshes Kobbelt (2000), or other more advanced geometry representations such as Catmull–Clark splines Halstead et al. (1993). In each of these studies, an energy functional is minimized to determine the coefficients or parameters of a curve or surface. Energy minimization has also been used for specific applications such as extending a curve or surface with additional points Mo and Zhao (2006), providing a framework for incorporating monotonicity constraints Wolberg and Alfy (2002), and inserting points in the domain Vassilev (1996).

Here, the continuous functional that is minimized is

$$\int_{a_{(n_x)}}^{b_{(n_x)}} \dots \int_{a_{(1)}}^{b_{(1)}} \left[ \left( \frac{\partial^2 f}{\partial x_1^2} \right)^2 + \dots + \left( \frac{\partial^2 f}{\partial x_{n_x}^2} \right)^2 \right] dx_{(1)} \dots dx_{(n_x)}. \tag{14}$$

The cross-derivatives are included in other contexts, but assembly of this functional can be expensive, and in numerical experiments including the additional terms did not provide a significant benefit.

For convenience, we introduce $\mathbf{H} \in \mathbb{R}^{n_w \times n_w}$ to represent the discretized matrix for the continuous energy functional (14). We also introduce $\bar{\mathbf{F}}$ such that $\bar{\mathbf{y}} = \bar{\mathbf{F}}\mathbf{w}$. Then, the constrained optimization problem that minimizes the energy subject to the training point constraints is

$$\begin{aligned} \min_{\mathbf{w}} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{H}\mathbf{w} \\ s.t. \quad & \bar{\mathbf{y}} = \bar{\mathbf{F}}\mathbf{w}. \end{aligned} \tag{15}$$

We use regularization to ensure that the matrix in the linear system is invertible. We add to the Lagrangian the norm of the degree-of-freedom vector with $\beta$ as the coefficient, and we subtract from it the norm of the Lagrange multipliers with $\alpha$ as the coefficient. Figure 1 shows the resulting unconstrained minimization problem and the linear system that provides necessary conditions for a stationary point.

### 3.3 Reformulation using the Schur complement

We now reformulate Eq. (16) to reduce the size of the system and obtain a more flexible formulation. Since the linear system in Fig. 1 has a block structure, the $\alpha$ regularization allows us to apply block Gaussian

$$\min_{\mathbf{w},\lambda} \quad \tfrac{1}{2}\mathbf{w}^T\mathbf{H}\mathbf{w} + \lambda^T\left(\bar{\mathbf{F}}\mathbf{w} - \bar{\mathbf{y}}\right) + \tfrac{1}{2}\beta\mathbf{w}^T\mathbf{w} - \tfrac{1}{2}\alpha\lambda^T\lambda \tag{16}$$

$$\begin{bmatrix} \mathbf{H} + \beta\mathcal{I} & \bar{\mathbf{F}}^T \\ \bar{\mathbf{F}} & -\alpha\mathcal{I} \end{bmatrix} \begin{bmatrix} \mathbf{w} \\ \lambda \end{bmatrix} = \begin{bmatrix} 0 \\ \bar{\mathbf{y}} \end{bmatrix}$$

Figure 1: Regularized unconstrained minimization problem with the method of Lagrange multipliers already applied to the equality constraints (left), and the resulting block linear system (right).

$$\min_{\mathbf{w}} \quad \tfrac{1}{2}\mathbf{w}^T\mathbf{H}\mathbf{w} + \tfrac{1}{2}\beta\mathbf{w}^T\mathbf{w} + \tfrac{1}{2}\tfrac{1}{\alpha}\left(\bar{\mathbf{F}}\mathbf{w} - \bar{\mathbf{y}}\right)^T\left(\bar{\mathbf{F}}\mathbf{w} - \bar{\mathbf{y}}\right) \tag{17}$$

$$\left[\mathbf{H} + \beta\mathcal{I} + \tfrac{1}{\alpha}\bar{\mathbf{F}}^T\bar{\mathbf{F}}\right]\mathbf{w} = \tfrac{1}{\alpha}\bar{\mathbf{F}}^T\bar{\mathbf{y}}$$

Figure 2: Minimization of a weighted combination of the bending energy term and the least-squares training data approximation term (left), and the resulting linear system (right).

elimination. In the linear system that results, the matrix is then the Schur complement, as we see in Fig. 2. The new optimization problem in Fig. 2 is an unconstrained minimization of the sum of the energy term, a least-squares approximation term, and regularization terms. This contrasts with the optimization problem in Fig. 1, which is an energy minimization subject to training-data constraints with regularization.

The size of the linear system is reduced but we lose sparsity through this reformulation. Both $\mathbf{H}$ and $\bar{\mathbf{F}}$ are sparse, so the matrix in Fig. 1 is sparse, but the product $\bar{\mathbf{F}}^T\bar{\mathbf{F}}$ causes us to lose sparsity in Fig. 2. However, numerical experiments show that the matrix assembly time and the linear solution time are both lower with the smaller, denser matrix.

Another advantage is that this new formulation is more versatile. As we see in Section 4, energy minimization is not necessary when the number of training points is large (say, $n_t > 10^3$). This formulation allows us to exclude the energy minimization terms easily, without changing the structure of the linear system; this is an option in the implementation of RMTS in SMT. The reformulated problem can be interpreted as a weighted combination of energy minimization and least squares approximation with regularization.

### 3.4 Nonlinear reformulation

In problems with few training points and a smooth function, energy minimization significantly increases the accuracy of the interpolant. However, it can increase the error in fitting the training points when the true function has large variations in curvature, especially with a large number of training points.

Reformulating the problem once more to make the problem nonlinear improves accuracy in these situations. This is done by generalizing the training point error term to powers greater than 2. Thus, we replace $(\bar{\mathbf{F}}\mathbf{w} - \bar{\mathbf{y}})^T(\bar{\mathbf{F}}\mathbf{w} - \bar{\mathbf{y}})$ by $\sum_i(\bar{\mathbf{F}}\mathbf{w} - \bar{\mathbf{y}})_i^p$, where $p > 2$. With this change, the objective function is no longer quadratic, so multiple iterations of Newton's method must be performed, as shown in Fig. 3.

We can interpret this reformulation in two ways. First, the nonlinear approach in Fig. 3 can be interpreted as repeatedly solving the original problem of weighted energy minimization and least squares approximation (17), but with the approximation terms weighted by the error from the previous iteration. This can be viewed as repeatedly solving the original problem (17), where the weights are updated each time to place more emphasis on the training points that were not well-approximated in the previous iteration.

Second, the nonlinear reformulation can be interpreted as making our approach closer to the minimization

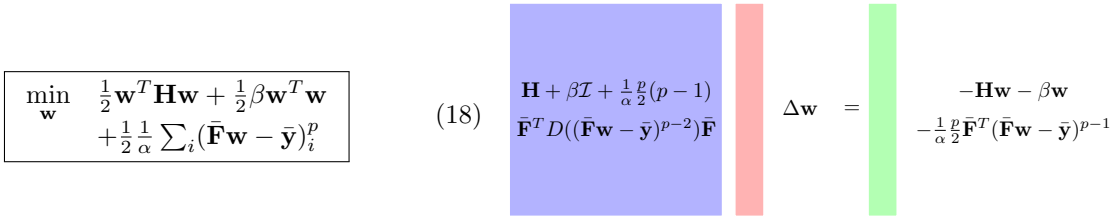


Figure 3: Reformulation of the weighted energy and least squares minimization (17) so that the optimality conditions result in a nonlinear system (left), shown along with the resulting Newton system (right).

of the largest error among the training points. Since the $p$-norm approaches the max-norm as $p$ approaches infinity, increasing the value of $p$ above 2 equalizes the error and results in a better overall fit.

In numerical experiments, the nonlinear reformulation either improves or matches the accuracy of the original formulation (17). However, it increases the training time because each nonlinear iteration involves solving a linear system.

In our implementation of RMTS in SMT, the number of nonlinear iterations is a user-specified option. The matrices are assembled and solved using sparse linear algebra via the *NumPy* and *SciPy* packages in Python. Since the matrix is always symmetric, we use the conjugate gradient (CG) method with a maximum of 100 iterations by default. For most problems, the matrix assembly time is larger than the linear solution time, which motivates us to perform a larger number of CG iterations. In problems in which the matrix is poorly conditioned, the nonlinear formulation provides another benefit—if the linear system is not adequately converged in the first nonlinear iteration, the linear systems in subsequent nonlinear iterations have tighter convergence. As a result, the nonlinear form of RMTS is robust, and the linear solver parameters did not require tuning in any of the numerical experiments. All of the benchmarking results shown in the next section were obtained using the default settings.

### 3.5 Summary and implementation

In this section, we summarize the training procedure for RMTS and describe its practical implementation. We first present a slight modification of Eq. (17) and Eq. (18) that is more convenient to use, and we describe the RMTS implementation in algorithmic form.

First, we multiply the objective function by $\alpha$, and replace the parameters $\alpha$ and $\beta$ with $\alpha' = \alpha$ and $\beta' = \alpha\beta$. We define $\alpha'$ and $\beta'$ in this way so that they represent scaling factors for the energy term ($\mathbf{H}$) and regularization term ($\mathcal{I}$), respectively. With this small reformulation, the quadratic optimization problem in Eq. (17) becomes

$$\min_{\mathbf{w}} \quad \frac{1}{2}\alpha'\mathbf{w}^T\mathbf{H}\mathbf{w} + \frac{1}{2}\beta'\mathbf{w}^T\mathbf{w} + \frac{1}{2}\left(\bar{\mathbf{F}}\mathbf{w} - \bar{\mathbf{y}}\right)^T\left(\bar{\mathbf{F}}\mathbf{w} - \bar{\mathbf{y}}\right), \tag{19}$$

and its solution is computed by solving the linear system,

$$\left(\alpha'\mathbf{H} + \beta'\mathcal{I} + \bar{\mathbf{F}}^T\bar{\mathbf{F}}\right)\mathbf{w} = \bar{\mathbf{F}}^T\bar{\mathbf{y}}. \tag{20}$$

Similarly, the nonlinear optimization problem in Eq. (18) becomes

$$\min_{\mathbf{w}} \quad \frac{1}{2}\alpha'\mathbf{w}^T\mathbf{H}\mathbf{w} + \frac{1}{2}\beta'\mathbf{w}^T\mathbf{w} + \frac{1}{2}\sum_i(\bar{\mathbf{F}}\mathbf{w} - \bar{\mathbf{y}})_i^p, \tag{21}$$

and the Newton iterations for solving this problem are given by

$$\left(\alpha'\mathbf{H} + \beta'\mathcal{I} + \frac{p}{2}(p-1)\bar{\mathbf{F}}^T D((\bar{\mathbf{F}}\mathbf{w} - \bar{\mathbf{y}})^{p-2})\bar{\mathbf{F}}\right)\Delta\mathbf{w} = -\alpha'\mathbf{H}\mathbf{w} - \beta'\mathbf{w} - \frac{p}{2}\bar{\mathbf{F}}^T(\bar{\mathbf{F}}\mathbf{w} - \bar{\mathbf{y}})^{p-1}. \tag{22}$$

This form is more convenient to use since the energy minimization or the regularization terms can be 'turned off' by setting either $\alpha'$ or $\beta'$, respectively, to zero.

With these modifications, we now present the RMTS training procedure in Alg. 1. We start by pre-computing $\mathbf{F}$ and $\mathbf{H}$ as an initialization step; however, $\mathbf{H}$ is only computed if $\alpha'$ is not zero since this

step forms a significant portion of the computation time in most situations. Next, we solve the quadratic optimization problem from Fig. 2, reformulated as Eq. (20). In some cases, the result is sufficiently accurate, and we can skip the nonlinear optimization solution by setting *num_nonlinear_iterations* to zero. Otherwise, we use the quadratic optimization solution as an initial guess and solve the nonlinear optimization problem from Fig. 3, reformulated as Eq. (22). We loop over output variables and apply a Newton iteration.

**Algorithm 1** RMTS training algorithm

```
compute F as a sparse matrix
if α' ≠ 0 then
  compute H as a sparse matrix
else
  initialize H as a sparse matrix of zeros
end if

# solve the quadratic optimization problem
assemble the matrix of the linear system in Eq. (20)
for each output variable do
  solve Eq. (20)
end for

# solve the nonlinear optimization problem (optional)
if num_nonlinear_iterations > 0 then
  for each output variable do
    while not converged and iteration < num_nonlinear_iterations do
      assemble the matrix of the linear system in Eq. (22)
      solve Eq. (22)
    end while
  end for
end if
```

Gradient information in the training data set can be easily incorporated via $\bar{\mathbf{F}}$ and $\bar{\mathbf{y}}$. Without gradient information, the number of rows of $\bar{\mathbf{F}}$ and the size of the $\bar{\mathbf{y}}$ vector are both equal to the number of training points. When gradient information is available, it is simply appended to $\bar{\mathbf{F}}$ and $\bar{\mathbf{y}}$. Therefore, if we have $n_t$ training points and $n_x$ input variables with gradient information available at all training points with respect to all $n_x$ input variables, the number of rows of $\bar{\mathbf{F}}$ and the size of $\bar{\mathbf{y}}$ would both be $n_t + n_t n_x$ since there $n_t$ training output values in addition to $n_x$ derivatives at each of the $n_t$ training points.

## 4 Results

We now present results investigating the accuracy and efficiency of RMTS, and we compare it to other surrogate modeling approaches. We first present a suite of benchmarking problems and then use this problem suite to compare RMTS to IDW, RBFs, and kriging. Next, we examine the effect of energy minimization and the nonlinear reformulation of RMTS presented in Section 3, and the use of gradient information. Finally, we apply RMTS to a practical MDO application.

### 4.1 Benchmarking problems

To compare RMTS to other approaches, we use a suite of 1-D, 2-D, 3-D, and 4-D benchmarking problems where the true function is known. Four problems are $n$-dimensional (where $n$ can be any number), two problems are 3-D, and another two are 4-D. Therefore, there are four 1-D, four 2-D, six 3-D, and six 4-D problems in the benchmarking suite.

The $n$-dimensional problems include the position tracking of a robot arm An and Owen (2001), the tip deflection of a cantilever beam Cheng et al. (2015), a tensor product of hyperbolic tangent functions, and the multidimensional Rosenbrock function. The 3-D problems include the shear stress of a welded beam Deb (2000) and the flow of water through a borehole Morris et al. (1993). The 4-D problems include the weight

of an aircraft wing Forrester et al. (2008) and the vibration frequency of a torsion beam Wang et al. (2006). The problems from the literature were compiled by Liu et al. (2016). The tensor-product hyperbolic tangent problem is included because it approximates a step function, which is known to cause oscillations with some surrogate modeling approaches. For all the problems, the function, the list of inputs and outputs, and their ranges are given in B. In all cases, we use Latin hypercube sampling (LHS) to generate an unstructured set of training data.

For the benchmarking results, RMTS, IDW, RBFs, and kriging are all used with their default settings in SMT. We do this to enable a fair comparison free of bias or tuning for a particular set of problems. For RMTS, we show separate results for RMTB and RMTC, and both are run with default settings: RMTB is run with 15 control points and an order of 3, and RMTC is run with 4 cubic Hermite spline elements. The RBFs are run with Gaussian basis functions, no polynomial trend functions, and regularization involving the addition of $10^{-10}$ to the diagonal. We use ordinary kriging with Gaussian correlation functions. For kriging, we use the implementation of Bouhlel et al. (2016), with ordinary kriging, which is the default option for kriging in SMT. For the maximum likelihood estimation, the initial guess for all hyperparameters is $10^{-2}$.

### 4.2 Comparison with other surrogate modeling approaches

In Fig. 4, we compare the training time, prediction time, and RMS error for IDW, RBFs, kriging, and four forms of RMTS: RMTB, RMTC, and their equivalents with energy minimization. As a reminder, RMTB and RMTC stand for regularized minimal-energy tensor-product B-splines and cubic Hermite splines, respectively. Energy minimization increases the training time while improving accuracy in problems with sparse datasets; to quantify this tradeoff, we consider RMTB and RMTC with and without energy minimization.

#### 4.2.1 Training time

We first examine the training time plots in Fig. 4 and discuss the trend versus the number of training points. We see that RBFs and kriging are the fastest for smaller numbers of training points. However, they scale poorly as the number of training points increases while *the training time for RMTS scales better than linearly up to at least $\mathcal{O}(10^5)$ training points*. Moreover, kriging fails to converge outright with more than $n_t = 1000$ training points. These observations are expected, since RBF requires the solution of a dense matrix of size $n_t$, and kriging requires the inversion of a dense matrix of size $n_t$. In terms of training time versus the number of inputs, kriging and RBFs show low sensitivity to $n_x$, while *the training time for RMTS scales poorly with $n_x$, especially when using energy minimization*. Furthermore, using energy minimization increases the training time for RMTS by at least an order of magnitude. There is no training time data for IDW since it does not require training.

#### 4.2.2 Prediction time

Examining the prediction time plots in Fig. 4, we can arrive at three conclusions. First, IDW, RBFs, and kriging scale linearly with the number of training points because the prediction requires computing the distance to each training point. Second, the prediction times for all variants of RMTS are independent of the number of training points, since the coefficients of the B-spline or cubic Hermite spline are precomputed. Finally, *RMTS has a lower prediction time than the other methods for problems with more than roughly 100 training points*. However, the RMTS prediction time increases with $n_x$.

#### 4.2.3 Error

We now examine the error plots in Fig. 4. The error is quantified by the evaluation of 5000 points, also selected using LHS in the input space. IDW has a consistent but poor rate of convergence, because the surrogate model is locally flat at all training points—i.e., it has zero partial derivatives at the training point locations. As expected, kriging is the most accurate when it converges, i.e., when $n_t < \mathcal{O}(10^3)$, but it fails in problems with more training points. RMTS does not have this issue, and its accuracy improves with more training points. Therefore, the conclusion is that *when a large number of training points is available, RMTS achieves a higher accuracy than the other methods considered*. However, it plateaus at a certain point when the resolution of the spline is insufficient to fit the function to the desired level of accuracy. We see that energy minimization either improves the accuracy or has no effect, but the improvement is larger with fewer training points, as expected.

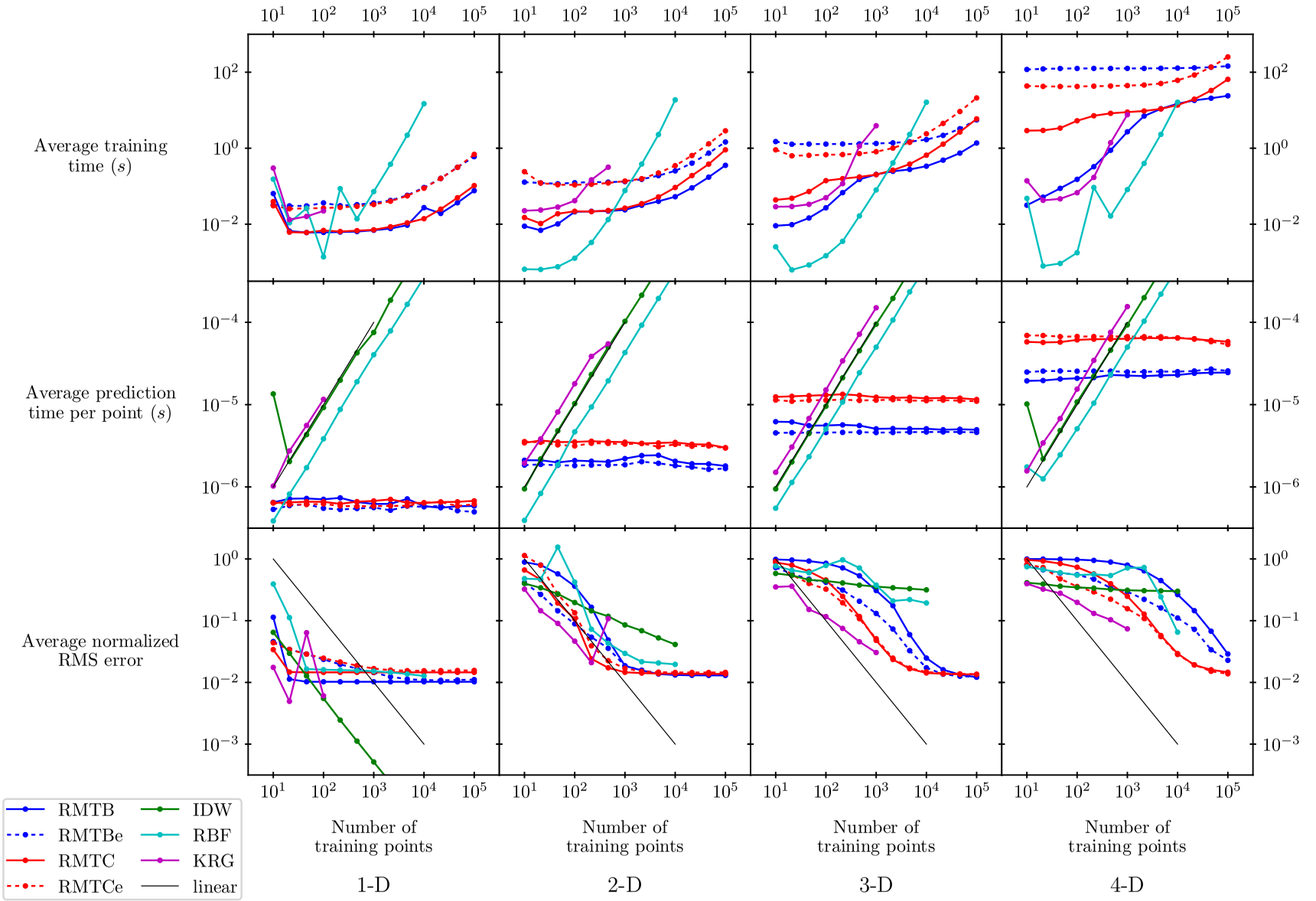


Figure 4: Comparison of the surrogate modeling approaches. The training time, prediction time, and root mean square (RMS) error are all averaged across the four 1-D, four 2-D, six 3-D, and six 4-D test problems. The "e" in RMTBe and RMTCe indicates energy minimization. We can draw three conclusions: (1) the training time for RMTS scales well with the number of training points but poorly with dimensionality; (2) the prediction time for RMTS is the lowest above about 100 training points; and (3) when a large dataset is available, RMTS achieves the lowest error.

### 4.3 Effect of energy minimization and the nonlinear reformulation

We now investigate the effect of energy minimization and the nonlinear reformulation in more depth via the cantilever beam and robot arm problems. In Fig. 5, we plot the RMTS interpolation functions with no energy minimization, with energy minimization and no nonlinear iterations, and with energy minimization and 20 nonlinear iterations. *We see that energy minimization and the nonlinear reformulation are both beneficial despite the increase in training time for small to moderate datasets.*

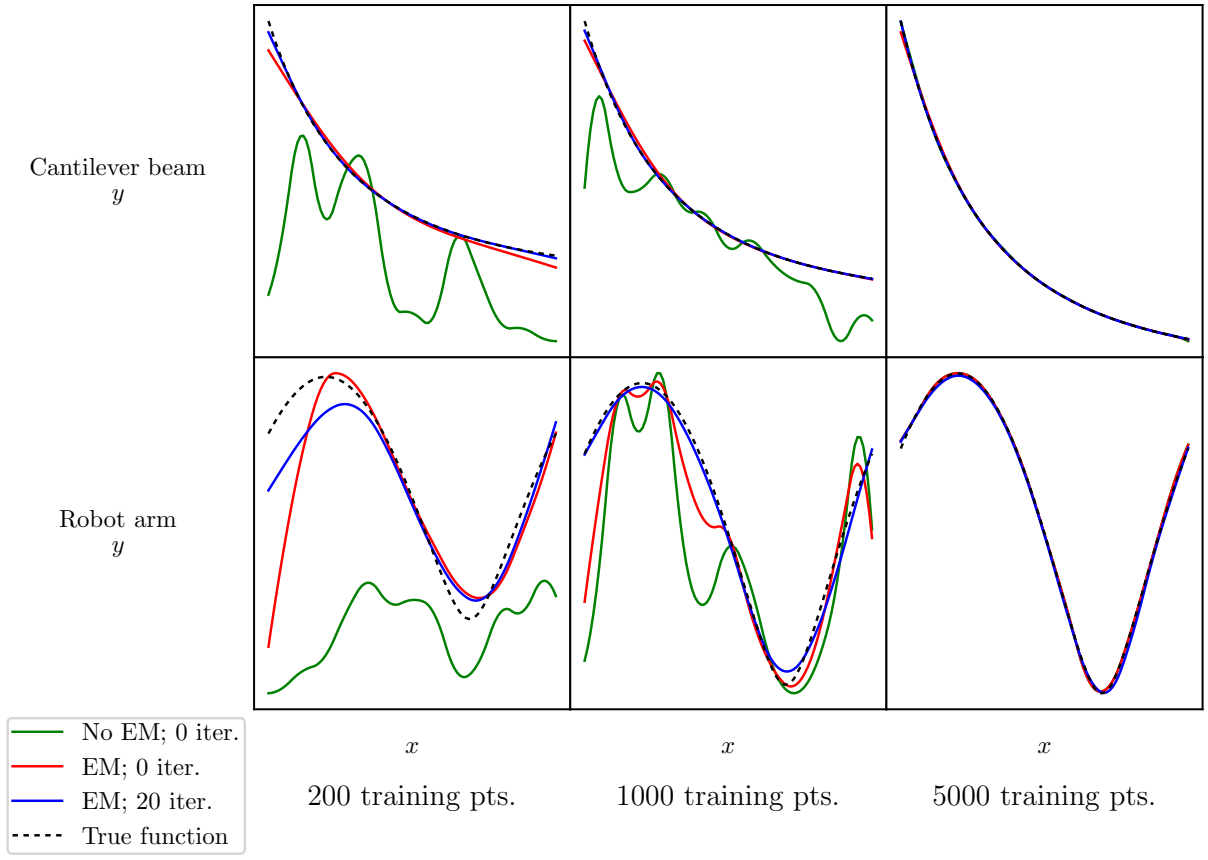


Figure 5: The true function compared to the RMTS interpolant with and without energy minimization and nonlinear iterations. The test problems are the 3-D cantilever beam and 3-D robot arm problems. The conclusion is that energy minimization and nonlinear iterations are both necessary and beneficial for smaller datasets with fewer than $\mathcal{O}(10^3)$ points.

In Fig. 6, we show the error versus the number of nonlinear iterations. For the cantilever beam, the error decreases continuously with the number of iterations, while for the robot arm, only the first iteration provides any noticeable benefit. Therefore, the number of iterations required is problem-dependent.

### 4.4 Gradient-enhanced RMTS

Gradient-enhanced surrogate models use the gradients of the function with respect to the inputs in addition to the function values in the surrogate model construction Forrester et al. (2008). RMTS readily generalizes to gradient-enhanced interpolation via additional approximation terms. In the approximation error equation $\bar{\mathbf{F}}\bar{w} = \bar{\mathbf{y}}$ we simply add rows to $\bar{\mathbf{F}}$ and $\bar{\mathbf{y}}$ for the derivative data.

In Fig. 7, we show how each of RMTB, RMTBe, RMTC, and RMTCe compare to the corresponding versions with gradient information available at each training point location. We can draw several conclusions from these results. First, *energy minimization increases the training time by almost an order of magnitude,*

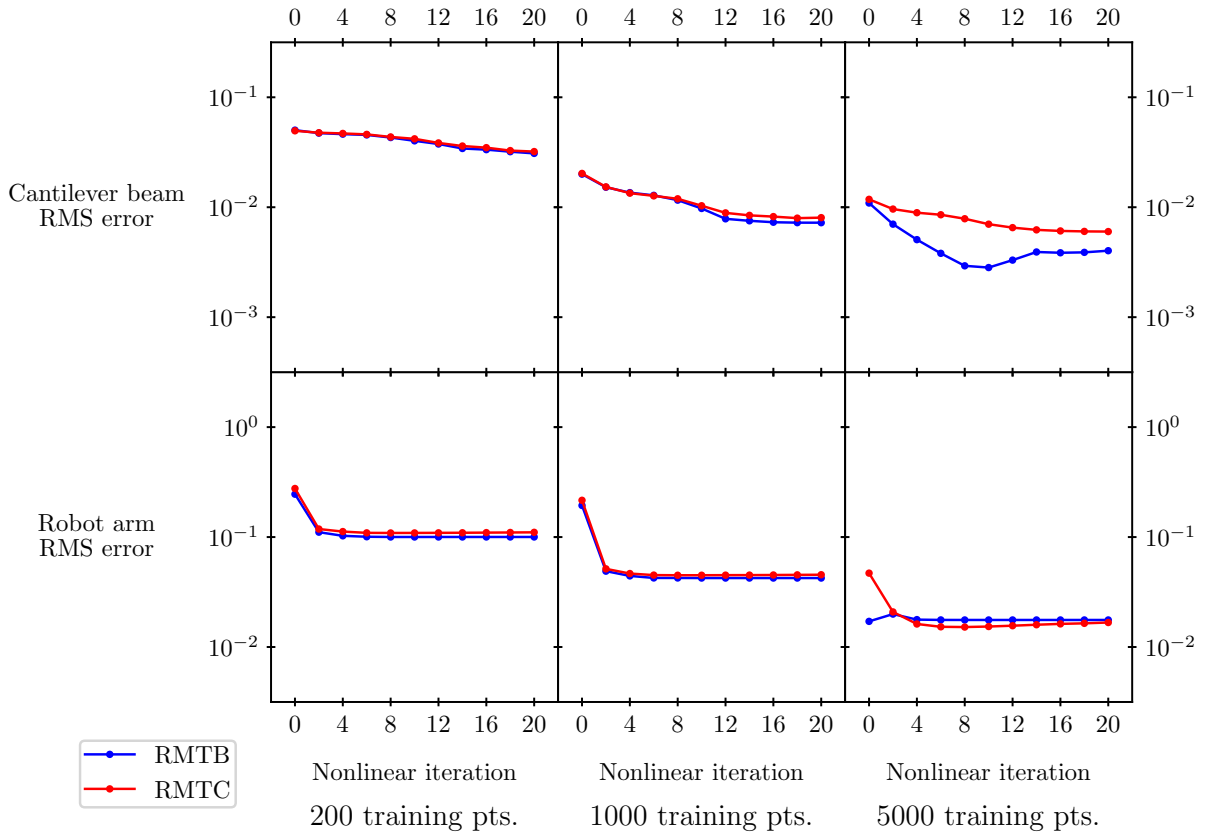


Figure 6: Error versus number of nonlinear iterations for the 3-D cantilever beam and 3-D robot arm problems. The number of iterations after which the improvement plateaus is problem-dependent, but as a general recommendation 10–20 iterations are worthwhile.

*but for small to moderate datasets, it may be justified if improved accuracy is desired.* Second, the use of gradient information causes a smaller increase in training time, and it improves accuracy for moderate-sized datasets. However, gradient information does not consistently improve accuracy. This is to be expected because with larger datasets, we may sacrifice accurate interpolation of the training values to more closely interpolate the derivatives. Third, *RMTB has a lower prediction time but a higher error than RMTC with the default settings.* The default settings are fourth-order B-splines with 15 control points for RMTB and 4 cubic elements for RMTC.

The interpretation of this result is that when the spline does not have sufficient resolution, including gradient information hurts accuracy in the output value because some of the limited degrees of freedom are used to fit the derivatives, taking away from the ability to fit the function itself. Therefore, these results suggest that with RMTS, it may not always be worthwhile to compute gradient information, especially in situations with a large number of training points and a small number of B-spline control points or cubic spline elements. However, these results were generated with the default settings of RMTB and RMTC. Gradient information is likely to improve accuracy in problems with higher-resolution splines.

### 4.5 RMTS application in a practical MDO context

We now use RMTS in a practical MDO problem, aircraft mission analysis and optimization. This problem seeks to design the full altitude and Mach number profile for a commercial transport aircraft by modeling the aircraft equations of motion over the full flight profile Kao et al. (2015). This problem is motivated by recent

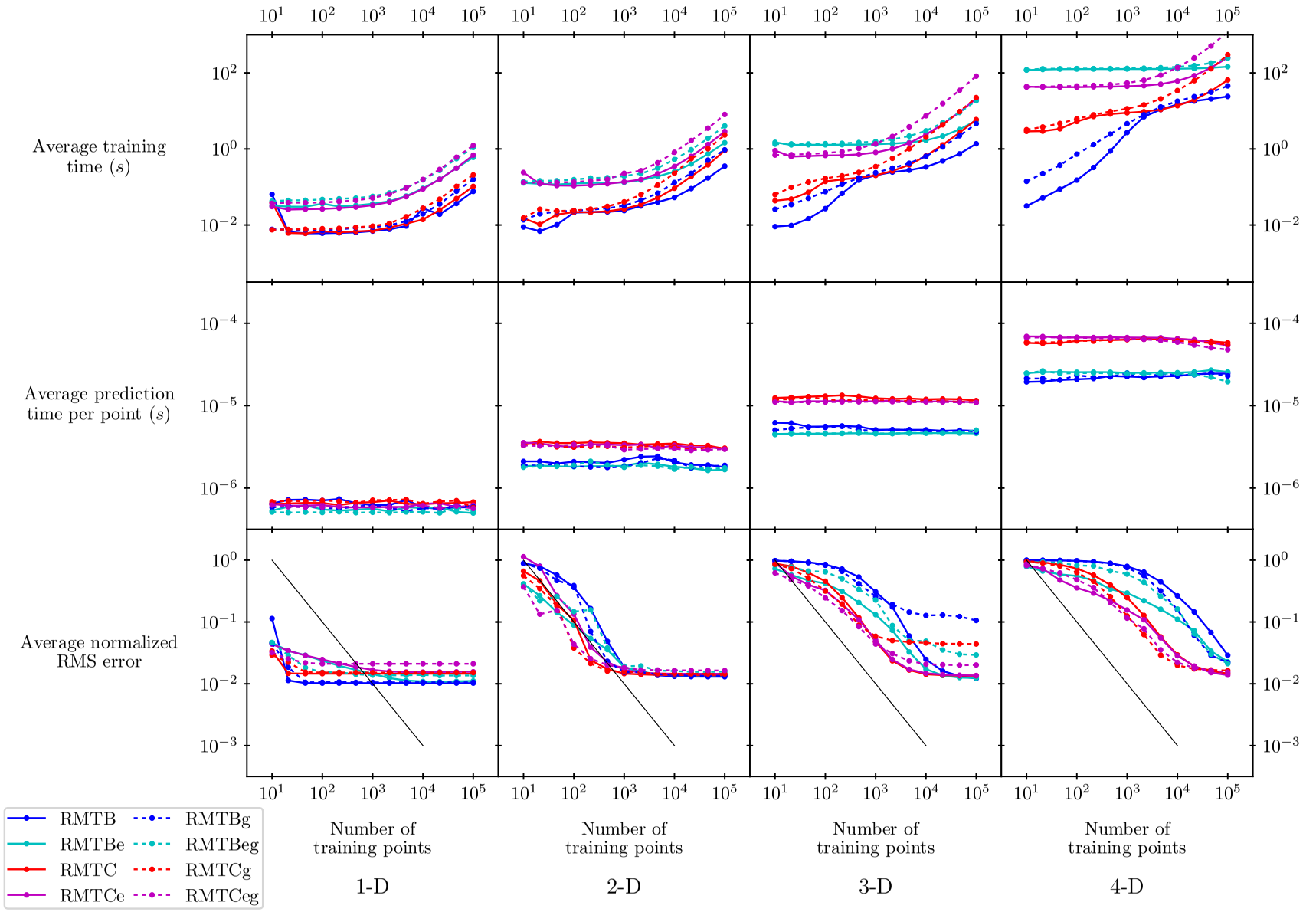


Figure 7: Comparison of gradient-enhanced and regular RMTS. The training time, prediction time, and root mean square (RMS) error are all averaged across the four 1-D, four 2-D, six 3-D, and six 4-D test problems. The "e" indicates energy minimization, and "g" indicates gradient information. Energy minimization increases the training time by an order of magnitude but improves accuracy. RMTB has a faster prediction but lower accuracy than RMTC with the default settings.

research into unconventional aircraft configurations, morphing wing technologies Burdette et al. (2016), and new optimal descent profiles for reduced fuel burn. The problem involves several coupled disciplines because the weight of the aircraft affects its lift, which sets up a cascade of dependencies on drag, required thrust, fuel burn rate, and total mission fuel weight, looping back to total aircraft weight. This loop involves aerodynamics and propulsion models, along with the equations of motion.

The aerodynamics and propulsion models must be evaluated once for each: 1) mission discretization point, 2) Newton iteration, 3) mission (if multiple missions are simultaneously considered), and 4) optimization iteration. When we multiply the resulting nested loops, tens of millions of evaluations of the aerodynamic and propulsion models Hwang and Martins (2016) are required, so direct evaluations of high-fidelity models are not feasible. Moreover, the most important consideration in this problem is to minimize the prediction time.

Figures 8 and 9 show 1-D slices of the 2-D aerodynamics and 3-D propulsion models where the training points are generated offline using high-fidelity models. The training points are arranged in a structured grid, which allows us to plot 1-D slices of the RMTS surrogate model along with training points to evaluate the quality of the approximation. The aerodynamics problem has 506 training points, and the propulsion problem has 1056 training points. The high nonlinearity and multiple inflection points present in both datasets make them challenging to fit accurately and without oscillations. We fit the aerodynamic model using 50 B-spline control points with RMTB, and we fit the propulsion model with RMTC with 6 elements by turning off energy minimization. Some tuning was required to obtain the fits, but the main advantage of using RMTS is that the prediction time is an order of magnitude lower than that for a kriging surrogate model, as we can infer from Fig. 4. Moreover, it is helpful that the parameters of RMTS (e.g., number of control points, nonlinear iterations, weight of the energy minimization term) are easily understood by a user, facilitating manual tuning of the model.

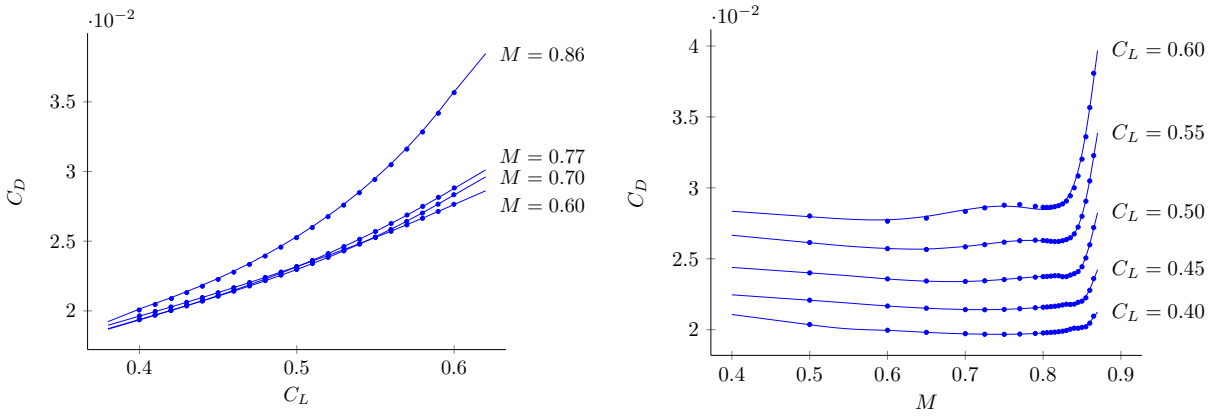


Figure 8: 2-D surrogate model for aircraft aerodynamics with drag coefficient computed as a function of lift coefficient and Mach number.

## 5 Conclusion

In this paper, we developed regularized minimal-energy tensor-product splines (RMTS), a surrogate modeling technique for problems with up to four dimensions. Kriging is widely used for low-dimensional surrogate modeling applications because it is generally fast and accurate; however, it can fail to train and is slower to predict in problems with thousands of training points. In contrast, tensor-product splines are more robust, and the prediction time is independent of the number of training points. RMTS uses energy minimization to handle under-constrained problems in which there are more spline coefficients than training points, and it makes use of a nonlinear generalization of the constrained energy minimization problem to improve accuracy.

We compared the performance of RMTS to inverse distance weighting (IDW), radial basis functions (RBFs), and kriging in terms of accuracy and efficiency. We benchmarked each method using a suite of 1-D, 2-D, 3-D, and 4-D functions, representing a mixture of practical engineering models and simple academic functions. Using these functions, we generated unstructured sets of training data using LHS.

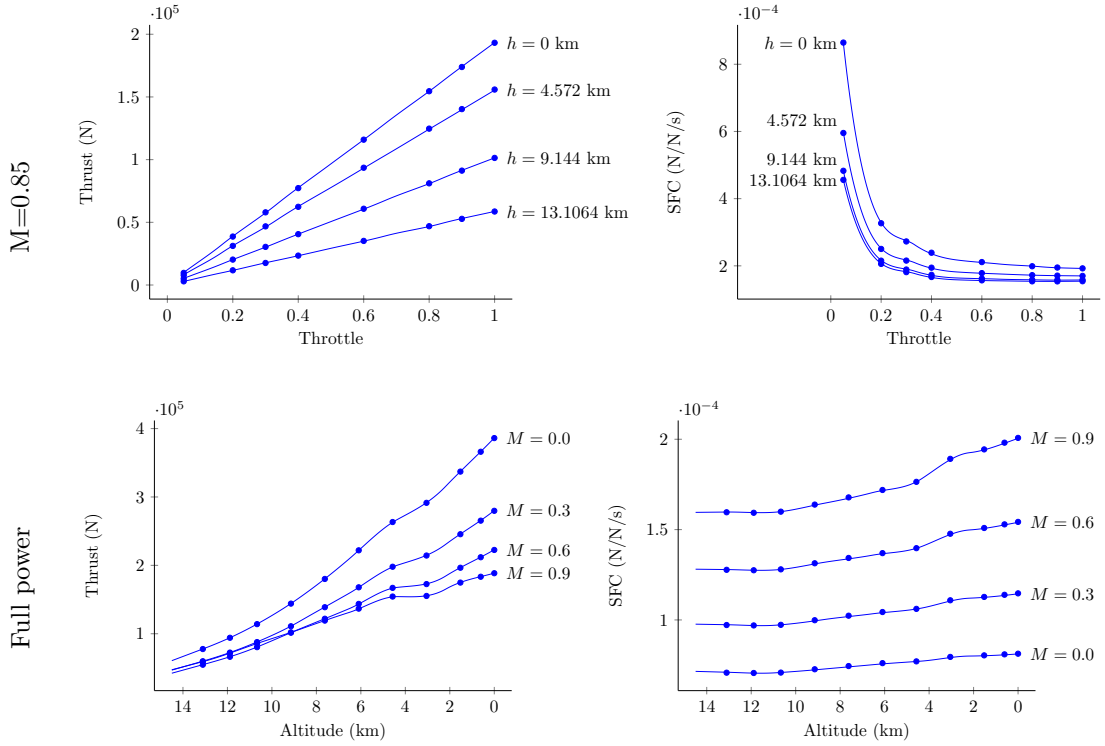


Figure 9: 1-D slices versus throttle (top) and altitude (bottom) for the 3-D aircraft engine model.

The results of the benchmarking can be summarized as follows. In terms of training time, RMTS is the slowest for most reasonably sized training datasets. This is because the training time scales poorly with the number of dimensions, although it scales better than linearly with the number of training points. In terms of prediction time, RMTS is the best choice above a certain number of training points ($\mathcal{O}(10)$ for 1-D and $\mathcal{O}(100)$ for 4-D). In applications where the surrogate model is evaluated thousands of times or more in discretization or solver loops, the benefit of the fast prediction can offset a high training time. In terms of accuracy, kriging is generally better than RMTS, but RMTS is more consistent across problems without the risk of convergence failure that kriging is prone to.

Thus, RMTS is an attractive surrogate modeling technique for low-dimensional problems with many training points, or when prediction time is the most important criterion. In other situations, kriging has a lower training time, and it is generally more accurate. When hundreds of training points are available, kriging has a higher prediction time that increases with the number of training points. Moreover, it can also fail to converge. There is also a risk of convergence failure for training datasets in which some of the points are too close together in the input space.

RMTS is expected to be useful primarily for engineering problems where the surrogate model replaces a computational model that incurs significant run time. For instance, if a moderately expensive computational model is called within a loop during an optimization, it would be beneficial to precompute a large set of training data once and replace the model with RMTS for its fast prediction time. The upfront cost would be especially worthwhile in MDO problems, since the surrogate model would be run many times while converging coupled multidisciplinary analyses and debugging other disciplines.

Even for computational models with significant run times and thus small training datasets, RMTS can be useful when kriging has convergence failures or produces an interpolant with spurious oscillations. In the benchmarking tests, RMTS had an error two or three times higher than that of kriging, but it never failed to converge. Moreover, the RMTS parameters are easy to understand: in particular, the number of B-spline control points or the number of cubic elements, the relative weight on the energy minimization terms, and the number of nonlinear iterations. Therefore, when the application allows for manually tuning the surrogate

model, the intuitive and physically meaningful parameters in RMTS can be an advantage that enables a more accurate interpolant to be achieved.

Our implementation of RMTS and the other surrogate models and problems benchmarked in this paper are available in the open-source SMT Python package[2].

## 6 Conflict of interest statement

The authors declare that there is no conflict of interest in relation to this article.

## 7 Acknowledgments

This work was partially supported by NASA through award number NNX11AI19A and contract number NNC14BA04B.

## 8 Bibliography

### References

J. T. Hwang, D. Y. Lee, J. W. Cutler, J. R. R. A. Martins, Large-scale multidisciplinary optimization of a small satellite's design and operation, Journal of Spacecraft and Rockets 51 (2014) 1648–1663.

T. W. Simpson, J. Poplinski, P. N. Koch, J. K. Allen, Metamodels for computer-based engineering design: Survey and recommendations, Engineering with Computers 17 (2001a) 129–150.

T. W. Simpson, T. M. Mauery, J. J. Korte, F. Mistree, Kriging models for global approximation in simulation-based multidisciplinary design optimization, AIAA Journal 39 (2001b) 2233–2241.

J. P. Kleijnen, Kriging metamodeling in simulation: A review, European Journal of Operational Research 192 (2009) 707–716.

J. T. Hwang, J. R. R. A. Martins, Allocation-mission-design optimization of next-generation aircraft using a parallel computational framework, in: 57th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, American Institute of Aeronautics and Astronautics, 2016. doi:10.2514/6.2016-1662.

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

T. Simpson, V. Toropov, V. Balabanov, F. Viana, Design and analysis of computer experiments in multidisciplinary design optimization: A review of how far we have come-or not, in: 12th AIAA/ISSMO multidisciplinary analysis and optimization conference, 2008, p. 5802.

A. Forrester, A. Sobester, A. Keane, Engineering design via surrogate modelling: A practical guide, John Wiley & Sons, 2008.

G. E. P. Box, K. B. Wilson, On the experimental attainment of optimum conditions, Journal of the Royal Statistical Society. Series B (Methodological) 13 (1951) 1–45.

J. H. Friedman, Multivariate adaptive regression splines, The Annals of Statistics 19 (1991) 1–67.

C. Cortes, V. Vapnik, Support-vector networks, Machine Learning 20 (1995) 273–297.

D. Shepard, A two-dimensional interpolation function for irregularly-spaced data, in: Proceedings of the 1968 23rd ACM National Conference, ACM '68, ACM, New York, NY, USA, 1968, pp. 517–524. doi:10.1145/800186.810616.

R. J. Renka, Multivariate interpolation of large sets of scattered data, ACM Trans. Math. Softw. 14 (1988) 139–148.

G. Y. Lu, D. W. Wong, An adaptive inverse-distance weighting spatial interpolation technique, Computers & Geosciences 34 (2008) 1044–1055.

W. J. Gordon, J. A. Wixom, Shepard's method of "metric interpolation" to bivariate and multivariate interpolation, Mathematics of Computation 32 (1978) 253–264.

J. Duchon, Splines minimizing rotation-invariant semi-norms in Sobolev spaces, in: W. Schempp, K. Zeller (Eds.), Constructive Theory of Functions of Several Variables, volume 571 of *Lecture Notes in Mathematics*, Springer Berlin Heidelberg, 1977, pp. 85–100. doi:10.1007/BFb0086566.

R. L. Hardy, Multiquadric equations of topography and other irregular surfaces, Journal of Geophysical Research 76 (1971) 1905–1915.

R. E. Carlson, T. A. Foley, The parameter $R^2$ in multiquadric interpolation, Computers & Mathematics with Applications 21 (1991) 29–42.

E. Kansa, R. Carlson, Improved accuracy of multiquadric interpolation using variable shape parameters, Computers & Mathematics with Applications 24 (1992) 99–120.

H. Mitášová, L. Mitáš, Interpolation by regularized spline with tension: I. Theory and implementation, Mathematical Geology 25 (1993) 641–655.

H. Q. Dinh, G. Turk, G. Slabaugh, Reconstructing surfaces using anisotropic basis functions, in: Proceedings of the Eighth IEEE International Conference on Computer Vision, volume 2, 2001, pp. 606–613. doi:10.1109/ICCV.2001.937682.

G. Casciola, L. Montefusco, S. Morigi, The regularizing properties of anisotropic radial basis functions, Applied Mathematics and Computation 190 (2007) 1050–1062.

G. Casciola, L. Montefusco, S. Morigi, Edge-driven image interpolation using adaptive anisotropic radial basis functions, Journal of Mathematical Imaging and Vision 36 (2010) 125–139.

R. Beatson, O. Davydov, J. Levesley, Error bounds for anisotropic RBF interpolation, Journal of Approximation Theory 162 (2010) 512–527.

J. Yu, G. Turk, Reconstructing surfaces of particle-based fluids using anisotropic kernels, ACM Trans. Graph. 32 (2013) 1–12.

H. Wendland, Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree, Advances in Computational Mathematics 4 (1995) 389–396.

Z. Wu, Compactly supported positive definite radial functions, Advances in Computational Mathematics 4 (1995) 283–292.

M. D. Buhmann, A new class of radial basis functions with compact support, Mathematics of Computation (2000) 307–318.

D. G. Krige, A statistical approach to some basic mine valuation problems on the Witwatersrand, Journal of the Chemical, Metallurgical and Mining Society of South Africa 52 (1951) 119–139.

G. Matheron, Principles of geostatistics, Economic Geology 58 (1963) 1246–1266.

N. Cressie, The origins of kriging, Mathematical Geology 22 (1990) 239–252.

J. Sacks, W. J. Welch, T. J. Mitchell, H. P. Wynn, Design and analysis of computer experiments, Statist. Sci. 4 (1989) 409–423.

D. Zimmerman, C. Pavlik, A. Ruggles, M. P. Armstrong, An experimental comparison of ordinary and universal kriging and inverse distance weighting, Mathematical Geology 31 (1999) 375–390.

D. J. J. Toal, N. W. Bressloff, A. J. Keane, Kriging hyperparameter tuning strategies, AIAA Journal 46 (2008) 1240–1252.

J. Martin, T. Simpson, Use of kriging models to approximate deterministic computer models, AIAA Journal 43 (2005) 853–863.

T. J. Mitchell, M. D. Morris, Bayesian design and analysis of computer experiments: Two examples, Statistica Sinica 2 (1992) 359–379.

K. Kerry, K. Hawick, Kriging interpolation on high-performance computers, in: P. Sloot, M. Bubak, B. Hertzberger (Eds.), High-Performance Computing and Networking, volume 1401 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, 1998, pp. 429–438. doi:10.1007/BFb0037170.

N. Cressie, G. Johannesson, Fixed rank kriging for very large spatial data sets, Journal of the Royal Statistical Society: Series B (Statistical Methodology) 70 (2008) 209–226.

W. E. Leithead, Y. Zhang, $O(N^2)$-operation approximation of covariance matrix inverse in Gaussian process regression based on quasi-Newton BFGS method, Communications in Statistics - Simulation and Computation 36 (2007) 367–380.

D. G. Schweikert, An interpolation curve using a spline in tension, Journal of Mathematics and Physics 45 (1966) 312–317.

E. H. Lee, G. E. Forsythe, Variational study of nonlinear spline curves, SIAM Review 15 (1973) pp. 120–133.

G. Brunnett, J. Kiefer, Interpolation with minimal-energy splines, Computer-Aided Design 26 (1994) 137–144.

C. Zhang, P. Zhang, F. F. Cheng, Fairing spline curves and surfaces by minimizing energy, Computer-Aided Design 33 (2001) 913–923.

H. P. Moreton, C. H. Séquin, Functional optimization for fair surface design, SIGGRAPH Comput. Graph. 26 (1992) 167–176.

W. Welch, A. Witkin, Variational surface modeling, SIGGRAPH Comput. Graph. 26 (1992) 157–166.

L. P. Kobbelt, Discrete fairing and variational subdivision for freeform surface design, The Visual Computer 16 (2000) 142–158.

M. Halstead, M. Kass, T. DeRose, Efficient, fair interpolation using Catmull-Clark surfaces, in: Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '93, ACM, New York, NY, USA, 1993, pp. 35–44. doi:10.1145/166117.166121.

G.-L. Mo, Y.-N. Zhao, A new extension algorithm for cubic B-splines based on minimal strain energy, Journal of Zhejiang University SCIENCE A 7 (2006) 2043–2049.

G. Wolberg, I. Alfy, An energy-minimization framework for monotonic cubic spline interpolation, Journal of Computational and Applied Mathematics 143 (2002) 145–188.

T. I. Vassilev, Fair interpolation and approximation of B-splines by energy minimization and points insertion, Computer-Aided Design 28 (1996) 753–760.

J. An, A. Owen, Quasi-regression, Journal of Complexity 17 (2001) 588–607.

G. H. Cheng, A. Younis, K. H. Hajikolaei, G. G. Wang, Trust region based mode pursuing sampling method for global optimization of high dimensional design problems, Journal of Mechanical Design 137 (2015) 021407.

K. Deb, An efficient constraint handling method for genetic algorithms, Computer Methods in Applied Mechanics and Engineering 186 (2000) 311–338.

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

A. Forrester, A. Sobester, A. Keane, Engineering design via surrogate modelling: A practical guide, John Wiley & Sons, 2008.

L. Wang, D. Beeson, G. Wiggs, M. Rayasam, Structures, Structural Dynamics, and Materials and Co-located Conferences, American Institute of Aeronautics and Astronautics, 2006. URL: http://dx.doi.org/10.2514/6.2006-1811. doi:10.2514/6.2006-1811, 0.

H. Liu, S. Xu, X. Wang, Sampling strategies and metamodeling techniques for engineering design: Comparison and application, in: ASME Turbo Expo 2016: Turbomachinery Technical Conference and Exposition, American Society of Mechanical Engineers, 2016, pp. V02CT45A019–V02CT45A019.

M. A. Bouhlel, N. Bartoli, A. Otsmane, J. Morlier, Improving kriging surrogates of high-dimensional design models by partial least squares dimension reduction, Structural and Multidisciplinary Optimization 53 (2016) 935–952.

J. Y. Kao, J. T. Hwang, J. R. R. A. Martins, A modular approach for mission analysis and optimization, in: 56th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, 2015.

D. A. Burdette, G. K. Kenway, J. R. R. A. Martins, Performance evaluation of a morphing trailing edge using multipoint aerostructural design optimization, in: 57th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, American Institute of Aeronautics and Astronautics, 2016. doi:10.2514/6.2016-0159.

## A Derivation of the kriging equation

Here, we show how we derive $\mathbf{y} - \mathbf{P}\mathbf{a} = \boldsymbol{\Psi}\bar{\boldsymbol{\Psi}}^{-1}(\bar{\mathbf{y}} - \bar{\mathbf{P}}\mathbf{a})$ in the development of the kriging equations.

We first define $\bar{\mathbf{z}} = \bar{\mathbf{y}} - \bar{\mathbf{P}}\mathbf{a}$ and $\mathbf{z} = \mathbf{y} - \mathbf{P}\mathbf{a}$, where both $\bar{\mathbf{z}}$ and $\mathbf{z}$ are random vectors. Moreover, we assume $\mathbf{z} = \mathbf{C}\bar{\mathbf{z}}$, signifying that the detrended output at each prediction point is a random variable defined as a linear combination of detrended training point outputs, which are also random variables. We also define $\hat{\mathbf{z}}$ as the random vector of the true detrended outputs at the prediction points, from the true underlying model.

We assume that the polynomial trend model accurately captures the unknown function in an average sense, neglecting random variations. Then, any linear combination of the random variables at the training points is also expected to be accurate and thus have zero mean. We use the coefficients of the linear combination to minimize the variance of the error, which is $\epsilon = \mathbf{C}\bar{\mathbf{z}} - \hat{\mathbf{z}}$, since a good interpolant would not deviate significantly from zero error. Therefore, for the $i$th prediction point, the vector of coefficients is computed by solving

$$\min_{\mathbf{C}_{i*}} \mathbb{E}(\epsilon_i^2). \tag{23}$$

The variance of the error can be written as

$$\mathbb{E}(\epsilon_i^2) = \mathbf{C}_{i*}\mathbb{E}(\bar{\mathbf{z}}\bar{\mathbf{z}}^T)\mathbf{C}_{i*}^T - 2\mathbf{C}_{i*}\mathbb{E}(\hat{z}_i\bar{\mathbf{z}}) + \mathbb{E}(\hat{z}_i\hat{z}_i). \tag{24}$$

Setting the gradient with respect to $\mathbf{C}_{i*}$ to zero reveals that the minimum of this objective function is given by

$$\mathbb{E}(\bar{\mathbf{z}}\bar{\mathbf{z}}^T)\mathbf{C}_{i*}^T = \mathbb{E}(\hat{z}_i\bar{\mathbf{z}}), \tag{25}$$

which can be simplified to

$$\bar{\boldsymbol{\Psi}}\bar{\mathbf{C}}^T = \boldsymbol{\Psi}^T. \tag{26}$$

Since $\bar{\boldsymbol{\Psi}}$ is symmetric, this leads to $\mathbf{C} = \boldsymbol{\Psi}\bar{\boldsymbol{\Psi}}^{-1}$, and inserting this into $\mathbf{z} = \mathbf{C}\bar{\mathbf{z}}$ yields

$$\mathbf{y} - \mathbf{P}\mathbf{a} = \boldsymbol{\Psi}\bar{\boldsymbol{\Psi}}^{-1}(\bar{\mathbf{y}} - \bar{\mathbf{P}}\mathbf{a}), \tag{27}$$

which is what we set out to prove. Therefore, the form of the training equations used in kriging (10) is identical to that of RBFs, but with a different derivation originating from the interpretation of the training and prediction outputs as random variables.

## B Benchmarking problems

### $n$-D robot arm problem

$$r = \sqrt{\left(\sum_{i=1}^{n+1} L_i \cos\left(\sum_{j=1}^{i} \theta_j\right)\right)^2 + \left(\sum_{i=1}^{n+1} L_i \sin\left(\sum_{j=1}^{i} \theta_j\right)\right)^2} \tag{28}$$

| | Quantity | | Description |
|---|---|---|---|
| $y_1$ | $r$ | | Distance of tip of robot arm |
| | $0 \leq \theta_1$ | $\leq 2\pi$ | Angle of the first arm segment |
| $x_1$ | $0 \leq \theta_2$ | $\leq 2\pi$ | Angle of the second arm segment |
| $\vdots$ | $\vdots$ | | |
| $x_n$ | $0 \leq \theta_{n+1}$ | $\leq 2\pi$ | Angle of the last arm segment |
| | $0 \leq L_1$ | $\leq 1$ | Length of the first arm segment |
| | $\vdots$ | | |
| | $0 \leq L_{n+1}$ | $\leq 1$ | Length of the last arm segment |

### $n$-D cantilever beam problem

$$w = \frac{P}{3E} \sum_{i=1}^{n} \left[ \frac{12}{b_i h_i^3} \left( \left( \sum_{j=i}^{n} l_j \right)^3 - \left( \sum_{j=i+1}^{n} l_j \right)^3 \right) \right] \tag{29}$$

| | Quantity | | Description |
|---|---|---|---|
| $y_1$ | $w$ | | Tip deflection |
| $x_1$ | $0.5 \leq l_1$ | $\leq 1.0$ | Length of the first element |
| $\vdots$ | $\vdots$ | | |
| $x_n$ | $0.5 \leq l_n$ | $\leq 1.0$ | Length of the last element |
| | $0.01 \leq b_1$ | $\leq 0.05$ | Width of the first element |
| | $\vdots$ | | |
| | $0.01 \leq b_n$ | $\leq 0.05$ | Width of the last element |
| | $0.30 \leq h_1$ | $\leq 0.65$ | Height of the first element |
| | $\vdots$ | | |
| | $0.30 \leq h_n$ | $\leq 0.65$ | Height of the last element |
| | $P = 50$ kN | | Applied force at the tip |
| | $E = 200$ GPa | | Young's modulus |

### $n$-D tensor-product hyperbolic tangent function

$$f = \prod_{i=1}^{n} \tanh(a x_i) \tag{30}$$

| | Quantity | | Description |
|---|---|---|---|
| $y_1$ | $f$ | | |
| $x_1$ | $-1 \leq x_1$ | $\leq 1$ | First input variable |
| $\vdots$ | $\vdots$ | | |
| $x_n$ | $-1 \leq x_n$ | $\leq 1$ | Last input variable |
| | $a = 10$ | | Abruptness of the step function |

### $n$-D Rosenbrock function

$$f = \sum_{i=1}^{n-1} \left[ 100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2 \right] \tag{31}$$

| | Quantity | Description |
|---|---|---|
| $y_1$ | $f$ | |
| $x_1$ | $-2 \leq x_1 \leq 2$ | First input variable |
| $\vdots$ | $\vdots$ | |
| $x_n$ | $-2 \leq x_n \leq 2$ | Last input variable |

**3-D welded beam problem**

$$\tau = \sqrt{\tau'^2 + \tau''^2 + \frac{l\tau'\tau''}{\sqrt{0.25\,(l^2 + (h+t)^2)}}} \tag{32}$$

$$\text{where} \quad \tau' = \frac{6000}{\sqrt{2}hl}\,,\ \tau'' = \frac{6000(14 + 0.5l)\sqrt{0.25\,(l^2 + (h+t)^2)}}{2\,[0.707hl\,(l^2/12 + 0.25(h+t)^2)]}$$

| | Quantity | Description |
|---|---|---|
| $y_1$ | $\tau$ | Shear stress |
| $x_1$ | $5 \leq t \leq 10$ | Beam thickness |
| $x_2$ | $0.125 \leq h \leq 1$ | Beam height |
| $x_3$ | $5 \leq l \leq 10$ | Beam length |

**3-D water flow problem**

$$f = \frac{2\pi T_u(H_u - H_l)}{\ln(r/r_w)\left[1 + \frac{2LT_u}{\ln(r/r_w)r_w^2 K_w} + \frac{T_u}{T_l}\right]} \tag{33}$$

| | Quantity | Description |
|---|---|---|
| $y_1$ | $f$ | Water flow |
| $x_1$ | $0.05 \leq r_w \leq 0.15$ | Radius of borehole (m) |
| $x_2$ | $100 \leq r \leq 50000$ | Radius of influence (m) |
| $x_3$ | $1120 \leq L \leq 1680$ | Length of borehole (m) |
| | $63070 \leq T_u \leq 115600$ | Transmissivity of upper aquifer ($m^2$/y) |
| | $990 \leq H_u \leq 1110$ | Potentiometric head of upper aquifer (m) |
| | $63.1 \leq T_l \leq 116$ | Transmissivity of lower aquifer ($m^2$/y) |
| | $700 \leq H_l \leq 820$ | Potentiometric head of lower aquifer (m) |
| | $9855 \leq K_w \leq 12045$ | Hydraulic conductivity of borehole (m/y) |

**4-D wing weight problem**

$$W_w = 0.036 S_w^{0.758} W_{fw}^{0.0035} \left(\frac{A}{\cos^2 \Lambda}\right) q^{0.006} \lambda^{0.04} \left(\frac{100tc}{\cos\lambda}\right)^{-0.3} (N_z W_{dg})^{0.49} + S_w W_p \tag{34}$$

| | Quantity | Description |
|---|---|---|
| $y_1$ | $W_w$ | Wing weight |
| $x_1$ | $150 \leq S_w \leq 200$ | Wing area ($ft^2$) |
| $x_2$ | $6 \leq A \leq 10$ | Aspect ratio |
| $x_3$ | $-10 \leq \Lambda \leq 10$ | Quarter-chord sweep (deg) |
| $x_4$ | $0.5 \leq \lambda \leq 1$ | Taper ratio |
| | $220 \leq W_{fw} \leq 300$ | Weight of fuel in the ring (lb) |
| | $16 \leq q \leq 45$ | Dynamic pressure at cruise (lb/$ft^2$) |
| | $0.08 \leq tc \leq 0.18$ | Aerofoil thickness to chord ratio |
| | $2.5 \leq N_z \leq 6$ | Ultimate load factor |
| | $1700 \leq W_{dg} \leq 2500$ | Flight design gross weight (lb) |
| | $0.025 \leq W_p \leq 0.08$ | Paint weight (lb/$ft^2$) |

**4-D torsion vibration problem**

$$f = \frac{\sqrt{\frac{-b-\sqrt{b^2-4ac}}{2a}}}{2\pi} \text{ where } a = 1\,,\; b = -\left(\frac{K_1+K_2}{J_1} + \frac{K_2+K_3}{J_2}\right)\,,\; c = \frac{K_1K_2+K_2K_3+K_3K_1}{J_1J_2} \tag{35}$$

$$\text{and } K_i = \frac{\pi G_i d_i}{32L_i}\;\forall i \in \{1,2,3\}\,,\; M_i = \frac{\rho_i \pi t_i D_i}{4g}\;\forall i \in \{1,2\}\,,\; J_i = 0.5M_i\left(\frac{D_i}{2}\right)^2\;\forall i \in \{1,2\} \tag{36}$$

| | Quantity | | Description |
|---|---|---|---|
| $y_1$ | | $f$ | Lowest natural frequency |
| $x_1$ | 2.7 | $\leq t_1 \leq$ 3.3 | Thickness of disk 1 |
| $x_2$ | 3.6 | $\leq t_2 \leq$ 4.4 | Thickness of disk 2 |
| $x_3$ | 10.8 | $\leq D_1 \leq$ 13.2 | Diameter of disk 1 |
| $x_4$ | 12.6 | $\leq D_2 \leq$ 15.4 | Diameter of disk 2 |
| | 0.252 | $\leq \rho_1 \leq$ 0.308 | Density of disk 1 |
| | 0.09 | $\leq \rho_2 \leq$ 0.11 | Density of disk 2 |
| | 1.8 | $\leq d_1 \leq$ 2.2 | Diameter of shaft 1 |
| | 1.6425 | $\leq d_2 \leq$ 2.0075 | Diameter of shaft 2 |
| | 2.025 | $\leq d_3 \leq$ 2.475 | Diameter of shaft 3 |
| | 9 | $\leq L_1 \leq$ 11 | Length of shaft 1 |
| | 10.8 | $\leq L_2 \leq$ 13.2 | Length of shaft 2 |
| | 7.2 | $\leq L_3 \leq$ 8.8 | Length of shaft 3 |
| | 10.53e7 | $\leq G_1 \leq$ 12.87e7 | Shear modulus of shaft 1 |
| | 5.58e6 | $\leq G_2 \leq$ 6.82e6 | Shear modulus of shaft 2 |
| | 3.51e6 | $\leq G_3 \leq$ 4.29e6 | Shear modulus of shaft 3 |
| | 0.252 | $\leq \lambda_1 \leq$ 0.308 | Density of shaft 1 |
| | 0.144 | $\leq \lambda_2 \leq$ 0.176 | Density of shaft 2 |
| | 0.09 | $\leq \lambda_3 \leq$ 0.11 | Density of shaft 3 |