Proceedings of IDETC/CIE 2006
ASME 2006 International Design Engineering Technical Conferences &
Computers and Information in Engineering Conference
September 10-13, 2006, Philadelphia, Pennsylvania, USA

DETC2006-99643

MODELING DESIGN SPACES WITH DISCONTINUOUS VARIABLES USING NURBS HYPERMODELS

Cameron J. Turner
Los Alamos National Laboratory
PO Box 1663, MS J580
Los Alamos, NM 87545
cturner@lanl.gov or cturner@alumni.utexas.edu

Richard H. Crawford
The University of Texas at Austin
1 University Station, C2200
Department of Mechanical Engineering
Austin, TX 78712
rhc@mail.utexas.edu

Abstract

The vast majority of metamodeling demonstrations focuses on problems composed of continuous variables. However, important engineering design problems often include one or more discontinuous variables that require special attention. Previous work demonstrated the ability of Non-Uniform Rational B-spline HyPerModels to represent highly nonlinear functions composed of continuous variables. With minor modifications those capabilities can be extended to include functions defined by combinations of discontinuous input and output variables of different types, including discrete integer variables, feasibility variables and membership functions. Examples are used to demonstrate these modeling capabilities including applications developed from real engineering design problems such as the optimal positioning of a construction site crane and the optimal lay-up of a composite material I-beam.

1. INTRODUCTION

This research builds on previously published results to address the specific question of modeling discontinuous variables using Non-Uniform Rational B-spline metamodels, called Hyperdimensional Performance Models or HyPerModels. Turner [2005b] presented methods by which HyPerModels can be fit to continuous variables. Reviews of this paper raised the issue of whether the same could be accomplished for discontinuous variables. Turner [2005c] compares the performance of HyPerModels to other types of metamodels based solely on speed, accuracy and robustness. In Turner [2005a] this performance comparison was extended to additional metamodel types and additional HyPerModel capabilities were noted. Included among these capabilities are unique optimization properties, which form the basis for Turner [2006]. Each of these papers is distinct from this work.

1.1 INTRODUCTION

A metamodel is simply a model of models. Metamodels are used to encapsulate information from multiple simulations, experiments or other metamodels, themselves models of an actual system, into a single mathematical approximation, as conceptually shown in Fig. 1. In essence, a metamodel is a black box representation of an unknown system (or function).

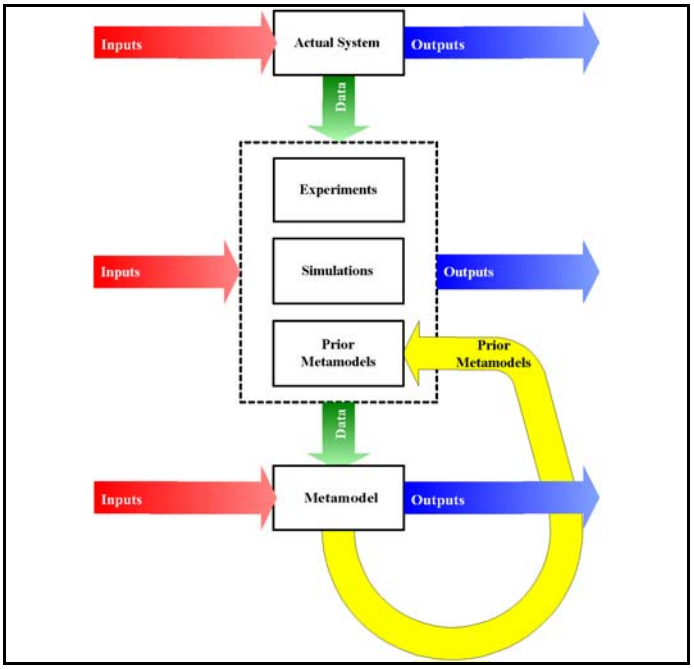

Figure 1. Metamodel Concept. Metamodels are a representation of a system of models, defined by data from one or more experiments, simulations or other metamodels.

A metamodel is a surrogate model for other applications, providing a fast and accurate data representation of the original sources with less expense than required from the original sources. Metamodels are increasingly common in engineering design applications where multiple model types are integrated (e.g. wind tunnel results with CFD and FEM simulations), control applications where as-built conditions render prior simulations inaccurate (chemical processing) and as objective function models in design optimization. [De Veaux, 1993; Rong, 1997; Simpson, 1998 & 2004; Sasena, 2002; Chandilla, 2004; McAllister, 2002]

For instance, a simulation or experiment results in a data point for each run conducted. So a series of simulations or experimental runs results in a data set, which is a subset of the underlying behavior represented by the simulation and/or experiment. A metamodel uses this data set to define a function approximating the behavior of the simulation or experiment – in effect filling in the spaces between the collected data points. Metamodels are particularly useful when data from multiple sources needs to be combined into a single representation.

If a large data set is already available from an exhaustive design space search, the data set may be directly sampled to create the metamodel. Otherwise, a data set can be obtained with adaptive sampling techniques [Sasena, 2002; Turner, 2004].

Several metamodel types are commonly used. Response Surface Models (RSMs), including classical curve and surface fitting techniques, are the most common. More complex techniques, including Kriging and Radial Basis Function Models are also employed as metamodels. All of these types are intended to represent continuous variables. Metamodels such as Multivariate Adaptive Regression Splines (MARS) and Non-Uniform Rational B-spline (NURBs) HyPerModels include capabilities that allow them to represent continuous and discontinuous variables. However, MARS is limited to binary integer input variables. [Salford Systems, 2001]

Many metamodeling surveys have been conducted, including De Veaux [1993], Laslett [1994], Barton [1998], Wang [1999a], Simpson [2001], Jin [2001], Sasena [2002] and Turner [2005c]. The interested reader should refer to these research papers for further discussion of the advantages and disadvantages of different types of metamodels.

1.2. METAMODEL DIMENSIONALITY

The dimensionality of a metamodel is a combination of the number of inputs (input dimensionality) and the number of outputs (output dimensionality). The input dimensions are the independent variables in a function while the output dimensions are the dependent variables from a function. In the case of a HyPerModel, the input dimensionality directly relates to the control point network dimensionality and the overall dimensionality directly relates to the control point coordinate dimensionality. Thus, the dimensionality of the metamodel also is related to its complexity. Metamodels are used to represent N-dimensional data sets; however, most examples presented in published works are limited to 3D or simpler functions that can be plotted with conventional techniques.

2. NURBS FUNDAMENTALS

Many Computer-Aided Design/Engineering (CAD/CAE) software systems use NURBs-based representations to describe geometric objects. However, until the work by Turner [2002], the literature shows little evidence for any similar development effort for NURBs-based metamodels. Turner [2005a] demonstrates that NURBs-based HyPerModels have attractive metamodel properties.

The mathematical basis for HyPerModels is derived from that of NURBs. For convenience, the fundamental NURBs equations are shown in this section. Equation 1, defines a planar NURBs curve, $\mathbf{p}(u)$, as:

$$\mathbf{p}(u) = \frac{\sum_{i=1}^{n_C} \mathbf{b}_i w_i N_{i,k}(u)}{\sum_{i=1}^{n_C} w_i N_{i,k}(u)} \qquad \text{for } a \le u \le b \tag{1}$$

where $\mathbf{b}$ is a vector defining the location of the i^{th} of n_C control points, w_i is a positive scalar defining the weight of the i^{th} particular control point, and $N_{i,k}(u)$ is the B-spline basis function given as a function of u. The parameter u defines a position along the curve length, which is equivalent to a point on the curve defined by the vector $\mathbf{p}(u)$. The B-spline basis function is a recursive function defined by Eqs. 2 and 3,

$$N_{i,k}(u) = \left(\frac{u - \mathbf{x}_i}{\mathbf{x}_{i+k-1} - \mathbf{x}_i}\right) N_{i,k-1}(u) + \left(\frac{\mathbf{x}_{i+k} - u}{\mathbf{x}_{i+k} - \mathbf{x}_{i+1}}\right) N_{i+1,k-1}(u) \tag{2}$$

$$N_{i,1}(u) = \begin{cases} 1 & \text{if } \mathbf{x}_i \le u < \mathbf{x}_{i+1} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

where $\mathbf{x}$ is the knot vector, a parametric value sequence defining the control point influence region within the NURBs metamodel. For the i^{th} control point, that region of influence is defined by the metamodel order, k. The B-spline basis function exhibits the behaviors defined by Eqs. 4[1], 5 and 6.

$$\tfrac{0}{0} \equiv 0 \tag{4}$$

$$\sum_{i=1}^{n+1} N_{i,k}(u) \equiv 1 \quad \forall\ k \text{ and } u \tag{5}$$

subject to the constraint $2 \le k \le n_C$ (6)

NURBs metamodels use the control point locations, control point weights (effectively a homogeneous coordinate of the control point), knot vectors, and the curve order, k, to produce a highly flexible curve definition. [Gopi, 1997]

Turner [2005a; 2005b] developed the HyPerFit algorithm to define HyPerModels using NURBs. Since the implementation of this algorithm allows continuous and discontinuous variables to be represented simultaneously, the

[1] Mathematically, the division in Eq. 4 is undefined. However, the convention in NURBs applications is to define 0 over 0 division as 0 for computational purposes. This definition is justified by the fact that 0 over 0 division occurs in the basis function definitions only when the basis function is inactive in the region queried.

next section reviews the fundamentals of the HyPerFit algorithm.

3. NURBS HYPERFIT ALGORITHM

Much of this section is based on work presented in Turner [2005b] but is reproduced here for convenience. The goals for metamodeling are somewhat different from the goals for geometric data fitting and surface reconstruction. The main requirement is a method that quickly generates accurate representations of data sets of unknown and arbitrary topology. The method should lend itself to representing spaces of arbitrary (and changing) dimensions, which the tensor product NURBs formulation readily supports. An overview of our fitting algorithm is given below and in Fig. 2:

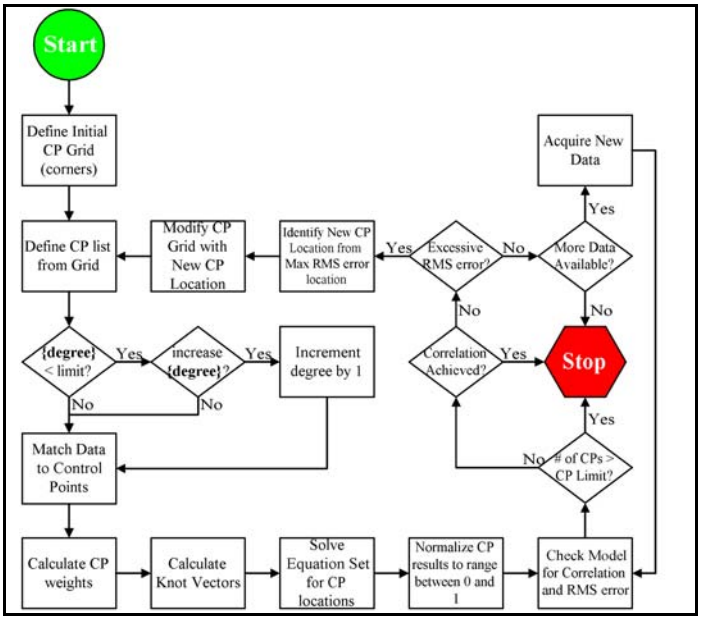

Figure 2. HyPerFit Algorithm. The basic fitting algorithm used in HyPerMaps to define a HyPerModel iteratively adds control points to the control net to reduce the maximum error in the metamodel. The model is refined until a stopping criterion is achieved.

1) Establish an initial linear model, with control points at each corner and an open knot vector. Control point weights are calculated based on the local data neighborhood near each control point. (This involves matching control points with their nearest data neighbors.) The control point locations are found through the solution of a simultaneous equation set.
2) Compare metamodel predictions to existing data set (used for fitting). Identify the maximum error location, compare this error to the user tolerance, and check the unused data set's correlation to the model data. Stop if the model has converged.
3) Insert a new primary control point at the equivalent parametric location.
4) Insert secondary control points as necessary to maintain the grid.
5) Increase the order as appropriate.
6) Recalculate the knot vector based on an open knot vector. Find internal knots by finding the midpoint between interior control points.
7) Calculate control point weights.
8) Calculate control point locations.
9) Repeat step 2 as necessary.

3.1. DATA PARAMETERIZATION

With a NURBs-based metamodel, several different variable types are used, unlike CAD applications where different coordinate system variables are a single variable type. The input (independent) metamodel variables are normalized to range from 0 to 1. These normalized variables correspond to the parametric NURBs coordinates and thus are known as parameterized coordinates. The metamodel output corresponds to the dependent variables and need not be normalized. These variables may be either continuous or a combination of continuous and discontinuous variables.

3.2. HYPERMODEL ORDER AND DEGREE

A NURBs metamodel initially uses an order of k=2, producing a linear model between bounding control points. As a third control point is added, the order is increased to k=3, producing a quadratic model. Previous trials [Legault, 2000; Turner, 2000] suggest that higher order models higher than quadratic produce little benefit, while diluting the local influence of control points. Consequently, model order is not increased further. If an input variable is defined as an integer variable, the order in that parametric direction is defined as k=2, producing a linear model in that parametric direction.

3.3. PARAMETRIC CONTROL POINT LOCATIONS

Unlike CAD/CAM/CAE applications, the parametric coordinates are determined independently of the dependent output coordinates of the control points. For instance, in a 2D planar plot where $y=f(x)$, y is a dependent coordinate and x is an independent coordinate that we parameterize so that:

$$u = \underline{x} = \frac{x}{x_{\max} - x_{\min}}, \tag{7}$$

where u is the parametric coordinate of the NURBs model, $\underline{x}$ is the parameterized (input) coordinate related to the normalized independent coordinate, x. Note that $u \in [0,1]$.

HyPerFit is initially *"seeded"* with a hypersurface generated from the corner control points. All of the initial control points lie at extreme values (0 or 1) of the parameterized input coordinates. Subsequent control points are iteratively identified based on the maximum root-mean square (RMS) error detected between the metamodel and the data set. This scheme begins with a minimal control net and inserts control points at the parametric location of maximum RMS error, along with additional control points so as to maintain a hypersquare of control points. Future versions of the algorithm will relax this constraint to maintain a hyperrectangular grid of

control points. The maximum RMS error location determines the parametric coordinates of all inserted control points for each iteration. Thus, for a 2D problem, the number of control points grows as m^D, where m=2, 3, 4… and D = 2, as shown in the example given in Fig. 3.

Once inserted, each control point is associated with its nearest neighboring data points. We assume that the nearest data point to each control point can be used to estimate the control point's dependent coordinate position. The local neighborhood of control points is used with a local kriging model to estimate the weight of the control point.

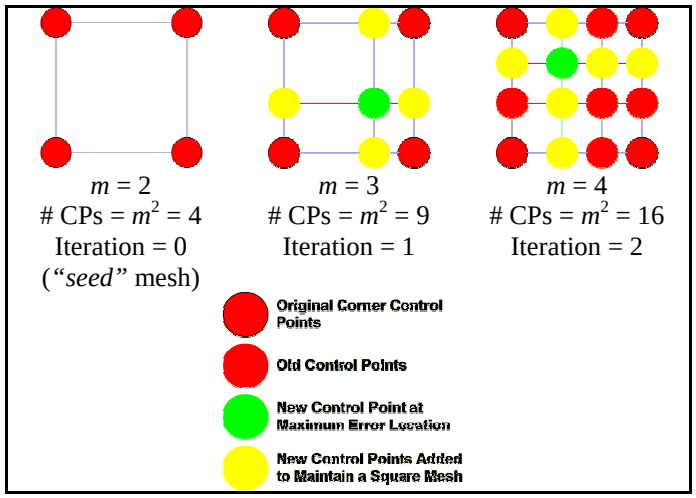

Figure 3. HyPerMaps Control Point Addition Scheme. The iterative control point (CP) addition scheme for a planar 2D input problem. The approach is readily extensible from 1 to N input dimensions.

3.4. ESTIMATING CONTROL POINT WEIGHTS

Each control point is associated with a predefined number (we use 10 in the examples below) of neighborhood data points selected by their proximity to the control point using only the parameterized coordinates. Using this neighborhood an appropriate control point weight is estimated with

$$w = w_{\min} + \left(w_{\max} - w_{\min}\right)\left(\mathbf{r}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{r}\right), \tag{8}$$

$$\text{recalling that} \quad u = \underline{x} \in [0,1] \tag{9}$$

$$\text{so} \quad 0 \le \left(\mathbf{r}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{r}\right) \le 1 \tag{10}$$

$$\text{and therefore} \quad 0 < w_{\min} \le w \le w_{\max} \tag{11}$$

where w is the weight of the control point, w_{min} is the minimum weight value, w_{max} is the maximum weight value, $\mathbf{r}$ is a vector derived from the spatial correlation function, $R(x_{CP}, u_i)$, relating the parametric control point location (x_{CP}) to the location of each nearby neighbor data points (u_i), defined in Eq. 12. The matrix $\mathbf{R}$ is derived from the spatial correlation function, $R(u_i, u_j)$, relating the location of data point u_i to the location of data point u_j, defined as:

$$R(x_{CP}, u_i) = e^{-\theta|u_i - x_{CP}|^p} \tag{12}$$

where θ defines the range of influence of the data and p defines the rate at which the influence of distant points will decrease. Reasonable parameter values can be obtained from w_{min} and the number of control points respectively, according to the relations defined in Eqs. 13 and 14.

$$\theta = \ln\left(w_{\min}\right) \tag{13}$$

$$p = \frac{\ln\left(\ln\left(C\right)\right)}{\ln\left(1/n_C\right)} \tag{14}$$

where C is a coefficient (C>1.0) defining the minimum weight of influence at the nearby neighborhood boundary. Values of w_{min}=0.1, w_{max}=1.0, and C=2 have yielded good results, and result in weights that range from 0.1 for a control point with little data near its location, to a value of 1.0 for a control point with many nearby and even coincident neighbors. In essence, the weight estimates confidence that can be placed in a control point location. Multiple dimensions can be handled through a tensor product of the single dimension weights.

A second set of weights is computed – one for monitoring the certainty with which we can identify a control point location, and another unitary "pseudoweight" used for the geometric fitting. Nonunitary pseudoweights are particularly useful for sequential sampling applications.

3.5. HYPERMODEL KNOT VECTORS

Consistent with the independent (input) variable parameterization, open knot vector(s), defined by the curve order and control point locations are used to define the HyPerModel. Intermediate values, which begin to emerge once n_C>k in a particular direction, are defined as the intermediate values between the interior control points. For example, the first intermediate knot location in Fig. 4, a k=3 curve defined by n_C=4 control points, lies halfway between control points 2 and 3 in the parameterized coordinate space. Note that the knot vector is defined in the NURBs parametric coordinate space. A knot vector exists in each parametric direction, and this vector is applicable throughout the NURBs hyperobject. In n-dimensional terms, these knot vectors are stored as a knot matrix with each column corresponding to an orthogonal parameterized (input) variable coordinate.

3.6. DEPENDENT CONTROL POINT LOCATIONS

With the model order, the parameterized control point locations, the control point weights, and the knot vectors determined, the only remaining unknowns are the dependent control point coordinates defining the metamodel outputs. Since each control point has an identified nearest neighbor data point that is the best approximation of the model at the parametric control point location, x_{CP}, a set of simultaneous equations can be defined (Eq. 15), such that

$$\left[\mathbf{N}\left(x_{CP}\right)\right]\left\{\mathbf{b}\right\} = \left\{\mathbf{p}\left(u_{NN}\right)\right\} \tag{15}$$

where $\mathbf{N}$ is the basis function matrix at a control point location, $\mathbf{b}$ is a vector of the independent control point coordinate locations, and $\mathbf{p}$ is a vector of the desired metamodel (output) values at the control point location based on the best available

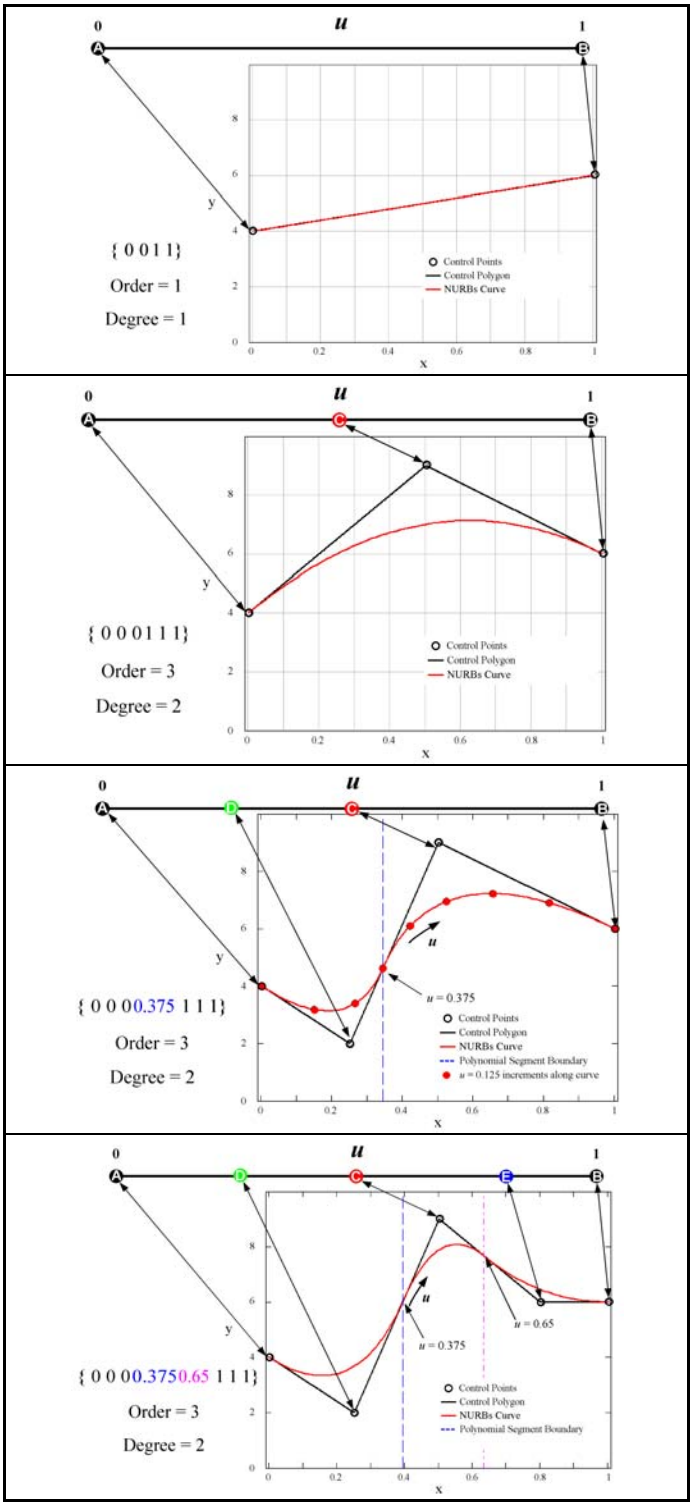

Figure 4. HyPerMaps Knot Vector Calculation. The evolution of the knot vector, from two control points (*first row*), to three control points (*second row*), to four control points (*third row*) and five control points (*fourth row*) along with the corresponding set of NURBs curves and the knot vector. As control points are added, the knot vector localizes the influence of individual control points.

information, the independent coordinate values of the nearest neighbor data point, u_{NN}. Note that x_{CP} and u_{NN} only exist in the parameterized coordinate space. u_{NN} should lie within the region influenced by the associated control point. The terms in the basis function matrix are simply the solutions to Eq. 16.

$$N_{i,j}\left(x_{CP_i}\right) = \frac{w_j N_{j,k}\left(x_{CP_i}\right)}{\sum_{l=1}^{n_C} w_l N_{l,k}\left(x_{CP_i}\right)} \tag{16}$$

Eqs. 15 and 16 are expressions of Eq. 1 as a set of simultaneous equations. Our control point fitting algorithm is based on algorithms described by Rogers [1990], and by Piegl [1997] that allow each parameterized coordinate to be fit individually, reducing the size of **N** at the expense of having to invert a smaller **N** in each parametric coordinate.

For each iteration, the solution of a set of n_i linear equations with n_i unknowns is required where n_i is the number of control points in the i^{th} parametric coordinate direction. This problem can be solved with standard matrix inversion algorithms such as an LU-factorization algorithm that decomposes **N**, into a lower triangular matrix **L** and an upper triangular matrix **U** for forward and backward substitution algorithms and matrix pivoting to maintain the population of the diagonals of **N**. [Griffiths, 1991]

3.7. HYPERFIT CONVERGENCE

Since HyPerFit is iterative, convergence criteria are necessary. The user defines three metrics that are used to test for convergence:

1) Model correlation calculated with Pearson's r^2 [Crow, 1960],
2) Maximum RMS error threshold, and
3) Minimum RMS error threshold.

Model correlation is restricted to data set elements not primarily associated with control point locations so as to avoid biasing the correlation metric. The unused sample set size must be at least as large as the used sample set size in order to be considered statistically significant.

Any data points whose RMS error is less than the minimum threshold are considered well approximated by the current metamodel iteration. However, it is possible to meet the global correlation metric and simultaneously exhibit large local RMS errors. Consequently, the maximum RMS error threshold prevents convergence due to correlation if significant local RMS errors still exist. RMS error is only a function of the dependent metamodel coordinates and is expressed as a percentage of full scale in the data set.

Correlation can occur if

1) The model achieves its correlation goals with a maximum RMS error less than the maximum RMS error threshold, or
2) If all data points are represented by less than the minimum RMS error threshold and no additional data is available.

Sequential sampling techniques can be integrated with the metamodel fitting process to collect data requested by the metamodel in this case.

3.8. HYPERMODEL REFINEMENT

HyPerFit initially defines a sparse control net and iteratively adds control points. In some cases, a control point added in a prior iteration should be removed because a subsequently added control point renders it unnecessary or even undesirable. This is particularly true when sequential sampling is used to generate a data set. Trials have shown that reverting to a sparse control net and completely rebuilding the metamodel from the collected data set allows sub-optimal control points to be removed, improving the accuracy of the HyPerModel.

4. HYPERMODEL VARIABLE TYPES

Many engineering design problems include combinations of continuous variables and discontinuous variables. Examples of continuous variables include part dimensions, operating temperatures, fluid flow rates, etc. Examples of discontinuous variables include material types, the dimensions of standardized parts (screws, bolts, etc), design feasibility, etc.

Discontinuous variables can occur in several ways. There are discrete or integer input variables (such as bolt size) where the variable can take on only a certain value. The terms, integer and discrete, are used interchangeably even though the discrete values may or may not take on integer values. Input variables can also be discontinuous if a void exists within the range of a continuous variable. Voids commonly occur when constraints are applied to the design problem. An example of a void might be a requirement for a train to maintain a velocity of at least 15 miles per hour or a velocity less than 8 miles per hour.

Discontinuous variables also exist as outputs to a design problem. For instance, an output that describes the feasibility of a design could be expressed as a binary output variable with two states: feasible and infeasible. Mathematically, such variables are expressed as step functions. More complex feasibility evaluations with multiple states can also be modeled as multiple step functions. These formulations are more generally known as membership functions.

5. DISCONTINUOUS VARIABLE HYPERMODELS

RSMs, kriging models, and RBFs are designed to deal with continuous variables. They have no provisions to explicitly deal with discontinuous input variables. MARS can incorporate binary input variables into its models [Salford Systems, 2001] but HyPerModels have more flexibility. HyPerModels with discrete integer input values or input variable voids also can be defined using membership functions or discrete integer output variables.

5.1. INTEGER INPUT VARIABLES

There are two ways to represent integer input variables with a HyPerModel: as multiple outputs (each corresponding to a particular integer value) or by embedding the integer values within the control point network structure. Defining the integer variables as multiple outputs reduces the HyPerModel input dimensionality, which also reduces the computational effort involved in defining the model. In effect, each possible integer value takes on a distinct output dimension. However, if there are already multiple performance indices in the problem, treating one or more integer variables as performance indices can dramatically increase the problem output dimensionality.

The alternative is to embed the integer variables into the control point network structure. This is accomplished by establishing control points at each integer value in the integer coordinate(s) of the problem, and by restricting the HyPerModel order in this direction to k=2. A HyPerModel of order k=2 in a particular direction, defines a model that will linearly interpolate between control points in this direction. Thus, the control point network defines the integer values for any integer coordinate direction(s) and constrains any optimum locations to coincide with integer values due to the linear constraint on the HyPerModel in the same direction(s).

The six-hump camel back function [Adorio, 2005] defined in Eq. 17 can be converted into an integer modeling problem by restricting x_1 to the set {-1, -½, 0, ½, 1}. Thus x_1 is defined as a discrete (integer) variable. The resulting metamodel is shown in Fig. 5. The similarity demonstrated between the actual function and the metamodel approximation is a highly desirable outcome of using a metamodel. However, the two representations are not identical with deviations measured through the global correlation coefficient, $r^2 = 99.9\%$.

$$f_{17}(x_0, x_1) = 4x_0^2 - 2.1x_0^4 + \frac{x_0^6}{3} + x_0x_1 - 4x_1^2 + 4x_1^4 \quad \text{for} \quad -2 \le x_0 \le 2 \quad \text{and} \quad x_1 = \{-1, -0.5, 0, 0.5, 1\} \tag{17}$$

The *"rows"* in the control point network correspond to integer values in the metamodel. Consequently, the metamodel is defined by a set of curves, rather than a continuous surface. The metamodel between the rows is a linear interpolation defined by a k=2 curve, while the metamodel along the row, is locally quadratic, defined by a k=3 curve.

5.2. VOIDS IN INPUT VARIABLES

Another possible input discontinuity is the case where a void exists in the data set. A region where data does not exist in the metamodel defines the void. Again, several approaches are available to cope with the void.

If the presence of the void is not known in advance, one possibility is to simply ignore the existence of the void in the output dimensions, except for an output dimension that defines the feasibility of the location. Regions in voids are associated with infeasible solutions in this dimension. This relies on the simulation or experiment to be able to detect infeasible solutions and produce reasonable output responses.

Figure 6 shows the result of creating a void within the Sasena [2002] sinusoidal function defined by Eq. 18. The region $4.5 < x < 6.5$ constitutes a void in the design space and also defines the corresponding feasibility dimension. Without any data provided by the void, the metamodel simply

interpolates across the void in the function. However, the feasibility dimension captures the absence of data in this void.

$$f_{18}(x) = 10 - \sin(x) - e^{x/100} \quad \text{for} \quad 0 \le x \le 10 \tag{18}$$

The difficulty in this approach is the reliance on the simulation's ability to detect a void. If no data is needed from the region defined by the void to define the metamodel, the void may not be detected. Consequently, the metamodel may predict a feasible solution in an infeasible region of the function. Furthermore, if data is obtained from experimental sources, the infeasible nature of the region in the function may mean that the data necessary to define the metamodel is unavailable. At a minimum, this highlights the need to confirm metamodel results with queries to original data sources.

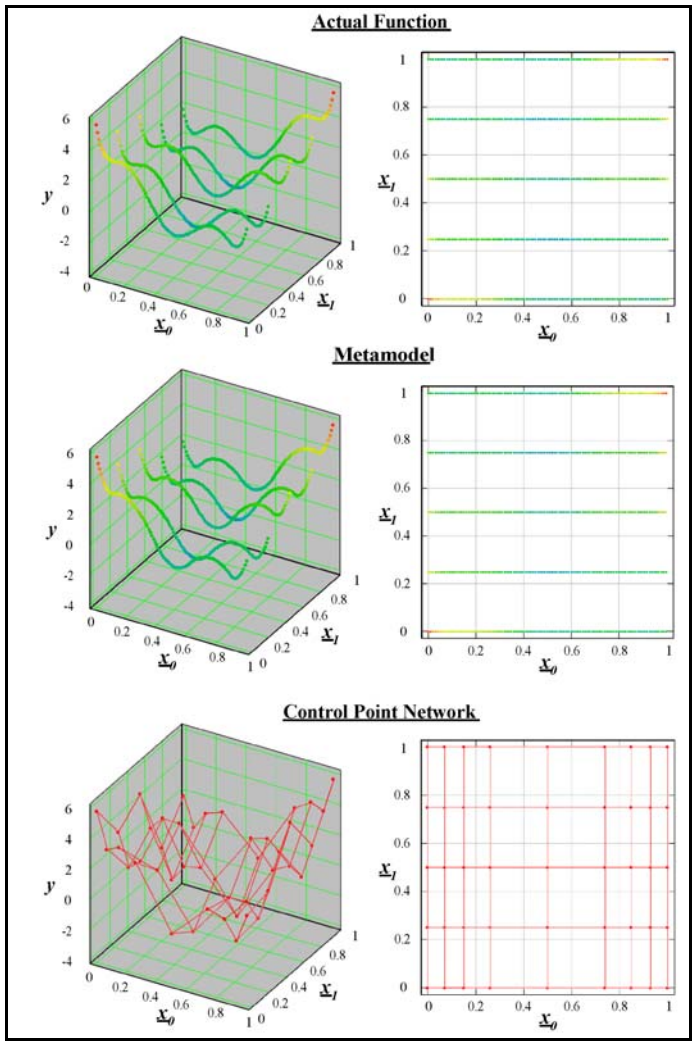

Figure 5. Integer Six-Hump Camel Back Function. If one of the axes of the six-hump camel back function is restricted to certain integer values, a HyPerModel can be constructed by embedding the integer behavior into the control point network. The actual function appears as a series of lines aligned with individual integer values (*top*), as does the resulting HyPerModel (*middle*). The lines are also aligned with rows in the control point network (*bottom*).

5.3. APPLYING MEMBERSHIP FUNCTIONS

A feasibility dimension in effect represents a membership function. A membership function is simply a function that defines which points in a range are members and which points are nonmembers of a set. In the case of the feasibility dimension in Fig. 6, membership is defined as the set of points x such that $0 < x \le 0.45$ or $0.65 < x < 1$, or the regions where the feasibility dimension has a value of 1. This is an arbitrary convention but is consistent with binary variables in C++ programming.

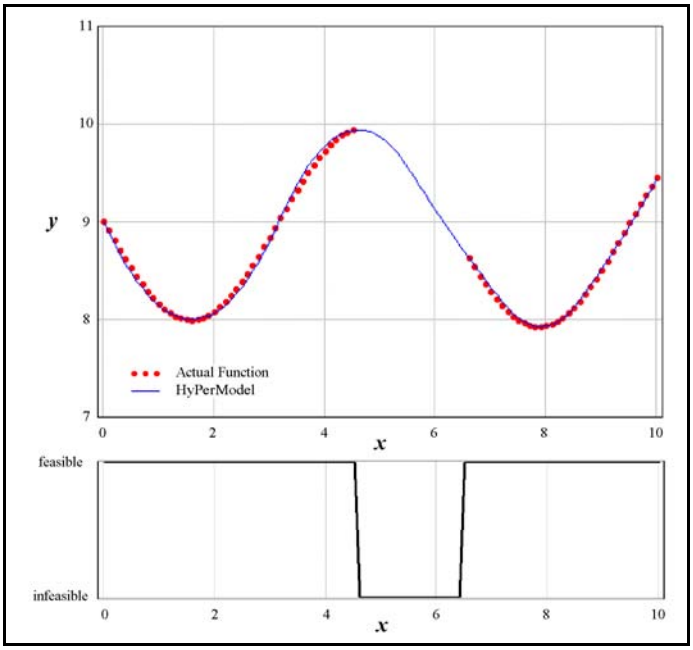

Figure 6. Sasena Sinusoidal Function with Void. The metamodel simply interpolates across the void defined in the Sasena Sinusoidal Function (*top*). If the simulation can detect that a void exists, a feasibility dimension (*bottom*) can also be defined.

Membership functions also allow the user to incorporate prior knowledge, if available, about voids within input variable ranges. Thus, the impetus for the simulation to provide all of the information about the presence of voids within the input variable ranges is decreased.

In 1D, a membership function is modeled with one or more step functions, such as the step function defined by Eq. 19 and modeled in Fig. 7. Step functions can be readily extended to higher dimensions, such as the 2D multiple step function shown in Fig. 8. The multiple steps demonstrate that a membership function can simultaneously model membership in several sets.

$$f_{A.3}(x) = if \begin{cases} x < 6 & 6 \\ x \ge 6 & 11 \end{cases} \quad \text{for} \quad 0 \le x \le 10 \tag{19}$$

5.4. INTEGER OUTPUT VARIABLES

Since the output of a membership function is a discrete (integer) value, a reasonable approach to modeling membership functions is to restrict the output to integer values. This can be accomplished through the application of rounding rules to the

continuous output of a HyPerModel, as was done to generate the membership model in Fig. 8.

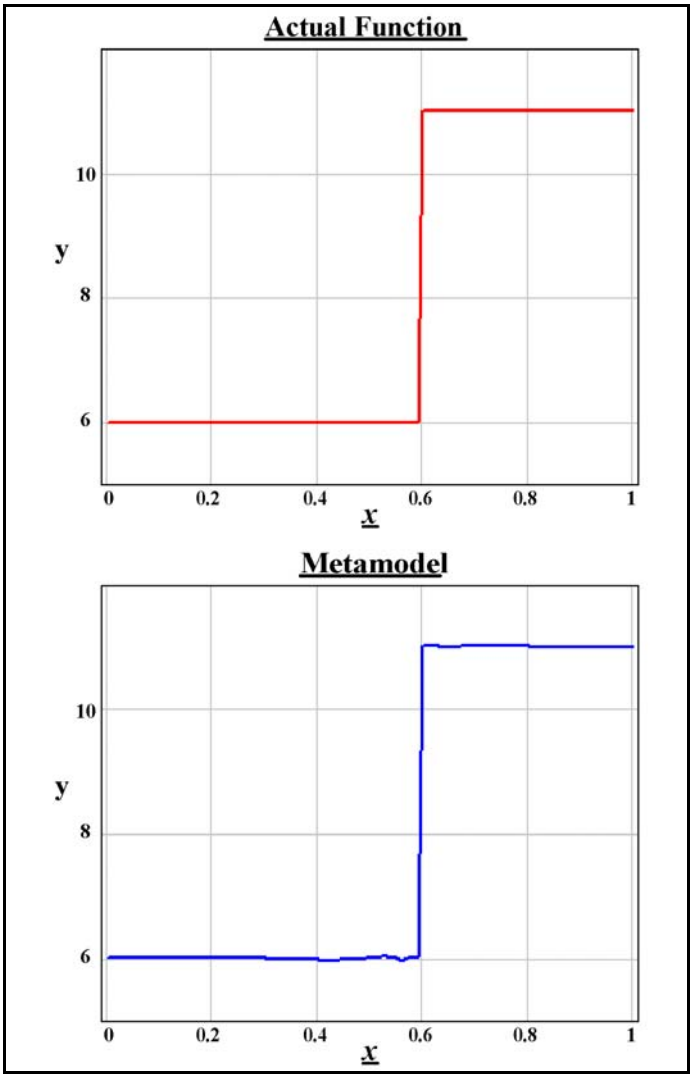

Figure 7. Step Function Example. The actual function (*top*) and the metamodel representation (*bottom*). Some residual variability still exists in the metamodel.

Since the cost of a HyPerModel, in terms of the number of control points, to model a membership function with continuous outputs can be significant for high dimensional input problems, applying these rounding rules within the fitting algorithm of the HyPerModel reduces the cost of fitting membership functions by reducing the number of control points necessary to model the membership function.

Both step functions demonstrated in Figs. 6 and 7 are based on rectangular topologies that compliment the rectangular grid topology of the control point network. However, voids may exhibit other, much more complex topologies that do not necessarily conform to rectangular grids. Membership functions defined by such complex topologies also can be represented with HyPerModels as demonstrated with the examples in Figs. 9 through 11.

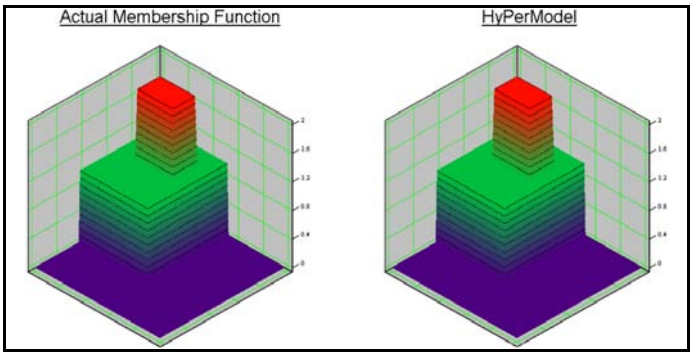

Figure 8. Multiple Step Function. A 2D step function, representing membership in three sets can be accurately modeled with a HyPerModel to a correlation of 100%.

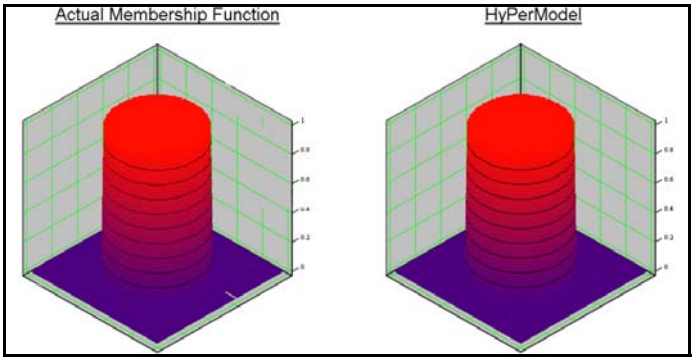

Figure 9. Circular Step Function. Representing a circular object with a rectangular grid of control points is a challenging task for a HyPerModel, but is achieved in this case with a correlation of 100%.

What is remarkable about several of these topologies is that the resulting HyPerModels also preserve the interior details of the membership functions. The mechanical part included holes through the central cube and the Stanford Bunny is in fact hollow. Figure 12 shows the interior detail within each object.

Once generated, HyPerModels of the membership functions allow set operations to be performed on metamodels. In terms of feasibility constraints, for discontinuous input variables, the primary interest is in calculating the intersection of two sets. This process was used to generate Fig. 12, using the set mathematics. Thus, complex voids can be calculated as shown in Fig. 13.

Applying the feasibility calculation to the raw data of a metamodel, such as that in Fig. 14, allows the metamodel to be restricted to the feasible regions of the problem. The example in Fig. 14 is for the crane location problem. In this example, the building at a construction site represents an infeasible position to locate the crane. The goal is to find the optimum feasible location for the crane within the construction site, but the building created a void in the construction site. The building is defined as an infeasible location by defining a membership function and feasibility constraints. The HyPerModel in this case was generated with adaptive sampling techniques.

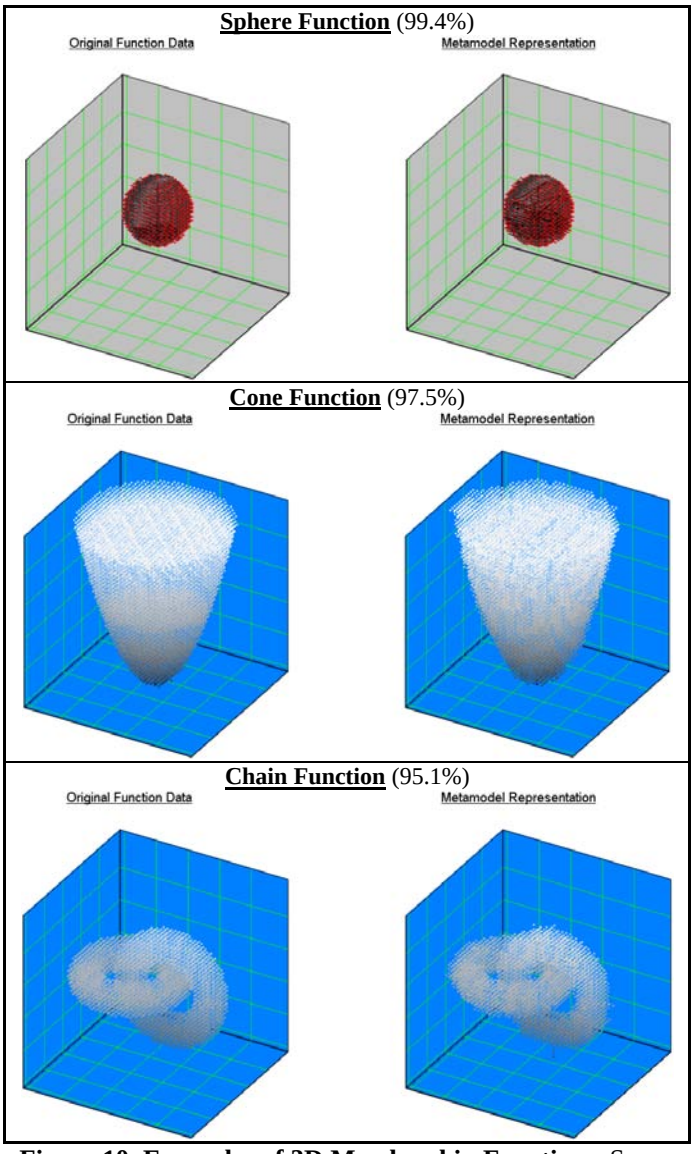

Figure 10. Examples of 3D Membership Functions. Seven examples of 3D membership function based on geometric objects. The correlation achieved by the HyPerModel (*right image*) to the actual function (*left image*) is given in parenthesis next to the name of each model. Plots are generated by only rendering the member points in the model.

5.5. DISCONTINUOUS HYPERMODEL SUMMARY

Together, the capabilities of HyPerModels to represent discontinuous integer input variables with voids allow HyPerModels to represent design functions representing complex engineering design problems. For instance, consider the design of a multi-laminate composite material I-beam, such as that shown in Fig. 15. Typically, composite material design is performed with an exhaustive search of a finite material set and a finite set of fiber orientations (i.e. 0°, ±15°, ±22.5°, ±30°, ±45°, ±60°, ±67.5°, ±75°, or ±90°) forming an integer (combinatorial) design problem. However, for a two ply design, the fiber angle can be considered a continuous variable and the material choice a discrete variable.

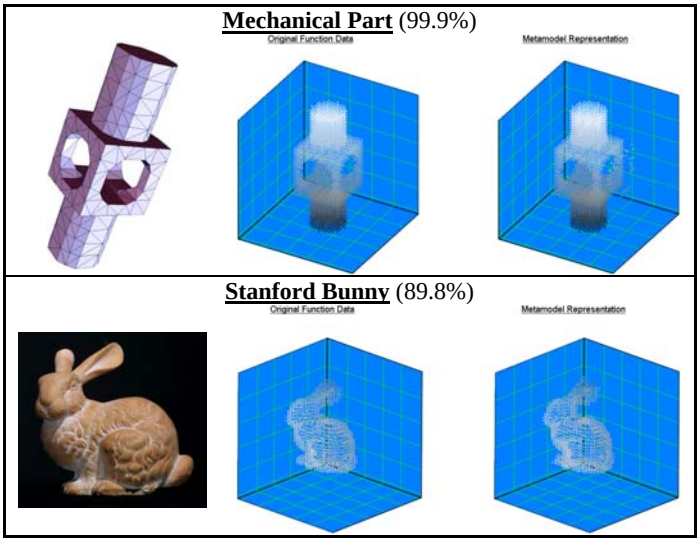

Figure 11. 3D Membership Function Examples for Real Objects. These membership functions are based on a solid model of a mechanical part (*top*) courtesy of the University of Texas at Austin Institute for Computational Engineering and Sciences and Computational Visualization Center [Zhang, 2005], and a 3D surface scan of the Stanford Bunny (*bottom*) courtesy of the Stanford 3D Scanning Repository [Stanford, 2005]. Shown are the actual objects (*left*), the actual function data sets (*middle*) and the HyPerModel representation (*right*).

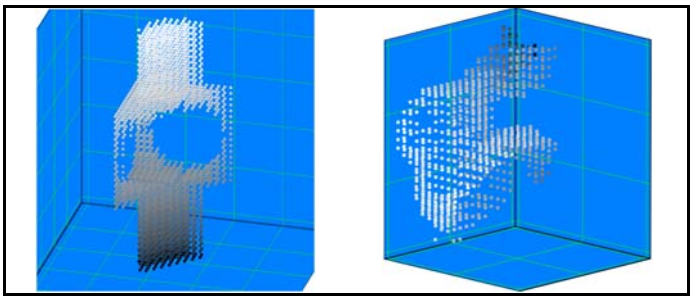

Figure 12. Interior Detail of 3D Membership Functions. Both the Mechanical Part (*left*) and the Stanford Bunny (*right*) include interior details. In the case of the Stanford Bunny, the data set represents a shell of points surrounding a void in the interior of the Stanford Bunny. The HyPerModel models the interiors as well as the exteriors of these objects.

The design objective for this problem is finding the lightest composite material configuration that meets or exceeds the stiffness of the I-beam to be replaced. This problem can be expressed as a Mixed Integer Programming (MIPs) optimization problem. Assuming that the geometry of the composite I-beam is the same as the original I-beam (in this case a W310 × 143 I-beam) only the composite material selected and the fiber orientation are unknown variables. Using penalty functions, infeasible designs (which would not meet the stiffness criterion) can be defined. The resulting design function can be represented with a HyPerModel as shown in Fig. 16.

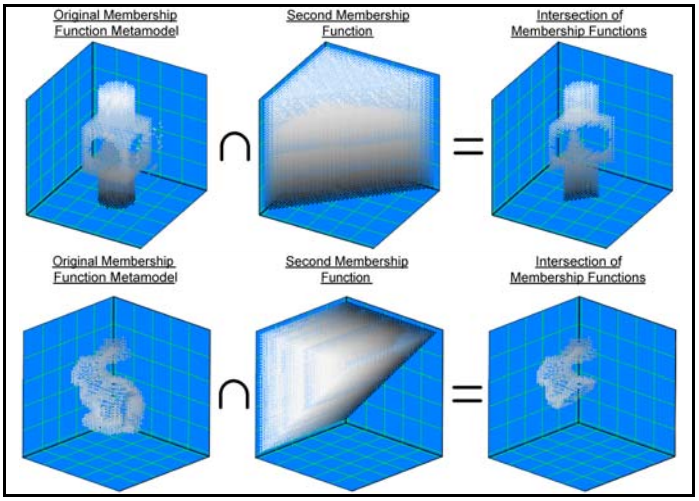

Figure 13. Intersection Calculation of Metamodels. The intersection of membership metamodels can be quickly calculated to yield the subset of points that are feasible members of both sets, yielding highly complex objects.

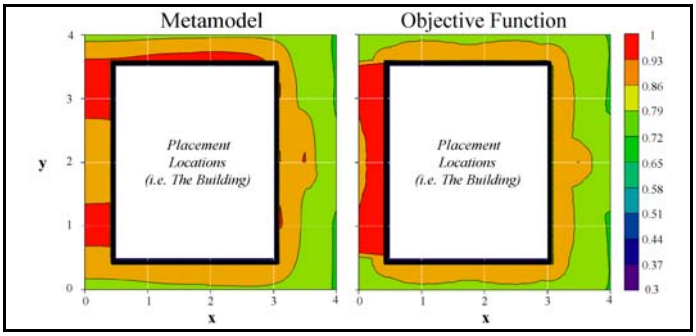

Figure 14. Infeasible Crane Locations. The building was defined as an infeasible position in this problem by using a membership function to define crane positions within the building as infeasible in both sequential sampling problems. A membership function, rather than the simulation defined the building location as infeasible.

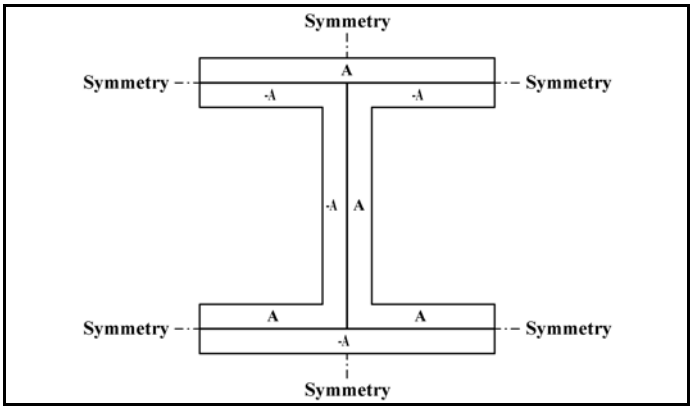

Figure 15. Antisymmetric 2-Ply Design Configuration. Antisymmetric composite layups define relations between fiber angles for plies arranged about a plane of symmetry. This design can be extended to cases involving larger numbers of plies.

It is well known in design optimization that the optimal solution to a MIP problem is not guaranteed to be the nearest integer solution. Nor is this how the optimal solution is located using the HyPerModel in Fig. 16 and the optimization techniques defined in Turner [2005a; 2006]. However, a detailed discussion is beyond the immediate scope of this paper. Interested readers are encouraged to review these sources for further information on using HyPerModels in optimization.

In addition, a large number of potential designs are now rendered infeasible since they would not result in an I-beam with sufficient stiffness to meet or exceed that of the original metal I-beam. The penalty functions incorporated into the design function in Fig. 16 also provide the ability to define a membership function that constrains the feasible regions of the HyPerModel, resulting in the regions shown in Fig. 17 for a Steel (Fe) I-beam replacement.

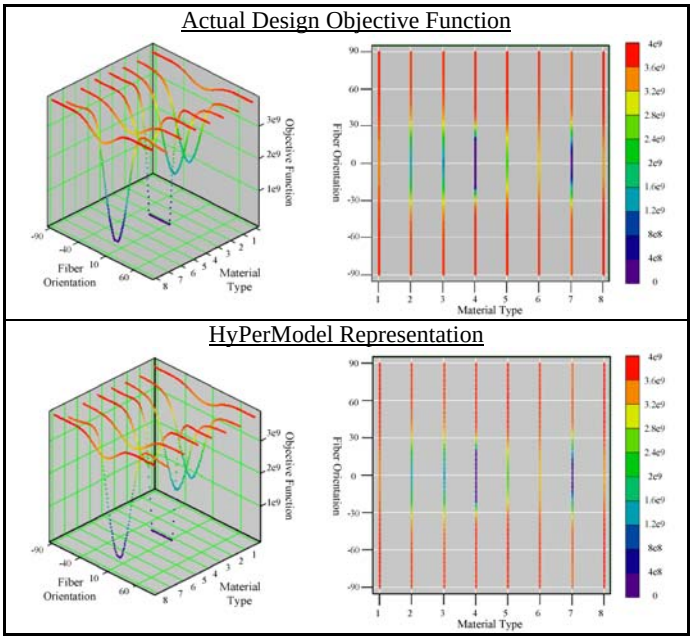

Figure 16. 2-Ply Design Function Models. The actual design objective function (*top*) and the HyPerModel approximation (*bottom*) indicate that the best designs are clustered in a small region of designs.

The HyPerModel of the design objective function includes a discrete integer variable (section 5.1) and uses a membership function (section 5.3) with an integer output value (section 5.4) to produce a void (section 5.2) in the resulting HyPerModel. Optimization results for these MIPs problems are given in Turner [2005a].

Turner [2005a] also developed HyPerModels of 75 trial functions defined by continuous variables and achieved an average global correlation of 99.0% with a standard deviation of 4.94%. The correlations for the 16 functions presented in this paper are summarized in Table 1. These functions have an overall average correlation of 97.4% with a standard deviation

of 4.11%. These values are comparable to those achieved for continuous variable functions.

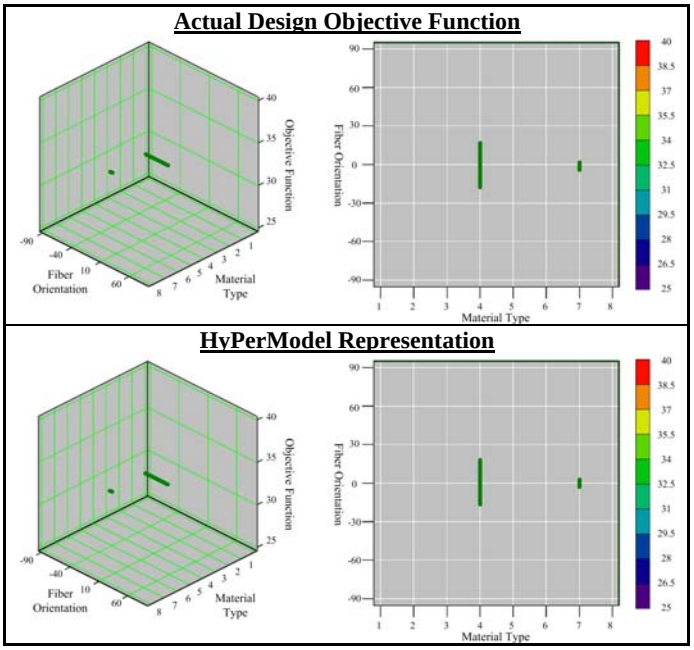

Figure 17. Feasible 2-Ply Design Function Models. The actual design objective function (*top*) and the HyPerModel approximation (*bottom*) after the infeasible designs have been eliminated from the metamodel with a membership function.

Table 1. Correlation Summary. The 16 functions used in this paper achieve an average correlation of 97.4%. The functions in *italics* are membership functions.

Function	Fig.	Correlation (%)
Integer 6-hump camel back	5	99.9
Sasena Sinusoidal with Void	6	99.8
Step	7	99.9+
Multiple Step	8	99.9+
Circular Step	9	99.9+
Box	n/a	99.9
Sphere	10	99.4
Cylinder	n/a	99.9+
Cone	10	97.5
Volcano	n/a	97.2
Torus	n/a	96.6
Chain	10	95.1
Mechanical Part	11	99.9
Stanford Bunny	11	89.8
Crane Location	14	85.8
Steel I-Beam Replacement	16	97.0
Average	-	97.4

6. CONCLUSIONS AND FUTURE WORK

By modifying the original HyPerFit algorithm, NURBs HyPerModels can model functions incorporating discontinuous variables. The resulting metamodels achieve correlations close to that achieved for functions defined by continuous input variables in Turner [2005a and 2005b]. The capability to model continuous and discontinuous variables is not matched by alternative metamodeling approaches. Thus, HyPerModels have a unique advantage in engineering design problems where discontinuous variables are common. The ability of HyPerModels to accurately represent complex functions defined by combinations of continuous and discontinuous variables supports unique design optimization and interesting modeling capabilities. [Turner, 2005a; 2006]

Future work is directed at improving the integration of these modifications into the current HyPerFit implementation in the HyPerMaps software system. Of particular interest is the ability for the HyPerModel to determine when an output variable exhibits integer behavior without user interaction. Additional research directed at examining the ability HyPerModels to assist in surface reconstruction from unstructured point cloud data also represents a potential avenue for further research.

ACKNOWLEDGEMENTS

This paper is approved for release by Los Alamos National Laboratory under LA-UR-06-3541. The assistance and support of Los Alamos National Laboratory and the Department of Mechanical Engineering at The University of Texas is greatly appreciated. This research was supported by the National Science Foundation under Grant No. DMI-0323838. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the author and do not necessarily reflect the views of the National Science Foundation or of Los Alamos National Laboratory.

REFERENCES

Adorio, E. (2005). *MVF - Multivariate Test Functions in C for Unconstrained Global Optimization*, Updated January 14, 2005, available at http://geocities.com/eadorio/mvf.pdf, Last Accessed February 6, 2005.

Barton, R. (1998). "Simulation Metamodels," *Proceedings of the 1998 ACM Winter Simulation Conference*, Medeiros, D.J., et al, eds., Washington, D.C., December 13-16, 1998, pp. 167-74.

Chandilla, P. (2004). *Strategy for Global Optimization and Post-Optimality Using Local Kriging Approximations*, Masters Thesis, Department of Aerospace and Mechanical Engineering, The University of Notre Dame, South Bend, Indiana.

Cohen, E., Riesenfeld, R., and Elber, G. (2001). *Geometric Modeling with Splines: An Introduction*, A.K. Peters, Natick, Massachusetts.

Crow, E., Davis, F., and Maxfield, M. (1960). *Statistics Manual*, Dover Publications, Inc., New York, New York.

De Veaux, R., Psichogios, D., and Ungar, L. (1993). "A Comparison of Two Nonparametric Estimation Schemes: MARS and Neural Networks," *Computers in Chemical Engineering*, **17**:8, pp. 819-37.

Gopi, M. and Manohar, S. (1997). "A Unified Architecture for the Computation of B-Spline Curves and Surfaces," *IEEE Transactions On Parallel and Distributed Systems*, **8**, pp. 1275-87.

Griffiths, D. and Smith, I. (1991). *Numerical Methods for Engineers*, CRC Press, Boca Raton, Florida.

Jin, R., Chen, W., and Simpson, T. (2001). "Comparative Studies of Metamodeling Techniques Under Multiple Modeling Criteria," *Journal of Structural Multidisciplinary Optimization*, **23**, p. 1-13.

Laslett, G. (1994). "Kriging and Splines: An Empirical Comparison of Their Predictive Performance in Some Applications," *Journal of the American Statistical Association*, **89**:426, pp. 391-400.

Legault, J. (2000). *A Complexity Management Framework for Open Architecture Agile Manufacturing Systems*, Masters Thesis, Department of Mechanical Engineering, The University of Texas at Austin, Austin, Texas, May 2000.

McAllister, C., Simpson, T., Hacker, K., and Lewis, K. (2002). "Application of Multidisciplinary Design Optimization to Racecar Design Analysis," *9th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, Atlanta, Georgia, September 4-6, 2002, AIAA-2002-5608.

Piegl, L. and Tiller, W. (1997). *The NURBS Book, 2nd Ed.*, Springer-Verlag, Berlin, Germany.

Rogers, D. and Adams, J. (1990). *Mathematical Elements of Computer Graphics, 2nd Ed.*, McGraw-Hill, New York, New York.

Rong, R., Lowther, D., Malik, Z., Su, H., Nelder, J., and Spence, R. (1997). "Applying Response Surface Methodology in the Design and Optimization of Electromagnetic Devices," *IEEE Transactions on Magnetics*, **33**:2, pp. 1916-9.

Salford Systems. (2001). *MARS™ User Guide*, Salford Systems, San Diego, California, 2001.

Sasena, M. (2002). *Flexibility and Efficiency Enhancements for Constrained Global Design Optimization with Kriging Approximations*, Ph.D. Dissertation, Department of Mechanical Engineering, The University of Michigan, Ann Arbor, Michigan, May 2002.

Simpson, T. (1998). *Comparison of Response Surface and Kriging Models in Multidisciplinary Design of an Aerospike Nozzle*, NASA/CR-1998-206935, ICASE Report No. 1998-16, NASA Langley Research Center, Hampton, Virginia.

Simpson, T., Peplinski, J., Koch, P., and Allen, J. (2001). "Metamodels for Computer-based Engineering Design: Survey and Recommendations," *Engineering with Computers*, **17**, pp. 129-50.

Simpson, T., Booker, A., Ghosh, D., Giunta, A., Koch, P., and Yang, R. (2004). "Approximation Methods in Multidisciplinary Analysis and Optimization: A Panel Discussion," *Journal of Structural Multidisciplinary Optimization*, **27**, pp. 302-13.

Stanford University. (2005). *The Stanford 3D Scanning Repository*, http://graphics.stanford.edu/data/3Dscanrep/, Last Accessed July 30, 2005.

Turner, C. (2000). *Developing Criteria for Actuator Resource Management*, Masters Thesis, Department of Mechanical Engineering, The University of Texas at Austin, Austin, Texas, August 2000.

Turner, C. (2002). "Metamodels for Planar 3R Workspace Optimization," CIE-34500, *Proceedings of the 2002 ASME International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, Montreal, Quebec, Canada, September 29-October 2, 2002.

Turner, C., Campbell, M., and Crawford, R. (2004). "Metamodel Defined Embedded Multidimensional Sequential Sampling Criteria," CIE-57722, *Proceedings of the 2004 ASME International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, Salt Lake City, Utah, September 28-October 2, 2004, LA-UR-04-3303.

Turner, C. (2005a). *HyPerModels: Hyperdimensional Performance Models for Engineering Design*, Ph.D. Dissertation, Department of Mechanical Engineering, The University of Texas at Austin, Austin, Texas, December 2005, LA-14264-T.

Turner, C. and Crawford, R. (2005b). "Adapting Non-Uniform Rational B-spline Fitting Techniques to Metamodeling," CIE-85544, *Proceedings of the 2005 ASME International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, Long Beach, California, September 24-8, 2005, LA-UR-05-3633.

Turner, C. and Crawford, R. (2005c). "Selecting an Appropriate Metamodel: The Case for NURBs Metamodels," DAC-85043, *Proceedings of the 2005 ASME International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, Long Beach, California, September 24-8, 2005, LA-UR-05-3632.

Turner, C., Crawford, R., and Campbell, M. (2006). "Global Optimization with NURBs-based Metamodels," *Engineering Optimization*, LA-UR-06-3543, *Accepted Pending Revisions*.

Wang, N. and Ge, P. (1999a). "Study of Metamodeling Techniques and Their Applications in Engineering Design," *ASME-MED Manufacturing Science and Engineering*, **10**, pp. 89-95.

Zhang, Y., Bajaj, C., and Sohn, B. (2005). "3D Finite Element Meshing from Imaging Data," *Computer Methods in Applied Mechanics and Engineering*, Available online May 13, 2005, http://www.sciencedirect.com/, Last Accessed July 30, 2005.