This is to certify that the

thesis entitled

AN OPTIMIZATION FRAMEWORK TO TUNE A LATTICE MODEL FOR IMPROVING CRASHWORTHINESS USING SURROGATES

presented by

Rakesh Kumar

has been accepted towards fulfillment
of the requirements for

MS degree in Mechanical Engineering

Major professor

Date 12 DEC 2005

AN OPTIMIZATION FRAMEWORK TO TUNE A LATTICE MODEL FOR IMPROVING CRASHWORTHINESS USING SURROGATES

By

Rakesh Kumar

A THESIS

Submitted to
Michigan State University
in partial fulfillments of the requirements
for the degree of

MASTER OF SCIENCE

Department of Mechanical Engineering

2001

ABSTRACT

AN OPTIMIZATION FRAMEWORK TO TUNE A LATTICE MODEL FOR IMPROVING CRASHWORTHINESS USING SURROGATES

By

Rakesh Kumar

The goal of this work is to develop a framework for optimization problems involving complex expensive functions for which traditional optimization techniques are not practical. Important concerns in such problems are high simulation cost and unreliable or unavailable derivative information. It's a common practice to work on a surrogate model (also called metamodel) of the original problem. We have used a 'kriging metamodel' to build the surrogate of the original problem and pattern search methods to solve the surrogate optimization problem. In turn the surrogate model is used to predict the optimum of original problem. We investigate the use of a lattice structure as a possible surrogate model for a more complex structure. We are concerned with design for crashworthiness of structures and an attempt has been made to 'tune' the lattice such that the response of lattice can be used to predict the response of the structure. The advantage of using such lattice model is low computational cost and less modeling time which, can help reduce the design cycle time and improved accuracy.

To my parents

ACKNOWLEDGEMENTS

This dissertation and the research would not have been possible without guidance of my advisor Dr. Alejandro Diaz. I really appreciate his valuable advice and patience. I want to acknowledge my thanks to Dr. Ronald Averill and Dr. Andre Benard for their support and valuable suggestions. I also want to thank my mentor Dr. Dinesh Balagangadhar for his invaluable help in developing skills which I consider a transparent part of this research. I wish he was here to share the achievement of this particular goal.

Ms. Cori Ignatovich has made many valuable contributions to this work. I appreciate her efforts. She is a wonderful person with great patience.

My parents, Mr. Jugal Kishore and Mrs. Indira Devi, always supported me with their infinite love and patience without any reservation. My family and friends have always inspired me for constructive research and I treasure them for the successful completion of this work.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Introduction

1.1 Motivation

There are many optimization problems in industry where the cost of arriving at an optimum solution is very high because of very 'expensive' objective functions or their derivatives. Computational optimization can guide us to better results and shorter design cycle times. In engineering, better designs can lead us to better performance of the product or process being designed. In many engineering disciplines better designs have many advantages such as reduced cost and better performance. Also, there are other areas where current design practice has reached heights; small improvements can be valuable addition. Faster turn-around time and shorter design cycle can improve the quality of product early in design cycle. When time is a critical factor and many design features cannot be changed later in the product development, computational design simulations can allow more flexible design to experiment.

Long solution time in the early stages of design cycle can negate many good features of design. We must look for methods which can give us competitive designs at reduced cost and time. In the present scenario we have simple and less accurate simulations but now the simulation of complex and accurate system is the subject of research in many industries. For such sophisticated simulations we need to have a framework of computational design which can satisfy such high requirements.

1.2 Background

The basic idea of replacing a complex function or constraints by using Taylor series is a very common practice. The more general work that has been the subject of research is the field of nonlinear programming. This has focused on Taylor series methods and gradients-based approach. Much less work has been done for derivative free methods ([6], [7], [8], [9]). The models used in engineering application can be classified mainly into two areas: numerical solution of governing equations of physical systems and functional approximation of the solution of equations constructed without actually knowing the physical systems, that is, by using the values of the function only. The former model characterizes the function at all points in the domain whereas later method approximates the function in the domain except at some points where it has the actual value of the function.

1.3 Statement of the problem

It is not uncommon to use computer simulation to arrive at decision before actually manufacturing the product. The industry offers gamut of choice ranging from a bicycle to a spacecraft. It takes considerable effort to model and simulate tests on computer. It is even more expensive sometime to test the actual product. Imagine a crash test performed on a car model. There are hundreds of design parameter and millions of degree of freedom associated with each test. The cost of such test can hinder to actually arrive at 'optimized' design. Replacing such expensive test by computer simulation may reduce

the cost of experiments by a considerable amount but it may take long time. It is a normal practice in industry to perform such simulation with the help of powerful computers. In the present work an attempt is being made to develop simulation methods, which allow maximum use of existing information to build a model, which has optimized design features. We have our problem definition as:

Minimize f (x) (1.1)

Subject to $a \le x \le b$

Where f: $R^n \rightarrow R \cup \{\infty\}, a; b \in R^n$

A typical multi-modal function can be shown in Figure 1

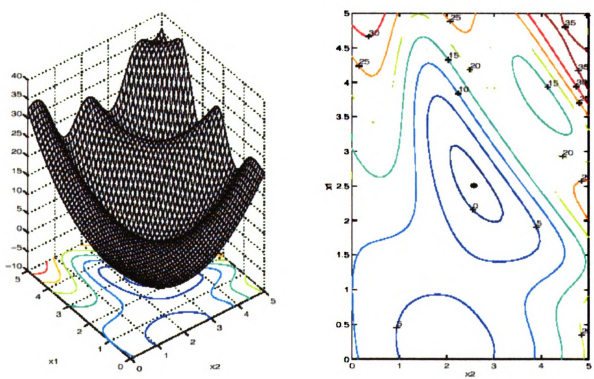


Figure 1: A two-dimensional multi-modal function.

(Images in this dissertation are presented in color)

- Physical problems can be very 'expensive' to model and may require many intermediate and state variable determinations to actually evaluate the function. Cost of evaluating f (x) could be very expensive in terms of computational time and resources. Some evaluation of f (x) could be available from other sources. If x is infeasible, f (x) may not be available.
- It is not necessary that f (x) be evaluated at all the possible design sites. There could be cases when f (x) need not be evaluated accurately at some design sites and it incurs the same computational cost. The function may fail to evaluate at some design points.
- The problem can be more acute if the optimization algorithm requires derivative information of the function. The derivative information may not be reliable and this certainly hinders derivative based optimization algorithms. Traditional non-linear optimization methods can be either inaccurate or very costly to use. Quasi-Newton method may not be a good choice in the light of foregoing discussion [1].

It is assumed that for application of interest in this work, the derivative information of function is not available or is very expensive to compute. A common approach for such problem is to replace the original expensive model by an approximate and cheaper model and then perform the optimization on it; we refer to this as the surrogate model. Clearly, the surrogate model will have cost advantage for evaluation and thus making the optimization problem easier. Another approach could be to use such optimization techniques which require function values and does not require derivative information.

The major disadvantage of replacing the original model by a surrogate model is that solution of surrogate model optimization problem may not be solution of original problem (We are actually solving a different problem). A significant disadvantage of using derivative free optimization methods is that they require many function evaluations as compared to gradient-based techniques. A framework which can solve most (if not all) of the problems addressed above will be chosen.

1.4 Selection of Framework

The computational cost of highly expensive simulations of the characteristics of physical process or systems makes it impractical to rely exclusively on such expensive simulations for the purposes of design and design optimization. Instead one needs to make as much use as possible of surrogates of lower physical or mathematical fidelity but lower computational cost, with only occasional recourse to expensive, high fidelity simulations. This is employed in many engineering disciplines especially in preliminary design where design cycle is very fast and one need to widely explore the design space. This approach is also in keeping with a tenet of nonlinear programming that suggests that one should try to avoid doing "too much work" when far from an optimum.

In the view of the foregoing discussion about function (1), there are many optimization methods proposed in the literature, which do not require any derivative information. The most widely used algorithm is direct search algorithm [2, 3, 4, 5]. These methods are affected very little by inaccuracies in function evaluation, as compared to other methods.

The properties of direct search method certainly qualify to use in our problem. However, the use of direct search method could be very expensive which does not satisfy our foremost important criterion of choosing a framework. The direct search method will be modified consequently to suit our need.

In the literature, the basic strategy used to tackle such problems is to work on a surrogate model problem (S), which is an approximate representation of the true function. The surrogate problem can be built in the following three ways:

- A physical approximate model (S_p), which closely represent the behavior of the true function.
- An approximate modeling of the characteristic behavior obtained by some data fitting technique based on sample design sites chosen by some mean (S_k).
- A combination of two method outlined above $S_p + S_k$.

After the surrogate is built we try to find the optimum X_s of this surrogate S. The true function is tested for f (X_s) and verified with respect to some reference f (X_{ref}) to determine if some improvement has been made. A very common method is to build the surrogate S before the optimization process and is not modified until the optimum of the original model is found. A basic premise of the method is to modify the surrogate S during the optimization process and the strategy of modification is guided by the value of original model evaluated during the optimization.

1.5 Example

For a better understanding of the problem at hand, an example (our test problem) is described next. In developing a crashworthiness model for a complex structure it is usual practice in industry to work on a structure already complete in its entirety. Computational modeling using this approach becomes a very high-fidelity simulation, with all design details incorporated into the model. If simulations of crash tests suggest changes in design parameters (as it often occurs), redesign can be even more expensive and time consuming. Our surrogate model build for high-fidelity simulations is an attempt to perform such crashworthiness analysis (or any other analysis) early in conceptual design stage of the product design.

Recent work in [20] has looked into optimization problems in the conceptual design stage. For *layout* optimization, one can use a *lattice* model that replicates major structural components. As described in [22, 29] lattice models are used to represent the behavior of a complex structure in an impact event. A lattice model developed for such analysis can be completely described with dimensional details including the material properties of its components. It becomes important to optimize these parameters in order to replicate the *relevant* behavior of the complex structure. In the present work an optimization problem has been formulated to investigate whether or not *relevant* measures of performance in optimization for enhanced crashworthiness are replicated with sufficient accuracy by the simplified, lattice model.

As described above, an attempt has been made on improving the accuracy of lattice models in crashworthiness analysis. An optimization problem is formulated to "tune" the lattice model to capture the relevant behavior of the complex structure. Kriging (introduced next) is used to develop a surrogate which can identify the systematic component of a signal characterizing the structural behavior from its random component. In this work kriging was used to develop a surrogate model. Named after Krige, a South African geologist who first developed the method, kriging is a type of surrogate model based on Spatial Correlation Functions (SCFs). Researchers were aware of this technique but it was not very widely acceptable until four statisticians wrote a paper on the kriging and this became a powerful mathematical technique ([14], [15]). Details about this method are included in chapter two of this dissertation.

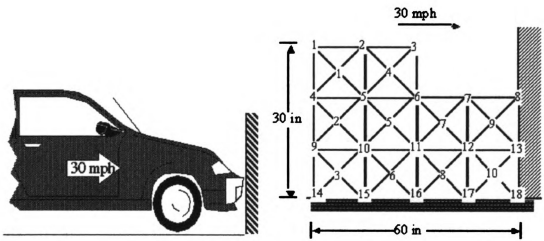


Figure 2: An example of a lattice model [22]

1.6 Outline of dissertation

The rest of this dissertation is arranged as follows. In chapter two, some theoretical background about the possible approach or a combination of them is outlined. The approach used in the present work is described in detail with emphasis on a 1-dimension problem. Chapter three describes one test example of tuning a lattice model for enhanced crashworthiness analysis followed by conclusion and future work in chapter four.

Chapter 2

Framework Design

2.1 Selection of Initial Design Sites

We will re-visit the original problem introduced in chapter one, before we consider some design issues in this chapter. A general optimization problem is described as follows:

Minimize f (x) (1.1)

Subject to $a \leq x \leq b$

Where f: $R^n \rightarrow R \cup \{\infty\}, a; b \in R^n$

It is worth emphasizing that cost of evaluating f (x) could be very expensive in terms of computational time and resources and the derivative information for f (x) may be unreliable or unavailable. Though computer simulations are cheaper and readily available than actual experiments, still they are time consuming and expensive. The goal here is to create an approximation of function f (x) by sampling it at some judiciously chosen x, the *design variable*. If f (x) is nonlinear and is of very high fidelity, this poses a difficulty of having a limited number of combinations of x. A feasible approach could be to develop a statistical framework from the results of 'as many' available function evaluations for some combinations of x and then use it to predict f (x) at any design point x in the

feasible design domain. The question is: How to sample x? Design of Experiments (DOE) provides several techniques to sample appropriate x or combinations of x.

2.2 Space Filling Designs

In this work the focus is on 'expensive' functions with high dimension and there is a possibility that we do not have a reliable evaluation of the function at all the design sites. In view of large dimensionality of the problem, 'space filling' experimental design could give promising *initial design sites*. There are several 'space filling' designs available in literature of which full factorial design, random Latin hypercubes, maximin Latin hypercubes, random orthogonal arrays, uniform designs, orthogonal Latin hypercubes and IMSE optimal Latin hypercubes are very commonly used methods. Few of these methods are described and investigated in present work.

2.2.1 Latin Hypercube Design

A Latin hypercube design (LHD) is a class experimental design that is defined as an n x k design matrix in which each column is a permutation of (1...n) which could be mapped onto the actual dimension, n being the number of *levels* and k is *factor* (number of design variable). The n x k design matrix can be explained as follows (particularly for Random Latin hypercube design): The bounds of each design variable are divided into n *levels*, and one *level* is chosen using random sampling with each *level* (or with certain criteria; to

be discussed later). Thus, there are *n* such chosen *levels* for each of the *k* design variables. One of the *levels* on x_1 is randomly selected (each *level* is equally likely to be selected), and matched with randomly selected *levels* on x_2, and so on through x_d. All these *levels* together constitute a possible design site P_1. For second design site P_2, one of the remaining *levels* on x_1 is then randomly selected and matched at random with one of the remaining observations on x_2 and so on to get P_2. A 10 x 5 Latin hypercube design, with 10 *levels* and 5 *factors* (design variables), is shown in figure 3.

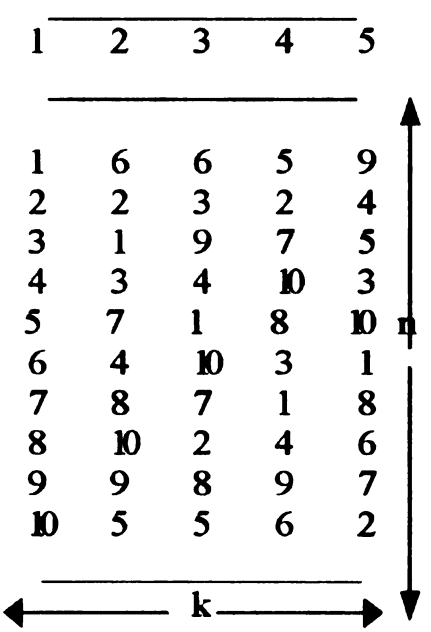


Figure 3: A Latin hypercube design with *level* 10 and *factor* 5.

Latin hypercube designs are stratified sampling which means that each design variables are sampled at n distinct *levels* so that none of the design sites are replicated. If Latin hypercube is projected into any single dimension, one can see exactly *n levels*.

This design was first proposed by (McKay et al 1979) and became very popular in computer experiments. Usual approach to generate a LHD matrix is to use optimal design criteria such as Integrated Mean Square (IMSE) (Sacks et al 1989), entropy (Wynn and Shewry 1987) and minimum inter design site distance (Johnsan et al 1990). In later sections of this chapter, some of these criteria are briefly introduced for generating LHD. These design criteria have been shown to be efficient for certain models. The construction of an optimal LHD can still be time consuming. Interested readers can refer to review of design and analysis of computer experiments, Koehler and Owen (1996).

In the design of experiments (DOE), there has been an important concern for many researchers to come up with an algorithm for automatically generating initial design points. The goal in DOE is to select the combination of design variables to be sampled. There are several criteria which can be used to measure experimental design's capability. One can choose a criterion based on proposed surrogate to model the behavior of the function. Other criteria could be symmetry of distribution of variance resulting from selection of design sites, ease of generating designs sites and number of experimental runs required to generate such combination of samples.

2.2.2 Full Factorial Design

Full factorial design (FFD) is one of the simplest experimental designs. FFD suggests fixing each input variable (dimension of design space or *factor*) at certain number of

levels. A combination of all such *factor*'s *level* is sampled as FFD. For example, in a design domain of dimension two, if the number of *levels* are 2 for each *factor*, we will have 2^2 design sites. The design sites thus generated are at the corners of a square. Similarly, in design domain of dimension three, if the number of *levels* are 2 for each *factor*, we will have 2^3 design sites. The design sites thus generated will be at the corners of a cube. Figure 4 shows one such examples of 3^2 full factorial design.

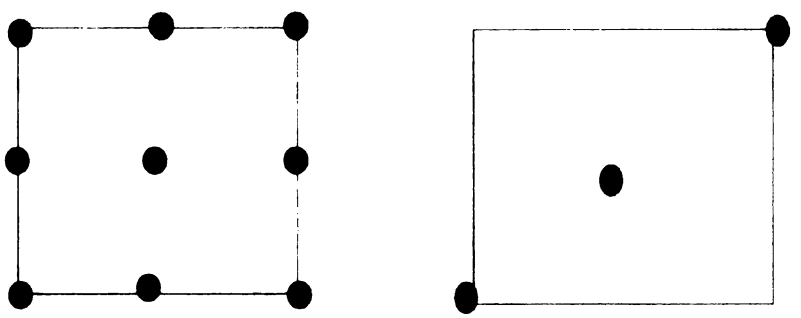


Figure 4: 3^2 Full Factorial Design and $3^{(2-1)}$ Fractional Factorial Design

It is easy to appreciate that as the number of *factors* and *levels* increase, the number of function evaluations required to generate a full factorial design is increased exponentially. Fractional factorial design can reduce the number of function evaluations at the cost of reduced refinement of design space and reduced accuracy of surrogate model. For example in a space of dimension two, if we have *k factors* (design variables) and *m* as reducing *factor*, then $2^{(k-m)}$ fractional factorial design sites can be generated. Number of designs is reduced by 2^m as compared with full factorial design. Figure 4 shows one such example of $3^{(2-1)}$ fractional factorial design. This is computationally less expensive as compared to full factorial design.

2.2.3 Random Latin Hypercubes

As explained in section 2.2.1, a Latin hypercube is a matrix of n x k dimension. In *random* Latin hypercube, each column has n different *levels* from 1,2...n *randomly* permuted and the k *random* columns are matched to form the Latin hypercube. Three random Latin hypercubes of dimension 2 (k =2) with 10 levels (n=10) are shown in Figure 5. It should be noted that the design sites in all the three cases are randomly selected throughout the design space. No optimality criteria (e.g. IMSE, Maximin, etc) has been used to generate such designs.

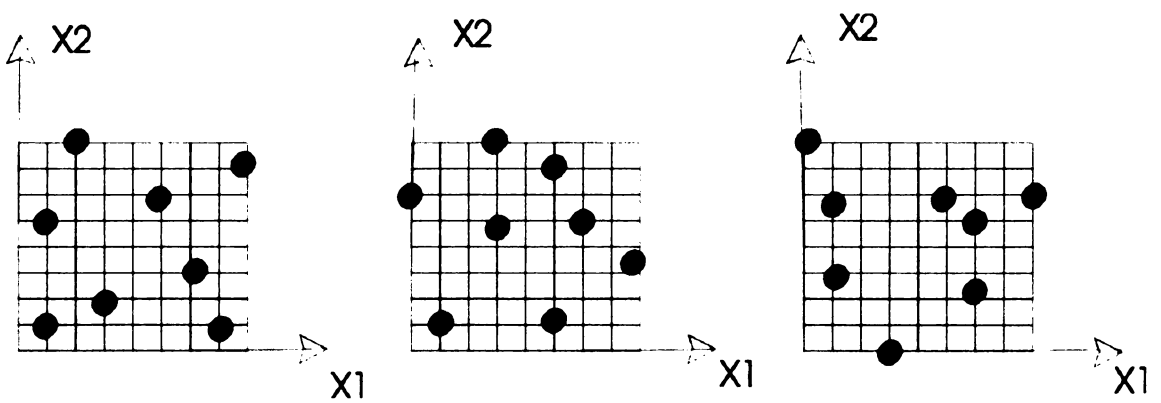


Figure 5: Random Latin hypercube design with a *factor* 3

The advantage of *random* Latin hypercube is that it requires only a random permutation of n *levels* in each column of the design matrix and is very easy to generate. It is computationally less expensive.

2.2.4 IMSE Optimal Latin Hypercube

Integrated mean square error (IMSE) design is generated with integrated mean square error over the design space between design sites as the optimality criteria. Integrating IMSE optimal designs and Latin hypercube designs (LHD) generate a hybrid set of a design which is referred to as optimal Latin hypercube design (OLHD). The striking features of this design methodology are that they are well spread in the design domain (because of IMSE criterion) and none of them are replicated (because of LHD criteria). They are often nearly symmetric and they are also stratified (because of LHD criteria). Two such design samples are shown in Figure 6 for 7 points OLHD and 8 points OLHD with a *factor* of two. It can be appreciated how well the design sites are spread in the domain with symmetricity though entire space is not filled. Park (1994) has developed an algorithm which can be used to generate such design; the reader is referred to (Park, 1994) for details about this algorithm.

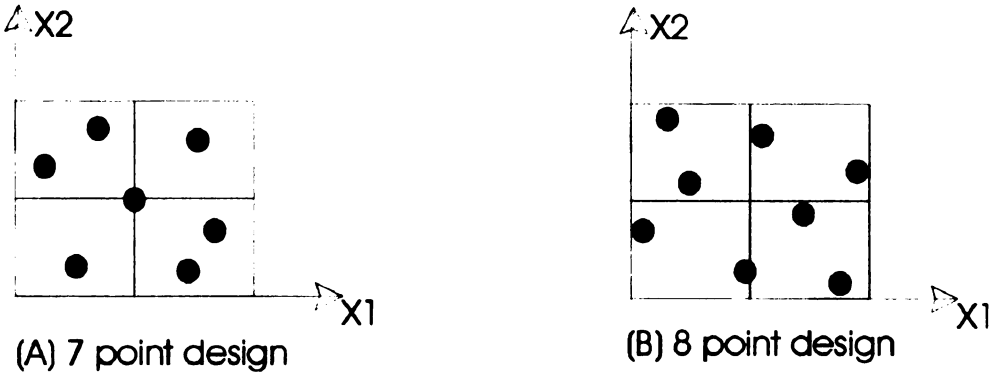


Figure 6: IMSE optimal Latin hypercube design with a *factor* 2

2.2.5 Maximin Latin Hypercubes

Maximin Latin hypercube design (MMLHD) method was developed by Morris and Mitchell (1995) for computer simulations. They used *maximin* distance as optimality criterion which is used to maximize the minimum distance (Euclidean or rectangular) between any two design sites. This ensures that design points will be spread as far as possible in the design domain. Figure 7 shows a two *factor* MMHLD with 7 and 8 points design sample. Morris and Mitchell used a simulated annealing search algorithm to generate these designs.

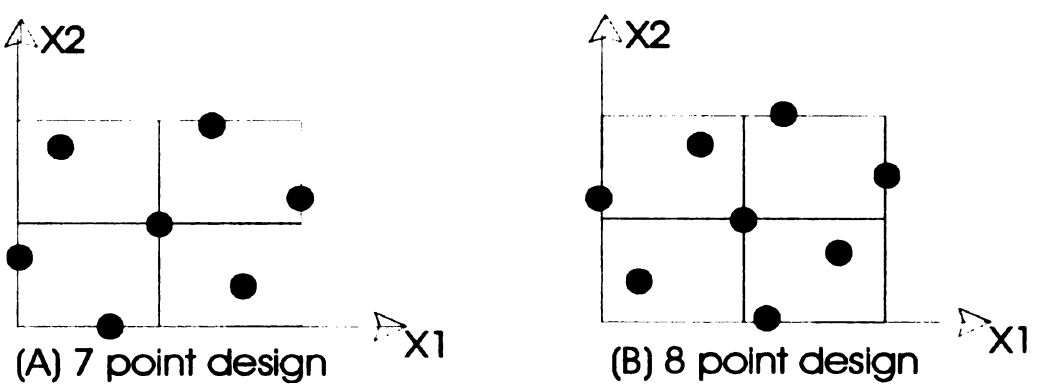


Figure 7: Maximin Latin hypercube design for a *factor* of 2

2.2.6 Orthogonal Latin Hypercubes

Orthogonal Latin hypercube design (OLHD) is a special case of LHD. Orthogonal Latin hypercube design (OLHD) was proposed by Ye (1997) which maintains orthogonality among the columns. These designs are "space filling" and they also maintain the

orthogonality between design sites. The orthogonality ensures that the quadratic and interaction effects are uncorrelated with estimates of linear effects. Interested readers are referred to (Ye, 1997) for an algorithmic construction of OLHD. Ye has stressed that orthogonality of these designs are independent of numerical values of *levels*. Figure 8 shows three different design matrix generated using this algorithm. These designs can be optimized for certain criteria e.g. minimum entropy, maximin criteria.

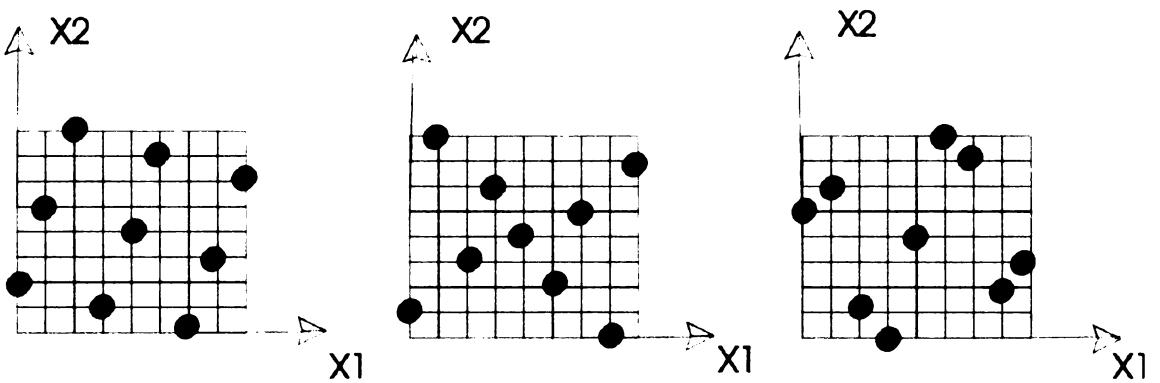


Figure 8: Orthogonal Latin hypercube design for a *factor* of 2

In the present work we are seeking design of "space filling" nature which will facilitate to capture the behavior of functions in global sense throughout the feasible design domain. We want to predict the response of the model in the entire feasible design space after we sample some intelligent design sites. The selection of initial design should allow us to explore a wide class of metamodels (e.g. least square polynomial metamodel, kriging metamodel, response surface metamodel and spline metamodel). We have used OLHD as the design criteria in the present work. The code for generating OLHD is freely available. In this work, I used one such code available at STATLIB (http://lib.stat.cmu.edu). It is worth mentioning at this point that we have tested our framework with some user

supplied design sites in combination with OLHD. This provides us some known design sites (at which we have some information about the function) on top of OLHD designs.

2.3 Metamodel Selection

We want to predict the response of the model in the entire feasible design space after we sample some intelligent design sites. Once the appropriate sample data has been obtained, we need to build the metamodel. This is described next. A metamodel associated with a model (or model's response) is defined as an *approximation* to the model itself (or its response). This section describes various techniques which can be used to construct a metamodel for expensive functions. The term surrogate (or metamodel) is used to denote any replacement for an expensive simulation. Few examples are least square polynomials, wavelets, radial basis, neural network, fuzzy logic, response surface methodology and kriging used to build a surrogate for function response.

The first class of surrogates will be called *models;* a terminology motivated by the use of model in, say, the models of crashworthiness analysis of varying physical fidelity (linear, full nonlinear model). These surrogates are based on modeling a response f at a *single design, x*. This modeling can be based on varying physical fidelity, as in the models of crashworthiness analysis just mentioned, or varying mathematical fidelity, as might be determined by mesh spacing or the order of accuracy of numerical schemes.

The second class of approximation can be a surrogate constructed from samples $f(x_1)$, _ _ _ , $f(x_n)$ of the original function $f(x)$ at different design sites x_i (and possibly values of the derivatives of f). A Surrogate is constructed from regression against this data (as in classical response surface methodology), interpolation of the data (as in spline fitting), or by a combination of regression and interpolation (as in kriging).

As discussed in chapter one, we can construct approximation of the original problem in two ways.

1) A physical approximate model (S_p), which closely represents the behavior of the true function.
2) An approximate modeling of the characteristic behavior obtained by some data fitting technique based on sample design sites chosen by some mean (S_k).

Questions that arise in the selection and use of surrogates in design includes the following:

1) Construction methodology of appropriate surrogates;
2) Validation of surrogates and estimation of the surrogate error;
3) Utility of surrogates in the multidisciplinary setting; and
4) Use of surrogates in optimization.

2.3.1 Least Square Polynomials

Polynomials are very popular methods known as data fitting and metamodeling techniques. They are simple and easy to compute. For a given design set (x_i, y_i), where, i = 1...n, its easy to fit a polynomial e.g. a quadratic curve. A second order polynomial can be represented by

$$f(x) = \beta_0 + \beta_1 x + \beta_2 x^2,$$

where, βs are regression coefficients, x is the design variable and $f(x)$ is function predicted value at x. $\beta' s$ are computed by solving an optimization problem by taking the mean of the sum of squared errors (MSSE) as the objective function at predicted values, x_i

$$\text{MSSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - f(x_i))^2 \qquad (2.1)$$

where, y_i are known values, $f(x_i)$ are predicted values of function and n is total number of points at which predicted values are compared with known values.

Once the optimal values of coefficients of polynomial are determined, it is easy to predict output values corresponding to any input x. This technique of constructing the surrogate

is very common and is termed as least square polynomial technique. It can be performed on polynomial of arbitrary degree with little difficulty, simply by changing the number of coefficients. Extension to design space of higher dimensions can also be done, however with less reliability.

Analysis of variance can be performed on this functional relation to determine the relative importance of each coefficient in the polynomial or the standard deviation of the result from the known results. Usually the model is changed heuristically, by adding or subtracting some terms so that the polynomial represents a satisfactory behavior in the entire design domain.

As we increase the dimension of the problem at hand or the size of the dataset, there are some obvious disadvantages of using this metamodeling technique. Coefficient evaluation takes longer time in higher dimensions as well as when size of the dataset is increased. Usually, the behavior of the function is not captured by one polynomial in the entire feasible domain. A quadratic polynomial can represent the global nature of function, but it cannot represent non-linearities in local regions. A higher order polynomial e.g. 19^{th} order polynomial will require twenty coefficients to be determined by least squares. Higher order polynomials tend to be highly oscillating between points and there in not enough smoothness in the function. In the present application, we are dealing with highly nonlinear and 'expensive' functions and for these reasons, we will not select least square polynomials as our metamodel.

2.3.2 Interpolating Splines

The drawbacks of polynomials can be avoided to a great extent by using interpolating splines as a metamodel. Splines are polynomials defined in a piecewise manner. We can benefit from the advantages of using polynomials while avoiding the drawbacks associated with polynomials. This could be especially useful for highly nonlinear nature of data. Whereas polynomials try to capture the behavior of the function in global sense with one functional representation, splines can model the behavior in each separate range of data (or piece defined by points or knots). The boundary conditions can be imposed on piecewise polynomials to ensure that these pieces match exactly with a prescribed degree of continuity. Usual practice is to model cubic splines as one piece with C^2 continuity imposed between pieces. This means that the pieces will have same function value and slope at boundary knots. This ensures the correct behavior of the function at knots (as compared to least square polynomials), but it does not tell us much about the values in between knots. They can be more oscillating or wavy.

A similar technique known as smoothing splines is an improvement over interpolating splines. Many researchers have worked on this method ([13], [32], [11]) and have proposed to adjust a weighting factor known as smoothing parameter. By using this parameter, a trade-off is sought between smoothness of polynomials of least square polynomials and the point-wise accuracy of piecewise polynomials in interpolating splines. Interested readers can refer to ([27], [32]).

2.3.3 Wavelets and Radial Basis

Radial basis and wavelets have earned popularity in recent years in several areas such reconstruction and image filtering. The characteristics of these metamodel is a *basis function* depending on the Euclidean distance between the sampled data points and the point to be predicted. They are similar to Fourier transforms, but wavelet transforms are capable of capturing a large amount of information with the help of small number of basis functions. Wavelet transforms are fast and require less number of coefficients to represent the metamodel. Diaz has used wavelet transforms to identify the systematic component of a signal characterizing the structural behavior from its random component ([22]).

Radial basis functions (RBF) have been developed for largely scattered multivariate data interpolation. A RBF method uses linear combinations of a radially symmetric function based on Euclidean distance (or other such metric) to approximate response functions. A radial basis function model can be expressed as:

$$f(x) = \sum a_i \|x - x_i\| \qquad (2.2)$$

where the sum is performed over the observed set of response $\{(x_i, y_i)\}$ and $\|\bullet\|$ represents the Euclidean norms. The coefficients a_i are found simply by replacing the left hand side of (2.2) with $g(\mathbf{x}_i)$, $i = 1, ..., n$, and solving the resulting linear system. Radial

basis function approximations have been shown to produce good fits to arbitrary contours of both deterministic and stochastic response functions (Powell, 1987).

2.3.4 Neural Network Metamodels

Neural networks can be described as flexible parallel computing devices for producing response that are complex functions of multivariate input design variables. They are capable of approximating arbitrary smooth functions and can be fitted using noisy response values. Neural networks are networks of numerical processors whose inputs and outputs are linked according to specific topologies. Interested readers can refer to Lippman (1987) or Masson and Wang (1990) for an introduction to neural networks. Networks used for function approximation are typically multi-layer feedforward networks. Feedforward layered networks have the flexibility to approximate arbitrary smooth functions very well, provided sufficient nodes and layers. This follows from the work of Kolmogorov (1961), whose results imply that any continuous function f: R" -> R can be exactly reproduced over a compact subset by a three-layer feedforward network. While there are some approximation schemes using three-layers, most approximations use a two-layer network structure, with a single output node for models having a univariate dependent variable. The overall metamodel is then a linear combination of linear or nonlinear functions of the design vector x. Strictly speaking, neural networks are assumed to use functions that are threshold functions. It is useful however, to allow more general functions and to think of neural networks as a technique for computing

metamodel coefficients and predicted values rather than as representing a particular class of modeling techniques.

2.3.5 Response Surface Methodology

A Response surface methodology (RSM) is a sequential process for data fitting. Least square polynomials, wavelets, radial basis and kriging are classified as global metamodels and a single surrogate maps the entire design domain. A RSM works sequentially, constructing a surrogate model in a local sense. This method is widely known as a tool to develop, improve and optimize a product or process. It has found its application in design, development and formulation of new products and improvement of existing product designs. Interested readers can refer to ([31], [19], [21], [16]).

A RSM is a sequential process which starts with one initial design point. Design of Experiments (DOE) is used to generate and sample a set of design sites in a small space around the initial design point. A linear least square model is constructed to fit the design points around initial design. The accuracy of the polynomial is verified using analysis of variance. Now, at this point using basic calculus, a new area of interest is identified in the design domain. The process is repeated until a linear fit of data no longer approximates the function. Since a RSM works sequentially in order to approximate the function and generate the metamodel, it is different from other methods. Our future work will deal with this type of methodology.

2.3.6 Kriging Metamodel

In all of the metamodeling techniques discussed, the fundamental assumption used to build the approximation is as follows: given a vector of independent factors **x** and response y, the relationship between y and **x** is:

$$y = f(x) + \varepsilon, \tag{2.3}$$

where ε represents a random error which is assumed to be independent and normally distributed with mean zero and standard deviation σ. The key concept in kriging metamodel constitutes that the error in the predicted values ε_i are not independent. Instead, errors are a systematic function of **x**. Since the true response surface function is usually unknown, a response surface f(**x**) is created to approximate it. The kriging metamodel take the form y(**x**) = f(**x**) + Z(**x**), where f(**x**) represents a polynomial and Z(**x**) represents a departure from the polynomial. Figure 9 illustrates how kriging error Z(**x**) is distributed as a function of **x** (its not assumed to be constant). A quadratic equation is fit via least squares to a given sample of design points. Figure 9 illustrates the underlying assumption that the distribution of error is a function of x.

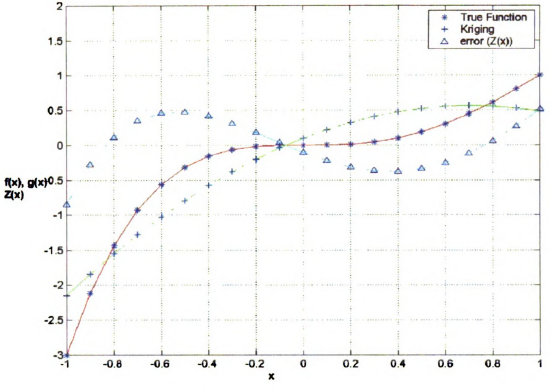


Figure 9: An illustration of error function Z(x) as a function of x

From figure 9 it can be observed that if the predicted value $f(\mathbf{x}_i)$ is far from the true value $y(\mathbf{x}_i)$, then a point very close to $\mathbf{x}_i$ will also have a predicted value $f(\mathbf{x}_i + \delta)$ that is far from the true value $y(\mathbf{x}_i + \delta)$. The underlying reason for such a behavior can be accounted to a systematic error associated with functional form of $f(\mathbf{x})$ which is the least squares fit.

Kriging metamodels assume that the deterministic response is a realization of a random function $y(\mathbf{x})$ that includes a regression model [15],

$$y(x) = f(x) + Z(x) \tag{2.4}$$

where, Z(**x**) (the error) is realized as a Gaussian process which represents uncertainty about the mean of y(**x**); Z(x) is assumed to have mean zero and covariance, V given by

$$V(w,x) = \sigma^2 R(w,x), \tag{2.5}$$

between Z(x) and Z(w) where σ^2 is the process variance and R(w,x) is the choice of the spatial correlation function between point w and x which actually determines how the metamodel fits the data. There are several choices of R(w, x) in the literature which can determine how quickly and how smoothly the function moves from point x to point w. One of the more common spatial correlation functions (SCFs) used in kriging models in a one-dimensional problem is

$$R(w^i - x^i) = e^{-\theta|w^i - x^i|^2}, \tag{2.6}$$

where θ>0 and the superscript refers to any design point, x. It can be noted from the above equation that as |w-x| increases, function goes to zero irrespective of any SCF used. As the distance between the points to be predicted and sampled point increases, the effect of sample point gets weaker on the predicted point. Also, $|\theta|$ dictates how fast this effect will take place.

The SCFs for multivariate case can be extended easily using "product correlation" which is described as multiplication of the correlation functions in several dimensions [15].

$$R(w-x)=\prod_{j=1}^{k} R_j(w_j-x_j) \tag{2.7}$$

where a subscript denote the dimension of the problem. A Kriging metamodel also allows to choose different SCF in different directions (i.e. different R_j) by choosing a different θ_j in different direction.

If a constant polynomial denoted by $\hat{\beta}$ is used for f(**x**), then the predicted values $\hat{y}(\mathbf{x})$ can be written as follows ([17]).

$$\hat{y}(x)=\hat{\beta}+r^T(x)R^{-1}(y-\mu J), \tag{2.8}$$

where **y** is the n x 1 column vector of observed response, where, **J** is a n x 1 column unity vector, **R** is the n x n symmetric matrix of correlations among the design points with the ones along the diagonal, and $\mathbf{r}=\{\mathbf{R}(x\text{-}x_i)\}$ is the n x 1 vector of correlations between the point of interest x and the sampled design points [17].

Mitchell and Morris and Sacks et al., reported that in the kriging metamode,l the only parameters dictating the polynomial f(**x**) are β_i, and θ_j. They are associated with the stochastic process Z(**x**). They suggested that the regression model f(**x**) does not greatly influence the metamodel fit ([17] , [15]). The distinct advantage of using kriging metamodel is from the fact that there is no longer any need to determine a specific functional form for kriging. Usually a constant $\hat{\beta}$ is used for f(**x**) even though a linear or quadratic functional form can be used. We can only use the correlation function parameter θ_j to describe a metamodel which can fit a given set of data points.

The estimate for $\hat{\beta}$ in the evaluation function (Eqn 2.8) is given by ([17])

$$\hat{\beta} = (J^T R^{-1} J)^{-1} J^T R^{-1} y \qquad (2.9)$$

The estimated variance from the underlying global model (as opposed to the variance in sampled data) is given by [33].

$$\hat{\sigma}^2 = [(y - J\hat{\beta})^T R^{-1} (y - J\hat{\beta})]/n \qquad (2.10)$$

Since variance (σ^2) and covariance matrix **R** are functions of θ_j, usually θ_j is found by maximizing a Maximum Likelihood Estimate (MLE) [17, 33].

$$MLE \equiv (-[nx \ln(\hat{\sigma}) + \ln|R|])/2 \tag{2.11}$$

MLE is computationally expensive and requires some knowledge of the distribution of noise in the data. There are hence other methods used for this purpose. Ordinary Cross Validation (OCV) and Generalized Cross Validation (GCV) are common examples. Interested readers can refer to ([17, 33, 15]).

2.4 Building and Validating a Kriging Model

Usually residual plots and square of residual (R^2) can be used to validate a metamodel but these methods are not suitable for kriging because there are no residuals in this case. Validating a kriging model using additional data points can be done, if possible. If additional points can be afforded then maximum absolute error, average absolute error and root Mean Square Error (MSE) can be used to validate the error with predicted values at additional points. These measures can be summarized as follows.

$$\text{Maximum abs error} \equiv \max\left|y_i - \hat{y}_i\right|; i = 1,...n_{error} \tag{2.12}$$

$$\text{Average abs error} \equiv \frac{1}{n_{error}} \sum_{i=1}^{n_{error}} |y_i - \hat{y}_i| \tag{2.13}$$

$$\text{MSE} \equiv \sqrt{\frac{\sum_{i=1}^{n_{error}} (y_i - \hat{y}_i)^2}{n_{error}}} \tag{2.14}$$

However, sometimes taking additional validation points is not possible due to constraints on expenditure of experiments. Thus an alternative approach proposed by Mitchell and Morris (1992) is leave-one-out cross validation approach. In this approach, each sample point used to fit the model is removed one at a time and, the model is rebuilt without that sample point (with same MLE) and the difference between the model without the sample and with the sample is computed at all the sample points. The formula for Cross Validation Square Error (CVRMSE) is

$$\text{CVRMSE} \equiv \sqrt{\frac{\sum_{i=1}^{n_s} (y_i - \hat{y}_i)^2}{n_s}} \tag{2.15}$$

2.5 Metamodel Comparisons

Some choices have to be made as to which type of metamodel is best suited for the simulation to be analyzed. Simpson provides some guidelines for appropriate choices for particular cases [6]. Another particularly useful guidance by Barton provides a list of seven very general criteria that aide in assessing any particular metamodel's merit [6].

- Ability to gain insight from the form of the metamodel.
- Ability to capture the shape of arbitrary smooth functions based on observed values which may be perturbed by stochastic components with general distribution.
- Ability to characterize the accuracy of the fit through confidence intervals, etc.
- Robustness of the prediction away from observed (**x**,y) pairs.
- Ease of computation of the approximate function f.
- Numerical stability of the computations, and consequent robustness of predictions to small changes in the parameters depending on function f.
- Does software exist for computing the metamodel, characterizing its fit, and using it for prediction?

2.6 Optimization

2.6.1 Background

Before we start our discussion of optimization techniques used in this work, we will revisit our selection criteria of our framework and nature of function we are trying to optimize (1.1)

Minimize f (x) (1.1)

Subject to $a \leq x \leq b$

where, f: $R^n \rightarrow R \cup \{\infty\}, a; b \in R^n$;

Our criteria for selection of a framework stemmed from these concerns.

- Cost of evaluating f (x) could be very expensive in terms of computational time and resources. Some evaluation of f (x) could be available from other sources. If x is infeasible f (x) may not be available.
- It is not necessary that f (x) will be evaluated at all the possible design sites. There could be cases when f (x) may not be evaluated accurately at some design sites and it incurs the same computational cost.
- Derivative information of f (x) is either not available or is not reliable to use.

We are concerned with evaluation of f(x) which is expensive and we do not have derivative information. It has been a long engineering practice to deal with such numerical optimization problem by replacing f(x) by a surrogate $\hat{f}(x)$. As described by Barthelemy and Haftka (1993) we can replace f(x) with an inexpensive surrogate $\hat{f}(x)$ and minimize $\hat{f}(x)$ instead. One such approach in the literature of DACE is to evaluate f(x) at V-1 (V is the total "budget" for function evaluation) carefully selected design sites and construct $\hat{f}(x)$ from the resulting evaluation. From here perform the numerical optimization to obtain minimum of $\hat{f}(x)$ and evaluate f(x) at the candidate minimizer thus obtained. In 1995 Frank suggested that "minimalist approach" of minimizing a

single $\hat{f}$ (x) is not likely to yield satisfactory results, and proposed several *sequential modeling techniques* as alternatives.

While discussing initial design issues, we have seen how to select design sites for design of experiments. We have also discussed various techniques on building the surrogate models from a set of evaluated f(x). In this section, we will explore some numerical optimization techniques used for such class of problems. It will be worth mentioning again at this point that the surrogate may not have very reliable derivative information in which case we are looking at direct search optimization techniques. In future, plan is to work with derivative based optimization techniques once we have useful information about the gradients of surrogates.

2.6.2 Pattern Search Methods

Consider a constrained optimization problem of type

$$\text{Minimize } f(x)$$
$$x \in \Omega$$

where, R is a set of real numbers, f: $R^n \rightarrow R$ is the objective function. Ω is the feasible domain of x.

Pattern search algorithms are a subclass of direct search methods for numerical optimization. We are looking for optimization techniques which do not require derivative

information and direct search methods fall in this category. They do not use any explicit derivative information. Interested readers can refer to [18], [3], [4], [36] for an explanation of these methods.

Let x_k be a point and $f(x_k)$ be the function value in k^{th} iteration. A pattern search technique should take three steps to get new point x_{k+1}. Firstly, at each iteration the next iterate is selected from a set of points which is determined by a *pattern* (described next). No explicit restriction is placed on $\|x_{k+1} - x_k\|$. Secondly, only the function values f(x) are used, not the derivatives, to get the next iterate. Thirdly, each iterate must satisfy a *simple decrease* condition. There is no *sufficient decrease* condition.

If $x_{k+1} \neq x_k$ then $f(x_{k+1}) < f(x_k)$.

A *pattern* is a collection of steps and each step can be added to current iterate to get next trial iterate. The orientation and scaling of the *pattern* can be changed as the algorithm proceeds. In another way a *pattern* can be thought of as a vector with a direction and magnitude from the current iterate. A *pattern* can be represented in several ways ([24], [35]).

We will use generalized pattern search method in our approach. Two features, a sequence of *meshes* and a list of *polling conditions* characterize this method. Mesh is a *pattern* (also called lattice) to which search is confined in a particular iteration. As optimization proceeds, the polling conditions (also called search conditions) dictate the change in the

current *pattern* thus qualifying the algorithm for convergence. The *pattern* keeps changing in each iteration.

To ensure convergence of pattern search algorithm, the primary condition enforced in search/polling technique is as follows. The set of vectors formed by taking the differences between the set of *trial points* at which objective function is to be evaluated (The *pattern*) and the current iteration x_k ($\|x_{k+1} - x_k\|$) must contain a positive basis for R^n. A positive basis is a set of vectors whose positive linear combination spans R^n, but for which no proper subset has that property. This is the basis of convergence for pattern search algorithm. Pattern search method can be outlined as follows [18]:

1. Construct or update the *pattern* of points around the current iterate x_k.
2. Evaluate certain points in the *pattern* and search the *pattern* to find a point that reduces the objective function.
3. Change the *pattern* if required, depending on whether the search point produces a decrease in the objective function.
4. Iterate until convergence criterion is met.

The following flowchart in Figure 10 describes the Generalized Pattern Search (GPS) algorithm, which is the basis of our Surrogate Management Framework (described next).

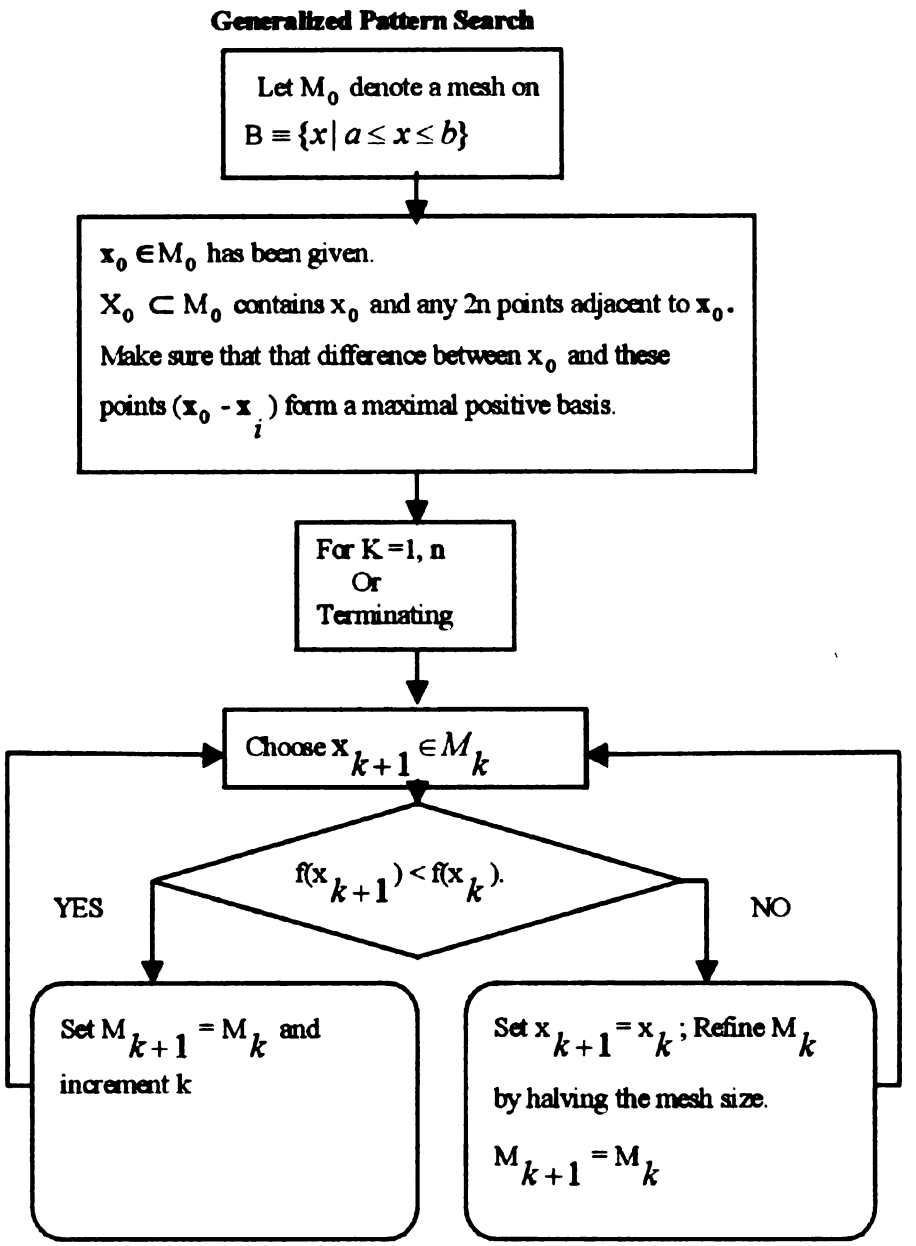


Figure 10: Generalized Pattern Search Method

In literature it is allowed to choose any set of trial points in M_0 at which the function is to be evaluated but the choice of initial design site will affect the result and efficiency of the simulation

2.7 Surrogate Management Framework

The basic idea behind Surrogate Management Framework (SMF) is built on GPS algorithm. There is one intermediate Evaluate/Calibrate step which allows a sequence of surrogate approximation as the algorithm proceeds (proposed by Frank 1995). We have a family of approximation algorithms, which can be used for surrogate metamodel creation (e.g. kriging, response surface, splines, polynomials) and update. Convergence of SMF comes directly from the convergence of GPS. The steps for our Surrogate Management Framework are outlined in the Figure 11.

The key to success of SMF is to define the search strategy that efficiently exploits the current surrogate, S_k. Also, notice that f is not evaluated at all the points in T_k before declaring a successful search. Any point identified with objective less than $f(x_k)$, it is a success. Except for evaluate/calibrate step in SMF, this is identical to GPS.

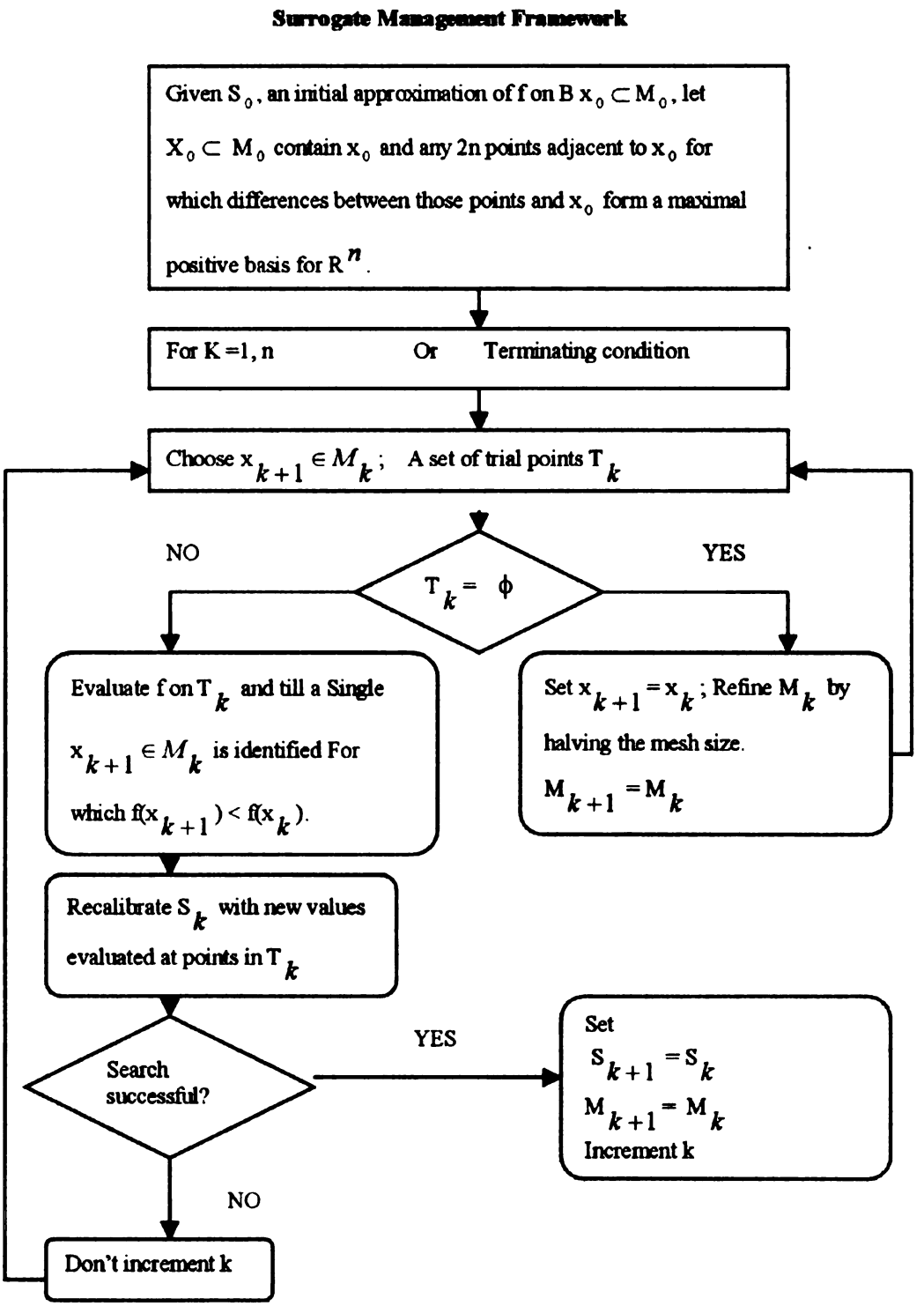


Figure 11: Surrogate Management Framework

The basic strategy for our framework now can be outlined in three steps

1) Choose an initial mesh over the feasible region [a, b] that signifies the desired degree of resolution. The resolution can be refined if needed and an initial baseline design $x_c \in$ [a, b] at which f is known
2) Perform an initial computer experiment to select N initial design sites, evaluate the true objective function f at the initial design sites, and construct an initial surrogate S of f from samples $f(x_1)$, _ _ _ , $f(x_n)$ of the original function $f(x)$ at different design sites x_i.
3) DO until a minimizer f is found (for the current resolution of the grid) or until the "budget" (V) is exhausted

- Find a candidate x_t that minimizes S on the grid and treats x_t as a site at which S predicts a minimizer for f on the grid.
- Evaluate $f(x_t)$.
- Update the approximation S to include the value of objective function $f(x_t)$.
- If $f(x_t) < f(x_c)$, then $x_c = x_t$ else leave x_c unchanged.
- Repeat step 3

An alternate strategy with some modification to basic SMF strategy is also proposed in literature ([34]). We notice that two important aspects affect the global optimization. The effectiveness with which we find the minimizer of S and how accurately we construct the

surrogate, *S*. In view of these two competing objectives, an alternate strategy which actually modifies the surrogate *S* and the next design point proposed by *S*. We hope that we will have better idea of the objective function in global sense in initial stages of iteration. In literature ([34,18]) a merit function (m_c) is formed as follows.

$$m_c(x) = s_c(x) - \rho_c d_c(x), \tag{2.16}$$

where, $\rho_c \geq 0$ and

$$d_c(x) = \min \|x - x_i\|$$

where, d_c is computed over all points x_i at which we know the value of the true objective function and ρ_c is a constant (we tried with ρ_c=2 and it worked very well in our case). Thus d_c(x) is the distance from x to the nearest previous design site. The merit function m_c(x) comprises two components, S_c and d_c. The approximation function S_c plays the same role as before: we want to minimize (or at least decrease) the value of S_c at x_c as a way of predicting decrease on f (x). The second function d_c is an experimental design criterion to ensure that the trial point is placed where information obtained from evaluating f will be useful in updating S_c. Many such criteria are possible in literature. We are using the criterion of maximum distance design which is an example of space filling design criteria.

2.8 An Example

We will use our framework for finding the minimum of the following problem.

Minimize $f(x) = e^{-2x}\sin(10\pi x)$, (2.17)

Subject to $x \in [0, 0.5]$ and number of function calls allowed (V) <10.

We assume that this function meets all the constraints (e.g. expensive function, no gradients available, may fail to evaluate at certain x). The program started with sampling 2 points (x_1, x_2) in the feasible design domain. The true function was evaluated at these two points and S_0 was built with the help of $f(x_1)$ and $f(x_2)$. Surrogate (S_0) built in the first iteration is a straight line (see figure 12). The program found a minimum of S_0 at x_3 (= 0.0). The original function was again evaluated at x_3 to get $f(x_3)$. At this point it is observed that *f(x) has failed to evaluate*. S_0 was updated to get a new surrogate S_1 and so on. Figures for all the iterations are shown in Figure 12(a) to Figure 12(c). The algorithm gives the minimum at x* = 0.1475 which turns out to be true minimum. It can be appreciated that the surrogate S keeps changing as the algorithm proceeds.

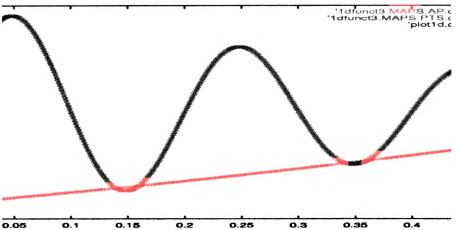


Figure 12(a): Iteration 1 Kriging with 2 base points

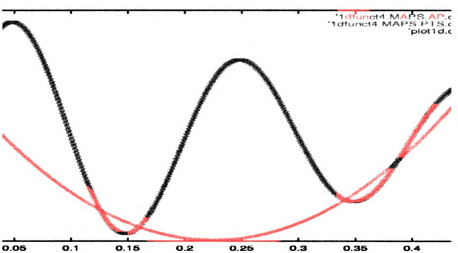


Figure 12(b) :Iteration 2 Kriging with 2 base points+ 1 opt points

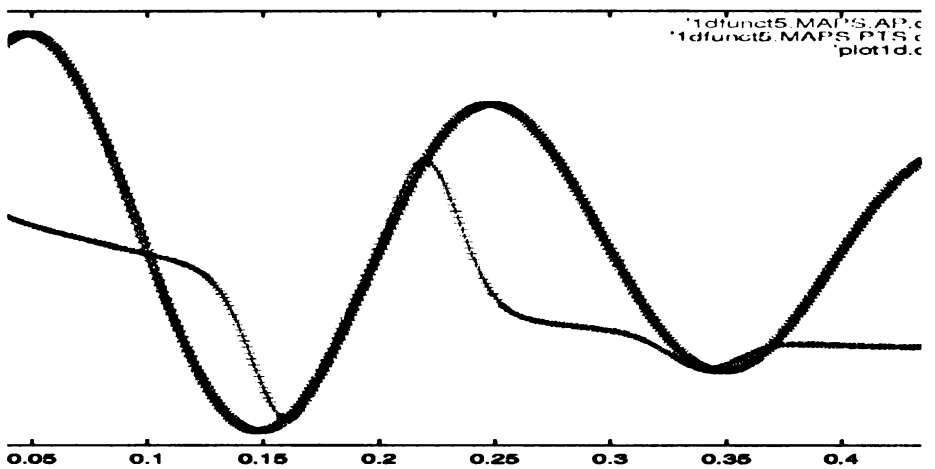


Figure 12(c): Iteration 3 Kriging with 2 base points+ 2 opt points

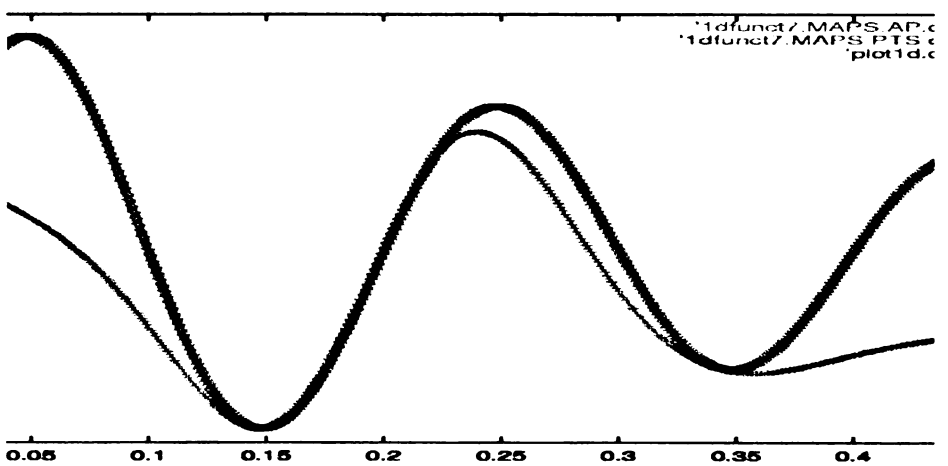


Figure 12(d) :Iteration 4 Kriging with 2 base points+ 3 opt points

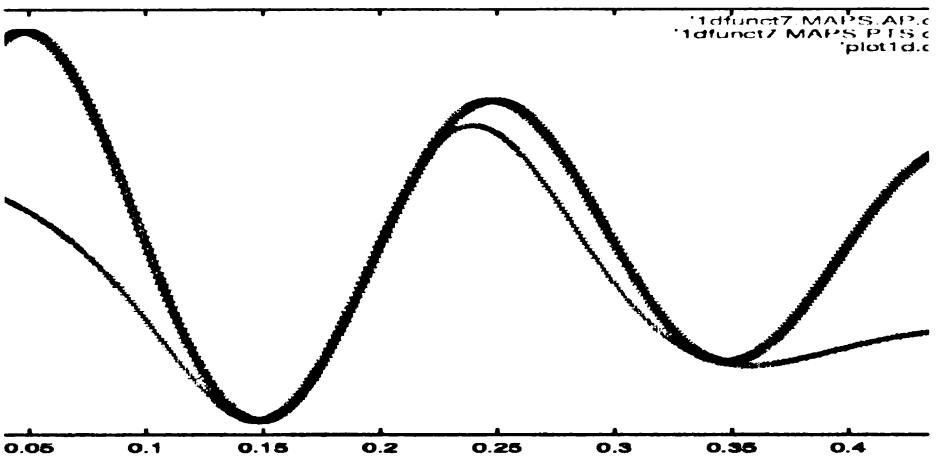


Figure 12(e): Iteration 5 Kriging with 2 base points+ 4 opt points

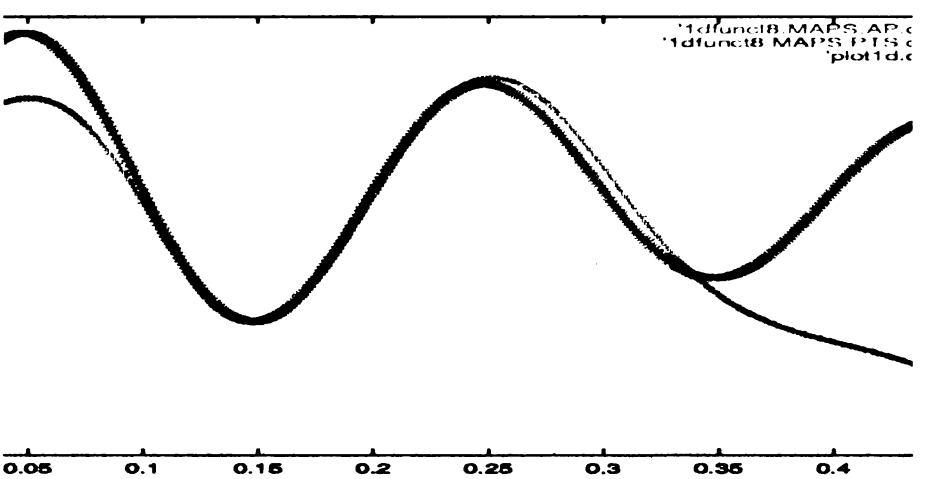


Figure 12(f): Iteration 6: Kriging with 2 base points+ 5 opt points

Chapter 3

Tuning Problem

3.1 Background

In chapter two, the framework developed for optimization of a non-linear and expensive function was described. Three main areas were investigated.

(1) Selection of initial design, e.g. space filling Latin hypercube designs for function evaluations to be used in construction of surrogate model.
(2) Metamodeling techniques e.g. kriging metamodel, to build the surrogate of expensive functions.
(3) Optimization techniques e.g. pattern search methods, to optimize the surrogate and hence to predict the optima of expensive functions.

In chapter one of this report, a *lattice* model was described that replicates major structural components of a complex structure (a vehicle or a component of a vehicle). In this chapter, a lattice model is used to represent the behavior of a complex structure (a square tube) in an impact event. The lattice model is described with dimensional details including the material properties of its components. The goal in the present work is to formulate an optimization problem, which can be used to optimize these parameters, in order to replicate the *relevant* response of the real structure (the square tube). The *relevant* response of the real structure could be a measure of crashworthiness analysis.

An optimization problem is formulated to *tune* the lattice model to capture the *relevant* response of the structure.

3.2 Lattice Model

Figure 1 (chapter one) shows a vehicle or a component in a vehicle (a *complex* structure) and a lattice model (a *simple* structure). The optimization problem can be formulated using material and/or geometric properties of each cell in the lattice as design variable. It is expected that the lattice mode will capture the response of the vehicle (complex structure). The objective in the optimization problem is to enhance the protection of the passenger. This can be realized, at least in part, by controlling the acceleration experienced by the passenger and controlling the deformation of the structure in the immediate vicinity of the passenger. Thus, for the lattice model to be useful, it should reproduce with sufficient accuracy, acceleration and deflections at points of interest in the real structure.

Performance measures such as acceleration in a crash event are notoriously poorly behaved functions. They are highly non-linear, non-smooth, and very sensitive to uncertainties in parameters. A typical acceleration pulse in a crash event has the form illustrated in Figure 13 ([22]). Along with relevant information, this signal also contains a significant, random component that is not of relevance to the design problem, and can even hide relevant behavior.

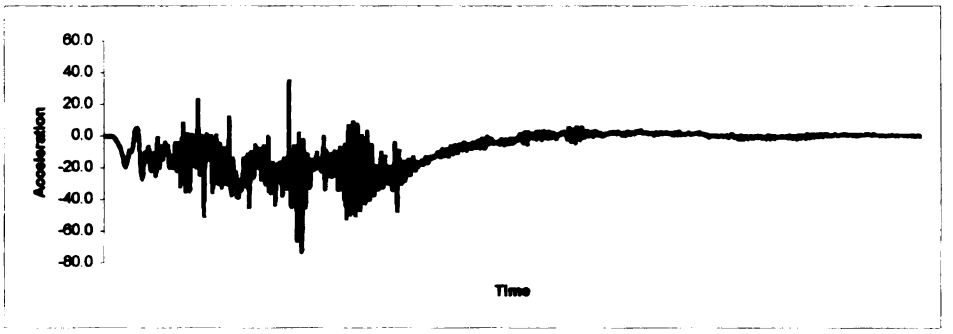


Figure 13: Typical acceleration signal ([22])

The lattice is used to model only *some* relevant aspect of the behavior of a real structure during an impact. It is unrealistic to expect that such simplified model can accurately reproduce *all* the complexities of the behavior of the complex structure. We make the claim that for the purposes of design, only a small number of signals are needed to evaluate *some* performance, e.g., accelerations or deformation of the structure at a given location or locations. The goal of optimization problem is then to "tune" the lattice to reproduce only these relevant features. This is done by setting up an optimization problem to *minimize the difference in relevant behavior between the lattice and the real structure*.

3.2.1 Characteristics of Lattice Model

The truss-lattice model considered here is an assembly of basic *cell* units formed by six bar finite elements (eight degrees of freedom per cell), in an arrangement shown in

Figure 14. Details of the construction of this model can be found in [20]. Characteristics of the model are outlined as:

1. The pins in the cell are frictionless and hence each bar in the cell is capable of transmitting loads only along its axis. The full geometric non-linear behavior is modeled.

2. The stiffness matrix associated with a bar finite element of (initial) area A and length L in a cell, includes geometric and material nonlinearities.

3. The material of each bar element can include a number of non-linear features and is shown in Figure 15.

4. The dynamic behavior of the complex structure is captured by adding (lumped) masses to each degree of freedom. The cell itself has no structural mass.

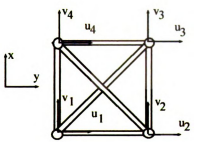


Figure 14: Basic cell in lattice model ([20])

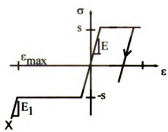


Figure 15: Material model for each bar in a lattice cell ([20])

3.2.2 Proposed Lattice Model

For simplicity, we consider the impact of a hollow tube with a square cross section against a rigid wall. The tube has dimension 100 x 50 x 50 mm (Figure 16). The structure impacts a rigid wall at a velocity V_0 and deforms plastically. It is assumed that the performance function of the structure can be measured by investigating the motion of a selected number of points P_1, P_2,.., P_r. One such point of interest (P_1) is shown in Figure 16.

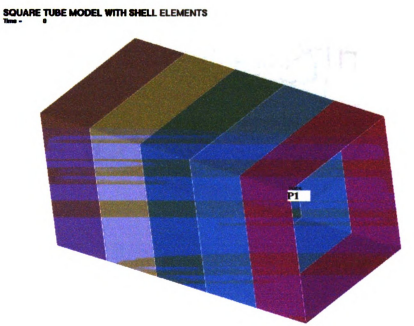


Figure 16: Real structure (A square cross section tube)

The tube is modeled in *Unigraphics* and meshed with shell elements in *Hypermesh*. The structure is grouped in to five different sections (can be distinguished by different colors in Figure 16 and Figure 17) and each section can have different properties e.g. different yield stress (S_y), different shell thickness. The tube is analyzed in an impact event using LS-DYNA3D. The performance of the structure is assumed to be characterized by the acceleration of point P_1 on the hollow tube. The performance function is computed for point P_1 as a weighted sum of RMS of acceleration and maximum absolute displacement at this point. Figure 17 shows the deformation of tube in an impact event.

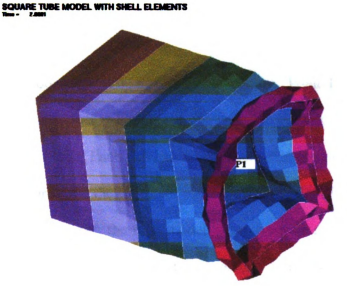


Figure 17: Deformed tube structure in an impact event

Rather than analyzing the structure itself, we represent it by a lattice model. The actual shape and density of the lattice is "somewhat" close to the real structure, as the goal is not

to reproduce the real structure but only the set of relevant *signals* measured at point P_1. One such possible lattice is shown in Figure 18.

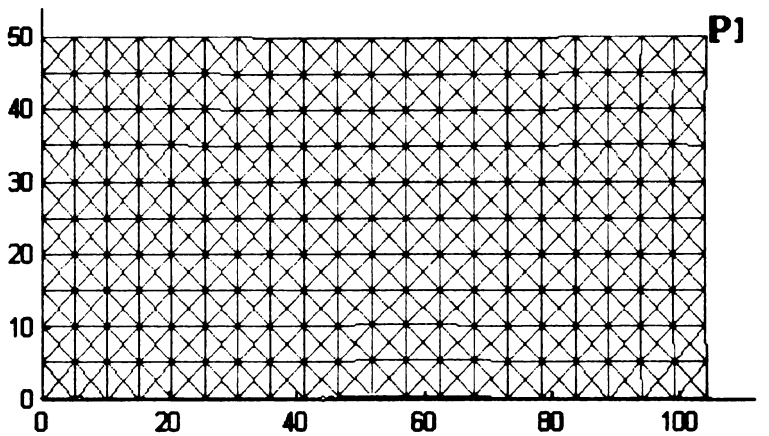


Figure 18: Lattice model of the real structure.

Proposed lattice model has same dimension as of tube. Point P_1 is reproduced on the lattice at the location shown in Figure 17. The lattice model has 20 columns and 10 rows resulting in total 200 cells. Each cell has six bar finite elements. All six elements in a cell have same (initial) area and they are kept same for a particular FE analysis. To match the lattice model as closely with the tube model, some possible grouping of cells has been done. Five consecutive columns are grouped together and they will have the same yield stress (S_y). This will divide the lattice in five different groups as in the case of tube. This also reduces the computational size of the problem. To reduce the size of the problem further, we assume that all cells at the same distance from the wall have the same area. This reduces the problem size to 20 design variables in area domain and five design variable problem in yield stress domain.

Cori ([22]) has implemented the lattice model (Code written in FORTRAN) and it was provided for the present work. This program can evaluate the lattice model for desired and *relevant* response for twenty given areas and five yield stresses for the lattice. This program does the grouping of cells as desired. Figure 19 shows the deformed lattice model after an impact event. Node 21 shown in the Figure 19 corresponds to point P_1 in square tube. The lattice model does not implement the contact algorithm.

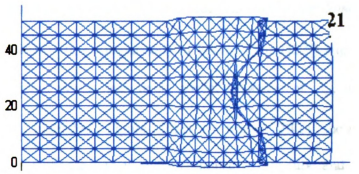


Figure 19: Deformed lattice model after an impact event.

3.3 An Optimization Problem

In section 3.2.2, we described the real structure (the tube) and the lattice model hoping that the lattice model will reproduce some relevant complexities of the response of the real structure. Before we formulate our optimization problem, the lattice model and the tube are compared for some responses (e.g. acceleration signals, displacement signals and

velocity signals) at point P_1 of the tube and corresponding point P_1 of the lattice to get some idea about the *feasible design domain* for lattice model. It is worth mentioning at this point that the tube has only yield stresses (S_y) as the *design parameters* (other parameters are constant) and lattice has S_y and 20 grouped areas (A) as the design *parameters*. We will compare the responses for some 'reasonable' set of yield stresses (we need 5 yield stresses for tube and for the lattice, S_{y0}) and some reasonable sets of areas (we need 20 areas for lattice, A_0). This will ensure that the lattice model is *capable* of reproducing response of the tube in the *feasible design domain* formed about S_{y0}, A_0 (to be decided by this comparison) of lattice. Diaz ([22]) suggested to experiment with this comparison, which was really useful to arrive at A_0, around which the feasible domain was extended. Figure 20(a) to Figure 20(c) shows acceleration response, displacement response and velocity response of the tube and the lattice for some yield stress values (S_{y0}= {0.097, 0.097, 0.097, 0.097, 0.097}) and Area (A_0 ={15.0, 15.0, 15.0, 15.0, 10.0, 10.0, 10.0, 10.0, 2.0, 2.0, 2.0, 2.0, 2.0, 2.0, 2.0, 2.0, 12.0, 12.0, 12.0}).

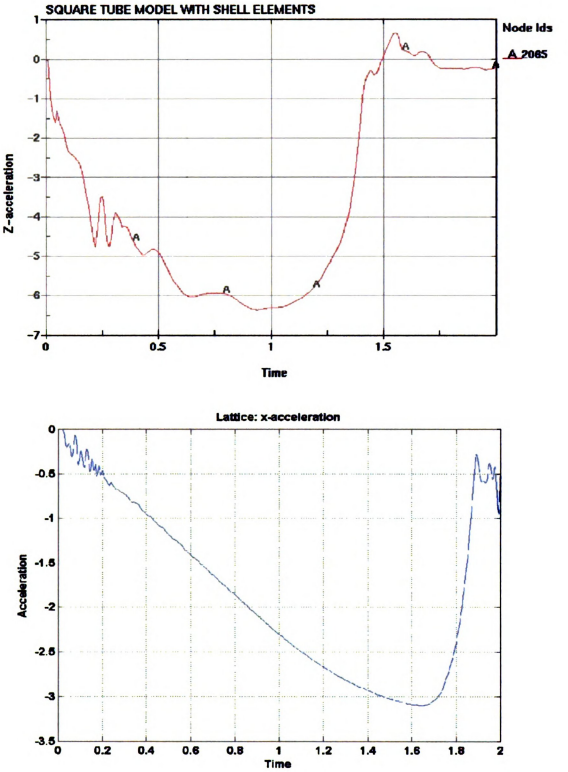


Figure 20(a): Comparison of acceleration response of the tube and the lattice.

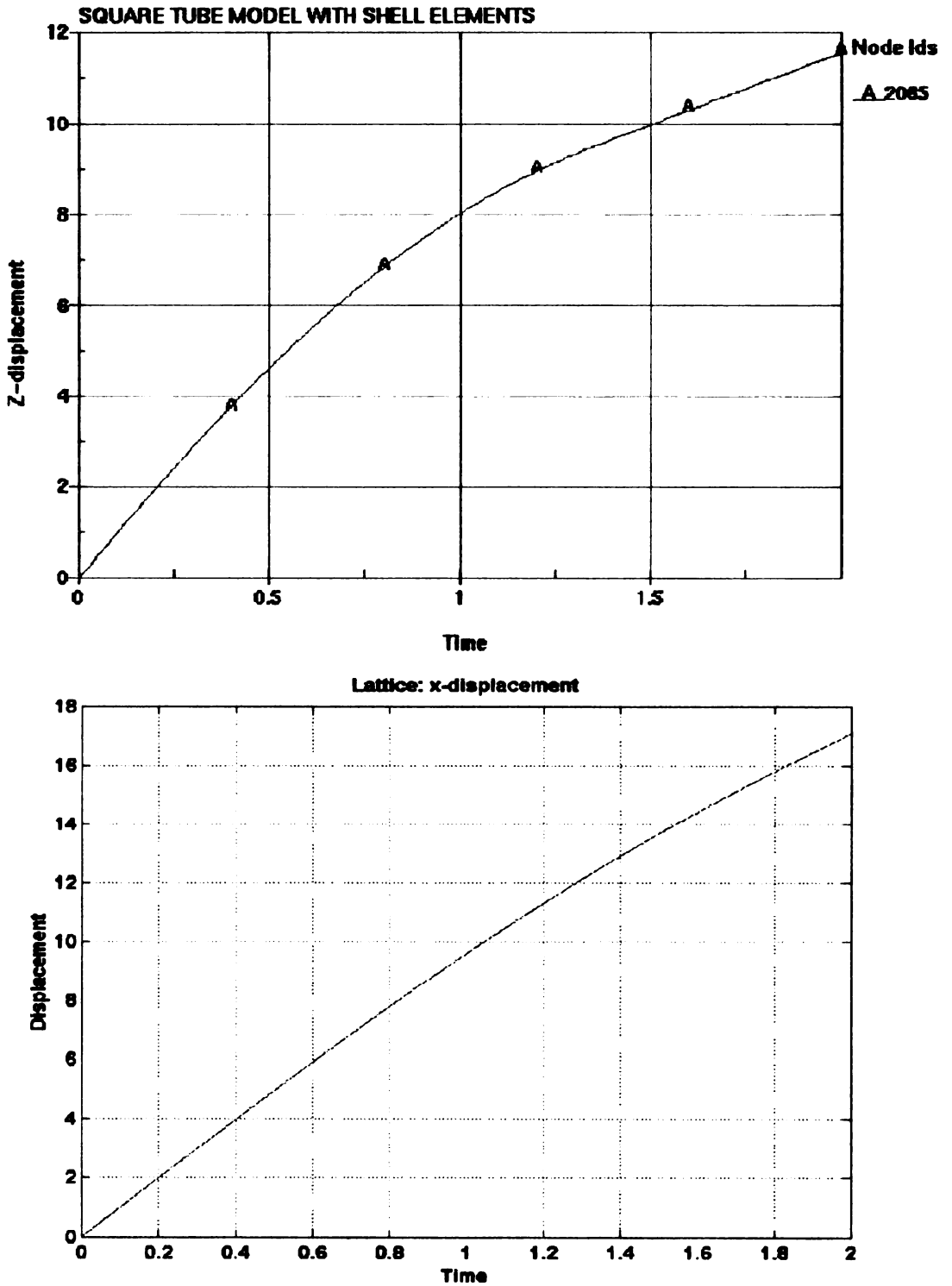


Figure 20(b): Comparison of displacement response of the tube and the lattice

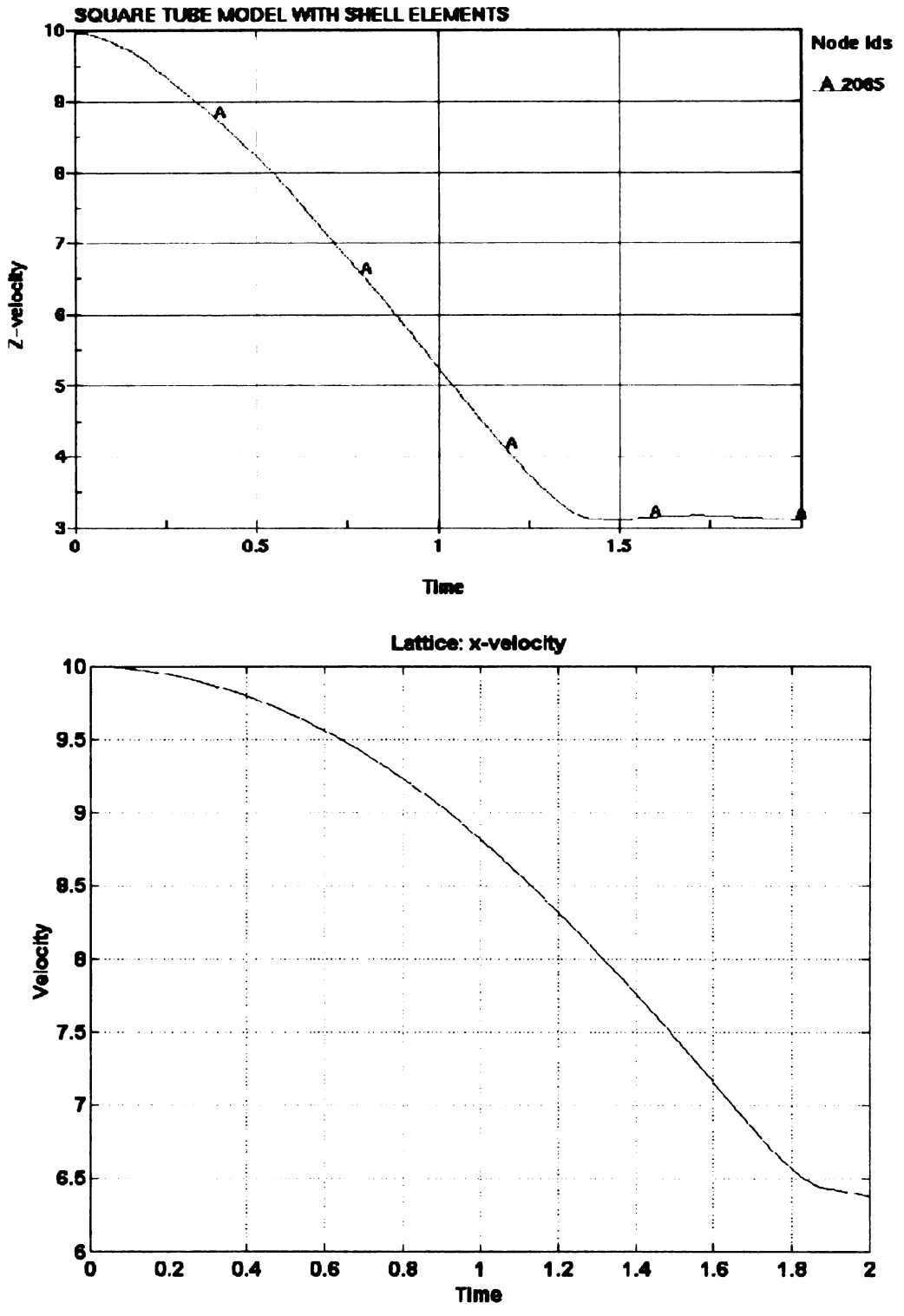


Figure 20(c): Comparison of velocity response of the tube and the lattice.

3.3.1 Response function formulation

The performance function (f) is computed for point P_1 as a weighted sum of RMS of acceleration and maximum absolute displacement at this point.

$$f = w_1 * \left(\sqrt{\frac{1}{n}\sum_{i=1}^{n}\left\|\ddot{u}_i^{\,p}\right\|^2}\right) + w_2 * \max\left(\left\|u_i^{\,p}\right\|\right) \tag{3.1}$$

where n is the total number of design points in the yield stress domain (S_y), $u_i^{\,p}$ is the displacement signal of point P_1 at S_{yi} point in S_y domain and $\ddot{u}_i^{\,p}$ is the acceleration signal of point P_1 at S_{yi} point in S_y domain. w_1 and w_2 are weights for RMS of acceleration and maximum displacement of point P_1 respectively and f is the response function. We used equal weight for RMS of acceleration and maximum displacement subject to $w_1 + w_2 = 1$.

There is an important distinction between the performance function of the tube (f_{Tube}) and the lattice model ($f_{Lattice}$). The response function of the tube is obtained in the yield stress domain (S_y) for a *fixed* set of shell thickness. The response function of the lattice is obtained in the yield stress domain (S_y) for a *fixed* set of areas. The goal is to *find an*

optimum set of twenty areas ($A_{optimum}$) that will give *same response of the lattice as of tube in the yield stress domain.*

3.3.2 Tuning Problem

The response function formulated in section 3.3.1 for the tube (f_{Tube}) was evaluated with the help of LS-DYNA3D at 25 yield stress points and it is shown in Table 1. f_{Tube} is plotted against 25 yield stress points in Figure 21 in MATLAB.

Table 1: f_{Tube} at 25 set of yield stress points

N	Sy1	Sy2	Sy3	Sy4	Sy5	f_{Tube}
1	0.015	0.034	0.292	0.203	0.03	9.10422
2	0.053	0.207	0.017	0.247	0.019	8.98162
3	0.223	0.167	0.131	0.097	0.258	9.09545
4	0.124	0.255	0.175	0.136	0.111	7.98922
5	0.265	0.285	0.185	0.014	0.056	9.41589
6	0.091	0.267	0.277	0.237	0.14	7.95098
7	0.028	0.112	0.077	0.108	0.071	8.224
8	0.079	0.114	0.221	0.171	0.136	8.03872
9	0.295	0.242	0.039	0.196	0.094	9.4277
10	0.01	0.018	0.204	0.059	0.268	9.34061
11	0.138	0.226	0.002	0.295	0.295	9.80016
12	0.154	0.181	0.198	0.066	0.004	9.56331
13	0.235	0.084	0.032	0.28	0.217	9.20342
14	0.131	0.002	0.162	0.225	0.192	8.86714
15	0.253	0.126	0.06	0.006	0.277	9.39228
16	0.179	0.222	0.122	0.15	0.174	8.36045
17	0.045	0.057	0.076	0.084	0.077	9.00829
18	0.064	0.163	0.243	0.251	0.125	8.04092
19	0.277	0.194	0.103	0.16	0.201	8.84209
20	0.244	0.092	0.274	0.182	0.233	8.81943
21	0.201	0.068	0.23	0.265	0.153	8.63996
22	0.198	0.143	0.262	0.039	0.047	9.48524
23	0.111	0.05	0.098	0.121	0.245	8.8231
24	0.169	0.291	0.144	0.033	0.181	9.47534
25	0.092	0.267	0.277	0.237	0.14	7.95086

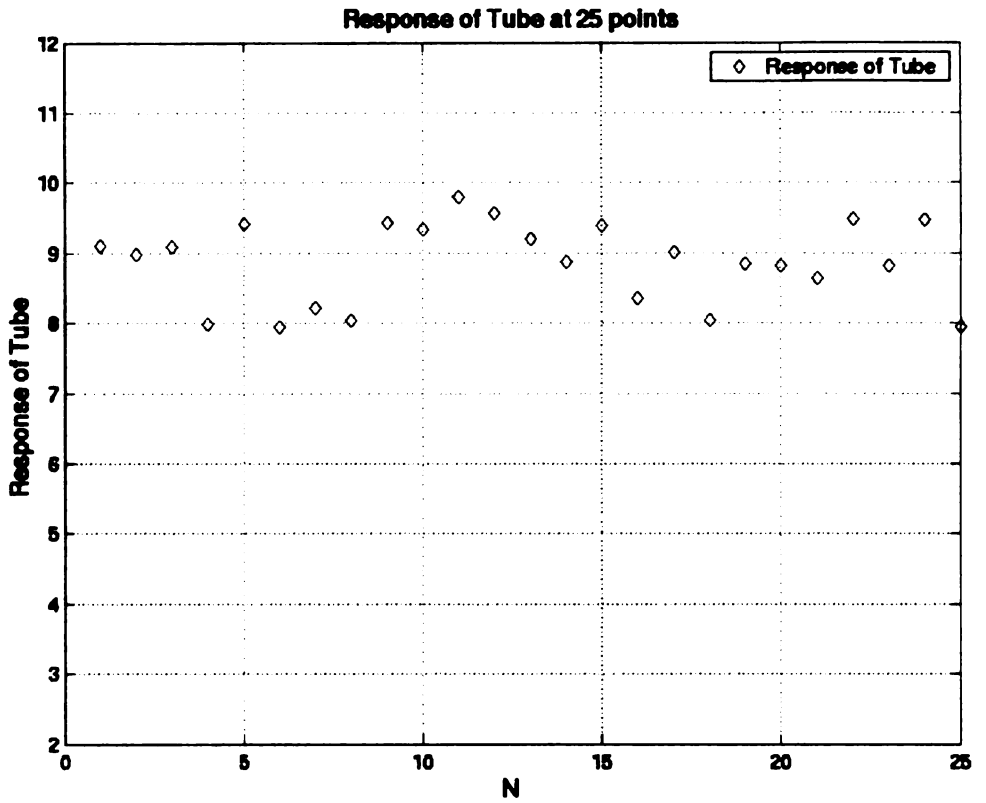


Figure 21: Response of tube (f_{Tube}) plotted against 25 points in S_y domain.

The lattice has twenty areas of the cells as the design variables as shown in Figure 22. Response function for lattice in S_y domain is plotted for some *fixed* values of areas of the lattice (20 areas). Two such responses are plotted corresponding to two different sets of areas in Figure 23(a) and Figure 23(b). The tuning problem will be formulated to find a set of areas ($\{A_1, A_2, ..., A_{20}\}$) which will make the lattice response same as the response of the tube.

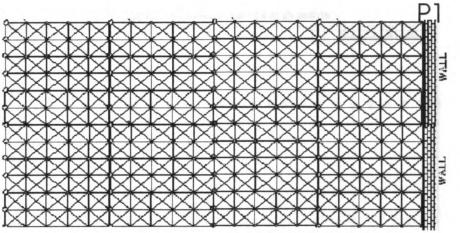
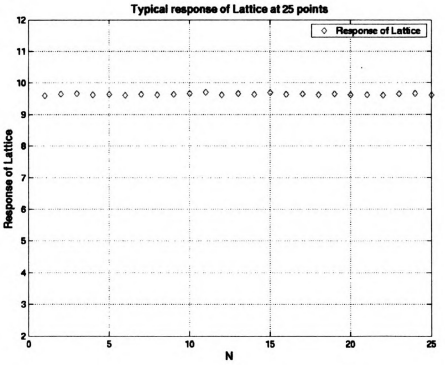


Figure 22: The lattice has twenty areas as design variables



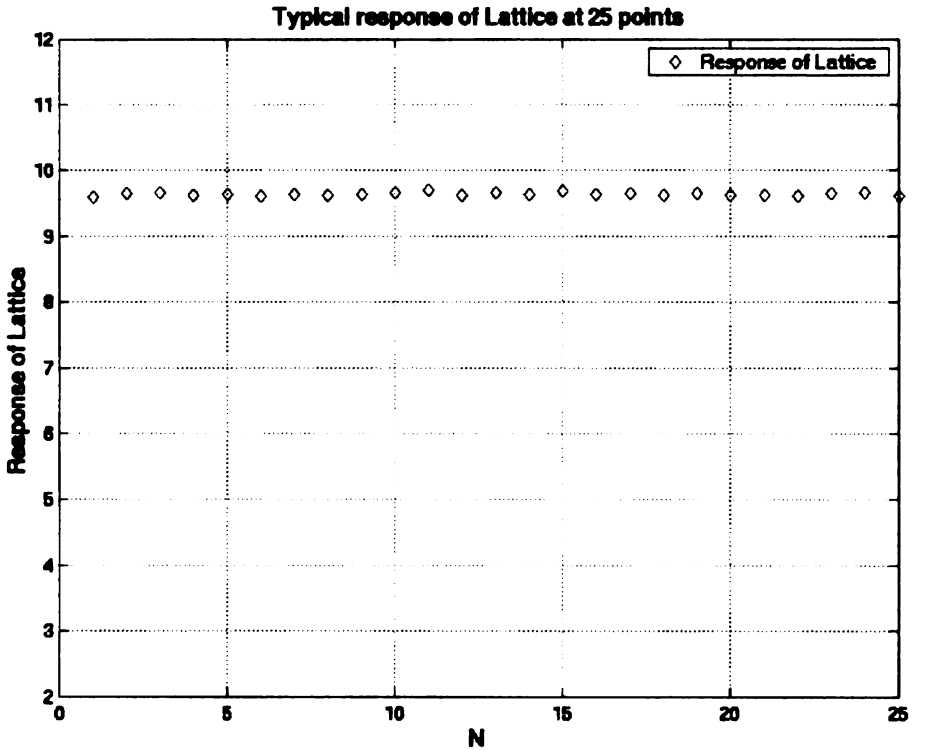


Figure 23: Response of lattice for 2 different sets of areas.

Lattice model will be evaluated for different set of areas ($\{A_1, A_2, ..., A_{20}\}$; a design point A^i is a set of twenty areas $\{A_1, A_2, ..., A_{20}\}$. The optimization problem is then:

Find $\{A_1, A_2, ..., A_{20}\}$ of the lattice that

$$minimize \left\| f_{Lattice} - f_{Tube} \right\|, \tag{3.2}$$

subject to $A_{min} < A_i \leq A_{max}$

V <25.

where, $\|\bullet\|$ denotes the Euclidean norm and V is Total *Budget for objective function evaluation*. Note that for one evaluation of objective function, lattice model will be called 25 times (we have a set of 25 yield stress points).

The optimization problem thus formulated is solved using the framework developed in chapter two. Optimization parameters chosen to solve the particular problem are listed below:

Total Budget [V]: 25

Initial design budget [N] (less than V): 10

Initial Mesh size [M_0] = 0.01; (denotes the resolution of the domain.)

A_{min} = {5, 5, 5, 5, 4, 4, 4, 4, 1, 1, 1, 1, 1, 1, 1, 1, 6, 6, 6, 6}.

A_{max} = {60, 60, 60, 60, 30, 30, 30, 30, 10, 10, 10, 10, 10, 10, 10, 10, 40, 40, 40, 40}.

A block-diagram in Figure 24 describes the procedure for solving tuning problem.

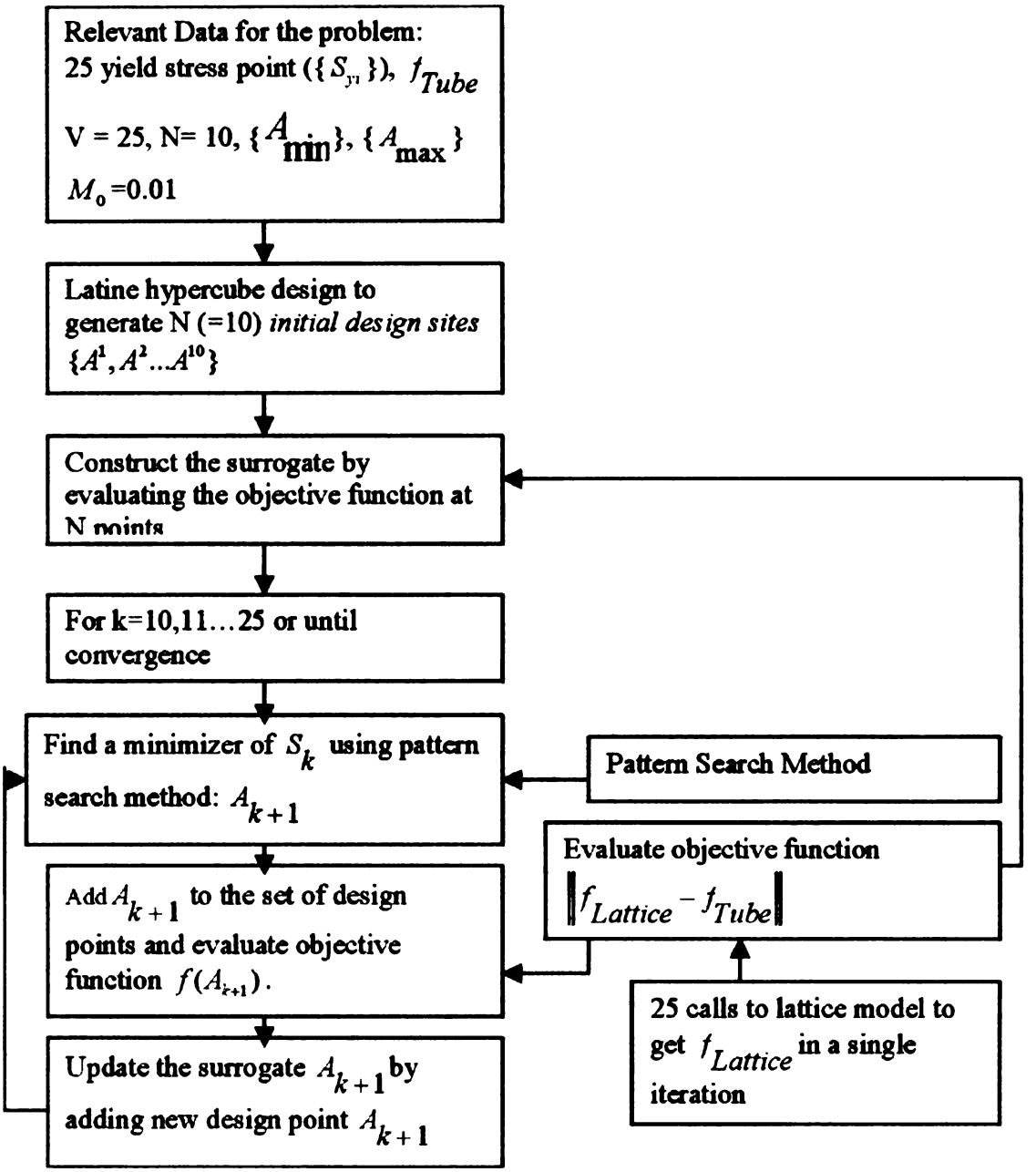


Figure 24: Block-diagram describing the procedure of tuning problem.

3.3.3 Results

The lattice (simple structure) response function was tuned to match with the response function of the tube (complex structure). Figure 25 shows the plots of response of tube

and lattice structure. The tuning procedure proposed the optimum areas of cells of the lattice as $A_{optimum}$ = {8.8009, 8.8581, 8.9932, 8.9468, 8.8588, 8.8739, 8.8047, 8.9180, 8.8259, 9.0288, 8.8476, 9.1696, 8.8776, 9.0579, 8.8805, 9.0138, 8.9347, 8.9248, 8.8103, 8.9723, 8.8083, 8.8320, 8.8466, 8.9180, 8.9090}.

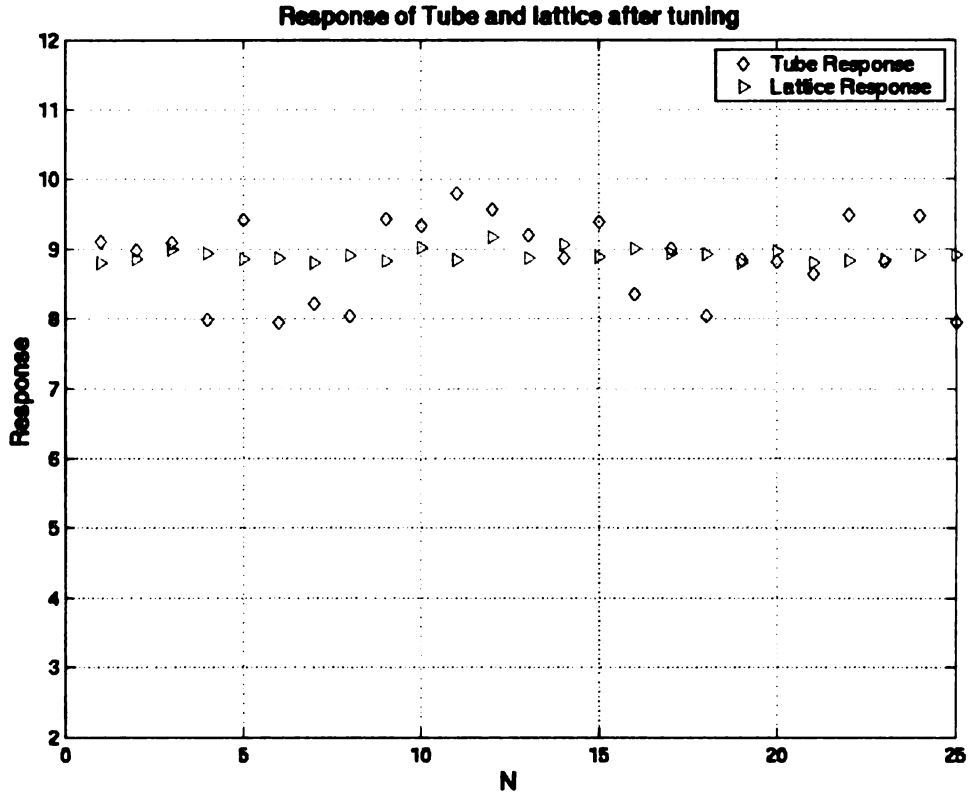


Figure 25: Response function of the tube and the lattice at 25 S_y points.

The response of the lattice closely follows the response of the tube. The maximum error in the response of the lattice is 0.307 and minimum error is 0.0. The average error is 0.124 over 25 points. The goal of the lattice design is to predict the response in initial design stages of the product and it would be difficult to expect that the lattice will replicate the behavior with zero error.

To demonstrate the robustness of the optimization framework for tuning procedure, several test cases with different initial design sites (Both in yield stress domain and area domain) and different mesh sizes were performed. Twenty-five different yield stress points were samples for the tube and response was plotted along with the response of the lattice for $A_{optimum}$ of the lattice. Figure 26 shows the comparison between response of the tube and the lattice at 25 different points. The maximum error in the response of the lattice is 0.319 and minimum error is 0.0. The average error is 0.132 over 25 points

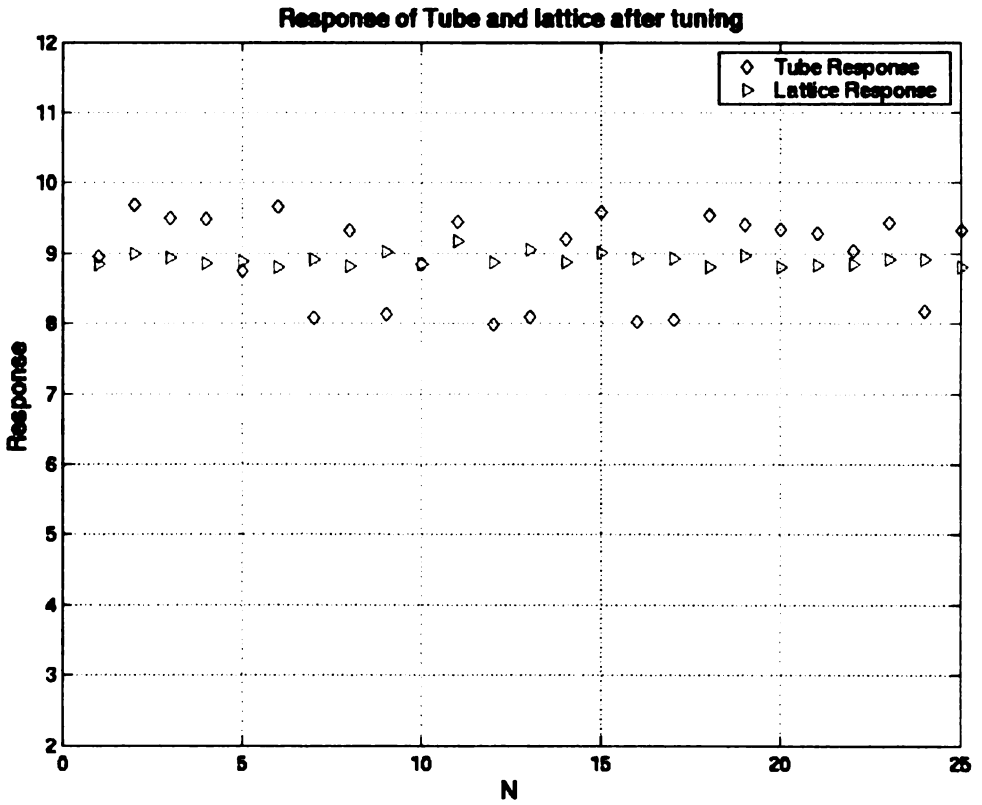


Figure 26: Response function of the tube and the lattice at 25 S_y points

Chapter 4

Conclusion

4.1 Conclusion

The goal of this work was to facilitate faster turn-around time and shorter design cycle in the initial stages of product and process design. The lattice model is proposed for improved crashworthiness analysis of complex structures. Complex structures are computationally expensive and not many changes in the design features can be done in the later stages of design cycle. Lattice model can predict some *relevant* responses of a complex structure with less computational cost. In chapter three we formulated an optimization problem to determine the material and dimensional parameters of the lattice, hoping to replicate the response of the complex structure. This work focused on three main areas as follows:

(1) Selection of initial design, e.g. space filling Latin hypercube designs for function evaluations to be used in construction of surrogate model.

(2) Metamodeling techniques e.g. kriging metamodel, to build the surrogate of expensive functions.

(3) Optimization techniques e.g. pattern search methods, to optimize the surrogate and hence to predict the optimum of expensive functions.

The surrogate management framework unified three areas together to achieve the goal of this work.

4.2 Future work

There are some concerns left at the time of writing this dissertation which must be addresses in order to explore full capabilities of the framework. The framework allows a great flexibility in selection of design sites, construction of surrogates and implementation of different optimization algorithms.

In this work 'space filling' nature of initial design satisfy most of the criteria, if not all, but there should be some method to generate initial design which uses some information from the nature of the objective function (if available).

There are many techniques used for construction of surrogates (few of them are described in chapter two). In the framework developed, kriging metamodel is used to construct the surrogate of the objective function. Kriging is a very powerful interpolation method and it worked out pretty well in this project. Other metamodeling techniques e.g. Response surface method and Wavelet transform can be a good basis for the construction of surrogate in different applications. We plan to explore Response surface method in near future.

Pattern search algorithm is implemented in the framework primarily because we do not have derivative information of objective function. The framework allowed using sequential updates of surrogate which probably improved the robustness of the

framework. We will have to wait until, sophisticated techniques are easily available to compute the sensitivities of nonlinear and expensive functions, before we can use derivative based optimization techniques.

The optimization problem formulated to "tune" the lattice has several features that make it very difficult to solve: it is highly non linear and non smooth. This is a weakness that must be addressed as we focus our attention on larger-scale problems. Diaz ([22]) has raised some issues in his work ([22]) which is quoted here as some of the potential to be explored. There is no proof of existence that guarantees that *all* relevant structural behavior can be reproduced by a given lattice.

BIBLIOGRAPHY

[1] Dennis, J.E. Jr. and Walker, H.F. 1984: Inaccuracy in quasi-Newton methods: local improvement theorems. Mathematical Programming Study. 22, 70-85.

[2] Dennis, J.E. Jr and torczon, V. 1991: Direct search methods on parallel machines. SIAM J. Optimization. 1(4), 448-474.

[3] Torczon, V.. 1995: Pattern search methods for nonlinear optimization. SIAG/OPT Views and News. 6, 7-11.

[4] Trosset, M.W. 1997: I know it when I see it: toward a definition of direct search methods. SIAG/OPT Views and News. 9, 7-10.

[5] Trosset, M.W. and Torczon, V. 1997: Numerical optimization using computer experiments. Technical Report 97-38, ICASE, NASA Langley Research Center, Hampton, Virginia 23681-2199.

[6] Torkel Glad and Allen Goldstein. Optimization of functions whose values are subject to small errors. BIT, 17(2):160-169, 1977.

[7] J.H. May. Linearly Constrained Nonlinear Programming: A Solution Method That Does Not Require Analytic Derivatives. PhD thesis, Yale University, December 1974.

[8] R. Mifflin. A superlinearly convergent algorithm for minimization without evaluating derivatives. Mathematical Programming, 9:100-117, 1975.

[9] M.J.D. Powell. Trust region methods that employ quadratic interpolation of the objective function. Presented at the Fifth SIAM Conference on Optimization, Victoria, B.C., May 20-22, 1996, May 1996.

[10] C. de Boor , 1997. Spline Toolbox User's Guide, The mathworks, Inc., Nattick, MA.

[11] M. Bozzini, F. de Tisi, 1988. An Algorithm for knot location in bivariate least squares spline approximation, Algorithm to Approximations II, Chapman and Hall, New York, pp. 30-36.

[12] M.G. Cox, P.M. Harris, H.M. Jones, 1988. A knot placement strategy for least squares fitting based on the use of local polynomial approximations, Algorithms for Approximation II, Chapman and Hall, New York, pp. 37-45.

[13] P. Craven, G. Wahba, 1978. Smoothing noisy data with spline functions: estimating the correct degree of smoothening by the method of generalized cross-validation, Numerical Mathematics, vol. 31, pp. 377-403.

[14] N. Cressie, 1988. Spatial Prediction and Ordinary Kriging, Mathematical Geology, vol. 20, no.4, pp.405-421.

[15] J. Sacks, W. Welch, W.J. Mitchell, H.P. Wynn, 1989. Designs and Analysis of Computer Experiments, Statistical Science, vol. 4, no.4, pp. 409-435(with discussion).

[16] R.H. Myers, D.C. Montgomery, 1995. Response Surface Methodology: Process and Product Optimization Using Designed Experiments, J. Wiley & Sons, New York.

[17] T.J. Mitchell, M.D. Morris, 1992. The Spatial Correlation Function Approach to Response Surface Estimation, Proceedings of the 1992 Winter Simulation Conference, Arlington, VA, pp. 565-571.

[18] Torczon, V. 1992: PDS: direct search methods for unconstrained optimization on either sequential or parallel machines. Technical Report 92-9, Department of Computational and Applied Mathematics, Rice University, Houston, Texas 77005-1892. In revision for ACM Transactions on Mathematical Software.

[19] G.E.P. Box, N.R. Draper, 1987. Empirical Model-Building and Response Surfaces, J. Wiley & Sons, New York.

[21] J.P.C. Kleinjnen, 1987. Statistical Tools for Simulation Practitionaers, Dekker, New York.

[22] Cori L. Ignatovich, A.R. Diaz, C.A. Soto. Strategies In Design For Enhanced Crashworthiness. Proceedings of DETC00, 2000 ASME Design Engineering Technical Conference, September 10-13, 2000 – Baltimore, Maryland.

[23] Trosset, M.W. and Torczon, V. 1998: From evolutionary operation to parallel direct search: pattern search algorithms for numerical optimization. Computing Science and Statistics. 29, 396-401.

[24] Torczon, V. 1997: On the Convergence of pattern search algorithms. SIAM J. Optimization. 7(1), 1-25.

[25] Robert Michael Lewis and Virginia Torczon. Pattern search algorithms for bound constrained minimization. Technical Report 96-20, ICASE, NASA Langley Research Center, Hampton, VA 23681-0001, January 1998. Submitted to SIAM Journal on Optimization.

[26] Andrew J. Booker, J. E. Dennis, Jr , Paul D. Frank, david B. Serafini, Virginia Torczon, Michael W. Trosset, A Rigorous Framework for Optimization of Expensive Function by Surrogates, Structural Optimization, November 20, 1998

[27] J.H. Friedman, 1991. Multivariate Adaptive Regression Splines, The Annals of Statistics, vol. 19, pp. 1-141.

[28] D. B. Serafini, A Framework for Managing Models in optimization for Computationally Expensive Functions, PhD thesis, Dept of Computational and Applied Mathematics, Rice University, Houston, Texas.

[29] A.R. Diaz, C.L. Ignatovich, C.A. Soto. Strategies In Design For Enhanced Crashworthiness.

[30] D.C. Montgomery, 1997. Design and Analysis of Experiments 4th Edition, J. Wiley & Sons, New York.

[31] G.E.P. Box, W. Hunger, J. Hunter, 1978. Statistics for Experimenters, Wiley, Inc., New York.

[32] G. Wahba, 1990. Spline Models for Observational Data, regional Conference Series in Applied Mathematics, no.59, Society for Industrial & Applied Mathematics, Philadelphia, PA.

[33] T.W. Simpson, J. Peplinski, P.N. Koch, J.K. Allen, 1997. Metamodels for Computer-Based Engineering Design: Survey and Recommendations, submitted to Research in Engineering Design.

[34] Virginia Torczan, Michael W. Trosset. Using Approximation to Accelerate Engineering Design Optimization.

[35] Virginia Torczan. On the convergence of pattern search methods . SIAM J. Optimization, 7:1-25, February 1997.