Modified Loss Functions and Artificial Neural Networks in Nonlinear Multi-Response Optimization Problems

by

Ismael Torres-Pizarro

A thesis submitted in partial fulfillment of the requirements for the degree of

MASTER IN ENGINEERING
in
MANAGEMENT SYSTEMS ENGINEERING

UNIVERSITY OF PUERTO RICO
MAYAGÜEZ CAMPUS
MAY 2008

Approved by:

_____	_____
Mercedes Ferrer, ME Member, Graduate Committee	Date
_____	_____
Viviana Cesaní-Vázquez, PhD Member, Graduate Committee	Date
_____	_____
Noel Artiles-León, PhD President, Graduate Committee	Date
_____	_____
Pedro Vásquez-Urbano, DSc. Representative, Graduate Studies	Date
_____	_____
Agustín Rullán, PhD Department Chairperson	Date

ABSTRACT

This work presents four case studies that are used to compare the suitability of several techniques to solve multiple response optimization problems. The approaches compared include linear regression using Lagrange optimization, linear regression using a "brute force" search optimization approach, and a neural network with the same "brute force" optimization method. Our approach uses the loss function technique described by Taguchi (1986) and modified by Artiles-León (1996) for multiple responses optimization. In general, the regression approach with the Lagrange optimization provided the best results with expected loss values up to 1.17 times the actual minimum loss, but the linear regression using a "brute force" optimization proved comparable with up to 1.31 times the actual minimum loss. Results using neural network were neither acceptable nor expected with expected loss values up to 2.4 times the actual minimum loss.

RESUMEN

Este trabajo presenta cuatro casos de estudios que fueron usados para comparar la adecuacidad de varias metodologías para resolver problemas de optimización multi-respuesta. Las metodologías comparadas incluyeron la regresión lineal usando optimización de Lagrange, la regresión lineal utilizando un método de optimización de búsqueda "fuerza bruta" y una red neural con el mismo método de optimización de búsqueda "fuerza bruta". Nuestra metodología utiliza la función de pérdidas descrita por Taguchi (1986) y modificada por Artiles-León (1996) para la optimización de múltiples respuestas. En general, la metodología de regresión utilizando optimización de Lagrange obtuvo los mejores resultados con valores esperados para la pérdida de hasta 1.17 veces la pérdida óptima, pero el método de regresión lineal utilizando optimización de búsqueda "fuerza bruta" probó ser comparable con valores esperados para la pérdida de hasta 1.31 veces la pérdida óptima. Los resultados de la red neural no fueron ni aceptables ni esperados con valores esperados para la pérdida de hasta 2.4 veces la pérdida óptima.

To my family . . . in particular to my beloved daughter and son, Sonia & Milton…I miss you.

ACKNOWLEDGEMENTS

During the development of my graduate studies in the University of Puerto Rico, several persons and institutions collaborated directly and indirectly with my research. Without their support, it would have been impossible for me to finish my work. That is why I wish to dedicate this section to recognize their support.

I want to express a sincere acknowledgement to my advisor, Dr. Noel Artiles-León because he gave me the opportunity to do research under his guidance and supervision.

TABLE OF CONTENTS

Table List

Tables **Page**

Figure List

Figures **Page**

1 INTRODUCTION

Optimization techniques are very powerful tools used in manufacturing and service industries; for instance, in a production facility, the development of new product characteristics or the improvement of existing ones are common tasks where optimization methods are used. Another example would be a financial broker trying to assess the risk in a financial portfolio who would employ optimization techniques to quantify such risks. Researchers have directed their attention to the case when there is only one goal to optimize. However, in the real world, products and services have several characteristics, often conflictive among themselves. For example, a production manager needs a faster vehicle with lighter materials to build his engine, but lighter materials often have a low stress resistance and those with high stress resistance are very expensive. Likewise, a broker wants to obtain the higher return on his investments, but this requires carrying additional risks in the portfolio. There have been several studies on the harmonization of such different and conflicting characteristics, since a tool exploiting such harmonization would be an excellent strategic weapon.

From those studies, it is clear that choosing parameters that provide global optimization for more than one characteristic of interest is not an easy task. We have to model a real world situation to account for multiple possibilities such as stochastic variables and time-varying "parameters", since highly nonlinear problems are frequently found in current practices of industrial optimization. We need a computerized system in order to deal with so many options.

The base for most current optimization methods is the development of a model, developing equations that could approximate the present system. For selected input, an optimum output is found by different techniques; for example, numerical search by exhausting and evaluating all possible solutions ("brute force" method) or some sort of algorithm that capitalizes on previous information to make a most effective search for the model optimum solution. Recently, the development of artificial neural network (ANN), fuzzy logic and other computerized "expert systems" provide additional alternatives for system optimization. Their major advantage is the design of new approaches without the need to develop system equations. To achieve some success, traditional approaches often require the linearization and simplification of the system's model equations. The expert systems approach is more heuristic in nature, but it is possible to develop the equations from the artificial neural net's response, if needed. (Ramírez, 1997). It has been argued, that it is precisely this heuristic approach that makes artificial neural nets more suitable to be used with forecasting and prediction of highly nonlinear, time-varying stochastic processes when the traditional linearization is often not possible or even desirable.

1.1 Motivation

The main interest in this project is to compare the use of artificial neural networks against multiple linear regressions in the search for optimum solutions for multiple-response optimization problems when the systems are nonlinear in nature. Our approach uses the loss function technique described in Taguchi (1986) and modified by Artiles-León (1996) for

multiple characteristics. Currently, there are some disagreements in the literature about which method provides superior results.

1.2 Literature Review

The experimental optimization of a single response is usually conducted in two phases or steps, following the advice of Box and Wilson (1951). The first phase consists of a sequence of line searches in the direction of maximum improvement. Each search in the sequence is continued until there is evidence that the direction chosen does not result in further improvements. The sequence of line searches is performed as long as there is no evidence of lack of fit for a simple first-order model of the form $\hat{Y}_i = \hat{b}_{0i} + X^T \hat{b}_i + \varepsilon_i$. The second phase is performed when there is lack of linear fit in Phase One, and instead, a second-order or quadratic polynomial regression of the form $\hat{Y}_i = \hat{b}_{0i} + X^T \hat{b}_i + X^T \hat{B}_i X + \varepsilon_i$ is fit.

In the multiple response case, finding process operating conditions that simultaneously maximize (or minimize, as desired) all the responses is quite difficult, and often impossible. Almost inevitably, the engineer must make some trade-offs in order to find process operating conditions that are satisfactory for most (and hopefully all) the responses. Two main approaches have been widely used in the industry for the optimization of multiple response processes: the loss function and the desirability function.

The desirability approach is a popular method that assigns a "score" to a set of responses and chooses factor settings that maximize that score. It is based on the idea that the

"quality" of a product or process that has multiple quality characteristics, with one of them outside of some "desired" limits, is completely unacceptable. The method finds operating conditions, X's that provide the "most desirable" response values. For each response $Y_i(x)$, a desirability function $d_i(Y_i)$ assigns numbers between 0 and 1 to the possible values of Y_i, with $d_i(Y_i) = 0$ representing a completely undesirable value of Y_i and $d_i(Y_i) = 1$ representing a completely desirable or ideal response value. Individual desirability's are then combined using the geometric mean, which gives the *overall desirability* $D = [\prod_i^k d_i(Y_i)]^{1/k}$ with k denoting the number of responses. Notice that if any response Yi is completely undesirable (di(Yi) = 0), then the overall desirability is zero. In practice, fitted response values are used in place of the Yi (Ventresca, 1993).

A major drawback of this approach is that we need a different desirability function for each response depending on the desired objective. A different function must be used to maximize the first response, yet another to minimize the second one, and so forth. The loss function tackles this situation by converting all responses into just one function that needs to be minimized.

The use of Taguchi's loss function as objective function for multi-response constrained optimization is not new, (Pignatiello,1993). However, a different approach was presented by Niebles-Bermúdez (1997) using a multi-response optimization methodology based on the loss function described by Taguchi, (1986) with the dimensionless modification proposed by Artiles-León (1996) that makes possible the combination of multiple loss functions corresponding to different responses. The main importance of this dimensionless

modification is that it makes possible to leverage all the responses in such a manner that a collective optimization function results. This overall optimization function makes the multi-response problems a constrained optimization event easily solved by a spreadsheet without the additional complexities worked out by Del Castillo and Montgomery (1993). Ames, et al. (1997) presented a similar approach.

Niebles-Bermúdez (1997) worked with different polynomial functions and provided recommendations on the sample size, function order, and the function approximation technique model, i.e., type of experimental design, depending on the type of process "true" function: linear, quadratic and cubic. He obtained his results by providing: a) a previously selected polynomial function, b) the "true" process equation function, and c) observing the results for changing sample size, function order, and approximation technique. His results showed, among other things, that: a) the Euclidean distance between the "true" optimum (the optimum for the "true" process equation) and the optimum found by the model being used correlate negatively with sample size b) the "true" process equation order and the order for the model being used should match for better adjustment between solutions and, c) in general, great caution must be exercised when selecting a model for process optimization, since for multiple response designs many factors seem to affect the results.

A similar approach is discussed in Ames et al. (1997), where the quality loss function is used to solve multi-response problems when describing the global quality loss function (GQLF) as the loss to society resulting from the deviation from the target for each of the different responses. The method is straightforward, but it has the complication of a relative weight parameter for each response. Ames, et al. (1997) neglect the selection of the correct

weight as a minor problem; however, their results found no single output response without the correct assessment of these parameters. On the other hand, the proposed method by Artiles-León (1996) and Niebles-Bermúdez (1997), to weigh each response by its own specifications makes handling the resulting global function computationally easier.

Del Castillo (1997) developed an ANSI FORTRAN program that solves a dual response system (DRS) using the algorithm (DRSALG) developed by Del Castillo,(1996). This method is a search technique where we describe the problem as a constrained optimization method within radial bound that solves dual response system equations. However, they used relatively old and particular software to solve this problem, thus discouraging the use of this method. A new algorithm compatible with relatively new spreadsheet packages, regardless of user preference and without additional cost, would bring wider acceptance for these techniques.

More recently, Kim and Lin (1998) worked a dual response surface methodology (DRSM) case with the methods proposed by Vining and Myers (1990), Vining (1998) and Myers and Carter (1973), and compared these techniques against the use of fuzzy logic models for simultaneous maximization of mean and standard deviation. They worked out problems on the quality of printing process and catapult parameters settings, where the RSM technique for each process developed experimentally with its respective polynomial equation model built from such experimental data. First, they constructed the RSM model and then developed a fuzzy set with the use of the membership function concept. They built the function concept with fuzzy logic rules and used it to measure the degree of satisfaction that a decision maker gets from the multi-response output. The objective is to maximize this

satisfaction by maximizing the membership function. This approach has a particular advantage over mean square error (MSE) minimization, which is the technique used in RSM and traditional linear regression techniques. We can get MSE minimization by variance minimization or bias minimization, whereas the membership function approach balances the contribution of the bias and the variance in the optimization process. In this sense, the membership function works as a dimensionless loss function, since both objective functions take into account more than one output response. We found an example of the use of this methodology in Lin and Tu (1995) as fuzzy logic theory in one of the recent approaches dealing with optimization of multi-response problems. However, the dimensionless loss function is a better approach, since the use of specifications as weights in the objective function makes the measurement for satisfaction to be already included in the model.

Another approach to optimization of multi-response problems is the use of artificial neural networks (ANN) as stated by Stern (1996). However, resistance to its use among some researchers has grown due to its "black box" approach. That is, determining why an artificial neural net makes a particular decision is a difficult task, but Benítez, et al. (1997), make an interpretation of an artificial neural net response. Innovative uses of artificial neural net include, among others: a) solving financial engineering problems by Refenes et al. (1997), where an artificial neural net is used to tackle the option pricing problem; b) multivariate engineering control by Zhang et al. (1997) and Odmidvar et al. (1997), where the learning ability of an artificial neural net is the main advantage used to derive a neural control algorithm independent of the mathematical model of the system being controlled; and c) self-

tuning exponential moving average control by Smith and Boning (1997), where the main characteristic of interest is the artificial neural net ability to approximate functions.

The flexibility of the artificial neural net to deal with so many engineering and scientific applications is the main reason artificial neural net advocates use them as optimization tool for multi-response problems not fully explored. However, De Veaux et al. (1998) advised against the "black box" approach when using artificial neural nets. De Veaux et al. explained that with enough parameters, the artificial neural network is able to approximate any reasonable function. However, there are dangers related to the over-fitting of the data. Known as the bias-variances trade-off, the ability of the artificial neural net to predict the previously unseen data is in conflict with its ability to fit the training data accurately. Although not all artificial neural nets need training data, most of the standard models do need this training. The need for the training data is for teaching the artificial neural network to identify a case; that is, for this input, this is the output and for this other input, this is the corresponding output, and so on. Once trained, an artificial neural network can predict, with some success rate, what the output would be for a given input; thus, the ability to mimic a nonlinear function is clearly within the artificial neural network possibilities.

On the other hand, as earlier as 1997, there was some research done by Smith and Mason (1997) comparing the performance of regression models and artificial neural nets for very simple linear equations. Their results showed that the regression models were a superior choice for prediction performance. It should be noted that regression techniques require the model to be additive. This is not a requirement for artificial neural nets, thus it might be an

advantage to explore. The study was limited to a one input-one output linear problem; thus, an artificial neural network might be a superior technique for non-linear systems.

Its users often overlook the downside of artificial neural net, left on their own to perform some sort of cross-validation in most software packages. De Veaux et al, (1998) proceeded to model a multilayer feedfoward neural network as a nonlinear regression method, and they applied the standard asymptotic theory from nonlinear regression modified by weight decay to find prediction intervals for the proposed artificial neural net. Townsend and Tarassenko (1999) worked with the limitation of lack of confidence measurements placed in the artificial neural net output estimates.

A large amount of literature has been published in journals about the different techniques and methods of applying artificial neural nets. The creation of new organizations and the development of an industry in software used for financial optimization using these techniques are just two of the signals that indicate that artificial neural networks are here to stay. One of the most successful packages is NeuralWorks Professional II/Plus by Neuralware, but other software packages like Matlab provide "toolboxes" with specialized functions to develop program applications.

It is by using one of these packages (namely, NeuralWorks Professional II/Plus) that Tong and Hsieh (2000) presented their conclusions about an artificial neural net used as optimization technique capable of identifying the real optimal parameters when used to solve multi-response problems. Particularly interesting is the fact that they recommended the use of artificial neural nets over "traditional" regression techniques together with the desirability function as an objective function, and concluded that the neural net approach was a superior

methodology. They reached this conclusion despite the fact that their approach predicted some responses that were not within the feasible solution space for the problem they selected as illustration. Fogliatto and Albin (2000) used the predicted coefficient of variation (predicted CV) as objective function. However, their designs were not orthogonal and a multicollinearity problem existed. In industrial optimization, we must avoid this type of problem by careful selection of input variables. An interesting question could be to see if the artificial neural net or the dimensionless loss function could tackle the problem of a careless selection of input variables.

In 2001, two papers dealing with the multi-response problem appeared in the Journal of Quality Engineering in the same issue. Surh and Baton (2001) reviewed the latest literature published and concluded that the use of the loss function as the objective function to be optimized was the superior approach to solve multi-response problems. Duarte-Ribeiro, et al. (2001) proposed the use of an objective function that includes the variance of the responses. In financial optimization problems, the typical two responses are the investor return on investment (ROI) (a response the investor wants to maximize) and the risk associated with that investment, usually modeled by the return on investment's standard deviation (a response the investor wants to minimize). However, Duarte-Ribeiro's approach is rather cumbersome, since instead of adding the standard deviation as just another response to optimize, they use the variance and set it as if it were a different type of function, not to be scaled. The dimensionless loss function would set the standard deviation as the response and would scale it by the interval of variation that the user would like or would be willing to accept.

Holmes and Mergen (2001) used capability ratios as objective functions. This approach is very complex and cumbersome and the authors concluded cautionary use of these capability indexes as objective function. Tong, et al. (2001-02), used the desirability function and standard dual response surface methodology. The methodology is simple, easy, and practical to follow, but the use of the desirability function as an objective function requires the system to be classified as nominal-the-best (NTB), larger-the-best (LTB) or smaller-the-best (STB) for each response as the desirability function changes with each classification. There is no need to do this when using the dimensionless loss function as the objective function.

A few years ago Ko, et al. (2005) utilized a loss function-based methodology similar to the one proposed by Pignatiello (1993), but extended it for multiple response optimization purposes. Their main contribution is to demonstrate how noise could affect the robustness of the solutions.

In conclusion, the literature revision suggests that it remains unknown if an artificial neural network could be a superior optimization technique for multi-response problems. Standard multiple regression methodology has provided outstanding results on more than one occasion with the advantage of being easy and fast to implement. The literature supporting the use of artificial neural nets largely ignores the words of caution from the studies that have shown the superiority of a more traditional methodology. However, there is general agreement that the loss function is a superior approach for benchmark purposes.

1.3 Summary of Following Chapters

The necessary background theory was developed in Chapter 1. Chapter 2 details the methodology used, as well as the particular application of the theory to this research. The third chapter presents the statement of the problem, the data and its analyses for the four cases study experiments done. Conclusions are presented in Chapter 4.

2 THEORETICAL BACKGROUND

2.1 Objectives

Our principal objective was to determine if the better optimization technique for a multi-response problem is an artificial neural network (ANN) or a regression model when a loss function is used.

We expect that the comparison would settle the controversy between the two methods as which one is a superior search technique for solutions of multi-response problems.

We also illustrate the superior approach using the loss function as the weighing function by showing the flexibility of its methodology when tackling diverse problems such as smaller-the-best, nominal-the-best or larger-the-best responses.

2.2 Methodology

An overview of the methodology followed will be presented first and detailed discussion of each step will follow afterward.

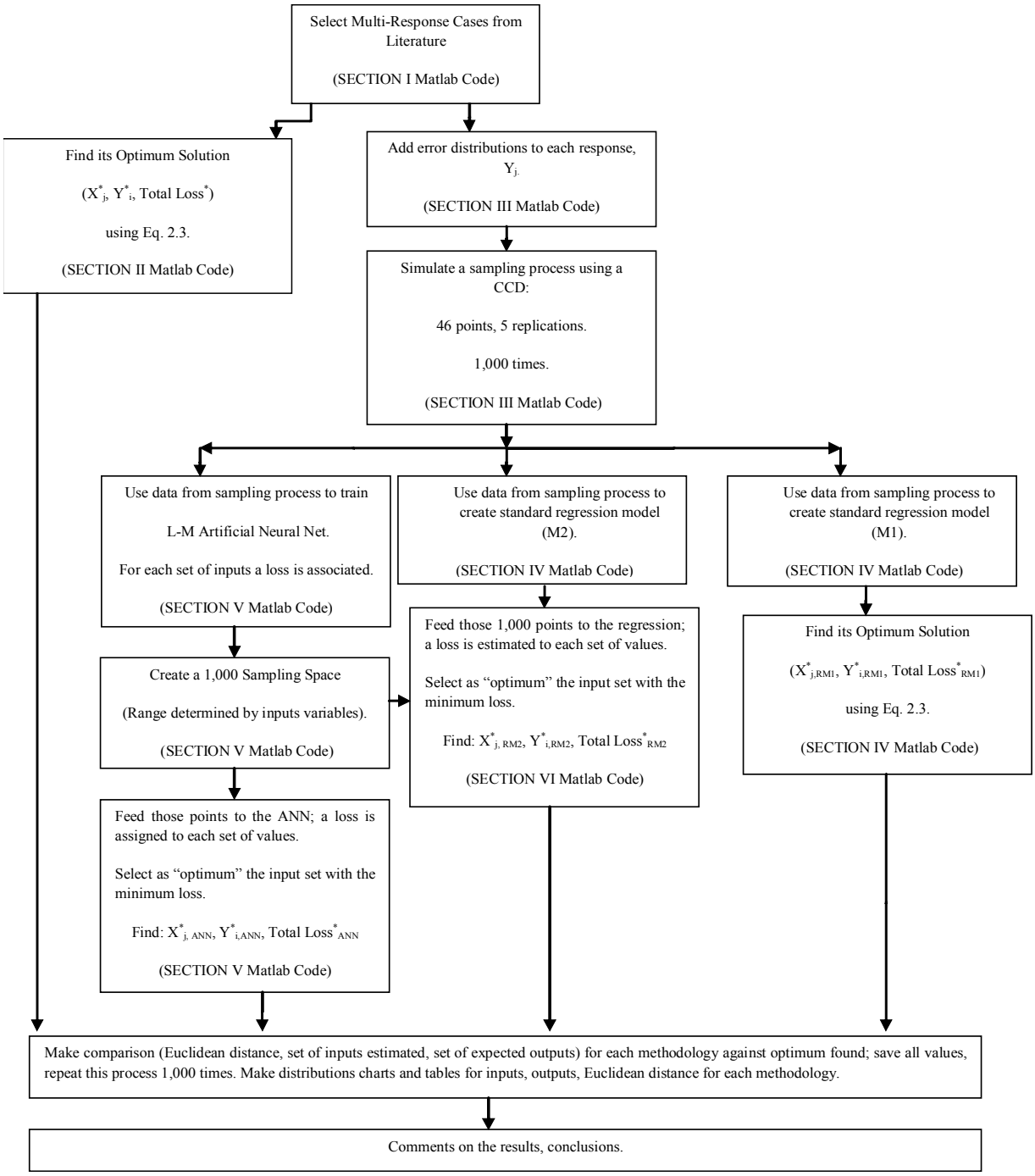

Figure 2.1.Methodology Process Flowchart.

A Matlab code addressing the objectives was developed for each case. Main Matlab reference used were: a) Etter (1996) b) Hansselman and Littlefield (1998) c) Marchand (1996) d) Pärt-Enander and Sjöberg (1999) e) Redfern and Campbell (1998). The first two sections of the Matlab code developed (Section I "DATA FOR PROBLEM #" & Section II "OPTIMIZATION FOR ORIGINAL PROBLEM"), found the true optimum for several selected multi-response problems from the existing literature. We selected the adjusted equations found by different authors; such as Kim and Lin (1998) and Del Castillo (1996); for their particular systems in various papers and worked them out as if they were the actual system response. In that manner, the solution found is taken as the optimum solution for the problem. For each response Y_i, where $i = 1,2,...I$, we approximated the multi-response problems in the region of interest by the quadratic form:

$$\hat{Y}_i = \hat{b}_{0i} + X^T \hat{b}_i + X^T \hat{B}_i X + \varepsilon_i \qquad 2.1$$

We assumed the actual quadratic form for the system input-output relationship was:

$$Y_i = b_{0i} + X^T b_i + X^T B_i X \qquad 2.2$$

where $\hat{Y}_i$ is the estimate for the actual Y_i response equation with K + 1 terms but with only J independent input where the K terms are linear combinations of the J independent input; $\hat{b}_{0i}$ is the estimated coefficient (scalar) for the b_{0i} scalar coefficient, $\hat{b}_i$ the K x 1 estimated coefficient vector for the K x 1 coefficient vector b_j, and $\hat{B}_i$ the estimated K x K coefficient matrix for the K x K coefficient matrix B_i and ε_i is the 1 x 1 error scalar accounting for the

difference between the observed value of the output and the estimated value of the output (that is, $\varepsilon_i = Y_i - \hat{Y}_i$); typically ε ~NIID (0, σ^2) where σ^2 is the variance of the response estimated by the mean square error (MSE).

We applied the modified loss function approach to this problem to obtain the optimum solution point for this particular problem. The dimensionless loss function (a total standardized loss, TSLoss) approach as described by Artiles-León (1996) is:

$$\min Z = \text{TSLOSS}\,(Y_i) = 4\sum_{i=1}^{I}\left[\frac{Y_i - T_i}{USL_i - LSL_i}\right]^2 \qquad 2.3$$

Subject to:

$$Y_i = f(X_j)$$
$$-LL_j \le X_j \le UL_j \;\; \forall \;\; j = 1,2,\ldots,J$$

where Y_i is the ith output response for the multi-response problem; and X_j is the independent input to the system; T_i is the nominal target for the ith response. T_i could be nominal the best, larger the best or lower the best values. USL_i is the upper specification level for the response and LSL_i its lower specification level for response variable i, $i = 1,2,\ldots I$. UL_j is the upper value for X_j and LL_j its lower value for each $j = 1,2,\ldots,J$.

Using this approach, a constrained optimization problem is formed where optimization techniques could be used to find the optimum set of inputs, namely, X_j^* that gives us Y_i^*, the optimum response and TSLoss*, the minimum total loss for that problem. Those values were used as our standards to measure the fitness of the two approaches, namely, the artificial neural network, and the regression model.

Section III of the Matlab code (Section III "CCD SIMULATION AND EXPERIMENTAL DATA GATHERING (COMPUTER SIMULATED") consisted of simulating the sampling from an actual noisy environment. We did this using a computer simulation that added some noise (error) to each response to simulate the sampling process using a central composite design (CCD). We were trying to simulate an actual sampling process where we would not get the same result every time, but variations in the output occurred for each set of input. For example, if Y_1 is the first response of the multi-response problem, the computer added an error from a normal distribution with e~NIID(0,σ^2); as another example, to Y_2 the second response in the same problem, an error following a 3-parameter Weibull distribution was added.

An important consideration for both responses is that we want to select the value of σ^2 so that the potential process capability, Cp, for each response equals 2. For example, Y_1 could be a mean as defined by Equation 2.1.We added to this mean the random error, ε_1, modeled by a normal distribution, with mean = 0 and variance = σ^2. The new random variable, $\hat{Y}_1$ has mean = $E(Y_1 + \varepsilon_1) = Y_1 + 0 = Y_1$ and, variance= var $(Y_1 + \varepsilon_1)= 0 + \sigma^2 = \sigma^2$ since Y_1 is a constant term. Then, from the definition of Cp,

$$C_p(\hat{Y}_1) = \frac{(USL - LSL)}{6 * STDDEV(\hat{Y}_1)} \qquad 2.4$$

or $\text{var}(\hat{Y}_1) = \text{var}(\varepsilon_1) = ((USL-LSL)/12)^2$, since we predetermined the Cp as 2. As for Y_2, could be a standard deviation, a similar procedure is applicable if we establish the addition of

a random variable with E(ε_2) = 0. Thus, for Y_2 we have var($\hat{Y}_2$) = var(ε_2) = ((USL-LSL)/12)^2. Typical models for the standard deviation are the lognormal random variable or the squared root of a χ^2 random variable, Roussas, 1997. However, modeling the random error for a standard deviation requires careful consideration. First, we want to add a random error that has a mean equal to zero, E(ε_2)=0; thus, some values of the random variable added to Y_2 are going to be negative, but, since Y_2 represents a standard deviation, we need to make sure that $Y_2 + \varepsilon_2 \geq 0$ always. We needed to calculate the minimum value that Y_2 could achieve within the experimental space region and then select an error, that when added to the minimum Y_2, ensures our non-negativity constraint. In summary, for the error added to Y_2 we needed a random variable that satisfies the conditions:

a) E(ε_2)=0;

b) $Y_2(1,X_1, X_2, X_3\ldots,X_k) + \varepsilon_2 \geq 0$ for all X_k, $k = 1,2,\ldots,K$ and,

c) var($\hat{Y}_2$) = var(ε_2)= ((USL-LSL)/12)^2.

A possible candidate for these requirements is to find, for each case, a 3-parameter Weibull distribution with CDF: $F(x) = 1-\exp\{[(x - \theta)/\beta]^{\alpha}\}$ and letting $\theta = e_2$ such as the errors, ε_2, randomly created by the distribution satisfy condition (b) above always. That is, we select the location parameter θ to be equal to the minimum possible value of the distribution, e_2 (a negative number) which has to be less than the minimum possible value of Y_2. The selection of α and β is done solving the simultaneous system of equations set up by conditions (a) and (c) above while satisfying condition (b) in other words: E(ε_2) =

$\beta*\Gamma\{(\alpha+1)/\alpha\} + \theta = 0$ and $var(\varepsilon_2) = \beta^2*\Gamma\{(\alpha+2)/\alpha\} - [\beta*\Gamma\{(\alpha+1)/\alpha\}]^2 = ((USL-LSL)/12)^2$ while MIN $Y_2 \geq e_2$. The goal is to simulate the execution of the actual sampling experiments using a Box-Wilson Central Composite Design (CCD) design. A CCD contains an imbedded factorial or fractional factorial design with center points that is augmented with a group of "star points" that allow estimation of curvature. If the distance from the center of the design space to a factorial point is ±1 unit for each factor, the distance from the center of the design space to a star point is $\pm \alpha$ with $|\alpha| > 1$. The precise value of α depends on certain properties desired for the design and on the number of factors involved. Similarly, the number of centerpoints runs the design will contain also depends on certain properties required for the design (Montgomery, 1991) and (Myers and Montgomery, 1995).

We selected the output specification limits for the responses by taking into consideration the type of problem and their maximum and minimum values reported by the author. If, for example, the output response corresponds to a mean output, it is reasonable to suppose that a normal distribution could serve as model for its sampling distribution. If the author maximized that response, that particular response is a "more is best" problem. The 3-parameter Weibull distribution modeled the standard deviation for those output responses simulating variation. If we want to minimize responses measuring variation; those particular responses are a "less is best" problem. A multiple input- multiple output (MIMO) problem such as these could have such conflicting responses. For both types, we developed a methodology to ensure the C_p of that particular response is two (2).

In section IV of the Matlab code (Section IV "MULTIPLE REGRESSION ESTIMATION"), we developed the regression model from the data generated. Next, we used Equation 2.3 to estimate optimal values for each experiment. From there, we found the optimum input setting, its associate optimum response and its total loss function.

In section V of the Matlab code (Section V "ARTIFICIAL NEURAL NETWORK MODEL"), we trained a neural network to estimate values for each of the sets of input coming from the CCD and their response plus the added errors. An artificial neural network (ANN), often just called a "neural network" (NN), is a mathematical model or computational model based on biological neural networks, (Wikipedia, 2008. http://en.wikipedia.org/wiki/ Artificial_neural_network). It consists of an interconnected group of artificial neurons and processes information using a connectionist approach to computation. Basically an ANN's is a large number of the simple processing units called neurons (or nodes), Haykin (1994). Each neuron is connected to other neurons by means of directed communications links, each with an associated weight (w's). In most cases an ANN is an adaptive system that changes its structure based on external or internal information that flows through the network during the learning phase. In more practical terms neural networks are non-linear statistical data modeling tools. They can be used to model complex relationships between inputs and outputs or to find patterns in data. Neural network models in artificial intelligence are usually referred to as artificial neural networks (ANNs); these are essentially simple mathematical models defining a function $f: X \rightarrow Y$. (Mathworks, 2008.

http://www.mathworks.com/access/helpdesk/help/ toolbox/nnet/). Each type of ANN model corresponds to a class of such functions.

The weights represent information being used by the net to solve a problem. Each neural has an internal state, also called activation level or transfer function, f, which normally is a nonlinear function of the inputs it has received. The particular behavior of an ANN depends on both, the weights and the input-output function (transfer function) that is specified for the units. This function typically falls into one of three categories: a) linear (or ramp) b) threshold or c) sigmoid. For linear units, the output activity, a, is proportional to the total weighted output. For threshold a unit, the output is set at one of two levels, depending on whether the total input is greater than or less than some threshold value. For sigmoid units, the output varies continuously but not linearly as the input changes. Sigmoid units bear a greater resemblance to real neurons than do linear or threshold units, but all three must be considered rough approximations. Due to its differentiability, one of the most commonly used functions is the log-sigmoid transfer function, a=logsig(n) = 1/(1+exp(-n)). This transfer function takes the input from plus or minus infinity and mapped an output into the [0, 1] range.

Typically, a neuron sends its activation as a signal to several others neurons, thus, several layers forms the net. The sizes of the input and output network layers are usually determined by the nature of the problem. In practice, it is common to choose the simplest network that performs. The neural network works by taking a scalar input p which is multiplied by some scalar weight w, to form wp, now one of the possible many terms that is

sent to a summer. It will nee another input, 1, which is multiplied by a bias b, and then passed to the summer. The summer output n, often referred to as the net input, goes into a transfer function f, which produces the scalar neuron output, a. The neuron output is calculated as a = f (wp + b). Note that w and b are both adjustable scalar parameters of the neuron. Typically the transfer function is chosen by the designer, and then the parameters w and b are adjusted by some learning rule so that the neuron input/output relationship meets some specific goal. The word network in the term 'artificial neural network' arises because the function f(x) is usually defined as a composition of other functions $g_i(x)$, which can further be defined as a composition of other functions. This can be conveniently represented as a network structure, with arrows depicting the dependencies between variables. A widely used type of composition is the nonlinear weighted sum, where $f(X) = K(\Sigma\, w_i g_i(x))$, where K is some predefined function, such as the log-sigmoid. Commonly one neuron, even with many inputs, is not sufficient. We might need R inputs, operating in parallel, in what is called a layer. Note that now, each of the R inputs is connected to each of the available neurons and that the weight matrix now has more than one row, say, S. The layer includes the weight matrix W, the summers, the bias vector b, the transfer function boxes and the output vector a. It is common for the number of inputs to a layer to be different from the number of neurons (i.e., $R \neq S$).

Once a network is selected, it needs to be trained in the sense that its initial weights must be determined for a particular function (Fausett F., 1994). There are several methods for training the net such as Supervised Training, one of the most typical net settings which is

accomplished by presenting a sequence of training vectors, or patterns, each with an associated target output vector. The weights are then adjusted according to a learning algorithm. Another popular technique is Unsupervised Training where a sequence of inputs vectors is provided, but no target vectors are specified. The net modifies the weights so that the most similar input vectors are assigned to the same output (or cluster) unit. The neural net will produce a representative vector for each cluster formed.

For nonlinear function approximation, the Levenberg-Marquardt algorithm with multiple hidden layers and the Log-Sigmoid transfer function is the most widely used method (Ramirez, N.D., 1997). It will be the neural net used for this work. The Levenberg-Marquardt algorithm is an effective general method of training multilayer neural networks and it played a major role in the reemergence of the ANN's as a tool for solving a wide variety of problems. The Supervised Training technique will be selected to train the net where the training vectors will be the series of observations coming from the simulation. Good introductions for artificial neural networks can be found at: a) Bose and Liang (1996) b) Fausett (1994) and c) Haykin (1994).

The neural net used here is a Levenberg-Marquardt (L-M) net that is a variation of the Newton Method and one of the most successful training methods available, (Ramírez, 1997). The Levenberg-Marquardt algorithm (or LMA) provides a numerical solution to the problem of minimizing a function, generally nonlinear, over a space of parameters of the function. These minimization problems arise especially in least squares curve fitting and nonlinear programming. The LMA interpolates between the Gauss-Newton algorithm (GNA) and the

method of gradient descent. The LMA is more robust than the GNA, which means that in many cases it finds a solution even if it starts very far off the final minimum. On the other hand, for well-behaved functions and reasonable starting parameters, the LMA tends to be a bit slower than the GNA. The LMA is a very popular curve-fitting algorithm; most software with generic curve-fitting capabilities provides an implementation of it. (Wikipedia, 2008. http://en.wikipedia.org/wiki/Levenberg-Marquardt_algorithm).

Once trained to identify the responses for any given set of input, the program generated one thousand sets of input in a random manner for the artificial neural network estimation of their correspondent output. Once completed, the program used Equation 2.3 and found the optimum input setting, its associate optimum response and its total loss function.

The artificial neural network selected the minimum total loss from the one thousand samples generated.

In section VI of the Matlab code, (Section VI "VALUES ACCUMULATION") is the comparison of optimum values found in the two techniques. We considered two different measurements for fitness. The Euclidean distance:

$$ED_G = \sqrt{\sum_{j=1}^{j=n} (X^*_j - X^*_{jG})^2} \qquad 2.5$$

where G = 0 or 1. When G = 0, the methodology used was the regression and when G = 1 it was the artificial neural network. This measurement considers the ability of the methodology to find inputs that optimize the system responses.

The second measurement will be the total dimensionless loss function. The expected value of this function is zero, thus, the best methodology should have targeted the zero as mean for this function. Graphical comparisons by means of histograms were prepared for these two measurements and results discussed.

One should note that the methodology to select the optimum points is different for the regression method and for the artificial neural network. The regression used an internal Matlab function, namely "fmincon", which works by using the Lagrange optimization method, well discussed in calculus (Moon and Stirling, 2000). We called this analytical procedure R(M1) for regression methodology one. Recall from basic calculus that Lagrange multipliers can be used to solve nonlinear problems in which all the constraints are equality or inequality constraints (Winston, 1995). For instance, consider nonlinear problems of the following type:

$$\max \text{ (}or\text{ min) } f(x)$$

$$\text{s.t.} \quad g_i(x_1,x_2,x_3,\ldots x_n) = a_i$$

$$w_k(x_1,x_2,x_3,\ldots x_n) \geq b_k$$

where

$$i = 1,2,\ldots I$$

and

$$k = 1,2,\ldots K$$

To solve the nonlinear problem, one associates a multiplier λ_i with the ith equality constraint and associate another multiplier, μ_k with kth constraint in the nonlinear problem and forms the Lagrangian function

$$\square\square\square_1, \square_2, \square_3, \ldots, \square_{\square}, \lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_{\square} \ldots, \mu_1, \mu_2, , \mu_3, \ldots, \mu_{\square} =$$

$$\square\square\square_1, \square_2, \square_3, \ldots, \square_{\square} + \sum_{\square=1}^{\square} [\lambda_{\square}(\square_{\square} - \square_{\square}\square_1, \square_2, \square_3, \ldots, \square_{\square}] + \sum_{\square=1}^{\square} [\mu_{\square}(\square_{\square} - \square_{\square}\square_1, \square_2, \square_3, \ldots, \square_{\square}]$$

where $\mu_k \geq 0$ ⊚ k,

Then, it can be shown that if (x_1^*, x_2^*, …, x_n^*, λ_1^*, λ_2^*, ⊚⊚⊚, λ_I^*,μ_1^*, μ_2^*, ⊚⊚⊚,, μ_K^*) solves the unconstrained problem:

Max (or Min) L(x_1, x_2, …, x_n, λ_1, λ_2, ⊚⊚⊚, , λ_I, μ_1, μ_2, ⊚⊚⊚⊚, , μ_K),

then (x_1^*, x_2^*, …, x_n^*) solves also the original constrained problem. Moreover, we know that for (x_1^*, x_2^*, …, x_n^*, λ_1^*, λ_2^*, ⊚⊚⊚,, λ_I^*,μ_1^*, μ_2^*, ⊚⊚⊚,, μ_K^*)) to solve associated Lagrangian problem, it is necessary that at (x_1^*, x_2^*, …, x_n^*, λ_1^*, λ_2^*, ⊚⊚⊚,, λ_I^*,μ_1^*, μ_2^*, ⊚⊚⊚,, μ_K), (Sethi and Thompson, 2000)

$$\frac{\partial L}{\partial x_1} = \frac{\partial L}{\partial x_2} = \cdots = \frac{\partial L}{\partial x_n} = \frac{\partial L}{\partial \lambda_i} = \frac{\partial L}{\partial \mu_k} = 0$$

⊚ k and i,

We called R(M1) to the analytical procedure that uses regression to model the process and this Lagrange optimization method to find the best settings. On the other hand, the artificial L-M neural network (ANN) selected the best (lower) total optimum from a 1,000 set points generated randomly over the input space. This is a "brute force" method that we called just NN (for neural net); finding of the optimum solution depends on the solution space sample size; therefore, we needed to ensure there was ample opportunity for the neural net to find the optimum. To be able to compare the solutions for both methods; a third methodology is also presented. A regression using the exact same 1,000 set points generated for the artificial neural network. We called this third method R(M2) for regression methodology two. We expected such "brute force" methodology used in R(M2) and the artificial neural network to produce results as good as R(M1).

3 Case Studies and Results

3.1 Introduction

The main objective of the study is to establish which methodology is a superior approach for multiple response problems when the loss function is used. To achieve that objective, we selected four cases from the literature and take the equations the authors adjusted for their estimate model of the process as the true equations for the system. We will not attribute the same physical meaning to the responses that the different papers do. For the purposes of this study they are just two responses with different optimization goals and different errors distributions.

The papers were selected because the variety of applications they showed. The familiar catapult example, a complex chemical situation and a production process allows us to realize the wide application of the multiple response problem studied. The particular characteristics of the cases: two dual quadratic responses problems, a dual linear case and a combination of a quadratic and linear response situation are examples of the type of equations that could be addressed.

It is important to ensure that the methodologies are robust to the error normality assumption; therefore to ensure such robustness we will add a normal error distribution to one of the responses and a non-normal error distribution to the other.

3.2 Case Problem 1: Dual Quadratic Responses A

The catapult experiment from Del Castilllo (1996) is an example to study the behavior of a system with dual quadratic responses. In this paper, the control variables (input) X_1, X_2 and X_3 were the arm length, the stop angle, and the pivot height respectively. The output variables (responses), Y_1 and Y_2, were the traveled distance and its standard deviation (σ) respectively. In this case, the adjusted equations as reported by Del Castillo are:

$$Y_1 = 84.88 + 15.29X_1 + 0.24X_2 + 18.8X_3 - 0.52X_1^2 - 1.8X_2^2 + 0.39X_3^2 + 0.22X_1X_2 + 3.60X_1X_3 - 4.42X_2X_3 \quad 3.1$$

and,

$$Y_2 = 4.53 + 1.84X_1 + 4.28X_2 + 3.73X_3 + 1.16X_1^2 + 4.40X_2^2 + 0.94X_3^2 + 1.20X_1X_2 + 0.73X_1X_3 + 3.49X_2X_3 \quad 3.2$$

where the input vector **X** is bounded between:

$$-1.68179 \le Xj \le 1.68197 \text{ for } j = 1,2,3 \quad 3.3$$

Del Castillo wanted to maximize Y_1 and minimize Y_2 simultaneously. In other words, he wanted to find the combination of inputs that gives us the longest (in the mean sense) and most consistent distance possible for any possible combination of input within that range. We set the same purpose for each response; therefore, we established the output (responses) specification limits from the Del Castillo example's results to compare our solutions. See Appendix A, Section I.

Del Castillo performed a sensitivity analysis upon output variable Y_1 resulting in a minimum value of 105.59 and a maximum of 116.55. Our methodology requires specification limits for each response and inputs. Since Y_1 is a variable to be maximized, it seems reasonable to establish the specification limits for such variable as [100, 150] since this interval contains Del Castillo's. See Appendix A, Section I.

Using a similar reasoning as that explained in the previous paragraph, the specification limits for Y_2, are [0, 11]. See Appendix A, Section I. This upper specification limit is taken from Del Castillo results. We solved the problem finding the set of inputs maximizing the Y_1 response while minimizing the Y_2 response using the loss function. The problem solved was a constrained minimization problem as described by Eq. 2.3. See Appendix A, Section II.

Then we needed to simulate a sampling process using a CCD arrangement where the values of the lower and upper limits for the inputs are given as $(-\alpha,\alpha)$ where α = [number of factorial runs] $^{0.25}$ in this case, we are using a full factorial with three factors and two levels, therefore $|\alpha| = [2^3]^{0.25} = 8^{0.25}$; therefore, $|\alpha| \approx 1.68179$. One thousand simulations runs were done, each a sample size, n, of five (5). See Appendix A, Section III.

An important consideration for both responses is that we want to select the value of their individual variances, σ_i^2 so that the potential process capability for each response equals 2. We added to Y_1 the random error, ε_1, modeled by a normal distribution, with mean = 0 and variance = σ^2. The new random variable has mean = $E(Y_1 + \varepsilon_1) = Y_1 + 0 = Y_1$ and, variance = var $(Y_1 + \varepsilon_1) = 0 + \sigma^2 = \sigma^2$ since Y_1 is a constant term. Then we want,

$C_p(\hat{Y}_1) = \frac{(USL - LSL)}{6 * STDDEV(\hat{Y}_1)} = 2 = (150\text{-}100)/(6*\sigma)$. Solving for σ, gives $\sigma = (50)/(6*2) = 50/12 \approx 4.167$. Therefore, $var(\varepsilon_1) = \sigma^2 \approx 17.3611$.

A similar procedure is applicable for Y_2; however, to have different error distributions to each response, we decided that the error random variable added must have $E(\varepsilon_2) = 0$, never make Y_2 a negative value and $C_p(\hat{Y}_2) = 2$. Thus, we have for Y_2 that $var(\hat{Y}_2) = var(\varepsilon_2) = ((USL\text{-}LSL)/12)^2 = ((11\text{-}0)/12)^2 \approx 0.8403$. For the error added to Y_2 we want a random variable that satisfies the conditions: a) $E(\varepsilon_2) = 0$; b) $Y_2 + \varepsilon_2 \geq 0$; and c) $var(\hat{Y}_2) = ((USL\text{-}LSL)/12)^2$. The 3-parameter Weibull distribution with CDF: $F(x) = 1\text{-}\exp\{[(x - \theta)/\beta]^{\alpha}\}$ will be used. Condition (b) must be true always. Since ε_2 is a random number that could be negative by condition (a), we must ensure that the sum of most negative random number that could arise from the selected Weibull distribution (that is, its location parameter, θ) plus the minimum possible value of Y_2 results in a non-negative value. Finding the minimum Y_2 in the stated range is a trivial problem using Eq. 3.2. Parameters are found by solving the simultaneous system of equations defined by conditions (a), (b) and (c) above: $E(\varepsilon_2) = \beta*\Gamma\{(\alpha+1)/\alpha\} + \theta = 0$ and $var(\varepsilon_2) = \beta^2*\Gamma\{(\alpha+2)/\alpha\} - [\beta*\Gamma\{(\alpha+1)/\alpha\}]^2 = ((USL\text{-}LSL)/12)^2 \approx 0.8403$ with the MIN $Y_2 = 0.612727977$, found by solving the constrained minimization Eq. 3.2 subject to Eq. 3.3; then, $\theta = MIN(\varepsilon_2) \geq -0.612727977$ (= - MIN Y_2). Using Mathcad, a solution found for this system is $F(x) = 1\text{-}\exp\{[(x + 0.247758)/0.056116]^{0.363836}\}$. This is just one of many possible solutions to this system. The

model for the first response is $Y_1 + \varepsilon_1$, $\varepsilon_1 \sim N(0, \sigma^2 \approx 17.3611)$ and the model for the second response is $Y_2 + \varepsilon_2$, $\varepsilon_2 \sim$ Weibull($\theta = -0.247758$, $\beta = 0.056116$, $\alpha = 0.363836$). Table 3.1 shows a summary of these results. See Appendix A, Section III, "Calculation of the random variable for Y2".

Table 3.1 Error Distributions for Random Variables Y_1 and Y_2 for Case Study 1

Distributions	**E(ε)**	**var(ε)**	**θ= MIN(ε_2)**	**α**	**β**
Y_1 Response: Normal NIID(0,σ^2)	0	17.36	-	-	-
Y_2 Response: 3-Parameter Weibull	0	0.840277	-0.247758	0.363836	0.056116

Table 3.2 shows details for this case. It summarizes the results for the three approaches by pointing out the distributions means, standard deviation, minimum and maximum as well as its percentiles. The table includes the actual optimums values for this problem; the results for proposed methods will be compared against these values. From Table 3.2 the closest mean values with the minimum variability for optimum values of the input variables were those found by method R1. It follows method R1 also has the minimum mean distance and distance variability, minimum mean loss and loss variability. Mean values for the responses are also the closest to the optimum responses with the minimum variability. Methodology R2 mean values for input variables were farther with their variability twice or three times R1's. This situation results in mean values for the distance and its variability as much as three times R1. Loss mean value was larger with a variability of almost 10 times R1. Mean values for the responses were farther with variability as much as twice R1. The neural

net results were the worst. The neural net mean values for the inputs variables were the farther from the optimum and their variability's were about twenty times R1's. This much variation resulted in mean values for the distance and its variability as high as twenty times. The neural network loss mean value was the larger with a variability of about 300 times R1. Mean values for the responses were the farther with variability as high as 10 times R1.

Table 3.2 Case Study 1: Summary of Results
Based on 1,000 Simulations, Sample Size n= 5

CHARACTERISTIC	X_1*	X_2*	X_3*	Distance	Loss	Y_1*	Y_2*
OPTIMUM	**0.3284**	**-1.2327**	**1.6818**	**0.0000**	**2.8706**	**130.5976**	**8.2834**
Responses Lower Specification Limits (LSL) and Inputs Ranges	**-1.68179**	**-1.68179**	**-1.68179**	**-**	**-**	**100**	**0**
Response Upper Specification Limits (USL) and Inputs Ranges	**1.68179**	**1.68179**	**1.68179**	**-**	**-**	**150**	**11**
Target	**-**	**-**	**-**	**-**	**-**	**150**	**0**
Regression Method 1 R1 (Formal Optimization)	X_1*	X_2*	X_3*	Distance	Loss	Y_1*	Y_2*
Mean	0.3351	-1.2235	1.6606	0.0885	2.8858	130.1338	8.2505
Std Dev	0.0941	0.0321	0.0471	0.0695	0.0189	1.4129	0.1604
Minimum	0.0199	-1.3091	1.2721	0.0025	2.8706	124.2123	7.6857
Maximum	0.7166	-1.0174	1.6818	0.6041	3.0715	134.0208	8.7271
Percentile 5%	0.1959	-1.2705	1.5583	0.0186	2.8712	127.7605	7.9868
Percentile 10%	0.2246	-1.2617	1.5969	0.0251	2.8718	128.3005	8.0461
Percentile 20%	0.2606	-1.2488	1.6490	0.0356	2.8730	128.9643	8.1144
Percentile 25%	0.2737	-1.2451	1.6691	0.0412	2.8739	129.2529	8.1434
Percentile 33%	0.2911	-1.2385	1.6818	0.0509	2.8753	129.5551	8.1805
Percentile 50%	0.3267	-1.2261	1.6818	0.0704	2.8795	130.2068	8.2465
Percentile 67%	0.3662	-1.2130	1.6818	0.0983	2.8860	130.7869	8.3198
Percentile 75%	0.3908	-1.2057	1.6818	0.1124	2.8902	131.1206	8.3646
Percentile 80%	0.4087	-1.2004	1.6818	0.1259	2.8942	131.3324	8.3864
Percentile 90%	0.4597	-1.1837	1.6818	0.1735	2.9076	131.8959	8.4577
Percentile 95%	0.5087	-1.1697	1.6818	0.2249	2.9193	132.3194	8.5116
Regression Method 2 R2 (Random Search)	X_1*	X_2*	X_3*	Distance	Loss	Y_1*	Y_2*
Mean	0.4421	-1.1805	1.5215	0.2953	2.9986	128.4539	8.2032
Std Dev	0.1877	0.1388	0.1334	0.1624	0.2004	3.9644	0.4325

CHARACTERISTIC	X_1*	X_2*	X_3*	Distance	Loss	Y_1*	Y_2*
OPTIMUM	**0.3284**	**-1.2327**	**1.6818**	**0.0000**	**2.8706**	**130.5976**	**8.2834**
Responses Lower Specification Limits (LSL) and Inputs Ranges	**-1.68179**	**-1.68179**	**-1.68179**	**-**	**-**	**100**	**0**
Response Upper Specification Limits (USL) and Inputs Ranges	**1.68179**	**1.68179**	**1.68179**	**-**	**-**	**150**	**11**
Target	**-**	**-**	**-**	**-**	**-**	**150**	**0**
Minimum	-0.1692	-1.5365	0.1665	0.0223	2.8720	90.5258	6.5005
Maximum	1.0288	0.1882	1.6818	2.0688	7.0564	138.5892	9.4763
Percentile 5%	0.1191	-1.3915	1.2839	0.1035	2.8916	122.4361	7.5102
Percentile 10%	0.1950	-1.3504	1.3595	0.1331	2.9032	123.8727	7.6452
Percentile 20%	0.2783	-1.2943	1.4347	0.1741	2.9199	125.4700	7.8485
Percentile 25%	0.3201	-1.2711	1.4584	0.1904	2.9290	126.1025	7.9167
Percentile 33%	0.3656	-1.2376	1.4923	0.2173	2.9429	126.9069	8.0011
Percentile 50%	0.4513	-1.1787	1.5495	0.2701	2.9721	128.4965	8.1982
Percentile 67%	0.5307	-1.1196	1.5987	0.3294	3.0084	130.2751	8.3952
Percentile 75%	0.5740	-1.0924	1.6163	0.3695	3.0328	131.111	8.5073
Percentile 80%	0.5959	-1.0763	1.6285	0.3971	3.0482	131.7434	8.5767
Percentile 90%	0.6823	-1.0214	1.6524	0.4846	3.1020	133.0859	8.7279
Percentile 95%	0.7371	-0.9801	1.6676	0.5609	3.1568	134.1857	8.9018
Artificial Neural Network ANN (Random Search)	**X_1***	**X_2***	**X_3***	**Distance**	**Loss**	**Y_1***	**Y_2***
Mean	0.5304	-0.7041	0.6345	1.6931	6.9490	107.0538	8.7216
Std Dev	0.9300	0.7090	0.9308	0.8848	3.7182	26.2166	3.4136
Minimum	-1.6765	-1.6811	-1.6815	0.1138	2.8926	32.6975	1.5702
Maximum	1.6800	1.6380	1.6810	4.3798	36.0076	157.3172	30.3249
Percentile 5%	0.1191	-1.6264	-1.2819	0.5159	3.3874	63.9620	3.5389
Percentile 10%	-0.8070	-1.5647	-0.8011	0.6988	3.6863	71.7075	4.4164
Percentile 20%	-0.3896	-1.4185	-0.1926	0.9485	4.2303	83.0535	5.9386
Percentile 25%	-0.2152	-1.3460	-0.0134	1.0716	4.4715	87.5501	6.5891
Percentile 33%	0.1343	-1.1033	0.3720	1.2270	4.8282	93.0905	7.2166
Percentile 50%	0.7228	-0.7502	0.8951	1.4855	5.6631	107.2558	8.4296
Percentile 67%	1.1869	-0.4682	1.3214	1.8928	7.2684	119.6361	9.9263
Percentile 75%	1.3971	-0.2978	1.4639	2.1599	8.5673	127.3310	10.9320
Percentile 80%	1.4755	-0.1737	1.5205	2.3944	9.3547	133.7662	11.5173
Percentile 90%	1.6034	0.2642	1.6122	3.0442	11.7030	143.1498	12.6958
Percentile 95%	1.6426	0.6516	1.6495	3.4942	14.2233	148.5200	13.8525

Figure 3.1 shows the distributions of the Euclidean distance from optimum for Case Problem 1 using all methods. The figure shows that the regression search method R(M1) was the method that produced distances closer to zero; 999 cases out of 1,000 were in the 0.0 to 0.5 interval compared to 914 out of 1,000 cases for the regression search method R(M2) and only 44 cases out of 1,000 for the neural net.

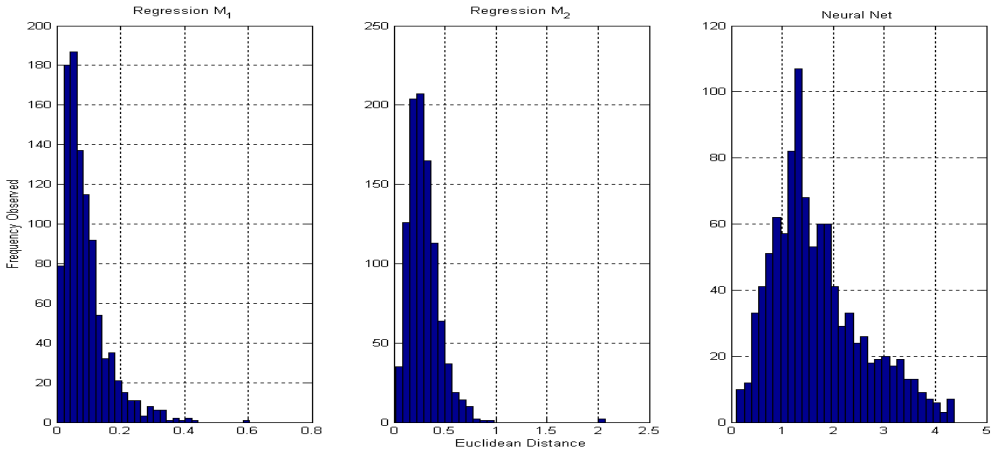

Figure 3.1. Euclidean distances from optimum for each of the three methods.

Figure 3.2 shows the scatter plot for the regression search method R(M1): (X_{R1}^*, X_{R12}^*, X_{R13}^*) and its location related to the true optimum (X_1^*, X_2^*, X_3^*) = (0.3284, -1.2327, 1.6818). Maximum distances give us some insight about the methodology's performance. Maximum distances are from X_1^* to X_{R11}^*: 0.3882 units, from X_2^* to X_{R12}^*: 0.2153 units and from X_3^* to X_{R13}^*: 0.4097 units.

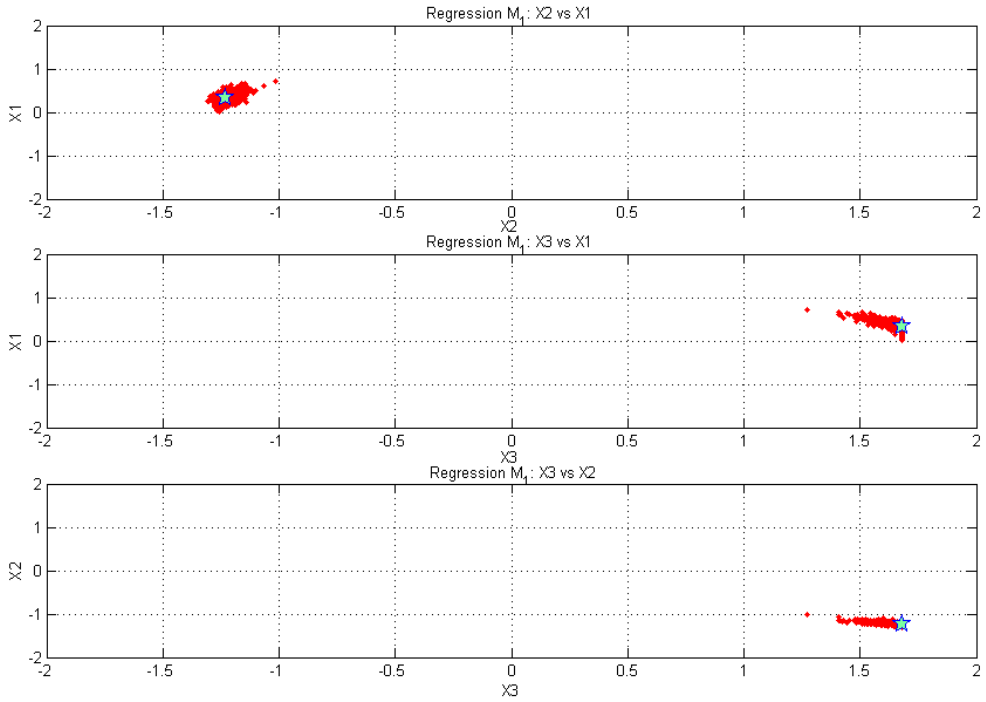

Figure 3.2. Scatter plot for regression search method R(M1)
(Optimum point starred for comparison)

Figure 3.3 shows the scatter plot for the regression search method R(M2): (X_{R21}^*, X_{R22}^*, X_{R23}^*) and its location related to the true optimum (X_1^*, X_2^*, X_3^*) = (0.3284, -1.2327, 1.6818). Maximum distances are from X_1^* to X_{R21}^*: 0.7004 units, from X_2^* to X_{R22}^*: 1.4209 units and from X_3^* to X_{R23}^*: 1.5153 units which are between 1.8 and 6.8 times larger than those of R1's.

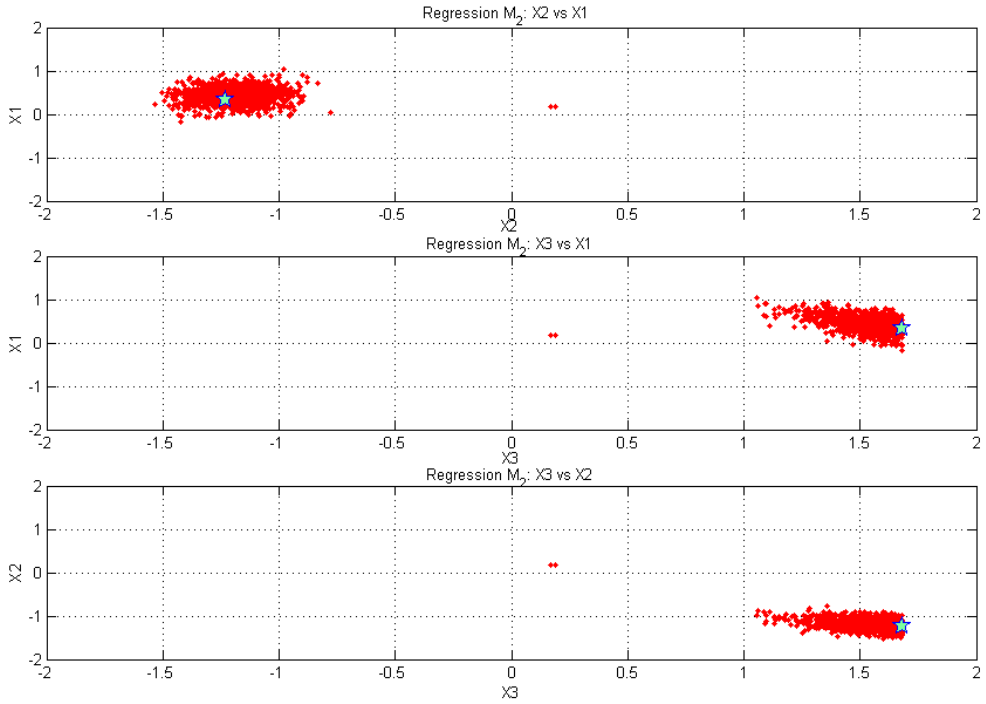

Figure 3.3. Scatter plot for regression search method R(M2)
(Optimum point starred for comparison)

Figure 3.4 shows the scatter plot between the sets found by the ANN (X_{N1}^*, X_{N2}^*, X_{N3}^*) and its location related to the true optimum (X_1^*, X_2^*, X_3^*) = (0.3284, -1.2327, 1.6818). It is worth noting it shows a diffusion tendency toward the location of the true optimum. Maximum distances are from X_1^* to X_{N1}^*: 2.0049 units, from X_2^* to X_{N2}^*: 2.8707 units and from X_3^* to X_{N3}^*: 3.3633 units which are between 5.1 and 13 times larger than those of R1's.

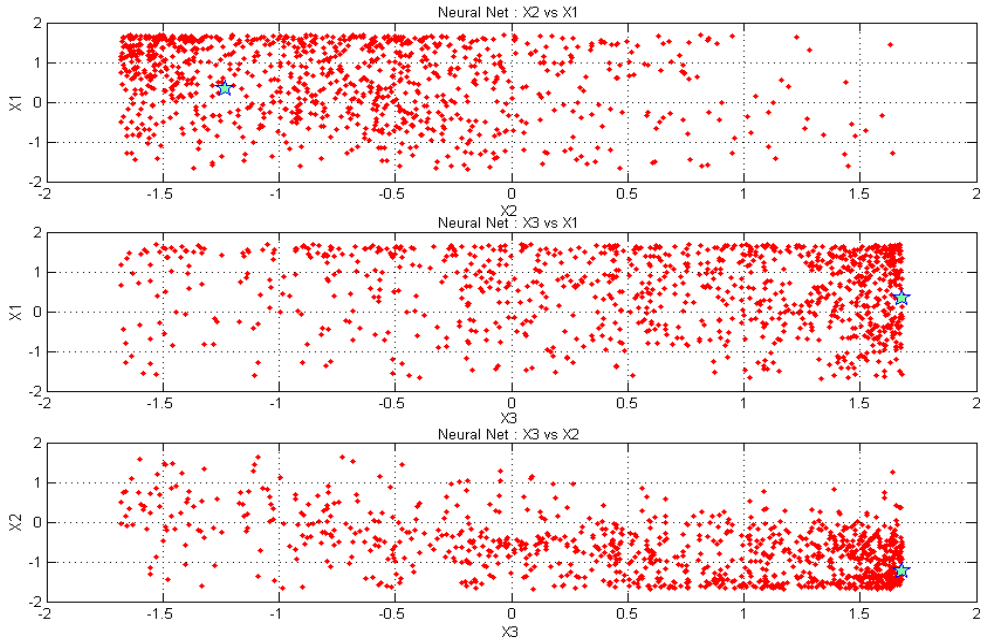

Figure 3.4 Scatter plot for neural network search method ANN
(Optimum point starred for comparison)

Figure 3.5 shows the distributions for the response Y_1^*. The [128, 132] interval contained 845 R(M1) cases, 370 R(M2) cases, and 26 cases for the neural net.

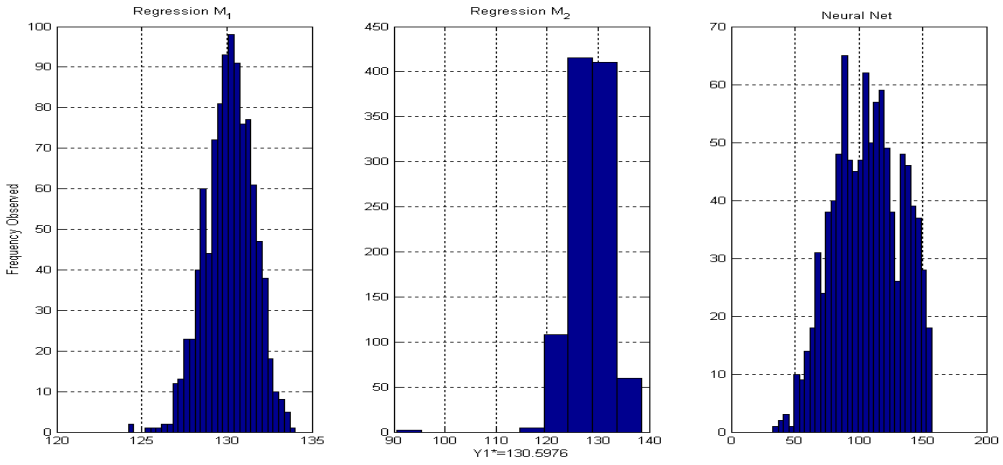

Figure 3.5 Distributions for the Y_1* Response

Figure 3.6 shows the distribution for the response Y_2^*. The [7.5, 8.5] interval contained 943 R(M1) cases, 697 R(M2) cases, and 136 cases for the neural net.

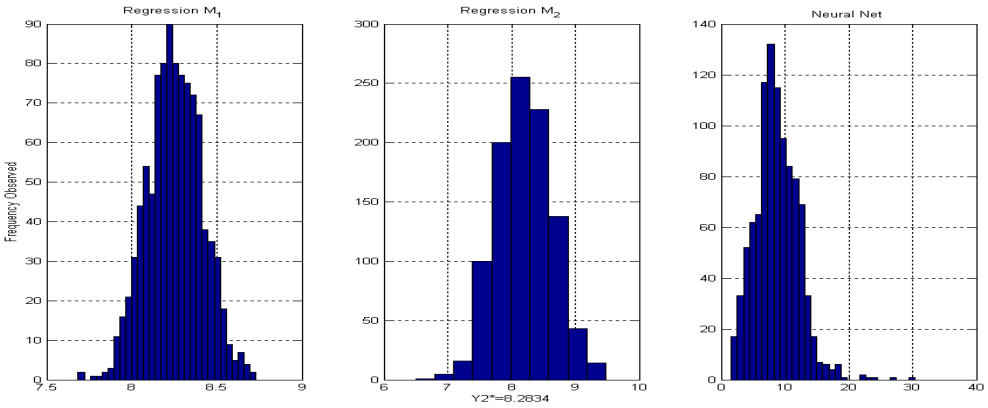

Figure 3.6 Distributions for the Y_2* Response.

The regression method R(M1) outperformed all other methodologies. From Figures 3.1, 3.2, 3.5, & 3.6 it is easily observed this approach consistently found the closest solution to the optimum. The regression method R(M2) results were similar to R(M1). Comparing Figures 3.2 and 3.3 and the maximum distances for both methodologies, it could be inferred the "brute force" methodology is not as effective as Lagrange optimization but is not as ineffective as to render the R2 method useless. The neural network ANN methodology was very ineffective in finding the optimum, its maximum distances were twice the R(M2) distances, which used the same brute force "optimization".

The effects on the input can be observed in Figures 3.5 and 3.6. The regression R(M1) method found the best possible distributions for the responses. We might infer that the brute force methodology is an unacceptable method to find optimum solutions but not the main cause of the failure of the neural network approach.

3.3 Case Problem. 2: Linear and Quadratic Responses

The chemical experiment from Del Castillo (1996) is an example to study the behavior of a system with linear and quadratic responses. In this problem, control (input) variables X_1 and X_2 were the temperature, and the reaction time, respectively. The output (responses) variables, Y_1 and Y_2, were the process yield and its standard deviation (σ) respectively. This case's adjusted equations as reported by Del Castillo are:

$$Y_1 = 760.5 + 50.9X_1 + 154.8X_2 - 4.4X_1^2 - 76X_2^2 - 24.7X_1X_2 \qquad 3.4$$

and,

$$Y_2 = 19.6 + 6.31X_1 + 6.28X_2 \qquad 3.5$$

where the input vector **X** is bounded between:

$$-1.414 \le Xj \le 1.414 \text{ for } j = 1,2 \qquad 3.6$$

The methodology is analogous to the technique used for Case Problem Number 1. For this case, as with the first one, Del Castillo wanted to maximize the Y_1 and minimize the Y_2. We will use the same criteria for each response. As we did in Case Problem 1, the output specification limits came from Del Castillo example's results. Del Castillo's sensitivity analysis upon Y_1 resulted in a minimum of 848.57 and a maximum of 863.83. To maximize Y_1 seems reasonable to establish the specification limits for such variable as [800 to 950]. The idea is to provide an area big enough that includes the author's findings. For Y_2 we

established specification limits from [0, 35], which were selected from the sensitivity analysis done by Del Castillo's. See Appendix B, Section I.

We need to simulate the CCD arrangement used for the sampling process, the values of the lower and upper limits for the inputs are given as (-α, α) where α = [number of factorial runs] $^{0.25}$; in this case, we are using a full factorial with two factors and two levels, therefore $|\alpha| = [2^2]^{0.25} = 4^{0.25}$, $|\alpha| \approx 1.414$. One thousand simulations runs were done, each a sample size, n, of five (5). See Appendix B, Section III.

An important consideration for both responses is that we want to select the value of its individual variance, σ_i^2 so that the potential process capability for each response equals 2. , We added Y_1, plus the random error, ε_1, modeled by a normal distribution, with mean = 0 and variance = σ^2. The new random variable has mean = $E(Y_1 + \varepsilon_1) = Y_1 + 0 = Y_1$ and, variance= var $(Y_1 + \varepsilon_1) = 0 + \sigma^2 = \sigma^2$ since Y_1 is a constant term. Then, $C_p(\hat{Y}_1) = \frac{(USL - LSL)}{6 * STDDEV(\hat{Y}_1)} = 2 = (950\text{-}800)/(6*\sigma)$. Solving for σ, gives σ = (150)/(6*2)= 150/12 = 12.50. Therefore, var(ε_1) = σ^2 = 156.25.

A similar procedure is applicable for Y_2; however, to have different error distributions to each response, we decided that random variable added must have $E(\varepsilon_2) = 0$, never make Y_2 a negative value and $C_p(\hat{Y}_2) = 2$.. Thus, we have for Y_2 that var $(\hat{Y}_2) = \text{var}(\varepsilon_2)$ = ((USL-LSL)/12)^2 = ((35-0)/12)^2 ≈ 8.506944. For the error added to Y_2 we want a random variable that satisfies the conditions: a) $E(\varepsilon_2) = 0$; b) $Y_2 + \varepsilon_2 \geq 0$; and c) var $(\hat{Y}_2)$ = ((USL-

LSL)/12)^2. The 3-parameter Weibull distribution with CDF: $F(x) = 1-\exp\{[(x - \theta)/\beta]^{\alpha}\}$ will be used. Condition (b) must be true always. Since ε_2 is a random number that could be negative by condition (a), we must ensure that the sum of most negative random number that could arise from the selected Weibull distribution (that is, its location parameter, θ) plus the minimum possible value of Y_2 results in a non-negative value. Finding the minimum Y_2 in the stated range is a trivial problem using Eq. 3.5. Parameters are found by solving the simultaneous system of equations defined by conditions (a), (b) and (c) above: $E(\varepsilon_2) = \beta*\Gamma\{(\alpha+1)/\alpha\} + \theta = 0$ and $var(\varepsilon_2) = \beta^2*\Gamma\{(\alpha+2)/\alpha\} - [\beta*\Gamma\{(\alpha+1)/\alpha\}]^2 = ((USL-LSL)/12)^2 \approx 8.506944$ with the MIN $Y_2 = 1.79774$ found by solving the constrained minimization Eq. 3.5 subject to Eq. 3.6; then, $\theta = MIN(\varepsilon_2) \geq -1.79774$ (= - MIN Y_2). Using Mathcad, a solution found for this system is $F(x) = 1-\exp\{[(x +1.79774)/1.294657]^{0.640724}\}$. This is just one of many possible solutions to this system. The model for the first response is $Y_1 + \varepsilon_1$, $\varepsilon_1 \sim N(0,\sigma^2 \approx 156.25)$ and the model for the second response is is $Y_2 + \varepsilon_2$, $\varepsilon_2 \sim$ Weibull(θ=-1.79774, β = 1.294657, α = 0.640724). Table 3.3 shows a summary of these results. See Appendix B, Section III "Calculation of the random variable for Y2".

Table 3.3 Error Distributions for Random Variables Y_1 and Y_2 for Case Study 2

Distributions	**E(ε)**	**var(ε)**	**θ= MIN(ε_2)**	**α**	**β**
Y_1 Response: Normal NIID(0,σ^2)	0	156.25	-	-	-
Y_2 Response: 3-Parameter Weibull	0	8.506929	-1.79774	0.64072	1.294657

Table 3.4 shows the characteristics' details for this case and its three methodologies by pointing out the distributions means, standard deviation, minimum and maximum as well as its percentiles. The table includes the actual optimums values for this problem; the results for the proposed methods will be compared against these values From Table 3.4 the closest mean values with the minimum variability for optimum values of the input variables were those found by method R2. It follows method R2 also has the minimum mean distance and distance variability, minimum mean loss and loss variability. Mean value for the Y_1 response was also the closest to the optimum responses with the minimum variability. Methodology R2 mean values for input variables were farther with their variability very close to R1's. This situation results in mean values for the distance and its variability are also very close to R2. Loss mean value was larger with a variability that is also very close to R2. Mean value for the Y_2 response was also the closest to the optimum responses with the variability close to R2. The neural net results were the worst, in general.. The neural net mean values for the inputs variables were between the two regressions methodologies, in the case of X_1, and the farther from the optimum in the case of X_2 and their variability's were about one and a half times R2's. This variation resulted in mean values for the distance as high as twice R2 but with less variation. The neural network loss mean value was the larger with a variability of about 6 times R2. Mean values for the responses were the farther, in the case of Y_1, with variability as about twice R2 and, for Y_2 almost as good as R2 but with double variability.

Table 3.4 Case Study 2: Summary of Results
Based on 1,000 Simulations, Sample Size n= 5

CHARACTERISTIC	X_1*	X_2*	Distance	Loss	Y_1*	Y_2*
OPTIMUM	**0.3064**	**0.779**	**0.0000**	**4.268**	**844.2564**	**26.4255**
Responses Lower Specification Limits (LSL) and Inputs Ranges)	**-1.414**	**-1.414**	**-**	**-**	**800**	**0**
Response Upper Specification Limits (USL) and Inputs Ranges	**1.414**	**1.414**	**-**	**-**	**950**	**35**
Target	**-**	**-**	**-**	**-**	**950**	**0**
Regression Method 1 R1 (Formal Optimization)	**X_1***	**X_2***	**Distance**	**Loss**	**Y_1***	**Y_2***
Mean	0.0602	0.8447	0.7231	4.4856	836.5413	25.2846
Std Dev	0.7811	0.2205	1.1189	0.3422	19.4075	3.6155
Minimum	-1.4140	0.3249	0.0001	4.2682	794.6082	17.1630
Maximum	1.4140	1.4140	3.3630	5.4625	865.8455	31.8419
Percentile 5%	-1.4140	0.5424	0.0025	4.2703	797.6138	18.2189
Percentile 10%	-1.4140	0.6069	0.0065	4.2730	798.0162	18.6106
Percentile 20%	-0.4815	0.6778	0.0195	4.2798	825.0904	22.7331
Percentile 25%	-0.2377	0.7050	0.0279	4.2833	831.0099	23.8004
Percentile 33%	-0.0084	0.7411	0.0525	4.2908	836.6620	24.8849
Percentile 50%	0.2359	0.7972	0.1476	4.3164	842.1887	26.0174
Percentile 67%	0.4376	0.8756	0.4047	4.3809	846.8253	27.0907
Percentile 75%	0.5611	0.9384	0.7436	4.4554	849.7889	27.7205
Percentile 80%	0.6609	1.0162	1.2202	4.5732	851.5140	28.1124
Percentile 90%	0.9246	1.2111	3.1426	5.1995	856.3326	29.3852
Percentile 95%	1.1430	1.2802	3.2090	5.2475	859.0569	30.3304
Regression Method 2 R2 (Random Search)	**X_1***	**X_2***	**Distance**	**Loss**	**Y_1***	**Y_2***
Mean	0.1618	0.8174	0.6249	4.4550	838.9031	25.7542
Std Dev	0.7455	0.2176	0.9704	0.2952	17.8900	3.4180
Minimum	-1.4140	0.2385	0.0001	4.2681	794.7462	17.1870
Maximum	1.4138	1.4074	3.3237	5.4303	865.8487	31.6596
Percentile 5%	-1.3854	0.4956	0.0033	4.2707	798.7400	18.6661
Percentile 10%	-1.3266	0.5681	0.0072	4.2738	800.6000	19.3170
Percentile 20%	-0.3011	0.6561	0.0190	4.2798	829.6970	23.6122
Percentile 25%	-0.1400	0.6869	0.0287	4.2836	833.5996	24.3506
Percentile 33%	0.0636	0.7266	0.0510	4.2915	838.3161	25.2097
Percentile 50%	0.2694	0.7863	0.1498	4.3198	842.8580	26.1534
Percentile 67%	0.4896	0.8568	0.3937	4.3796	848.2370	27.3544

CHARACTERISTIC	X_1*	X_2*	Distance	Loss	Y_1*	Y_2*
OPTIMUM	**0.3064**	**0.779**	**0.0000**	**4.268**	**844.2564**	**26.4255**
Responses Lower Specification Limits (LSL) and Inputs Ranges)	**-1.414**	**-1.414**	**-**	**-**	**800**	**0**
Response Upper Specification Limits (USL) and Inputs Ranges	**1.414**	**1.414**	**-**	**-**	**950**	**35**
Target	**-**	**-**	**-**	**-**	**950**	**0**
Percentile 75%	0.6387	0.9088	0.7096	4.4504	850.8877	28.0293
Percentile 80%	0.7464	0.9486	1.0435	4.5285	852.4687	28.4755
Percentile 90%	1.0334	1.1975	2.8358	5.1323	857.5086	29.9657
Percentile 95%	1.3030	1.2679	3.0702	5.1934	860.0910	30.6173
Artificial Neural Network ANN (Random Search)	X_1*	X_2*	Distance	Loss	Y_1*	Y_2*
Mean	0.0961	0.8813	1.3086	5.5816	819.2528	25.7413
Std Dev	1.0020	0.5011	0.9506	1.8385	36.5995	6.5743
Minimum	-1.4140	-1.0445	0.0008	4.2687	515.1320	6.4061
Maximum	1.4137	1.4130	5.0881	33.7536	870.9019	37.3336
Percentile 5%	-1.3854	0.0761	0.1047	4.3654	759.3797	15.8563
Percentile 10%	-1.3599	0.2391	0.2127	4.5032	777.3256	17.7961
Percentile 20%	-1.0078	0.4009	0.4195	4.6277	797.6755	19.8982
Percentile 25%	-0.8027	0.4829	0.5227	4.7408	799.9799	20.5749
Percentile 33%	-0.5283	0.6022	0.7591	4.9811	807.5432	21.9954
Percentile 50%	0.1503	0.9692	1.1944	5.2193	821.5790	24.3466
Percentile 67%	0.7774	1.3304	1.4325	5.5199	840.6954	29.3901
Percentile 75%	1.1484	1.3665	1.7000	5.8994	844.3465	31.5305
Percentile 80%	1.2930	1.3780	2.1724	6.1154	847.2961	32.8001
Percentile 90%	1.3864	1.3998	2.9286	6.4988	857.3638	35.6805
Percentile 95%	1.4033	1.4068	3.1263	7.4955	865.8818	36.5146

Figure 3.7 shows the distributions of the Euclidean distance from optimum for Case Problem 2 using all methods. The figure shows both regression methods produced similar distances; 697 cases out of 1,000 were in the 0.0 to 0.5 interval for the regression search method R(M1) compared to 705 out of 1,000 cases for the regression search method R(M2) and only 242 cases out of 1,000 for the neural net.

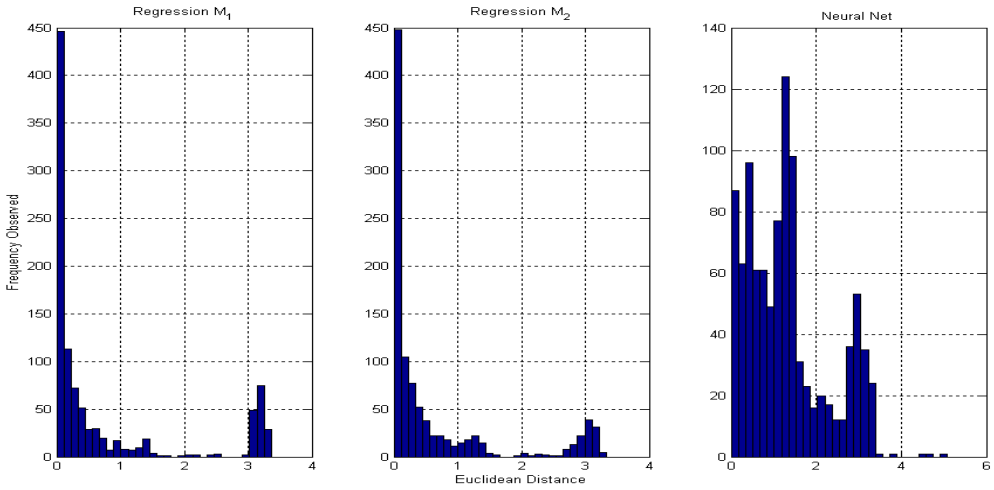

Figure 3.7. Euclidean distances from optimum for each of the three methods.

Figure 3.8 shows the scatter plot for all three approaches and their location related to the true optimum $(X_1^*, X_2^*) = (0.3064, 0.7790)$. For the regression search method R(M1): (X_{R11}^*, X_{R12}^*) maximum distances give us some insight about the methodology's performance. Maximum distances are from X_1^* to X_{R11}^*: 1.7204 units, from and X_2^* to X_{R12}^*: 0.6350 units which are between 1 and 1.01 times larger than those of R2's. For the regression search method R(M2): (X_{R21}^*, X_{R22}^*) maximum distances are from X_1^* to X_{R21}^*: 1.7204 units and from X_2^* to X_{R22}^*: 0.6284 units. For the neural net (X_{N1}^*, X_{N2}^*), maximum distances are from X_1^* to X_{N1}^*: 1.7204 units and from X_2^* to X_{N2}^*: 1.8235 units which are between 1 and 2.9 times larger than those of R2's.

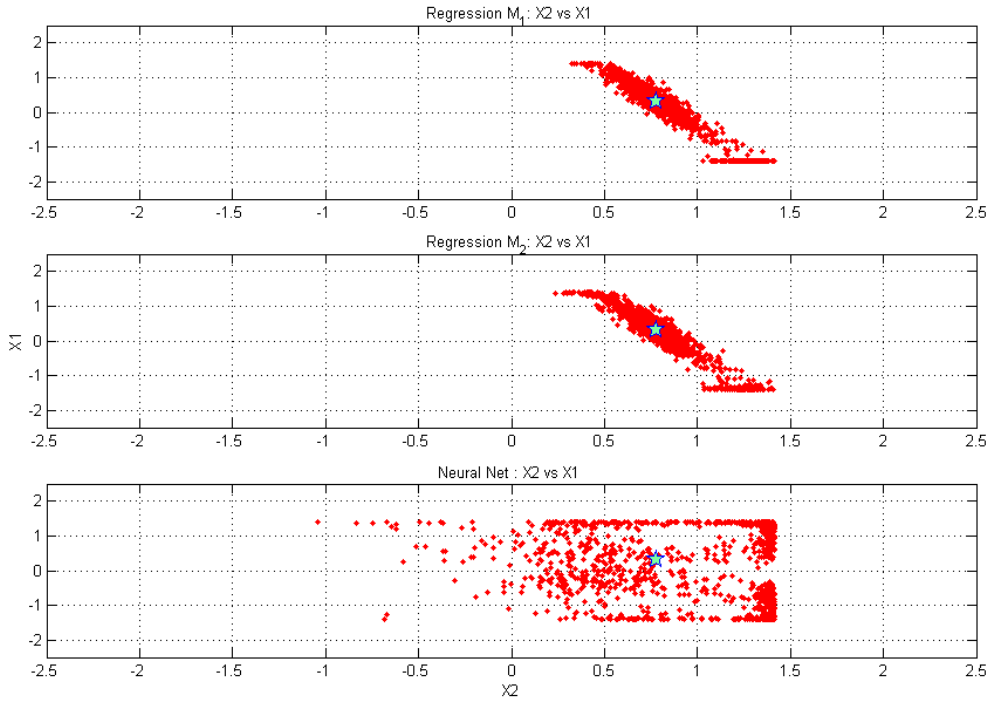

Figure 3.8. Scatter plot for all three approaches
(Optimum point starred for comparison)

Figure 3.9 shows the distributions for the response Y_1^*. The [840, 850] interval contained 334 R(M1) cases, 343 R(M2) cases, and 179 cases for the neural net.

Figure 3.10 shows the distribution for the response Y_2^*. The [25, 27] interval contained 312 R(M1) cases, 305 R(M2) cases, and 69 cases for the neural net. The regression methods gave substantially similar results.

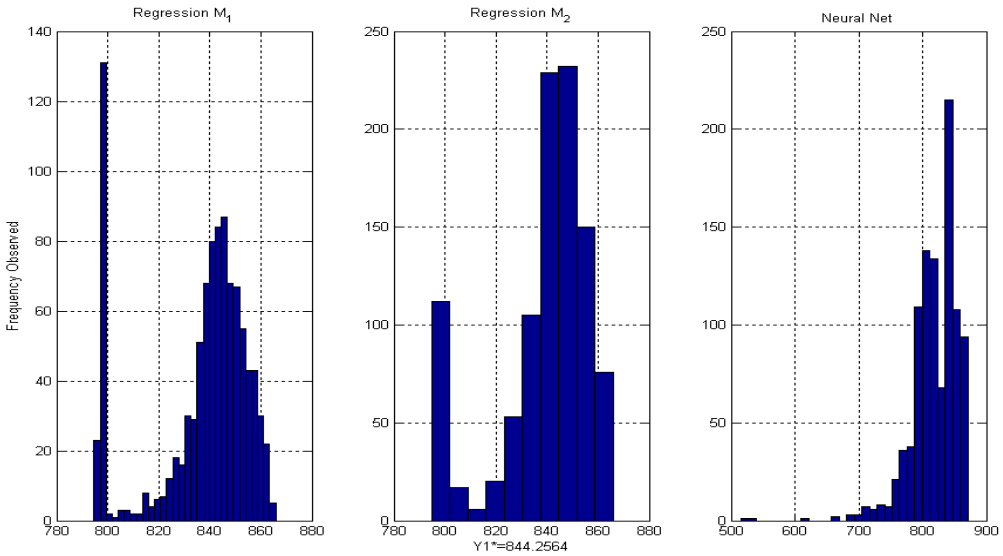

Figure 3.9 Distributions for the Y_1* Response

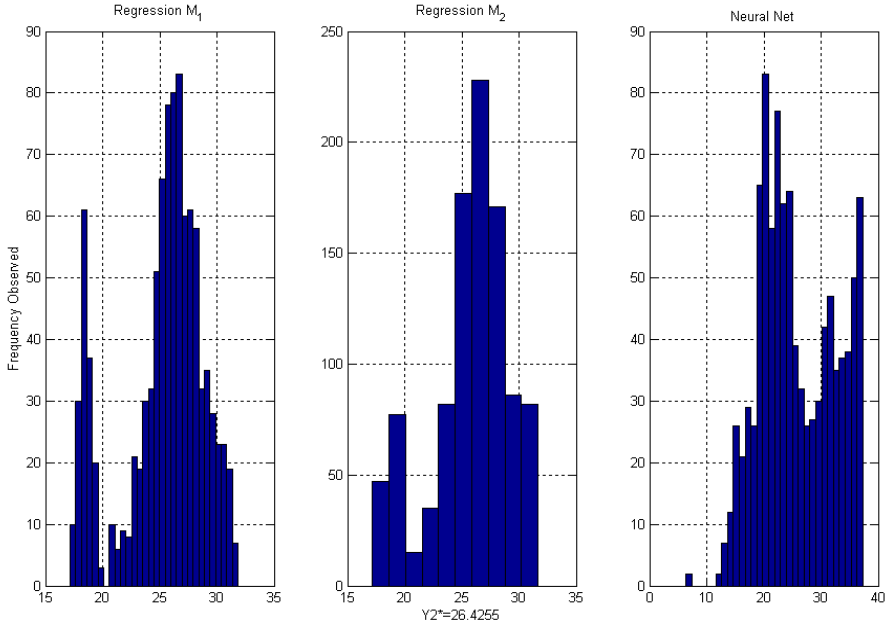

Figure 3.10 Distributions for the Y_2* Response.

From Figures 3.7 & 3.8, it can be observed that both approaches were almost identical. It might be hypothesized that the brute force was as effective as the Lagrange optimization. Results from the neural network were dispersing, particularly for the X_2 input. From Figure 3.9 we see that regression distributions for the Y_1 response were practically the same and the neural net was the method with the biggest offset from the mean; dispersion for the regressions were also similar with the neural net doubling it. From Figure 3.10 we see that regression distributions for the Y_2 response were also similar. All methodologies' means were similar; dispersions for the regressions were also similar with the neural net doubling it.

We might infer that the brute force methodology is an acceptable method to find optimum solutions.

3.4 Case Problem 3: All Linear Responses

The chemical experiment from Del Castilllo (1996) with appropriate modifications is the example to study the behavior of a system when both responses are linear equations. In this problem, as in case 2, the control (input) variables X_1 and X_2 were the temperature, and the reaction time respectively. The output (responses) variables, Y_1 and Y_2, were the process yield and its standard deviation (σ) respectively. This case's adjusted equations as reported by Del Castillo are:

$$Y_1 = 711 + 50.9X_1 + 154.8X_2 \qquad 3.7$$

and,

$$Y_2 = 19.26 + 6.31X_1 + 6.28X_2 \qquad 3.8$$

where the input vector **X** is bounded between:

$$-1.414 \le X_j \le 1.414 \text{ for } j = 1,2 \qquad 3.9$$

The methodology is analogous to the technique used for Case Problems Number 1 & 2. The case upper and lower limits were selected following the same procedure as Case 2. See Appendix C, Section I.

An important consideration for both responses is that we want to select the value of its individual variance, σ_i^2 so that the potential process capability for each response equals 2. We added Y_1, plus the random error, ε_1, modeled by a normal distribution, with mean = 0 and variance = σ^2. The new random variable has mean = $E(Y_1 + \varepsilon_1) = Y_1 + 0 = Y_1$ and, variance= var $(Y_1 + \varepsilon_1) = 0 + \sigma^2 = \sigma^2$ since Y_1 is a constant term. Then, $C_p(\hat{Y}_1) = \frac{(USL - LSL)}{6 * STDDEV(\hat{Y}_1)} = 2 = (950\text{-}800)/(6*\sigma)$. Solving for σ, gives $\sigma = (150)/(6*2) = 150/12 = 12.50$. Therefore, $\text{var}(\varepsilon_1) = \sigma^2 = 156.25$.

A similar procedure is applicable for Y_2; however, to have different error distributions to each response, we decided that the error random variable added must have $E(\varepsilon_2) = 0$,

never make Y_2 a negative value and $C_p(\hat{Y}_2) = 2$. Thus, we have for Y_2 that $\text{var}(\hat{Y}_2) = \text{var}(\varepsilon_2)$ $= ((\text{USL-LSL})/12)^2 = ((35-0)/12)^2 \approx 8.506944$. For the error added to Y_2 we want a random variable that satisfies the conditions: a) $E(\varepsilon_2) = 0$; b) $Y_2 + \varepsilon_2 \geq 0$; and c) $\text{var}(\hat{Y}_2) = ((\text{USL-LSL})/12)^2$. The 3-parameter Weibull distribution with CDF: $F(x) = 1-\exp\{[(x - \theta)/\beta]^{\alpha}\}$ will be used. Condition (b) must be true always. Since ε_2 is a random number that could be negative by condition (a), we must ensure that the sum of most negative random number that could arise from the selected Weibull distribution (that is, its location parameter, θ) plus the minimum possible value of Y_2 results in a non-negative value. Finding the minimum Y_2 in the stated range is a trivial problem using Eq. 3.8. Parameters are found by solving the simultaneous system of equations defined by conditions (a), (b) and (c) above: $E(\varepsilon_2) = \beta*\Gamma\{(\alpha+1)/\alpha\} + \theta = 0$ and $\text{var}(\varepsilon_2) = \beta^2*\Gamma\{(\alpha+2)/\alpha\} - [\beta*\Gamma\{(\alpha+1)/\alpha\}]^2 = ((\text{USL-LSL})/12)^2$ ≈ 8.506944 with the MIN $Y_2 = 1.45774$ found by solving the constrained minimization Eq. 3.8 subject to Eq. 3.9; then, $\theta = \text{MIN}(\varepsilon_2) \geq -1.45774$ (= - MIN Y_2). Using Mathcad, a solution found for this system is $F(x) = 1-\exp\{[(x +1.45774)/0.838157]^{0.542526}\}$. This is just one of many possible solutions to this system. The model for the first response is $= Y_1 + \varepsilon_1$, $\varepsilon_1 \sim N(0,\sigma^2 \approx 156.25)$ and the model for the second response is $Y_2 + \varepsilon_2$, $\varepsilon_2 \sim$ Weibull($\theta = -1.45774$, $\beta = 0.838157$, $\alpha = 0.542526$). Table 3.5 shows further details of the case setting such as the random variable used and its parameters for each response. Appendix C, Section III "Calculation of the random variable for Y2".

Table 3.5 Error Distributions for Random Variables Y_1 and Y_2 for Case Study 3

Specifications	**E($\underline{\varepsilon}$)**	**var($\underline{\varepsilon}$)**	**θ= MIN($\underline{\varepsilon}_2$)**	**α**	**β**
Y_1 Response: Normal NIID(0,σ^2)	0	156.25	-	-	-
Y_2 Response: 3-Parameter Weibull	0	8.506929	-1.45774	0.542526	0.838157

Table 3.6 shows the results for this case by pointing out the distributions means, standard deviation, minimum and maximum as well as its percentiles. The table includes the actual optimums values for this problem; the results for the proposed methods will be compared against these values. From Table 3.6 the closest mean values with the minimum variability for optimum values of input variables were those found by method R1. It follows method R1 also has the minimum mean distance and distance variability, minimum mean loss and loss variability. Mean value for the responses was also the closest to the optimum responses with the minimum variability. Methodology R2 mean values for input variables were farther with their variability very close to R1's. This situation results in mean values for the distance and its variability that are also very close to R1. Loss mean value was larger with variability close to R1's. The neural net results were the worst. The neural net mean values for the inputs variables were he farther from the optimum and their variability's were, at least, 5.2 times R1's. This variation resulted in mean values for the distance as high as 55 times R1's with variation as high as 34 times R1's. The neural network loss mean value was the larger with a variability of about 62 times R1's. Mean values for the responses were the farther with variability between 5 and 7.5 times R1's.

Table 3.6 Case Study 3: Summary of Results
Based on 1,000 Simulations, Sample Size n= 5

OPTIMUM	**X_1***	**X_2***	**Distance**	**Loss**	**Y_1***	**Y_2***
	-0.6735	**1.414**	**0.0000**	**2.4431**	**895.606**	**24.2301**
Responses Lower Specification Limits (LSL) and Inputs Ranges)	-1.4140	-1.4140	**-**	**-**	**800**	**0**
Response Upper Specification Limits (USL) and Inputs Ranges	1.4140	1.4140	**-**	**-**	**950**	**35**
Target	-	-	**-**	**-**	**950**	**0**
Regression Method 1 R1 (Formal Optimization)	**X_1***	**X_2***	**Distance**	**Loss**	**Y_1***	**Y_2***
Mean	-0.6800	1.4140	0.0314	2.4615	895.2739	24.1890
Std Dev	0.1773	0.0000	0.0490	0.0282	9.0245	1.1188
Minimum	-1.3514	1.4140	0.0000	2.4430	861.1009	19.9526
Maximum	-0.2680	1.4140	0.4595	2.7049	916.2460	26.7888
Percentile 5%	-0.9793	1.4140	0.0001	2.4430	880.0416	22.3006
Percentile 10%	-0.9024	1.4140	0.0004	2.4432	883.9560	22.7859
Percentile 20%	-0.8166	1.4140	0.0019	2.4441	888.3243	23.3274
Percentile 25%	-0.7901	1.4140	0.0029	2.4447	889.6711	23.4944
Percentile 33%	-0.7485	1.4140	0.0052	2.4462	891.7885	23.7569
Percentile 50%	-0.6740	1.4140	0.0135	2.4505	895.5806	24.2270
Percentile 67%	-0.6018	1.4140	0.0301	2.4602	899.2556	24.6826
Percentile 75%	-0.5597	1.4140	0.0409	2.4681	901.4010	24.9486
Percentile 80%	-0.5244	1.4140	0.0513	2.4730	903.1973	25.1712
Percentile 90%	-0.4528	1.4140	0.0765	2.4890	906.8422	25.6231
Percentile 95%	-0.4108	1.4140	0.1185	2.5127	908.9780	25.8879
Regression Method 2 R2 (Random Search)	**X_1***	**X_2***	**Distance**	**Loss**	**Y_1***	**Y_2***
Mean	-0.6466	1.4008	0.0545	2.5010	894.9284	24.3168
Std Dev	0.2313	0.0114	0.0729	0.0489	11.6648	1.4515
Minimum	-1.3465	1.3397	0.0000	2.4441	859.3678	19.9502
Maximum	-0.0189	1.4140	0.4530	2.7629	927.6106	28.2997
Percentile 5%	-1.0684	1.3788	0.0004	2.4506	874.1710	21.6832
Percentile 10%	-0.9569	1.3854	0.0010	2.4550	879.3154	22.4005
Percentile 20%	-0.8418	1.3928	0.0036	2.4635	885.4094	23.1166

OPTIMUM	X_1*	X_2*	Distance	Loss	Y_1*	Y_2*
	-0.6735	**1.414**	**0.0000**	**2.4431**	**895.606**	**24.2301**
Responses Lower Specification Limits (LSL) and Inputs Ranges)	-1.4140	-1.4140	-	-	**800**	**0**
Response Upper Specification Limits (USL) and Inputs Ranges	1.4140	1.4140	-	-	**950**	**35**
Target	-	-	-	-	**950**	**0**
Percentile 25%	-0.7867	1.3953	0.0058	2.4674	887.8668	23.4372
Percentile 33%	-0.7282	1.3983	0.0101	2.4734	890.6855	23.7974
Percentile 50%	-0.6342	1.4037	0.0251	2.4884	895.6731	24.3901
Percentile 67%	-0.5371	1.4077	0.0533	2.5064	900.4714	24.9806
Percentile 75%	-0.4851	1.4096	0.0737	2.5193	902.9082	25.3260
Percentile 80%	-0.4486	1.4105	0.0885	2.5286	904.6090	25.5678
Percentile 90%	-0.3634	1.4123	0.1519	2.5617	909.4424	26.1082
Percentile 95%	-0.2900	1.4132	0.2100	2.6003	913.1831	26.5511
Artificial Neural Network ANN (Random Search)	X_1*	X_2*	Distance	Loss	Y_1*	Y_2*
Mean	0.1522	1.1855	1.7355	3.9042	902.2625	28.0054
Std Dev	0.9320	0.3658	1.6897	1.7458	68.0175	5.9737
Minimum	-1.4137	-1.2024	0.0001	2.4439	556.8000	14.1851
Maximum	1.4138	1.4140	9.1211	28.3199	1001.1000	37.3363
Percentile 5%	-1.0684	0.3524	0.0094	2.4827	779.1600	19.4944
Percentile 10%	-1.1830	0.5571	0.0272	2.5143	812.8700	20.2000
Percentile 20%	-0.8035	0.9360	0.1138	2.6221	852.1800	22.4168
Percentile 25%	-0.6614	1.1327	0.2035	2.6997	859.3750	23.1185
Percentile 33%	-0.4514	1.3199	0.4475	2.8901	874.0000	24.0900
Percentile 50%	0.2432	1.3730	1.1625	3.5180	899.0500	26.7089
Percentile 67%	0.7574	1.3921	2.2992	4.2486	945.3000	31.8060
Percentile 75%	1.0714	1.4000	3.2703	4.5972	964.1500	33.5456
Percentile 80%	1.2546	1.4034	3.8179	4.7304	972.5200	34.7487
Percentile 90%	1.3782	1.4092	4.3000	5.4946	989.8000	36.4453
Percentile 95%	1.4010	1.4119	4.4933	6.6636	994.2000	36.8350

Figure 3.11 shows the distributions of the Euclidean distance from optimum for Case Problem 3 using all methods. The figure shows both regression methods produced similar

distances; all 1,000 cases were in the 0.0 to 0.5 interval for the regression method R1 and for the regression search method R(M2) and only 357cases out of 1,000 for the neural net.

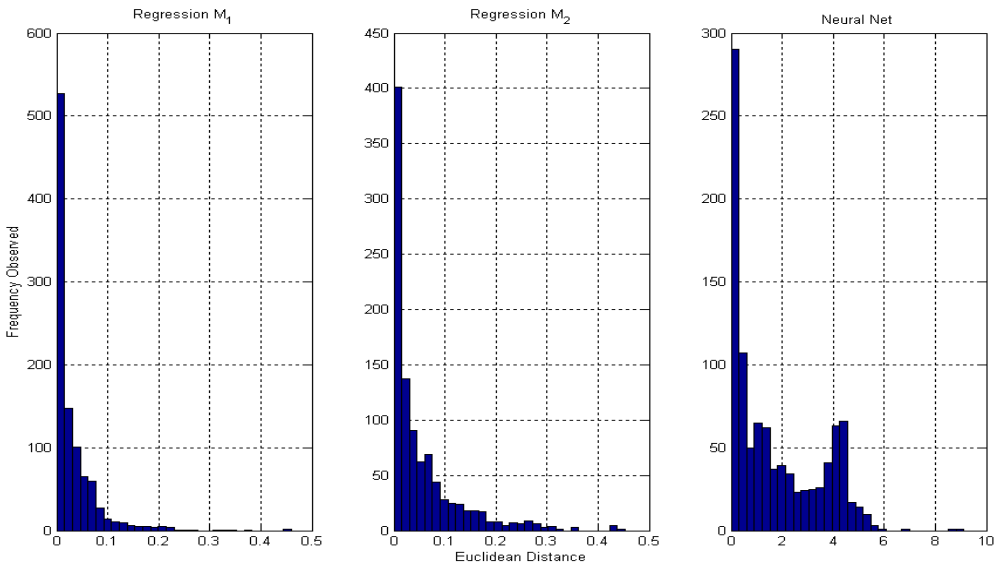

Figure 3.11. Euclidean distances from optimum for each of the three methods.

Figure 3.12 shows the scatter plot for all three approaches and their location related to the true optimum $(X_1^*, X_2^*) = (-0.6735, 1.4140)$. For the regression search method R(M1): (X_{R11}^*, X_{R12}^*) maximum distances give us some insight about the methodology's performance. Maximum distances are from X_1^* to X_{R11}^*: 0.6779 units, from and X_2^* to X_{R12}^*: 0 units. For the regression search method R(M2): (X_{R21}^*, X_{R22}^*) maximum distances are from X_1^* to X_{R21}^* 0.6730 units and from X_2^* to X_{R22}^*: 0.0743 units which are, at least, 0.96 times larger than those of R1's. For the neural net (X_{N1}^*, X_{N2}^*), maximum distances are from X_1^* to X_{N1}^*: 2.0873 units and from X_2^* to X_{N2}^*: 2.6164 units which are, at least, 3.1 times larger than those of R1's.

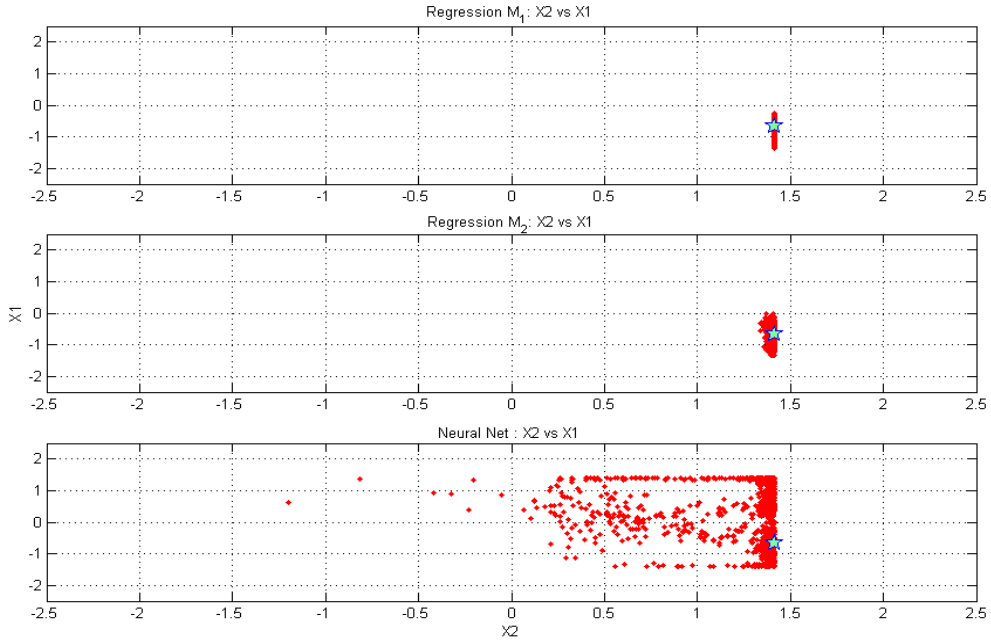

Figure 3.12. Scatter plot for all three approaches
(Optimum point starred for comparison)

Figure 3.13 shows the distributions for the response Y_1^*. The [890, 910] interval contained 705 R(M1) cases, 595 R(M2) cases, and 120 cases for the neural net.

Figure 3.14 shows the distribution for the response Y2*. The [23, 25] interval contained 635 R(M1) cases, 519 R(M2) cases, and 154 cases for the neural net.

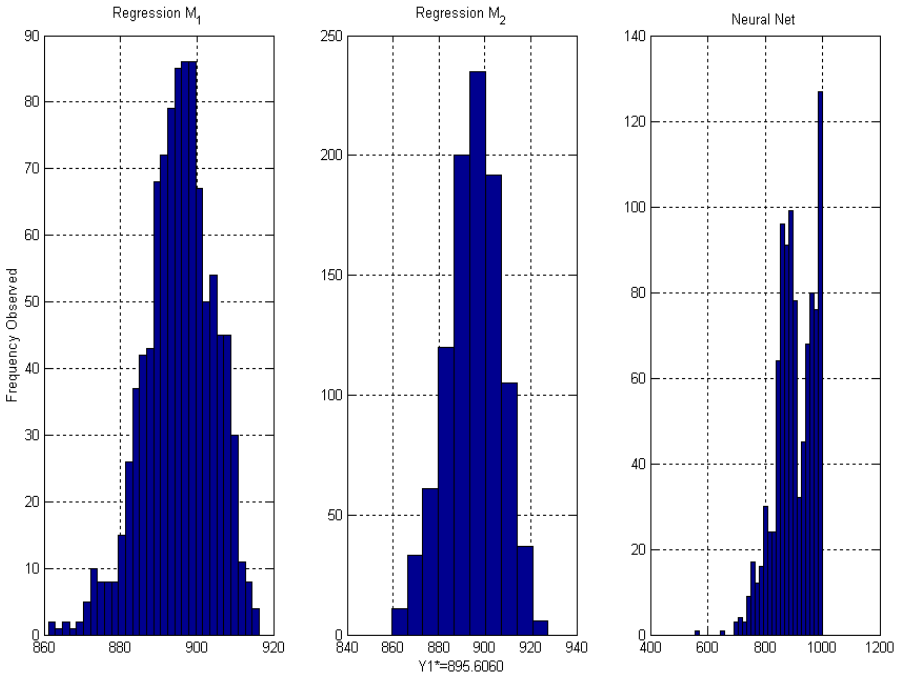

Figure 3.13 Distributions for the Y_1* Response

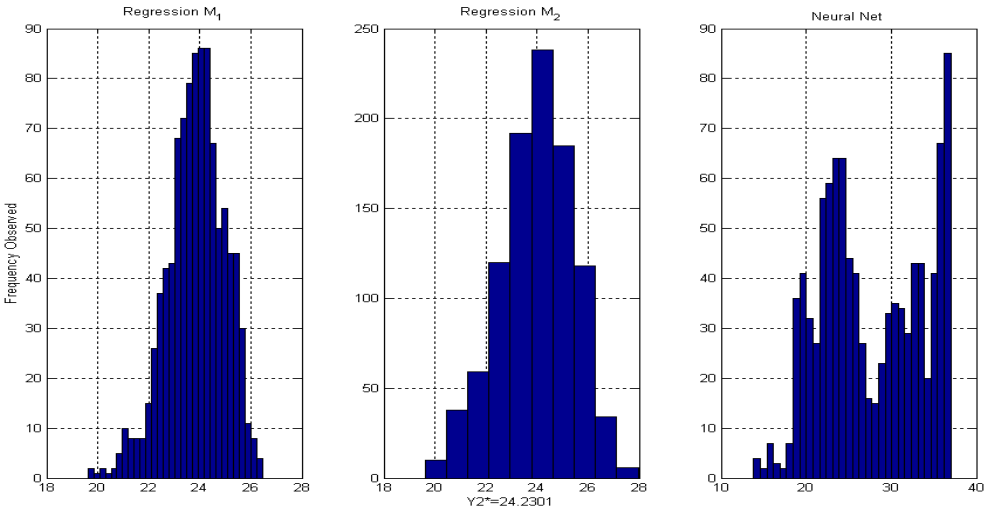

Figure 3.14 Distributions for the Y_2* Response.

The regression methods gave substantially similar results. From Figures 3.11 & 3.12, it can be observed that both approaches were almost identical. It might be hypothesized that the brute force was as effective as the Lagrange optimization. Results from the neural network were dispersing, particularly for the X_2 input. From Figure 3.13 we see that regression distributions for the Y_1 response were practically the same and the neural net was the method with the biggest offset from the mean; dispersion for the regressions were also similar with the neural net quadruplicating it. From Figure 3.14 we see that regression distributions for the Y_2 response were also similar. Regression means were practically the same with the neural net being the method with the biggest offset from the mean; dispersion for the regressions was also similar with the neural net triplicating it.

We might infer that the brute force methodology is an acceptable method to find optimum solutions.

3.5 Case Problem 4: Dual Quadratic Responses B

The speed, pressure, and distance experiment from Kim and Lin (1998) is another example for studying the behavior of a dual quadratic response system. In this paper, the control (input) variables X_1, X_2, and X_3 were the speed, pressure, and distance on the quality of a printing process respectively. The output (responses) variables, Y_1 and Y_2, were the process mean (μ) and its standard deviation (σ) respectively. In this case, the adjusted equations as reported by Kim & Lin are:

$$Y_1 = 327.6 + 177.0X_1 + 109.4X_2 + 131.5X_3 + 32.0X_1^2 - 22.4X_2^2 - 29.1X_3^2 + 66.0X_1X_2 + 75.5X_1X_3 + 43.6X_2X_3 \quad 3.10$$

and,

$$Y_2 = 34.9 + 11.5X_1 + 15.3X_2 + 29.2X_3 + 4.2X_1^2 - 1.3X_2^2 + 16.8X_3^2 + 7.7X_1X_2 + 5.1X_1X_3 + 14.1X_2X_3 \quad 3.11$$

where the input vector **X** is bounded between:

$$-1.68179 \le X_j \le 1.68197 \text{ for } j = 1,2,3 \quad 3.12$$

Kim& Lin wanted to set Y_1 and Y_2 to certain predetermined values. They wanted to find the combination of inputs that most consistently gives us a predetermined process mean possible and predetermined value of process deviation. We have the set the same purpose here for each response; therefore, we established the output (responses) specification limits from Kim& Lin's results to compare our solutions. Kim& Lin performed a sensitivity analysis upon output variable Y_1 resulting in a target value of 500 with minimum of 490 and maximum of 510. Their specification limits for Y_2, were (1,500)^.5 ≈ 38.7298 to (2,100)^.5 ≈ 45.8258 and targeting (1,500)^.5 ≈ 38.7298. We will use the same intervals and targets for both responses. See Appendix D, Section I

We solved the problem finding the set of inputs that set the Y_1 response at a specific value while minimizing the Y_2 response using the loss function. The problem solved was a constrained minimization problem as described by Eq. 2.3. See Appendix D, Section II.

We need to simulate the CCD arrangement used in the sampling process, the values of the lower and upper limits for the inputs are given as (-α, α) where | α = [number of factorial runs] $^{0.25}$; in this case, we are using a full factorial with three factors and two levels, therefore $|\alpha| = [2^3]^{0.25} = 8^{0.25}$, $|\alpha| \approx 1.68179$. One thousand simulations runs were done, each a sample size, n, of five (5). An important consideration for both responses is that we want to select the value of their individual variances, σ_i^2 so that the potential process capability for each response equals 2. We added to Y_1 the random error, ε_1 , modeled by a normal distribution, with mean = 0 and variance = σ^2 . The new random variable has mean = $E(Y_1 + \varepsilon_1) = Y_1 + 0 = Y_1$ and, variance= var $(Y_1 + \varepsilon_1) = 0 + \sigma^2 = \sigma^2$ since Y_1 is a constant term. Then, $C_p(\hat{Y}_1) = \frac{(USL - LSL)}{6 * STDDEV(\hat{Y}_1)}$, or, 2 = (510-490)/(6*σ). Solving for σ, gives σ = (20)/(6*2)= 20/12 ≈ 1.667. Therefore, var(ε_1) = $\sigma^2 \approx 2.77$.

A similar procedure is applicable for Y_2; however, to have different error distributions to each response, we decided that the error random variable added must have $E(\varepsilon_2) = 0$. Thus, we have for Y_2 that $\text{var}(\hat{Y}_2) = \text{var}(\varepsilon_2) = ((\text{USL-LSL})/12)^2 = ((2100\text{^}2 - 1500\text{^}2)/12)^2 \approx 0.3496757$. For the error added to Y_2 we want a random variable that satisfies the conditions: a) $E(\varepsilon_2) = 0$; b) $Y_2 + \varepsilon_2 \geq 0$; and c) $\text{var}(\hat{Y}_2) = ((\text{USL-LSL})/12)^2$. The 3-parameter Weibull distribution with CDF: $F(x) = 1-\exp\{[(x - \theta)/\beta]^{\alpha}\}$ will be used. Condition (b) must be true always. Since ε_2 is a random number that could be negative by condition (a),

we must ensure that the sum of most negative random number that could arise from the selected Weibull distribution (that is, its location parameter, θ) plus the minimum possible value of Y_2 results in a non-negative value. Finding the minimum Y_2 in the stated range is a trivial problem using Eq. 3.11. Parameters are found by solving the simultaneous system of equations defined by conditions (a), (b) and (c) above: $E(\varepsilon_2) = \beta*\Gamma\{(\alpha+1)/\alpha\} + \theta = 0$ and $var(\varepsilon_2) = \beta^2*\Gamma\{(\alpha+2)/\alpha\} - [\beta*\Gamma\{(\alpha+1)/\alpha\}]^2 = ((USL-LSL)/12)^2 \approx 0.3496757$ with the MIN $Y_2 = 38.7298$ found by solving the constrained minimization Eq. 3.11 subject to Eq. 3.12; then, $\theta = MIN(\varepsilon_2) \geq -38.7298$ (= - MIN Y_2). Using Mathcad, a solution found for this system is $F(x) = 1-\exp\{[(x +0.267926)/.136277]^{0.504702}\}$. (= - MIN Y_2). The model for the first response is $Y_1 + \varepsilon_1$, $\varepsilon_1 \sim N(0,\sigma^2 \approx 2.77)$ and the model for the second response is $Y_2 + \varepsilon_2$, $\varepsilon_2 \sim$ Weibull($\theta = -0.267926$, $\beta = 0.136277$, $\alpha = 0.504702$). Table 3.7 shows further details of the case's setting such as the random variable used and its parameters for each response. Appendix D, Section III "Calculation of the random variable for Y2".

Table 3.7 Error Distributions for Random Variables Y_1 and Y_2 for Case Study 4

Distributions	**$E(\varepsilon)$**	**$var(\varepsilon)$**	**θ= MIN(ε_2)**	**α**	**β**
Y_1 Response: Normal NIID$(0,\sigma^2)$	0	2.77	-	-	-
Y_2 Response: 3-Parameter Weibull	0	0.3496757	-0.267926	0.504702	0.136277

Table 3.8 shows details for this case. It summarizes the results for the three approaches by pointing out the distributions means, standard deviation, minimum and

maximum as well as its percentiles. From Table 3.8 the closest mean values with the minimum variability for optimum values of the input variables were those found by method R1. It follows method R1 also has the minimum mean distance and distance variability, minimum mean loss and loss variability. Mean value for the responses was also the closest to the optimum responses with the minimum variability. Methodology R2 mean values for input variables were farther with bigger variability than R1. This situation results in mean values for the distance and its variability larger than R1. Loss mean value was larger with a greater variability. The neural net results were the worst. The neural net mean values for the inputs variables were the farther from the optimum and their variability's were a hundredth times R1's. This much variation resulted in mean values for the distance and its variability as high as hundredth times R1. The neural network loss mean value was the larger with a variability of about several thousand times R1. Mean values for the responses were the farther with variability as high as 100 times R1.

Table 3.8 Case Study 4: Summary of Results
Based on 1,000 Simulations, Sample Size n= 5

OPTIMUM	**X_1***	**X_2***	**X_3***	**Distance**	**Loss**	**Y_1***	**Y_2***
	1.6818	**-0.9198**	**0.0233**	**0.0000**	**0.07**	**499.1874**	**39.6233**
Responses Lower Specification Limits (LSL) and Inputs Ranges	**-1.68179**	**- 1.68179**	**- 1.68179**	**-**	**-**	**490**	**(1,500)^.5 ≈ 38.7298**
Response Upper Specification Limits (USL) and Inputs Ranges	**-1.68179**	**1.68179**	**1.68179**	**-**	**-**	**510**	**(2,100)^.5 ≈ 45.8258**
Target	**-**	**-**	**-**	**-**	**-**	**500**	**(1,500)^.5 ≈ 38.7298**
Regression Method 1 R1 (Formal	**X_1***	**X_2***	**X_3***	**Distance**	**Loss**	**Y_1***	**Y_2***

OPTIMUM	X_1*	X_2*	X_3*	Distance	Loss	Y_1*	Y_2*
	1.6818	-0.9198	0.0233	0.0000	0.07	499.1874	39.6233
Responses Lower Specification Limits (LSL) and Inputs Ranges	-1.68179	-1.68179	-1.68179	-	-	490	(1,500)^.5 ≈ 38.7298
Response Upper Specification Limits (USL) and Inputs Ranges	-1.68179	1.68179	1.68179	-	-	510	(2,100)^.5 ≈ 45.8258
Target	-	-	-	-	-	500	(1,500)^.5 ≈ 38.7298
Optimization)							
Mean	1.6818	-0.9202	0.0236	0.0078	0.0822	499.1624	39.6207
Std Dev	0.0000	0.0076	0.0056	0.0053	0.0171	1.04810	0.1235
Minimum	1.6818	-0.9480	0.0066	0.0001	0.0700	495.7009	39.2127
Maximum	1.6818	-0.8955	0.0426	0.0342	0.2033	502.4897	40.0132
Percentile 5%	1.6818	-0.9329	0.0146	0.0015	0.0701	497.3989	39.4124
Percentile 10%	1.6818	-0.9304	0.0166	0.0020	0.0702	497.7744	39.4569
Percentile 20%	1.6818	-0.9265	0.0190	0.0032	0.0708	498.2642	39.5151
Percentile 25%	1.6818	-0.9251	0.0198	0.0036	0.0713	498.4455	39.5360
Percentile 33%	1.6818	-0.9231	0.0211	0.0045	0.0721	498.7387	39.5708
Percentile 50%	1.6818	-0.9201	0.0235	0.0067	0.0758	499.2019	39.6253
Percentile 67%	1.6818	-0.9166	0.0259	0.0091	0.0814	499.6121	39.6748
Percentile 75%	1.6818	-0.9149	0.0274	0.0110	0.0864	499.8825	39.7054
Percentile 80%	1.6818	-0.9137	0.0284	0.0122	0.0907	500.0373	39.7241
Percentile 90%	1.6818	-0.9104	0.0310	0.0151	0.1039	500.4927	39.7775
Percentile 95%	1.6818	-0.9084	0.0331	0.0181	0.1162	500.8337	39.8175
Regression Method 2 R2 (Random Search)	X1*	X2*	X3*	Distance	Loss	Y1*	Y2*
Mean	1.4104	-0.4904	-0.1255	0.6407	2.0666	496.7486	42.4336
Std Dev	0.2024	0.4115	0.2807	0.3983	1.8539	7.62810	1.9109
Minimum	0.2893	-1.3821	-0.7732	0.0185	0.1288	473.7393	37.6160
Maximum	1.6810	1.1324	0.6898	2.3080	33.2355	517.5233	57.8951
Percentile 5%	1.0301	-1.0704	-0.5606	0.1416	0.3792	484.5463	39.4768
Percentile 10%	1.1421	-1.0011	-0.4891	0.2021	0.5541	487.0178	40.0266
Percentile 20%	1.2361	-0.8638	-0.3832	0.2912	0.9140	489.9007	40.7712
Percentile 25%	1.2706	-0.7852	-0.3417	0.3458	1.0515	491.1889	41.1290

OPTIMUM	X_1*	X_2*	X_3*	Distance	Loss	Y_1*	Y_2*
	1.6818	-0.9198	0.0233	0.0000	0.07	499.1874	39.6233
Responses Lower Specification Limits (LSL) and Inputs Ranges	-1.68179	-1.68179	-1.68179	-	-	490	(1,500)^.5 ≈ 38.7298
Response Upper Specification Limits (USL) and Inputs Ranges	-1.68179	1.68179	1.68179	-	-	510	(2,100)^.5 ≈ 45.8258
Target	-	-	-	-	-	500	(1,500)^.5 ≈ 38.7298
Percentile 33%	1.3434	-0.7012	-0.2788	0.4151	1.3173	493.1828	41.5145
Percentile 50%	1.4410	-0.5183	-0.1353	0.5704	1.7617	496.6193	42.4238
Percentile 67%	1.5412	-0.3482	-0.0016	0.7561	2.4230	500.1911	43.2446
Percentile 75%	1.5827	-0.2520	0.0752	0.8530	2.7920	502.5792	43.6987
Percentile 80%	1.6032	-0.1839	0.1310	0.9259	3.0394	503.8933	44.0342
Percentile 90%	1.6453	0.0225	0.2585	1.1559	3.7446	506.7122	44.7565
Percentile 95%	1.6590	0.2606	0.3584	1.4037	4.2917	508.8235	45.4049
Artificial Neural Network ANN (Random Search)	X1*	X2*	X3*	Distance	Loss	Y1*	Y2*
Mean	0.2103	0.0402	0.1307	2.1434	87.1472	360.6749	53.8619
Std Dev	0.9458	0.9135	0.8088	0.9414	97.1875	235.1805	36.6541
Minimum	-1.6815	-1.6799	-1.6688	0.1038	0.1000	-108.0000	1.4309
Maximum	1.6811	1.6804	1.6808	4.2549	1071.3000	1385.0000	234.8270
Percentile 5%	1.0301	-1.4507	-1.2855	0.6301	2.7000	58.2250	16.0791
Percentile 10%	-1.1419	-1.2224	-0.9294	0.8374	6.4900	102.6700	19.1453
Percentile 20%	-0.6601	-0.8402	-0.5526	1.2275	15.8600	149.7600	25.1111
Percentile 25%	-0.5059	-0.6776	-0.4219	1.4164	20.3750	181.5000	27.2588
Percentile 33%	-0.2429	-0.4239	-0.2060	1.6617	30.0000	233.0000	31.4779
Percentile 50%	0.2251	0.0372	0.1366	2.1234	58.4000	332.3000	41.6020
Percentile 67%	0.7421	0.4806	0.4845	2.5938	104.2000	424.7000	59.3799
Percentile 75%	1.0346	0.7597	0.7043	2.8483	129.9250	487.0250	70.1565
Percentile 80%	1.2267	0.9507	0.8372	3.0375	141.6400	548.2200	80.1973
Percentile 90%	1.4965	1.3723	1.2625	3.4313	188.7500	669.4400	104.9514
Percentile 95%	1.5846	1.5569	1.5085	3.6858	243.2150	816.4750	132.3778

Figure 3.15 shows the distributions of the Euclidean distance from optimum for Case Problem 4 using all methods. The figure shows that the regression search method R(M1) was the method that produced distances closer to zero; all 1,000 cases were in the 0.0 to 0.5 interval compared to 428 out of 1,000 cases for the regression search method R(M2) and only 29 cases out of 1,000 for the neural net.

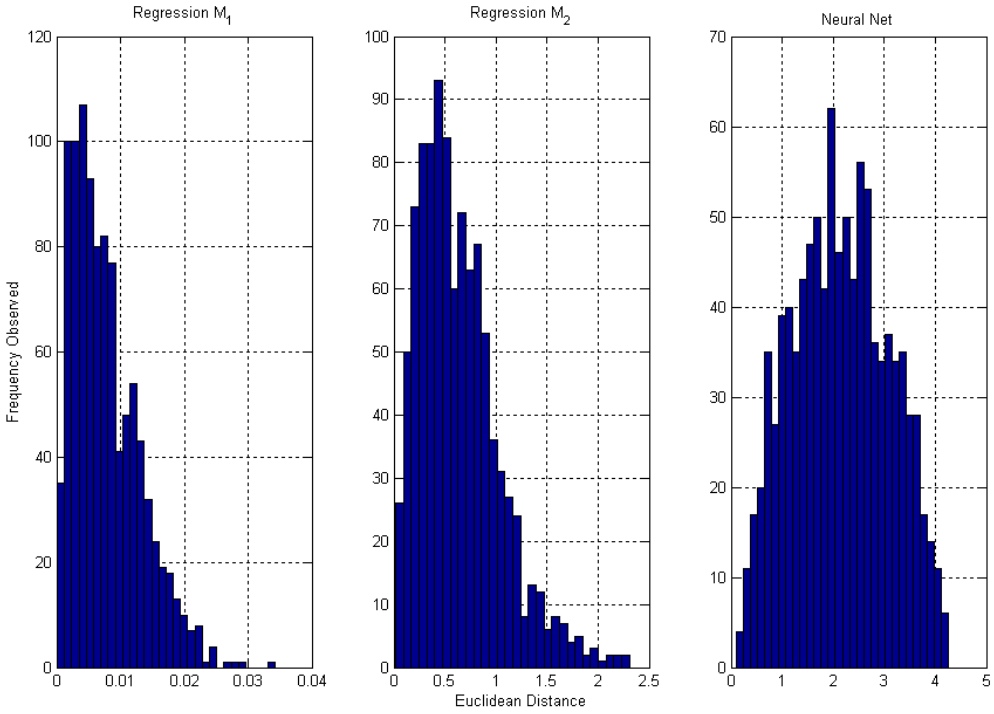

Figure 3.15. Euclidean distances from optimum for each of the three methods.

Figure 3.16 shows the scatter plot for the regression search method R(M1): (X_{R11}^*, X_{R12}^*, X_{R13}^*) and its location related to the true optimum (X_1^*, X_2^*, X_3^*) = (1.6818, -0.9198,

0.0233). Maximum distances give us some insight about the methodology's performance. Maximum distances are from X_1^* to X_{R11}^*: 0 units, from X_2^* to X_{R12}^*: 0.0282 units and from X_3^* to X_{R13}^*: 0.0193 units.

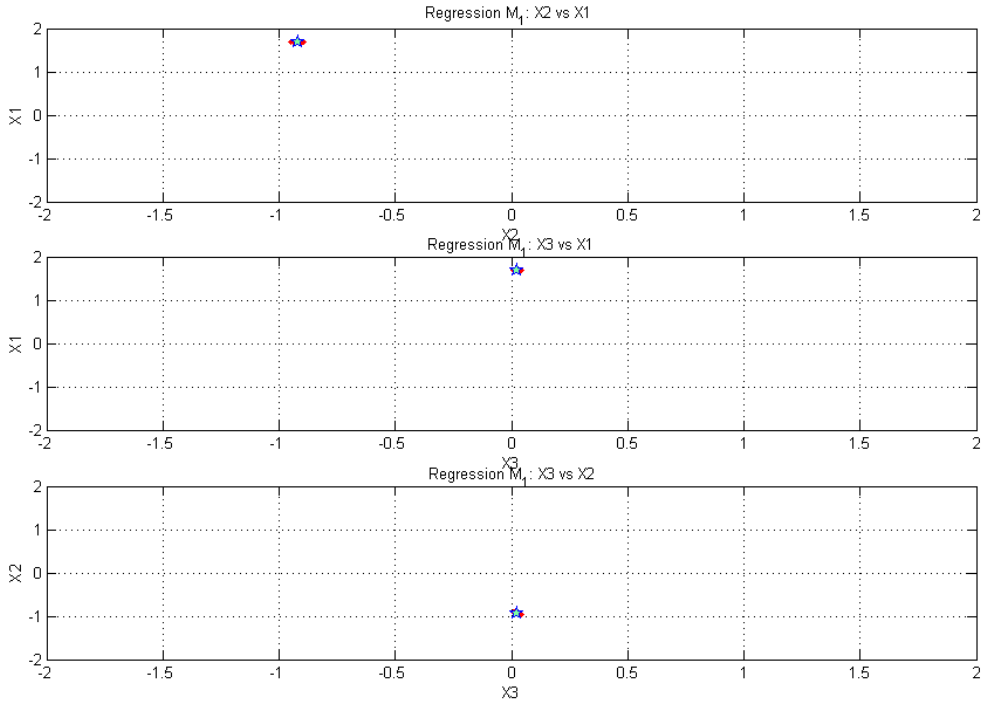

Figure 3.16. Scatter plot for regression search method R(M1)
(Optimum point starred for comparison)

Figure 3.17 shows the scatter plot for the regression search method R(M2): (X_{R21}^*, X_{R22}^*, X_{R23}^*) and its location related to the true optimum (X_1^*, X_2^*, X_3^*) = (1.6818, -0.9198, 0.0233). Maximum distances are from X_1^* to X_{R21}^*: 1.3925 units, from X_2^* to X_{R22}^*: 2.0522 units and from X_3^* to X_{R23}^*: 0.7965 units which are, at least, 41 times larger than those of R1's.

Figure 3.18 shows the scatter plot between the sets found by the ANN (X_{N1}^*, X_{N2}^*, X_{N3}^*) and its location related to the true optimum (X_1^*, X_2^*, X_3^*) = (1.6818, -0.9198, 0.0233). Maximum distances are from X_1^* to X_{N1}^*: 3.3633 units, from X_2^* to X_{N2}^*: 2.6002 units and from X_3^* to X_{N3}^*: 1.6921 units which are, at least, 87 times larger than those of R1's.

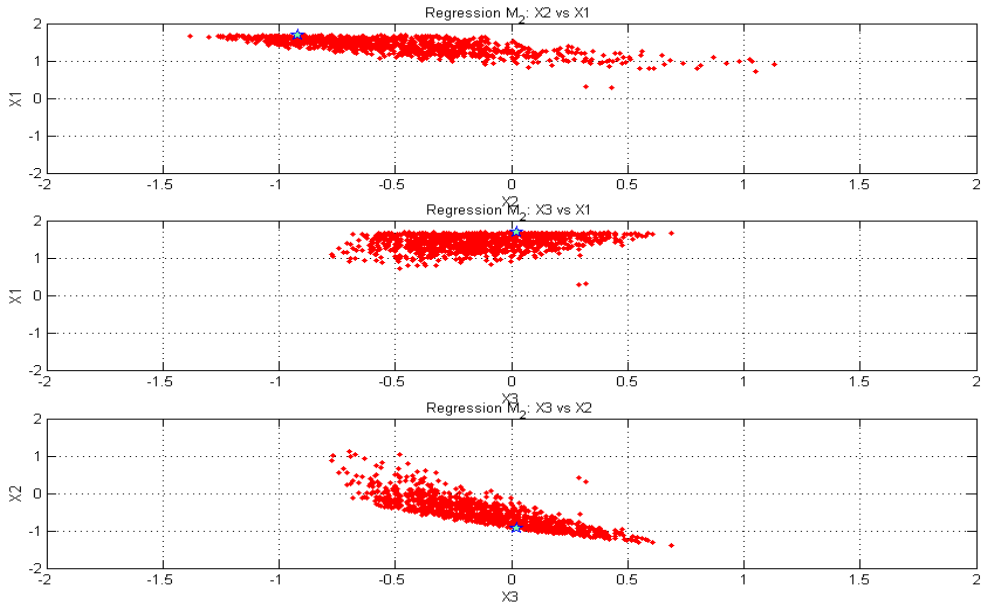

Figure 3.17. Scatter plot for regression search method R(M2)
(Optimum point starred for comparison)

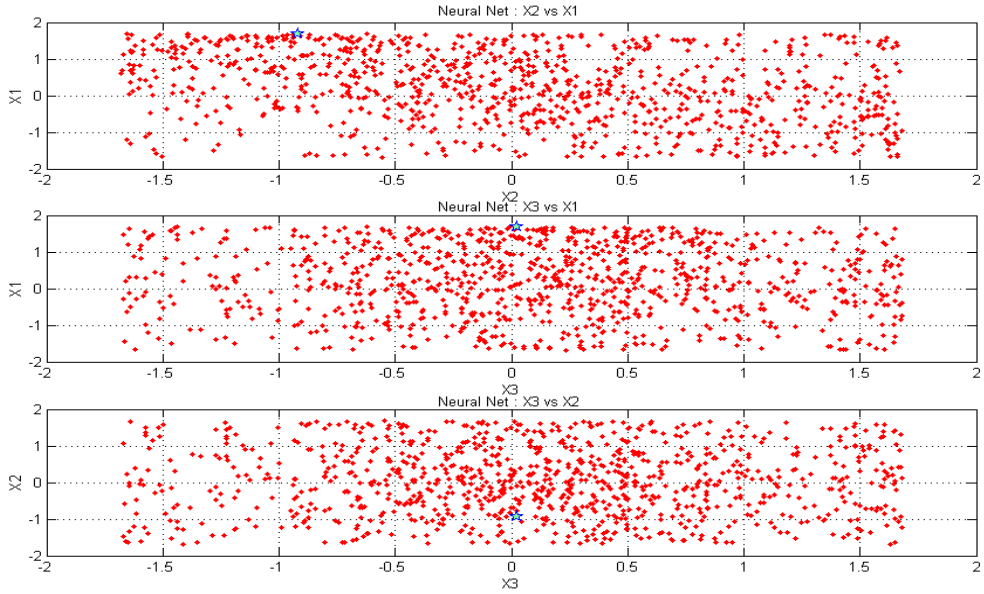

Figure 3.18 Scatter plot for neural network search method ANN
(Optimum point starred for comparison)

Figure 3.19 shows the distributions for the response Y_1^*. The [490, 510] interval contained all 1,000 R(M1) cases, 765 R(M2) cases, and 15 cases for the neural net.

Figure 3.20 shows the distribution for the response Y_2^*. The [38, 40] interval contained 999 R(M1) cases, 92 R(M2) cases, and 27 cases for the neural net.

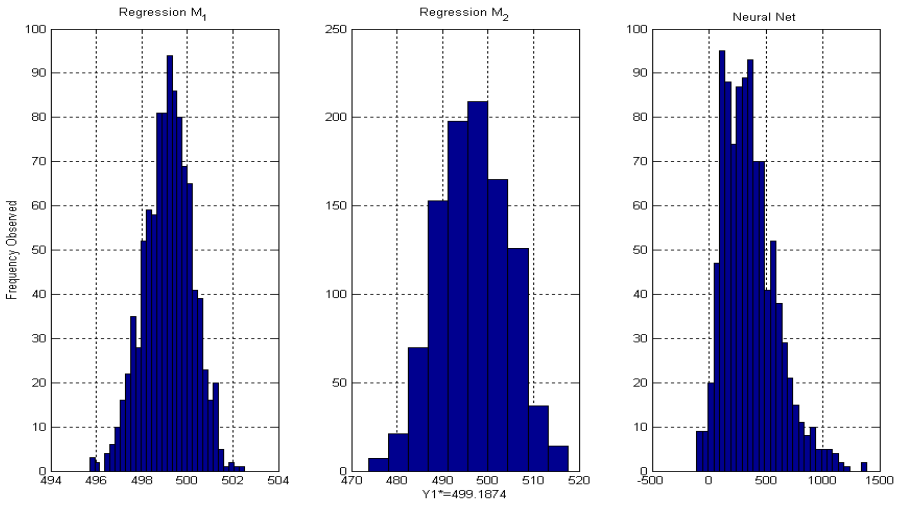

Figure 3.19 Distributions for the Y_1* Response

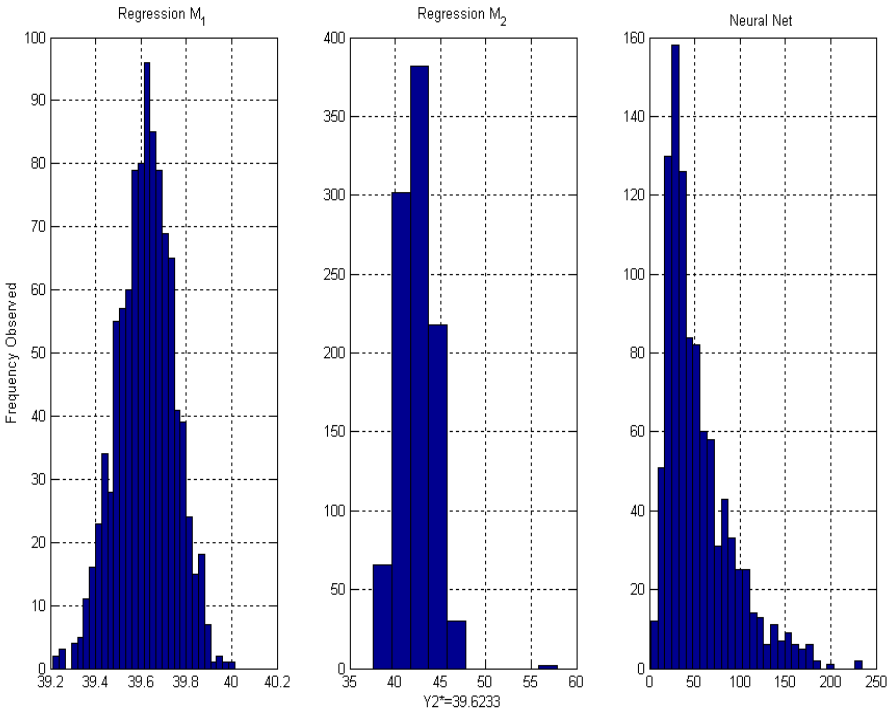

Figure 3.20 Distributions for the Y_2* Response.

The regression method R(M1) outperformed all others methodologies by far in this case. From Figures 3.15, 3.16, 3.17, & 3.18, it is easily observed that this approach consistently found the closest solution to the optimum. The neural network ANN methodology was very ineffective in finding the optimum, its maximum distances compared against R(M2) which used the same brute force "optimization" were, on occasions, several multiples.

The effects on the input can be observed in Figures 3.19 and 3.20. The regression R(M1) method found the best possible distributions for the responses. We might infer that the brute force methodology could be an acceptable method to find optimum solutions.

4 CONCLUSIONS AND FUTURE WORK

The R(M1) is the easier method to apply, provided the best solutions in the nonlinear cases and almost as good results as the R(M2) for cases with at least one linear response. It should be noted, that the best results for R(M1) were on target is best Case 4 situation, the all nonlinear case and, with its poorer results in the case with one linear and one nonlinear response.

The R(M2) approach gave excellent results. Since the only difference between this methodology and the R(M1) was the use of a "brute force" algorithm, we should conclude that, even though the Lagrange optimization is the superior method, this "brute force" methodology could be an acceptable one as well.

The neural network results were not acceptable. The best cases were the ones where at least one response was linear, which is an unexpected result. The poor performance of the neural network cannot be attributed to its "brute force" algorithm, as this was the same one that R(M2) used.

It is possible, that to be an effective methodology, the neural network required more data than what the study simulated as input. Nevertheless, regression approaches, in particular the R(M1), used the same amount of data and provided outstanding results. The superiority of the regression approach is clear from this study.

Recalling that Del Castillo's methodology depended on the confidence region being explored (Del Castillo, 1996) for Case Problem I, his sensitivity analysis showed an optimum pair (Y_1, Y_2) of (116.59, 7.41) when the confidence level was bigger, i.e., $\alpha = 0.2$. Del Castillo sensitivity shows a change of 4 units of Y_1 per unit of Y_2. Comparing regression methodologies results against those Del Castillo's predicted; both standard regression methods predicted greater Y_1 average values for about the same Y_2. For Y_2 equal to 8.205, Del Castillo's' Y_1 would be closer to 109, the standard regression prediction for Y_1 was 130.13. The "force brute" regression prediction for the average value of $Y_2 = 8.2032$ was $Y_1 = 128.45$. The neural network average value of $Y_2 = 8.7216$ with an associate prediction $Y_1 = 107.05$ was in closer agreement with Del Castillos' prediction.

For Case Problem II, Del Castillos' sensitivity analysis showed an optimum pair (Y_1, Y_2) of (863.83, 28.85) when the confidence level was bigger, i.e., $\alpha = 0.1$. Del Castillo sensitivity shows a change of 6 units of Y_1 per unit of Y_2. Comparing regression methodologies results against those Del Castillo's predicted; both standard regression methods predicted a little lower Y_1 average values for about the same Y_2. For Y_2 equal to 25.2846, Del Castillo's' Y_1 would be closer to 843, the standard regression prediction for Y_1 was 836.54. The "force brute" regression prediction for the average value of $Y_2 = 25.75$ was $Y_1 = 838.90$. The neural network average value of $Y_2 = 27.74$ with an associate prediction $Y_1 = 819.25$ was not in agreement with Del Castillos' prediction.

For Case Problem III, Del Castillo has not sensitivity analysis. However, this case is a modification of Case Problem II. Taking the sensitivity analysis from Case II for comparison purposes we would obtain the change of 6 units of Y_1 per unit of Y_2. Comparing regression methodologies results against those Del Castillo's predicted; both standard regression methods predicted a much bigger Y_1 average values for about the same Y_2. For Y_2 equal to 24.1890, Del Castillo's' Y_1 would be closer to 836, the standard regression prediction for Y_1 was 895.27. The "force brute" regression prediction for the average value of $Y_2 = 24.31$ was $Y_1 = 894.93$. The neural network average value of $Y_2 = 28.0054$ with an associate prediction $Y_1 = 902.26$ was not in agreement with Del Castillos' prediction.

For Case Problem IV, Kim and Lin (1998)' sensitivity analysis showed an optimum pair (Y_1, Y_2) of (496.08, 44.63) when the associate exponential membership function, called λ by the authors, used as minimization objective was minimum, i.e., $\lambda = 0.17$. Kim and Lin sensitivity shows a change of 0.12 units of Y_2 per unit of Y_1. Comparing regression methodologies results against those Kim and Lin's predicted; both standard regression methods predicted much lower Y_2 values for the same Y_1. For Y_1 equal to 500, Kim and Lin' Y_1 would be closer to 45.1, the standard regression prediction for Y_2 was 39.72. The "force brute" regression prediction for the value of $Y_1 = 500$ was $Y_2 = 43.25$. The neural network value of $Y_1 = 500$ with an associate prediction $Y_2 = 75$ was not in agreement with Kim and Lin's prediction.

In summary, the standard regression methodology using the loss function is a superior approach for the modeling and optimization of multiple response problems. In two out of the

four case problems studied, this methodology predicted superior average values for the response being maximized for about the same average value of the response being minimized. In one case, it predicted superior average values for the response being maximized for an average value of the response being minimized much lower than the one the original author predicted. For the target is best situation, this approach found a much lower average value of the response being minimized when the other response was kept constant. The force brute regression method provided similar results, on the average, than the standard regression for the three first cases. However, for the last case of target is best, the force brute regression method found a similar average value for the response being minimized as the one the original author predicted when the other response was kept constant. The neural network results were similar to the original author findings in the first case, worse in both responses in the second case, a little better in the third one and worse again in the last case.

We might conclude the standard regression is the best approach and the neural network should be discarded. The force brute regression although with acceptable results is not match for the standard regression but proved the optimization method use is not the cause of the neural network poor results.

Further research could include the use of both desirability and loss functions in cases where target is best, more is best, and less is best situations are examined. Also, different distributions for the error could be explored and their effect analyzed.

REFERENCES

Ames, A. E., Mattucci, N., MacDonald, S., Szony, G. and Hawkins, D. M. (1997). "Quality Loss Functions for Optimization across Multiple Response Surfaces". Journal of Quality Technology. 29(3), 339-346.

Artiles-León, N., (1996). "A Pragmatic Approach to Multiple-Response Problems Using Loss Function". Quality Engineering, 9(2), 213-220.

Benítez, J. M., Castro, J. L. and Requena, I., (1997). "Are Artificial Neural Networks Black Boxes?". IEEE Transaction on Neural Networks. 8(5), 1156-1164.

Bose, N. K. and Liang, P. (1996). "Neural Networks Fundamentals with Graphs, Algorithms and Applications". McGraw-Hill. New York.

Box, G.E.P. and Wilson, K.B. (1951). "On the Experimental Attainment of Optimum Conditions". Journal of the Royal Statistical Society, Series 13(1), 1-45.

Del Castillo, E. and Montgomery, D., (1993). "A Nonlinear Programming Solution to the Dual Response Problem". Journal of Quality Technology, 25(3), 199-204.

Del Castillo, E. (1996). "Multi-response Process Optimization via Constrained Confidence Regions". Journal of Quality Technology, 28(1). 61-70.

Del Castillo, E. (1997). "The Computational of Global Optima in Dual Response Systems". Journal of Quality Technology, 29(3). 347-373.

De Veaux R., Schweinsberg J. and Ungar, L .H. (1998). "Prediction Intervals for Neural Networks via Nonlinear Regression". Technometrics 40(4), 272-282.

Duarte-Ribeiro, J. L., Fogliatto, F. S. and ten-Caten, C.S. (2001). "Minimizing Manufacturing and Quality Costs in Multi-response Optimization". Quality Engineering 13(4), 559-569

Etter, D. M. (1996). "Introduction to Matlab for Engineers and Scientists". Prentice Hall. New Jersey.

Fausett, L. (1994). "Fundamentals of Neural Networks: Architectures, Algorithms, and Applications". Prentice Hall. New Jersey.

Fogliatto, F. S. and Albin, S. L. (2000). "Variance of Predicted Response as an Optimization Criterion in Multi-response Experiments". Quality Engineering 12(4), 523-533.

Hansselman, D. and Littlefield, B. (1998). "Mastering Matlab 5: A Comprehensive Tutorial and Reference". Prentice Hall, New Jersey.

Haykin, S. (1994). "Neural Networks: A Comprehensive Foundation". MacMillan. New York.

Holmes, D.S and Mergen, A. E. (2001). "Measuring Process Performance fro Multiple Variables: Revisited". Quality Engineering 13(4), 661-665.

Kim, K. and Lin, K.J., (1998). "Dual Response Surface Optimization: A Fuzzy Modeling Approach". Journal of Quality Technology, 30(1), 1-10.

Ko, Y., Kim K. and Jun, C. (2005). "A New Loss Function-Based Method for Multiresponse Optimization". Journal of Quality Technology. 37(1), 50-59.

Lin, D. and Tu, W., (1995). "Dual Response Surface Optimization". Journal of Quality Technology 27(1), 34-39.

Marchand, P. (1996). "Graphics and GUIs with Matlab". CRC Press. Boca Raton.

Mathworks. "Artificial neural network toolbox". http://www.mathworks.com/access/helpdesk/help/toolbox/nnet/. Active May 2008.

Montgomery, D.C. (1991). "Design and Analysis of Experiments". 3rd Ed. John Wiley & Sons, Inc. New York.

Moon, T.K. and Stirling, W. C. (2000). "Mathematical Methods and Algorithms for Signal Processing". Prentice Hall. New Jersey.

Myers, R. H. and Carter, W, H. (1973). "Response Surface Technique for Dual Response Systems". Technometrics 15, 301-317.

Myers, R.H. and Montgomery, D.C. (1995). "Response Surface Methodology: Process and Product Optimization Using Designed Experiments". John Wiley & Sons, Inc. New York.

Niebles-Bermúdez, R. A. (1997)."Un Nuevo Enfoque a Problemas de Múltiples Respuestas Usando Funciones de Perdida". Master in Engineering: Management Systems Engineering Thesis Report. University of PR, Mayagüez Campus. Unpublished.

Odmidvar O. and Elliot D.L. Editors. (1997). “Neural Systems for Control”. Academic Press. San Diego.

Pärt-Enander, E. and Sjöberg A. (1999) “The Matlab 5 Handbook”. Addison-Wesley. Harlow, England.

Pignatiello, J. J. (1993) “Strategies for Robust Multi-response Quality Engineering”. IIE Transactions 25(3), 5-15.

Ramírez, N. D. (1997). “Course on Neural Networks and Applications”. Seminar given at Abbott Diagnostics, Inc. Barceloneta, PR.

Redfern, D. and Campbell, C. (1998) “The Matlab 5 Handbook”. Springer. New York.

Refenes, A., Burges, A.N. and Bentz, Y., (1997). “Neural Networks in Financial Engineering: A Study in Methodology”. IEEE Transaction on Neural Networks. 8(6), 1222-1267.

Roussas, G. G. (1997) “A Course in Mathematical Statistics”. Academic Press. San Diego.

Sethi, S. and Thompson, G.L. (2000) "Optimal Control Theory: Applications to Management Science and Economics”. 2nd Ed. Kluwer Academic Publishers. Mass.

Smith, A. E. and Mason, A.K. (1997) “Cost Estimation Predictive Modeling: Regression versus Neural Network”. The Engineering Economist, Winter 1997. 42(2), 137-161.

Smith, T.H. and Boning, D.S. (1997). “A Self-Tuning EWMA Controller Utilizing Artificial Neural Network Function Approximation Techniques”. IEEE Transactions on Components, Packaging, and Manufacturing Technology, Part C, 20(2), 121-132.

Stern Hal S., (1996). “Neural Networks in Applied Statistics”. Technometrics 38(3), 205-214.

Surh, R. and Baston, R. G. (2001). “Constrained Multivariate Loss Function Minimization”. Quality Engineering 13(3), 475-483.

Taguchi, G., (1986). “Introduction to Quality Engineering”, Asian Productivity Organization. Distributed by American Supplier Institute, Inc. Dearborn, MI 48126.

Tong, L. and Hsieh, K. (2000). “A Novel Approach of Applying Neural Networks to Optimize the Multi-response Problem”. Quality Engineering 13(1), 11-18.

Tong, L., Wang, C., Houng, J. and Chen, J. (2001-02). “Optimizing Dynamic Multi-response Problems Using the Dual-Response-Surface Method”. Quality Engineering 14(1), 115-125.

Townsend, N. W. and Tarassenko, L. (1999). “Estimation of Errors Bounds for Neural Networks Functions Approximations”. IEEE Transactions on Neural Networks. 10(2), 219-230.

Ventresca, C. (1993). “Continuous Process Improvement trough Designed Experiments and Multi-attribute Desirability Optimization” ISA Transactions 32, 51-64.

Vining, G.G. and Myers, R.H., (1990). “Combining Taguchi and Response Surface Philosophies: A Dual Response Approach”. Journal of Quality Technology, 22(1), 38-45.

Vining, G.G., (1998). "A Compromise Approach to Multi-response Optimization", Journal of Quality Technology, 30(4), 309-313.

Winston, Wayne L. (1995). “Introduction to Mathematical Programming: Applications and Algorithms”. 2nd Edition. International Thompson Publishing.

Wikipedia. “Artificial neural network”.http://en.wikipedia.org/wiki/ Artificial_neural_network. Active May 2008.

Wikipedia. “Levenberg-Marquardt algorithm”.http://en.wikipedia.org/wiki/Levenberg-Marquardt_algorithm. Active May 2008.

Zhang, Y. Hearn, G.E. and Sen, P. (1997). “A Multivariable Neural Controller for Automatic Ship Berthing”. IEEE Control Systems Magazine. 17 (4), 31-45.

APPENDIX A. MATLAB CODES FOR DATA GENERATION CASE 1

Code for data generation Case 1

```
% Ismael Torres-Pizarro re-programmed original version (totally revamped) on April/27/07.

% Example 1: Catapult Experiment, Quadratic Responses
% Enrique del Castillo "Multiple Process Optimization via Constrained Confidence Regions"
%Journal of Quality Technology Vol 28. No.1 Jan/96 pp61 to 70.

clear all
clc
colorordermatrix=[0 0 1; 0 1 0;1 0 0];
global Y1LSL Y1USL Y2LSL Y2USL Y1Target Y2Target buyy bsyy Buyy Bsyy ucon scon
coef1 coef2 coef3 VUB VLB VUB2 VLB2;

% SECTION I: DATA FOR PROBLEM 1.
% Original Function Coeficient Set for u(media=ucon) & s(sigma=scon).
% u and s are the output variables Y1 and Y2.

ucon=[84.88 15.29 0.24 18.8 -0.52 -1.8 0.39 0.22 3.60 -4.42];
scon=[4.53 1.84 4.28 3.73 1.16 4.4 0.94 1.2 0.73 3.49];

% Data set: Specification Limits and desired target value for Y1 (u) and Y2 (s).

Y1LSL=100;
Y1USL=150;
Y2LSL=0;
Y2USL=11;

Y1Target=Y1USL;
Y2Target=Y2LSL;

% SECTION II: OPTIMIZATION FOR ORIGINAL PROBLEM.
% Optimization of the Original Function:

VLB=[-1.68179,-1.68179,-1.68179];
VUB=[1.68179, 1.68179, 1.68179];
VLB2=[1,VLB(1),VLB(2),VLB(3),VLB(1)^2,VLB(2)^2,VLB(3)^2,VLB(1)*VLB(2),VLB(1
)*VLB(3),VLB(2)*VLB(3)];
VUB2=[1,VUB(1),VUB(2),VUB(3),VUB(1)^2,VUB(2)^2,VUB(3)^2,VUB(1)*VUB(2),VU
B(1)*VUB(3),VUB(2)*VUB(3)];

coef1=[1,0,0,0,0,0,0,0,0,0];
```

```
coef2=[0,1,0,0,0,0,0,0,0,0];
coef3=[0,0,1,0,0,0,0,0,0,0];

options=optimset('LargeScale','off');

buyy=ucon;
bsyy=scon;

[x,feval]=fmincon('totaloss',VLB,[],[],[],[],VLB,VUB,'coeffun',options);

% The optimal solution to this problem has been found at:
% (x contains the true optimum settings for the input variables, that is
% the values of the input variables x's that minimize the original function).
% The function value at the optimum solution is feval.

xx=[1, x(1), x(2), x(3), x(1)^2, x(2)^2, x(3)^2, x(1)*x(2), x(1)*x(3), x(2)*x(3)];
media =ucon*xx';
sigma= scon*xx';
minimumloss=feval,

%(SLossY1 and SLossY2 contain the minimum possible Loss for each output
%response variable: mean and variation)

SLossY1=(2/(Y1USL-Y1LSL))^2*(media-Y1Target)^2,
SLossY2=(2/(Y2USL-Y2LSL))^2*(sigma-Y2Target)^2,
totaminloss=SLossY1 + SLossY2,

save minimumloss x SLossY1 SLossY2 media sigma;

for rl=1:1000 %Execute the sampling experiment 1,000 times!!!

% SECTION III: CCD SIMULATION AND EXPERIMENTAL DATA GATHERING
(COMPUTER SIMULATED).
%P defines 3-elements input vectors (row vectors) that are the 3 input variables (factors)
value:
% This process simulates the execution of a designed experiment, central composite design,
% size 46 settings. The experiments' order is randomized.
% The media experiment would have a normal variation as specified by S1 and the sigma a
χ^2 with S2=dof parameter.
%for rl=1:3,1000

        o = [0  0       0
        -1      1       -1
```

0 0 -1.68179
0 0 0
0 0 0
0 0 1.68179
0 0 0
0 0 0
0 -1.68179 0
1 1 1
0 1.68179 0
1 -1 -1
0 0 0
-1 -1 1
-1 -1 -1
1 1 -1
0 0 0
-1 1 1
-1.68179 0 0
0 0 0
0 0 0
1.68179 0 0
1 -1 1
0 0 0
-1 1 -1
0 0 -1.68179
0 0 0
0 0 0
0 0 1.68179
0 0 0
0 0 0
0 -1.68179 0
1 1 1
0 1.68179 0
1 -1 -1
0 0 0
-1 -1 1
-1 -1 -1
1 1 -1
0 0 0
-1 1 1
-1.68179 0 0
0 0 0
0 0 0
1.68179 0 0

```
        1       -1      1];

        P = o';

% Creation of random variables errors to be added to the CCD.
        S1=(Y1USL-Y1LSL)/12;
        S2=max(round(((Y2USL^2-Y2LSL^2)/12)^2/2),1);  %VAR() for Y2

        rand('seed',sum(rand*clock));
%s2=rand(46,5);
        rand('seed',sum(rand*clock));
        s1=randn(46,5);

        ss1=s1*S1;

% Simulation Begins. The following section will calculate a [46,10] vector OX for the
% CCD experiment. Then, it will use OX to calculate the expected mean and sigma for the
% experiment (sample size = 5) and will add the random values S1 and S2 to the mean and
the sigma matrix
% respectively. Matrixes mediaO and sigmaO are used to accumulate the results.

%Calculation of the random variable for Y2

        clear s;
        for j=1:5;% Sample Size = 5
                for i=1:length(o);
                OX(i,:)=[1, o(i,1), o(i,2), o(i,3),o(i,1)^2, o(i,2)^2, o(i,3)^2, o(i,1)*o(i,2),
                o(i,1)*o(i,3), o(i,2)*o(i,3)];
                u(i)=sum(ucon.*OX(i,:))+ss1(i,j);
                mediaO(i,j)=u(i);
                rand('seed',sum(rand*clock));

                error(i,j)=0.056116*(log(rand)*-1)^(1/.363836)-.247758; %CHANGE HERE

                dchi(i)=sum(scon.*OX(i,:))^2 + (error(i,j));
                s(i)=dchi(i)^(.5);
                sigmaO(i,j)=s(i);
        end;end;

% Below the section will use the mediaO and sigmaO matrixes to calculate the expected
% Loss for the mean and the sigma. The Total Loss (obtaining by adding both mean and
sigma
% losses) is accumulated in matrix EI.
```

```
    for j=1:5;
        for i=1:length(o);
        SLossY1A(i,j)=(2/(Y1USL-Y1LSL))^2*(mediaO(i,j)-Y1Target)^2;
        SLossY2A(i,j)=(2/(Y2USL-Y2LSL))^2*(sigmaO(i,j)-Y2Target)^2;
        EI(i,j) = [SLossY1A(i,j) + SLossY2A(i,j)];
    end;end;

% The result of this simulation is a matrix (EI) containing the resulting Function Loss from
% the experiment run size 46. The process will use these values as the input in matrix T.

% T defines the associated targets (a column vectors with 46 rows each) that are the output
variable response.
%That is, T contains the Total Loss that would have been obtained if we ran the CCD
experiment
%by setting the input variables X1,X2 and X3 to each of the 46 inputs settings specified at P
%This is the raw data gathered from the execution of the experiment corresponding to the
%Total Loss Function Obtained when the 3-input factors vector is run.

    T = EI';  %For future use with the neural net

% SECTION IV: MULTIPLE REGRESSION ESTIMATION.
% Now we calculate the regression coefficients for the CCD run.
% Multiple Variable Regression Analysis for 50 experiments runs size = lenght(o)=46

    for i=1:length(o);
        xc(i,:)=[1, o(i,1), o(i,2), o(i,3),o(i,1)^2, o(i,2)^2, o(i,3)^2, o(i,1)*o(i,2),
        o(i,1)*o(i,3), o(i,2)*o(i,3)];
    end;

% bu and bs will contain the sets of 10 regression coefficients for the mean and the sigma
% whem mediaO and sigmaO represents the mean and sigma obtained from the simulation.
    for y=1:1,
        Buyy(:,y) = REGRESS(mediaO(:,y),xc,.05);
        Bsyy(:,y) = REGRESS(sigmaO(:,y),xc,.05);
    end;

% Optimization routine forthe Individual Regressions Equations:
% The Optreg matrix will contain the estimated optimum settings for the input variables,that
is
% the set of values of the inputs variables x's that minimize the
% regression equation value.
```

```
%Here we use the user-funtion TSLOSS that simply calculate the Total Loss for each
% regression equation.

% Matrix TsLossreg contains the Total Loss value of the regression equations
% This TsLossreg(i) would be the value estimated for the true minimum of the original
function
% mediareg and sigmareg contains the values of the mean and variation for each one of the
% estimated minimum values. RegSLossY1 and RegSLossY2 contains their respective loss.

    [xreg,feval]=fmincon('totaloss_reg',VLB,[],[],[],[],VLB,VUB,'coeffun_reg',options);
    xxreg=[1, xreg(1), xreg(2), xreg(3), xreg(1)^2, xreg(2)^2, xreg(3)^2, xreg(1)*xreg(2),
    xreg(1)*xreg(3), xreg(2)*xreg(3)];
    mediareg =xxreg*Buyy;
    sigmareg= xxreg*Bsyy;
    minimumlossreg=feval;

%(SLossY1 and SLossY2 contain the minimum possible Loss for each output
%response variable: mean and variation)

    SLossY1reg=(2/(Y1USL-Y1LSL))^2*(mediareg-Y1Target)^2;
    SLossY2reg=(2/(Y2USL-Y2LSL))^2*(sigmareg-Y2Target)^2;
    totaminlossreg=SLossY1reg + SLossY2reg;

    clc

% SECTION VA: ARTIFICIAL NEURAL NETWORK MODEL.

    net=newff(minmax(P),[10, 5],{'logsig','purelin'},'trainlm');
    net.trainParam.show = 5;
    net.trainParam.epochs = 1000;
    net.trainParam.goal = 1e-5;
    [net,tr]=train(net,P,T);

    rand('seed',sum(rand*clock));
    setrv1=rand(1000,1)*3.36358-1.68179;
    rand('seed',sum(rand*clock));
    setrv2=rand(1000,1)*3.36358-1.68179;
    rand('seed',sum(rand*clock));
    setrv3=rand(1000,1)*3.36358-1.68179;

    PP=[setrv1,setrv2,setrv3]';
    a = sim(net,PP);
    aprom=mean(a);
```

```
lossann=[aprom;PP(1,:);PP(2,:);PP(3,:)]';
[optiorder,optindex]=sort(lossann(:,1));
minimumlossann=max(min(aprom),0);
xxann=PP(:,optindex(1))';

for i=1:1;
        xxann_ext(i,:)=[1, xxann(i,1), xxann(i,2), xxann(i,3),xxann(i,1)^2,
        xxann(i,2)^2, xxann(i,3)^2, xxann(i,1)*xxann(i,2),
        xxann(i,1)*xxann(i,3),xxann(i,2)*xxann(i,3)];
        mediaann(i)=sum(ucon.*xxann_ext(i,:));
        sigmaann(i)=sum(scon.*xxann_ext(i,:));
        SLossY1ann(i)=(2/(Y1USL-Y1LSL))^2*(mediaann(i)-Y1Target)^2;
        SLossY2ann(i)=(2/(Y2USL-Y2LSL))^2*(sigmaann(i)-Y2Target)^2;
        totaminlossreg(i)=SLossY1ann(i) + SLossY2ann(i);
end;


% SECTION VB: REGRESION WITH RANDOM SEARCH METHOD.
    clear xxranreg;
    for i=1:size(PP',1);
        xxranreg(i,:)=[1, PP(1,i), PP(2,i), PP(3,i),PP(1,i)^2, PP(2,i)^2, PP(3,i)^2,
        PP(1,i)*PP(2,i), PP(1,i)*PP(3,i), PP(2,i)*PP(3,i)];
        mediaranreg(i)=xxranreg(i,:)*Buyy,;
        sigmaranreg(i)=xxranreg(i,:)*Bsyy;
        SLossranregY1(i)=(2/(Y1USL-Y1LSL))^2*(mediaranreg(i)-Y1Target)^2;
        SLossranregY2(i)=(2/(Y2USL-Y2LSL))^2*(sigmaranreg(i)-Y2Target)^2;
        TOTLOSSRANREG(i) = [SLossranregY1(i) + SLossranregY2(i)];
    end;

    ran_regreloss=[TOTLOSSRANREG;PP(1,:);PP(2,:);PP(3,:)]';
    [optiorder,optindex]=sort(ran_regreloss(:,1));
    totaminlossranreg=max(min(TOTLOSSRANREG),0);
    xxranreg=PP(:,optindex(1))';

    xxranreg_opt=[1, xxranreg(1), xxranreg(2), xxranreg(3),xxranreg(1)^2,
    xxranreg(2)^2, xxranreg(3)^2, xxranreg(1)*xxranreg(2), xxranreg(1)*xxranreg(3),
    xxranreg(2)*xxranreg(3)];
    mediaranreg_opt=xxranreg_opt*Buyy;
    sigmaranreg_opt=xxranreg_opt*Bsyy;
    SLossranregY1_opt=(2/(Y1USL-Y1LSL))^2*(mediaranreg_opt-Y1Target)^2;
    SLossranregY2_opt=(2/(Y2USL-Y2LSL))^2*(sigmaranreg_opt-Y2Target)^2;
    TOTLOSSRANREG_OPT = [SLossranregY1_opt + SLossranregY2_opt];
```

```
% SECTION VI: VALUES ACCUMULATION.

%Accumulation of values for the Optimum:
%disp('The Total Loss Estimation from the Optimum=');

        Loss(rl,:)=totaminloss;
%disp('The Response Estimation for X1 from the Optimum=');
        X1(rl,:)=x(1);
%disp('TheResponse Estimation for X2 from the Optimum=');
        X2(rl,:)=x(2);
%disp('The Response Estimation for X3 from the Optimum=');
        X3(rl,:)=x(3);
%disp('The Response Estimation for the Media from the Optimum=');
        Media(rl,:)=media;
%disp('The Response Estimation for the Sigma from the Optimum=');
        Sigma(rl,:)=sigma;
%disp('The Response Estimation for Media (Y1) Total Loss from the Optimum=');
        SLOSSY1(rl,:)=SLossY1;
%disp('The Response Estimation for Sigma (Y2) Total Loss from the Optimum=');
        SLOSSY2(rl,:)=SLossY2;

%Accumulation of values for the regression using the brute force method (Regresion2):

%disp('The Total Loss Estimation from the Regression2=');
        Lossranreg(rl,:)=totaminlossranreg;
%disp('The Response Estimation for X1 from the Regression2=');
        Xranreg1(rl,:)=xxranreg(1);
%disp('TheResponse Estimation for X2 from the Regression2=');
        Xranreg2(rl,:)=xxranreg(2);
  %disp('The Response Estimation for X3 from the Regression2=');
        Xranreg3(rl,:)=xxranreg(3);
%disp('The Response Estimation for the Media from the Regression2=');
        Mediaranreg(rl,:)=mediaranreg_opt;
%disp('The Response Estimation for the Sigma from the Regression2=');
        Sigmaranreg(rl,:)=sigmaranreg_opt;
%disp('The Response Estimation for Media (Y1) Total Loss from the Regression2=');
        SLOSSRanRegY1(rl,:)=SLossranregY1_opt;
%disp('The Response Estimation for Sigma (Y2) Total Loss from the Regression2=');
        SLOSSRanRegY2(rl,:)=SLossranregY2_opt;

%Accumulation of values for the ANN:
```

```
        Lossann(rl,:)=totaminlossreg;
%disp('The Response Estimation for X1 from the ANN=');
        Xann1(rl,:)=xxann(1);
%disp('TheResponse Estimation for X2 from the ANN=');
        Xann2(rl,:)=xxann(2);
%disp('The Response Estimation for X3 from the ANN=');
        Xann3(rl,:)=xxann(3);
%disp('The Response Estimation for the Media from the ANN=');
        Mediaann(rl,:)=mediaann;
%disp('The Response Estimation for the Sigma from the ANN=');
        Sigmaann(rl,:)=sigmaann;
%disp('The Response Estimation for Media (Y1) Total Loss from the ANN=');
        SLOSSannY1(rl,:)=SLossY1ann;
%disp('The Response Estimation for Sigma (Y2) Total Loss from the ANN=');
        SLOSSannY2(rl,:)=SLossY2ann;

%Accumulation of values for the Regression using optimization function:

%disp('The Total Loss Estimation from the Regression1=');
        Lossreg(rl,:)=minimumlossreg;
%disp('The Response Estimation for X1 from the Regression1=');
        Xreg1(rl,:)=xreg(1);
%disp('TheResponse Estimation for X2 from the Regression1=');
        Xreg2(rl,:)=xreg(2);
%disp('The Response Estimation for X3 from the Regression1=');
        Xreg3(rl,:)=xreg(3);
%disp('The Response Estimation for the Media from the Regression1=');
        Mediareg(rl,:)=mediareg;
%disp('The Response Estimation for the Sigma from the Regression1=');
        Sigmareg(rl,:)=sigmareg;
%disp('The Response Estimation for Media (Y1) Total Loss from the Regression1=');
        SLOSSRegY1(rl,:)=SLossY1reg;
%disp('The Response Estimation for Sigma (Y2) Total Loss from the Regression1=');
        SLOSSRegY2(rl,:)=SLossY2reg;

        disp(rl),
end;


X1,pause
Xreg1,pause
Xranreg1,pause
Xann1,pause
```

```
X2,pause
Xreg2,pause
Xranreg2,pause
Xann2,pause

X3,pause
Xreg3,pause
Xranreg3,pause
Xann3,pause

PP(1,:)',pause
PP(2,:)',pause
PP(3,:)',pause
echo off
disp('End of Program')

function f  = totaloss(x)

%This function calculates the Total Standardize Loss Function for inputs given

global Y1LSL Y1USL Y2LSL Y2USL Y1Target Y2Target buyy bsyy;

options=foptions;

Y1=buyy*[1, x(1), x(2), x(3), x(1)^2, x(2)^2, x(3)^2, x(1)*x(2), x(1)*x(3), x(2)*x(3)]';
Y2=bsyy*[1, x(1), x(2), x(3), x(1)^2, x(2)^2, x(3)^2, x(1)*x(2), x(1)*x(3), x(2)*x(3)]';

f = 4*((Y1-Y1Target)^2/(Y1USL-Y1LSL)^2 + (Y2-Y2Target)^2/(Y2USL-Y2LSL)^2);

function [c,ceq]  = coeffun(x)

global Y1LSL Y1USL Y2LSL Y2USL Y1Target Y2Target buyy bsyy coef1 coef2 coef3
VUB VLB c;

Y1=buyy*[1, x(1), x(2), x(3), x(1)^2, x(2)^2, x(3)^2, x(1)*x(2), x(1)*x(3), x(2)*x(3)]';
Y2=bsyy*[1, x(1), x(2), x(3), x(1)^2, x(2)^2, x(3)^2, x(1)*x(2), x(1)*x(3), x(2)*x(3)]';
X1=coef1*[1, x(1), x(2), x(3), x(1)^2, x(2)^2, x(3)^2, x(1)*x(2), x(1)*x(3), x(2)*x(3)]';
X2=coef2*[1, x(1), x(2), x(3), x(1)^2, x(2)^2, x(3)^2, x(1)*x(2), x(1)*x(3), x(2)*x(3)]';
X3=coef3*[1, x(1), x(2), x(3), x(1)^2, x(2)^2, x(3)^2, x(1)*x(2), x(1)*x(3), x(2)*x(3)]';

c= [Y1-Y1USL;-Y1+Y1LSL;Y2-Y2USL;-Y2+Y2LSL;X1-VUB(1);-X1+VLB(1);X2-
VUB(2);-X2+VLB(2);X3-VUB(3);-X3+VLB(3)]';
```

```
ceq=[];

function f  = totaloss_reg(x)

%This function calculates the Total Standardize Loss Function for inputs given

global Y1LSL Y1USL Y2LSL Y2USL Y1Target Y2Target buyy bsyy  Buyy Bsyy;

options=foptions;

Y1=Buyy'*[1, x(1), x(2), x(3), x(1)^2, x(2)^2, x(3)^2, x(1)*x(2), x(1)*x(3), x(2)*x(3)]';
Y2=Bsyy'*[1, x(1), x(2), x(3), x(1)^2, x(2)^2, x(3)^2, x(1)*x(2), x(1)*x(3), x(2)*x(3)]';

f= 4*((Y1-Y1Target)^2/(Y1USL-Y1LSL)^2 + (Y2-Y2Target)^2/(Y2USL-Y2LSL)^2);

function [c,ceq]  = coeffun_reg(x)

global Y1LSL Y1USL Y2LSL Y2USL Y1Target Y2Target buyy bsyy coef1 coef2 coef3
VUB VLB c Buyy Bsyy;


Y1=Buyy'*[1, x(1), x(2), x(3), x(1)^2, x(2)^2, x(3)^2, x(1)*x(2), x(1)*x(3), x(2)*x(3)]';
Y2=Bsyy'*[1, x(1), x(2), x(3), x(1)^2, x(2)^2, x(3)^2, x(1)*x(2), x(1)*x(3), x(2)*x(3)]';
X1=coef1*[1, x(1), x(2), x(3), x(1)^2, x(2)^2, x(3)^2, x(1)*x(2), x(1)*x(3), x(2)*x(3)]';
X2=coef2*[1, x(1), x(2), x(3), x(1)^2, x(2)^2, x(3)^2, x(1)*x(2), x(1)*x(3), x(2)*x(3)]';
X3=coef3*[1, x(1), x(2), x(3), x(1)^2, x(2)^2, x(3)^2, x(1)*x(2), x(1)*x(3), x(2)*x(3)]';

c= [Y1-Y1USL;-Y1+Y1LSL;Y2-Y2USL;-Y2+Y2LSL;X1-VUB(1);-X1+VLB(1);X2-
VUB(2);-X2+VLB(2);X3-VUB(3);-X3+VLB(3)]';
ceq=[];
```

APPENDIX B. MATLAB CODES FOR DATA GENERATION CASE 2

Code for data generation Case 2

```
% Ismael Torres-Pizarro re-programmed original version (totally revamped) on April/27/07.


% Example 2: Chemical Experiment, Linear & Quadratic Responses
% Enrique del Castillo "Multiple Process Optimization via Constrained Confidence Regions"
% Journal of Quality Technology Vol 28. No.1 Jan/96 pp61 to 70.
clc
colorordermatrix=[0 0 1; 0 1 0;1 0 0];
```

```
global Y1LSL Y1USL Y2LSL Y2USL Y1Target Y2Target buyy bsyy Buyy Bsyy ucon scon
coef1 coef2  VUB VLB VUB2 VLB2;

% SECTION I: DATA FOR PROBLEM 1.
% Original Function Coeficient Set for u(media=ucon) & s(sigma=scon).
% u and s are the output variables Y1 and Y2.
ucon=[760.5 50.9 154.8 -4.4 -76.0 -24.7];
scon=[19.6 6.31 6.28 0 0 0];

% Data set: Specification Limits and desired target value for Y1 (u) and Y2 (s).
Y1LSL=800;
Y1USL=950;
Y2LSL=0;
Y2USL=35;

Y1Target=Y1USL;
Y2Target=Y2LSL;

% SECTION II: OPTIMIZATION FOR ORIGINAL PROBLEM.
% Optimization of the Original Function:

VLB=[-1.414,1.414];
VUB=[1.414, 1.414];
VLB2=[1,VLB(1),VLB(2),VLB(1)^2,VLB(2)^2,VLB(1)*VLB(2)];
VUB2=[1,VUB(1),VUB(2),VUB(1)^2,VUB(2)^2,VUB(1)*VUB(2)];
coef1=[1,0,0,0,0,0];
coef2=[0,1,0,0,0,0];

options=optimset('LargeScale','off');

buyy=ucon;
bsyy=scon;

[x,feval]=fmincon('totalossB',VLB,[],[],[],[],VLB,VUB,'coeffunB',options);
% The optimal solution to this problem has been found at:
% (x contains the true optimum settings for the input variables,that is
% the values of the input variables x's that minimize the original function).
% The function value at the optimum solution is feval.
xx=[1, x(1), x(2),  x(1)^2, x(2)^2, x(1)*x(2)];
media =ucon*xx';
sigma= scon*xx';
minimumloss=feval,
```

```
%(SLossY1 and SLossY2 contain the minimum possible Loss for each output
%response variable: mean and variation)

SLossY1=(2/(Y1USL-Y1LSL))^2*(media-Y1Target)^2,
SLossY2=(2/(Y2USL-Y2LSL))^2*(sigma-Y2Target)^2,
totaminloss=SLossY1 + SLossY2,

save minimumloss x SLossY1 SLossY2 media sigma;

for rl=1:1000 %Execute the sampling experiment 1,000 times!!!

% SECTION III: CCD SIMULATION AND EXPERIMENTAL DATA GATHERING
(COMPUTER SIMULATED).
%P defines 2-elements input vectors (row vectors) that are the 2 input variables (factors)
value:
% This process simulates the execution of a designed experiment, central composite design,
% size 13 settings. The experiments order is randomized.
% The media experiment would have a normal variation as specified by S1 and the sigma a
χ^2 with S2=dof parameter.
%for rl=1:3,1000

        o = [-1 -1; 1 1; 0 0;-1 1; 0 0;0 0; -1 -1; 0 0; 0 0; 0 1.414;-1.414 0;1.414 0;0 -1.414];

        P = o';

% Creation of random variables errors to be added to the CCD.
        S1=(Y1USL-Y1LSL)/12;
        S2=max(round(((Y2USL^2-Y2LSL^2)/12)^2/2),1);  %VAR() for Y2

        rand('seed',sum(rand*clock));
%s2=rand(46,5);
        rand('seed',sum(rand*clock));
        s1=randn(13,5);

        ss1=s1*S1;

% Simulation Begins. The following section will calculate a [13,6] vector OX for the
% CCD experiment. Then, it will use OX to calculate the expected mean and sigma for the
% experiment (sample size = 5) and will add the random values S1 and S2 to the mean and
the sigma matrix
% respectively. Matrixes mediaO and sigmaO are used to accumulate the results.

%Calculation of the random variable for Y2
```

```
clear s;
for j=1:5;% Sample Size = 5
        for i=1:length(o);
                OX(i,:)=[1, o(i,1), o(i,2), o(i,1)^2, o(i,2)^2, o(i,1)*o(i,2)];
                u(i)=sum(ucon.*OX(i,:))+ss1(i,j);
                mediaO(i,j)=u(i);
                rand('seed',sum(rand*clock));

                error(i,j)=1.294657*(log(rand)*-1)^(1/.64072) -1.79774; %CHANGE
                HERE

                dchi(i)=sum(scon.*OX(i,:))^2 + (error(i,j));
                s(i)=dchi(i)^(.5);
                sigmaO(i,j)=s(i);
end;end;

% Below the section will use the mediaO and sigmaO matrixes to calculate the expected
% Loss for the mean and the sigma. The Total Loss (obtaining by adding both mean and
sigma
% losses) is accumulated in matrix EI.

for j=1:5;
        for i=1:length(o);
                SLossY1A(i,j)=(2/(Y1USL-Y1LSL))^2*(mediaO(i,j)-Y1Target)^2;
                SLossY2A(i,j)=(2/(Y2USL-Y2LSL))^2*(sigmaO(i,j)-Y2Target)^2;
                EI(i,j) = [SLossY1A(i,j) + SLossY2A(i,j)];
end;end;

% The result of this simulation is a matrix (EI) containing the resulting Function Loss from
% the experiment run size 46. The process will use these values as the input in matrix T.

%T defines the associated targets (a column vectors with 13 rows each) that are the output
variable response.
%That is, T contains the Total Loss that would have been obtained if we ran the CCD
experiment
%by setting the input variables X1,X2 and X3 to each of the 13 inputs settings specified at P
%This is the raw data gathered from the execution of the experiment corresponding to the
%Total Loss Function Obtained when the 3-input factors vector is run.

        T = EI';  %For future use with the neural net

% SECTION IV: MULTIPLE REGRESSION ESTIMATION.
```

```
% Now we calculate the regression coefficients for the CCD run.
% Multiple Variable Regression Analysis for 50 experiments runs size = lenght(o)=46

        for i=1:length(o);
                xc(i,:)=[1, o(i,1), o(i,2), o(i,1)^2, o(i,2)^2, o(i,1)*o(i,2)];
        end;

% bu and bs will contain the sets of 10 regression coefficients for the mean and the sigma
% whem mediaO and sigmaO represents the mean and sigma obtained from the simulation.
        for y=1:1,
                Buyy(:,y) = REGRESS(mediaO(:,y),xc,.05);
                Bsyy(:,y) = REGRESS(sigmaO(:,y),xc,.05);
        end;

% Optimization routine for the Individual Regressions Equations:
% The Optreg matrix will contain the estimated optimum settings for the input variables, that
is
% the set of values of the inputs variables x's that minimize the
% regression equation value.

%Here we use the user-funtion TSLOSS that simply calculate the Total Loss for each
% regression equation.

% Matrix TsLossreg contains the Total Loss value of the regression equations
% This TsLossreg(i) would be the value estimated for the true minimum of the original
function
% mediareg and sigmareg contains the values of the mean and variation for each one of the
% estimated minimum values. RegSLossY1 and RegSLossY2 contain their respective loss.

        [xreg,feval]=fmincon('totaloss_regB',VLB,[],[],[],[],VLB,VUB,'coeffun_regB',option
        s);
        xxreg=[1, xreg(1), xreg(2), xreg(1)^2, xreg(2)^2, xreg(1)*xreg(2)];
        mediareg =xxreg*Buyy;
        sigmareg= xxreg*Bsyy;
        minimumlossreg=feval;

%(SLossY1 and SLossY2 contain the minimum possible Loss for each output
%response variable: mean and variation)

        SLossY1reg=(2/(Y1USL-Y1LSL))^2*(mediareg-Y1Target)^2;
        SLossY2reg=(2/(Y2USL-Y2LSL))^2*(sigmareg-Y2Target)^2;
        totaminlossreg=SLossY1reg + SLossY2reg;
```

```
clc

% SECTION VA: ARTIFICIAL NEURAL NETWORK MODEL.

net=newff(minmax(P),[10, 5],{'logsig','purelin'},'trainlm');
net.trainParam.show = 5;
net.trainParam.epochs = 1000;
net.trainParam.goal = 1e-5;
[net,tr]=train(net,P,T);

rand('seed',sum(rand*clock));
setrv1=rand(1000,1)*2.828-1.414;
rand('seed',sum(rand*clock));
setrv2=rand(1000,1)*2.828-1.414;

PP=[setrv1,setrv2]';
a = sim(net,PP);
aprom=mean(a);

lossann=[aprom;PP(1,:);PP(2,:)]';
[optiorder,optindex]=sort(lossann(:,1));
minimumlossann=max(min(aprom),0);
xxann=PP(:,optindex(1))';

for i=1:1;
        xxann_ext(i,:)=[1, xxann(i,1), xxann(i,2), xxann(i,1)^2, xxann(i,2)^2,
        xxann(i,1)*xxann(i,2)];
        mediaann(i)=sum(ucon.*xxann_ext(i,:));
        sigmaann(i)=sum(scon.*xxann_ext(i,:));
        SLossY1ann(i)=(2/(Y1USL-Y1LSL))^2*(mediaann(i)-Y1Target)^2;
        SLossY2ann(i)=(2/(Y2USL-Y2LSL))^2*(sigmaann(i)-Y2Target)^2;
        totaminlossreg(i)=SLossY1ann(i) + SLossY2ann(i);
end;


% SECTION VB: REGRESION WITH RANDOM SEARCH METHOD.
clear xxranreg;
        for i=1:size(PP',1);
                xxranreg(i,:)=[1, PP(1,i), PP(2,i), PP(1,i)^2, PP(2,i)^2, PP(1,i)*PP(2,i)];
                mediaranreg(i)=xxranreg(i,:)*Buyy,;
                sigmaranreg(i)=xxranreg(i,:)*Bsyy;
                SLossranregY1(i)=(2/(Y1USL-Y1LSL))^2*(mediaranreg(i)-Y1Target)^2;
                SLossranregY2(i)=(2/(Y2USL-Y2LSL))^2*(sigmaranreg(i)-Y2Target)^2;
```

```
            TOTLOSSRANREG(i) = [SLossranregY1(i) + SLossranregY2(i)];
        end;

        ran_regreloss=[TOTLOSSRANREG;PP(1,:);PP(2,:)]';
        [optiorder,optindex]=sort(ran_regreloss(:,1));
        totaminlossranreg=max(min(TOTLOSSRANREG),0);
        xxranreg=PP(:,optindex(1))';

        xxranreg_opt=[1, xxranreg(1), xxranreg(2), xxranreg(1)^2, xxranreg(2)^2,
        xxranreg(1)*xxranreg(2)];
        mediaranreg_opt=xxranreg_opt*Buyy;
        sigmaranreg_opt=xxranreg_opt*Bsyy;
        SLossranregY1_opt=(2/(Y1USL-Y1LSL))^2*(mediaranreg_opt-Y1Target)^2;
        SLossranregY2_opt=(2/(Y2USL-Y2LSL))^2*(sigmaranreg_opt-Y2Target)^2;
        TOTLOSSRANREG_OPT = [SLossranregY1_opt + SLossranregY2_opt];

% SECTION VI: VALUES ACCUMULATION.

%Accumulation of values for the Optimum:
%disp('The Total Loss Estimation from the Optimum=');
        Loss(rl,:)=totaminloss;
%disp('The Response Estimation for X1 from the Optimum=');
        X1(rl,:)=x(1);
%disp('TheResponse Estimation for X2 from the Optimum=');
        X2(rl,:)=x(2);
%disp('The Response Estimation for the Media from the Optimum=');
        Media(rl,:)=media;
%disp('The Response Estimation for the Sigma from the Optimum=');
        Sigma(rl,:)=sigma;
%disp('The Response Estimation for Media (Y1) Total Loss from the Optimum=');
        SLOSSY1(rl,:)=SLossY1;
%disp('The Response Estimation for Sigma (Y2) Total Loss from the Optimum=');
        SLOSSY2(rl,:)=SLossY2;

%Accumulation of values for the regression using the brute force method (Regresion2):

%disp('The Total Loss Estimation from the Regression2=');
        Lossranreg(rl,:)=totaminlossranreg;
%disp('The Response Estimation for X1 from the Regression2=');
        Xranreg1(rl,:)=xxranreg(1);
 %disp('TheResponse Estimation for X2 from the Regression2=');
        Xranreg2(rl,:)=xxranreg(2);
%disp('The Response Estimation for the Media from the Regression2=');
```

```
        Mediaranreg(rl,:)=mediaranreg_opt;
%disp('The Response Estimation for the Sigma from the Regression2=');
        Sigmaranreg(rl,:)=sigmaranreg_opt;
%disp('The Response Estimation for Media (Y1) Total Loss from the Regression2=');
        SLOSSRanRegY1(rl,:)=SLossranregY1_opt;
%disp('The Response Estimation for Sigma (Y2) Total Loss from the Regression2=');
        SLOSSRanRegY2(rl,:)=SLossranregY2_opt;

%Accumulation of values for the ANN:

        Lossann(rl,:)=totaminlossreg;
%disp('The Response Estimation for X1 from the ANN=');
        Xann1(rl,:)=xxann(1);
%disp('TheResponse Estimation for X2 from the ANN=');
        Xann2(rl,:)=xxann(2);
%disp('The Response Estimation for the Media from the ANN=');
        Mediaann(rl,:)=mediaann;
%disp('The Response Estimation for the Sigma from the ANN=');
        Sigmaann(rl,:)=sigmaann;
%disp('The Response Estimation for Media (Y1) Total Loss from the ANN=');
        SLOSSannY1(rl,:)=SLossY1ann;
%disp('The Response Estimation for Sigma (Y2) Total Loss from the ANN=');
        SLOSSannY2(rl,:)=SLossY2ann;

%Accumulation of values for the Regression using optimization function:

%disp('The Total Loss Estimation from the Regression1=');
        Lossreg(rl,:)=minimumlossreg;
%disp('The Response Estimation for X1 from the Regression1=');
        Xreg1(rl,:)=xreg(1);
%disp('TheResponse Estimation for X2 from the Regression1=');
        Xreg2(rl,:)=xreg(2);

%disp('The Response Estimation for the Media from the Regression1=');
        Mediareg(rl,:)=mediareg;
%disp('The Response Estimation for the Sigma from the Regression1=');
        Sigmareg(rl,:)=sigmareg;
%disp('The Response Estimation for Media (Y1) Total Loss from the Regression1=');
        SLOSSRegY1(rl,:)=SLossY1reg;
%disp('The Response Estimation for Sigma (Y2) Total Loss from the Regression1=');
        SLOSSRegY2(rl,:)=SLossY2reg;

        disp(rl),
```

```
end;


X1,pause
Xreg1,pause
Xranreg1,pause
Xann1,pause

X2,pause
Xreg2,pause
Xranreg2,pause
Xann2,pause

PP(1,:)',pause
PP(2,:)',pause
echo off
disp('End of Program')

function f  = totaloss(x)

%This function calculates the Total Standardize Loss Function for inputs given

global Y1LSL Y1USL Y2LSL Y2USL Y1Target Y2Target buyy bsyy;

options=foptions;

Y1=buyy*[1, x(1), x(2), x(3), x(1)^2, x(2)^2, x(3)^2, x(1)*x(2), x(1)*x(3), x(2)*x(3)]';
Y2=bsyy*[1, x(1), x(2), x(3), x(1)^2, x(2)^2, x(3)^2, x(1)*x(2), x(1)*x(3), x(2)*x(3)]';

f= 4*((Y1-Y1Target)^2/(Y1USL-Y1LSL)^2 + (Y2-Y2Target)^2/(Y2USL-Y2LSL)^2);

function [c,ceq]  = coeffun(x)

global Y1LSL Y1USL Y2LSL Y2USL Y1Target Y2Target buyy bsyy coef1 coef2 coef3
VUB VLB c;

Y1=buyy*[1, x(1), x(2), x(3), x(1)^2, x(2)^2, x(3)^2, x(1)*x(2), x(1)*x(3), x(2)*x(3)]';
Y2=bsyy*[1, x(1), x(2), x(3), x(1)^2, x(2)^2, x(3)^2, x(1)*x(2), x(1)*x(3), x(2)*x(3)]';
X1=coef1*[1, x(1), x(2), x(3), x(1)^2, x(2)^2, x(3)^2, x(1)*x(2), x(1)*x(3), x(2)*x(3)]';
X2=coef2*[1, x(1), x(2), x(3), x(1)^2, x(2)^2, x(3)^2, x(1)*x(2), x(1)*x(3), x(2)*x(3)]';
X3=coef3*[1, x(1), x(2), x(3), x(1)^2, x(2)^2, x(3)^2, x(1)*x(2), x(1)*x(3), x(2)*x(3)]';
```

```
c= [Y1-Y1USL;-Y1+Y1LSL;Y2-Y2USL;-Y2+Y2LSL;X1-VUB(1);-X1+VLB(1);X2-
VUB(2);-X2+VLB(2);X3-VUB(3);-X3+VLB(3)]';
ceq=[];

function f  = totaloss_reg(x)

%This function calculates the Total Standardize Loss Function for inputs given

global Y1LSL Y1USL Y2LSL Y2USL Y1Target Y2Target buyy bsyy  Buyy Bsyy;

options=foptions;

Y1=Buyy'*[1, x(1), x(2), x(3), x(1)^2, x(2)^2, x(3)^2, x(1)*x(2), x(1)*x(3), x(2)*x(3)]';
Y2=Bsyy'*[1, x(1), x(2), x(3), x(1)^2, x(2)^2, x(3)^2, x(1)*x(2), x(1)*x(3), x(2)*x(3)]';

f = 4*((Y1-Y1Target)^2/(Y1USL-Y1LSL)^2 + (Y2-Y2Target)^2/(Y2USL-Y2LSL)^2);

function [c,ceq]  = coeffun_reg(x)

global Y1LSL Y1USL Y2LSL Y2USL Y1Target Y2Target buyy bsyy coef1 coef2 coef3
VUB VLB c Buyy Bsyy;


Y1=Buyy'*[1, x(1), x(2), x(3), x(1)^2, x(2)^2, x(3)^2, x(1)*x(2), x(1)*x(3), x(2)*x(3)]';
Y2=Bsyy'*[1, x(1), x(2), x(3), x(1)^2, x(2)^2, x(3)^2, x(1)*x(2), x(1)*x(3), x(2)*x(3)]';
X1=coef1*[1, x(1), x(2), x(3), x(1)^2, x(2)^2, x(3)^2, x(1)*x(2), x(1)*x(3), x(2)*x(3)]';
X2=coef2*[1, x(1), x(2), x(3), x(1)^2, x(2)^2, x(3)^2, x(1)*x(2), x(1)*x(3), x(2)*x(3)]';
X3=coef3*[1, x(1), x(2), x(3), x(1)^2, x(2)^2, x(3)^2, x(1)*x(2), x(1)*x(3), x(2)*x(3)]';

c= [Y1-Y1USL;-Y1+Y1LSL;Y2-Y2USL;-Y2+Y2LSL;X1-VUB(1);-X1+VLB(1);X2-
VUB(2);-X2+VLB(2);X3-VUB(3);-X3+VLB(3)]';
ceq=[];
```

APPENDIX C. MATLAB CODES FOR DATA GENERATION CASE 3

Code for data generation Case 3

```
% Ismael Torres-Pizarro re-programmed original version (totally revamped) on March/25/07.


% Example 3: Chemical Experiment, All Linear Responses
% Enrique del Castillo "Multiple Process Optimization via Constrained Confidence Regions"
% Journal of Quality Technology Vol 28. No.1 Jan/96 pp61 to 70.
```

```
clc
colorordermatrix=[0 0 1; 0 1 0;1 0 0];
global Y1LSL Y1USL Y2LSL Y2USL Y1Target Y2Target buyy bsyy Buyy Bsyy ucon scon
coef1 coef2 coef3 VUB VLB VUB2 VLB2;

% SECTION I: DATA FOR PROBLEM 1.
% Original Function Coeficient Set for u(media=ucon) & s(sigma=scon).
% u and s are the output variables Y1 and Y2.
ucon=[711.0 50.9 154.8];
scon=[19.26 6.31 6.28];

% Data set: Specification Limits and desired target value for Y1 (u) and Y2 (s).
Y1LSL=800;
Y1USL=950;
Y2LSL=0;
Y2USL=35;

Y1Target=Y1USL;
Y2Target=Y2LSL;

% SECTION II: OPTIMIZATION FOR ORIGINAL PROBLEM.
% Optimization of the Original Function:

VLB=[-1.414,-1.414];
VUB=[1.414, 1.414];
VLB2=[1,VLB(1),VLB(2),VLB(1)*VLB(2)];
VUB2=[1,VUB(1),VUB(2),VUB(1)*VUB(2)];
coef1=[1,0,0,0,0,0];
coef2=[0,1,0,0,0,0];

options=optimset('LargeScale','off');

buyy=ucon;
bsyy=scon;

[x,feval]=fmincon('totalossB',VLB,[],[],[],[],VLB,VUB,'coeffunB',options);
% The optimal solution to this problem has been found at:
% (x contains the true optimum settings for the input variables,that is
% the values of the input variables x's that minimize the original function).
% The function value at the optimum solution is feval.
xx=[1, x(1), x(2), x(1)*x(2)];
media =ucon*xx';
sigma= scon*xx';
```

```
minimumloss=feval,

%(SLossY1 and SLossY2 contain the minimum possible Loss for each output
%response variable: mean and variation)

SLossY1=(2/(Y1USL-Y1LSL))^2*(media-Y1Target)^2,
SLossY2=(2/(Y2USL-Y2LSL))^2*(sigma-Y2Target)^2,
totaminloss=SLossY1 + SLossY2,

save minimumloss x SLossY1 SLossY2 media sigma;

for rl=1:1000 %Execute the sampling experiment 1,000 times!!!

% SECTION III: CCD SIMULATION AND EXPERIMENTAL DATA GATHERING
(COMPUTER SIMULATED).
%P defines 2-elements input vectors (row vectors) that are the 2 input variables (factors)
value:
% This process simulates the execution of a designed experiment, central composite design,
% size 13 settings. The experiments order is randomized.
% The media experiment would have a normal variation as specified by S1 and the sigma a
χ^2 with S2=dof parameter.
%for rl=1:3,1000

        o = [-1 -1; 1 1; 0 0;-1 1; 0 0;0 0; -1 -1; 0 0; 0 0; 0 1.414;-1.414 0;1.414 0;0 -1.414];

        P = o';

% Creation of random variables errors to be added to the CCD.
        S1=(Y1USL-Y1LSL)/12;
        S2=max(round(((Y2USL^2-Y2LSL^2)/12)^2/2),1);  %VAR() for Y2

        rand('seed',sum(rand*clock));
%s2=rand(46,5);
        rand('seed',sum(rand*clock));
        s1=randn(13,5);

        ss1=s1*S1;

% Simulation Begins. The following section will calculate a [13,6] vector OX for the
% CCD experiment. Then, it will use OX to calculate the expected mean and sigma for the
% experiment (sample size = 5) and will add the random values S1 and S2 to the mean and
the sigma matrix
% respectively. Matrixes mediaO and sigmaO are used to accumulate the results.
```

```
%Calculation of the random variable for Y2

        clear s;
        for j=1:5;% Sample Size = 5
                for i=1:length(o);
                        OX(i,:)=[1, o(i,1), o(i,2), o(i,1)^2, o(i,2)^2, o(i,1)*o(i,2)];
                        u(i)=sum(ucon.*OX(i,:))+ss1(i,j);
                        mediaO(i,j)=u(i);
                        rand('seed',sum(rand*clock));

                        error(i,j)= 0.838157*(log(rand)*-1)^(1/0.542526) -1.45774;
                        %CHANGE HERE

                        dchi(i)=sum(scon.*OX(i,:))^2 + (error(i,j));
                        s(i)=dchi(i)^(.5);
                        sigmaO(i,j)=s(i);
        end;end;

% Below the section will use the mediaO and sigmaO matrixes to calculate the expected
% Loss for the mean and the sigma. The Total Loss (obtaining by adding both mean and
sigma
% losses) is accumulated in matrix EI.
        for j=1:5;
                for i=1:length(o);
                        SLossY1A(i,j)=(2/(Y1USL-Y1LSL))^2*(mediaO(i,j)-Y1Target)^2;
                        SLossY2A(i,j)=(2/(Y2USL-Y2LSL))^2*(sigmaO(i,j)-Y2Target)^2;
                        EI(i,j) = [SLossY1A(i,j) + SLossY2A(i,j)];
        end;end;

% The result of this simulation is a matrix (EI) containing the resulting Function Loss from
% the experiment run size 46. The process will use these values as the input in matrix T.

%T defines the associated targets (a column vectors with 13 rows each) that are the output
variable response.
%That is, T contains the Total Loss that would have been obtained if we ran the CCD
experiment
%by setting the input variables X1,X2 and X3 to each of the 13 inputs settings specified at P
%This is the raw data gathered from the execution of the experiment corresponding to the
% Total Loss Function Obtained when the 3-input factors vector is run.

        T = EI';  %For future use with the neural net
```

```
% SECTION IV: MULTIPLE REGRESSION ESTIMATION.
% Now we calculate the regression coefficients for the CCD run.
% Multiple Variable Regression Analysis for 50 experiments runs size = lenght(o)=46

        for i=1:length(o);
                xc(i,:)=[1, o(i,1), o(i,2), o(i,1)*o(i,2)];
        end;

% bu and bs will contain the sets of 10 regression coefficients for the mean and the sigma
% whem mediaO and sigmaO represents the mean and sigma obtained from the simulation.
        for y=1:1,
                Buyy(:,y) = REGRESS(mediaO(:,y),xc,.05);
                Bsyy(:,y) = REGRESS(sigmaO(:,y),xc,.05);
        end;

% Optimization routine for the Individual Regression Equations:
% The Optreg matrix will contain the estimated optimum settings for the input variables, that
is
% the set of values of the inputs variables x's that minimize the
% regression equation value.

%Here we use the user-funtion TSLOSS that simply calculate the Total Loss for each
% regression equation.

% Matrix TsLossreg contains the Total Loss value of the regression equations
% This TsLossreg(i) would be the value estimated for the true minimum of the original
function
% mediareg and sigmareg contains the values of the mean and variation for each one of the
% estimated minimum values. RegSLossY1 and RegSLossY2 contain their respective loss.

        [xreg,feval]=fmincon('totaloss_regB',VLB,[],[],[],[],VLB,VUB,'coeffun_regB',option
        s);
        xxreg=[1, xreg(1), xreg(2), xreg(1)^2, xreg(2)^2, xreg(1)*xreg(2)];
        mediareg =xxreg*Buyy;
        sigmareg= xxreg*Bsyy;
        minimumlossreg=feval;

%(SLossY1 and SLossY2 contain the minimum possible Loss for each output
%response variable: mean and variation)

        SLossY1reg=(2/(Y1USL-Y1LSL))^2*(mediareg-Y1Target)^2;
        SLossY2reg=(2/(Y2USL-Y2LSL))^2*(sigmareg-Y2Target)^2;
        totaminlossreg=SLossY1reg + SLossY2reg;
```

```
clc

% SECTION VA: ARTIFICIAL NEURAL NETWORK MODEL.

net=newff(minmax(P),[10, 5],{'logsig','purelin'},'trainlm');
net.trainParam.show = 5;
net.trainParam.epochs = 1000;
net.trainParam.goal = 1e-5;
[net,tr]=train(net,P,T);

rand('seed',sum(rand*clock));
setrv1=rand(1000,1)*2.828-1.414;
rand('seed',sum(rand*clock));
setrv2=rand(1000,1)*2.828-1.414;

PP=[setrv1,setrv2]';
a = sim(net,PP);
aprom=mean(a);

lossann=[aprom;PP(1,:);PP(2,:)]';
[optiorder,optindex]=sort(lossann(:,1));
minimumlossann=max(min(aprom),0);
xxann=PP(:,optindex(1))';

for i=1:1;
        xxann_ext(i,:)=[1, xxann(i,1), xxann(i,2), xxann(i,1)^2, xxann(i,2)^2,
        xxann(i,1)*xxann(i,2)];
        mediaann(i)=sum(ucon.*xxann_ext(i,:));
        sigmaann(i)=sum(scon.*xxann_ext(i,:));
        SLossY1ann(i)=(2/(Y1USL-Y1LSL))^2*(mediaann(i)-Y1Target)^2;
        SLossY2ann(i)=(2/(Y2USL-Y2LSL))^2*(sigmaann(i)-Y2Target)^2;
        totaminlossreg(i)=SLossY1ann(i) + SLossY2ann(i);
end;


% SECTION VB: REGRESION WITH RANDOM SEARCH METHOD.
    clear xxranreg;
    for i=1:size(PP',1);
        xxranreg(i,:)=[1, PP(1,i), PP(2,i), PP(1,i)^2, PP(2,i)^2, PP(1,i)*PP(2,i)];
        mediaranreg(i)=xxranreg(i,:)*Buyy,;
        sigmaranreg(i)=xxranreg(i,:)*Bsyy;
        SLossranregY1(i)=(2/(Y1USL-Y1LSL))^2*(mediaranreg(i)-Y1Target)^2;
```

```
            SLossranregY2(i)=(2/(Y2USL-Y2LSL))^2*(sigmaranreg(i)-Y2Target)^2;
            TOTLOSSRANREG(i) = [SLossranregY1(i) + SLossranregY2(i)];
        end;

        ran_regreloss=[TOTLOSSRANREG;PP(1,:);PP(2,:)]';
        [optiorder,optindex]=sort(ran_regreloss(:,1));
        totaminlossranreg=max(min(TOTLOSSRANREG),0);
        xxranreg=PP(:,optindex(1))';

        xxranreg_opt=[1, xxranreg(1), xxranreg(2), xxranreg(1)^2, xxranreg(2)^2,
        xxranreg(1)*xxranreg(2)];
        mediaranreg_opt=xxranreg_opt*Buyy;
        sigmaranreg_opt=xxranreg_opt*Bsyy;
        SLossranregY1_opt=(2/(Y1USL-Y1LSL))^2*(mediaranreg_opt-Y1Target)^2;
        SLossranregY2_opt=(2/(Y2USL-Y2LSL))^2*(sigmaranreg_opt-Y2Target)^2;
        TOTLOSSRANREG_OPT = [SLossranregY1_opt + SLossranregY2_opt];

% SECTION VI: VALUES ACCUMULATION.

%Accumulation of values for the Optimum:
%disp('The Total Loss Estimation from the Optimum=');
        Loss(rl,:)=totaminloss;
%disp('The Response Estimation for X1 from the Optimum=');
        X1(rl,:)=x(1);
%disp('TheResponse Estimation for X2 from the Optimum=');
        X2(rl,:)=x(2);
%disp('The Response Estimation for the Media from the Optimum=');
        Media(rl,:)=media;
%disp('The Response Estimation for the Sigma from the Optimum=');
        Sigma(rl,:)=sigma;
%disp('The Response Estimation for Media (Y1) Total Loss from the Optimum=');
        SLOSSY1(rl,:)=SLossY1;
%disp('The Response Estimation for Sigma (Y2) Total Loss from the Optimum=');
        SLOSSY2(rl,:)=SLossY2;

%Accumulation of values for the regression using the brute force method (Regresion2):

%disp('The Total Loss Estimation from the Regression2=');
        Lossranreg(rl,:)=totaminlossranreg;
%disp('The Response Estimation for X1 from the Regression2=');
        Xranreg1(rl,:)=xxranreg(1);
%disp('TheResponse Estimation for X2 from the Regression2=');
        Xranreg2(rl,:)=xxranreg(2);
```

```
%disp('The Response Estimation for the Media from the Regression2=');
        Mediaranreg(rl,:)=mediaranreg_opt;
%disp('The Response Estimation for the Sigma from the Regression2=');
        Sigmaranreg(rl,:)=sigmaranreg_opt;
%disp('The Response Estimation for Media (Y1) Total Loss from the Regression2=');
        SLOSSRanRegY1(rl,:)=SLossranregY1_opt;
%disp('The Response Estimation for Sigma (Y2) Total Loss from the Regression2=');
        SLOSSRanRegY2(rl,:)=SLossranregY2_opt;

%Accumulation of values for the ANN:

        Lossann(rl,:)=totaminlossreg;
%disp('The Response Estimation for X1 from the ANN=');
        Xann1(rl,:)=xxann(1);
%disp('TheResponse Estimation for X2 from the ANN=');
        Xann2(rl,:)=xxann(2);
%disp('The Response Estimation for the Media from the ANN=');
        Mediaann(rl,:)=mediaann;
%disp('The Response Estimation for the Sigma from the ANN=');
        Sigmaann(rl,:)=sigmaann;
%disp('The Response Estimation for Media (Y1) Total Loss from the ANN=');
        SLOSSannY1(rl,:)=SLossY1ann;
%disp('The Response Estimation for Sigma (Y2) Total Loss from the ANN=');
        SLOSSannY2(rl,:)=SLossY2ann;

%Accumulation of values for the Regression using optimization function:

%disp('The Total Loss Estimation from the Regression1=');
        Lossreg(rl,:)=minimumlossreg;
%disp('The Response Estimation for X1 from the Regression1=');
        Xreg1(rl,:)=xreg(1);
%disp('TheResponse Estimation for X2 from the Regression1=');
        Xreg2(rl,:)=xreg(2);

%disp('The Response Estimation for the Media from the Regression1=');
        Mediareg(rl,:)=mediareg;
%disp('The Response Estimation for the Sigma from the Regression1=');
        Sigmareg(rl,:)=sigmareg;
%disp('The Response Estimation for Media (Y1) Total Loss from the Regression1=');
        SLOSSRegY1(rl,:)=SLossY1reg;
%disp('The Response Estimation for Sigma (Y2) Total Loss from the Regression1=');
        SLOSSRegY2(rl,:)=SLossY2reg;
```

```
        disp(rl),
end;


X1,pause
Xreg1,pause
Xranreg1,pause
Xann1,pause

X2,pause
Xreg2,pause
Xranreg2,pause
Xann2,pause

PP(1,:)',pause
PP(2,:)',pause
echo off
disp('End of Program')

function f  = totaloss(x)

%This function calculates the Total Standardize Loss Function for inputs given

global Y1LSL Y1USL Y2LSL Y2USL Y1Target Y2Target buyy bsyy;

options=foptions;

Y1=buyy*[1, x(1), x(2), x(3), x(1)^2, x(2)^2, x(3)^2, x(1)*x(2), x(1)*x(3), x(2)*x(3)]';
Y2=bsyy*[1, x(1), x(2), x(3), x(1)^2, x(2)^2, x(3)^2, x(1)*x(2), x(1)*x(3), x(2)*x(3)]';

f = 4*((Y1-Y1Target)^2/(Y1USL-Y1LSL)^2 + (Y2-Y2Target)^2/(Y2USL-Y2LSL)^2);

function [c,ceq]  = coeffun(x)

global Y1LSL Y1USL Y2LSL Y2USL Y1Target Y2Target buyy bsyy coef1 coef2 coef3
VUB VLB c;

Y1=buyy*[1, x(1), x(2), x(3), x(1)^2, x(2)^2, x(3)^2, x(1)*x(2), x(1)*x(3), x(2)*x(3)]';
Y2=bsyy*[1, x(1), x(2), x(3), x(1)^2, x(2)^2, x(3)^2, x(1)*x(2), x(1)*x(3), x(2)*x(3)]';
X1=coef1*[1, x(1), x(2), x(3), x(1)^2, x(2)^2, x(3)^2, x(1)*x(2), x(1)*x(3), x(2)*x(3)]';
X2=coef2*[1, x(1), x(2), x(3), x(1)^2, x(2)^2, x(3)^2, x(1)*x(2), x(1)*x(3), x(2)*x(3)]';
X3=coef3*[1, x(1), x(2), x(3), x(1)^2, x(2)^2, x(3)^2, x(1)*x(2), x(1)*x(3), x(2)*x(3)]';
```

```
c= [Y1-Y1USL;-Y1+Y1LSL;Y2-Y2USL;-Y2+Y2LSL;X1-VUB(1);-X1+VLB(1);X2-
VUB(2);-X2+VLB(2);X3-VUB(3);-X3+VLB(3)]';
ceq=[];

function f  = totaloss_reg(x)

%This function calculates the Total Standardize Loss Function for inputs given

global Y1LSL Y1USL Y2LSL Y2USL Y1Target Y2Target buyy bsyy  Buyy Bsyy;

options=foptions;

Y1=Buyy'*[1, x(1), x(2), x(3), x(1)^2, x(2)^2, x(3)^2, x(1)*x(2), x(1)*x(3), x(2)*x(3)]';
Y2=Bsyy'*[1, x(1), x(2), x(3), x(1)^2, x(2)^2, x(3)^2, x(1)*x(2), x(1)*x(3), x(2)*x(3)]';

f = 4*((Y1-Y1Target)^2/(Y1USL-Y1LSL)^2 + (Y2-Y2Target)^2/(Y2USL-Y2LSL)^2);

function [c,ceq]  = coeffun_reg(x)

global Y1LSL Y1USL Y2LSL Y2USL Y1Target Y2Target buyy bsyy coef1 coef2 coef3
VUB VLB c Buyy Bsyy;


Y1=Buyy'*[1, x(1), x(2), x(3), x(1)^2, x(2)^2, x(3)^2, x(1)*x(2), x(1)*x(3), x(2)*x(3)]';
Y2=Bsyy'*[1, x(1), x(2), x(3), x(1)^2, x(2)^2, x(3)^2, x(1)*x(2), x(1)*x(3), x(2)*x(3)]';
X1=coef1*[1, x(1), x(2), x(3), x(1)^2, x(2)^2, x(3)^2, x(1)*x(2), x(1)*x(3), x(2)*x(3)]';
X2=coef2*[1, x(1), x(2), x(3), x(1)^2, x(2)^2, x(3)^2, x(1)*x(2), x(1)*x(3), x(2)*x(3)]';
X3=coef3*[1, x(1), x(2), x(3), x(1)^2, x(2)^2, x(3)^2, x(1)*x(2), x(1)*x(3), x(2)*x(3)]';

c= [Y1-Y1USL;-Y1+Y1LSL;Y2-Y2USL;-Y2+Y2LSL;X1-VUB(1);-X1+VLB(1);X2-
VUB(2);-X2+VLB(2);X3-VUB(3);-X3+VLB(3)]';
ceq=[];
```

APPENDIX D. MATLAB CODES FOR DATA GENERATION CASE 4

Code for data generation Case 4

```
% Ismael Torres-Pizarro re-programed original version (totally revamped) on April/27/07.

% Example 1: Speed, Pressure & Distance Experiment, Quadratic Responses, Target is Best
% Kwang-Jae Kim & Dennis K.J. Lin "Dual Response Surface Optimization: A Fuzzy
Modeling Approach"
% Journal of Quality Technology Vol 30. No.1 Jan/98 pp1 to 10.
```

```
clear all
clc
colorordermatrix=[0 0 1; 0 1 0;1 0 0];
global Y1LSL Y1USL Y2LSL Y2USL Y1Target Y2Target buyy bsyy Buyy Bsyy ucon scon
coef1 coef2 coef3 VUB VLB VUB2 VLB2;

% SECTION I: DATA FOR PROBLEM 1.
% Original Function Coeficient Set for u(media=ucon) & s(sigma=scon).
% u and s are the output variables Y1 and Y2.
ucon=[327.6 177.0 109.4 131.5 32.0 -22.4 -29.1 66.0 75.5 43.6];
scon=[34.9 11.5 15.3 29.2 4.2 -1.3 16.8 7.7 5.1 14.1];

% Data set: Specification Limits and desired target value for Y1 (u) and Y2 (s).
Y1LSL=490;
Y1USL=510;
Y2LSL=1500^.5;
Y2USL=2100^.5;

Y1Target=(Y1USL+Y1LSL)/2;
Y2Target=Y2LSL;

% SECTION II: OPTIMIZATION FOR ORIGINAL PROBLEM.
% Optimization of the Original Function:

VLB=[-1.68179,-1.68179,-1.68179];
VUB=[1.68179, 1.68179, 1.68179];
VLB2=[1,VLB(1),VLB(2),VLB(3),VLB(1)^2,VLB(2)^2,VLB(3)^2,VLB(1)*VLB(2),VLB(1
)*VLB(3),VLB(2)*VLB(3)];
VUB2=[1,VUB(1),VUB(2),VUB(3),VUB(1)^2,VUB(2)^2,VUB(3)^2,VUB(1)*VUB(2),VU
B(1)*VUB(3),VUB(2)*VUB(3)];
coef1=[1,0,0,0,0,0,0,0,0,0];
coef2=[0,1,0,0,0,0,0,0,0,0];
coef3=[0,0,1,0,0,0,0,0,0,0];

options=optimset('LargeScale','off');

buyy=ucon;
bsyy=scon;

[x,feval]=fmincon('totaloss',VLB,[],[],[],[],VUB,VUB,'coeffun',options);
  % The optimal solution to this problem has been found at:
% (x contains the true optimum settings for the input variables,that is
% the values of the input variables x's that minimize the original function).
```

```
% The function value at the optimum solution is feval.
xx=[1, x(1), x(2), x(3), x(1)^2, x(2)^2, x(3)^2, x(1)*x(2), x(1)*x(3), x(2)*x(3)];
media =ucon*xx';
sigma= scon*xx';
minimumloss=feval,

%(SLossY1 and SLossY2 contain the minimum possible Loss for each output
  %response variable: mean and variation)

SLossY1=(2/(Y1USL-Y1LSL))^2*(media-Y1Target)^2,
SLossY2=(2/(Y2USL-Y2LSL))^2*(sigma-Y2Target)^2,
totaminloss=SLossY1 + SLossY2,

save minimumloss x SLossY1 SLossY2 media sigma;

for rl=1:5 %Execute the sampling experiment 1,000 times!!!

% SECTION III: CCD SIMULATION AND EXPERIMENTAL DATA GATHERING
(COMPUTER SIMULATED).
%P defines 3-elements input vectors (row vectors) that are the 3 input variables (factors)
value:
% This process simulates the execution of a designed experiment, central composite design,
% size 46 settings. The experiments order is randomized.
% The media experiment would have a normal variation as specified by S1 and the sigma a
χ^2 with S2=dof parameter.
%for rl=1:3,1000

        o = [0  0       0
        -1      1       -1
        0       0       -1.68179
        0       0       0
        0       0       0
        0       0       1.68179
        0       0       0
        0       0       0
        0       -1.68179        0
        1       1       1
        0       1.68179         0
        1       -1      -1
        0       0       0
        -1      -1      1
        -1      -1      -1
        1       1       -1
```

```
        0       0       0
        -1      1       1
        -1.68179        0       0
        0       0       0
        0       0       0
        1.68179         0       0
        1       -1      1
        0       0       0
        -1      1       -1
        0       0       -1.68179
        0       0       0
        0       0       0
        0       0       1.68179
        0       0       0
        0       0       0
        0       -1.68179        0
        1       1       1
        0       1.68179         0
        1       -1      -1
        0       0       0
        -1      -1      1
        -1      -1      -1
        1       1       -1
        0       0       0
        -1      1       1
        -1.68179        0       0
        0       0       0
        0       0       0
        1.68179         0       0
        1       -1      1];

        P = o';

% Creation of random variable errors to be added to the CCD.
        S1=(Y1USL-Y1LSL)/12;
        S2=max(round(((Y2USL^2-Y2LSL^2)/12)^2/2),1);  %VAR() for Y2

        rand('seed',sum(rand*clock));
%s2=rand(46,5);
        rand('seed',sum(rand*clock));
        s1=randn(46,5);

        ss1=s1*S1;
```

```
% Simulation Begins. The following section will calculate a [46,10] vector OX for the
% CCD experiment. Then, it will use OX to calculate the expected mean and sigma for the
% experiment (sample size = 5) and will add the random values S1 and S2 to the mean and
the sigma matrix
% respectively. Matrixes mediaO and sigmaO are used to accumulate the results.

%Calculation of the random variable for Y2

        clear s;
        for j=1:5;% Sample Size = 5
                for i=1:length(o);
                OX(i,:)=[1, o(i,1), o(i,2), o(i,3),o(i,1)^2, o(i,2)^2, o(i,3)^2, o(i,1)*o(i,2),
                o(i,1)*o(i,3), o(i,2)*o(i,3)];
                u(i)=sum(ucon.*OX(i,:))+ss1(i,j);
                mediaO(i,j)=u(i);
                rand('seed',sum(rand*clock));

                error(i,j)=.136277*(log(rand)*-1)^(1/.504702)-.267926 %CHANGE HERE

                dchi(i)=sum(scon.*OX(i,:))^2 + (error(i,j));
                s(i)=dchi(i)^(.5);
                sigmaO(i,j)=s(i);
        end;end;

% Below the section will use the mediaO and sigmaO matrixes to calculate the expected
% Loss for the mean and the sigma. The Total Loss (obtaining by adding both mean and
sigma
% losses) is accumulated in matrix EI.

        for j=1:5;
                for i=1:length(o);
                        SLossY1A(i,j)=(2/(Y1USL-Y1LSL))^2*(mediaO(i,j)-Y1Target)^2;
                        SLossY2A(i,j)=(2/(Y2USL-Y2LSL))^2*(sigmaO(i,j)-Y2Target)^2;
                        EI(i,j) = [SLossY1A(i,j) + SLossY2A(i,j)];
        end;end;

% The result of this simulation is a matrix (EI) containing the resulting Function Loss from
% the experiment run size 46. The process will use these values as the input in matrix T.

%T defines the associated targets (a column vectors with 46 rows each) that are the output
variable response.
```

```
%That is, T contains the Total Loss that would have been obtained if we ran the CCD
experiment
%by setting the input variables X1,X2 and X3 to each of the 46 inputs settings specified at P
%This is the raw data gathered from the execution of the experiment corresponding to the
%Total Loss Function Obtained when the 3-input factors vector is run.

        T = EI';  %For future use with the neural net

% SECTION IV: MULTIPLE REGRESSION ESTIMATION.
% Now we calculate the regression coefficients for the CCD run.
% Multiple Variable Regression Analysis for 50 experiments runs size = lenght(o)=46

        for i=1:length(o);
                xc(i,:)=[1, o(i,1), o(i,2), o(i,3),o(i,1)^2, o(i,2)^2, o(i,3)^2, o(i,1)*o(i,2),
                o(i,1)*o(i,3), o(i,2)*o(i,3)];
        end;

% bu and bs will contain the sets of 10 regression coefficients for the mean and the sigma
% whem mediaO and sigmaO represents the mean and sigma obtained from the simulation.
        for y=1:1,
                Buyy(:,y) = REGRESS(mediaO(:,y),xc,.05);
                Bsyy(:,y) = REGRESS(sigmaO(:,y),xc,.05);
        end;

% Optimization routine for the Individual Regressions Equations:
% The Optreg matrix will contain the estimated optimum settings for the input variables, that
is
% the set of values of the inputs variables x's that minimize the
% regression equation value.

%Here we use the user-funtion TSLOSS that simply calculate the Total Loss for each
% regression equation.

% Matrix TsLossreg contains the Total Loss value of the regression equations
% This TsLossreg(i) would be the value estimated for the true minimum of the original
function
% mediareg and sigmareg contains the values of the mean and variation for each one of the
% estimated minimum values. RegSLossY1 and RegSLossY2 contains their respective loss.

        [xreg,feval]=fmincon('totaloss_reg',VLB,[],[],[],[],VLB,VUB,'coeffun_reg',options);
        xxreg=[1, xreg(1), xreg(2), xreg(3), xreg(1)^2, xreg(2)^2, xreg(3)^2, xreg(1)*xreg(2),
        xreg(1)*xreg(3), xreg(2)*xreg(3)];
        mediareg =xxreg*Buyy;
```

```
        sigmareg= xxreg*Bsyy;
        minimumlossreg=feval;

%(SLossY1 and SLossY2 contain the minimum possible Loss for each output
%response variable: mean and variation)

        SLossY1reg=(2/(Y1USL-Y1LSL))^2*(mediareg-Y1Target)^2;
        SLossY2reg=(2/(Y2USL-Y2LSL))^2*(sigmareg-Y2Target)^2;
        totaminlossreg=SLossY1reg + SLossY2reg;

        clc

% SECTION VA: ARTIFICIAL NEURAL NETWORK MODEL.

        net=newff(minmax(P),[10, 5],{'logsig','purelin'},'trainlm');
        net.trainParam.show = 5;
        net.trainParam.epochs = 1000;
        net.trainParam.goal = 1e-5;
        [net,tr]=train(net,P,T);

        rand('seed',sum(rand*clock));
        setrv1=rand(1000,1)*3.36358-1.68179;
        rand('seed',sum(rand*clock));
        setrv2=rand(1000,1)*3.36358-1.68179;
        rand('seed',sum(rand*clock));
        setrv3=rand(1000,1)*3.36358-1.68179;

        PP=[setrv1,setrv2,setrv3]';
        a = sim(net,PP);
        aprom=mean(a);

        lossann=[aprom;PP(1,:);PP(2,:);PP(3,:)]';
        [optiorder,optindex]=sort(lossann(:,1));
        minimumlossann=max(min(aprom),0);
        xxann=PP(:,optindex(1))';

        for i=1:1;
                xxann_ext(i,:)=[1, xxann(i,1), xxann(i,2), xxann(i,3),xxann(i,1)^2,
                xxann(i,2)^2, xxann(i,3)^2, xxann(i,1)*xxann(i,2),
                xxann(i,1)*xxann(i,3),xxann(i,2)*xxann(i,3)];
                mediaann(i)=sum(ucon.*xxann_ext(i,:));
                sigmaann(i)=sum(scon.*xxann_ext(i,:));
                SLossY1ann(i)=(2/(Y1USL-Y1LSL))^2*(mediaann(i)-Y1Target)^2;
```

```
            SLossY2ann(i)=(2/(Y2USL-Y2LSL))^2*(sigmaann(i)-Y2Target)^2;
            totaminlossreg(i)=SLossY1ann(i) + SLossY2ann(i);
        end;


% SECTION VB: REGRESION WITH RANDOM SEARCH METHOD.
        clear xxranreg;
        for i=1:size(PP',1);
            xxranreg(i,:)=[1, PP(1,i), PP(2,i), PP(3,i),PP(1,i)^2, PP(2,i)^2, PP(3,i)^2,
            PP(1,i)*PP(2,i), PP(1,i)*PP(3,i), PP(2,i)*PP(3,i)];
            mediaranreg(i)=xxranreg(i,:)*Buyy,;
            sigmaranreg(i)=xxranreg(i,:)*Bsyy;
            SLossranregY1(i)=(2/(Y1USL-Y1LSL))^2*(mediaranreg(i)-Y1Target)^2;
            SLossranregY2(i)=(2/(Y2USL-Y2LSL))^2*(sigmaranreg(i)-Y2Target)^2;
            TOTLOSSRANREG(i) = [SLossranregY1(i) + SLossranregY2(i)];
        end;

        ran_regreloss=[TOTLOSSRANREG;PP(1,:);PP(2,:);PP(3,:)]';
        [optiorder,optindex]=sort(ran_regreloss(:,1));
        totaminlossranreg=max(min(TOTLOSSRANREG),0);
        xxranreg=PP(:,optindex(1))';

        xxranreg_opt=[1, xxranreg(1), xxranreg(2), xxranreg(3),xxranreg(1)^2,
        xxranreg(2)^2, xxranreg(3)^2, xxranreg(1)*xxranreg(2), xxranreg(1)*xxranreg(3),
        xxranreg(2)*xxranreg(3)];
        mediaranreg_opt=xxranreg_opt*Buyy;
        sigmaranreg_opt=xxranreg_opt*Bsyy;
        SLossranregY1_opt=(2/(Y1USL-Y1LSL))^2*(mediaranreg_opt-Y1Target)^2;
        SLossranregY2_opt=(2/(Y2USL-Y2LSL))^2*(sigmaranreg_opt-Y2Target)^2;
        TOTLOSSRANREG_OPT = [SLossranregY1_opt + SLossranregY2_opt];

% SECTION VI: VALUES ACCUMULATION.

%Accumulation of values for the Optimum:
%disp('The Total Loss Estimation from the Optimum=');
        Loss(rl,:)=totaminloss;
%disp('The Response Estimation for X1 from the Optimum=');
        X1(rl,:)=x(1);
%disp('TheResponse Estimation for X2 from the Optimum=');
        X2(rl,:)=x(2);
%disp('The Response Estimation for X3 from the Optimum=');
        X3(rl,:)=x(3);
%disp('The Response Estimation for the Media from the Optimum=');
```

```
        Media(rl,:)=media;
%disp('The Response Estimation for the Sigma from the Optimum=');
        Sigma(rl,:)=sigma;
%disp('The Response Estimation for Media (Y1) Total Loss from the Optimum=');
        SLOSSY1(rl,:)=SLossY1;
%disp('The Response Estimation for Sigma (Y2) Total Loss from the Optimum=');
        SLOSSY2(rl,:)=SLossY2;

%Accumulation of values for the regression using the brute force method (Regresion2):

%disp('The Total Loss Estimation from the Regression2=');
        Lossranreg(rl,:)=totaminlossranreg;
%disp('The Response Estimation for X1 from the Regression2=');
        Xranreg1(rl,:)=xxranreg(1);
%disp('TheResponse Estimation for X2 from the Regression2=');
        Xranreg2(rl,:)=xxranreg(2);
%disp('The Response Estimation for X3 from the Regression2=');
        Xranreg3(rl,:)=xxranreg(3);
%disp('The Response Estimation for the Media from the Regression2=');
%disp('The Response Estimation for the Sigma from the Regression2=');
        Sigmaranreg(rl,:)=sigmaranreg_opt;
 %disp('The Response Estimation for Media (Y1) Total Loss from the Regression2=');
        SLOSSRanRegY1(rl,:)=SLossranregY1_opt;
%disp('The Response Estimation for Sigma (Y2) Total Loss from the Regression2=');
        SLOSSRanRegY2(rl,:)=SLossranregY2_opt;

%Accumulation of values for the ANN:

        Lossann(rl,:)=totaminlossreg;
%disp('The Response Estimation for X1 from the ANN=');
        Xann1(rl,:)=xxann(1);
%disp('TheResponse Estimation for X2 from the ANN=');
        Xann2(rl,:)=xxann(2);
%disp('The Response Estimation for X3 from the ANN=');
        Xann3(rl,:)=xxann(3);
%disp('The Response Estimation for the Media from the ANN=');
        Mediaann(rl,:)=mediaann;
%disp('The Response Estimation for the Sigma from the ANN=');
        Sigmaann(rl,:)=sigmaann;
%disp('The Response Estimation for Media (Y1) Total Loss from the ANN=');
        SLOSSannY1(rl,:)=SLossY1ann;
%disp('The Response Estimation for Sigma (Y2) Total Loss from the ANN=');
        SLOSSannY2(rl,:)=SLossY2ann;
```

```
%Accumulation of values for the Regression using optimization function:

%disp('The Total Loss Estimation from the Regression1=');
        Lossreg(rl,:)=minimumlossreg;
%disp('The Response Estimation for X1 from the Regression1=');
        Xreg1(rl,:)=xreg(1);
%disp('TheResponse Estimation for X2 from the Regression1=');
        Xreg2(rl,:)=xreg(2);
%disp('The Response Estimation for X3 from the Regression1=');
        Xreg3(rl,:)=xreg(3);
%disp('The Response Estimation for the Media from the Regression1=');
        Mediareg(rl,:)=mediareg;
%disp('The Response Estimation for the Sigma from the Regression1=');
        Sigmareg(rl,:)=sigmareg;
%disp('The Response Estimation for Media (Y1) Total Loss from the Regression1=');
        SLOSSRegY1(rl,:)=SLossY1reg;
%disp('The Response Estimation for Sigma (Y2) Total Loss from the Regression1=');
        SLOSSRegY2(rl,:)=SLossY2reg;

disp(rl),
end;


X1,pause
Xreg1,pause
Xranreg1,pause
Xann1,pause

X2,pause
Xreg2,pause
Xranreg2,pause
Xann2,pause

X3,pause
Xreg3,pause
Xranreg3,pause
Xann3,pause

PP(1,:)',pause
PP(2,:)',pause
PP(3,:)',pause
echo off
```

```
disp('End of Program')

function [c,ceq]  = coeffun(x)

global Y1LSL Y1USL Y2LSL Y2USL Y1Target Y2Target buyy bsyy coef1 coef2 coef3
VUB VLB c;

Y1=buyy*[1, x(1), x(2), x(3), x(1)^2, x(2)^2, x(3)^2, x(1)*x(2), x(1)*x(3), x(2)*x(3)]';
Y2=bsyy*[1, x(1), x(2), x(3), x(1)^2, x(2)^2, x(3)^2, x(1)*x(2), x(1)*x(3), x(2)*x(3)]';
X1=coef1*[1, x(1), x(2), x(3), x(1)^2, x(2)^2, x(3)^2, x(1)*x(2), x(1)*x(3), x(2)*x(3)]';
X2=coef2*[1, x(1), x(2), x(3), x(1)^2, x(2)^2, x(3)^2, x(1)*x(2), x(1)*x(3), x(2)*x(3)]';
X3=coef3*[1, x(1), x(2), x(3), x(1)^2, x(2)^2, x(3)^2, x(1)*x(2), x(1)*x(3), x(2)*x(3)]';

c= [Y1-Y1USL;-Y1+Y1LSL;Y2-Y2USL;-Y2+Y2LSL;X1-VUB(1);-X1+VLB(1);X2-
VUB(2);-X2+VLB(2);X3-VUB(3);-X3+VLB(3)]';
ceq=[];

function f  = totaloss(x)

%This function calculates the Total Standardize Loss Function for inputs given

global Y1LSL Y1USL Y2LSL Y2USL Y1Target Y2Target buyy bsyy;

options=foptions;

Y1=buyy*[1, x(1), x(2), x(3), x(1)^2, x(2)^2, x(3)^2, x(1)*x(2), x(1)*x(3), x(2)*x(3)]';
Y2=bsyy*[1, x(1), x(2), x(3), x(1)^2, x(2)^2, x(3)^2, x(1)*x(2), x(1)*x(3), x(2)*x(3)]';

f = 4*((Y1-Y1Target)^2/(Y1USL-Y1LSL)^2 + (Y2-Y2Target)^2/(Y2USL-Y2LSL)^2);

function [c,ceq]  = coeffun_reg(x)

global Y1LSL Y1USL Y2LSL Y2USL Y1Target Y2Target buyy bsyy coef1 coef2 coef3
VUB VLB c Buyy Bsyy;


Y1=Buyy'*[1, x(1), x(2), x(3), x(1)^2, x(2)^2, x(3)^2, x(1)*x(2), x(1)*x(3), x(2)*x(3)]';
Y2=Bsyy'*[1, x(1), x(2), x(3), x(1)^2, x(2)^2, x(3)^2, x(1)*x(2), x(1)*x(3), x(2)*x(3)]';
X1=coef1*[1, x(1), x(2), x(3), x(1)^2, x(2)^2, x(3)^2, x(1)*x(2), x(1)*x(3), x(2)*x(3)]';
X2=coef2*[1, x(1), x(2), x(3), x(1)^2, x(2)^2, x(3)^2, x(1)*x(2), x(1)*x(3), x(2)*x(3)]';
X3=coef3*[1, x(1), x(2), x(3), x(1)^2, x(2)^2, x(3)^2, x(1)*x(2), x(1)*x(3), x(2)*x(3)]';
```

```
c= [Y1-Y1USL;-Y1+Y1LSL;Y2-Y2USL;-Y2+Y2LSL;X1-VUB(1);-X1+VLB(1);X2-
VUB(2);-X2+VLB(2);X3-VUB(3);-X3+VLB(3)]';
ceq=[];

function f  = totaloss_reg(x)

%This function calculates the Total Standardize Loss Function for inputs given

global Y1LSL Y1USL Y2LSL Y2USL Y1Target Y2Target buyy bsyy  Buyy Bsyy;

options=foptions;

Y1=Buyy'*[1, x(1), x(2), x(3), x(1)^2, x(2)^2, x(3)^2, x(1)*x(2), x(1)*x(3), x(2)*x(3)]';
Y2=Bsyy'*[1, x(1), x(2), x(3), x(1)^2, x(2)^2, x(3)^2, x(1)*x(2), x(1)*x(3), x(2)*x(3)]';

f = 4*((Y1-Y1Target)^2/(Y1USL-Y1LSL)^2 + (Y2-Y2Target)^2/(Y2USL-Y2LSL)^2);
```