Mathematically Reduced Chemical Reaction Mechanism Using Neural Networks

Technical Progress Report

Period ending: 09/30/2005

Prepared by

Nelson Butuk
Principal Investigator
Department of Mathematics
Prairie View A & M University
Prairie View Texas. 77446-4189

December, 2005

DOE Grant Number: DE-FG26-03NT-41913

Office of Sponsored Programs
Prairie View A & M University
P. O. Box 667
Prairie View Texas 77446-0667

Disclaimer

Abstract

This is an annual technical report for the work done over the last year (period ending 9/30/2005) on the project titled "Mathematically Reduced Chemical Reaction Mechanism Using Neural Networks." The aim of the project is to develop an efficient chemistry model for combustion simulations. The reduced chemistry model will be developed mathematically without the need of having extensive knowledge of the chemistry involved. To aid in the development of the model, Neural Networks (NN) will be used via a new network topology know as Non-linear Principal Components Analysis (NPCA).

We report on the development of a novel procedure to speed up the training of NPCA. The same procedure termed L_2Boost can be used to increase the order of approximation of the Generalized Regression Neural Network (GRNN). It is pointed out that GRNN is a basic procedure for the emerging mesh free CFD. Also reported is an efficient simple approach of computing the derivatives of GRNN function approximation using complex variables or the Complex Step Method (CSM). The results presented demonstrate the significance of the methods developed and will be useful in many areas of applied science and engineering.

TABLE CONTENTS

Introduction

This is an annual technical report for the work done over the last year (period ending 9/30/2005) on the project titled "Mathematically Reduced Chemical Reaction Mechanism Using Neural Networks." The aim of the project is to develop an efficient chemistry model for combustion simulations. The reduced chemistry model will be developed mathematically without the need of having extensive knowledge of the chemistry involved. To aid in the development of the model, Neural Networks (NN) will be used via a new network topology know as Non-linear Principal Components Analysis (NPCA).

Many Combustion systems are modeled by very high-dimensional systems of non-linear differential equations. These equations often exhibit solutions which are un-evenly distributed in phase-space, and which may exist as circles, tori or other manifolds. It is desirable to approximate these isolated regions of the phase-space by a mathematical model of lower dimension than the dimension of the original ambient space. We propose to develop NPCA to accomplish this task using NN.

Given a data set $X \in R^n$ NPCA determines a reduction mapping

$$G: X \rightarrow Y$$

where the set $Y \in R^m$ has reduced dimension $m < n$. Y is then said to be a reduction of X. The inverse mapping reproduces the original data X from the reduced data set Y as

$$H: Y \rightarrow X$$

Hence, NPCA is a composition of mappings which is the same as the identity mapping, i.e.

$$H \circ G: X \rightarrow X$$

The NPCA is a NN topology with five layers, in which the input layer has the same number of nodes as the output layer. This allows the input values to be used as target output values of the network during training. The first part of the network approximates the mapping G. It contains the mapping layer. The middle layer is the bottle-neck layer consisting of the m nodes (i.e. desired reduced dimension m). The last part of the network implements the mapping H and contains the de-mapping layer. This layer takes the output of the middle layer and maps it onto the output layer. It is on the bottle-neck layer, that the reduced manifold of the chemistry is reconstructed.

The Objectives of this research are therefore:

1. Improve the training rate of the NPCA-NN algorithm developed previously.
2. Develop a method to determine optimum trajectory data of reaction mechanisms needed to use NPCA-NN.
3. Apply the NPCA-NN algorithm to the reduction of Dimethyl ether (DME) mechanism.
4. Couple the developed NPCA-NN model to the KIVA CFD code.
5. Test the CFD code on a few other simple sample mechanisms
6. Student Education in Computational Applied Mathematics

During the past year, we have spent considerable amount of time developing a detailed understanding of the mathematical theory of function approximators (or estimators). The various techniques developed for function approximation are all grounded in well developed mathematical procedures, which if well understood will guide in the development of new consistent application methods. Because of this main reason, we have reviewed the basic theory

underlying most function estimators. As a result of this review this report presents significant new results that will aid in the overall long term objectives of this work. An approach to improve the training speed of NPCA-NN will be described as well as recent published theoretical results to support the method. Also to be described is a new mesh free approach to modify our in-house developed CFD code in partial fulfillment of objective 4. The approach to be described, we believe will have significant impact to the emerging field of mesh free CFD. First to be described is some mathematical theory of function estimators.

In general, function estimators can be classified as being parametric or nonparametric. Most modern approaches, that include Neural Networks (NNs), the subject of this work fall under the category of nonparametric methods. Within the nonparametric category, we also classify methods according to the approach used in constructing the estimator. These include methods based on; (i) local averaging (ii) local modeling, and (iii) global modeling or least squares estimation. Examples of local averaging methods include; K-nearest neighbor (K-NN), Generalized Regression Neural Network (GRNN) also known as Nadaraya Watson (NW) estimator, and the partitioning estimator, as well as several kernel based estimators, Hastie et al. (2001). These methods fit a constant locally to the data hence the term "local averaging". In methods based on local modeling, instead of fitting a constant to the data, a more general function is fitted locally to the data. Examples of this approach include; local polynomial kernel estimators and the popular Moving Least Squares (MLS) estimator, used in emerging meshfree CFD, Liu (2003). Finaly, in the global modeling approach, data are fitted globally. Example methods include; orthogonal series estimators, cubic spline smoothers, and neural networks, the approach of this research. Wavelets are also classified under this category.

In our research to develop efficient neutral network based chemical reaction mechanism, reduction model, NPCA-NN, we have sought to develop a model that will incorporate some of the desirable properties of GRNN into the NPCA-NN model. In order to be able to do this, we have focused our study of the theory of function approximation on the GRNN. Below will be presented a summary of this theory, which will allow us to point out some of the characteristics of the estimator. As part of a masters thesis supported by this research, we have successfully demonstrated a new approach of estimating the derivatives of the GRNN approximated function. This new approach uses the Complex Step Method (CSM) to be described below, to compute accurate derivatives of the approximated function. This technique was demonstrated using the GRNN estimator, however, the approach is applicable to any of the nonparametric estimators mentioned above. Derivatives approximation is an important part of this research as they will be required in the coupling of developed NPCA-NN model to CFD, the 4^{th} objective of this research.

Compared to the traditional approaches of computing numerical derivatives using finite differencing or kernel differentiation, Mustafi (1978) and Schuster (1969), we show that the CSM approach is faster and much simpler to implement. Demonstrated also is a simple and fast boosting technique for GRNN that significantly improves the accuracy of first order derivatives. It is shown that this boosting is necessary because one of the arguments against GRNN is its "low order" of consistency. Consistency is one way of comparing estimators. The accuracy or desirability of an estimator can be based on the highest order polynomial that the estimator can represent "exactly". An estimator is then said to be consistent to order n, if n is the highest order

polynomial it can represent exactly. Because the GRNN fits a constant to data, it is of 0-order consistency. This has been the main argument used against the use of GRNN as an estimator, despite it being of low computational cost and simplicity of implementation. It is shown that with boosting this arguments is no longer valid and when combined with CSM makes GRNN attractive not only for this work but many application areas of science and engineering. For example the new emerging field of mesh free CFD and nano-computing.

In this report it is demonstrated the "boosting" of GRNN to increase its consistency and we also show that the CSM is superior to the traditional robust approach of kernel differentiation to obtain derivatives of approximated functions. It is considered that this successful demonstration of the superior advantages of computing the derivatives of GRNN using CSM is significant development resulting from this research. In CFD, GRNN is a basic method for the emerging mesh free methods, Liu (2003). One argument against these mesh free methods is the difficulty and time consuming approaches of computing derivatives necessary to solve the Navier-Stokes equations of fluid flow. In fact it is often necessary to introduce approximation methods of computing derivatives (due to the complexity of analytic differentiation methods) as is done in Onate, et al. (1996). By successfully demonstrating an efficient accurate approach of computing derivatives of GRNN, a solution for the major argument against mesh free methods will begin to emerge. For this reason, the new algorithm (that combines "boosting" and CSM) to be fully presented in the results section of this report, may be subject to Intellectual Property Protection (IPP), prior to public release.

The rest of this report is organized as follows; first we present a summary of the basic mathematical theory of GRNN and related kernel based methods. Next will be described a simple recently introduced booting technique to improve the consistency of GRNN. The relationship of this boosting technique to the hierarchical training of NPCA-NN (HT of NPCA-NN), 1st objective above will be pointed out. The approach of CSM will be given a brief summary. Methods of improving the computational efficiency of GRNN will be described as the report evolves. Finally, the main results of this work will be presented.

Basic Theory of GRNN

The presentation below can be found in many statistical books including Scott (1992) and Wand and Jones (1995). We have tried to present a simplified explanation of mathematical derivations and or proofs that should be accessible to many audiences across different fields.

In function approximation one considers a data vector (X,Y), where X is $\Re^d$ - valued and maybe fixed deterministic set of points or random set of points. This will result in what is popularly referred to as a fixed design model or a random design model. Y is random and $\Re$ -valued. Our interest is to specify a relationship between the dependent variable Y, and the independent variables X. That is we need to find a measurable function m: $\Re^d \rightarrow \Re$, such that m(X) is a "good approximation of Y" in some sense. We usually require that the L_2 risk or mean squared error of m

$$L_2 = E\left[\left|m(X)-Y\right|^2\right] \tag{1}$$

be as small as possible. The symbol E[u] is known as the expectation of u, Higgins and Keller-McNulty (1995). The expectation of a continuous variable u with density f(u) is

$$E[u] = \int u f(u) du \tag{2}$$

So our interest is to minimize the expectation of the squared distance between m(X) and Y. For a given data point (or instance of data) (x,y) our model becomes

$$y = m(x) + \varepsilon \tag{3}$$

where ε denotes an iid independent realizations of a random variable (error) with mean 0 and variance σ^2. Note that our data samples consists of n-pairs of observations $\{(X_1,Y_1)\ldots(X_n,Y_n)\}$. In this report we will interchangeably use subscripted upper case variables (X_i,Y_i) or just lower case variables (x,y) to refer to a single observation from the set of available data points (X,Y). The sampled data points are drawn from a population with a joint probability density function (pdf), f(x,y), whose form is unknown. In order to estimate this joint pdf, we need to be able to estimate the univariate pdf, f(x) of the independent variables X. The details of derivation of an estimate of f(x) from data follows.

In estimating f(x), we shall follow the approach first introduced by Parzen (1962). At a point x we want to be able to use local information as much as possible to estimate f. One simple classical estimator for f is the histogram. However, with the histogram, the estimated function is not smooth and it also has jumps which gives the estimation of f a step wise nature. Parzen addressed this issue by introducing a real positive kernel function K(u) to specify the way the local information is averaged by weighting. Where now the kernel function specifies the weights. The general kernel density estimator at a point x is

$$\hat{f}(x) = \frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{x - X_i}{h}\right) \tag{4}$$

where, h is termed the bandwidth of the kernel. A kernel function assigns weights to the contribution given by each data point, X_i to the kernel density estimator, depending on the proximity of X_i to x. Typically kernel functions are positive everywhere and symmetric about zero. K is usually a density such as the normal pdf. If f is multivariate and of dimension p, i.e. $x = (x_1 \ldots x_p)$, $X_i = (X_{i1} \ldots X_{ip})$, then a so called "product kernel" is used and the density estimate for f is now

$$\hat{f}(x) = \frac{1}{n}\sum_{i=1}^{n}\prod_{j=1}^{p}\frac{1}{h_j} K\left(\frac{x_j - X_{ij}}{h_j}\right) \tag{5}$$

We will now continue our derivation of the function estimator, $\hat{m}(x)$ of our data (X,Y). If we consider a single observation of the data, the conditional mean is

$$E\left[Y \middle| X = x\right] = m(x) \tag{6}$$

and the conditional variance

$$Var\left(Y \middle| X = x\right) = \sigma^2(x) \tag{7}$$

In order to simply analysis, as states above, we will assume that the variance is constant $\sigma^2(x) = \sigma^2$. From the basic definition of conditional expectation:

$$\mathrm{E}[Y \mid X = x] = \int y f(y \mid x) dx$$

$$= \frac{\int y f(x,y)dy}{\int f(x,y)dy} \tag{8}$$

As discussed above, we can now use a product kernel to approximate $f(x,y)$ the joint pdf.

$$\hat{f}(x,y) = \frac{1}{nh_x h_y}\sum_{i=1}^{n} K\left(\frac{x-X_i}{h_x}\right)K\left(\frac{y-Y_i}{h_y}\right) \tag{9}$$

From the definition of marginal density,

$$f(s) = \int f(s,t)dt \tag{10}$$

The denominator of (8) becomes

$$\int \hat{f}(x,y)dy = \frac{1}{nh_x h_y}\sum_{i=1}^{n} K\left(\frac{x-X_i}{h_x}\right)\int K\frac{(y-Y_i)}{h_y}dy \tag{11}$$

At this point it is convenient to introduce a scaled Kernel $K_h(x)$

$$K_h(x) = \frac{1}{h}K\left(\frac{x}{h}\right) \tag{12}$$

An intuitive interpretation of this scaling, is that if K(x) is the density of some random variable z, then $K_n(x)$ is the density of the scaled random variable hz, that is, h is the scaling parameter. In particular, if K(x) is the standard normal density (as used in this work) then h plays the role of the standard deviation. With this scaling we can now write.

$$\hat{f}(x,y) = \frac{1}{n}\sum_{i=1}^{n} K_{h_x}(x-X_i)K_{h_y}(y-Y_i) \tag{13}$$

Using the general definition of marginal density, equation (10) above, of variable X, it can be easily shown that the denominator of (8) reduces to the marginal density of x

$$\int \hat{f}(x,y)dy = \frac{1}{n}\sum_{i=i}^{n} K_{h_x}(x-X_i)\int K_{h_y}(y-Y_i)dy$$

$$= \frac{1}{n}\sum_{i=1}^{n} K_{h_x}(x-X_i) \tag{14}$$

Here we have used one of the properties of the Kernel

$$\int K_h(t)dt = 1 \tag{15}$$

Also it is required that

$$\int tK_h(t)dt = 0 \tag{16}$$

Implying an order-2 Kernel. A Kernel is classified as belonging to a certain order, if it satisfies certain moment conditions, Hardle (1990). Using (16) and letting $t=y-y_i$, it can be shown that

$$\int yK_h(y-Y_i)dy = Y_i \tag{17}$$

therefore, the numerator of (8) is

$$\int y\hat{f}(x,y)dy = \frac{1}{n}\sum_{i=1}^{n} Y_i K_{h_x}(x - X_i)$$

Our final function approximator becomes

$$\hat{m}(x) = \frac{\sum_{i=1}^{n} Y_i K_{h_x}(x - X_i)}{\sum_{i=1}^{n} K_{h_x}(x - X_i)} \tag{18}$$

This is the GRNN, as we have reported without detailed derivations in our past reports. In statistics literature this approximator is also known as Nadaraya Watson Kernel Regression estimator, Nadaraya, (1964) and Watson, (1964).

Our next question to consider is how good an estimator is GRNN?. To answer this question we have to determine the mean squared error (MSE) or the L_2-norm mentioned above (1). To do this, our first step is to decompose the MSE into the so called "Bias-Variance Decomposition". We now describe how this is done. In classical parametric estimation, it is customary to measure the performance of an estimator, $\hat{\theta}$, in comparison to the true value, θ, by the mean square error (MSE).

$$MSE(\hat{\theta}) = E[(\hat{\theta} - \theta)^2]$$

If we subtract E[$\hat{\theta}$] and add E[$\hat{\theta}$] to the term in parenthesis

$$MSE(\hat{\theta}) = E\left[(\hat{\theta} - E(\hat{\theta}) + E(\hat{\theta}) - \theta)^2\right]$$

Now let μ = E[$\hat{\theta}$]

$$MSE(\hat{\theta}) = E\left[(\hat{\theta} - \mu + \mu - \theta)^2\right]$$

$$= E\left[(\hat{\theta} - \mu)^2 + 2(\hat{\theta} - \mu)(\mu - \theta) + (\mu + \theta)^2\right]$$

Term by term expectation results in the middle term being zero

$$= E\left[(\hat{\theta} - \mu)^2 + (\mu + \theta)^2\right]$$

The first term inside the brackets is a random variable with mean zero and the second term is a constant. From definition, Higgins and Keller-McNulty (1995),

$$E\left[(c + z)^2\right] = c^2 + E\left[z^2\right]$$

where c is a constant and z is a random variable with zero mean and finite variance. Therefore

$$MSE(\hat{\theta}) = (\mu + \theta)^2 + E\left[(\hat{\theta} - \mu)^2\right]$$

replacing μ

$$MSE(\hat{\theta}) = (E[\hat{\theta}] + \theta)^2 + E\left[(\hat{\theta} - E[\hat{\theta}])^2\right]$$

finally

$$MSE(\hat{\theta}) = Bias^2(\hat{\theta}) + Var(\hat{\theta}) \tag{19}$$

This is the decomposition of MSE into variance and squared bias. This result tells us that in order to analyze a given estimator, via the MSE criteria, all that we need to do is to compute the expected mean and variance of the estimator. This will be done for the GRNN next.

First define two functionals S and R

$$S(g) = \int z^2 g(z)dz$$
$$R(g) = \int g^2(z)dz \tag{20}$$

recall that the GRNN equation above can be written as

$$\hat{m}(x) = \frac{\hat{r}(x)}{\hat{f}(x)} \tag{21}$$

In order to compute the expectation and variance of $\hat{m}(x)$, we will need to compute the values for $\hat{r}(x)$ and $\hat{f}(x)$ separately. These derivations are long and tedious. Here in this report we will present only the results and will describe detail derivations in a follow-up report. First the values for $\hat{f}(x)$ are

$$E\left[\hat{f}(x)\right] \simeq f(x) + \frac{f''(x)h^2}{2} S(K)$$

$$Var\left(\hat{f}(x)\right) \simeq \frac{R(K) f(x)}{nh} \tag{22}$$

The values for $\hat{r}(x)$ are

$$E\left[\hat{r}(x)\right] \simeq f(x)m(x) + h^2 S(K)\left[f'(x)m'(x) + \frac{1}{2} f''(x)m(x) + \frac{1}{2} m''(x) f(x) \right]$$
$$Var\left(\hat{r}(x)\right) \simeq \frac{R(K) f(x)}{nh}[\sigma^2 + m^2(x)] \tag{23}$$

where σ^2 is the variance in (3). Combining (22) and (23) to obtain values for $\hat{m}(x)$ is also more involved since you now have a ratio of two random variables. The final results are:

$$E\left[\hat{m}(x)\right] \simeq m(x) + \frac{1}{2} h^2 S(K)\left[m''(x) + 2m'(x)\frac{f'(x)}{f(x)} \right]$$
$$Var\left(\hat{m}(x)\right) \simeq \frac{\sigma^2 R(K)}{nhf(x)} \tag{24}$$

The MSE($\hat{m}(x)$) can now be written in the form of (19) as

$$MSE\left(\hat{m}(x)\right) \simeq \frac{1}{4} h^4 S^2(K)\left[m''(x) + 2m'(x)\frac{f'(x)}{f(x)} \right]^2 + \frac{\sigma^2 R(K)}{nhf(x)} \tag{25}$$

One of the beneficial results of performing the above tedious analysis for any given estimator is that we can now look at the final result , i.e equation (25) and make certain conclusions.

One of the main conclusions is made by inspecting the bias term (first term of 25). If the term f(x) is present, then it means that the estimator is dependent on the density of the design points. In other words, in the language of CFD, the estimator is not "mesh free". It is design points or grid points dependent. In this sense, the GRNN is not truly mesh free, despite being considered as the simplest basic method of all mesh free approaches. It is possible to construct, weight kernels, K, which will make an estimator truly mesh free. This is an area of active research in mesh free CFD.

The other conclusion to draw form the MSE, equation is on the accuracy (or consistency) of the estimator. In other words, how will the estimator converge to the true estimate of the unknown function, m(x) if at all ?. From inspecting (25) it is clear that the GRNN is consistent. This means that as the number of data points increases the error of approximation decreases. In the limit as $n\to\infty$, $h \to 0$ and nh should tend to ∞. This means that h should approach zero at a rate that is slower than order $O(n^{-1})$. A simple optimization of MSE to obtain an optimal h, will reveal the rate of convergence of MSE to zero. In the case of GRNN, the rate of convergence is of order $O(n^{-4/5})$, which as expected is slower than for most parametric estimators with rates of order $O(n^{-1})$.

Complex Variables Method of Derivatives (CSM)

Given a complex number z = x + iy, we can write the complex function $f(z)$ as

$$f(z) = u(x, y) + iv(x, y) \tag{26}$$

where u and v are real valued functions. Using the basic definition of derivative, we can write

$$f'(z) = \lim_{h\to 0} \frac{f(z+h) - f(z)}{h} \tag{27}$$

h can be both real or imaginary. Now as $h\to 0$ through real values, the limit $f'(z)$ must be the same for both approaches of h to zero. First if h is real,

$$\lim_{h\to 0} \frac{f(z+h) - f(z)}{h} = \lim_{h\to 0} \frac{u(x+h, y) - u(x, y)}{h} + i \lim_{h\to 0} \frac{v(x+h, y) - v(x, y)}{h} \tag{28}$$

$$= \frac{\partial u}{\partial x} + i \frac{\partial v}{\partial x} \tag{29}$$

Now if h is imaginary

$$\lim_{h\to 0} \frac{f(z+ih) - f(z)}{ih} = \lim_{h\to 0} \frac{u(x, y+h) - u(x, y)}{ih} + i \lim_{h\to 0} \frac{v(x, y+h) - v(x, y)}{ih} \tag{30}$$

$$= \frac{\partial v}{\partial y} - i \frac{\partial u}{\partial y} \tag{31}$$

Since the limit $f'(z)$ must be the same, we obtain

$$\frac{\partial u}{\partial x} = \frac{\partial v}{\partial y} \;\; and \;\; \frac{\partial u}{\partial y} = -\frac{\partial v}{\partial x} \tag{32}$$

Which are the Cauchy-Riemann equations. From the first equation

$$\frac{\partial u}{\partial x} = \lim_{h\to 0} \frac{v(x, y+h) - v(x, y)}{h} \tag{33}$$

and on the real axis y = 0, u(x) = f(x) and v(x) = 0, therefore for real functions

$$\frac{\partial u}{\partial x} = \lim_{h\to 0} \frac{v(x,h) - v(x)}{h} \tag{34}$$

which can now be written as

$$\frac{\partial f}{\partial x} = \lim_{h\to 0} \frac{\mathrm{Im}\left[f(x+ih)\right]}{h} \tag{35}$$

This equation is then the basis for the Complex Step Method (CSM) of derivatives. Any code that computes any given function can be made to compute the first derivatives of the function by simply converting the code to run using complex variables. In the case of GRNN, when the code is converted to run using complex variables, the number of operations to compute N values given n data points is of order O(kNn) (see below). Where k can be as high as 5 depending on the arithmetic operations used to compute the function, Smith (1998). It can be shown via a Taylor series expansion that the estimated derivatives are of $O(h^2)$, Anderson et.al., (2000).

Boosting

Here is described, a new novel algorithm for raising the order of consistency (reducing the bias) of the GRNN estimator. The algorithm is based on boosting and is known as L_2Boost. Basically boosting applies a B-steps algorithm to compute B function estimates by repeatedly applying a given method, called a weak learner (WL) to B different re-weighted samples. The estimates are then combined into a single estimate which is the final output.

Thus, the basic requirements of boosting is a WL and a method of assigning importance to data (i.e. a weighting strategy). The WL must have low variance. This is so that each boosting step makes a small, low variance step, iteratively reducing the bias of the estimator. This means that from the MSE equation for the GRNN above, a crude GRNN (or WL) will have a large bandwidth, h, which is larger than the optimal value. The other requirement for boosting, i.e the weighting scheme is inherent in the implementation of GRNN, via the kernel.

The boosting algorithm implemented in this report is the L_2Boost method recently described by Buhlmann and Yu, (2003). This method was applied to the GRNN, boosting its accuracy significantly. In L_2Boost, a WL is used to iteratively learn the residuals of the estimator in B-steps. The final result is a combination of the estimates of all of the B-steps. The steps of implementation are as follows:

Step 1
Apply the crude learner to available data to obtain the pilot estimate $\hat{m}_0$

$$\hat{m}_0(x) = GRNN(X_i, Y_i) \tag{36}$$

Step 2 repeat for b = 1 … B
Compute the residuals

$$\varepsilon_i = Y_i - \hat{m}_{b-1}(X_i) \tag{37}$$

i.e. use the previous estimate of $\hat{m}(x)$ to compute the residuals at design points X_i

Step 3
Learn (or fit) the residuals to the WL

$$\hat{r}_{b-1}(x) = GRNN(X_i, \varepsilon_i) \tag{38}$$

Step 4
Update the estimator

$$\hat{m}_b(x) = \hat{m}_{b-1}(x) + \hat{r}_{b-1}(x) \tag{39}$$

Step 5
Go back to step 2 and repeat loop until B, steps.

That is a brief basic description of L_2Boost for more details consult recent papers by Di Marzio and Taylor, (2004) and Buhlmann and Yu, (2003) as well as Friedman, (2001).

Boosting and Hierarchical Training of NPCA-NN (HT of NPCA-NN)

In our previous report, Butuk, (2002) we describe a novel method developed to speed up the training of NPCA-NN model. The method developed was termed Hierarchical Training of NPCA-NN (HT of NPCA-NN). At the time of developing HT of NPCA-NN, there appeared to be no theoretical justification as to why the method worked. This has now changed with the recent publication of Buhlmann and Yu, (2003) paper. It turns out that HT of NPCA-NN is only a slightly different implementation of L_2Boost. For example the WLs are the networks 5-8-2-8-5, 5-3-2-3-5 and 5-3-2-3-5 as described in that earlier report. Therefore HT of NPCA-NN was the first attempt to apply a form of L_2Boost to standard neural networks. With guidance of new results, this algorithm will now be easily improved. At that time, it was demonstrated that it resulted in significant improvement in the convergence of NPCA-NN algorithm.

Results and Discussions

This section of the report presents the numerical experimental results carried out on a test function. First to be described is the function used as a test function. Next will be described the implementation of GRNN to estimate the known function from sampled design or grid points. The GRNN prediction will be compared with known values. The efficiency of GRNN implementation will be noted and the strategy used to improve this efficiency will be described. Next to be described is the determination of derivatives of GRNN analytically and via the Complex Step Method (CSM). CSM will be shown to be fast and superior to the analytic approach of kernel derivatives. This will be followed by the description of implementation of the new boosting algorithm that significantly improves the accuracy of GRNN and allows the use of a single bandwidth that is chosen conservatively to be sub-optimal. Finally different combinations of results will be presented.

A two dimensional function was chosen as the test function and also to serve as a simulator of typical 2-dimensional CFD problem. The test function chosen was

$$f(x, y) = y\cos x + 3x^2 e^y \tag{40}$$

The partial first order derivatives of the function are

$$\begin{aligned} f_x(x, y) &= -y\sin x + 6xe^y \\ f_y(x, y) &= \cos x + 3x^2 e^y \end{aligned} \tag{41}$$

The design points chosen were $x \in [-7,7]$ and $y \in [-7,7]$.

The implementation of GRNN on the 2-dimensional test function was implemented using the equation:

$$f(x,y) = \frac{\sum_{j=1}^{m}\sum_{i=1}^{n} f_{i,j} e^{\frac{-(x-X_i)^2}{2h^2}} e^{\frac{-(y-Y_i)^2}{2h^2}}}{\sum_{j=1}^{m}\sum_{i=1}^{n} e^{\frac{-(x-X_i)^2}{2h^2}} e^{\frac{-(y-Y_i)^2}{2h^2}}} \tag{42}$$

Note that the recommended product kernel was used because the function is two dimensional. As is evident, our chosen kernel is the normal Gaussian. With regards to the efficiency of implementation, suppose that we need to evaluate the GRNN at N distinct data points given n test data. A direct naïve application of (42) would result in O(Nn) operations for the determination of the estimator at N grid points. Now since the Gaussian kernel used has compact support roughly at [-4h,4h], significant computer time can be saved by performing the local averaging within the region of support of the kernel. With this speed up the number of operations could then be O(Nnh). In all of the results to be presented below, this speedup of GRNN was implemented.

There are two current approaches of estimating the partial derivatives of GRNN equation (42). The first method is the finite difference approach, which suffers from subtractive cancellation errors and is therefore not considered in this report, Anderson, et. al.,(2000). The second method is the kernel derivatives approach. Here (42) is differentiated directly. To obtain the partial derivatives with respect to x, let

$$Ax = \sum_{j=1}^{m}\sum_{i=1}^{n} -\frac{f_{i,j}(x-X_i) e^{-\frac{(x-X_i)^2}{2h^2}} e^{-\frac{(y-Y_j)^2}{2h^2}}}{h^2}$$

$$Bx = \sum_{j=1}^{m}\sum_{i=1}^{n} e^{-\frac{(x-X_i)^2}{2h^2}} e^{-\frac{(y-Y_j)^2}{2h^2}}$$

$$Cx = \sum_{j=1}^{m}\sum_{i=1}^{n} f_{i,j} e^{-\frac{(x-X_i)^2}{2h^2}} e^{-\frac{(y-Y_j)^2}{2h^2}}$$

$$Dx = \sum_{j=1}^{m}\sum_{i=1}^{n} -\frac{(x-X_i) e^{-\frac{(x-X_i)^2}{2h^2}} e^{-\frac{(y-Y_j)^2}{2h^2}}}{h^2}$$

Then

$$f_x(x,y) = \frac{Ax}{Bx} - \frac{CxDx}{(Bx)^2} \tag{43}$$

To obtain the partial derivatives with respect to y, let

$$Ay = \sum_{j=1}^{m}\sum_{i=1}^{n} -\frac{f_{i,j}e^{-\frac{(x-X_i)^2}{2h^2}}(y-Y_j)e^{-\frac{(y-Y_j)^2}{2h^2}}}{h^2}$$

$$By = \sum_{j=1}^{m}\sum_{i=1}^{n} e^{-\frac{(x-X_i)^2}{2h^2}}e^{-\frac{(y-Y_j)^2}{2h^2}}$$

$$Cy = \sum_{j=1}^{m}\sum_{i=1}^{n} f_{i,j}e^{-\frac{(x-X_i)^2}{2h^2}}e^{-\frac{(y-Y_j)^2}{2h^2}}$$

$$Dy = \sum_{j=1}^{m}\sum_{i=1}^{n} -\frac{e^{-\frac{(x-X_i)^2}{2h^2}}(y-Y_j)e^{-\frac{(y-Y_j)^2}{2h^2}}}{h^2}$$

Then

$$f_y(x,y) = \frac{Ay}{By} - \frac{CyDy}{(By)^2} \tag{44}$$

It is clear from the above that the kernel derivatives method of computing partial derivatives is complicated and involves up to five summands for each of the partial derivatives. In fact for higher order derivatives the analytic formulae becomes quite unwieldy because of the ratio form of GRNN. MSE of the derivatives kernel estimators, indicate that for estimating the r^{th} derivatives, $m^r(x)$, is difficult with the MSE of order O(-4/n(2r+5)) resulting in a slower rate of convergence for higher values of r, Stone (1982). Below are two complete Fortran programs to compute derivatives of GRNN estimator using the analytic kernel differentiation approach and the new CSM.

Program 1. Fortan Program for GRNN and its 1^{st} order Derivatives Using Kernel Derivatives Method (equations (42-44) above):

```
C     **********MAIN PROGRAM ^************************************
      PROGRAM MULTIVARI
      IMPLICIT REAL (A-H,O-Z)
      PARAMETER (RHO=0.05,NX=500,NY=500,NT=20)
      DIMENSION X(NX),Y(NY),Y1(NX,NY),Y2(NT),YDX(NT),
     +Y3(NT),W(NT),W1(NT),Y4(NT),Y5(NT),YDY(NT),elapsed(2)
C
C COMPLEX VARIABLES
C

      OPEN(UNIT=77,FILE='aout.dat',FORM='FORMATTED',
     &       STATUS='unknown')
c
C     Generate Random Design Test variables between 0 and 7
c
      IDUM=6
      DO 10 I = 1,NT
      W(I) = RAN0(IDUM)*7
      W1(I) = RAN0(IDUM)*7
  10  CONTINUE

C
```

```
C GENERATE DESIGN POINTS
C
      X(1)=-10
      X(NX)=10
      Y(1)=-10
      Y(NY)=10
      DX=(X(NX)-X(1))/FLOAT(NX)
      DY=(Y(NY)-Y(1))/FLOAT(NY)
      DO 13 I=2,NX-1
      X(I)=X(I-1)+DX
 13   CONTINUE
      DO 14 I=2,NY-1
      Y(I)=Y(I-1)+DY
 14   CONTINUE
C
C       COMPUTE THE FUNCTIONAL VALUES AT DESIGN POINTS: THIS IS WHERE YOU CHANGE THE
FUNCTION
C    MAKE SURE CHOSEN FUNCTION MATCHES WITH (X,Y) DESIGN POINTS ABOVE
C
      DO 15 I = 1,NX
      DO 15 J=1,NY
      Y1(I,J)=Y(J)*COS(X(I))+3*(X(I)**2)*EXP(Y(J))
  15  CONTINUE
C
C      NOW ESTIMATE FUNCTION AND ITS DERIVATIVES VIA NATARAY-WATSON SMOOTHER
C

      CALL FUNCF(NX,NY,NT,X,Y,Y1,Y2,W,W1,RHO)
      CALL FUNCDX(NX,NY,NT,X,Y,Y1,YDX,W,W1,RHO)
      CALL FUNCDY(NX,NY,NT,X,Y,Y1,YDY,W,W1,RHO)

C
C           OUTPUT RESULTS
C

      WRITE(77,*)'X   Y   F   F-EST  F-ERR  FDX  FDX-ERR  FDY  FDY-ERR'

      DO 20 K = 1,NT
      Y3(K)=W1(K)*COS(W(K))+3*(W(K)**2)*EXP(W1(K))
      Y4(K)=-W1(K)*SIN(W(K))+6*W(K)*EXP(W1(K))
            Y5(K) = COS(W(K)) + 3*(W(K)**2)*EXP(W1(K))
      err = abs(Y2(K)- Y3(K))
      errx = abs(YDX(K)- Y4(K))
            erry = abs(YDY(K)- Y5(K))

      WRITE(77,1) W(K),W1(K),Y3(K),Y2(K),err,YDX(K),errx,YDY(K),erry
   20 CONTINUE


    1  FORMAT(1X,F10.3,1X,F10.3,1X,F10.3,1X,F10.3,1X,F10.3,1X,F10.3,1X,
     +F10.3,1X,F10.3,1X,7(F10.3,1X))


      total = ETIME(elapsed)
      print *, 'End: total=', total, ' user=', elapsed(1),
     &         ' system=', elapsed(2)

      STOP
      END

      SUBROUTINE FUNCF(NX,NY,NT,X,Y,Y1,Y2,W,W1,RHO)
      IMPLICIT REAL (A-H,O-Z)
```

```
      DIMENSION X(NX),Y(NY),Y1(NX,NY),Y2(NT),W(NT),W1(NT)
      DO 20 K = 1,NT
      SUM1=0.0
      SUM2=0.0
         DO 30 I= 1,NX
         D1 =(X(I)-W(K))**2
         DO 30 J=1,NY
         D2 =(Y(J)-W1(K))**2
         H =(EXP(-(D1+D2)/(2*(RHO)**2)))
         H1=H*Y1(I,J)
         SUM1=SUM1+H
         SUM2=SUM2+H1
30       CONTINUE
      Y2(K)= SUM2/SUM1
20    CONTINUE
      RETURN
      END

      SUBROUTINE FUNCDX(NX,NY,NT,X,Y,Y1,Y2,W,W1,RHO)
      IMPLICIT REAL (A-H,O-Z)
      DIMENSION X(NX),Y(NY),Y1(NX,NY),Y2(NT),W(NT),W1(NT)
      DO 20 K = 1,NT
      SUMN1=0.0
      SUMN2=0.0
      SUMN3=0.0
      SUMD1=0.0
      SUMD2=0.0
       DO 30 I= 1,NX
       D1 =(X(I)-W(K))**2
       DO 30 J=1,NY
       D2 =(Y(J)-W1(K))**2
       HN1=Y1(I,J)*(EXP(-(D1+D2)/(2*(RHO)**2)))*(-(W(K)-X(I))/(RHO)**2)
       HD1=(EXP(-(D1+D2)/(2*(RHO)**2)))
       HD2=(EXP(-(D1+D2)/(2*(RHO)**2)))**2
       HN2=Y1(I,J)*(EXP(-(D1+D2)/(2*(RHO)**2)))
       HN3=(EXP(-(D1+D2)/(2*(RHO)**2)))*(-(W(K)-X(I))/(RHO)**2)
       SUMN1=SUMN1+HN1
       SUMN2=SUMN2+HN2
       SUMN3=SUMN3+HN3
       SUMD1=SUMD1+HD1
       SUMD2=SUMD2+HD2
30    CONTINUE
      Y2(K)= (SUMN1/SUMD1)-((SUMN2*SUMN3)/SUMD2)
20    CONTINUE
      RETURN
      END

      SUBROUTINE FUNCDY(NX,NY,NT,X,Y,Y1,Y2,W,W1,RHO)
      IMPLICIT REAL (A-H,O-Z)
      DIMENSION X(NX),Y(NY),Y1(NX,NY),Y2(NT),W(NT),W1(NT)
      DO 20 K = 1,NT
      SUMN1=0.0
      SUMN2=0.0
      SUMN3=0.0
      SUMD1=0.0
      SUMD2=0.0
       DO 30 I= 1,NX
       D1 =(X(I)-W(K))**2
       DO 30 J=1,NY
       D2 =(Y(J)-W1(K))**2
       HN1=Y1(I,J)*(EXP(-(D1+D2)/(2*(RHO)**2)))*(-(W1(K)-Y(J))/(RHO)**2)
       HD1=(EXP(-(D1+D2)/(2*(RHO)**2)))
```

```
      HD2=(EXP(-(D1+D2)/(2*(RHO)**2)))**2
      HN2=Y1(I,J)*(EXP(-(D1+D2)/(2*(RHO)**2)))
      HN3=(EXP(-(D1+D2)/(2*(RHO)**2)))*(-(W1(K)-Y(J))/(RHO)**2)
      SUMN1=SUMN1+HN1
      SUMN2=SUMN2+HN2
      SUMN3=SUMN3+HN3
      SUMD1=SUMD1+HD1
      SUMD2=SUMD2+HD2
 30   CONTINUE
      Y2(K)= (SUMN1/SUMD1)-((SUMN2*SUMN3)/SUMD2)
 20   CONTINUE
      RETURN
      END

C     **********RAMDOM NUMBER GENERATOR*********************
      FUNCTION RAN0(IDUM)
      IMPLICIT REAL (A-H,O-Z)
      PARAMETER(IA=16807,IM=2147483647,AM=1./IM,
     +        IQ=127773,IR=2836,MASK=123459876)

      IDUM=IEOR(IDUM,MASK)
      K=IDUM/IQ
      IDUM=IA*(IDUM-K*IQ)-IR*K
      IF(IDUM.LT.0)IDUM=IDUM+IM
      RAN0=(AM*IDUM)
      IDUM=IEOR(IDUM,MASK)
      RETURN
      END
```

Program 2. Fortan Program for GRNN and its 1st order Derivatives Using Complex Step Method (CSM):

```
C     ***************MAIN PROGRAM ^*************************************
      PROGRAM COMPMULTI
      IMPLICIT REAL (A-H,O-Z)
      PARAMETER (RHO=0.05,NX=500,NY=500,NT=20,H=0.01)
      DIMENSION X(NX),Y(NY),Y1(NX,NY),Y2(NT),
     +Y3(NT),W(NT),W1(NT),Y4(NT),Y5(NT),CYDX(NT),CYDY(NT),elapsed(2)
C
C COMPLEX VARIABLES
C
            COMPLEX*8 CX(NX),CY(NY),CY1(NX,NY),CY2(NT),CW(NT),CW1(NT)
     +              ,CRHO,TEMP,HX


      OPEN(UNIT=77,FILE='aout.dat',FORM='FORMATTED',
     &      STATUS='unknown')
c
C     Generate Random Design Test variables between 0 and 7
c
      IDUM=6
      DO 10 I = 1,NT
      W(I) = RAN0(IDUM)*7
      W1(I) = RAN0(IDUM)*7
  10  CONTINUE

C
C GENERATE DESIGN POINTS
C
      X(1)=-10
      X(NX)=10
```

```
      Y(1)=-10
      Y(NY)=10
      DX=(X(NX)-X(1))/FLOAT(NX-1)
      DY=(Y(NY)-Y(1))/FLOAT(NY-1)
      DO 13 I=2,NX-1
      X(I)=X(I-1)+DX
 13   CONTINUE
      DO 14 I=2,NY-1
      Y(I)=Y(I-1)+DY
 14   CONTINUE
C
C       COMPUTE THE FUNCTIONAL VALUES AT DESIGN POINTS: THIS IS WHERE YOU CHANGE THE
FUNCTION
C    MAKE SURE CHOSEN FUNCTION MATCHES WITH (X,Y) DESIGN POINTS ABOVE
C
      DO 15 I = 1,NX
      DO 15 J=1,NY
      Y1(I,J)=Y(J)*COS(X(I))+3*(X(I)**2)*EXP(Y(J))
  15  CONTINUE

C
C           NEXT IS TO COMPUTE COMPLEX DERIVATIVES CDX AND CDY CONVERT SUBROUTINE FUNCF TO
COMPLEX CFUNCF
C
            HX= CMPLX(H)
      DO 25 I = 1,NX
      DO 25 J=1,NY
      CY1(I,J)=CMPLX(Y1(I,J))
  25  CONTINUE
      DO 26 I=1,NX
      CX(I)=CMPLX(X(I))
  26  CONTINUE
      DO 27 I=1,NY
      CY(I)=CMPLX(Y(I))
  27  CONTINUE
      DO 28 I=1,NT
      CW(I)=CMPLX(W(I))
      CW1(I)=CMPLX(W1(I))
  28  CONTINUE
      CRHO = CMPLX(RHO)
C     DX DERIVATIVES
            CALL CFUNCDX(NX,NY,NT,CX,CY,CY1,CY2,CW,CW1,CRHO,HX)
            DO 29 J=1,NT
            CYDX(J)=AIMAG(CY2(J))/H
            Y2(J) = REAL(CY2(J))
  29  CONTINUE
C     DY DERIVATIVES
            CALL CFUNCDY(NX,NY,NT,CX,CY,CY1,CY2,CW,CW1,CRHO,HX)
            DO 30 J=1,NT
            CYDY(J)=AIMAG(CY2(J))/H
  30  CONTINUE

C
C           OUTPUT RESULTS
C

      WRITE(77,*)'X     Y     F   F-EST F-ERR CDX   CERRX   CDY   CERRY'

      DO 20 K = 1,NT
      Y3(K)=W1(K)*COS(W(K))+3*(W(K)**2)*EXP(W1(K))
      Y4(K)=-W1(K)*SIN(W(K))+6*W(K)*EXP(W1(K))
            Y5(K) = COS(W(K)) + 3*(W(K)**2)*EXP(W1(K))
```

```
      err = abs(Y2(K)- Y3(K))
      cerrx = abs(CYDX(K)- Y4(K))
            cerry = abs(CYDY(K)- Y5(K))

      WRITE(77,1) W(K),W1(K),Y3(K),Y2(K),err,CYDX(K),cerrx,CYDY(K),cerry
   20 CONTINUE

    1  FORMAT(1X,F10.3,1X,F10.3,1X,F10.3,1X,F10.3,1X,F10.3,1X,F10.3,1X,
     +F10.3,1X,F10.3,1X,7(F10.3,1X))

      total = ETIME(elapsed)
      print *, 'End: total=', total, ' user=', elapsed(1),
     &         ' system=', elapsed(2)

      STOP
      END

      SUBROUTINE CFUNCDX(NX,NY,NT,X,Y,Y1,Y2,W,W1,RHO,HH)
      IMPLICIT COMPLEX*8 (A-H,O-Z), INTEGER(I-N)
      DIMENSION X(NX),Y(NY),Y1(NX,NY),Y2(NT),W(NT),W1(NT)
      DO 20 K = 1,NT
            TEMP=W(K)
            W(K)=CMPLX(REAL(W(K)),REAL(HH))
      SUM1=0.0
      SUM2=0.0
         DO 30 I= 1,NX
         D1 =(X(I)-W(K))**2
         DO 30 J=1,NY
         D2 =(Y(J)-W1(K))**2
         H =(CEXP(-(D1+D2)/(2*(RHO)**2)))
         H1=H*Y1(I,J)
         SUM1=SUM1+H
         SUM2=SUM2+H1
   30    CONTINUE
      Y2(K)= SUM2/SUM1
            W(K)=TEMP
   20  CONTINUE
      RETURN
      END

      SUBROUTINE CFUNCDY(NX,NY,NT,X,Y,Y1,Y2,W,W1,RHO,HH)
      IMPLICIT COMPLEX*8 (A-H,O-Z), INTEGER(I-N)
      DIMENSION X(NX),Y(NY),Y1(NX,NY),Y2(NT),W(NT),W1(NT)
      DO 20 K = 1,NT
            TEMP=W1(K)
            W1(K)=CMPLX(REAL(W1(K)),REAL(HH))
      SUM1=0.0
      SUM2=0.0
         DO 30 I= 1,NX
         D1 =(X(I)-W(K))**2
         DO 30 J=1,NY
         D2 =(Y(J)-W1(K))**2
         H =(CEXP(-(D1+D2)/(2*(RHO)**2)))
         H1=H*Y1(I,J)
         SUM1=SUM1+H
         SUM2=SUM2+H1
   30    CONTINUE
      Y2(K)= SUM2/SUM1
            W1(K)=TEMP
   20  CONTINUE
      RETURN
      END
```

The above two programs to compute 20 function values and their partial derivatives using the kernel derivatives method and the complex step method were timed using the Fortran timing function "etime". The results for the CSM method was 9.073sec and the kernel derivatives method took 36.893sec. The speedup using the CSM method is therefore superior by about a factor of 4. The grid mesh was 500x500 and a bandwidth of 0.05 was used for both methods.

Next to report is the results of applying the novel L_2Boost algorithm described above to GRNN. First following the recommendations above a crude GRNN was chosen. Application of the L_2Boost algorithm turned the crude GRNN into an accurate estimator in only a few B-steps (5 steps). Direct implementation of GRNN using the above test function required an h value of 0.05 and a 500 x 500 grid size in order to reduce the relative error of function approximation to less than 1%. Note that in kernel function approximation, the higher the dimension the more data points are required to achieve a desired accuracy. This is due to the so called "curse of dimensionality" inherent in many kernel based estimators. Now with L_2Boosting, we were able to obtain similar accuracy using only 50 x 50 grid points and a larger h value of 0.5. Only 5 boosting steps were required. This shows that L_2Boost is a simple approach to raise the consistency (or order) of approximation of an estimator. L_2Boosting may also address the curse of dimensionality for higher dimensional data as shown by the reduced number of grid points required for a desired accuracy. With respect to a choice of a bandwidth, h, boosting makes its choice robust as the results no longer depend on it. This is as long as the initial choice which is kept constant is chosen so as to make the variance of the MSE of the estimator low. For the GRNN case, this means that h should be chosen to be larger than the optimal value obtained via analysis of MSE. This L_2Boosted GRNN maybe comparable in efficiency but superior in simplicity to the modified GRNN reported in; Krysl and Belytschko (2000). The modified GRNN reported therein is in fact closely related to the popular Local Polynomial Kernel Estimators (LPKE), using a local linear model, Wand and Jones (1995). However, in that paper there is no mention of how the bandwidth was determined since its determination can be costly computationally.

Figure 1 shows the exact functional values of the test function, (40). The exponential nature of the function is evident in this figure. Functional values range from less than 1 to over 150,000. In order to compare the performance of GRNN of the test function on various estimates, we define the relative error as follows

$$\text{Re}\,lative\;\; Error\;\; = \frac{True - Estimate}{True} \tag{45}$$

With this definition of relative error, Figure 2 compares the relative error of GRNN estimates of the test function using the L2Boost algorithm and the unboosted GRNN. The figure shows estimates for 25 equally spaced (x,y) points. It is clear from the figure that the boosting algorithm significantly improves the accuracy of the function estimates. The largest errors of the L_2Boost algorithm are at the boundaries. This, as expected is due to the boundary effects, which makes the bias of the GRNN to be largest at the boundary. It can be shown that without accounting for the boundary effects, the bias for our $\hat{m}(x)$ estimate is O(h) instead of O(h^2) at boundary points. This means that for $x \in \left[[0,h) \cup (a-h,a]\right]$ where 0 is the left boundary and a the right boundary,

the performance of GRNN is not consistent. This inconsistency can easily be corrected, however, this has not been done in this report.

There are two basic approaches for accounting for boundary effects; the first approach is to use specially constructed kernels, that are constructed so that the bias is a order $O(h^2)$ for all $x > 0$. Zhang and Karunamuni, (2000). The other approach for correcting for boundary effects, is to use the new approach of R-Functions, Rvachev and Sheiko (1995). This is an elegant approach that will be adopted in this work.

Figures 3 and 4 shows the comparison of actual partial derivatives of the test function with the respect to x and y respectively. The errors of the partial derivatives with respect to x are larger especially at the boundaries. We believe that these errors can be significantly reduced once the kernel boundary effects are taken into account.

Figures 5 to 10 shows coutour plots of percentage relative errors for boosted and unboosted estimates of the test function and its first partial derivatives. These figures are meant to highlight the significant accuracy obtained by the new L_2Boost algorithm via the combination of GRNN and CSM. Figure 5 is the coutour plot of unboosted function estimate. Figure 6 is for the L_2Boosted function estimate. The improvement in the estimate is quite significant. This was done with a fixed bandwidth, h, without the need of the usual time-consuming optimization algorithms to choose the best bandwidth parameter. Finally Figures 7-10 shows the significant reduction in relative errors for both the partial derivatives estimates of the test function $\partial f/\partial x$ and $\partial f/\partial y$.

Conclusions and Recommendations

A detailed mathematical theory of the Generalized Regression Neural Network (GRNN) function approximator has been presented. GRNN has a fast training time than traditional Neural Networks (NN). It has been shown that GRNN can be combined with L_2Boost algorithm to significantly improve its accuracy in a few B-boosting steps. It has also been demonstrated that GRNN can be combined with the new Complex Step Method (CSM) to obtain accurate 1^{st} order derivatives. As far as the objectives of this research is concerned the following recommendations can be made

1. The L_2Boost algorithm will be used to improve the Hierarchical Training of NPCA-NN (HT of NPCA-NN) algorithm developed for chemical mechanism data. Note that HT of NPCA-NN was the first attempt to apply a form of L_2Boost to NN.
2. Due to the extremely high dimensions of chemical mechanism data, GRNN cannot be used to replace NPCA-NN despite its fast training characteristic.
3. GRNN is only good for data of dimensions 3 or less, such as CFD. Our hope for studying GRNN was to take advantage of its fast training attributes and design it as NPCA-NN. However due to the “curse of dimensionality” it will be inefficient if implemented as NPCA-NN. It best use, however is in mesh free CFD which will be an interesting application with a high pay off.
4. GRNN combined with CSM and L_2Boost will be a novel and efficient approach for mesh free CFD. This is the approach we will follow to develop our in-house 2-dimensional CFD solver to test the NPCA-NN model that has been successfully demonstrated. The

previously reported 2-dimensional Euler solver will be extended to include reactive flow equations implemented via the mesh free GRNN. Note that current CFD solvers experience convergence difficulties when solving reactive flows. This is usually attributed to two main causes; the stiffness of the partial differential equations of reactive flows, and the quality of the mesh. If the solver can be made mesh free, then the second cause can be eliminated and better convergence of the solvers obtained. When coupling NPCA-NN to CFD, therefore, it will be advantageous to couple it to a mesh free solver.

5. Recent publication of paper by Buhlmann and Yu, (2003) has given credibility and basic theory to the HT of NPCA-NN algorithm developed in our earlier work.

References

Anderson, W.K., Newman III, J.C., Whitfield, D.L., and Nielsen, E.J., 2000, "Sensitivity Analysis for the Navier-Stokes equations on Unstructured Meshes Using Complex Variables," AIAA Journal, Vol. 39, No. 1, pp 56-63

Buhlmann, P., and Yu, B., 2003, "Boosting with the L_2 Loss: Regression and Classification," Journal of the American Statistical Association, Vol. 98, pp 324-339

Butuk, N.K., 2002, Mathematically Reduced Chemical Reaction Mechanism of DME Using Neural Networks, Final Report US Department of Energy Grant Number: DE-FG26-00NT-40830

Di Marzio, M., and Taylor, C.C.,2004, "Boosting Kernel Density Estimates: a bias reduction technique," Biometrika , Vol.91, pp 226-233

Friedman, J., 2001, "Greedy Function Approximation: a gradient boosting machine," The Annals of Statistics, Vol. 29, pp 1189-1232

Hardle, W.,1990, Applied Nonparametric Regression. Cambridge University Press, Cambridge

Hastie, T., Tibshirani, R., and Friedman, J., 2001, The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer, New York.

Higgins, J.J, and Keller-McNulty, S., 1995, Concepts in Probability and Stochastic Modeling. Duxbury Press, New York.

Krysl, P., and Belytschko, T., 2000, "An Efficient Linear-Precision Partition of Unity Basis for Unstructured Meshless Methods," Communications in Numerical Methods in Engineering, Vol. 16, pp 239-255

Liu, G.R., 2003, Mesh Free Methods: Moving beyond the Finite Element Method. CRC Press, New York.

Mustafi,C.K.,1978, “On the Asymptotic Distribution of the Estimates of the Derivatives of a Distribution Function,” SIAM Journal on Applied Mathematics, Vol. 34, pp 73-77

Nadaraya, E.A.,1964, “On Estimating Regression,” Theory of Probably and its Applications. 10, pp 186-90.

Onate, E., Idelsohn, S., Zienkiewic, O. C., Taylor, R.L., and Sacco, C., 1996, “A stabilized Finite Point Method for Analysis of Fluid Mechanics Problems,” Computer Methods In Applied Mechanics and Engineering, Vol. 139, pp 315-346

Parzen, E.,1962, “On Estimation of a Probability Density Function and Mode,” The Annals of Mathematical Statistics, Vol.33,No.3, pp 1065-76

Rvachev, V.L., and Sheiko, T.I., 1995, “R-Functions in Boundary Value Problems in Mechanics,” Applied Mechanics Review, Vol. 48, No. 4, pp 151-188

Schuster, E.F.,1969, “Estimation of a Probability Density Function and Its Derivatives,” Annals of Mathematical Statistics, Vol. 40, pp 1187-95

Scott, D.W.,1992, Multivariate Density Estimation: Theory, Practice, and Visualization. Wiley, New York.

Smith, D.M.,1998, “Multiple Precision Complex Arithmetic and Functions,” ACM Transactions on Mathematical Software, Vol.24, pp 359-367

Stone, C.J., 1982, “Optimal Global Rates of Convergence for Nonparametric Regression,” The Annals of Statistics, Vol. 10, No. 4, pp 1040-1053

Wand, M.P., and Jones, M.C., 1995, Kernel Smoothing. Chapman and Hall, New York.

Watson, G.S.,1964, “Smooth Regression Analysis,” Sankhya, Series A, 26, pp 359-372.

Zhang, S., and Karunamuni, R.J., 2000, “On Nonparametric Density Estimation at the Boundary,” Nonparametric Statistics, Vol. 12, pp 197-221

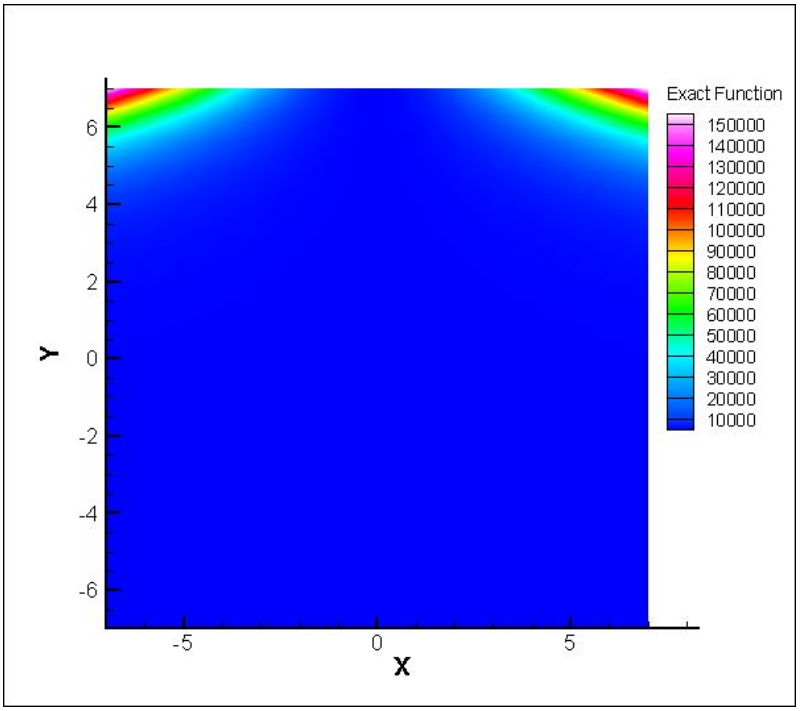

Figure 1. Exact Function Values of Equation (40).

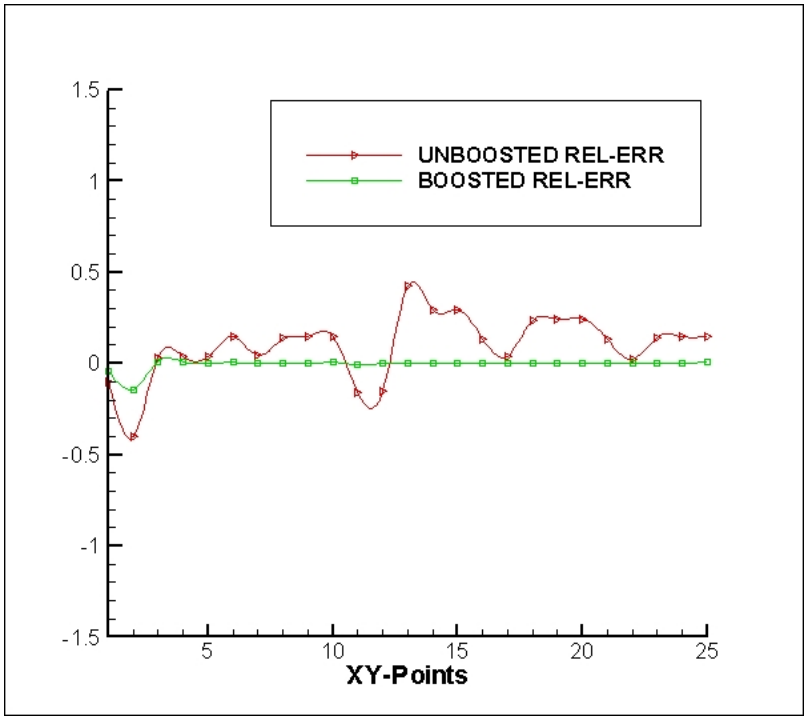

Figure 2: Comparison of L_2Boost and UnBoosted GRNN estimates of Test Function.

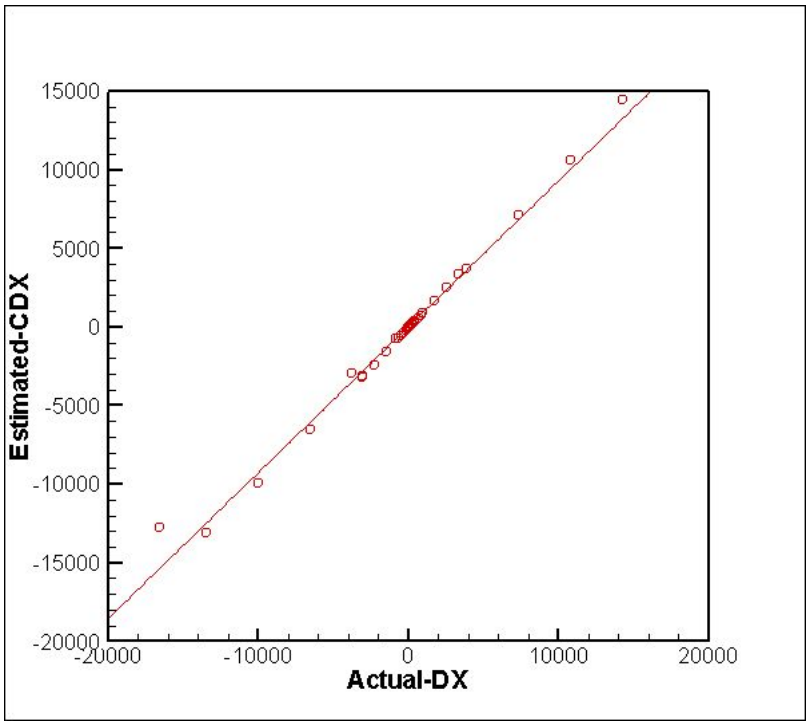

Figure 3: Comparison of Partial Derivatives, df/dx. Estimated vs. Actual

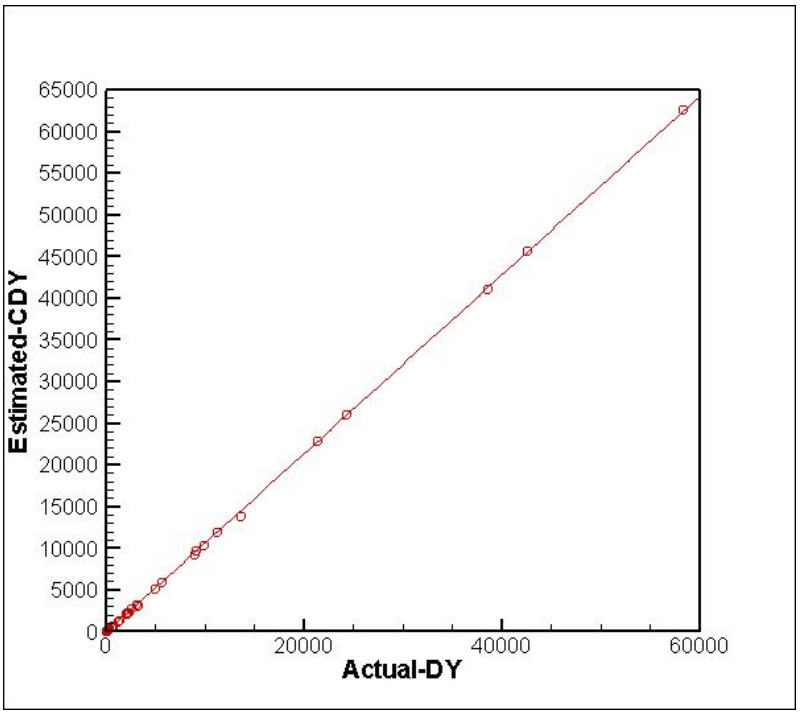

Figure 4: Comparison of Partials Derivatives, df/dy. Estimated vs. Actual.

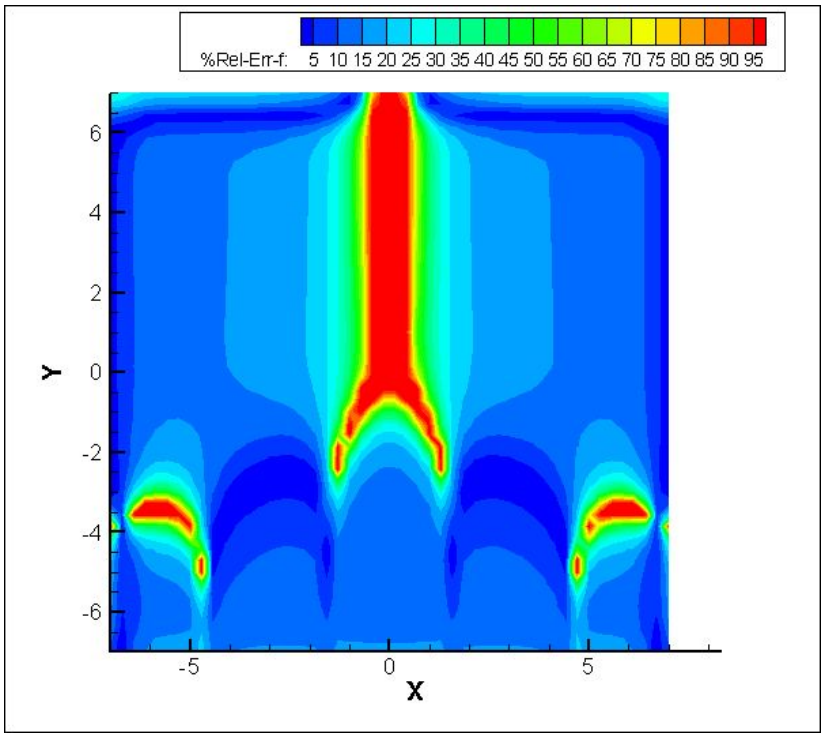

Figure 5: Percentage Relative Errors of UnBoosted Function Estimated.

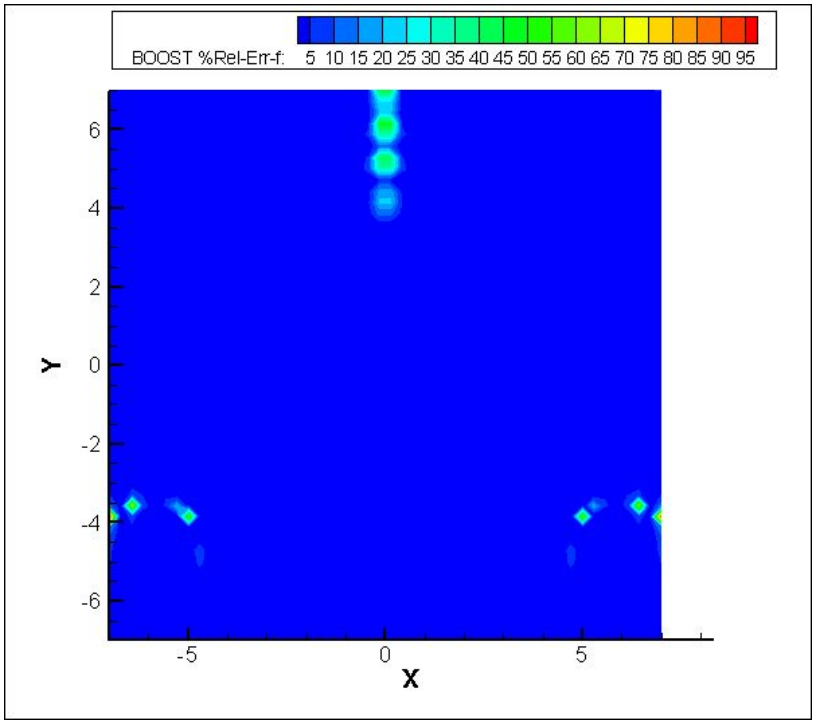

Figure 6: Percentage Relative Errors of L_2Boosted Test Function Estimate

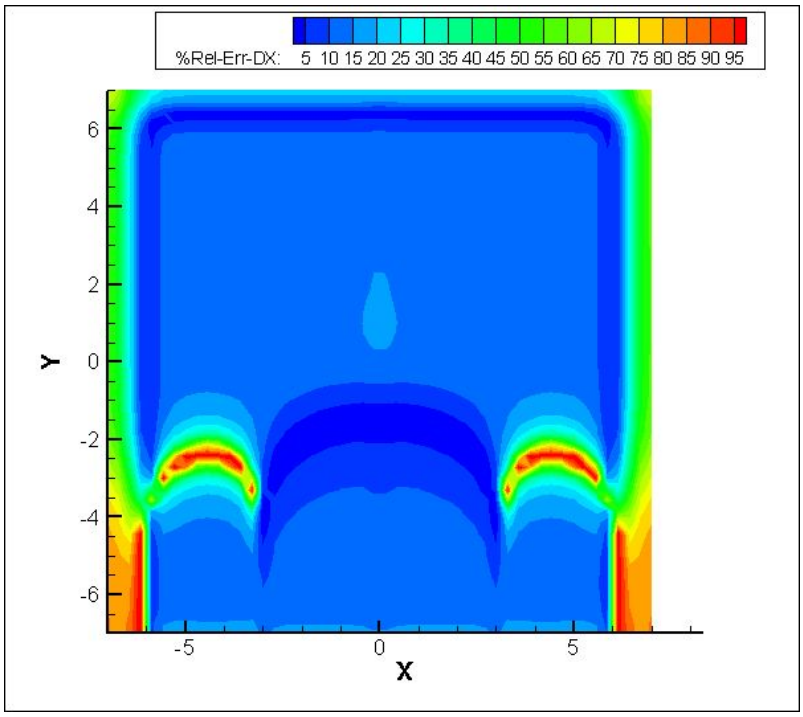

Figure 7: Percentage Relative Errors of UnBoosted Partial Derivatives, df/dx, Estimate

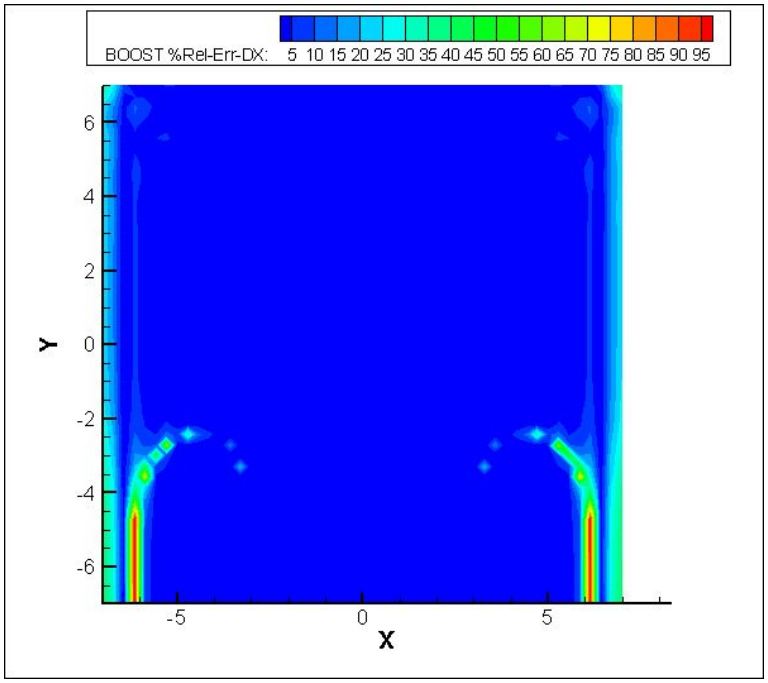

Figure 8: Percentage Relative Errors of L_2Boosted Partial Derivatives, df/dx, of Test Function.

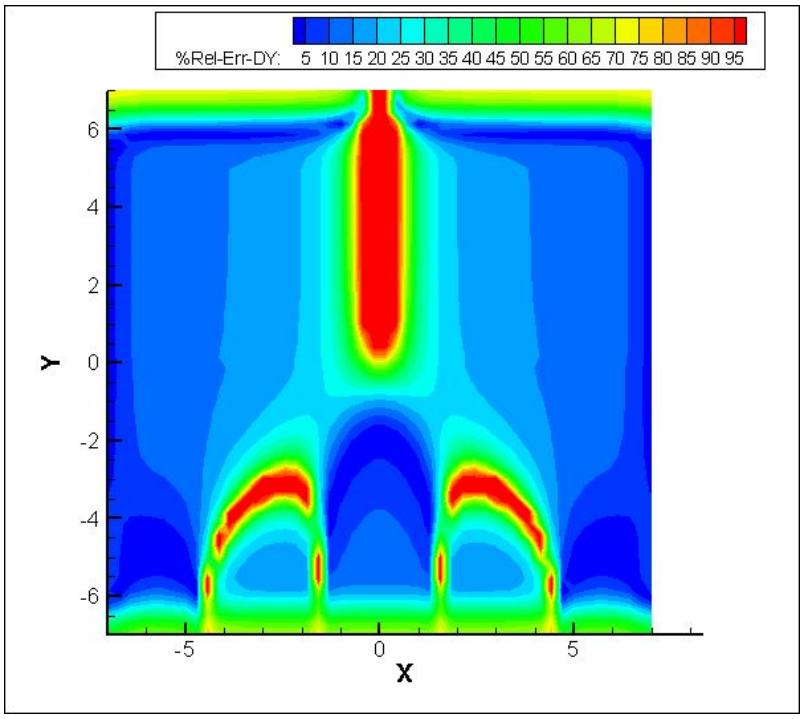

Figure 9: Percentage Relative Errors of UnBoosted Partial Derivatives, df/dy, Estimate

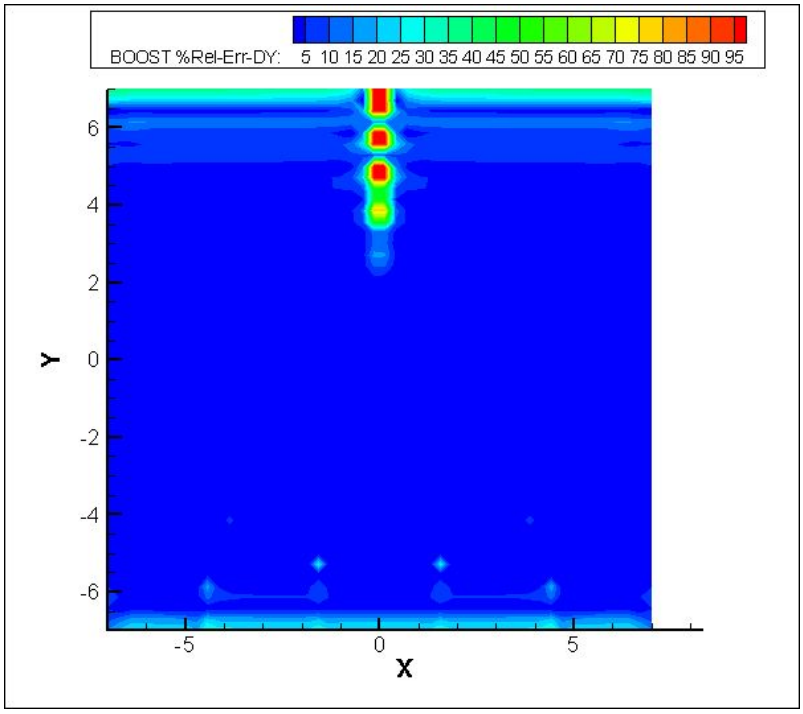

Figure 10: Percentage Relative Errors of L_2Boosted Partial Derivatives, df/dy, of Test Function.