Creation of a Class of N-Dimensional Interpolation Methods for use with Turbine and Compressor Maps

# Stephen Marone

# NASA Glenn Summer Intern

# June 25th, 2015

## Abstract

The project of designing a set of n-dimensional interpolation methods is mostly comprised of finding appropriate schemes and ensuring that the cost of calling such schemes is minimal. The code created for this process begins with integrating four basic schemes: linear nearest neighbors, weighted nearest neighbors, cosine neighbor interpolation, and Hermite Neighbor Interpolation. These schemes are written in python with the use of the numpy libraries to ensure ease of use and low costs. Each interpolation method, when performed, requires an input of training points with n dimensions and predicted points of n-1 dimensions, the nth dimension always being considered as the dependent dimension. The output gives a list of values for the predicted points along with gradient values at each point.

## Program Structure

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Percent Differences | | Size Independent Avg. Percent Diff. |  |
|  | Normal (V\*) | Large (LV\*) | Overall Avg. Percent Diff. |
| LN | -6.88% | -6.03% | -6.46% |
| WN | -4.38% | -4.64% | -4.51% | -5.41% |
| CN | -5.04% | -4.96% | -5.00% |
| HN | -6.02% | -5.36% | -5.69% |  |

Cost Improvements -Table

The program has gone through 5 versions and is currently set with a structure of 2 base classes and a child class for each interpolation method. Versions 1 and 2 were written roughly with iteration loops commonly used through them. Version 3 and 4 proceeded to remove a lot of these loops and sped up the interpolations by a noticeable amount. The cost improvements are listed per interpolation method and grid size in Cost Improvements -Table 1. These improvements will continue as the program develops further.

## Problem Creation

A small part of the program is its default problems, which the user can call to test the interpolation schemes easily. The choices are currently 3rd and 5th order 2 dimensional Legendre Polynomials, a piecewise problem consisting of three intersecting planes, an egg crate problem, and 2nd and 4th order 5 dimensional problems. Four of the problem sets are of three or less dimensions, so they can be visualized. A representation of each of the 3 dimensional problems can be seen in Piecewise Representation - Figure 1 and Crate Representation - Figure 2.



Piecewise Representation - Figure

These representations show that the problems each have areas where many interpolation techniques will have difficulty. The 2 dimensional problems are not shown here because of their ability to be included in a plot accompanying the results for direct comparison. 3 dimensional results later in the report can be compared with these figures.

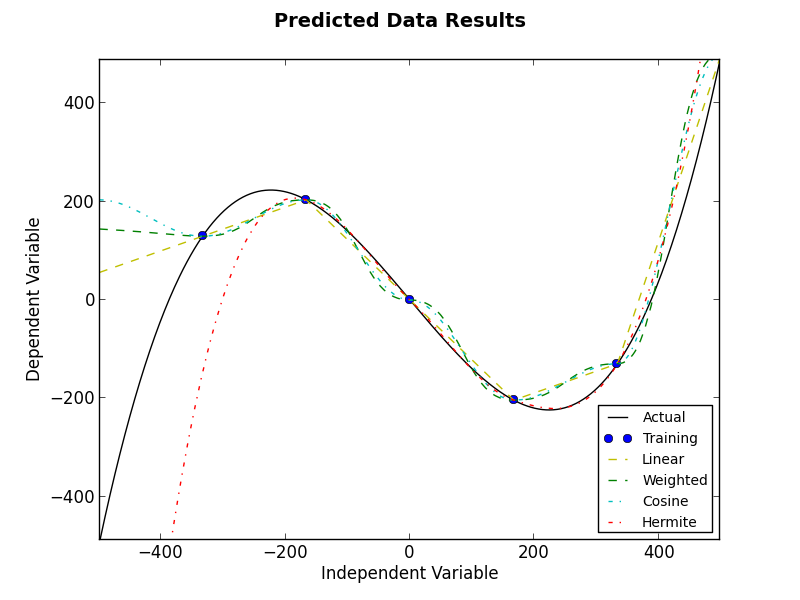


Crate Representation - Figure

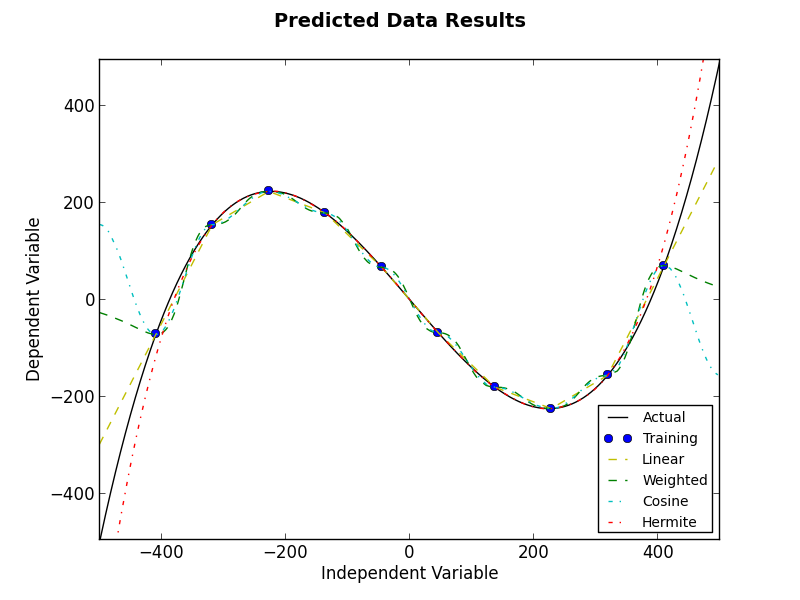
## 2 Dimensional Interpolation Results

Validation of this program is most easily visualized with a simple case in a low dimension. Since at least one independent and one dependent are necessary for any type of interpolation, the initial problem set consists of 2 dimensional problems. Both of these problems are Legendre Polynomials scaled by 500, one being of the 3rd order and the other of the 5th order. This scaling is to ensure accuracy relations with the other test problems used.

With the problem functions integrated into the program, each interpolation is performed to see how well they model the function as the number of training points increase. The plots that best modeled the changes can be seen in 3rd Order 2D with 5 Training Points - Figure 3 through 5th Order 2D with 100 Training Points - Figure 8



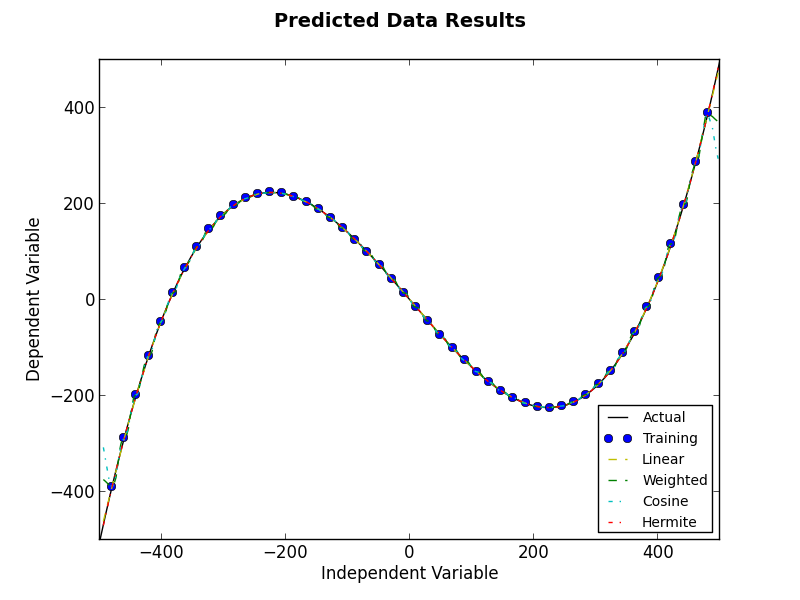
3rd Order 2D with 5 Training Points - Figure



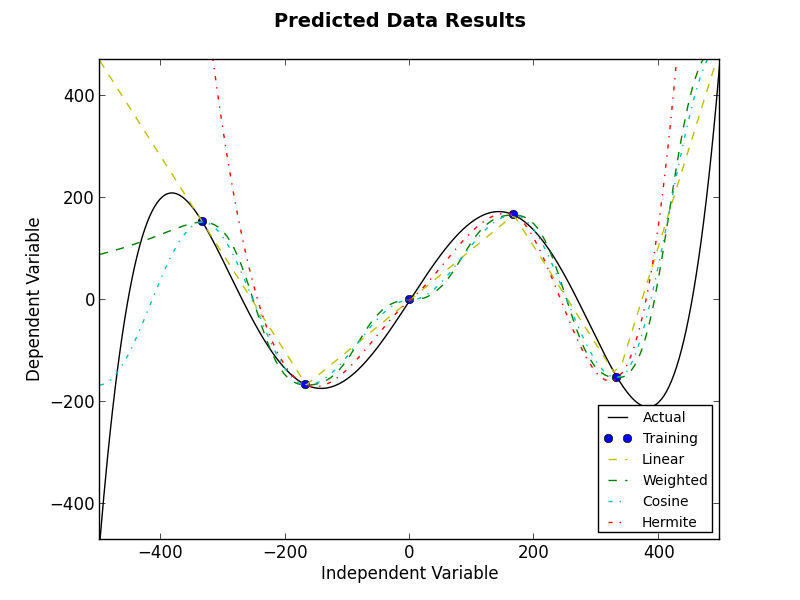
3rd Order 2D with 10 Training Points - Figure

The expected outcome can be observed; a direct correlation with the quantity training points and the interpolations fit to the function is apparent in all of the figures. In addition to this, the figures provide a nice visual understanding of how the interpolation types vary. It should be noted that both the cosine and the weighted interpolations tend to diverge toward the boundaries. Also note that it is not uncommon for the Hermite interpolation to miss an extreme training point completely.

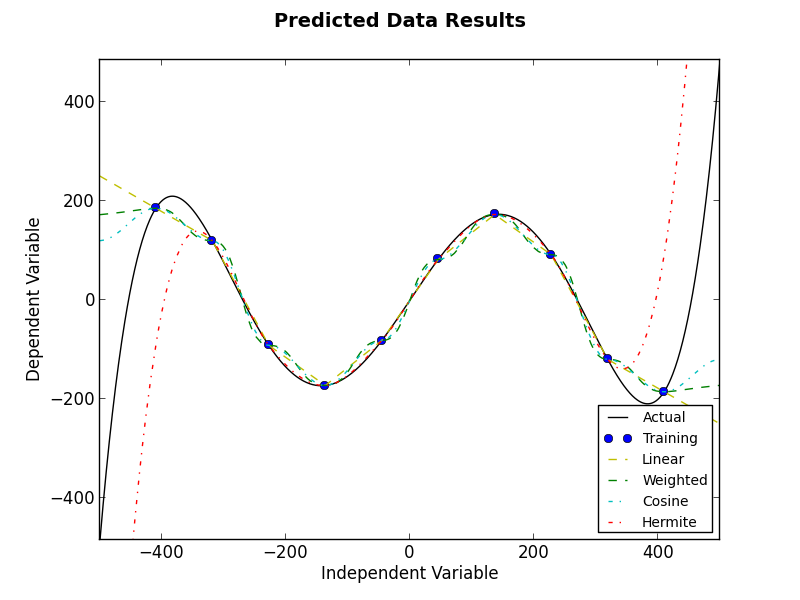
More results can be found at the end of this report. A table of the numeric results is provided there and further inferences can be performed with its data. An important observation from the error results is the increase of the effectiveness of the Hermite interpolation as more training points are added to the problem. Also note that it did not perform as well on low levels of the 5th order polynomial.



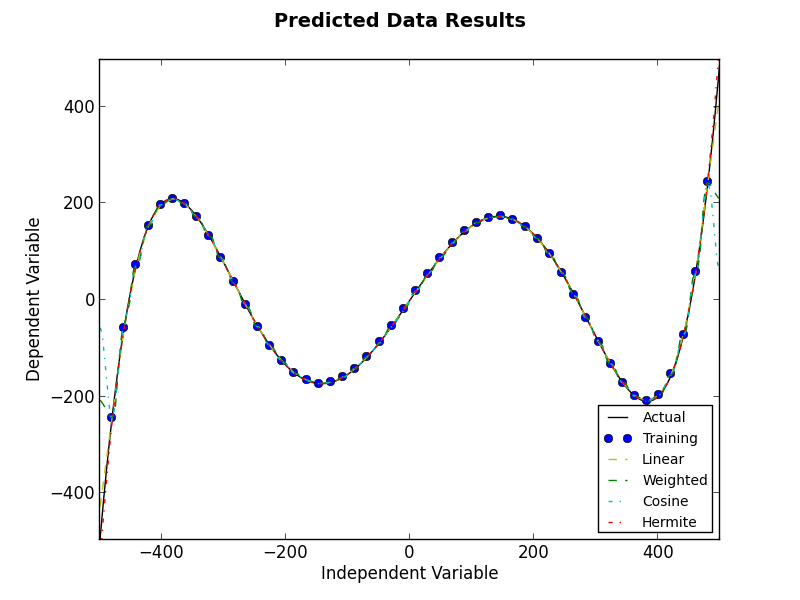
3rd Order 2D with 50 Training Points - Figure



5th Order 2D with 5 Training Points - Figure



5th Order 2D with 10 Training Points - Figure

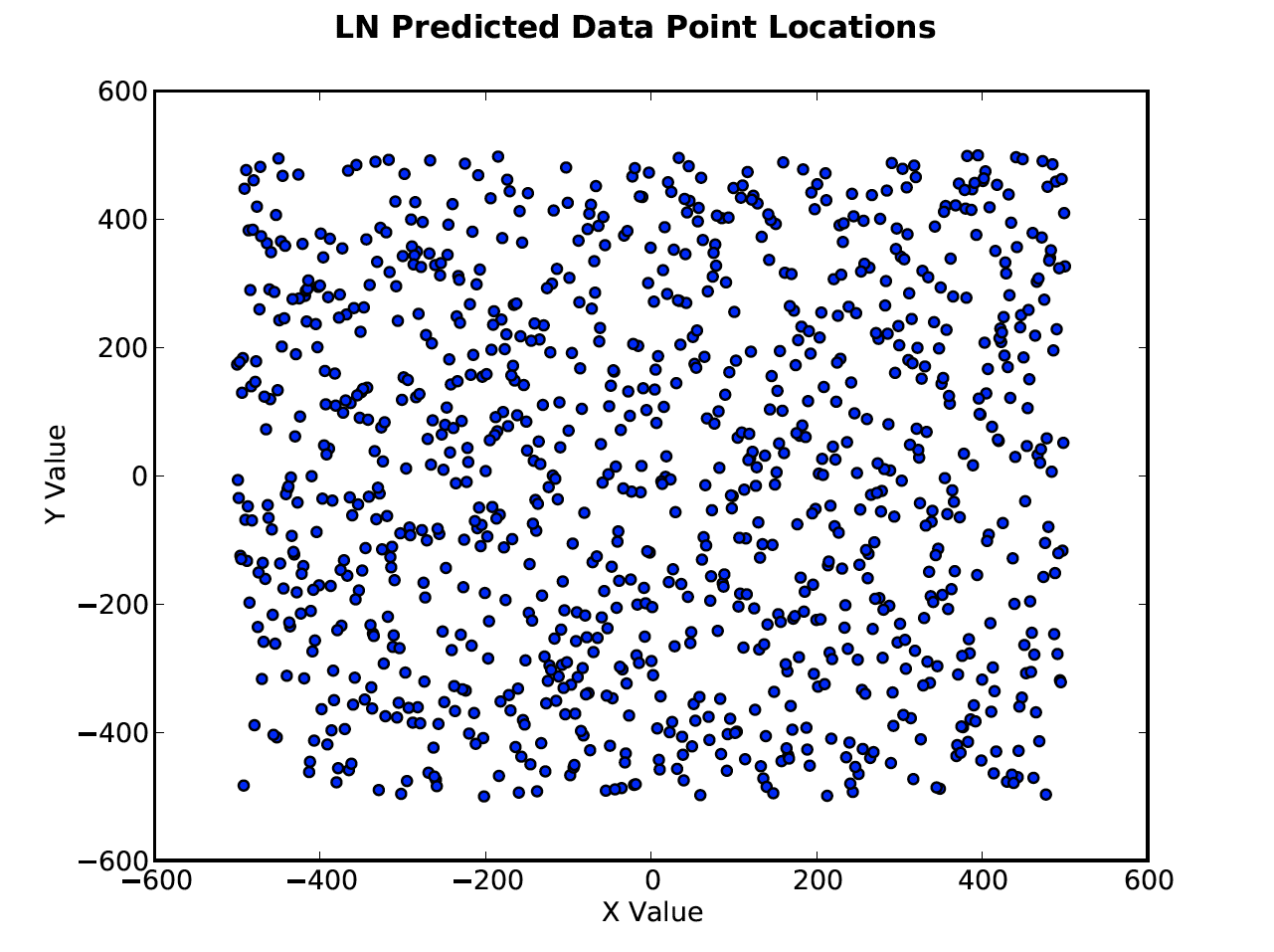


5th Order 2D with 50 Training Points - Figure

## 3 Dimensional Interpolation Results

With the values being predicted, error is found by simply solving the problem at each prediction point for an exact solution and recording a percent difference. The gradients are done similarly. Since their exact value cannot be found easily at each location, a good replacement was to use a high accuracy step in the complex step method.

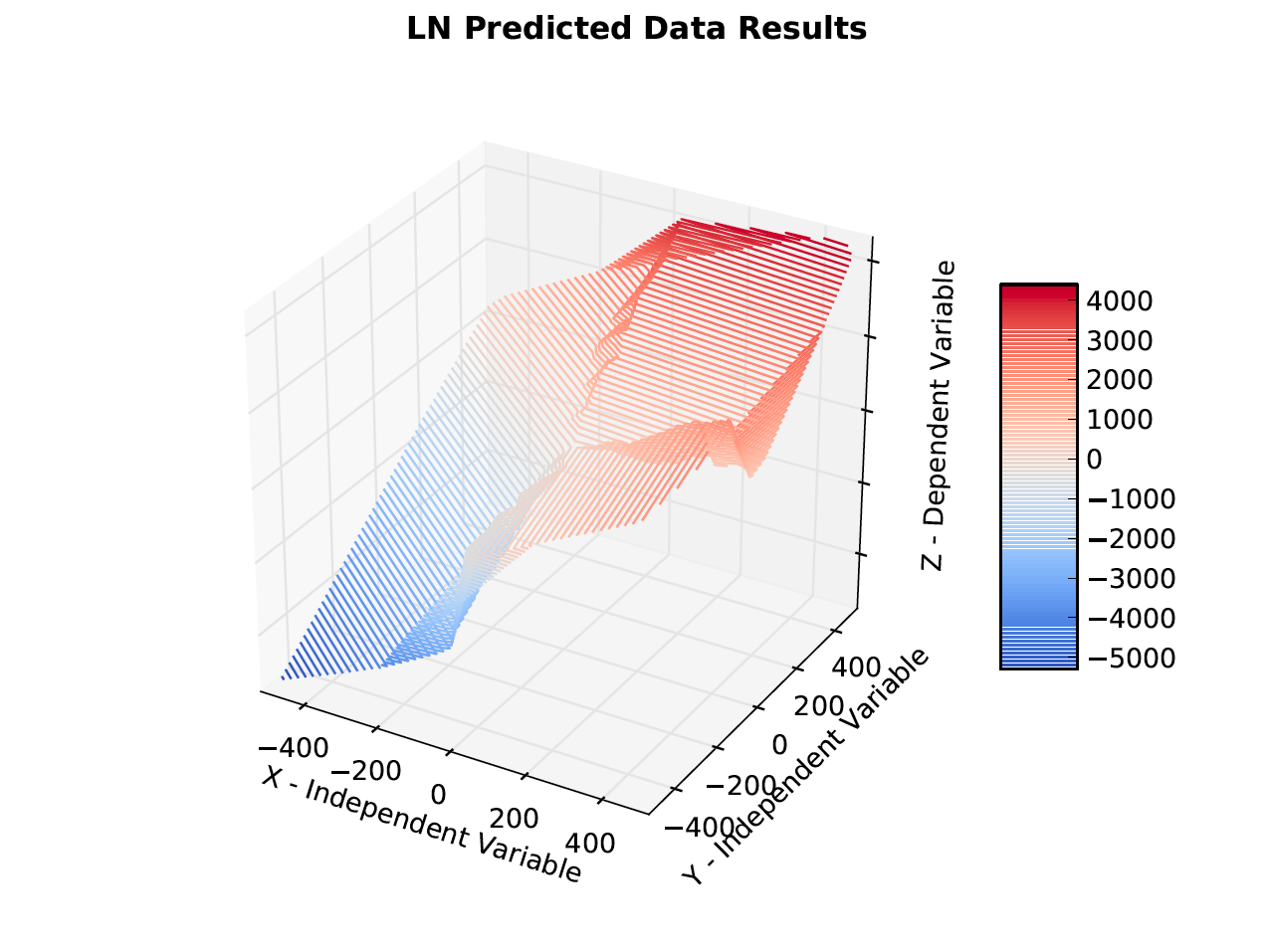
The program can optionally plot the predicted point distribution for 3 dimensional problems as well. The results in this paper for the 3 dimensional predicted points follow a Latin Hypercube distribution so that the point locations might comprise a high quality distribution. The point locations for one run can be seen in Prediction Point Distribution - Figure 9.



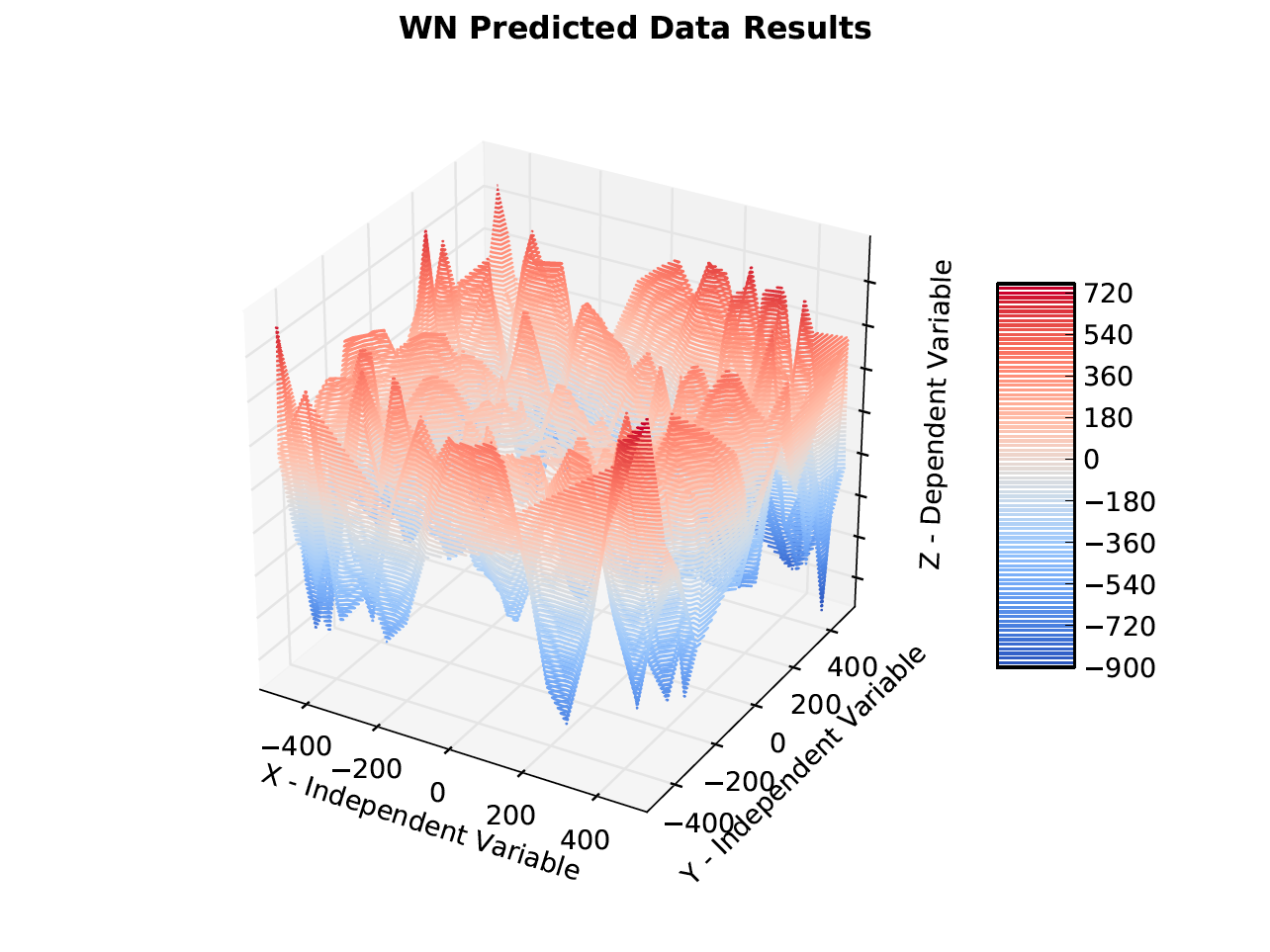
Prediction Point Distribution - Figure

This distribution, once loaded through each interpolation method, produced favorable results. For comparison, some example representations of each finished interpolation can be seen in Linear Nearest Neighbor Results - Figure 10 through Hermite Neighbor Interpolation Results- Figure 13.

Discontinuities can be noticed at extreme locations such as the intersection of the planes in the piecewise problem or the peaks and valleys of the crate problem. These apparently small differences from the training data plots at the beginning of the report can conglomerate to substantial error in each problem.

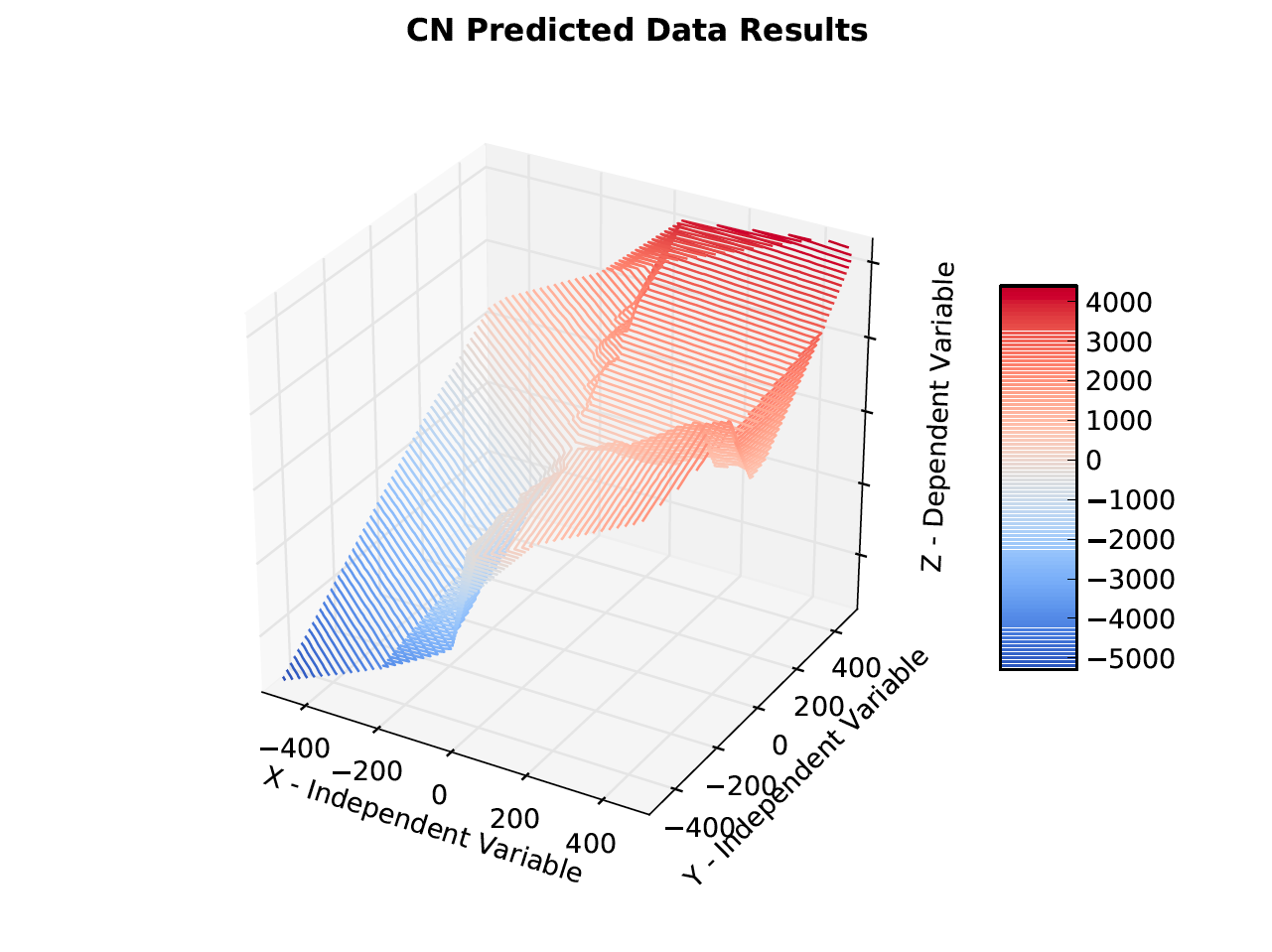


Linear Nearest Neighbor Results - Figure

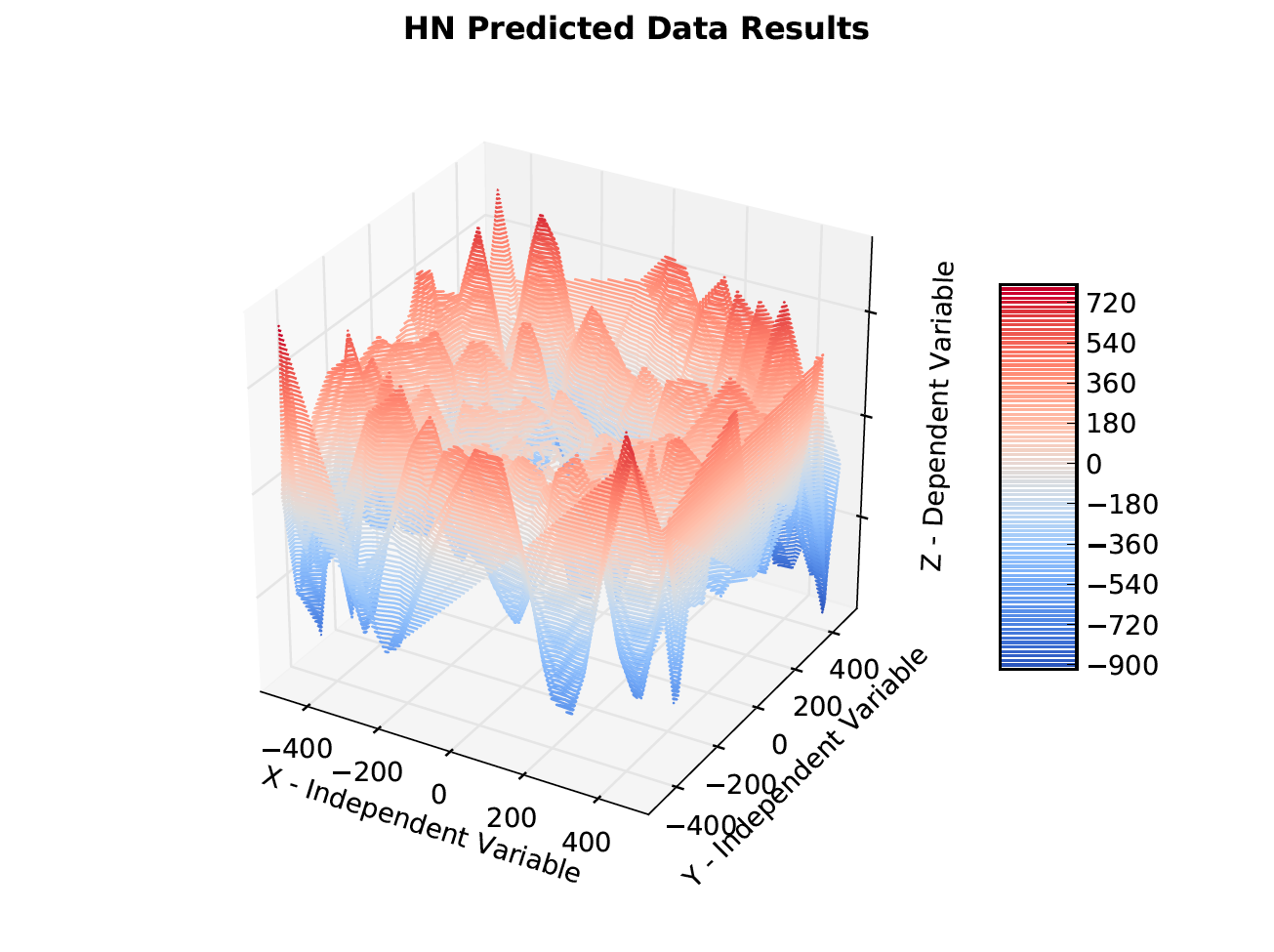


Weighted Nearest Neighbor Results - Figure

These representations poorly portray the differences in interpolation schemes since higher dimensional problems are too complex for visual inspection. One method to observe major differences between the schemes is to analyze their error distributions. By doing so, the linear nearest neighbor scheme appears to have a generally lower average with an occasional spike while the other three methods have a more uniform distribution. This can be seen in Linear Nearest Neighbor Error – Figure 14 and Hermite Neighbor Interpolation Error - Figure 15.

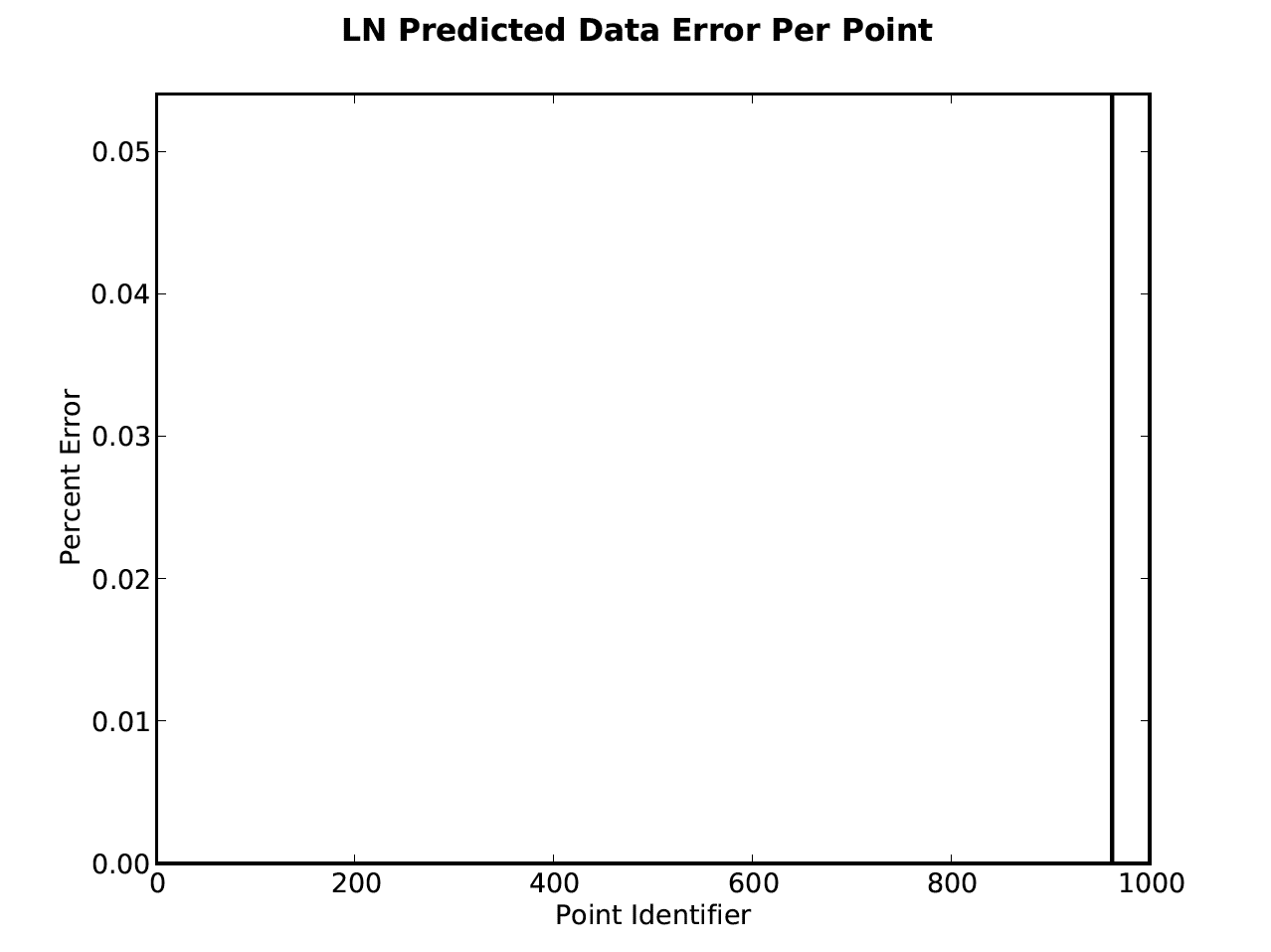


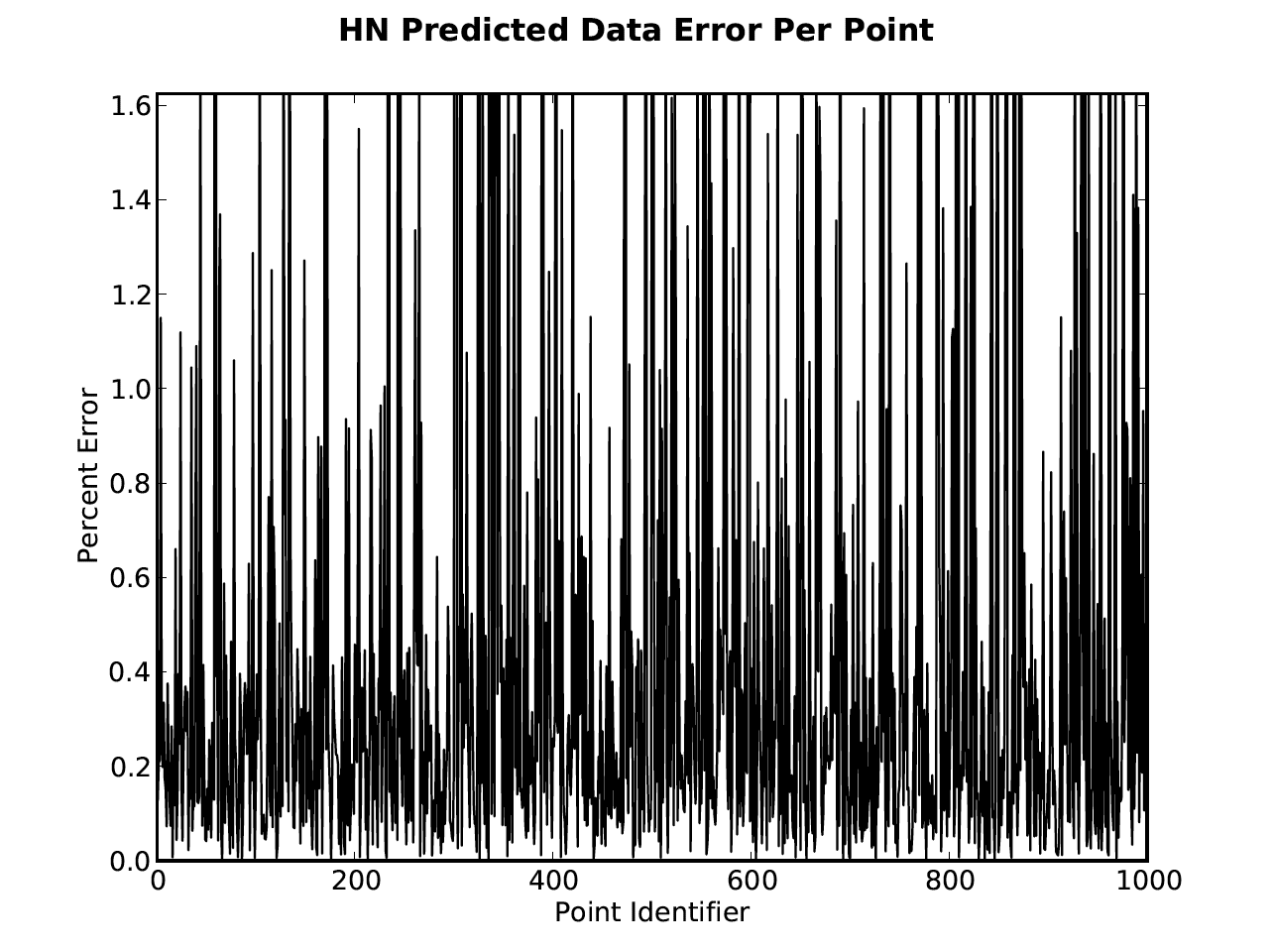
Cosine Neighbor Interpolation Results – Figure



Hermite Neighbor Interpolation Results- Figure

Although understanding of the problem is improved with error plots, the terminal output is still generally more useful for quality observations. Inputs For Average Error Table Runs - Table 2 and Error From Average of 3 Run Sets - Table 3 provide the inputs and error results for 3 averaged runs of each possible problem listed in the beginning of the report. Currently, the amount of error is not satisfactory for the gradients in regards to the actual amounts, but it should be noted that the gradients are accurate to the interpolation method. It is also important to observe that all errors are reduced if more training points are added into the problem.

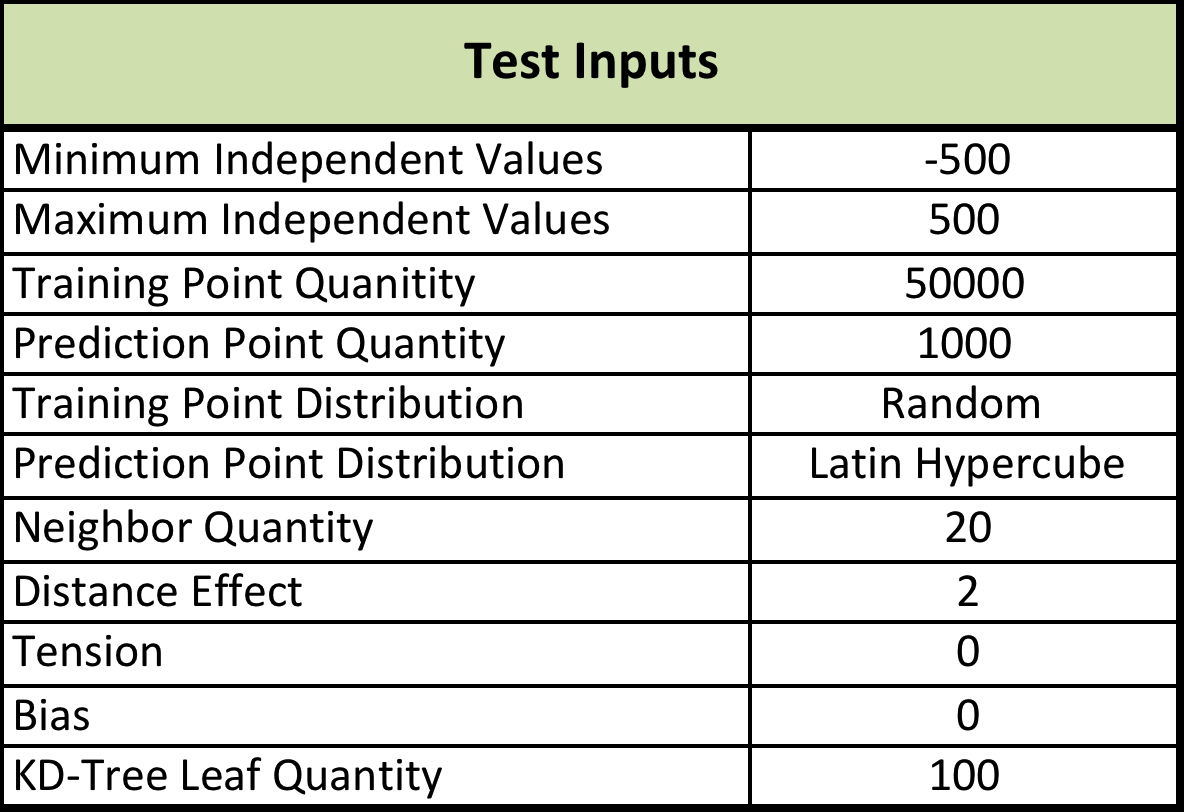


Linear Nearest Neighbor Error – Figure Hermite Neighbor Interpolation Error - Figure

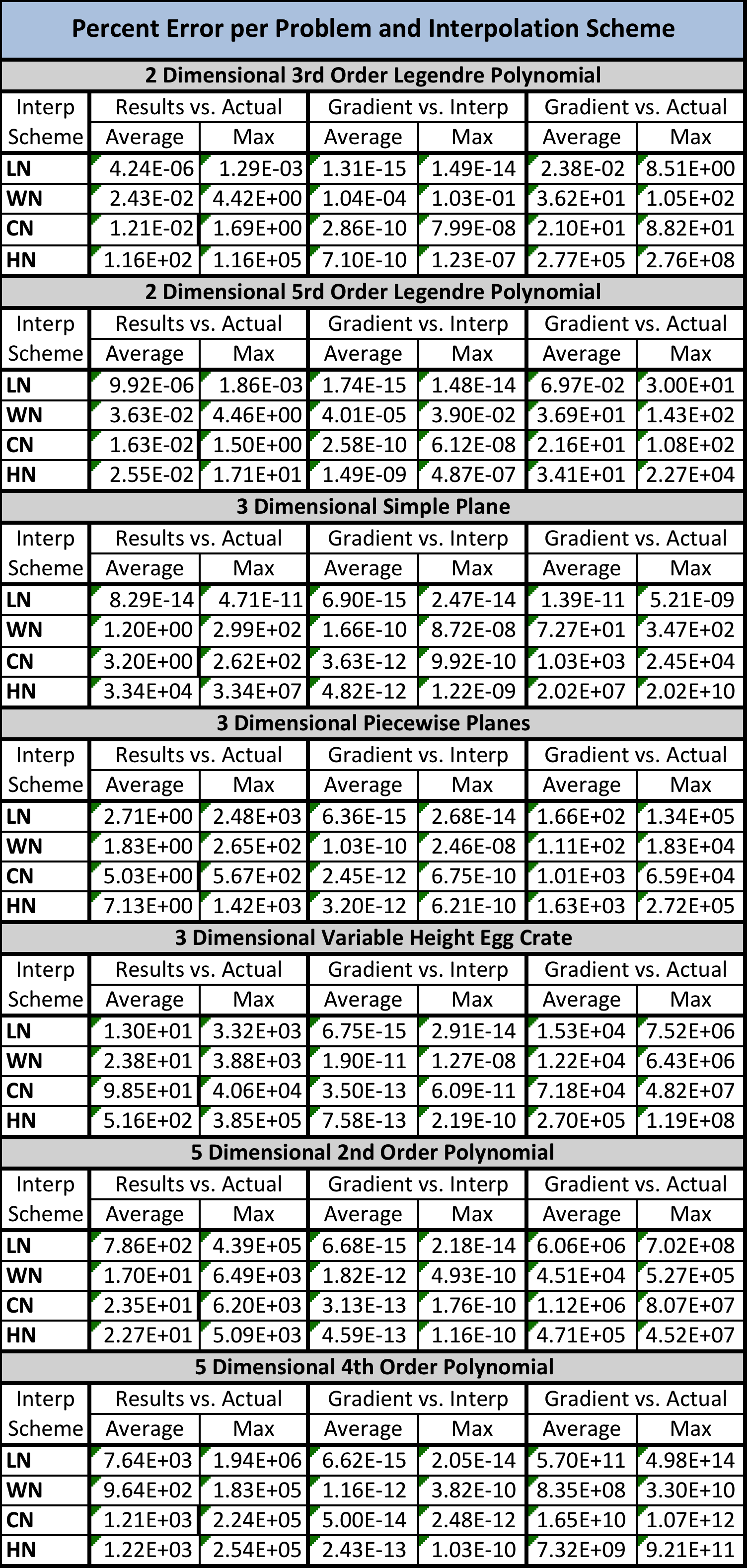
## Conclusions

The n-dimensional interpolation program created provides satisfactory results in nearly all aspects. It also does not require more than a couple of seconds per interpolation for the average problem setup of 50000 training points and 1000 predicted points along with other default values.

Despite this, there is a large possibility that many improvements can still be integrated into it. The future goals are to reduce costs continuously, improve the results found for the gradient solutions, and possibly even integrate more interpolation schemes.



Inputs For Average Error Table Runs - Table



Error From Average of 3 Run Sets - Table