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 a piecewise problem, which consists of three intersecting planes, and an egg crate problem. Both problem sets are of three dimensions so they can be visualized. A representation of each problem 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. Results later in the report can be compared with these figures.

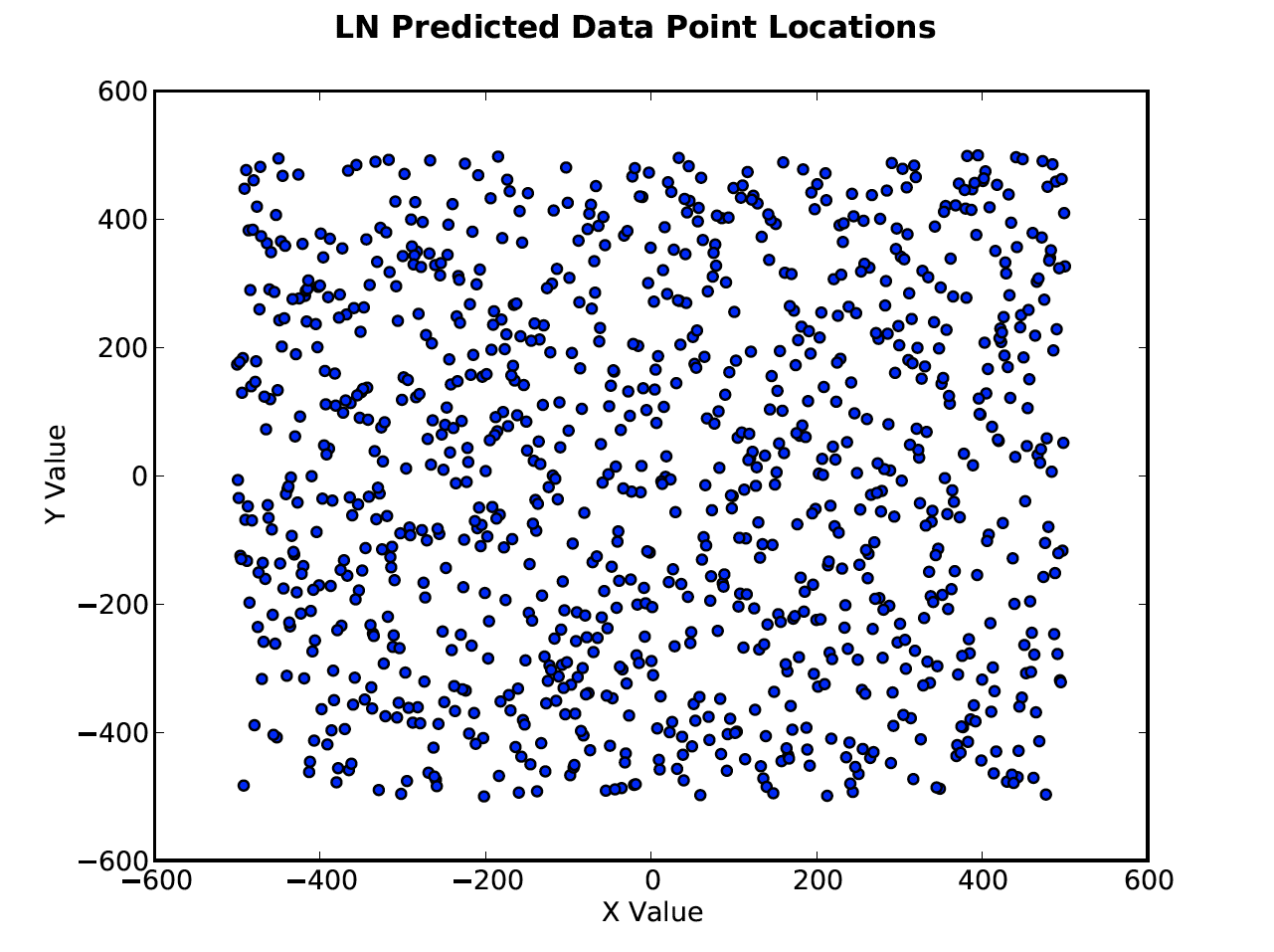


Crate Representation - Figure

## Interpolation Results

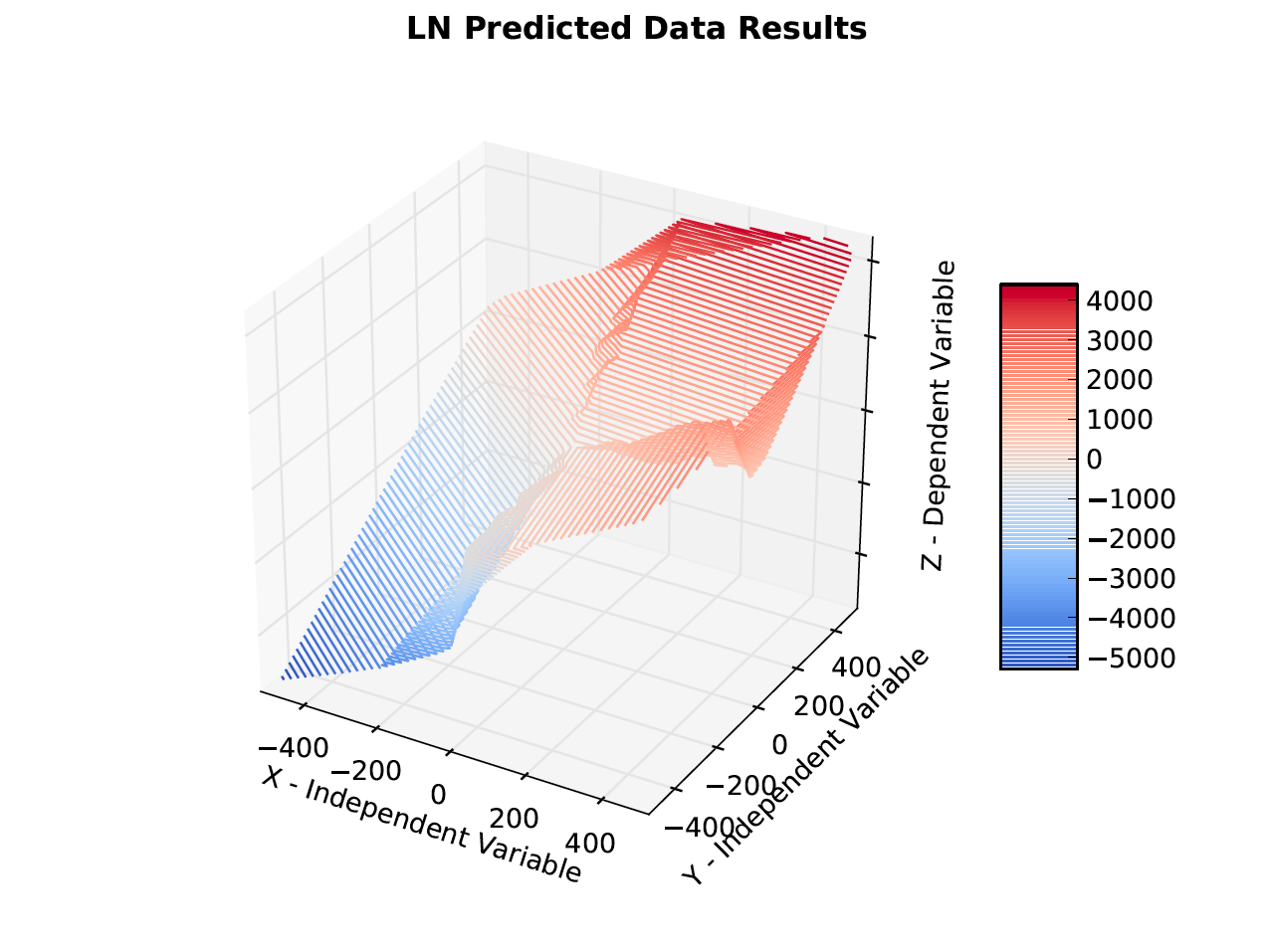
With the values being predicted, error is found by simply solving the problem at each prediction point for an exact solution and recording a percent difference. The gradients are done similarly. Since their exact value cannot be found easily at each location, a good replacement was to use a high accuracy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

The program can optionally plot the predicted point distribution. The results in this paper follow a Latin Hypercube distribution so that the point locations might comprise a good distribution. The point locations for one run can be seen in Prediction Point Distribution - Figure 3.

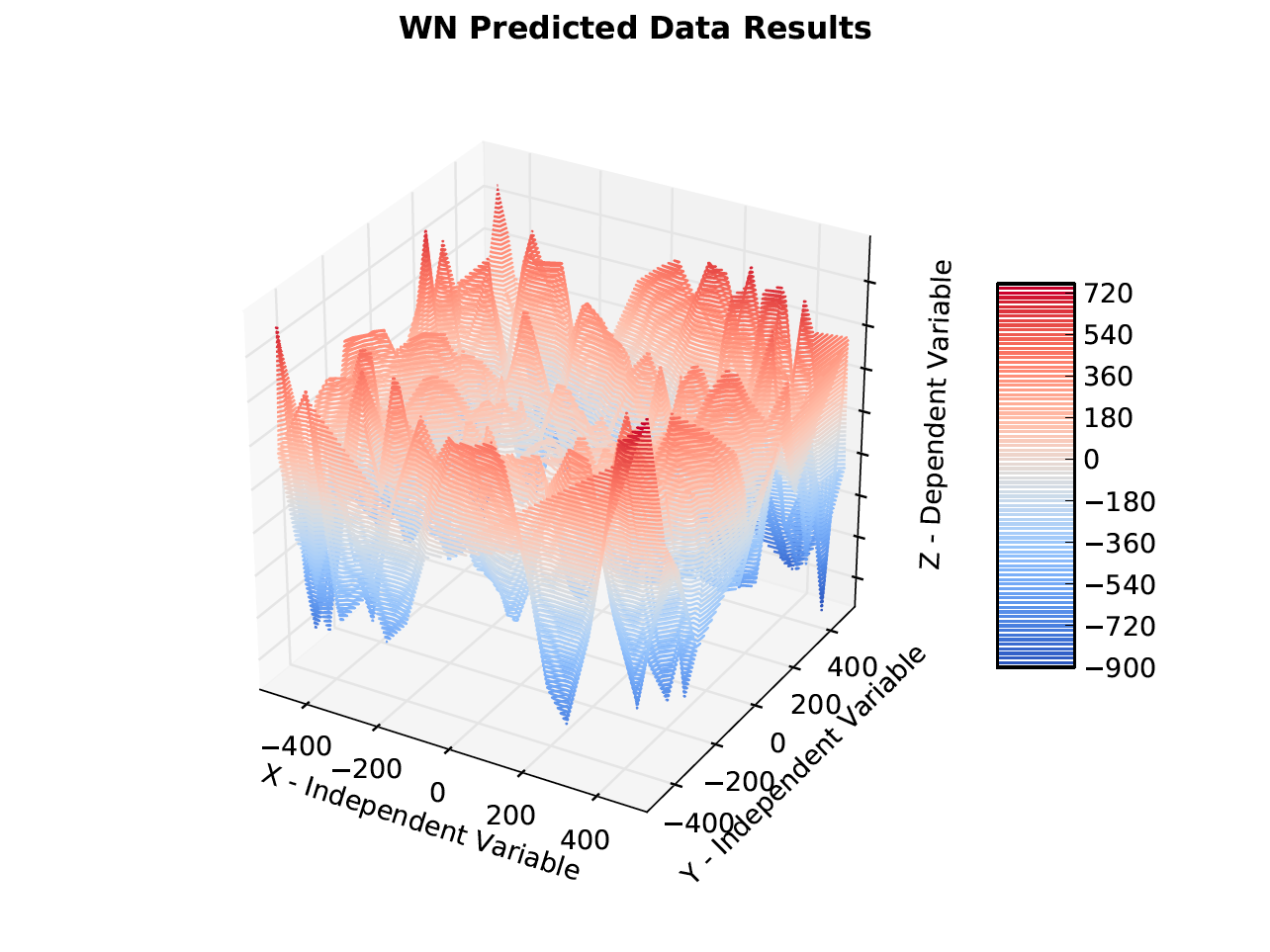


Prediction Point Distribution - Figure

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 4 through Hermite Neighbor Interpolation Results- Figure 7. Mistakes 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.

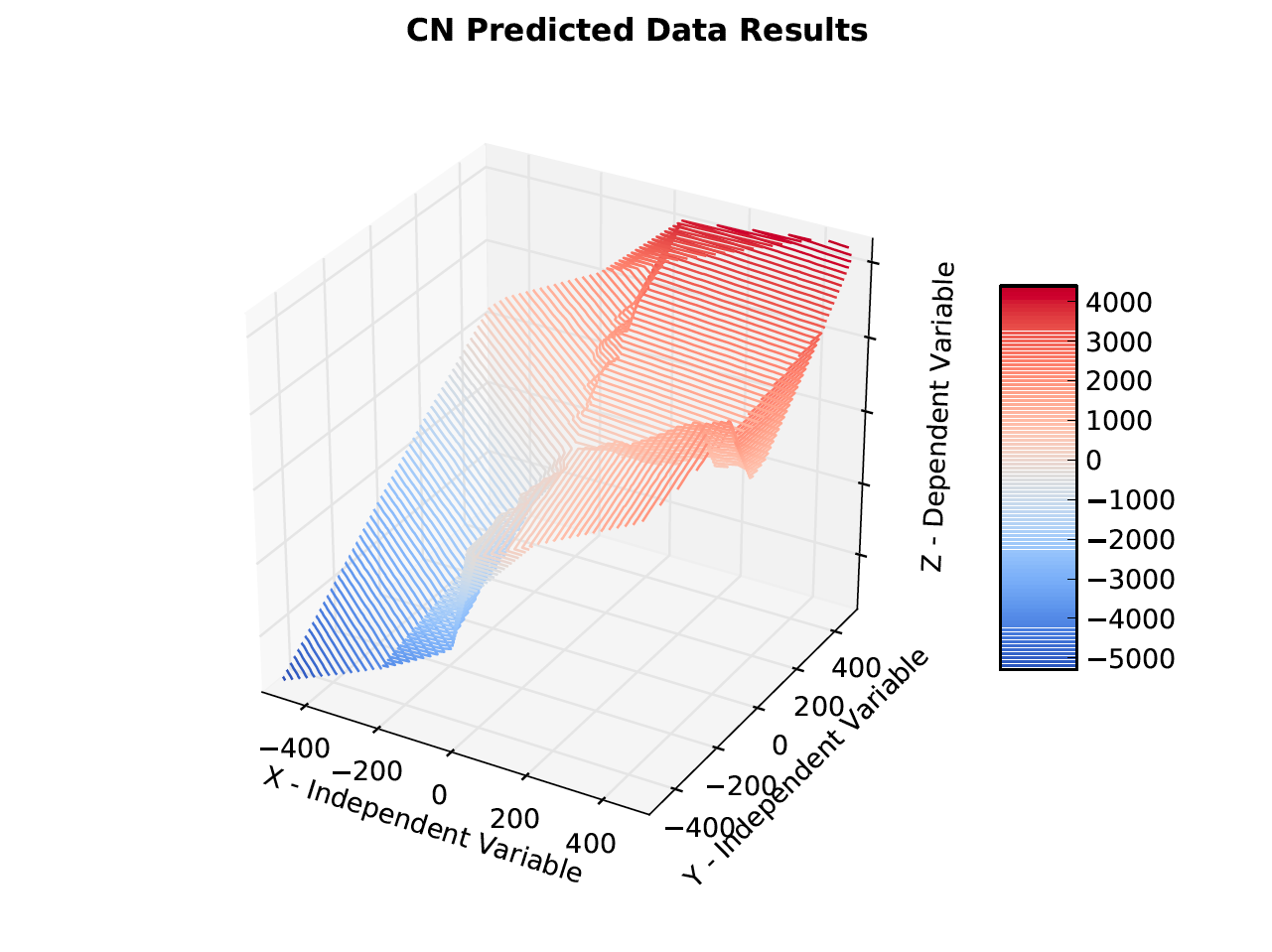


Linear Nearest Neighbor Results - Figure

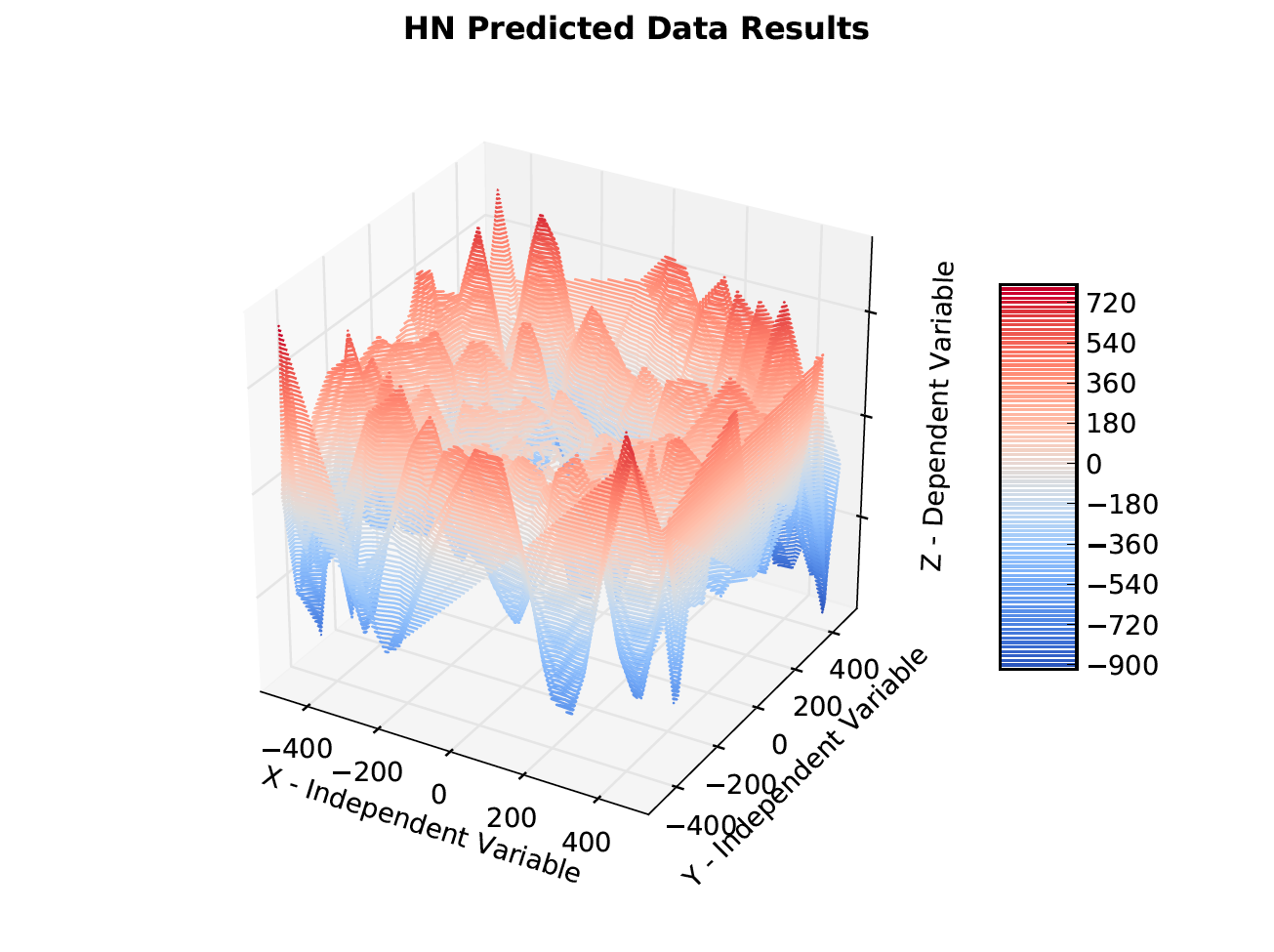


Weighted Nearest Neighbor Results - Figure

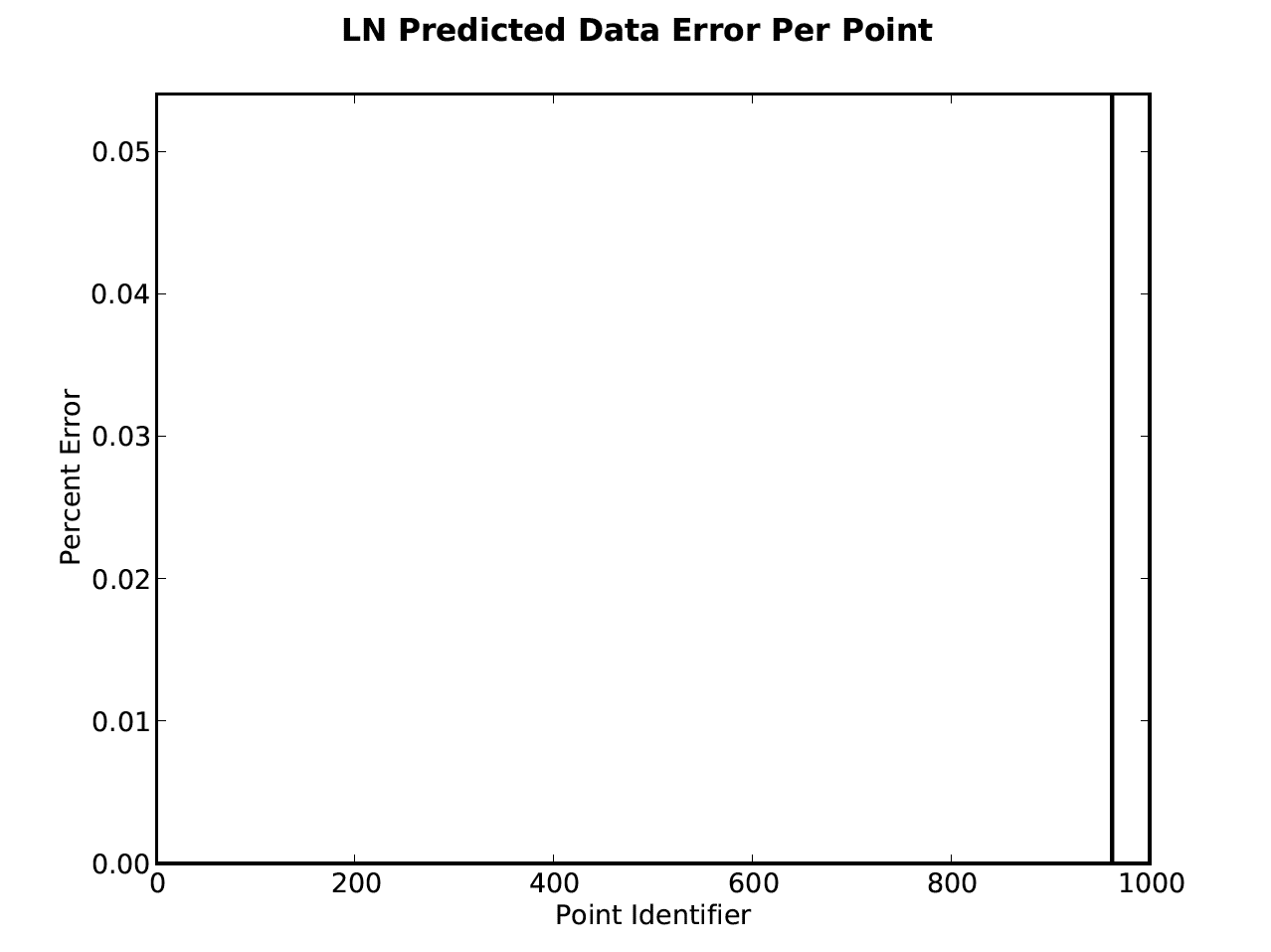
These representations poorly portray the differences in interpolation schemes since the problems are too complex for visual inspection. One major difference to note between the schemes is their error distributions. The linear nearest neighbor scheme appears to have a generally lower average with an occasional spike. The other three methods have a more uniform distribution. This can be seen in Linear Nearest Neighbor Error – Figure 8 and Hermite Neighbor Interpolation Error - Figure 9.



Cosine Neighbor Interpolation Results – Figure

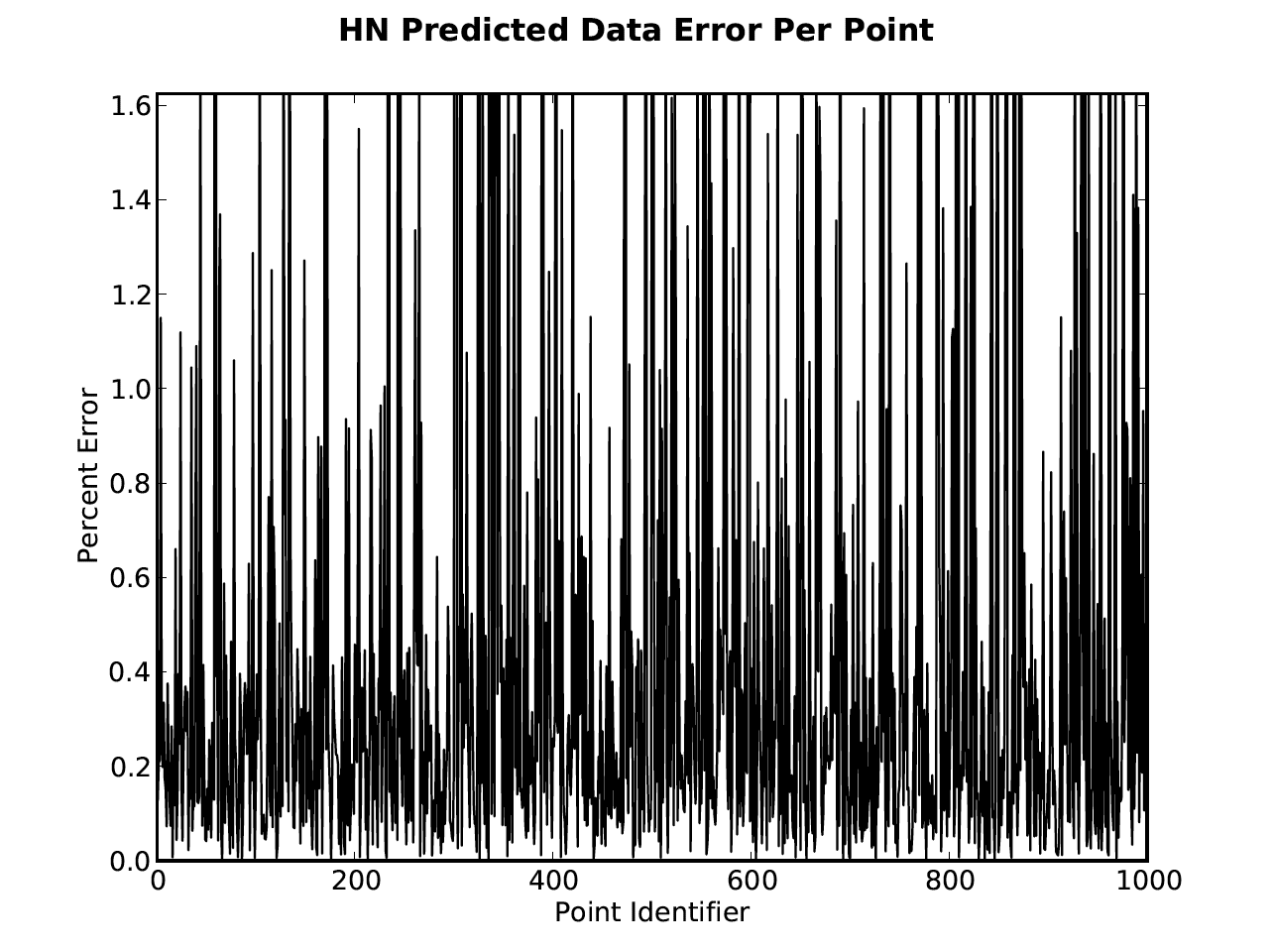


Hermite Neighbor Interpolation Results- Figure



Linear Nearest Neighbor Error – Figure

When the program is run, a plethora of plots are created to supply information, but for aspects such as error the terminal output is generally more useful for observation. Error Results - Table 2 provides the error results for four runs, one initial simple plane run to ensure that the code is working properly, 1 of each three-dimensional problem type discussed earlier in the report, and 1 run of a five dimensional problem. Currently the amount of error is not satisfactory for the gradients although this may not be possible for improvement. All errors are reduced if more training points are added into the problem.



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. 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.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Simple Plane Problem | | | | |
|  | Result Percent Error | | Gradient Percent Error | |
|  | Average | Max | Average | Max |
| LN | 0.00% | 0.00% | 0.00% | 0.00% |
| WN | 1.00% | 423.54% | 161.38% | 717.20% |
| CN | 3.91% | 3081.27% | 778.86% | 67067.49% |
| HN | 2.87% | 1613.00% | 358.85% | 8383.06% |
| Piecewise Problem | | | | |
|  | Result Percent Error | | Gradient Percent Error | |
|  | Average | Max | Average | Max |
| LN | 0.03% | 29.28% | 17.14% | 17135.67% |
| WN | 0.47% | 66.42% | 152.59% | 8698.89% |
| CN | 0.73% | 181.86% | 1008.23% | 226247.63% |
| HN | 0.84% | 127.81% | 466.49% | 130590.82% |
| Crate Problem | | | | |
|  | Result Percent Error | | Gradient Percent Error | |
|  | Average | Max | Average | Max |
| LN | 1.7% | 399.5% | 8564.3% | 3448795.3% |
| WN | 7.6% | 2277.4% | 20551.9% | 16849956.9% |
| CN | 9.0% | 1163.7% | 18747.5% | 5957799.6% |
| HN | 10.5% | 1567.8% | 6542.3% | 759336.5% |
| 5 Dimensional Problem\* | | | | |
|  | Result Percent Error | | Gradient Percent Error | |
|  | Average | Max | Average | Max |
| LN | 713% | 118629% | 8591137714690000% | 8228737272090000000% |
| WN | 17% | 4291% | 103582123653000% | 79139268871300000% |
| CN | 20% | 3845% | 20065237564300000% | 18891377453000000000% |
| HN | 27% | 5651% | 2672592143220000% | 2594958263020000000% |
|  |  |  |  | \* Contains 10 times as many training points as the other problems |

Error Results - Table