ABSTRACT

An Automated Method for Fitting Variograms from Finite Element Displacement Data in the Hot-To-Cold Process

Brandon Doolin
Department of Mechanical Engineering, BYU
Master of Science

Kriging is a linear, three-dimensional interpolation method for unstructured grids. The process centers around estimating a functions autocorrelation through a separate function called the variogram. The variogram can be sampled at all locations to determine what is termed the empirical variogram, but a model of this function must be determined in order to use in the Kriging process. Even if a poorly fitted variogram is used, Kriging will still give generally good results. However, for applications that require higher levels of precision, a well-fitted variogram is essential.

Blade un-running, or more commonly called hot-to-cold, is a process that uses a finite element solver to remove centripetal and aerodynamic loads from a hot mesh representation of the blade and provide a cold mesh of the blade. A geometry mapping must then take place to transform the hot geometry into a cold geometry useable by CAD systems i.e. a non-discretized version.

This project analyzes the shape and trends of the empirical variograms at different locations across both a cantilever beam and a rotor blade of similar sizes. Trends will be identified between the two to identify the best variogram models to fit to the empirical variogram. The models are then fitted automatically by different optimization techniques and different objective functions. Techniques include gradient-based methods and various gradient-free methods. Objective functions will include the sum of squares, the average of absolute differences, and the Hausdorff metric. These method/objective pairs are compared by their ”goodness of fit” and computational cost to determine the best technique and function to use in the Kriging process.

Keywords: variogram, hot-to-cold, optimization

CHAPTER 1. INTRODUCTION AND BACKGROUND

”Blade un-running”, or more commonly called ”hot-to-cold”, is a process that uses a finite element solver to remove centripetal and aerodynamic loads from a ”hot” mesh representation of the blade and provide a ”cold” mesh of the blade.

The un-running begins by meshing the geometry with finite element software and applying loads to the hot mesh. The result of this analysis is referred to as the “double deflected” mesh. From the double deflected mesh, a set of node displacements is calculated and the negative of these are input along with the original hot mesh into an iteration loop. At the beginning of the loop the displacements are applied to the hot mesh to create an approximation to the cold mesh. Next, the aerodynamic and centripetal forces are applied to the cold approximation to create a hot mesh approximation. If the difference between the hot mesh and the approximate hot mesh is within tolerance, then the cold mesh approximation is accepted as the actual cold mesh. Otherwise, the displacements from the cold approximation to the hot approximation are calculated, and the negative of these is input to the beginning of the loop. This loop is repeated until the correct cold mesh is found [1]. A diagram of this process can be seen in figure 1.1.

While understanding this hot-to-cold process is not completely necessary for this work, it is useful to understand the inputs and the outputs of the system in order to understand the context for variogram fitting, the focus of this work which will be discussed later. In this sense, the hot-to-cold process can be considered a black box that takes as input a hot (deformed) geometry file, and outputs a hot (deformed) mesh as well as a cold (undeformed) mesh.

The designer will be left with these three sets of data: a hot and cold mesh and one geometry representation. This geometry file differs from the mesh file in its placement of points or nodes. The mesh files have nodes placed at locations and in densities chosen by the meshing software. The geometry files have points chosen in specific locations and densities for use in the CAD system they come from. Consequently, these files almost always differ from each other.

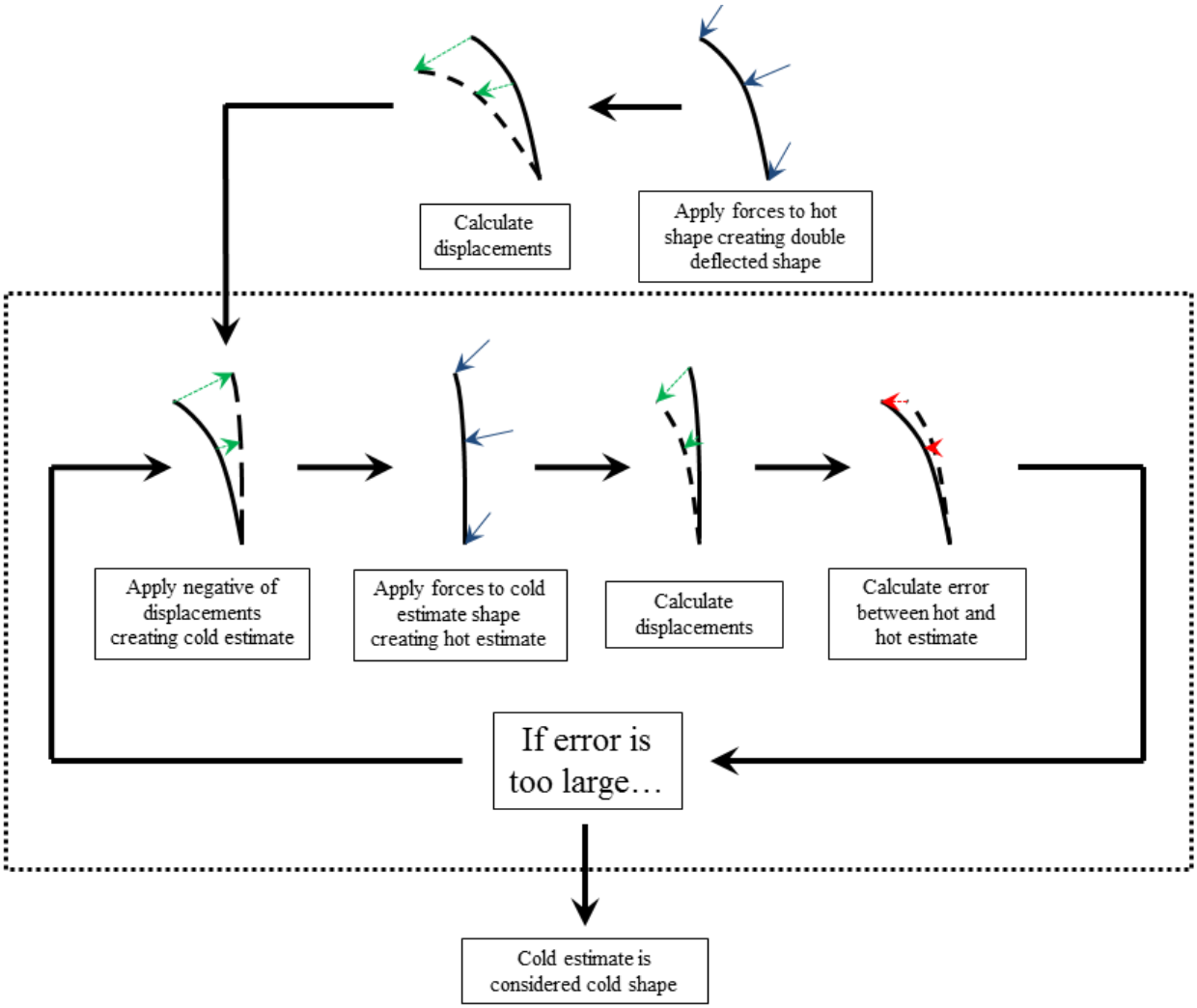


Figure 1.1: A flow chart illustrating the iterative loop for the hot-to-cold transformation

A cold geometry file is needed for many stages of analysis, even after the hot geometry has been determined. The cold file is used to perform off-design analyses as well as create manufacturing drawings and processes [2]. The meshes are not useable in many of these analyses and almost never in any of the manufacturing steps due to their discretized nature. Therefore, engineers must map their existing geometry file from its current hot state to a cold state.

This mapping can be accomplished by a simple interpolation scheme. The displacement from one hot node to its corresponding cold node can be calculated for each of the Euclidean directions. These displacement values are then interpolated for each hot geometry point.

One method of performing this interpolation is using Kriging. Kriging is an n-dimensional interpolation method on an unstructured grid. Kriging works by constructing weights based off of minimizing variance between the value at the point that is to be interpolated and the values of the surrounding points. The variance in this case is measured by a function called the semivariogram.

The semivariogram, more commonly referred to as just the variogram, is a measure of the dissimilarity or variance of function values based off of a separation distance in space of the points corresponding to the function values. It is defined as in equation (1.1).

$$\gamma(h) = \frac{1}{n}\sum_{i=1}^{n}[z(x_i+h) - z(x_i)]^2 \tag{1.1}$$

Although this equation cannot be determined analytically, it can be sampled at discrete locations. For all known points, the entire set of possible pairs is generated. For each pair, the semivariance is calculated and plotted. This plot is commonly referred to as the variogram cloud as it contains a large number of data points in a very scattered fashion. This cloud is a rough estimate of the variogram itself though it contains a considerable amount of noise [3].

Generally, to reduce the noise in the variogram cloud, the data is binned together and averaged into ranges of separation distance. Therefore, instead of a cloud of points at each specific h, the bins would represent all points in a range of hh. This tends to smooth out the noise in the variogram cloud (though noise is still present in this representation). This binned collection is commonly referred to as the empirical variogram. The empirical variogram gives a good insight into the shape and behavior of the variogram. However, the empirical variogram cannot be evaluated at arbitrary locations. This requires a function to be fitted to it first [4].

Although a function model need only be non-negative definite to be considered a licit variogram, some shapes of variograms appear very often. A set of function models that are guaranteed to be non-negative definite that match these shapes has been developed and are commonly used. Some of these common functions can be seen in figure 1.2.

Fitting a variogram has few rules to it; in fact, it is generally thought of as more of an art than a science. Usually a variogram model is chosen that best resembles the empirical variogram, then parameter values are manipulated until a “good-looking” fit is achieved. If a single variogram

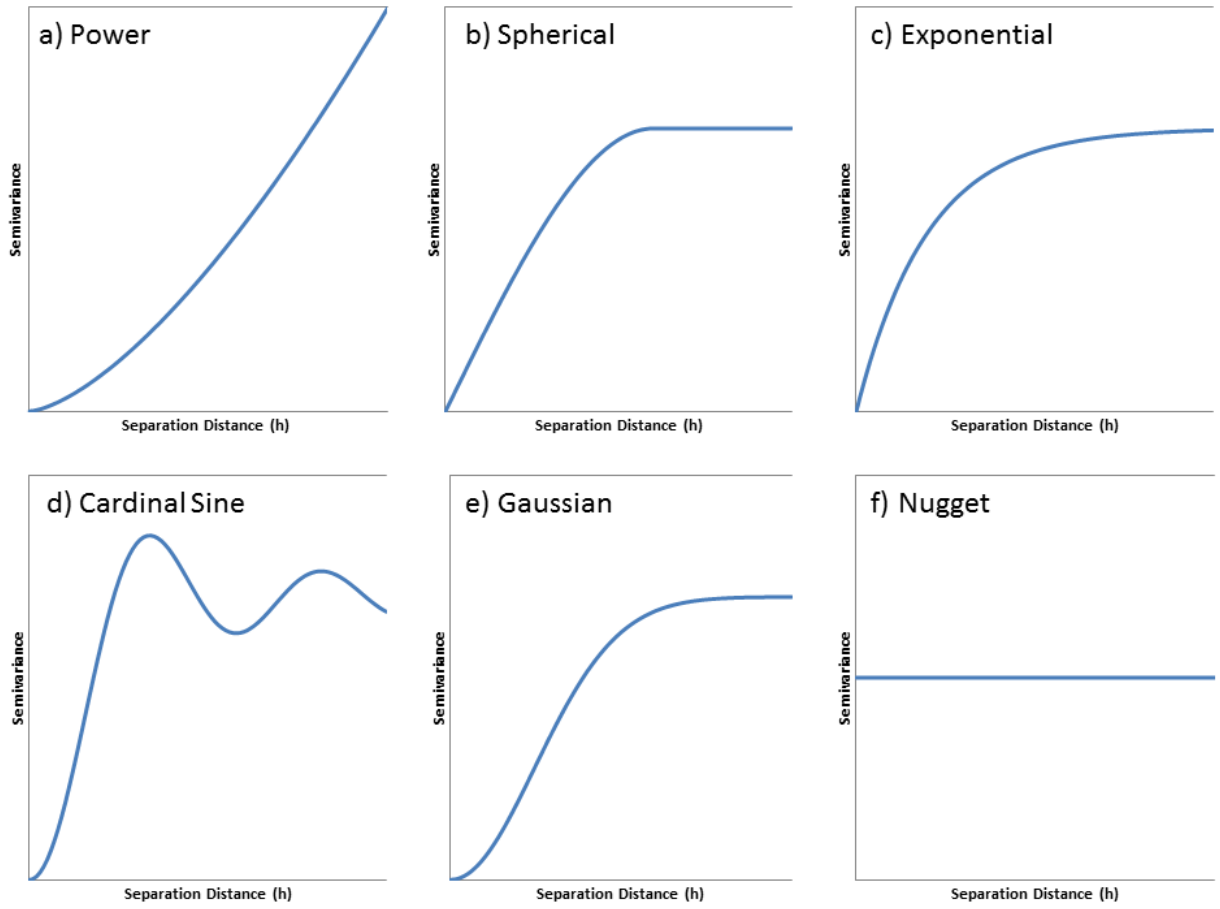


Figure 1.2: Different variogram models. From top left to bottom right: *a*) power, *b*) spherical, *c*) exponential, *d*) cardinal sine, *e*) Gaussian, *f*) nugget

model does not yield good results, another model can be added to the current one. This is possible because the sum of any two licit variograms is also a licit variogram [4].

Some have tried to fit variograms using more a more scientific approach such as least squares estimation. These methods often require non-linear solution techniques—such as the Levenberg-Marquardt algorithm—since many variograms are not linear in their parameters. These methods have had varying levels of success in fitting variogram models to empirical variograms [3, 5].

Hot-to-cold mapping with kriging requires that three variograms be calculated and fitted, one for each direction. Also, since meshes generally contain tens of thousands of nodes, kriging is much more feasible to do on small localities of the mesh instead of the entire mesh at once. This

means that these three variograms must be calculated and fitted at numerous localities when performing interpolation, possibly needing thousands of variograms. Therefore, this scale of problem requires an efficient, robust, and automated method for fitting these variograms.

This work seeks to create such a method by posing the problem as a general optimization problem instead of the more traditional least squares optimization approach for automation.

CHAPTER 2. METHOD

Least squares problems are essentially optimization problems where an analytical solution exists so long as the function is linear in its coefficients. When it is non-linear in its coefficients (as is the case for most variograms) a different approach must be taken.

At the heart of the problem is an optimization to try to minimize the difference between the sampled points and the fitted function. The least squares process uses the sum of squared differences as their difference metric, but this could also be accomplished using the sum of differences, the average difference, or the Hausdorff distance.

The least squares method also uses an analytical approach to minimizing the sum of squared differences. This, however, could be done by a numerical approach such as gradient-based optimizers (e.g. GRG, SQP, etc.) or gradient-free optimizers (e.g. genetic algorithm, particle swarm, etc.). While the analytical approach is faster and more accurate than a numerical approach, solutions only exist for the linear case.

This work seeks to investigate which method is the most suitable for optimizing the problem:

$$
\begin{aligned}
&\textit{minimize } f(x) \\
&\textit{by varying } x \in \mathbb{R}^7 \\
&\textit{subject to } c_i \geq 0, \quad i = 1, 2, \ldots, n
\end{aligned}
$$

f : objective (sum of squared differences, average difference, or Hausdorff metric)

x : vector of design variables for the variogram model

c : vector of bound constraints

For each combination of function and optimizer this work will evaluate several metrics to help determine which is best suited. The optimization schemes will be run to fit 141 different

empirical variograms. The time it takes to complete the optimization will be recorded to provide an estimation of algorithm speed. Also, the minimized value for each of the 141 variograms will be normalized, averaged, and recorded to provide an estimation of the quality of the fit. These results will then be compared to generally determine the best practice for variogram fitting with hot-to-cold data.

The fitting procedure will be done using a nested variogram model. The model will consist of the sum of a nugget model, a linear model (power model with exponent of 1.0), a near-quadratic model (power model with exponent of 1.99), a cardinal sine model, and a Gaussian model. The constraints for the optimization will consist of simple bound constraints to reduce the search space.

CHAPTER 3. RESULTS

Preliminary tests of a gradient-based optimizer (in this case Matlabs fmincon) with simple bound constraints yielded poor results. The optimizer tended to give a purely linear fit, which is far from ideal. This happened regardless if the bounds were normalized to order 1 or their original values. This can be seen in figure 3.1.

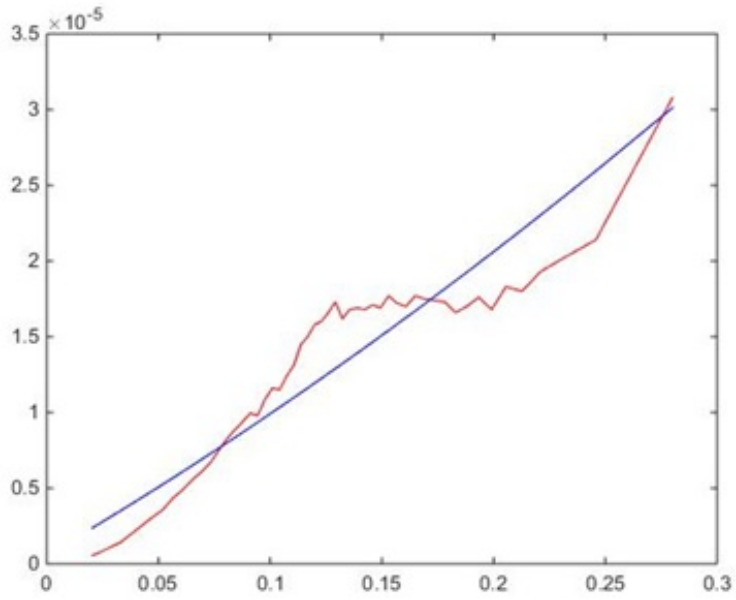


Figure 3.1: Empirical variogram 4 (of 141) (red) and the fitted variogram (blue) using fmincon. The unlabeled axes are separation distance (x) and semivariance (y).

Further investigation led to also normalizing all of the y-values to order 1 (multiplying the maximum y-value by a factor to yield 1 and multiplying all other y-values by the same factor). When this was done, the optimizer was able to find good solutions to the optimization using all the metrics (sum of squared differences, average differences, and Hausdorff distance. These can be seen in figure 4. It can be seen in figure 3.2 that the sum of squared differences had the best fit

(though it is the most expensive to evaluate) whereas the Hausdorff distance had the worst fit of the three.

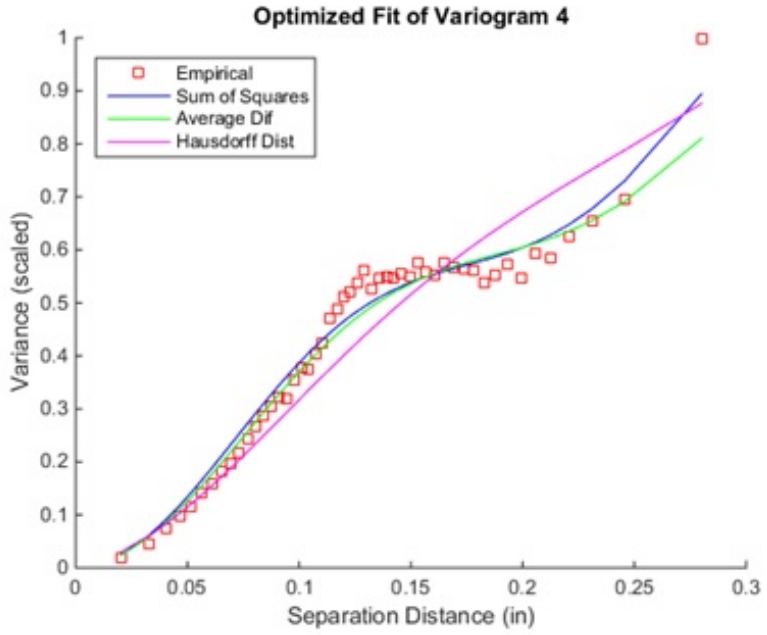


Figure 3.2: Empirical variogram 4(of 141) (red) and the fitted variogram using the sum of squared differences metric (blue), average difference (green) and Hausdorff distance (yellow).

To determine if the fits obtained represented a global optimum or a local optimum, the process was compared to global optimizers. A genetic algorithm, particle swarm optimizer, and simulated annealing optimizer were all implemented in Java. An optimization was run on the optimizers to determine the best parameters to use with these optimizers. These optimizers were then run on all 141 variograms with the mean output metric (sum of squares, average, or Hausdorff) calculated for comparisons. This was likewise done with the gradient-based optimizer. A comparison of these values can be seen in table 3.1.

It can be seen that the gradient-based optimizer did not, in fact, find the global minimum for any of the three metrics. It was especially poor at finding the global optimum when using the Hausdorff metric. During execution of the optimization with the Hausdorff metric (and occasionally with the average difference metric), the computer noted that the matrices being formed in the optimization were close to singular, likely due to difficulties calculating gradients with the Hausdorff metric (essentially a max and min function which is difficult to get a gradient from).

Table 3.1: "Goodness of fit" or accuracy results for each metric and optimizer

Optimizer	Mean Metric Value		
	Sum of Squares	Average	Hausdorff
Gradient-Based	0.0637	0.0217	0.0978
Simulated Annealing	0.0602	0.0205	0.0718
Particle Swarm	0.0641	0.0216	0.0738
Genetic Algorithm	0.0621	0.0210	0.0751

This leads to the question of trade-offs. Obviously the simulated annealing algorithm was the best at finding the global optimum. However, gradient-free methods tend to use more function calls to reach the optimum than gradient-based methods. The function calls were tallied for the full 141 variogram set and compiled as in table 3.2.

Table 3.2: Call totals for each optimizer and metric

Optimizer	Mean Function Call Totals		
	Sum of Squares	Average	Hausdorff
Gradient-Based	331,557	282,356	167,022
Simulated Annealing	2,094,937	1,893,418	2,084,481
Particle Swarm	3,769,800	3,444,000	3,497,700
Genetic Algorithm	4,011,600	3,264,600	2,774,100

Table 3.2 clearly shows that the gradient-based optimizer uses an order of magnitude less calls than the gradient-free methods. The other clear finding from the table is that simulated annealing uses considerably fewer function calls than the other gradient-free methods. By these two tables a clear conclusion is that the two best optimizers are the gradient-based and simulated annealing.

In determining the best metric to use, a comparison of cost needs to be considered as well as goodness of fit. Each of the three metrics must first compute the difference and sum it up. This is exactly what is done for the average difference metric, so it can be said that it has the base cost of these methods. The sum of squares metric does this and adds a multiplication. Depending on the computer architecture this added multiplication could add little or no time. Suffice it to label the cost as base+. The Hausdorff metric has the base cost plus a conditional comparison. Once

again, this conditional comparison will add a little or no additional execution time. Once again we can label its cost as base+.

Because the Hausdorff metric was more difficult to get a good fit with the gradient-based method as well as having a cost of base+, it can be eliminated as the metric of choice. To determine which of the remaining two metrics to use, a visual fit comparison was done using variograms 4, 9, 120, 126, and 138. This can be seen in figures 3.3 and 3.4.

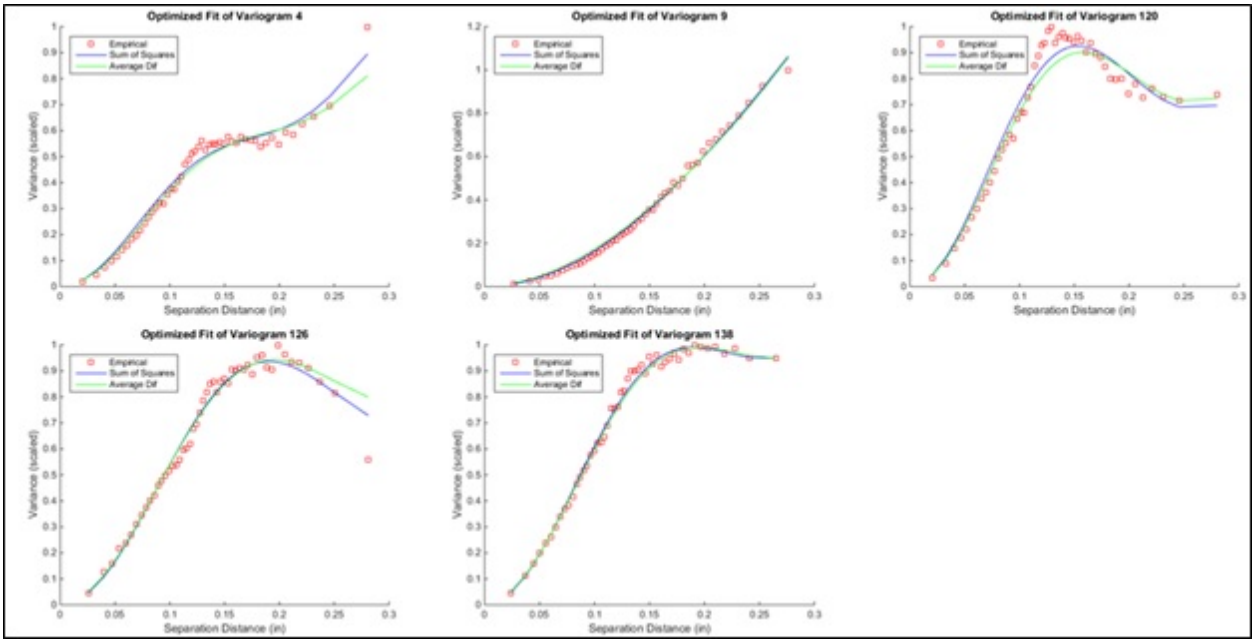


Figure 3.3: Fits to several empirical variograms using Matlabs fmincon with the sum of squares and average difference metrics.

Visually, the fits for the sum of squares metrics and average difference metrics were roughly the same for these five variograms using the gradient-based optimizer. The sum of squares tended to fit a bit better when large changes in curvature happened as in variogram 4, 120, and 126. However, this was not a large difference visually speaking.

The same trend that existed with the gradient-based optimizer also existed with the gradient-free method, but to a lesser extent. The sum of squares metric tends to exaggerate differences more than the average difference metric does. This is likely the cause for the observed difference in fit with both optimizers. However, this metric is still cheaper to use than the sum of squares metric.

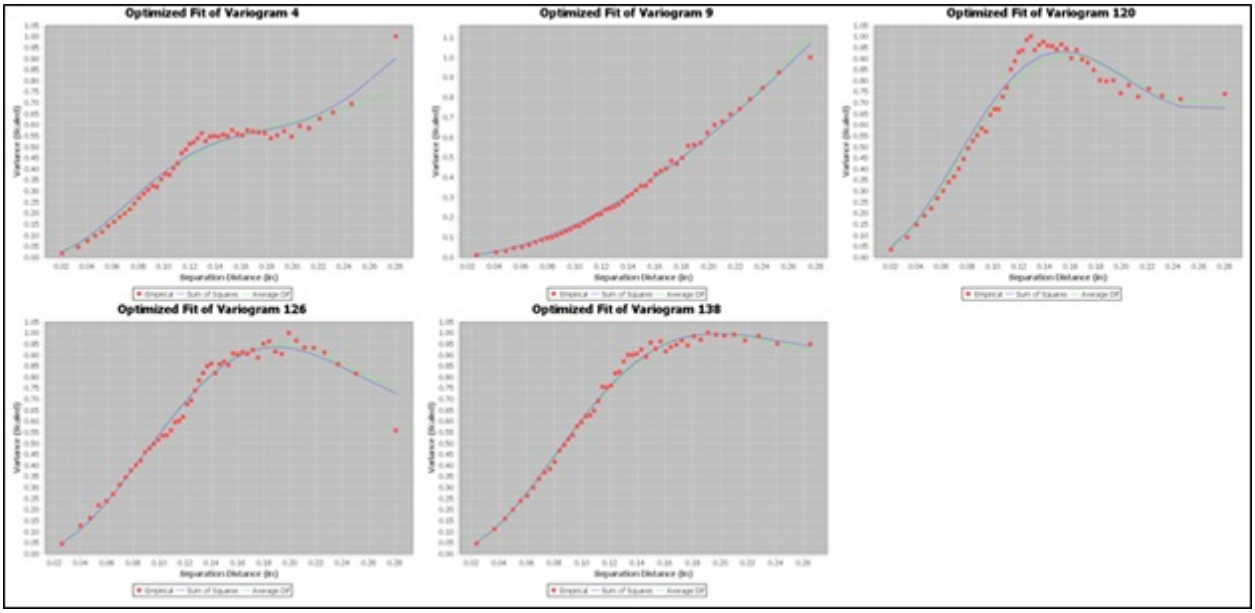


Figure 3.4: Fits to several empirical variograms using simulated annealing with the sum of squares and average difference metrics.

CHAPTER 4. CONCLUSION

Four optimizers and three metrics were evaluated for their viability in performing variogram fitting. All methods were found to be robust, though simulated annealing stood out as the most accurate while the gradient-based method was the fastest. The Hausdorff distance proved to be a less desirable metric when compared with the other two. The sum of squares yielded a better fit, but the average difference was computationally cheaper.

This gives the user several good options to choose from when performing automated variogram fitting based off of finite element displacement results. When the absolute fastest performance is needed, a gradient-based optimizer with the average difference metric will fit the best. On the other hand, when the absolute most accurate performance is needed, simulated annealing with the sum of squares metric is ideal. For different needs of speed and accuracy, different combinations of these 2 optimizers and metrics can be used.

REFERENCES

[1] Paldi, F., 2007. "Conversion of 'hot' geometry into 'cold' geometry using the finite element method program ansys®." Master of science, Engineering and Computational Engineering, Technical University of Berlin. 1

[2] Mahajan, A. J., and Stefko, G. L., 1993. "An iterative multidisciplinary analysis for rotor blade shape determination." In *AIAA/SAE/ASME/ASEE 29th Joint Propulsion Conference and Exhibit*, Vol. 29. 2

[3] Chiles, J.-P., and Delfiner, P., 2012. *Geostatistics: Modeling Spatial Uncertainty*., 2 ed. John Wiley & Sons. 3, 4

[4] Bohlin, G., 2005. Variogram analysis `http://people.ku.edu/~gbohling/cpe940/Variograms.pdf`. 3, 4

[5] Brunell, R. M., 1992. An automatic procedure for fitting variograms by cressie's approximate weighted least squares criterion Tech. rep., Southern Methodist University, October. 4