Volumetric Visualization of Non-Point Source Contaminant Flow Utilizing Four-Dimensional Geostatistical Kriging

A THESIS

Presented in Partial Fulfillment of the Requirements for the Degree Master of Science in the Graduate School of The Ohio State University

By

Eric J. Miller, B.S.CISEng

* * * * *

The Ohio State University

1994

Master's Examination Committee:

Dr. Duane Marble

Dr. Dale White

Dr. Roni Yagel

Approved by

[Signature]

Adviser

School of Natural Resources
This is dedicated to my wife, Lisa.
# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEDICATION</td>
<td>ii</td>
</tr>
<tr>
<td>LIST OF TABLES</td>
<td>vi</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>vii</td>
</tr>
<tr>
<td><strong>CHAPTER</strong></td>
<td></td>
</tr>
<tr>
<td>I Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Nature of the Problem</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Problem Statement</td>
<td>1</td>
</tr>
<tr>
<td>1.3 Objectives</td>
<td>3</td>
</tr>
<tr>
<td>1.4 Background</td>
<td>5</td>
</tr>
<tr>
<td>II Four-Dimensional Geostatistical Kriging</td>
<td>11</td>
</tr>
<tr>
<td>2.1 Geostatistics</td>
<td>11</td>
</tr>
<tr>
<td>2.2 The Sample Variogram</td>
<td>15</td>
</tr>
<tr>
<td>2.3 The Four-Dimensional Sample Variogram</td>
<td>18</td>
</tr>
<tr>
<td>2.4 Modeling the Four-Dimensional Sample Variogram</td>
<td>21</td>
</tr>
<tr>
<td>2.4.1 Omnidirectional Variograms</td>
<td>21</td>
</tr>
<tr>
<td>2.4.2 Directional Variograms</td>
<td>25</td>
</tr>
<tr>
<td>2.4.3 Models of Anisotropy</td>
<td>28</td>
</tr>
<tr>
<td>2.5 Four-Dimensional Geostatistical Kriging</td>
<td>30</td>
</tr>
<tr>
<td>2.5.1 Estimation of an Unknown Spatial and Temporal Location</td>
<td>32</td>
</tr>
<tr>
<td>2.5.2 Variogram and Covariance Function Estimation and Modeling</td>
<td>38</td>
</tr>
<tr>
<td>III Methodology</td>
<td>44</td>
</tr>
<tr>
<td>3.1 Statistical Analysis of Four-Dimensional Nitrate Concentration</td>
<td>44</td>
</tr>
<tr>
<td>3.1.1 Multi-Dimensional Filter Windows</td>
<td>52</td>
</tr>
</tbody>
</table>
3.1.2 Interactive Variogram Modeling .................................. 55
3.1.3 Horizontal-Spatial Variogram ...................................... 58
3.1.4 Vertical Variogram .................................................. 61
3.1.5 Temporal Variogram ................................................ 64
3.2 Four-Dimensional Kriging ............................................. 66
  3.2.1 Incorporation of the Variogram Model ......................... 67
  3.2.2 Incorporation of the Anisotropic Model ....................... 67
  3.2.3 Spatial and Temporal Estimation and Variance of Nitrate
        Concentrations .................................................... 72

IV Volumetric Visualization .............................................. 77
  4.1 Background .......................................................... 78
  4.2 Volume Representation ............................................. 79
  4.3 Disadvantages of Volume Graphics ............................... 82
  4.4 Advantages of Volume Graphics .................................. 83
  4.5 Volumetric Visualization and Geostatistical Kriging ........... 85

V Results and Discussion ................................................ 87
  5.1 Application of Four-Dimensional Kriging ....................... 87
  5.2 Volume Visualization of Spatial and Temporal Estimations and
        Variances .......................................................... 88
  5.3 Assessment of Four-Dimensional Kriging: Cross-Validation of
        Spatial and Temporal Estimations ............................... 89
  5.4 Future Work ........................................................ 94
    5.4.1 Search Strategies ............................................ 94
    5.4.2 Matrix Solutions ............................................. 95
    5.4.3 Parallel Architecture Implementation ..................... 96
    5.4.4 Dynamic Volume Visualization .............................. 96
    5.4.5 Probabilistic-Deterministic Modeling: Kriging and Hydrody-
          namic Equations ................................................ 97

VI Conclusion ........................................................... 99

APPENDICES

A Site Characteristics .................................................. 102
# List of Tables

<table>
<thead>
<tr>
<th>TABLE</th>
<th>Description</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Upper most multi-port $NO_3$ concentrations for local and regional wells in the MSEA test site.</td>
<td>7</td>
</tr>
<tr>
<td>2</td>
<td>Theoretical variogram parameters for each primary dimension.</td>
<td>74</td>
</tr>
<tr>
<td>3</td>
<td>Comparison between surface and volume graphics.</td>
<td>81</td>
</tr>
<tr>
<td>4</td>
<td>Cross-validation consistency criteria and results.</td>
<td>91</td>
</tr>
</tbody>
</table>
# List of Figures

<table>
<thead>
<tr>
<th>FIGURE</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>Interpolation of $NO_3$ concentrations based on upper most multi-port locations in a sample test site.</td>
<td>8</td>
</tr>
<tr>
<td>2</td>
<td>12</td>
</tr>
<tr>
<td>A simple example of an estimation problem. The dots represent fictitious sample points on a given profile to be estimated.</td>
<td>12</td>
</tr>
<tr>
<td>3</td>
<td>13</td>
</tr>
<tr>
<td>The seven sampled points represented as discrete locations along the deterministic modeled path of a bouncing ball.</td>
<td>13</td>
</tr>
<tr>
<td>4</td>
<td>17</td>
</tr>
<tr>
<td>Illustration of the $h_{ij}$ vector notation in two-dimensional space.</td>
<td>17</td>
</tr>
<tr>
<td>5</td>
<td>19</td>
</tr>
<tr>
<td>Illustration of the $h_{ij}$ vector notation in four-dimensional space.</td>
<td>19</td>
</tr>
<tr>
<td>6</td>
<td>23</td>
</tr>
<tr>
<td>An illustration of the omnidirectional selection of data pairs. A tolerance of ±1 is allowed on $</td>
<td>h</td>
</tr>
<tr>
<td>7</td>
<td>24</td>
</tr>
<tr>
<td>Experimental omnidirectional variogram plot of paired values at a lag distance of 5 ± 1.</td>
<td>24</td>
</tr>
<tr>
<td>8</td>
<td>26</td>
</tr>
<tr>
<td>The ranges associated with four directional variogram functions a plotted on a rose diagram in order to define the directional axes of anisotropy.</td>
<td>26</td>
</tr>
<tr>
<td>9</td>
<td>27</td>
</tr>
<tr>
<td>An illustration of the tolerances on $h$ for the selection of data pairs. A tolerance of ±1 is allowed on $</td>
<td>h</td>
</tr>
</tbody>
</table>
10 An example of a (1) variogram model and corresponding (2) covariance function shown with nugget effect defined by \((C_0)\), range defined by \((a)\) and sill defined by \((C_0 + C_1)\). ........................................... 29

11 Geometric \((a)\) and Zonal \((b)\) anisotropy as evident in the variogram function. The sill is represented for each of the dimensions \((x, y, z)\) is represented by \((sx, sy, sz)\) respectively. The range is represented by \((ax, ay, az)\). ................................. 30

12 Effects of (A) Isotropy and (B) Anisotropy on estimation at the V0. .......................... 36

13 Fitting of theoretical variogram function to an experimental variogram plot of \(\gamma(h)\) verses distance \(h\). ................................................................. 40

14 Three most commonly used transition models: Exponential, Spherical and Gaussian models with a range and sill of 1. ......................... 42

15 Histogram distribution of nitrate concentration (ppm). The histogram distribution approximates a lognormal distribution. ..................... 46

16 Histogram distribution of the log of nitrate concentration (ppm). The histogram pattern approximates a normal distribution. ..................... 47

17 Horizontal/Spatial distribution of both local and regional multi-port well locations. .......................... 49

18 Vertical distribution of both local and regional multi-port well locations. ........................ ..... 50

19 Temporal distribution of nitrate sampling for both local and regional multi-port wells. ............................. 51
20 Multi-dimensional filter windows used to mask influences of nearby data located in a dimension that is not currently being analyzed. (A) Filter windows corresponding to the Horizontal-Spatial dimension are specified by a particular point \(X_c, Y_c\) and radius \(r\). (B) Filter windows corresponding to the vertical dimension are specified by a particular depth \(V\) and a depth tolerance \(V_t\). (C) Filter window corresponding to the temporal dimension are specified by a particular time \(T\) and temporal tolerance \(T_t\).

21 X-Windows GUI interface incorporating Variogram Modeling Analysis Class (VMAC) of subroutines to quickly visualize: multi-dimensional data filtering, lag and distance spacing and tolerances, range, sill and nugget modification, and theoretical function overlay.

22 The results of multi-dimensional filter windows for the determination of the horizontal-spatial variogram. Figure (a) represents the horizontal-spatial constraint: all data located in the circle as defined by the center point (5, 5) with a radius of 25 is used. Figure (b) represents the vertical constraints: all data located at a depth of 265 ± 2 feet is used. Figure (c) represents the temporal constraints: all data collected at day 100 ± 5 is used. These constraints are used to filter out multi-dimensional data that may affect the horizontal-spatial variogram calculations.

23 Horizontal-spatial variogram mist and corresponding theoretical variogram function.

24 The results of multi-dimensional filter windows for the determination of the vertical variogram. Figure (a) represents the horizontal-spatial constraint: all data located in the circle as defined by the center point (0, 0) with a radius of 7 is used. Figure (b) represents the vertical constraints: all data located at a depth of 265 ± 15 feet is used. Figure (c) represents the temporal constraints: all data collected at day 100 ± 5 is used. These constraints are used to filter out multi-dimensional data that may affect the vertical variogram calculations.

25 Vertical variogram mist and corresponding theoretical variogram function.
The results of multi-dimensional filter windows for the determination of the temporal variogram. Figure (a) represents the horizontal-spatial constraint: all data located in the circle as defined by the center point (0,0) with a radius of 7 is used. Figure (b) represents the vertical constraints: all data located at a depth of 265 ± 2 feet is used. Figure (c) represents the temporal constraints: all data collected at day 60±60 is used. These constraints are used to filter out multi-dimensional data that may affect the temporal variogram calculations.

Temporal variogram mist and corresponding theoretical variogram function.

Illustration of multi-dimensional distance and direction variogram functions plotted on the same graph. The variogram functions corresponding to the ranges AX and AY represent the anisotropic horizontal and vertical vectors in the X and Y directions accordingly. The variogram function corresponding to the range AZ represents the anisotropic distance vector in the vertical dimension, and the variogram function corresponding to the range AT represents the temporal anisotropic vector. In this case geometric anisotropy is assumed as evident in similar sills and differing ranges in the corresponding dimensions.

Illustration of Anisotropic distance vectors. The vectors (AX, AY, AZ, AT) represent preferential flow in the corresponding X, Y, Z, and temporal directions.

The transformation of anisotropic distance vectors, (AX, AY, AZ) in (a) can be rotated around a three-dimensional coordinate system as defined by two angles of rotation. The first rotational angle, φ, is defined as the clockwise rotation about the z axis forming the new anisotropic axes (AX’, AY’, AZ’) and shown in (b). The second rotational angle, θ, is defined as the clockwise rotation about the new AY’ axis forming the new anisotropic axes (AX'', AY'', AZ'') and is shown in (c).

A volumetric torus.
Spatial and temporal distribution of geostatistical kriged area. Estimated values and corresponding variances were done for the volume corresponding to -20 to 20 in the X direction, 20 to -10 in the Y direction (Figure(a)), at a depth of 280 to 240 in the Z direction (Figure(b)). Temporal estimation was done for day 1 (April 1, 1991) through day 121 (July 1, 1991). The temporal distribution can be seen in Figure(c). 88

Histogram of kriging variance. .......................... 92

Scatter plot of actual nitrate concentration verses estimates obtained from ordinary kriging. .......................... 93

General site layout of the MSEA project including both local or site specific (S) and regional specific (R) multi-port well locations. The dashed rectangle over the agricultural plots in the North-East part of the site indicates the area of interest kriged in this investigation. 103

RGB table and opacities corresponding to nitrate levels used in the rendering of the kriged volumes of estimated concentrations. .......................... 105

Sequential kriged volumes of nitrate concentrations: Day 4 (April 4, 1991) through Day 18 (April 18, 1991) every two days. Animated sequence is from right to left, top to bottom. .......................... 106

Sequential kriged volumes of nitrate concentrations: Day 20 (April 20, 1991) through Day 34 (May 4, 1991) every two days. Animated sequence is from right to left, top to bottom. .......................... 107

Sequential kriged volumes of nitrate concentrations: Day 36 (May 6, 1991) through Day 50 (May 20, 1991) every two days. Animated sequence is from right to left, top to bottom. .......................... 108

Sequential kriged volumes of nitrate concentrations: Day 52 (May 22, 1991) through Day 66 (June 5, 1991) every two days. Animated sequence is from right to left, top to bottom. .......................... 109
Sequential kriged volumes of nitrate concentrations: Day 68 (June 7, 1991) through Day 82 (June 21, 1991) every two days. Animated sequence is from right to left, top to bottom. ........................................ 110

Sequential kriged volumes of nitrate concentrations: Day 84 (June 23, 1991) through Day 99 (July 7, 1991) every two days. Animated sequence is from right to left, top to bottom. ........................................ 111

Sequential kriged volumes of nitrate concentrations: Day 100 (July 9, 1991) through Day 114 (July 23, 1991) every two days. Animated sequence is from right to left, top to bottom. ........................................ 112

Sequential kriged volumes of nitrate concentrations: Day 116 (July 25, 1991) through Day 120 (July 29, 1991) every two days. Animated sequence is from right to left, top to bottom. ........................................ 113

RGB table and opacities corresponding to variance levels used in the rendering of the volumes of variances associated with the kriged estimated nitrate concentrations. ........................................ 115

Sequential variances associated with kriged volumes of nitrate concentrations: Day 4 (April 4, 1991) through Day 18 (April 18, 1991) every two days. Animated sequence is from right to left, top to bottom. ........................................ 116

Sequential variances associated with kriged volumes of nitrate concentrations: Day 20 (April 20, 1991) through Day 34 (May 4, 1991) every two days. Animated sequence is from right to left, top to bottom. ........................................ 117

Sequential variances associated with kriged volumes of nitrate concentrations: Day 36 (May 6, 1991) through Day 50 (May 20, 1991) every two days. Animated sequence is from right to left, top to bottom. ........................................ 118

Sequential variances associated with kriged volumes of nitrate concentrations: Day 52 (May 22, 1991) through Day 66 (June 5, 1991) every two days. Animated sequence is from right to left, top to bottom. ........................................ 119
Sequential variances associated with kriged volumes of nitrate concentrations: Day 68 (June 7, 1991) through Day 82 (June 21, 1991) every two days. Animated sequence is from right to left, top to bottom.  

Sequential variances associated with kriged volumes of nitrate concentrations: Day 84 (June 23, 1991) through Day 99 (July 7, 1991) every two days. Animated sequence is from right to left, top to bottom.  

Sequential variances associated with kriged volumes of nitrate concentrations: Day 100 (July 9, 1991) through Day 114 (July 23, 1991) every two days. Animated sequence is from right to left, top to bottom.
CHAPTER I

Introduction

1.1 Nature of the Problem

The Midwest is one of the most extensive agricultural areas in the country, producing approximately two-thirds of all U.S. corn and soybeans [1]. While the development of commercial fertilizers and pesticides has led to substantial increases in the productivity of corn and soybean crops in the region, improper agricultural uses of these agrichemicals has increasingly resulted in negative environmental impacts [1]. Improper use of agrichemicals have resulted in significant increases in non-point source contamination of both nitrate and pesticides in surface and subsurface water [1]. The growing importance of groundwater resources along with the health threat posed by agrichemical contamination emphasize the need for improved methods of predicting and monitoring non-point source contamination in groundwater systems.

1.2 Problem Statement

Understanding the processes affecting the transport and ultimate fate of subsurface non-point contamination is an integral part of water resource management. The detection and tracking of non-point source groundwater pollution are the first steps toward the development of effective management techniques. Effective water resources
management requires efficient and reliable methods of determining and characterizing the nature of subsurface contamination [43]. Depending on the particular contaminants and hydrogeological characteristics of the site, detection of subsurface contamination may be accomplished using several different direct and indirect point source methods including: borehole sampling, soil-gas analysis and geophysical techniques. The process of mapping and describing ground water contamination based upon data collected from a fixed, point source network is extremely complex with a large potential for error. Usually a limited number of samples in both spatial and temporal dimensions are available to assess a complex, continuous, dynamic process whose state variables change in both time and space [42]. Nevertheless, the characterization and extent of the contamination plume is generally assessed based on the point source data collected by subsurface contamination detection methods and traditional mapping techniques such as least squares estimators, spline fitting, and Gauss-Markov. These techniques, however, make interpretations based solely on point source concentrations and are unable to incorporate additional site specific hydrogeological information, such as anisotropic flow, into the estimation process. In addition, the characterization and extent of the contamination is traditionally done in a two-dimensional framework yielding, at most, concentration contour maps at different depths. Subsurface contamination, however, by its very nature, is a four-dimensional phenomenon. Two-dimensional analysis of problems that are inherently three and four-dimensional in nature may yield helpful qualitative displays but minimal quantitative significance [17].
1.3 Objectives

With the growing importance of groundwater resources coupled with the health threat posed by improper use of nitrate and other agrichemical contaminates, improved methods of predicting and monitoring non-point source contamination are needed. The first step in the classification and portrayal of a contaminant plume is to create an interpretation of the phenomenon by utilizing field data and mapping techniques. Utilizing field data consisting of nitrate concentrations analyzed from monthly sampling of multi-port well locations and \textit{a priori}, site specific information, this investigation will provide estimates of nitrate concentrations for unsampled spatial and temporal locations. This estimation of spatial and temporal nitrate concentrations from discrete, point source sampling data will allow for a dynamic, volumetric representation of non-point source nitrate contamination. Four-dimensional geostatistical kriging, a technique based on regionalized variable theory, will be introduced for the estimation/interpolation of spatially and temporally located and correlated data. Current techniques in volumetric visualization will then be utilized to dynamically visualize and analyze non-point source agrichemical flow though the subsurface system.

Utilizing spatial ($\{x,y,z\}$, multi-port well locations) and temporal ($\{t\}$, monthly sampling) nitrate concentrations, this investigation seeks to accomplish the following:

- Analyze spatial and temporal data using traditional exploratory statistical techniques including h-scatterplot and variogram analysis.
- Determine spatial and temporal continuity represented in the sampled data.
Two sampled data locations close together in space and time are more likely to have similar values than two data locations further apart [17]. This spatial and temporal continuity reflects the statistical relationship between sampled points and indicates how close in space and time sampled location may be useful in estimating unknown locations.

- Determine anisotropic or directional preferences in the sampled spatial data. In most earth science data sets, spatial continuity is often more prevalent in a particular direction. This anisotropic characteristic may be evident in the sampled data and consequently utilized in the estimation of unsampled locations.

- Determine the appropriate model for a random function which quantifies the spatial and temporal correlation and anisotropic nature of the data. A random function model recognizes fundamental uncertainties concerning complex processes and provides a mechanism for estimating values at unknown locations based on assumptions made concerning statistical characteristics of the phenomenon.

- Develop four dimensional \( \{x,y,z,t\} \) geostatistical kriging techniques for the estimation of unsampled spatial and temporal locations.

- Introduce an appropriate random function model into the system of kriging equations to estimate substance concentrations at specified, unsampled spatial and temporal locations.

- Utilize volumetric computer graphic techniques to visualize and navigate through
the spatial and temporal estimation of substance concentrations in order to draw inferences with respect to migration patterns of non-point source contaminants. Additional relationships may also be drawn between non-point source migration and associated agrichemical farming practices.

- Utilize volumetric computer graphic techniques to visualize three and four dimensional sampling errors associated with the estimation of unsampled spatial and temporal locations. Additional inferences may be drawn with respect to the location of future monitoring sites.

- Cross-validation of spatial and temporal estimations by removing a sampled value in space and time, estimating this location based on other sampled data, and comparing with the observed value.

- Provide a framework for the dynamic, volumetric construction of subsurface contamination from point source sampling methods, that may be utilized as both a comparison and tuning mechanism for deterministic hydrodynamic equations.

1.4 Background

Previous studies of subsurface characterization of groundwater contamination have documented the movement of contaminants, but have generally been prevented by their retrospective nature from providing quantitative insights into the processes that govern transport and fate. In most cases, it has not been feasible to quantify the initial mass of contaminant that enters the groundwater, nor to locate the emission
source precisely in space and time [26]. A further review of groundwater pollution mapping reveals that most previous analytical techniques have been confined to a two-dimensional framework utilizing traditional contouring methods [43]. Two-dimensional analysis of problems that are inherently three and four-dimensional in nature may not convey accurate descriptions of phenomenon in question, and subsequently may complicate matters by offering a simplified view of a more complex problem. In addition, automatic contouring of irregularly gridded data usually requires that the data values be interpolated to a regular grid. These interpolated values are usually less variable than the original data values and thus make the contoured surface appear smoother. This is aesthetically an asset, but a smoother surface often understates the variability of the original sampled values, and may be misleading from a quantitative view [17].

Non-point source contamination is generally characterized by a “large-scale”, relatively diffuse plume, originating from many smaller sources, whose locations are often poorly defined. Typically, there are no well-defined plumes in these cases but a large enclave of contamination with extremely variable concentrations [8]. Because of the large spatial scale and variability of non-point contamination, traditional two-dimensional representations are not adequate for representing the dynamic and amorphic nature of this phenomenon. The ability to visualize the dynamic nature of this diffuse contamination will allow for better interpretation and quantitative insight concerning this complex phenomenon.

It is difficult for the human brain to make sense out of the large volume of spatial
Table 1: Upper most multi-port $NO_3$ concentrations for local and regional wells in the MSEA test site.

<table>
<thead>
<tr>
<th>Well ID</th>
<th>X (ft x 100)</th>
<th>Y (ft x 100)</th>
<th>Z (ft x 2)</th>
<th>Day</th>
<th>$NO_3$(ppm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>R51</td>
<td>-43.09</td>
<td>7.11</td>
<td>243.83</td>
<td>9</td>
<td>0.33</td>
</tr>
<tr>
<td>R61</td>
<td>-37.16</td>
<td>-16.98</td>
<td>246.25</td>
<td>9</td>
<td>0.31</td>
</tr>
<tr>
<td>R71</td>
<td>-29.65</td>
<td>-28.61</td>
<td>241.30</td>
<td>9</td>
<td>0.35</td>
</tr>
<tr>
<td>R41</td>
<td>-21.73</td>
<td>4.70</td>
<td>267.49</td>
<td>9</td>
<td>0.62</td>
</tr>
<tr>
<td>R31</td>
<td>-20.04</td>
<td>19.59</td>
<td>244.04</td>
<td>10</td>
<td>4.00</td>
</tr>
<tr>
<td>S31</td>
<td>-12.31</td>
<td>15.78</td>
<td>260.95</td>
<td>4</td>
<td>2.17</td>
</tr>
<tr>
<td>S121</td>
<td>-9.53</td>
<td>-2.44</td>
<td>261.68</td>
<td>3</td>
<td>1.73</td>
</tr>
<tr>
<td>S41</td>
<td>-8.18</td>
<td>10.92</td>
<td>260.38</td>
<td>4</td>
<td>2.53</td>
</tr>
<tr>
<td>S81</td>
<td>-6.33</td>
<td>5.96</td>
<td>262.07</td>
<td>3</td>
<td>0.14</td>
</tr>
<tr>
<td>S131</td>
<td>-5.92</td>
<td>-10.31</td>
<td>268.21</td>
<td>3</td>
<td>0.79</td>
</tr>
<tr>
<td>S51</td>
<td>-5.17</td>
<td>15.50</td>
<td>243.93</td>
<td>3</td>
<td>0.43</td>
</tr>
<tr>
<td>S141</td>
<td>-4.76</td>
<td>-2.15</td>
<td>246.18</td>
<td>3</td>
<td>0.45</td>
</tr>
<tr>
<td>S21</td>
<td>-1.70</td>
<td>17.86</td>
<td>261.84</td>
<td>4</td>
<td>1.26</td>
</tr>
<tr>
<td>S91</td>
<td>-0.88</td>
<td>1.61</td>
<td>263.61</td>
<td>2</td>
<td>0.95</td>
</tr>
<tr>
<td>S111</td>
<td>-0.72</td>
<td>-4.23</td>
<td>263.66</td>
<td>3</td>
<td>1.62</td>
</tr>
<tr>
<td>S11</td>
<td>1.46</td>
<td>15.06</td>
<td>276.77</td>
<td>4</td>
<td>0.41</td>
</tr>
<tr>
<td>S101</td>
<td>1.53</td>
<td>5.13</td>
<td>246.38</td>
<td>2</td>
<td>0.67</td>
</tr>
<tr>
<td>R21</td>
<td>5.01</td>
<td>11.22</td>
<td>245.15</td>
<td>10</td>
<td>2.89</td>
</tr>
<tr>
<td>R81</td>
<td>5.65</td>
<td>-34.15</td>
<td>262.20</td>
<td>10</td>
<td>0.94</td>
</tr>
<tr>
<td>S71</td>
<td>5.71</td>
<td>7.70</td>
<td>260.85</td>
<td>38</td>
<td>25.22</td>
</tr>
<tr>
<td>S61</td>
<td>9.95</td>
<td>5.24</td>
<td>278.77</td>
<td>2</td>
<td>0.31</td>
</tr>
<tr>
<td>R11</td>
<td>16.96</td>
<td>3.39</td>
<td>247.10</td>
<td>9</td>
<td>0.54</td>
</tr>
</tbody>
</table>

and temporal data associated with observations of natural phenomenon. Whereas numerical and statistical methods are useful for solving these problems, visualization techniques can provide additional insights that may be missed by tabulated statistical methods [12]. For example, Table 1 consists of both spatial locations and nitrate concentrations for the uppermost multi-port well locations from a sample test site. In this tabular representation, nitrate concentrations may be used to calculate varying degrees of statistical moments including mean, mode and median contaminant concentrations for particular times. Spatial correlation, the continuity between concentrations with respect to physical location, however, is difficult, if not impossible to ascertain from tabular representation. Visualization techniques, however, can be
used to immediately convey a general sense of this spatial information. As shown in Figure 1, the same sample points as shown in Table 1 are used to create a spatial interpretation of the distribution of concentrations. Immediately the analyst is shown spatial variability in concentrations that is not readily obtained from the tabular representation.

![Image of concentration map]

Figure 1: Interpolation of NO₃ concentrations based on upper most multi-port locations in a sample test site.

Visualization is an important component of any effort to understand, analyze or explain natural phenomenon, and will become increasingly important as larger volumes of digital spatial data become more unmanageable [5]. Many natural phenomenon are intrinsically volumetric and dynamic in nature. Volumetric Visualization is a method of representing, manipulating and rendering data that is volumetric in nature. Unlike traditional graphics techniques, which represent three-dimensional
objects by surfaces and edges, approximated by polygons and lines, volumetric visualization represents three-dimensional objects using discretized entities called voxels [46]. Each voxel has one or more associated values which represent some measurable property (e.g. color, opacity, density, material, velocity, etc.). A voxel is a cubic unit of volume centered at the integral grid point. As a unit of volume, the voxel is a three-dimensional counterpart of the two-dimensional pixel, which represents a unit of area [21].

Spatial and temporal databases and data-structures are important and well established subjects of interest in both computer science and geography. There are many difficult issues concerning the concept and representation of spatial and temporal data. It is important to recognize that this investigation does not attempt to solve many of the complex issues and questions surrounding temporal data. Time, with respect to this investigation, is a direct, measurable extension to three-dimensional space. Every \( \{x,y,z\} \) coordinate undergoes a state change, at time \( t \), when some measurable attribute changes. In essence, time is defined here as a list of state changes at a specific spatial location. In a basic, topological sense, subsurface contamination may be defined by an amalgamation of discrete, contiguous spatial locations, \( \{x,y,z\} \), of contaminant concentrations whose degrees of concentration change with respect to time \( t \). In other words, time is represented as the list of substantive changes at a specific spatial location. With respect to this analogy of time, by the subsequent gathering of data from subsurface contaminant detection methods over a period of time, spatially and temporally referenced contaminate concentrations may be ob-
tained. This investigation will introduce a mechanism for estimating concentrations at unsampled locations in space and time based on the statistical correlations, the continuity between spatially and temporally referenced point source concentrations, and additional a priori, site specific information. Utilizing this technique, this investigation intends to classify, portray, and visually navigate through dynamic non-point source nitrate contamination in order to better understand complex hydrogeological flow regimes that are impacted by agricultural management systems.
CHAPTER II

Four-Dimensional Geostatistical Kriging

2.1 Geostatistics

In many environmental sciences, such as hydrogeology, geology, meteorology and oceanography, data is inherently four dimensional in nature. The ability to both visualize and animate a spatial and temporal data set allows the analyst to see complex patterns and trends associated with the given phenomenon. Detailed spatial data is often needed to provide a contiguous, volumetric representation of natural phenomenon. Detailed temporal data is often needed to provide a continuous, dynamic representation. Spatial and temporal data associated with environmental processes however, are difficult, if not impossible to collect at a scale sufficient to represent the phenomenon in both a contiguous and continuous manner. Environmental science, field-data sampling techniques generally result in relatively sparse, spatial and temporal point source samples. Unfortunately, point source sampling methods do not excuse the analyst from attempting to understand the processes and visualize the natural phenomenon in question. Estimation at unsampled spatial and temporal locations is consequently required for the analyst to understand and describe the phenomenon.
Figure 2: A simple example of an estimation problem. The dots represent fictitious sample points on a given profile to be estimated.

Perhaps the most desirable information that can be incorporated into the problem of estimation is a detailed description of underlying processes with respect to how the phenomenon was generated. For example, Figure 2 consists of seven locations represented on the $x$ axis and seven values represented on the $v$ axis. The data by itself tells the analyst virtually nothing about the profile of the process or how the phenomenon was generated. Estimation of the values at unsampled locations requires a model of how the phenomenon behaves. Without a model, only sampled values are available, and no inferences can be made of the values at unsampled locations [17]. If however, the physical or chemical process that generated the data was known in sufficient detail, then an accurate description of the process may be generated from
only a few sampled observations. Ideally, this would be the case, and a deterministic model considering all characteristic information would be used. Figure 3, for example, shows the deterministic model of the seven sampled points represented as discrete points along the path of a bouncing ball. In this case, knowledge of the physics of the problem, including initial trajectory and velocity, elasticity and radius of the ball and various other parameters, allows for the calculation of the exact trajectory of the ball. In the case of deterministic modeling, requiring a complete understanding of the underlying processes, an accurate recreation and prediction of the phenomenon may be made based on only a few sampled observations.

![Diagram](image)

Figure 3: The seven sampled points represented as discrete locations along the deterministic modeled path of a bouncing ball.

Unfortunately, very few earth science processes are understood well enough to permit deterministic modeling [17]. Where the fundamental physical and chemical
processes may be known, the variables of interest in earth science data sets are typically the results of vast processes and complex interactions that are not able to be described quantitatively. For the majority of earth science data sets, the analyst is forced to admit that there are complex processes too difficult to understand. There is a degree of uncertainty or "probability" about how the phenomenon in question behaves between sampled points. In probabilistic modeling, the available sample data are viewed as the result of some random function [17]. Though the word random often connotes "unpredictable," it turns out that viewing data as the outcome of some random process does indeed help with the problem of predicting unknown values. Viewing data as generated by a random function not only provides estimation procedures that, in practice, have proved to be accurate, but also provides the ability to gauge the accuracy of the estimation [17]. The random function recognizes fundamental uncertainties concerning complex processes and provides the analyst with tools for estimating values at unknown locations based on assumptions made concerning statistical characteristics of the phenomenon. This statistical approach to the estimation of attribute values at spatial and temporal locations is often referred to as geostatistics.

The term "geostatistics" designates the statistical study of natural phenomena. Matheron was the first to use this term extensively: "Geostatistics is the application of the formalism of random functions to the reconnaissance and estimation of natural phenomena" [28]. Geostatistics is concerned with regionalized variables, which have continuity from point to point but vary in a manner too complex for simple mathe-
mational description [32]. As stated before, the estimation at unsampled locations first requires a model of how the phenomenon behaves: without a model, one has only the sampled data and no inferences can be made about the unknown values. One of the important contributions of the geostatistical framework is the emphasis it places on the underlying model. Unlike many other methods of interpolation, including distance weighting, spline fitting, least squares estimation and Gauss-Markov techniques, geostatistical estimation methods clearly state the model on which they are based [17]. Many traditional statistical tools are useful in applying qualitative analysis to a wide variety of natural phenomenon; many others can be utilized to develop quantitative answers to specific questions. Unfortunately, most classical statistical methods make no use of the spatial information intrinsic to the natural phenomenon data sets. Geostatistics offers a new way of describing the spatial continuity that is an essential feature of many natural phenomena and provides adaptations of classical regression techniques to take advantage of this continuity [17].

2.2 The Sample Variogram

Spatial continuity exists in most earth science data sets. Two data values close to each other in space are more likely to have similar values than two data values further apart. Tools used to describe the relationship between two variables can also be used to describe the relationship between the value of one variable and the value of the same variable at nearby locations [17].

As an example, consider a sample site, \( S \), for which sampling methods are being conducted to monitor or observe a particular phenomenon. The sampled observations
can be defined as \( z(v_i) \) for all \( i = 1 \ldots n \) where the function \( z() \) is used to represent the complex processes corresponding to the phenomenon; \( v_i \) identifies a sampled coordinate position represented in space and time and \( n \) represents the number of sampled values. If the assumption is made that the sampled values, \( v_i \), which are generated by the function \( z() \), are the result of large and complex interactions that are not able to be described quantitatively, then each regionalized variable \( z(v_i) \) can be considered a realization of a certain random variable \( Z(v_i) \). This set of random variables is called a random function and is written as \( Z^*(v_i) \) for a given \( S \) [19].

One of the oldest methods of defining space dependency between neighboring observations is through autocorrelation [40]. When the neighboring observations are distributed two-dimensionally, two-dimensional autocovariance functions may be used to ascertain the spatial dependency [40]. However, when interpolation between measurements is needed, a more adequate tool is needed to measure the correlation between measurements and the spatial continuity of our random function. This is the \textit{semivariogram}\(^1\) or more commonly called the \textit{variogram}, which is defined as

\[
\gamma^*(h) = \frac{1}{2} E \{ [Z(v_i) + Z(v_i + h)]^2 \}
\]  

(2.1)

in which \( E \) is defined to be the expected value of the set of random variables \( Z(v_i) \) which are separated by a distance \( h \) [19]. Which, in turn, can be estimated by the average squared difference between the paired values:

\[
\gamma(h) = \frac{1}{2N\{h\}} \sum_{(i,j)|h_{ij}=h} (v_i - v_j)^2
\]

(2.2)

\(^1\)The prefix \textit{semi} comes from the \( \frac{1}{2} \) in Equation 2.1. It has become common, however, to refer to the semivariogram simply as the variogram [17]
where $N(h)$ is the number of paired values, $v_i$ and $v_j$, whose corresponding $h_{ij}$ vector equals the lag vector $h$. The vector $h_{ij}$, as shown in Figure 4, is defined to be the difference of vectors $t_i$ and $t_j$ which correspond to the spatial locations of $v_i$ and $v_j$.

![Diagram showing $v_j = (x_j, y_j)$, $t_j - t_i = h_{ij}$, $v_i = (x_i, y_i)$, and $(0,0)$]

Figure 4: Illustration of the $h_{ij}$ vector notation in two-dimensional space.

With a single sample, $v_i$, all we know about our random function $Z(v_i)$ is one realization. If estimated values are required for unsampled locations an intrinsic assumption concerning the random function is needed. The intrinsic assumption requires the constraint of stationarity over the random function. A random function is defined as stationary if the statistics on the random variables ($Z(v_i + h)$) are the same for every vector $h$ [40]. In other words, all pairs of random variables separated by a particular distance $h$, regardless of their location, have the same joint probability
distribution [17]. A random function may be considered intrinsic if

\[ E[Z(v_i)] = m \quad \forall v_i \subset S \]  

(2.3)

where \( E \) is defined to be the expected value equal to a constant \( m \) for all samples inside of \( S \). In the practice of geostatistics, the adoption of the variogram function to represent the stationary random function satisfies the intrinsic hypothesis [17].

2.3 The Four-Dimensional Sample Variogram

Traditionally, the variogram function represents the stationary random function in a two-dimensional framework. Geostatistical analysis of a four-dimensional phenomenon, however, requires the extension of traditional geostatistical methods to include the third (vertical) and forth (temporal) dimensions. The typical method used to extend two-dimensional geostatistical methods into the third dimension is to broaden the intrinsic hypothesis to include the vertical dimension [19]. Similar methods are used in this investigation to broaden geostatistical methods from the third dimension to include the fourth. Thus, the random function \( Z(v_i) \) may be considered intrinsic in both spatial and temporal dimensions if

\[ E[Z(v_i)] = m \quad \forall v_i \subset S \]  

(2.4)

where \( E \) is the expected value, \( m \) is a constant, and \( v_i \) is now defined as a value in \( S \) which represents a location referenced in four-dimensional space: \( \{x_i, y_i, z_i, t_i\} \).

The four-dimensional spatial-temporal variogram may now be expressed similar to Equation 2.1 as:

\[ \gamma^*(h) = \frac{1}{2} E\{[Z(v_i) + Z(v_i + h)]^2\} \]  

(2.5)
and can once again be approximated by the average squared difference between the paired values:

$$\gamma(h) = \frac{1}{2N\{h\}} \sum_{(i,j)|h_{ij} = h} (v_i - v_j)^2$$ (2.6)

where $N(h)$ is the number of paired values, $v_i$ and $v_j$, whose corresponding $h_{ij}$ vector equals the lag vector $h$. The lag vector $h$, however, is now defined as $[x_i, y_i, z_i, t_i]^T$ to include the vertical and temporal component. The lag vector $h$ in four-space can be seen in Figure 5.

![Diagram](image)

Figure 5: Illustration of the $h_{ij}$ vector notation in four-dimensional space.

With the assumption of the intrinsic hypothesis, the spatial-temporal variogram is independent with respect to any spatial or temporal location. The spatial-temporal variogram is only dependent between the spatial-temporal “distance” between the data points. “Distance” in this context is used rather loosely when dealing with
spatially and temporally referenced data. A point may be sampled at one time, \( t_0 \), and then again at time \( t_1 \). There is no spatial difference in sample location, however, the "distance" in this case, is defined as a difference in time. In this investigation, time is considered a direct and measurable extension to space. Thus, the spatial-temporal variogram indicates over what "distance", and to what extent, values at a given point influences values at adjacent points; and conversely therefore, how close together points must be for a value at one point to be capable of predicting an unknown value at another [32]. With the additional representation of the temporal component, the variogram also encompasses the aspect of how close together sampling points in time may be capable of predicting unsampled temporal locations.

Of the many characteristics of earth science data sets, the pattern of spatial continuity is among the most important for the problem of estimation. If the observed phenomenon is very contiguous, then estimates based on only the closest samples may be quite reliable. On the other hand, if the phenomenon is erratic, then estimates based on only the closest available samples may be unreliable; for such an erratic phenomenon, good estimates may require the use of many more sample data points beyond the closest ones [17]. If the phenomenon is extremely static with respect to time, then temporal estimates may also be quite reliable. Consequently, if the phenomenon in question is extremely dynamic, temporal estimates based on available samples may also be unreliable. For such dynamic phenomenon, an increase in sampling frequency may be needed.
2.4 Modeling the Four-Dimensional Sample Variogram

As usually the case in any data-based methodology, geostatistical analysis is adversely affected by lack of information. In particular, obtaining an experimental variogram that describes the spatial and temporal variability of the data can be extremely difficult if there are not enough samples available. This is so because each experimental variogram value requires a minimum number of sample pairs for the variogram function to be stable and statistically significant [19]. The problems associated with the inference of statistical moments have been discussed by several authors [27, 30, 34]. Any analysis that requires statistical inference will suffer from sparse information.

2.4.1 Omnidirectional Variograms

The analysis of spatial continuity typically begins with an omnidirectional variogram for which the directional tolerance is large enough that the direction of any particular separation vector, \( h_{ij} \), is unimportant. With all possible directions combined into a single variogram, only the magnitude of \( h_{ij} \) is important. In the case of multidimensionally referenced data, omnidirectional variograms are generally classified according to the coordinate system in which the variables are being analyzed [17]. In the case of four-dimensionally referenced data, for example, omnidirectional variograms would be generated for the horizontal-spatial, vertical and temporal dimensions. The calculation of the omnidirectional variograms does not imply a belief that spatial continuity is the same in all directions; rather it merely serves as an initial starting point for assessing some of the parameters for the variogram calculations [17].
The variogram function, as defined in Equation 2.6, represents a summary statistic over the \( N(h) \) pairs of observations whose separation distance is exactly \(|h|\). If a large, and regularly gridded data set is available, this equation may suffice. If the data set is irregularly gridded, however, which is more characteristic of earth science observations, a more flexible representation of the variogram function is needed. This flexibility of the variogram function is incorporated by summing the \( N(h) \) pairs of observations whose separation distance is approximately \(|h|\) rather than exact. The approximate variogram equation defined by

\[
\gamma(h) = \frac{1}{2N(h)} \sum_{(i,j) | h_{ij}=h} (v_i - v_j)^2
\]

(2.7)
is more traditionally used due to the ability of including more observations and consequently obtain a better understanding of the spatial and temporal continuity evident in relatively sparse and/or irregularly gridded sampled data.

There are two distance parameters that are specified when analyzing spatial continuity with an approximate, omnidirectional variogram function. One is the lag spacing and the other is the lag tolerance. The lag spacing is defined as the incremental distance between successive pairing of variogram values, and the lag tolerance is defined as the allowance associated with the distance. As shown in Figure 6, the lag spacing and lag tolerance are used to ascertain omnidirectional paired values. A tolerance of \( \pm 1 \) is allowed on \(|h| = 5\) for any direction. In this case, any sampled location within the shaded area would be paired with the sample \( V0 \). If the samples are located on a pseudo-regular grid, the grid spacing is usually a good lag spacing. If the sampling is random, an initial lag spacing may be based on the average spacing
Figure 6: An illustration of the omnidirectional selection of data pairs. A tolerance of ±1 is allowed on |h| = 5 for any direction. Any sampled location within the shaded area would be paired with the sample V0.
between neighboring samples. The most common choice for the lag tolerance is half of the lag spacing [17].

\[ \gamma(h) \]

Figure 7: Experimental omnidirectional variogram plot of paired values at a lag distance of 5 ± 1.

The hypothetical omnidirectional variogram plot of paired values at a lag distance of 5 ± 1 can be seen in Figure 7. The omnidirectional variogram plot generally indicates continuity that is reflected in the sampled data irrespective of direction. Values with closer distances have a corresponding smaller result in the variogram function, \( \gamma(h) \), and consequently a larger result corresponds to greater distance. When dealing with random sampling locations, or relative sparse samplings, the variogram plot may be erratic. By combining and iteratively analyzing the variogram plot for various lag spacing and lag tolerances, the variogram plot generally smooths and spatial continuity may become apparent.
2.4.2 Directional Variograms

Once the omnidirectional variograms corresponding to each dimension are relatively "well-behaved", patterns of anisotropy evident in the data can be explored through the use of directional variograms. In many applications, prior knowledge of the axes of anisotropy may be known. In the case of subsurface contamination, for example, hydrogeological information concerning the contaminated aquifer may be helpful in choosing directions for variogram analysis. In this case, directional variogram analysis would be used to corroborate anisotropic characteristics. Directional variogram analysis generally indicate anisotropic characteristics evident in the data set by a corresponding increase in directional ranges. As shown in Figure 8, for example, the ranges of four fictitious directional variograms are plotted on a rose diagram and are used to indicate the anisotropic axes evident in the sampled data.

When ascertaining the directional variogram, two additional parameters other than the lag spacing and lag tolerance are needed. These parameters are defined as the angular direction and the angular tolerance. The angular direction indicates the particular direction from any given value. The angular tolerance is a tolerance neighborhood associated with the particular direction. Figure 9, for example, is a graphical representation of directional variogram calculations using both a directional angle and directional tolerance. A angular direction of 0, coupled with an angular tolerance of ±20 is allowed on |h| of 5 at a distance tolerance of ±1. Any sample falling within the shaded area would be paired with the sample V0.

By combining the "best-fit" lag spacing and lag tolerances for the omnidirectional
Figure 8: The ranges associated with four directional variogram functions are plotted on a rose diagram in order to define the directional axes of anisotropy.
Figure 9: An illustration of the tolerances on $h$ for the selection of data pairs. A tolerance of $\pm 1$ is allowed on $|h|$ and a tolerance of $\pm 20$ degrees is allowed on the direction. Any sample located within the shaded area would be paired with the sample $V0$. 
variogram, coupled with angular directions and tolerances, relatively smooth variogram functions may be obtained at specific directions. The difference in directional variogram plots, generally indicates anisotropic nature evident in the sampled data. By analyzing directional variograms for the horizontal-spatial and vertical dimensions, three dimensional spatial anisotropy evident in the data may be ascertained and used in the estimation of unsampled locations.

The concept of the directional variogram associated with temporal continuity is somewhat ambiguous. The anisotropic characteristics represented by changes in the temporal variogram structure at different "directions" are not as clear as in the spatial domain. Directional anisotropic characteristics evident in the temporal dimension are most likely a reflection of spatial anisotropic characteristics changing over time. This investigation does not attempt to analyze or incorporate directional temporal anisotropy. The variogram function in the temporal dimension, at this time, is only an indication of how continuous the phenomenon is over time.

2.4.3 Models of Anisotropy

As indicated earlier, the variogram indicates over what distance and direction values at one given spatial and temporal locations effect the estimation of a value at another. For example, utilizing Equations 2.5 and 2.6 with the magnitude of \( h \) equal to 0, a sampled value is being compared with itself. Though the value of the variogram for the magnitude of \( h \) equal to 0 is strictly 0, several factors, such as sampling error and short scale variability, may cause sampled values separated by extremely short distances to be quite dissimilar. As shown in Figure 10(1), this causes a discontinuity
Figure 10: An example of a (1) variogram model and corresponding (2) covariance function shown with nugget effect defined by \((C_0)\), range defined by \(a\) and sill defined by \((C_0 + C_1)\).

at the origin of the variogram. This discontinuity from 0 at the origin of the variogram at extremely small separation distances is called the nugget effect [17]. The nugget effect implies that values are highly variable at distances less than the sampling interval [7]. As the separation distance, \(h\), between the pairs of values increases, the corresponding variogram value, \(\gamma(h)\), will also generally increase. Eventually however, an increase in the separation distance, \(h\), no longer causes a corresponding increase in the average squared difference between pairs of values, \(\gamma(h)\), and the variogram reaches a plateau. The point at which \(\gamma(h)\) equals the overall variance is called the range and is defined by \(a\). At this point values are considered unrelated. The plateau where the variogram reaches the range is called the sill and it is defined by \(C_0 + C_1\) [17].

When dealing with multi-dimensionally referenced data, the variogram function may be classified according to the coordinate systems for the dimensions being analyzed. As mentioned earlier, in the case of four-dimensionally referenced data, var-
Figure 11: Geometric (a) and Zonal (b) anisotropy as evident in the variogram function. The sill is represented for each of the dimensions \((x, y, z)\) is represented by \((sx, sy, sz)\) respectively. The range is represented by \((ax, ay, az)\).

Variogram analysis may be generated for the primary dimensions: horizontal-spatial, vertical, and temporal. The variogram functions associated with each dimension analyzed generally change with distance or direction and have different ranges and/or sills. Anisotropic behavior is evident in the sampled data when the range associated with the directional variograms change, but the sill remain the same. This type of anisotropy is known as geometric anisotropy and is shown in Figure 11(a). In the case of zonal anisotropy, the sill changes with direction while the range remains constant. This type of anisotropy is shown in Figure 11(b).

2.5 Four-Dimensional Geostatistical Kriging

Geostatistical techniques, as mentioned earlier, are based on the probabilistic concept of a random function model built under the following assumptions:
- The data are statistically homogeneous (stationary).

- The statistics inferred from the sampled data are representative of the global population.

The spatial and temporal dependence between two variables at different locations within the area of interest is inferred and modeled from the sample data. A correlation function (either a variogram or covariance) is expressed as a function of the distance between samples. This correlation function is obtained from the samples and which is then modeled with an analytical equation. This mathematical model thus represents how “close” two samples are in space and time, and uses this information to interpolate values at unknown locations by using a generalized least-squares algorithm called kriging [19].

The kriging method of interpolation, which is based on the theory of regionalized variables while using the degree of autocorrelation between adjacent samples, estimates values for any coordinate position within the domain measured without bias and with minimum variance [40]. There are several derivations of the kriging method of interpolation. Due to the point source sampling techniques used in this investigation, a more traditional type of kriging defined as “ordinary kriging” will be discussed. Additional information concerning other kriging techniques including, co-kriging, block kriging, and universal kriging may be found in [19, 17, 28, 31].

Ordinary kriging is generally considered the “best linear unbiased estimator” as compared to other methods of interpolation [17]. Ordinary kriging is “linear” because its estimates are linear combinations of the available data; it is “unbiased” since it
tries to have \( m_R \), the mean residual or error, equal to 0; and it is "best" because it aims at minimizing \( \sigma_R^2 \), the variance of the errors [17].

The goals of ordinary kriging are ambitious and, in a practical sense, unattainable since \( m_R \) and \( \sigma_R^2 \) are always unknown. The best we can do is to build a variogram model of the data that we have observed and work with the average error and error variance associated with the model. In ordinary kriging, a probabilistic model is used, in which the bias and the error variance can both be calculated. From there, weights for the nearby samples can be chosen to ensure that the average for the model is exactly 0 and that the modeled error variance is minimized [17]. The set of weights that will produce unbiased estimates with the minimum estimation variance is directly dependent on the variogram model associated with the data.

2.5.1 Estimation of an Unknown Spatial and Temporal Location

Suppose that we want to estimate a unmeasured value \( z^*(v_0) \). At every point where we wish to estimate, a weighted linear combination of the available samples will be utilized:

\[
z^*(v_0) = \sum_{i=1}^{n} \lambda_i z(v_i)
\]

where \( n \) is the number of measured values \( z(v_i) \) and the \( \lambda_i \) are the corresponding weights attached to each measured value [19]. By taking \( z(v_i) \) as a realization of the random function \( Z(v_i) \) and assuming stationarity, the estimator becomes:

\[
Z^*(v_0) = \sum_{i=1}^{n} \lambda_i Z(v_i)
\]
thus we must determine the weights, \( \lambda_i \), before the estimation of our unmeasured value can be produced. There are numerous ways of distributing the weights: inverse of the square of the distance, inverse distances, and the inverse of the number of values are a few examples [40]. Kriging, as stated above, however, provides the best estimator for distributing the weights to ensure that the average for the model, \( \hat{m}_R \), is exactly 0 and that the modeled error variance, \( \hat{\sigma}_R^2 \), is minimized [17].

The system of equations that minimize the error variance, \( \sigma_R^2 \), often referred to as the ordinary kriging system, can be written in matrix notation as:

\[
\begin{align*}
\gamma & \cdot \lambda &= D \\
\begin{bmatrix}
\hat{\gamma}_{11} & \cdots & \hat{\gamma}_{1n} & 1 \\
\vdots & \ddots & \vdots & \vdots \\
\hat{\gamma}_{n1} & \cdots & \hat{\gamma}_{nn} & 1 \\
1 & \cdots & 1 & 0
\end{bmatrix}
\begin{bmatrix}
\lambda_1 \\
\vdots \\
\lambda_n \\
\mu
\end{bmatrix} &=
\begin{bmatrix}
\hat{\gamma}_{10} \\
\vdots \\
\hat{\gamma}_{n0} \\
1
\end{bmatrix}
\end{align*}
\]

(2.10)

in which \( \gamma \) is defined as the \((n+1)^2\) variogram matrix which describes the spatial and temporal continuity of the random function, \( \lambda \) is defined as the \((n+1)\) weight vector, \( \mu \) is defined as the Lagrange parameter which is used for converting a constrained minimalization problem into an unconstrained one [9], and \( D \) is defined as the distance vector.

To minimize the modeled error variance, the \((n+1)^2\) variances that describe the spatial continuity in our random function model need to be defined. The spatial continuity of the random function model is usually defined through the variogram, \( \gamma(h) \), however, the ordinary kriging system matrix is usually defined using the covariance function \( C(h) \). Thus the ordinary kriging system, as defined in Equation 2.10, is
more commonly written in terms of the covariance function:

\[
\begin{pmatrix}
\tilde{C}_{11} & \cdots & \tilde{C}_{1n} & 1 \\
\vdots & \ddots & \vdots & \vdots \\
\tilde{C}_{n1} & \cdots & \tilde{C}_{nn} & 1 \\
1 & \cdots & 1 & 0
\end{pmatrix}
\begin{pmatrix}
\lambda_1 \\
\vdots \\
\lambda_n \\
\mu
\end{pmatrix}
= 
\begin{pmatrix}
\tilde{C}_{10} \\
\vdots \\
\tilde{C}_{n0} \\
1
\end{pmatrix}
\]

in which \( C \) is defined as the \((n+1)^2\) covariance matrix, \( \lambda \) is the weight vector, \( \mu \) is the Lagrange parameter, and \( D \) is the distance vector.

The covariance function is used this investigation in solving the ordinary kriging matrices for the sake of convenience and computational performance. The covariance function, which is another way of representing the same data, is simply defined in terms of the variogram as:

\[
\gamma(h) = C(0) - C(h)
\]

Thus the covariance function \( C(h) \) is defined as:

\[
C(h) = C(0) - \gamma(h)
\]

and is represented in graphical form in Figure 10 (b).

It is common practice in geostatistics to calculate modeled variograms from the sampled data, and then for reasons of computational efficiency, to solve the ordinary kriging equations in terms of the covariances [17]. By utilizing the covariance function, the largest elements of the covariance matrix will be located on the diagonal. Thus for solving linear equations by methods of Gaussian elimination, such as the software utilized in this investigation [38], there is less need for a pivot search and the exchanging of rows.
To solve for the weights, both sides of Equation 2.11 are multiplied by $C^{-1}$, the inverse of the covariance matrix:

$$C \cdot \lambda = D$$

$$C^{-1} \cdot C \cdot \lambda = C^{-1} \cdot D$$

$$I \cdot \lambda = C^{-1} \cdot D$$

$$\lambda = C^{-1} \cdot D$$ (2.14)

Once the $(n+1)^2$ covariances have been chosen, the $C$ matrix and the $D$ vector can be built. The $D$ vector on the right hand side of Equations 2.10 and 2.11 represent a weighting scheme similar to that seen in inverse distance approaches [17]. The $D$ vector, however, contains a form of inverse distance weights in which the “distance” is not based on the geometric distance to the samples but rather upon “statistical distance”. For example, the covariance function as shown in Figure 10(2) provides decreasing weights up to some distance, $a$, and provides a weight of 0 for distances greater than $a$. As the magnitude of the vector $h$ increases, the corresponding weight decreases and consequently the corresponding term is less important relative to the estimation. As mentioned earlier, the variogram, and the corresponding covariance function, provide a mechanism for modeling spatial continuity as a function of distance and direction. Many environmental data sets exist where the data values are more continuous, i.e. anisotropic, along a certain direction. Anisotropic, and consequently isotropic, preferential directions can also be incorporated into the variogram structure. Thus the “statistical distance” that corresponds to the $D$ vector is the distance based on the covariance function as well as the isotropic or anisotropic nature of the data.
For example, Figure 12 consists of two similar estimations. V0 is being estimated based on the available data corresponding to V1, V2, V3, and V4. In Figure 12(a), the data are assumed to be isotropic. In this case the statistical distance is based on the corresponding covariance function which defines this data set. Consequently, the weights associated with V1 are the largest, corresponding to the shortest geometric distance from V0, and have the most influence on the estimation. In Figure 12 (b), however, the data are assumed anisotropic along the N60°E direction with a ratio of 3:1 to N30°W. In this case, V1 is no longer as influential. Rather the weights corresponding to V2, which lies in the N60°E direction, prove to be most relevant to the estimation of V0.

![Diagram of isotropic and anisotropic estimations](image)

**Figure 12**: Effects of (A) Isotropy and (B) Anisotropy on estimation at the V0.

Similar to the D vector, the C matrix records the distance in terms of statistical distance rather than geometric distance. The C matrix records statistical distances
between each sample and every other sample, providing the ordinary kriging system with information on the clustering of the available sampling [17]. If two samples are close together in space and time, this will be represented by a large entry in the \( C \) matrix. Two values far apart, consequently, will be represented by a small entry. The multiplication of \( D \) by \( C^{-1} \) adjusts the raw inverse statistical distance weights in \( D \) to account for possible redundancies between the samples [17]. The information on the distances to the various samples and the clustering between the samples is all recorded in terms of a statistical distance, thereby customizing the estimation procedure to site specific patterns of spatial and temporal continuity.

The ordinary kriging system therefore takes into account two of the most important aspects of estimation problems: distance and clustering [17]. Utilizing these aspects, the set of weights that will produce unbiased estimates with the minimum error variances for the random function model is simply defined as:

\[
\lambda = \frac{C^{-1}_{\text{Clustering}} \cdot D_{\text{Distance}}}{(2.15)}
\]

Utilizing this set of weights, the resultant estimate for any given spatial and temporal location is

\[
v_0 = \sum_{i=1}^{n} \lambda_i v_i \quad (2.16)
\]

where \( n \) is the number of measured values \( v_i \) and \( \lambda_i \) are the corresponding weights attached to each measured value.

As mentioned earlier, another powerful mechanism of the kriging method of interpolation is the ability to gauge the accuracy of the estimates. By utilizing these weights, the minimized estimation variance at any unsampled point in space and time
is:

\[ \hat{s}_R^2 = \hat{s}^2 - \sum_{i=1}^{n} \lambda_i \hat{C}_{i0} + \mu \]  \hspace{1cm} (2.17)

where \( \hat{s}^2 \) is the sill of the covariance structure, \( \lambda_i \) are the corresponding weights attached to each measured value, \( C_{i0} \) are the covariance function calculation for the distance \( D \) vector and \( \mu \) is the Lagrange parameter.

### 2.5.2 Variogram and Covariance Function Estimation and Modeling

The set of weights that will produce unbiased estimates with the minimum estimation variance is directly dependent on the choice of covariance function, and consequently the variogram, for the \( C \) and \( D \) matrices. The covariance model is an essential prerequisite for ordinary kriging. Though this makes kriging more time consuming than other interpolative techniques, the covariance function provides a flexible framework for incorporating site specific characteristics, such as preferential or anisotropic flow, that other techniques lack. The patterns of spatial and temporal continuity chosen for the random function model are usually taken from the continuity evident in the sample data set. These patterns of continuity, as described earlier, are represented by the variogram defined in Equation 2.2 and shown in Figure 7.

When the variogram is calculated for a sample site, the plot of \( \gamma(h) \) verses \( h \) generally results in an increasing distribution of points which represent spatial, and in this investigation, temporal, continuity as a function of distance and direction. This distribution of points is generally referred to as the "experimental variogram" and can be seen in Figure 7. The experimental variogram serves as an indicator of the spatial
and temporal continuity evident in the sample data. The ordinary kriging system defined in Equation 2.11, however, not only requires values of the covariance function (or variogram) between all pairs of sampled locations, but covariance values between all sampled locations and the unsampled locations where estimates are needed as well. With respect to the covariance matrix \( C \) as defined in Equation 2.11, every entry \( \hat{C}_{ij} \) represents a covariance value corresponding to the distance, \( h \), between the values \( v_i \) and \( v_j \). As mentioned earlier, however, the experiment variogram and covariance function only represent values at specific lag distances \( h \) and tolerances. Values between each pair of values are not represented in the experimental variogram or covariance functions. With respect to the distance vector \( D \) as defined in Equation 2.11, a covariance value \( \hat{C}_{n0} \) is needed for every distance between the \( n \) sampled locations and the location to be estimated. Once again, the sample experimental variogram and covariance functions only reflect values at specific lag distances \( h \) and tolerances. The covariance values needed for both the \( C \) matrix and \( D \) vector may or may not be present in the discrete calculations represented in the experimental variogram. Thus, the ordinary kriging system will undoubtedly require variogram values other than those discrete, known values evident in the experimental variogram.

One method of obtaining variogram values at all necessary locations is to simply interpolate between the known values of the experimental variogram. Although this method will provide numbers, it introduces a significant problem: the solution of the ordinary kriging equations derived using these interpolated numbers may not exist or may not be unique. In other words, the kriging matrices built using such techniques
are not likely to be *positive definite* [17]. In order to ensure that the ordinary kriging equations have a unique and stable solution, the left hand matrix $\mathbf{C}$ in Equations 2.11 must satisfy the mathematical condition known as positive definiteness [17].

\[
\gamma(h) = 1 - e^{-\frac{3h}{a}}
\]

![Image](image.png)

Figure 13: Fitting of theoretical variogram function to an experimental variogram plot of $\gamma(h)$ versus distance $h$.

One way of satisfying the positive definite condition of the ordinary kriging system is to use only existing functions that are known to be positive definite. In other words, utilizing the experimental variogram, a corresponding "theoretical variogram" may be utilized to represent the spatial and temporal continuity of the random function which represents the sampled data. For example, Figure 7 represented a hypothetical "experimental variogram" plot of $\gamma(h)$ and $h$. A "theoretical variogram" function fitted to the same data can be seen in Figure 13.

Fitting a theoretical variogram to the experimental variogram is one of the most
important aspects of the applications of regionalized variable theory and may be one of the major sources of ambiguity [41]. Nevertheless, a “good” fit of the theoretical variogram to the observed variogram allows $\gamma(h)$ to be confidently estimated for any value of $h$, rather than only the sampled points [32].

Theoretical variogram models can be conveniently separated into two types; those that reach a plateau and those that do not. Theoretical variogram functions of the first type are often referred to as transitional models. The plateau they reach, as mentioned earlier, is called the sill and the distance at which they reach this plateau is called the range. Theoretical variogram functions of the second type, those that do not reach a plateau, continue increasing as the magnitude of $|h|$ increases. Such models are often necessary when there is a trend or drift in the data values [17], or when the models correspond to an infinite capacity of dispersion, for which neither the variance or covariance of the model can be defined [19].

The main theoretical variogram models are defined as:

**Spherical Model:**

$$\gamma(h) = \begin{cases} C_0 + C_1 \left( \frac{3h}{2a} - \frac{h^3}{2a^3} \right) & \text{if } h \leq a \\ C_0 + C_1 & \text{otherwise} \end{cases}$$  \hspace{1cm} (2.18)

**Exponential Model:**

$$\gamma(h) = C_0 + C_1 (1 - e^{-\frac{3h}{a}})$$ \hspace{1cm} (2.19)

**Gaussian Model:**

$$\gamma(h) = C_0 + C_1 (1 - e^{-\frac{3h^2}{a^2}})$$ \hspace{1cm} (2.20)

**Linear Model:**

$$\gamma(h) = \begin{cases} C_0 + \alpha h & 0 \leq h \leq a \\ C_0 + C_1 & \text{otherwise} \end{cases}$$  \hspace{1cm} (2.21)

where $\alpha$ is defined as a constant slope. The graphical representation of these theoretical models are shown in Figure 14.

The set of weights that will produce estimated values in unsampled spatial and temporal locations is directly dependent on the theoretical function chosen to repre-
Figure 14: Three most commonly used transition models: Exponential, Spherical and Gaussian models with a range and sill of 1.
sent the spatial and temporal continuity of the phenomenon. Experimental variogram structures are generated for each of the sampling dimensions of the random variables. Theoretical functions are then fitted to each dimensional experimental variogram in order to generate positive definite values for any lag distance corresponding to the appropriate dimension. Geostatistical analysis of spatial and temporal continuity evident in a data set, however, is often a frustrating and difficult process. Multi-dimensionally referenced data, sparse data samplings, irregular sample distributions, and anisotropically located sampling locations are all problematic factors when trying to assess the statistical correlations between sampled values. The many combinations and permutations of distance lag spacing, lag tolerances, angular directions and directional tolerances provide additional difficulties when analyzing variograms in different dimensions. A thorough statistical analysis of the data, and a good understanding of the genesis, are general prerequisites for the assessment of spatial and temporal continuity of the given phenomenon.
CHAPTER III

Methodology

3.1 Statistical Analysis of Four-Dimensional Nitrate Concentration

The initial step of assessing the spatial and temporal continuity of a given phenomenon is the exploratory analysis of the available sampled information. In order to utilize traditional statistical tools, such as histograms, scatterplots, and normal probability plots, and spatial statistical tools, such as variogram plots, the available sampled information was managed into a more coherent form.

This investigation was conducted at the Management Systems Evaluation Area (MSEA) site in Piketon, Ohio. Multi-port well sampling of nitrate has occurred monthly from April, 1991 through September, 1992. Additional hydrogeological characteristics and well locations of the MSEA site can be found in Appendix A.

Utilizing the MSEA well log data, which consists of multi-port well identification numbers and corresponding spatial $x$, $y$, $z$ locations, cross-listed with the nitrate sampling data, which consists of well identification numbers, sampling dates $t$, and nitrate concentrations, a new file was created. This new file represented a comprehensive list of nitrate concentrations at varying spatial and temporal locations. A separate list containing water table measurements, which were sampled less frequently than nitrate
concentrations, were also incorporated into this file. Since the water table measurements were primarily depth measurements and did not provide nitrate concentrations, the corresponding spatial locations were assigned a 0.0 initial concentration. The resultant file consisted of 1,594 spatially and temporally reference point source samples of nitrate concentration collected from April, 1991 through September, 1992.

In general, geostatistical analysis requires that the random variable to be normally distributed [42]. While various methodologies have been proposed to consider random variables that deviate from normal distributions, these techniques may not be required in all cases [18]. It has been found that the deviation from normality can usually be corrected by an appropriate transformation of the variable [2]. In many instances, if the random function of contaminant densities exhibits a lognormal distribution, the transformation of groundwater related data using a log-transformation will achieve the desired results of approximating a normal distribution [6]. As shown by the histogram in Figure 15, the nitrate concentration, which would be represented by a random function, exhibit a lognormal distribution. The log-transformation of the nitrate concentration, as shown in Figure 16, approximates a normal distribution.

Another requirement for geostatistical analysis is the assessment of spatial, and in the case of this investigation temporal, continuity evident in the given data. If the sampled locations were all equally distributed in space and time, the distance \(^1\) between sampled values could be used to assess the spatial and temporal correlations.

\(^1\)As mentioned earlier, "distance" in this investigation is representative as the "distance" between both the spatial and temporal locations between two separate samples. Time is defined as a direct and measurable extension to space.
Figure 15: Histogram distribution of nitrate concentration (ppm). The histogram distribution approximates a lognormal distribution.
Figure 16: Histogram distribution of the log of nitrate concentration (ppm). The histogram pattern approximates a normal distribution.
tigation, however, is the irregular spatial distribution of point source observations. This is irregularity is evident in the horizontal-spatial distribution as shown in Figure 17 compared to the vertical distribution of sample locations shown in Figure 18. In addition, temporal sampling was conducted in periodic, pseudo-random, intervals. This irregular frequency in sampling periods can be seen in Figure 19. Due to this irregular nature of the sampling distributions, geostatistical analysis was accomplished in each of the primary dimensions: horizontal-spatial, vertical, and temporal. By analyzing each of the sampling dimensions individually, the least biased assessment of continuity associated with the random variable along a given dimension may be obtained. A variogram computed for horizontal-spatially related data would be similar to traditional two-dimensional geostatistical analysis and, in this investigation, address the horizontal variability of nitrate. A second variogram, however, is calculated by considering the vertical variability of nitrate. Additionally, a third temporal variogram is calculated expressing the variability of nitrates at a fixed location in space over time.

As evident in Figure 17, Figure 18 and Figure 19, the sampled distribution represented in each dimension differs greatly. There are only a sparse horizontal-spatial observations relative to much greater numbers in the vertical and temporal dimensions. This irregular distribution and frequency of sampling locations in each of the sampling dimensions is extremely problematic for assessing the spatial and temporal continuity evident in the sampled data. For this reason, this investigation has incorporated various mechanism to help define the spatial and temporal correlations
Figure 17: Horizontal/Spatial distribution of both local and regional multi-port well locations.
Figure 18: Vertical distribution of both local and regional multi-port well locations.
Figure 19: Temporal distribution of nitrate sampling for both local and regional multi-port wells.
evident in irregularly spaced, multi-dimensional data.

3.1.1 Multi-Dimensional Filter Windows

Statistically bias data generally results from irregularly spaced sampled data which is generally found in most earth science data sets and is evident in this investigation. In this particular investigation, multi-port wells are used for collecting nitrate concentrations. As mentioned earlier, the sampling distances in the vertical dimension are much closer than the sampling distances in the horizontal-spatial. In this particular example, without the ability to filter out statistically biased sampled data, and consequently rely solely on Euclidean proximity for all sampled locations, the variogram analysis would indicate an erroneous, more continuous, spatial correlation in the vertical direction than is actually present. In addition, as indicated in Appendix A, Figure 35, the spatial location of the regional and local wells is generally clustered around the North-East section of the study site corresponding to the agricultural plots. From a geostatistical standpoint, the “spurious” regional wells located outside the clustering of local wells, corresponding to the agricultural plots, have adverse effects on the analysis of spatial continuity in the horizontal plane due to their outlying distances.

Multi-dimensional filter windows are implemented in this investigation to filter spurious or statistically biased data. These filter windows not only allow for the masking of spurious data outside a particular region, but provide a mechanism for masking the influence of nearby data located in a dimension that is not currently being analyzed. As shown in Figure 20 the dimensional filter windows correspond-
Figure 20: Multi-dimensional filter windows used to mask influences of nearby data located in a dimension that is not currently being analyzed. (A) Filter windows corresponding to the Horizontal-Spatial dimension are specified by a particular point $X_c, Y_c$ and radius $r$. (B) Filter windows corresponding to the vertical dimension are specified by a particular depth $V$ and a depth tolerance $V_t$. (C) Filter window corresponding to the temporal dimension are specified by a particular time $T$ and temporal tolerance $T_t$. 
ing to the horizontal-spatial, vertical, and temporal dimensions are shown in their graphical form. Multi-dimensional filter windows are used to mask influences outside the dimension being analyzed by filtering data located outside a temporal cylinder defined by the point \((X_c, Y_c)\) with radius \(r\) at a depth of \((V \pm V_t)\) from a start time \((T_{s-t})\) through end time \((T_{e-t})\). For example, horizontal-spatial variogram calculations for a given multi-port sampling network would best be assessed by geostatistically analyzing all data associated at the same port level at a fixed time. In this case, a circle \((X_c, Y_c)\) with radius \(r\) would be used to group sampling locations which would ideally best represent the horizontal-spatial distribution. The distribution in the vertical dimension would be masked by filter out all data outside the specified port level \((V)\) with a small tolerance \((V_t)\) to compensate for the gradient at the site. And finally, since nitrate concentrations change with respect to time, the temporal dimension would need filtered as well. This is provided by filtering all data outside of a particular sampling time \((T)\) with a small corresponding tolerance \((T_t)\) to allow for fact that all samples were not collected on exactly the same day. Similar to the horizontal-spatial analysis, the vertical and temporal dimensions would need to be assessed and filtered as well. In the case of the vertical dimension, a single well would generally be chosen to represent the vertical distribution of nitrate. This well would be selected by specifying a much smaller circle with a center \((X_c, Y_c)\) and a depth \((V_{s-t} \text{ to } V_{e-t})\) comparable to the spatial location and depth of the multi-port well. Once again, nitrate concentrations change over time and would need filtered by specifying a sampling day and tolerance. In the case of the temporal variogram
analysis, the assessment of a single point (specified by the cylinder defined by a small circle and similar depth) might be used to represent the variability of nitrate over a large degree of time. Ideally, the change of nitrate over time at this single point would best represent the temporal continuity of the phenomenon. The combinations of horizontal-spatial, vertical and temporal filter windows coupled with their appropriate tolerances, allow for an extremely flexible mechanism for masking statistically biased data located in both space and time.

3.1.2 Interactive Variogram Modeling

The analysis of spatial and temporal continuity evident in a data set is often a frustrating and difficult processes. Multi-dimensional data, sparse data samplings, statistically biased sampled data and irregular or anisotropically located sampling locations are all problematic factors when trying to analyze the statistical correlations between sampled values. As mentioned earlier, the many combinations and permutations of distance lag spacing, lag tolerances, angular directions and directional tolerances are additional complications when analyzing dimensional variograms. To facilitate the analysis of the spatial and temporal continuity evident in the sampled data, this investigation has developed several tools for iteratively visualizing and analyzing a multi-dimensional data set. This iterative and interactive process helps in the analysis of spatial and temporal continuity by providing mechanisms for analyzing the data and quickly visualize the results.

The Variogram Mapping Analysis Class (VMAC) of software routines incorporates techniques such as multi-dimensional filters to handle irregularly spaced data,
Figure 21: X-Windows GUI interface incorporating Variogram Modeling Analysis Class (VMAC) of subroutines to quickly visualize: multi-dimensional data filtering, lag and distance spacing and tolerances, range, sill and nugget modification, and theoretical function overlay.
and configurable lag spacing, tolerances, directional angles and angular tolerances, to facilitate directional variogram calculations. These routines are incorporated into a graphical user interface (GUI), Figure 21, to iteratively and graphically analyze the spatial and temporal continuity evident in the data. By selecting various "point-and-click" parameters, including multi-dimensional filters and corresponding filter tolerances, the ability to visualize filtered data used is available. By visualizing the data, and analyzing the distributions in the different dimensions, spurious data can be iteratively isolated and filtered. By iteratively selecting different variogram directional angles, directional tolerances, lag spacing, and lag tolerances, the user can quickly visualize both omnidirectional and direction variogram plots associated with the filtered data. Additional functionality includes the ability to overlay theoretical variogram functions with user specified range, sill, and nugget parameters to provide a "by-eye", "best-fit" relationship to the experimental variogram plots. The ability to quickly visualize and iterate over a large range of possible geostatistical variogram analysis parameters, allows for a relatively quick and thorough analysis of the spatial and temporal continuity that exists in the data set.

The VMAC class of subroutines are written in C and C++ and compiled for the UNIX environment on a Sparc10 workstation. The graphical user interface incorporating the VMAC class of subroutines was also developed in C and C++ and uses X-Windows and the OPENLOOK™ widget toolkit.
3.1.3 Horizontal-Spatial Variogram

The horizontal-spatial variogram represents the spatial variability of the random variable in the horizontal dimension. Directional variograms in this dimension represent the anisotropic characteristics intrinsic to the horizontally referenced data.

As shown in Figure 17, the horizontal-spatial distribution is relatively sparse compared to the distributions in the other dimensions. As mentioned earlier, obtaining an experimental variogram is extremely difficult if there are not enough samples available in the dimension analyzed. In addition, the experimental variogram is susceptible to statistically biased and/or spurious data. The analysis of spatial continuity is rarely a straightforward process. The ability to filter out spurious or statistically biased data and still provide enough sampled data to reflect the spatial continuity evident for the particular dimension analyzed is generally an iterative, and often times frustrating, process. By utilizing graphical and interactive geostatistical tools, such as VMAC interface, the ability to iteratively filter multiple dimensions and analyze the results can help determine the continuity evident in the sampled data.

As shown in Figure 22, the horizontal-spatial data used in the variogram calculation is restricted in each of the primary dimensions in order to best assess the horizontal-spatial continuity without the effects of additional, multi-dimensional data that may bias the experimental variogram. As mentioned earlier, ideally the horizontal-spatial variogram would be assessed by geostatistically analyzing all data associated at the same port level at a fixed time. This assessment, however, provided only a few sampling locations and consequently required a slight increase in window tolerances.
Figure 22: The results of multi-dimensional filter windows for the determination of the horizontal-spatial variogram. Figure (a) represents the horizontal-spatial constraint: all data located in the circle as defined by the center point (5, 5) with a radius of 25 is used. Figure (b) represents the vertical constraints: all data located at a depth of 265 ± 2 feet is used. Figure (c) represents the temporal constraints: all data collected at day 100 ± 5 is used. These constraints are used to filter out multi-dimensional data that may affect the horizontal-spatial variogram calculations.
to obtain additional information. Figure 22(a) shows the data filter in the horizontal plane as defined by the center point (5, 5) with a radius of 25. This filter is used to isolate the cluster of wells that ideally best represent the horizontal-spatial continuity. Figure 22(b) represents the data filter in the vertical plane as defined by a vertical depth of 265 feet with a vertical tolerance of 2 feet. Figure 22(c) represents the data filter associated with the temporal dimension as defined by only including the samples taken on day 100 with a temporal tolerance of 10 days. By restricting and filtering data in each of these dimensions, statistically biased and spurious data, which may lead to erratic variogram behavior, is filtered out. This results in the ability to collect data that ideally best represents the continuity in the horizontal-spatial dimension.

![Variogram Graph](image)

Figure 23: Horizontal-spatial variogram mist and corresponding theoretical variogram function.
Omnidirectional variogram analysis was conducted to ascertain non-directional, horizontal-spatial continuity of nitrate. Once the omnidirectional variogram was determined, directional variogram analysis was attempted. Due to the limited number of available sampled locations in the horizontal-spatial dimension, however, directional variogram analysis was unclear. To compensate for the lack of horizontal-spatial data, and to gain a clearer and less-biased understanding of the spatial structure of the variogram, a series of four different lag distances and tolerances were used to create four separate omnidirectional variograms. The overlay of these four separate variogram calculations represent an omnidirectional variogram "mist" and can be seen in Figure 23. An exponential theoretical variogram function was mapped onto the horizontal-spatial experimental variogram and can also be seen in Figure 23.

3.1.4 Vertical Variogram

The vertical variogram represents the spatial variability of the random variable in the vertical dimension. Directional variograms in this dimension represent the anisotropic characteristics intrinsic to the vertically referenced data.

As shown in Figure 18, the vertical distribution is more dense than that of the horizontal-spatial plane. Similar to the horizontal-spatial variogram analysis, the ability to filter out statistically biased data is required in order to ascertain the vertical continuity of nitrate. Ideally, a single multi-port well for a fixed time would be utilized to characterize the vertical variability of nitrate concentration. In practice, however, not enough data is available to accurately ascertain the vertical correlation. Filter windows would be expanded slightly to incorporate more data values in other
dimensions.

Figure 24: The results of multi-dimensional filter windows for the determination of the vertical variogram. Figure (a) represents the horizontal-spatial constraint: all data located in the circle as defined by the center point \((0, 0)\) with a radius of 7 is used. Figure (b) represents the vertical constraints: all data located at a depth of \(265 \pm 15\) feet is used. Figure (c) represents the temporal constraints: all data collected at day \(100 \pm 5\) is used. These constraints are used to filter out multi-dimensional data that may affect the vertical variogram calculations.

As shown in Figure 24, the vertical data used in the variogram calculation is restricted in each of the primary dimensions. Figure 24(a) shows the data filter in the horizontal plane as defined a center point \((0, 0)\) with a radius of 7 thus restricting analysis to only a few wells. Figure 24(b) represents the data filter in the vertical plane as defined by a vertical depth of 265 feet with a vertical tolerance of 15 feet. This filter is used to capture the most dense sampling locations in this dimension in hopes that the correlational insight obtained from this data would be representative for the vertical domain. Figure 24(c) represents the data filter associated with the temporal dimension as defined by only including the samples taken on day 100 with
a temporal tolerance of 5 days. This restriction is used to filter out the variability of nitrate that may change over time. By restricting and filtering data in each of these dimensions, statistically biased data is minimized. This results in the ability to utilize the most representative data to evaluate the vertical correlations of nitrate.

Figure 25: Vertical variogram mist and corresponding theoretical variogram function.

To gain a clearer and less-biased understanding of the spatial structure of the variogram, a series of four different lag distances and lag tolerances were used to create four separate variograms for various directions. Similar to the horizontal-spatial variogram analysis, directional analysis was inconclusive. Omnidirectional variogram analysis was conducted to ascertain non-directional spatial continuity in the vertical dimension. The omnidirectional variogram "mist" for the vertical dimension was represented by four separate omnidirectional variogram calculations with corresponding
lag and tolerances. The vertical variogram "mist" and the corresponding theoretical variogram function is illustrated in Figure 25.

3.1.5 Temporal Variogram

The temporal variogram represents the variability of the random variable over time. In other words, how continuous the random function is, and how close together sampling frequency is needed to predict unsampled locations at unsampled times.

![Temporal Variogram Diagrams](image)

Figure 26: The results of multi-dimensional filter windows for the determination of the temporal variogram. Figure (a) represents the horizontal-spatial constraint: all data located in the circle as defined by the center point (0,0) with a radius of 7 is used. Figure (b) represents the vertical constraints: all data located at a depth of 265 ± 2 feet is used. Figure (c) represents the temporal constraints: all data collected at day 60 ± 60 is used. These constraints are used to filter out multi-dimensional data that may affect the temporal variogram calculations.

Ideally, temporal variogram analysis would consist of a examining a single point window for a given temporal range. The result would be an experimental temporal variogram for exactly one point, thereby indicating the continuity of the random variable over time. As shown in Figure 18, the temporal distribution is more dense
than that of the horizontal-spatial or vertical planes. The variogram analysis from a single point over time, however, was extremely erratic and inconclusive. As a consequence, it was necessary to expand the spatial window so that enough data couples were obtained. By expanding the spatial window, the temporal variability within a specified volume was assessed and used to represent the temporal continuity of the random variable. This volume was comprised of horizontal-spatial and vertical filter windows. All data located in the volume defined by a center point of (0, 0) with a radius of 7 (Figure 26(a)) from 264 to 266 feet in depth (Figure 26(b)). In order to assess the temporal continuity of this volume, variogram analysis was for various temporal lags and tolerances between days 1 through 120 (Figure 26(c)).

Figure 27: Temporal variogram mist and corresponding theoretical variogram function.
A series of four different temporal lags and tolerances were used to characterize the structure of the temporal variogram. Omnidirectional variogram analysis was conducted to ascertain temporal continuity. No directional analysis in the temporal domain was attempted. The omnidirectional variogram "mist" for the temporal dimension was represented by four separate omnidirectional variogram calculations with corresponding lag and tolerances. The temporal "mist" and the corresponding theoretical variogram function is illustrated in Figure 27.

### 3.2 Four-Dimensional Kriging

As mentioned earlier, the kriging method of interpolation, which is based on the theory of regionalized variables while using the degree of autocorrelation between adjacent samples, estimates values for any coordinate position within the domain measured without bias and with minimum variance. A probabilistic model is used, in which the bias and the error variance can both be calculated. From there, weights for the nearby samples can be chosen to ensure that the average for the model is exactly 0 and that the modeled error variance minimized. The set of weights that will produce unbiased estimates with the minimum estimation variance is directly dependent on the variogram model associated with the data. By utilizing geostatistical tools, such as VMAC interface, the spatial and temporal variogram model associated with the data can be quickly determined and entered into the system of kriging equations.

The Four Dimensional Kriging (4DK) class of subroutines was developed to estimate values at unsampled spatial and temporal locations based on regionalized variable theory. The 4DK class of subroutines utilizes a specified structured input
file, defined as the DAT file (Appendix F) as a mechanism for defining a variety of parameters into the system of kriging equations. The 4DK class of subroutines estimates values and variances for a point, plane, or volume for any temporal range within the time domain of the given data set based on several DAT specified parameters.

The 4DK class of subroutines was written in C and C++, and compiled under UNIX on a Sparc10 workstation.

3.2.1 Incorporation of the Variogram Model

The estimation at unsampled locations first requires a model of how the phenomenon behaves; without a model, one has only the sampled data and no inferences can be made about the unknown values. By utilizing geostatistical analysis tools, such as the interactive VMAP class interface, the models of spatial and temporal continuity which are represented by the theoretical variogram functions, can be obtained. Each dimension analyzed has a corresponding theoretical variogram function. The 4DK class of subroutines support the major theoretical function including spherical, Gaussian, and exponential. The type of theoretical variogram function and the corresponding range, sill and nugget for each dimension is configurable in the DAT file.

3.2.2 Incorporation of the Anisotropic Model

Qualitative information, such as the orientation of lithologic units or bedding planes, may be helpful in identifying anisotropic characteristics evident at the site. Qualitative information known a priori coupled with geostatistical analysis of the sampled
data provides extremely useful information for the estimation of unsampled locations.

Incorporation of Anisotropic Distance

![Graph showing variogram functions](image)

Figure 28: Illustration of multi-dimensional distance and direction variogram functions plotted on the same graph. The variogram functions corresponding to the ranges $AX$ and $AY$ represent the anisotropic horizontal and vertical vectors in the $X$ and $Y$ directions accordingly. The variogram function corresponding to the range $AZ$ represents the anisotropic distance vector in the vertical dimension, and the variogram function corresponding to the range $AT$ represents the temporal anisotropic vector. In this case geometric anisotropy is assumed as evident in similar sills and differing ranges in the corresponding dimensions.

The variogram function for each dimension, and corresponding sill, range and nugget values, represent the anisotropic axis. As shown in Figure 28, hypothetical directional and distance variogram functions are plotted on the same graph. The range of each of these variogram functions reflect the anisotropic characteristics of the sampled data and correspondingly indicate the degree of preferential flow in the sampled directions. Figure 29 is graphical representation of hypothetical anisotropic distance vectors corresponding to Cartesian space. Each anisotropic vector corresponds to its
Figure 29: Illustration of Anisotropic distance vectors. The vectors ($AX$, $AY$, $AZ$, $AT$) represent preferential flow in the corresponding $X$, $Y$, $Z$, and temporal directions.

particular range in the associative variogram function shown in Figure 28. In Figure 29, preferential flow is indicated by a relatively longer vector in the $X$ direction, with lesser flow tendencies in the $Y$ and $Z$ directions. The anisotropic temporal component $T$ represents a relatively continuous process compared to the corresponding spatial axes.

For simplicity, if anisotropic direction is assumed not present, the distances vectors of anisotropy corresponds the Cartesian coordinate system and can be seen in Figure 29. In order to incorporate anisotropic distances in the estimation of unsampled locations, the variogram model which represents the anisotropic distances must somehow be transformed into a model used in the data coordinate system. The method used in this investigation is to define a transformation that reduces all directional
variograms to a common model with a normalized range of 1. Each separation distance, therefore, needs to be transformed so that the standardized model will provide a variogram value that is identical to any directional models for the pre-transformed separation distance. Any directional model along a particular dimension with a range of \( a_d \) can be reduced to a standardized model with a range of 1 simply by replacing the separation distance of the corresponding dimension, \( h_d \), by a reduced distance \( \frac{a_d}{h_d} \). Therefore, the anisotropic variogram model in four dimensions, with corresponding ranges \( a_x, a_y, a_z \) and \( a_t \) can be expressed as

\[
\gamma(h) = \gamma_1(h_n) \tag{3.1}
\]

where \( h_n \) is defined as

\[
h_n = \sqrt{\left(\frac{h_x}{a_x}\right)^2 + \left(\frac{h_y}{a_y}\right)^2 + \left(\frac{h_z}{a_z}\right)^2 + \left(\frac{h_t}{a_t}\right)^2} \tag{3.2}
\]

In matrix notation, the reduced distances can be summarized by the vector \( h_n \) that is defined by

\[
h_n = Th \tag{3.3}
\]

where the matrix \( T \) is defined by

\[
T = \begin{bmatrix}
\frac{1}{a_x} & 0 & 0 & 0 \\
0 & \frac{1}{a_y} & 0 & 0 \\
0 & 0 & \frac{1}{a_z} & 0 \\
0 & 0 & 0 & \frac{1}{a_t}
\end{bmatrix} \tag{3.4}
\]

and \( a_x, a_y, a_z \) and \( a_t \) are the ranges of the anisotropic distance models along the coordinate axes \( x, y \) and \( z \).
Incorporation of Anisotropic Direction

The assumption that axes of anisotropy correspond to the coordinate axis is, for the most part, incorrect. Where there may be preferential flow along the horizontal-spatial direction rather than in the vertical direction, these directions are representative of the data and often do not correspond to the traditional Cartesian $(X, Y, Z)$ coordinate axis. By analysing the directions of preferential flow, however, an approximation, and consequent mapping of these directions to Cartesian space is possible. Utilizing directional variogram analysis, coupled with a priori, site specific characteristics, anisotropic directions can be determined and incorporated into the kriging estimation process.

![Diagram](image)

Figure 30: The transformation of anisotropic distance vectors, $(AX, AY, AZ)$ in (a) can be rotated around a three-dimensional coordinate system as defined by two angles of rotation. The first rotational angle, $\phi$, is defined as the clockwise rotation about the $z$ axis forming the new anisotropic axes $(AX'$, $AY'$, $AZ'$) and shown in (b). The second rotational angle, $\theta$, is defined as the clockwise rotation about the new $AY'$ axis forming the new anisotropic axes $(AX''$, $AY''$, $AZ''$) and is shown in (c).

The transformation of spatial anisotropic distance vectors can be rotated around
a three-dimensional coordinate system as a function on two rotational angles. Given
the set of anisotropic vectors \((AX, AY, AZ)\), as shown in Figure 30(a), the rotational
transformation matrix \(R\) can be defined by two angles of rotation which correspond
to basic trigonometric operation in Cartesian space. The first angle of rotation, \(\phi\),
is defined as the clockwise \(^2\) rotation around the \(Z\) axis resulting in the new vectors
\((AX', AY', AZ')\) and shown in Figure 30(b). The second angle of rotation, \(\theta\), is shown
in Figure 30(c) and is defined as the clockwise rotation around the new \(AY'\) vector
forming the new set of vectors \((AX'', AY'', AZ'')\). The transformation of a three-
dimensional coordinate system can be defined by two angles of rotation and can be
defined in matrix notation as:

\[
R = \begin{bmatrix}
\cos(\phi)\cos(\theta) & \sin(\phi)\cos(\theta) & \sin(\theta) & 0 \\
-sin(\phi) & \cos(\phi) & 0 & 0 \\
cos(\phi)\sin(\theta) & -\sin(\phi)\sin(\theta) & \cos(\theta) & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]  

(3.5)

In other words, given a data coordinate system defined by the \((X, Y, Z)\) axes and
an anisotropic coordinate system defined by the \((AX'', AY'', AZ'')\) axes, the trans-
formation matrix \(R\) as defined by Equation 3.5 will transform any vector \(h\) in the
data coordinate system to \(h''\) defined in the anisotropic data system. The anisotropic
variogram model can then be correctly evaluated using the vector \(h''\).

### 3.2.3 Spatial and Temporal Estimation and Variance of Nitrate Concentrations

In order to begin estimating unsampled values in space and time for a particular
random function (in this case, nitrate concentration) several sets of parameters need

\(^2\)The clockwise direction around an axis is defined as the rotation around the axis as defined by
the positive direction of the given coordinate system.
to be specified. These parameters include the location of the file containing the $n$ sampled observations, spatial and temporal variogram functions for each primary dimension, anisotropic characteristics associated with the site and the spatial and temporal regions that require estimations. These parameters are stored in the DAT file (Appendix F) and are used as input to create the set of kriging equations.

The theoretical variograms used to describe the spatial and temporal continuity of the random variable (Figure 23, Figure 25 and Figure 27) are somewhat inconclusive in nature. Due to the relatively erratic behavior and high nugget values associated with each of the primary omni-dimensional variograms and the inconclusive nature of the directional variograms, additional site specific hydrogeological assumptions were incorporated into the set of kriging equations. These assumptions include the incorporation of preferential flow in the direction of $S15^\circ W$ toward the Scioto River (see Appendix A, Figure 35) with a ratio of 2:1 in the horizontal-spatial plane. The ratio of 2:0.5 in the horizontal-spatial to vertical plane as evident in the ranges of the appropriate dimensional variograms was assumed correct. In addition, a downward dip of $5^\circ$ was assumed due to the slight gradient of the Scioto Valley. The variogram parameters used for the estimation of nitrate concentrations at the MSEA site can be seen in Table 2.

Utilizing the $n$ sampled values, and corresponding theoretical variogram function and anisotropic characteristics in each dimension, the $(n + 1)^2$ covariance matrix $C$ used in Ordinary Kriging System (Equation 2.11) was generated. The corresponding lag vector $h$ used for each covariance cell $\tilde{C}_{ij}$ calculation, as mentioned before, is based
Table 2: Theoretical variogram parameters for each primary dimension.

<table>
<thead>
<tr>
<th>Plane</th>
<th>Function</th>
<th>$C_0$</th>
<th>$C_1$</th>
<th>$a$</th>
<th>Rotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Horizontal-Spatial ($S15^\circ W$)</td>
<td>Exponential</td>
<td>0.17</td>
<td>0.06</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>Horizontal-Spatial ($N105^\circ W$)</td>
<td>Exponential</td>
<td>0.17</td>
<td>0.06</td>
<td>10</td>
<td>$\phi = 15^\circ$</td>
</tr>
<tr>
<td>Vertical</td>
<td>Exponential</td>
<td>0.17</td>
<td>0.06</td>
<td>4</td>
<td>$\theta = 5^\circ$</td>
</tr>
<tr>
<td>Temporal</td>
<td>Exponential</td>
<td>0.13</td>
<td>0.21</td>
<td>50</td>
<td></td>
</tr>
</tbody>
</table>

on the statistical distance between the sampled values $v_i$ and $v_j$ rather than Euclidean. The statistical distance for any lag vector can be defined as the transformation that incorporates both anisotropic distance and direction. By utilizing Equation 3.3 and Equation 3.4, which define the anisotropic distance transformation, and Equation 3.5, which define the anisotropic directional transformation, the normalized lag vector $h_n$ corresponding to $v_i$ and $v_j$ can be written as:

$$h_n = TRh$$ (3.6)

Therefore, for each $\tilde{C}_{ij}$ calculation, the magnitude of the corresponding $h_n$ vector is used as input to the corresponding theoretical covariance function.

Once the covariance matrix $C$ is built, $C^{-1}$, the inverse of $C$, is determined utilizing a variation of Gaussian elimination with threshold pivoting [23]. $C^{-1}$ is used to solve for each weight vector $\lambda$ needed to calculate the estimation and variance associated with a particular unsampled location in space and time (Equation 2.14).

Once $C^{-1}$ is determined, for every unsampled location in space and time a corresponding $(n + 1)$ distance vector $D$ is created between the unsampled location and the $n$ available samples as defined in Equation 2.11. Once again, each entry in the
The $D$ vector is based on the statistical distance between sampled values and can be calculated using Equation 3.6. Utilizing the inverse covariance $C^{-1}$ matrix and the distance $D$ vector, the set of weights that will produce unbiased estimates with the minimum error variances for the random function model is defined in Equation 2.15 and reiterated as:

$$\lambda = C^{-1} \cdot D$$  \hspace{1cm} (3.7)

Utilizing this set of weights, the resultant estimate for any given spatial and temporal location defined in Equation 2.16 and reiterated as:

$$v_0 = \sum_{i=1}^{n} \lambda_i v_i$$  \hspace{1cm} (3.8)

where $n$ is the number of measured values $v_i$ and $\lambda_i$ are the corresponding weights attached to each measured value. And the minimized estimation variance at the same location is defined in Equation 2.17 and reiterated as

$$\tilde{\sigma}_R^2 = \tilde{\sigma}^2 - \sum_{i=1}^{n} \lambda_i \tilde{C}_{i0} + \mu$$  \hspace{1cm} (3.9)

The application of four-dimensional kriging to an available data set provides the ability to generate volumetric realizations of nitrate concentration for any given time period. By incrementally increasing the temporal period for the estimation of unsampled spatial locations, the ability to create dynamic progressions of nitrate concentration are provided. The ability to estimate unsampled values in a contiguous spatial locations and over a continuous time frame, produces a vast amount of data. As mentioned earlier, the visualization of large amounts of data associated with earth science data sets is an important component to understanding, analyzing or explaining the phe-
nomenon. The *type* of visualization technique used is also a crucial consideration. Traditional methods of creating two-dimensional contours slices at different depths, or three-dimensional iso-surface representations, are simplistic representations of a much more complex problem. In addition, natural phenomenon, such as non-point source subsurface contamination, are amorphic in nature. As mentioned earlier, non-point source subsurface contamination are relatively diffuse in nature with extremely variable concentrations. The representation of this type of phenomenon in terms of surfaces and edges not only understates the true complexity of the phenomenon, but may in fact, complicate matters by understating the variability of the phenomenon as well.
CHAPTER IV

Volumetric Visualization

A rising awareness among scientists of the complexities of natural phenomenon, such as non-point source groundwater pollution, is providing an impetus to understand the spatial and temporal structure associated with these processes. It is difficult, however, for the human brain to make sense of the large volume of spatial and temporal data associated with such natural phenomena. As the volume of data provided by enhanced observational techniques increases, and as additional data is generated by computational operations applied to existing observations, the role of visualizing and understanding these data sets is crucial. Visualization is an important component of any effort to understand natural phenomenon [5]. In addition, visualization is an inherently iterative task, in which the researcher hopes that successive iterations will provide a better understanding of the structure of the data [33].

The understanding and visualization of natural phenomenon, however, poses a formidable obstacle for traditional graphical techniques. Natural phenomena, is often voluminous, not consisting of surfaces and edges, but densities or concentrations. Surface and subsurface contamination, air pollution, ozone depletion, hurricane and weather tracking, for example, have no tangible “surfaces” or well defined boundaries. Such boundaries are a necessity for conventional graphical analysis techniques.
In addition, natural phenomena are both three and four-dimensional, consisting of volumetric information often continuously changing over time. These obstacles, coupled with the need to make sense from spatial and temporally referenced point source data, demands mechanisms which perform iterative and dynamic visualization of volumetric data.

*Volumetric visualization*, also called volume visualization and three-dimensional volume graphics, is a method of producing two-dimensional screen images from three and four-dimensional volumetric data [46]. Unlike traditional techniques, which represent three-dimensional objects in terms of surfaces and edges approximated by polygons and lines, volume data are three-dimensional entities that have information inside of them. This data may not consist of surfaces and edges at all, or may be too voluminous to be represented geometrically [46]. Volume visualization is concerned with the representation, manipulation, navigation and rendering of volume data. The objective of volume visualization is to peer inside volumetric objects, to provide a means of viewing that which is not ordinarily viewable, and to probe into voluminous and complex dynamic structures in order to comprehend spatial and temporal trends and patterns [46].

### 4.1 Background

Volume visualization is a rapidly growing field in computer graphics and imaging [46]. The field of volume graphics can be traced back to the mid-70's, where the use of volumetric data, particularly in three-dimensional medical imaging, has been reported [46]. Three-dimensional medical imaging (e.g. computed tomography, mag-
netic resonance imaging, positron emission tomography, and ultrasonography) have been the primary application of volume graphics and a driving force for the development in this field [3, 4, 10, 13]. With the recent advances in computer hardware, however, including faster CPU's, and larger and cheaper memory, coupled with a substantial drop in the cost of hardware, volume graphics is readily being utilized in various fields of research in which acquired data is already in volumetric form. These fields include: biology (e.g., confocal microscopy [22, 24]), geoscience (e.g., seismic measurements [25, 35, 44]), meteorology [15, 16, 39] and molecular systems (e.g. electron density maps [14]). In addition to fields in which the acquired data is already in volumetric form, volume visualization offers an extremely effective method for interacting with model-based volumetric applications. as well. Some of these applications include: computer aided design, (e.g., solid modeling, finite element analysis, material stress pattern analysis), simulation and animation (e.g., flight simulation), and scientific visualization (e.g., astrophysical simulation, fluid flow dynamics [36, 45, 39]).

Volume visualization has recently emerged as a key research and development area of computer graphics [46]. As the progress in volume visualization and computer development increases, coupled with the desire to reveal inner structures and patterns in volumetric and dynamic data sets, volume graphics will play a major role in the field of computer graphics and a necessary role in the associated applications.

4.2 Volume Representation

Typically, the volumetric data set is represented as a three-dimensional discrete grid of volume elements called voxels and commonly stored in a volume buffer, which
is a large three-dimensional array of voxels. Alternatively, other data structures and formats have been utilized for storing and manipulating volumetric data sets including: octrees, sparse volume matrices, semiboundaries, voxel runs and irregular grids [21].

![Figure 31: A volumetric torus.](image)

A voxel is the cubic unit of volume centered at an integral grid point [21]. As a unit of volume, the voxel is the three-dimensional counterpart to the unit of area traditionally defined by the two-dimensional pixel. Each voxel generally has one or more associated values which represent some measurable properties. These properties may include, but are not limited to, color, opacity, density, material, velocity, etc.. Figure 31, is an example of a basic geometric object represented in volumetric form. The torus has been decomposed into identical cells arranged in a fixed, regular grid of voxels.
Table 3: Comparison between surface and volume graphics.

<table>
<thead>
<tr>
<th>Compatibility</th>
<th>Surface Graphics</th>
<th>Volume Graphics</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Rendering Performance</td>
<td>Sensitive to scene and object complexity</td>
<td>Same, regardless to scene and/or object complexity</td>
</tr>
<tr>
<td>2 Memory and Processing requirement</td>
<td>Variable, depending in scene and object complexity</td>
<td>Large but constant</td>
</tr>
<tr>
<td>3 Object-space aliasing</td>
<td>None</td>
<td>Frequent</td>
</tr>
<tr>
<td>4 Transformation</td>
<td>Continuous; performed on the geometric definition of objects</td>
<td>Discrete; performed on subvolumes</td>
</tr>
<tr>
<td>5 Scan conversion and rendering</td>
<td>Pixelization is embedded in viewing</td>
<td>Voxelization is decoupled from viewing</td>
</tr>
<tr>
<td>6 Boolean and block operations</td>
<td>Difficult; must be performed analytically</td>
<td>Trivial; by using voxel, voxel-by-voxel operations, aggregation, octrees</td>
</tr>
<tr>
<td>7 Rendering of interior</td>
<td>No; surfaces only</td>
<td>Yes; rendering of inner structures as well as surfaces</td>
</tr>
<tr>
<td>8 Adequacy for sampled data and intermixing with geometric data</td>
<td>Partially and indirectly (fitting followed by surface rendering</td>
<td>Supports representation and direct rendering</td>
</tr>
<tr>
<td>9 Measurements (for example, distance, area, volume, normal)</td>
<td>Analytical, but may be complex</td>
<td>Discrete approximation, but simple</td>
</tr>
<tr>
<td>10 Viewpoint dependency</td>
<td>Requires recalculation for every changed viewpoint</td>
<td>Precomputes and stores viewpoint independent information</td>
</tr>
</tbody>
</table>
4.3 Disadvantages of Volume Graphics

Table 3 compares and contrasts volume graphics and traditional surface graphics. The major disadvantages of volume graphics are due to its discrete form (rows 3 and 4 in Table 3), the loss of geometric information (row 9) and the memory and processing power it requires (row 2).

- **Discrete Form**: Unlike surface graphics, volume graphics represent three-dimensional data in discrete forms. Figure 31, for example, is a volumetric torus composed of volumetric discritized entities (voxels). Consequently, volume based graphics have some problems similar to two-dimensional raster graphics. The finite resolution of the raster poses a limit to the accuracy of operations that are based on voxel counting (row 9 in Table 3). For example, volume and area measurements are more difficult due to the discritized nature of the data as opposed to traditional surface graphics. In addition, it is difficult to manipulate and transform a discritized volume set without degrading the image quality or losing some information (row 4 in Table 3) [21].

- **Loss of Geometric Information**: In volume graphics, each voxel generally maintains only local information pertaining to the volume unit it represents [21]. Consequently, when exact measurements are required, such as distance, area or volume, it is advantageous to use a surface based model, where the geometric surface definition of the object is available. As mentioned before, the voxel-based object is only a discrete approximation to the continuous object.
Consequently the volume buffer resolution determines the precision of the measurement [21]. On the other hand, several measurement types are more easily computed in voxel space including mass properties, adjacency detection, and computation of volume (row 9 in Table 3) [21].

- **Memory and Processing:** A typical volume buffer generally occupies a large amount of memory. When considering a frame buffer for a two-dimensional 512 x 512 raster image, assuming one byte per pixel, approximately 256 Kbytes of memory is needed. When considering a volume buffer, however, for a 512 x 512 x 512 voxel image, 128 Mbytes will be required. This volume buffer consists of more than 1x10^8 voxels (row 2 in Table 3). Since computer memory prices are significantly decreasing, coupled with tremendous increases in memory performance, such large memory capacity is becoming more feasible. Concerns of large memory needs, similar to those concerns with volume graphics, surfaced in the mid-70’s when raster graphics emerged as a new technology [21]. With greater compactness accompanying the rapid decline in memory price, it is safe to predict that, just as with raster graphics, memory will soon cease to be a stumbling block for volume graphics [21].

### 4.4 Advantages of Volume Graphics

The advantages of volume graphics include its capacity to represent sampled and simulated volumetric data sets (row 8 in Table 3), its unique capability to represent inner information and amorphous information (row 7, Table 3) and its capability to
support various block operations (row 6, Table 3).

- **Sampled and Simulated Data Sets**: Sampled data sets, like those in three-dimensional medical imaging, and simulated data sets, like those in computational fluid dynamics, are often three-dimensional in nature. Three-dimensional data is simply reconstructed into a regular grid of voxels and stored in a volume buffer. Unlike surface graphics, volume graphics naturally supports the representation, manipulation, and rendering of such data sets (row 7 in Table 3) [21]. In addition, the volume buffer medium is useful for intermixing volumetric data sets with traditional geometric objects (row 8 in Table 3).

- **Representation of Inner Information**: One of the most appealing attributes of volume graphics is its unique ability to represent internal information. Unlike surface graphics, volume graphics can represent inner structures of objects which can be revealed and explored with manipulation and rendering techniques (row 7 in Table 3) [21]. Natural objects are more likely to be solid rather than hollow. The inner structure of these objects are a crucial aspect and are easily explored using volume graphics (row 7 in Table 3). In addition, while translucent objects can be represented by traditional surface methods, these methods cannot efficiently support the modeling and rendering of amorphous phenomenon that are volumetric and do not have tangible surfaces [21].

- **Block Operations**: An intrinsic characteristic of volume buffers is that adjacent objects in the scene are also represented by neighboring memory cells [21].
Therefore, volume buffers lend themselves to various meaningful grouping-based operations such as in two-dimensional framework blit (bit block transfer) operations, or their three-dimensional counterpart, voxblt (voxel block transfer) operations [20]. Such block operations support transfer of cuboidal voxel blocks with a variety of voxel-by-voxel operations between source and destination blocks (row 6 in Table 3) [21]. In addition, the volume buffer lends itself to Boolean operations that can also be performed on a voxel-by-voxel manner as in Constructive Solid Geometry (CSG) [11] applications concerned with subtraction, union, and intersection between two voxelized objects.

4.5 Volumetric Visualization and Geostatistical Kriging

As mentioned earlier, the visualization of natural phenomenon is a formidable task due to the general, voluminous nature of the phenomenon, the spatial and temporal dynamic processes that are involved and the sparse nature of observational sampling techniques. As also mentioned, volume visualization offers an extremely effective method for interacting with output from volumetric observations and model-based applications. Utilizing observational sampling techniques, geostatistical insights into the spatial and temporal continuity of the phenomenon may be gained. This spatial and temporal continuity, coupled with a priori, site specific, information can then be used as a mechanism for defining the random function with represents the natural phenomenon. From this representation, the set of multi-dimensional kriging equations can be generated which can be used to estimate values and variances at unsampled locations. Estimated values and variances can be stored directly into a volumetric voxel
data structure and assigned a corresponding color and opacity. The ability to animate over time, rotate, slice orthogonally through the volume, and strip away successive layers, are all possible by utilizing volumetric visualization techniques. Volumetric visualization techniques coupled with four-dimensional geostatistical kriging offers not only a new way of estimating values and variances for any given spatial and/or temporal location, but also a better way of analyzing and visualizing the results.
CHAPTER V

Results and Discussion

5.1 Application of Four-Dimensional Kriging

Application of the four-dimensional kriging algorithm to the available data set, after the transformation to log space, results in the ability to generate volumetric realizations of non-point source subsurface nitrate concentrations at any time period in the given domain. Due to the computational complexity, however, of the kriging algorithm, the estimation of unsampled spatial and temporal locations was done for only a subset of the available data. Spatial and temporal estimation was done for the north-eastern section of the MSEA study site corresponding to the site specific wells located around the agricultural plots as shown in Appendix A, Figure 35. The spatial and temporal distributions that were kriged can be seen in Figure 32. Temporal estimation was done for every other day between April 1, 1991 through day August 1, 1991, and can be seen in Figure 32(c). For every temporal increment, in this case 2 days, the spatial volume corresponding to −20 to 20 in the X direction, 20 to −10 in the Y direction (Figure 32(a)), at a depth of 280 to 240 in the Z direction (Figure 32(b)) was kriged. The kriged value of nitrate was stored directly into a voxel array data structure. The corresponding variance associated with the es-
timation was also stored in an additional voxel array data structure. Both the kriged nitrate concentration volume and the variance volume was comprised of 40x30x20 voxels with each voxel representing 100x100x1 feet.

![Image](image_url)

Figure 32: Spatial and temporal distribution of geostatistical kriged area. Estimated values and corresponding variances were done for the volume corresponding to −20 to 20 in the X direction, 20 to −10 in the Y direction (Figure(a)), at a depth of 280 to 240 in the Z direction (Figure(b)). Temporal estimation was done for day 1 (April 1, 1991) through day 121 (July 1, 1991). The temporal distribution can be seen in Figure(c).

5.2 Volume Visualization of Spatial and Temporal Estimations and Variances

The ability to animate through the volume of data was accomplished utilizing SunVision’s™ volume rendering module SunVoxel from Sun™ Microsystems. SunVoxel is a forward projection, interactive volume rendering tool that provides [29] :

- Wireframe display for interactive manipulation of volumes.

- Arbitrary image display sizing.
- Various shading methods including ray casting, texture mapping, and maximum value display.

- Interactive modification of opacity and color attributes.

- Interactive substance classification.

- Orthogonal and arbitrary oblique angle cuts through volumes.

- Voxel examination and modification in model space.

The ability to assign opacity values and color to the log transformed concentrations of nitrate, allowed for the visualization of the dynamic, amorphic plume. In addition, opacity values and colors were associated with the estimated variances in order to visualize the spatial and temporal errors associated with the kriging technique. The amorphic subsurface nitrate concentrations can be seen at various time in Appendix B. In addition, the corresponding dynamic variance volumes can be seen at similar times in Appendix C.

5.3 Assessment of Four-Dimensional Kriging: Cross-Validation of Spatial and Temporal Estimations

Cross-validation was used to verify the choice of theoretical variogram functions derived from the experimental variogram. The cross-validation process can be described as follows:

- Delete a measured sample from the available data set.

- Kriged at the deleted sample location using remaining data.
• Compute differences between kriged value and measured value.

• Repeat process for each of the remaining samples.

Statistical analysis of the differences between the kriged value and the measured value, defined as the kriging error, may provide insight related to the choice of the variogram model selected as well as the validity of the kriging process for the given data set.

Three different summary statistics were used to evaluate the cross-validation results. The average kriging error (AKE) provides a measure of the degree of bias introduced by the kriging process. The AKE is defined as [43]:

\[
AKE = \frac{1}{n} \sum_{i=1}^{n} [v_i^* - v_i]
\]  

(5.1)

where \(v_i\) is defined as the value at a specific location in space and time, \(v_i^*\) is the kriged value at the specified location, and \(n\) is the number of samples. If the average kriging error (AKE) is close to zero, the kriging process may be considered unbiased [43].

A second statistical measure used to evaluate the kriging process is the mean squared error (MSE) and is defined as [43]:

\[
MSE = \frac{1}{n} \sum_{i=1}^{n} [v_i^* - v_i]^2
\]

(5.2)

Assessment of the theoretical variogram models should be guided by the combination of models that minimize the MSE. In practice, a global minimum would be difficult to verify, hence the MSE should be less than the variance \(\sigma^2\) of the sampled values [43]. The variance of the sampled values is defined as the average squared difference of the sampled values from their mean, and is represented as:

\[
\sigma^2 = \frac{1}{n} \sum_{i=1}^{n} (v_i - m)^2
\]

(5.3)
Table 4: Cross-validation consistency criteria and results.

<table>
<thead>
<tr>
<th></th>
<th>$AKE$</th>
<th>$MSE$</th>
<th>$SMSE$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recommended Value</td>
<td>0.0</td>
<td>13.4995</td>
<td>$1.0 \pm 1.7982 \times 10^{-5}$</td>
</tr>
<tr>
<td>Cross-Validation Results</td>
<td>$-2.2718 \times 10^{-2}$</td>
<td>9.5813</td>
<td>1.6613</td>
</tr>
</tbody>
</table>

Consistency of the estimation errors and kriging variances can be examined by evaluating the variance of the standardized mean squared error ($SMSE$) defined as [43]:

$$SMSE = \frac{1}{n-1} \sum_{i=1}^{n} \frac{[v_i^* - v_i]^2}{\sigma^2_{\hat{R}_i}}$$

(5.4)

where $\sigma^2_{\hat{R}_i}$ is the variance at point $i$. The consistency criteria as satisfied if the variance of $SMSE$ is defined to be within the interval $1.0 \pm (2(\frac{2}{n})^2)$ [43]. Cross-validation consistency criteria along with the results of the cross-validation test are presented in Table 4.

The histogram of error frequency associated with the estimated values is shown in Figure 33. As seen from this figure, the variance associated with the estimations is greatly skewed, indicate a relatively large degree of error associated with the estimation. This corroborates the $SMSE$ calculations in Table 4. A scatter plot of the actual value versus the kriged value is shown in Figure 34. Ideally, the scatterplot of kriged estimates to actual values would produce a thin cloud along the $X = Y$ slope. As shown in Figure 34, however, the thickness of the scatterplot is relatively large with a definite discontinuity at the origin. Least squares method of fitting a line to a scatterplot is used to indicate a line with a slope approximately equal to 0.2 and a
Figure 33: Histogram of kriging variance.
Figure 34: Scatter plot of actual nitrate concentration verses estimates obtained from ordinary kriging.
y-intercept of about 3.2 and can also be seen in Figure 34.

The results of the average kriging error ($AKE$) and the mean squared error ($MSE$) as shown in Table 4, were relatively encouraging. These calculations indicated that the theoretical functions used to represent the variogram functions were relatively unbiased. The results of the standardized mean squared error ($SMSE$) as shown in Table 4 as well as the histogram distribution of the variance (Figure 33) and the scatterplot of kriged estimates to actual values (Figure 34), however, were somewhat discouraging. These indicators, in addition to the large nugget values in each of the experimental variogram structures calculated for each dimension, indicate a large uncertainty associated with the estimation of this phenomenon utilizing only the point samplings provided. These factors clearly indicate the need for additional sampled locations in both space and time. These indicators are additionally supported through the dynamic and volumetric visualization of the estimated variance as shown in Appendix C.

5.4 Future Work

5.4.1 Search Strategies

Additional work concerning spatial and temporal estimation using four-dimensional geostatistical kriging is needed. The four-dimensional kriging technique described in this investigation is a resource intensive and computationally complex algorithm. Not all sampled values are needed in space and time to estimate an unsampled location. Additional research is needed concerning searching strategies and searching neighborhoods in four dimensional space.
5.4.2 Matrix Solutions

Due to the relatively unique nature of the covariance matrix as defined in the set of ordinary kriging equations (Equation 2.11), additional research is needed concerning the solution to the system of kriging equations. As mentioned earlier, the use of theoretical variogram functions provide a positive definite matrix, with a single, unique solution. Due to roundoff error associated with very small entries in the covariance matrix, however, research is needed concerning the determination of pivots in order to solve the system of linear equations need researched. Several matrix packages including Meschach [38] were used to solve the system of kriging equations, however, due to the choice of pivots and roundoff error, produced erroneous results. As the result, the SPARSE [23] package, which uses a variation of Gaussian Elimination with threshold pivoting, was used to solve the system of kriging equations. The SPARSE package, however, was designed to solve large, sparse (zero entry) matrices with a particular emphasis on circuit simulations [23]. This package uses a series of linked lists to maintain and track the non-zero entries in the matrix. The covariance matrix, however, used to solve the ordinary kriging system of equations is extremely dense. The overhead, consequently, drastically increased with respect to the number of point source samples. As a result, the scalability of this algorithm utilizing the SPARSE package is not recommended. Additional research is needed concerning the solution to high-precision, dense matrices.
5.4.3 Parallel Architecture Implementation

The kriging algorithm, as described by this investigation, is extremely parallelizable. Initial preprocessing stages include the inverting of the covariance matrix, however, once this is accomplished, it is theoretically possible to be stored in a global section of memory. Each unsampled location in space and time may then calculate the its corresponding distances with respect to the other sampled locations and solve for the system of kriging equations for its own unique weights. Additional research concerning the implementation of this algorithm on parallel processing machines is needed.

5.4.4 Dynamic Volume Visualization

Research concerning the visualization of dynamic volumes is needed. With respect to volume graphics, the ability is needed to quickly iterate over large sequences of volume data. Ideally, a forward projection algorithm to initially display the volume would be the most desirable. Preprocessing rays from the eyepoint to the volume could be stored and re-used to display a series of volumes representing dynamic evolutions of a phenomenon. Backward projection of only the attributably changed voxels of sequencing volumes would then provide a mechanism for quickly iterating of dynamic volumes of data.

SunVision's<sup>TM</sup> volume rendering software SunVoxel was used in this investigation to volumetrically animate over the estimated values and variances. The ability to volume render a volumetric data set is provided by SunVoxel. The animation of a series of volumes is not readily supported in this package. This investigation developed
a simple "patch" to interface with SunVoxel database, in order to sequentially load
and render a series of volume data sets representing the dynamic changes of the
subsurface plume. Once the volumes were rendered, a "snap-shot" of the image was
obtained. By concatenating these series of images, an animation of the both the plume
and the associated variance was generated. Ideally, this type of functionality would be
supplied by an application and the analyst would be able to interact and eventually
steer the animating volume. This investigation plans to research additional volume
rendering applications including such packages as DataExplorer™ and AVSTM.

5.4.5 Probabilistic-Deterministic Modeling: Kriging and Hydrodynamic Equations

The problems of estimating values of dispersivity from various types of field data con-
tinues to be an area of active research in the field of contaminant hydrogeology [8].
New techniques based on stochastic transport theory utilizing geostatistical concepts
of regionalized variable theory are becoming more popular for extracting sets of trans-
port parameters from point source observations. These techniques utilize point source
observation for determining horizontal and vertical correlation length scales which in
turn can be used to calculate the asymptotic longitudinal dispersivity values. These
values can then be incorporated into the hydrodynamic modeling equations and used
to model and predict subsurface flow.

Utilizing this approach, however, the longitudinal dispersivity values calculated
from point source observations are assumed constant throughout the entire plume.
By utilizing geostatistical kriging techniques, asymptotic longitudinal dispersivity
values could be calculated at discretized locations based on the determined correlation lengths, *a priori*, site specific, information, and point source values of hydraulic conductivity. By utilizing probabilistic modeling techniques, such a geostatistical kriging, dispersivity values more representative of the actual site could be obtained and incorporated into hydrodynamic calculations. This coupling of probabilistic and deterministic hydrogeological modeling techniques would ideally provide the "best" representation of subsurface contaminant flow. This investigation plans additional research in the coupling of these methodologies.
CHAPTER VI

Conclusion

Understanding the processes affecting the transport and ultimate fate of subsurface non-point contamination is an integral part of water resource management. The detection, mapping and tracking of non-point source groundwater pollution are the first steps toward the development of effective management techniques. Effective water resources management requires efficient and reliable methods of determining, characterizing, and visualizing the dynamic nature of subsurface contamination. The process of mapping and describing ground water contamination based upon data collected from a fixed network, however, is extremely complex with a large potential for error. Usually a limited number of samples in both spatial and temporal dimensions are available to assess a complex, continuous, dynamic process whose state variables change in both time and space. Nevertheless, the characterization and extent of the contamination plume is generally assessed based on the point source data collected by subsurface contamination detection methods. Estimation at unsampled spatial and temporal locations is consequently required for the analyst to understand and describe the dynamic and amorphic nature of non-point source subsurface contamination. The ability to ascertain, and incorporate a priori, site specific, hydrogeological characteristics is a necessary improvement toward the estimation of unsampled spa-
tial and temporal locations. In addition, the ability to assess the variance associated with the estimated location not only is a powerful tool for assess the accuracy of the estimation, but served as an indicator of where additional sampled information is needed. Four-dimensional geostatistical kriging, a technique based on regionalized variable theory, is introduced in this investigation as a powerful mechanism that provides both estimates and variances for unsampled spatial and temporal locations based on the statistical correlations between point source observations.

The capability of estimate spatial and temporal estimated values and variances of complex and dynamic natural phenomenon, however, is extremely compromised by traditional visualization techniques. Phenomena that are amorphic and volumetric in nature, such as subsurface contamination, require visualization techniques that do not approximate or transform the true nature of the data. The approximation of dynamic and amorphic phenomenon by surfaces and edges, understates the complexity of the problem and thus complicate matters by offering simplified, and possibly erroneous results. By assigning the nitrate concentrations estimations and variances generated from the set of kriging equations directly into a volumetric data structure, the ability to volumetrically visualize the phenomenon was possible. The ability to animate through volumes of nitrate concentrations with associated opacities and colors, provides additional insight into the dynamic and complex processes associated with non-point source subsurface contamination. In addition, by animating through volume of variances, and visualize the dynamic nature of error, additional insight is gained for future sampling locations in both space and time.
Much research is still needed concerning spatial and temporal estimation. The ability to estimate unsampled spatial and temporal locations and to calculate the variance associated with estimate, provides great possibilities for understanding complex processes. The ability to volumetrically navigate through dynamic and volumetric data provides crucial insight into the complex patterns and trends. The coupling of four-dimensional geostatistical kriging and volumetric visualization provides a new and powerful mechanism for understanding and visualizing complex, dynamic natural phenomenon.
Appendix A

Site Characteristics

This investigation was conducted at the Management Systems Evaluation Area (MSEA) site in Piketon, Ohio over the period of April, 1991 through September, 1992. The 650 acre farm site contains 14 local wells and 7 regional groundwater sampling wells (see Figure 35). All wells are Waterloo multi-level type. The site wells have port depths of 3.66 m (12 ft), 4.88 m (16 ft), 6.1 m (20 ft), and 7.32 m (24 ft) from the ground surface. The Ohio MSEA research site is located over the Scioto River Buried Valley Aquifer which consists of a series of homogeneous layers of glacial outwash consisting primarily of sand and medium-sized gravel at a depth of 23 to 26 m thick. This aquifer consists of highly transmissive glaciofluvial deposits that are used extensively as public and rural-water supplies [37]. Sampling has occurred monthly over the past three years, yielding spatial and temporal samplings of nitrate concentrations at discrete locations.
Figure 35: General site layout of the MSEA project including both local or site specific (S) and regional specific (R) multi-port well locations. The dashed rectangle over the agricultural plots in the North-East part of the site indicates the area of interest kriged in this investigation.
Appendix B

Animation of Estimated Nitrate Concentrations at Ohio MSEA Site: April, 1, 1991 through July 29, 1991

The following is a series of rendered volumes generated every two days from Day 4 (April 4, 1991) through Day 120 (July 29, 1991) of kriged estimated nitrate concentrations at the Ohio MSEA site. The kriged nitrate concentration volumes were comprised of 40x30x20 voxels with each voxel representing 100x100x1 feet. The animating volumes of data reflect the estimated change of nitrate concentrations under the agricultural plots located in the North-West corner of the MSEA site (Appendix A, Figure 35).
<table>
<thead>
<tr>
<th>Name</th>
<th>Minimum Value</th>
<th>Name</th>
<th>Opacity</th>
<th>Skip Layers</th>
<th>Color</th>
</tr>
</thead>
<tbody>
<tr>
<td>3ppm no3</td>
<td>060</td>
<td>3ppm no3</td>
<td>0.0001</td>
<td>00</td>
<td>blue</td>
</tr>
<tr>
<td>4ppm no3</td>
<td>080</td>
<td>4ppm no3</td>
<td>0.0010</td>
<td>00</td>
<td>blue</td>
</tr>
<tr>
<td>5ppm no3</td>
<td>100</td>
<td>5ppm no3</td>
<td>0.0100</td>
<td>00</td>
<td>blue</td>
</tr>
<tr>
<td>6ppm no3</td>
<td>120</td>
<td>6ppm no3</td>
<td>0.1500</td>
<td>00</td>
<td>blue</td>
</tr>
<tr>
<td>7ppm no3</td>
<td>140</td>
<td>7ppm no3</td>
<td>0.2000</td>
<td>00</td>
<td>green</td>
</tr>
<tr>
<td>8ppm no3</td>
<td>160</td>
<td>8ppm no3</td>
<td>0.3000</td>
<td>00</td>
<td>yellow</td>
</tr>
<tr>
<td>9ppm no3</td>
<td>180</td>
<td>9ppm no3</td>
<td>0.4000</td>
<td>00</td>
<td>red</td>
</tr>
<tr>
<td>&gt;=10ppm</td>
<td>200</td>
<td>&gt;=10ppm</td>
<td>0.8000</td>
<td>00</td>
<td>red</td>
</tr>
</tbody>
</table>

Figure 36: RGB table and opacities corresponding to nitrate levels used in the rendering of the kriged volumes of estimated concentrations.
Figure 37: Sequential kriged volumes of nitrate concentrations: Day 4 (April 4, 1991) through Day 18 (April 18, 1991) every two days. Animated sequence is from right to left, top to bottom.
Figure 38: Sequential kriged volumes of nitrate concentrations: Day 20 (April 20, 1991) through Day 34 (May 4, 1991) every two days. Animated sequence is from right to left, top to bottom.
Figure 39: Sequential kriged volumes of nitrate concentrations: Day 36 (May 6, 1991) through Day 50 (May 20, 1991) every two days. Animated sequence is from right to left, top to bottom.
Figure 40: Sequential kriged volumes of nitrate concentrations: Day 52 (May 22, 1991) through Day 66 (June 5, 1991) every two days. Animated sequence is from right to left, top to bottom.
Figure 41: Sequential kriged volumes of nitrate concentrations: Day 68 (June 7, 1991) through Day 82 (June 21, 1991) every two days. Animated sequence is from right to left, top to bottom.
Figure 42: Sequential kriged volumes of nitrate concentrations: Day 84 (June 23, 1991) through Day 99 (July 7, 1991) every two days. Animated sequence is from right to left, top to bottom.
Figure 43: Sequential kriged volumes of nitrate concentrations: Day 100 (July 9, 1991) through Day 114 (July 23, 1991) every two days. Animated sequence is from right to left, top to bottom.
Figure 44: Sequential kriged volumes of nitrate concentrations: Day 116 (July 25, 1991) through Day 120 (July 29, 1991) every two days. Animated sequence is from right to left, top to bottom.
Appendix C

Animation of Estimate Variances at Ohio MSEA Site: April, 1, 1991 through July 29, 1991

The following is a series of rendered volumes generated every two days from Day 4 (April 4, 1991) through Day 120 (July 29, 1991) of the variances associated with the kriged nitrate concentrations at the Ohio MSEA site. The variance volumes were comprised of 40x30x20 voxels with each voxel representing 100x100x1 feet. The animating volumes of data reflect the changing variances associated with the estimated nitrate concentrations under the agricultural plots located in the North-West corner of the MSEA site (Appendix A, Figure 35).
Figure 45: RGB table and opacities corresponding to variance levels used in the rendering of the volumes of variances associated with the kriged estimated nitrate concentrations.
Figure 46: Sequential variances associated with kriged volumes of nitrate concentrations: Day 4 (April 4, 1991) through Day 18 (April 18, 1991) every two days. Animated sequence is from right to left, top to bottom.
Figure 47: Sequential variances associated with kriged volumes of nitrate concentrations: Day 20 (April 20, 1991) through Day 34 (May 4, 1991) every two days. Animated sequence is from right to left, top to bottom.
Figure 48: Sequential variances associated with kriged volumes of nitrate concentrations: Day 36 (May 6, 1991) through Day 50 (May 20, 1991) every two days. Animated sequence is from right to left, top to bottom.
Figure 49: Sequential variances associated with kriged volumes of nitrate concentrations: Day 52 (May 22, 1991) through Day 66 (June 5, 1991) every two days. Animated sequence is from right to left, top to bottom.
Figure 50: Sequential variances associated with kriged volumes of nitrate concentrations: Day 68 (June 7, 1991) through Day 82 (June 21, 1991) every two days. Animated sequence is from right to left, top to bottom.
Figure 51: Sequential variances associated with kriged volumes of nitrate concentrations: Day 84 (June 23, 1991) through Day 99 (July 7, 1991) every two days. Animated sequence is from right to left, top to bottom.
Figure 52: *Sequential* variances associated with kriged volumes of nitrate concentrations: Day 100 (July 9, 1991) through Day 114 (July 23, 1991) every two days. Animated sequence is from right to left, top to bottom.
Figure 53: Sequential variances associated with kriged volumes of nitrate concentrations: Day 116 (July 25, 1991) through Day 120 (July 29, 1991) every two days. Animated sequence is from right to left, top to bottom.
Appendix D

Source Code for Four-Dimensional Geostatistical Kriging

/*
 * krigs.h
 *
 * Copyright (c) 1993 eric j. miller
 *
 * Permission to use, copy, and distribute this software and its
 * documentation for any purpose with or without fee is hereby granted,
 * provided that the above copyright notice appear in all copies and
 * that both that copyright notice and this permission notice appear
 * in supporting documentation.
 *
 * Permission to modify the software is granted, but not the right to
 * distribute the modified codes. Modifications are to be distributed
 * as patches to released version.
 *
 * This software is provided "as is" without express or implied warranty.
 *
 * AUTHOR: Eric J. Miller
 * emiller@olc.org
 * emiller@cis.ohio-state.edu
 *
 * DATE: 08/01/93
 *
 */

#define TRUE 1
#define FALSE 0

#define EXPONENTIAL 1
#define SPHERICAL 2
#define GAUSSIAN 3
#define LINEAR 4
#define SPECIFIC 5
#define NOLOG 0
#define LOG 1
#define BOTHLOG 2
#define NORMAL 0
#define NOSCALE 1
#define IN 0
#define OUT 1

124
/* Variogram function (action()) declarations */

extern double Exponential();  /* exponential variogram */
extern double Spherical();    /* Spherical variogram function */
extern double Gaussian();     /* Gaussian variogram function */

typedef struct vector
{
    float mag, dir;  /* Magnitude, direction */
} Vector;

typedef struct ipoint
{
    int x, y, z, t;
} iPoint;

typedef struct dpoint
{
    double x, y, z, t;
} dPoint;

typedef struct site  /* site specific kriging characteristics */
{
    iPoint ipt;
    int rows, cols, depth, window, wincr;
} Site;

typedef struct aniso
{
    double phi;  /* first rotational angle around axis */
    double theta;  /* second rotational angle around axis */
    double cosphi, sinphi;  /* Sin/Cos calculations for phi */
    double costheta, sintheta;  /* Sin/Cos calculations for theta */
} Aniso;

typedef struct variogram
{
    int type;  /* Type of variogram */
    float a;  /* range */
    float c0;  /* Nugget effect */
    float c1;  /* c0 + c1 = sill */
    double (*action)();  /* appropriate variogram action function */
} Variogram;

typedef struct geostuct
{
    int single;
    int isotropic;  /* Isotropic flag */
    Aniso aniso;  /* Anisotropic structure */
    Site site;  /* Site characteristics */
```c
Variogram variogram[DIM]; // Variogram structure for each of the dimensional analysis */
int c0, cl, a;
double (*action)(); /* appropriate variogram action function */
}Geostruct;

typedef struct temp_dist
{
    dPoint dpt;
    double v;
}DT;

DT *gdt; /* global data table */
int gdts; /* Number of data entries */

char *GMatrix = NULL; /* Covariance matrix (n+1)x(n+1) */
RealVector DMMatrix = (spREAL *) NULL; /* Right hand side distance matrix (n+1)x1 */
RealVector Weights = (spREAL *) NULL;

int debug;

FILE *fout1, *fout2;
FILE *fin1, *fin2;
FILE *fexcept1, *fexcept2;
FILE *fexcept1, *fexcept2;

FILE *flog;

int gvalmax, gvarmax;
int gvalmin, gvarmin;
int gvalmax, gvarmax;
int gvalmin, gvarmin;

int scale;

int kpts;

int logscale;

/* procedural function declaration */
extern double ProcessValue(int);
extern double ProcessVariance(Geostruct *);
extern int ProcessInputFile(struct cdat *, Geostruct *);
extern int CreateCovarianceMatrix(Geostruct *);
extern int ProcessSingle(struct cdat *, Geostruct *);
extern int ProcessEstimation(struct cdat *, Geostruct *);
extern int LoadGeostruct(struct cdat *, Geostruct *);
extern int PRINT(int, int, char *, iPoint *);
extern int Krige(Geostruct *, iPoint *);
extern double (*VariogramAction())(Geostruct *, double);
extern double StatisticalDist(dPoint *, Geostruct *);
extern void InitWeights();
```
/*
 * 4dhrig.c v.1.0 beta
 * 
 * Copyright (C) 1993 Eric J. Miller
 * 
 * Permission to use, copy, and distribute this software and its 
 * documentation for any purpose with or without fee is hereby granted, 
 * provided that the above copyright notice appear in all copies and 
 * that both that copyright notice and this permission notice appear 
 * in supporting documentation.
 * 
 * Permission to modify the software is granted, but not the right to 
 * distribute the modified code. Modifications are to be distributed 
 * as patches to released version.
 * 
 * This software is provided "as is" without express or implied warranty.
 * 
 * AUTHOR: Eric J. Miller
 * emiller@oclc.org
 * emiller@cis.ohio-state.edu
 * 
 * DATE: 08/01/93
 * */

#include <stdio.h>
#include <math.h>
#include <string.h>

#include "spMatrix.h"     /* Utilize sparse matrix package from NETLIB */
#include "spDef.h"
#include "cdat.h"
#include "trig.h"

main(argc, argv)
int argc;
char *argv[];
{
    struct cdat cdat;    /* DAT structure */
    int Single;
    int sx, sy;
    Geosstruct geostruct;  /* geosstruct structure */

    /* Initialize dat structure */

    if (argc < 2)
    {
        printf("Usage: exe dat-file\n");
        exit(0);
    }

dat = cdat_init(argv[1]);
    /* Extract debug */
    cdat_inti(dat, &debug, "Debug");
/* Init and load geostruct */
LoadGeostruct(dat, &geostruct);

/* Process input file and calculate distance matrix */
ProcessInputFile(dat, &geostruct);

/* Check to see if we are interested in single point of array */
cdat_gint2(dat, &Single, "SinglePoint", "Flag");
geostruct.single = Single;

if (Single)
{
    ProcessSingle(dat, &geostruct);
}
else
{
    ProcessEstimation(dat, &geostruct);
}

int LoadGeostruct(dat, geostruct)
struct dat *dat;
Geostruct *geostruct;
{
    char *sox, *soy, *ssoz, *sso;
    char *so0, *so1, *s1;

    int ox, oy, oz, ot;
    int rows, cols, depth, window;
    int tv;
    char *tc;

    /* Get site and geostruct specific information */
    cdat_gstring2(dat, &sox, "Origin", "x");
    cdat_gstring2(dat, &soy, "Origin", "y");
    cdat_gstring2(dat, &ssoz, "Origin", "z");
    cdat_gstring2(dat, &sso, "Origin", "t");
    cdat_gstring2(dat, &srows, "Site", "rows");
    cdat_gstring2(dat, &sdepth, "Site", "depth");
    cdat_gstring2(dat, &swindow, "Site", "window");
    cdat_gstring2(dat, &swincr, "Site", "inincr");

    geostruct->site.ipt.x = (int) atoi(sox);
    geostruct->site.ipt.y = (int) atoi(soy);
    geostruct->site.ipt.z = (int) atoi(ssoz);
    geostruct->site.ipt.t = (int) atoi(sso);
    geostruct->site.rows = (int) atoi(srows);
    geostruct->site.cols = (int) atoi(scols);
    geostruct->site.depth = (int) atoi(sdepth);
    geostruct->site.window = (int) atoi(swindow);
    geostruct->site.incr = (int) atoi(swincr);
/* Load isotropic flag */

/* Load geostruct variogram information */

cdat_gint2(dat, &geostruct->variogram[X].type, "VariogramX", "type");
geostruct->variogram[X].action = VariogramAction(geostruct->variogram[X].type);
cdat_gstring2(dat, &c, "VariogramX", "c0");
geostruct->variogram[X].c0 = atof(c);
cdat_gstring2(dat, &c, "VariogramX", "c1");
geostruct->variogram[X].c1 = atof(c);
cdat_gstring2(dat, &c, "VariogramX", "a");
geostruct->variogram[X].a = atof(c);

cdat_gint2(dat, &geostruct->variogram[Y].type, "VariogramY", "type");
geostruct->variogram[Y].action = VariogramAction(geostruct->variogram[Y].type);
cdat_gstring2(dat, &c, "VariogramY", "c0");
geostruct->variogram[Y].c0 = atof(c);
cdat_gstring2(dat, &c, "VariogramY", "c1");
geostruct->variogram[Y].c1 = atof(c);
cdat_gstring2(dat, &c, "VariogramY", "a");
geostruct->variogram[Y].a = atof(c);

cdat_gint2(dat, &geostruct->variogram[Z].type, "VariogramZ", "type");
geostruct->variogram[Z].action = VariogramAction(geostruct->variogram[Z].type);
cdat_gstring2(dat, &c, "VariogramZ", "c0");
geostruct->variogram[Z].c0 = atof(c);
cdat_gstring2(dat, &c, "VariogramZ", "c1");
geostruct->variogram[Z].c1 = atof(c);
cdat_gstring2(dat, &c, "VariogramZ", "a");
geostruct->variogram[Z].a = atof(c);

cdat_gint2(dat, &geostruct->variogram[T].type, "VariogramT", "type");
geostruct->variogram[T].action = VariogramAction(geostruct->variogram[T].type);
cdat_gstring2(dat, &c, "VariogramT", "c0");
geostruct->variogram[T].c0 = atof(c);
cdat_gstring2(dat, &c, "VariogramT", "c1");
geostruct->variogram[T].c1 = atof(c);
cdat_gstring2(dat, &c, "VariogramT", "a");
geostruct->variogram[T].a = atof(c);

/* Normalize variogram structure */

geostruct->action = geostruct->variogram[X].action;
geosstruct->c0 = geostruct->variogram[X].c0/geostruct->variogram[X].ci;
geostuct->c1 = (double) 1.0;
geosstruct->a = (double) 1.0;

if (geostruct->variogram[X].c0 == geostruct->variogram[Y].c0 &&
geostruct->variogram[X].c0 == geostruct->variogram[Z].c0 &&
geostruct->variogram[X].c0 == geostruct->variogram[T].c0 &&
geostruct->variogram[X].c1 == geostruct->variogram[Y].c1 &&
geostruct->variogram[X].c1 == geostruct->variogram[Z].c1 &&
geostruct->variogram[X].c1 == geostruct->variogram[T].c1 &&
geostruct->variogram[X].a == geostruct->variogram[Y].a &&
geostruct->variogram[X].a == geostruct->variogram[Z].a &&
geostruct->variogram[X].a == geostruct->variogram[T].a)
{
    geostruct->isotropic = TRUE;
else
    geostuct->isotropic = FALSE;

/* Load anisotropic information (if needed) */
cdat_gint2(dat, &tv, "AnisoRotDirection", "phi");
geostuct->aniso.phi = (double)((double)tv * 2.0 + 3.14159265)/360.0;
cdat_gint2(dat, &tv, "AnisoRotDirection", "theta");
geostuct->aniso.theta = (double)((double)tv * 2.0 + 3.14159265)/360.0;

/* Calculate static values for performance */
geostuct->aniso.cosphi = (double)cos(geostuct->aniso.phi);
geostuct->aniso.sinphi = (double)sin(geostuct->aniso.phi);
geostuct->aniso.costheta = (double)cos(geostuct->aniso.theta);
geostuct->aniso.sintheta = (double)sin(geostuct->aniso.theta);

/* Get some global values */
cdat_gint1(dat, &scale, "VScale");
cdat_gint1(dat, &logscale, "Logscale");
}

double (*VariogramAction(geotype))( )
int geotype;
{
    switch(geotype)
    {
        case EXPO\ential:
            return Exponential;
            break;
        case SPHERICAL:
            return Spherical;
            break;
        case GAUSSIAN:
            return Gaussian;
            break;
        default:
            printf("Unknown geostuct type \n");
            exit(1);
            break;
    }
}

int OpenLog(outfile)
char *outfile;
{
    char logfile[80];
    /* Open log file */
sprintf(logfile, "%s.log", outfile);

    if ((flog = (FILE *) fopen(logfile, "w")) == (FILE *) NULL)
    {
        printf("Unable to open log files...\n");
        exit(1);
    }
int IOFile(ft, scale, outfile, ipt)
int ft; /* File type (IN | OUT) */
int scale; /* Scale (NORMAL | RESCALE) */
char *outfile;
int *ipt;
{
    char file1[80];
    char file2[80];
    char ifile1[80];
    char ifile2[80];

    if (logscale == NOLOG)
    {
        printf(file1, (scale == NORMAL) ? "%s.val.t%03d" : "%s.val.t%03d.res", outfile, ipt->t);
        printf(file2, (scale == NORMAL) ? "%s.var.t%03d" : "%s.var.t%03d.res", outfile, ipt->t);

        if (ft == IN)
        {
            if (((fin = (FILE *) fopen(file1, "r")) == (FILE *) NULL) ||
                ((fin = (FILE *) fopen(file2, "r")) == (FILE *) NULL))
            {
                printf("Unable to i/o files...\n");
                exit(1);
            }
        }
        else
        {
            if (((fout1 = (FILE *) fopen(file1, "w")) == (FILE *) NULL) ||
                ((fout2 = (FILE *) fopen(file2, "w")) == (FILE *) NULL))
            {
                printf("Unable to i/o files...\n");
                exit(1);
            }
        }
    }

    /* Log */
    fprintf(flog, "Processing: %s and %s\n", file1, file2);
    fflush(flog);
}
else if (logscale == LOG)
{
    printf(ifile1, (scale == NORMAL) ? "%s.val.log.t%03d" : "%s.val.log.t%03d.res", outfile, ipt->t);
    printf(ifile2, (scale == NORMAL) ? "%s.var.log.t%03d" : "%s.var.log.t%03d.res", outfile, ipt->t);

    if (ft == IN)
    {
        if (((fin = (FILE *) fopen(ifile1, "r")) == (FILE *) NULL) ||
            ((fin = (FILE *) fopen(ifile2, "r")) == (FILE *) NULL))
        {
            printf("Unable to i/o files...\n");
            exit(1);
        }
    }
}
else
{
    if (((float1 = (FILE *) fopen(file1, "r")) == (FILE *) NULL) ||
         ((float2 = (FILE *) fopen(file2, "r")) == (FILE *) NULL))
    {
        printf("Unable to i/o files...\n");
        exit(1);
    }
}

/* Log */

fprintf(flog, "Processing: %s and %s\n", file1, file2);
fflush(flog);

else if (logscale == BOTHLOG)
{
    sprintf(file1, (scale == NORMAL) ? "%s.val.tX03d" : "%s.val.tX03d.res", outfile, ipt->t);
    sprintf(file2, (scale == NORMAL) ? "%s.var.SX03d" : "%s.var.SX03d.res", outfile, ipt->t);
    sprintf(file1, (scale == NORMAL) ? "%s.val.log.tX03d" : "%s.val.log.tX03d.res", outfile, ipt->t);

    if (ft == IE)
    {
        if (((fnew1 = (FILE *) fopen(file1, "w")) == (FILE *) NULL) ||
             ((fnew2 = (FILE *) fopen(file2, "w")) == (FILE *) NULL) ||
             ((fnew1 = (FILE *) fopen(file1, "r")) == (FILE *) NULL))
        {
            printf("Unable to i/o files...\n");
            exit(1);
        }
    }
    else
    {
        if (((fnew1 = (FILE *) fopen(file1, "w")) == (FILE *) NULL) ||
             ((fnew2 = (FILE *) fopen(file2, "w")) == (FILE *) NULL) ||
             ((fnew1 = (FILE *) fopen(file1, "r")) == (FILE *) NULL))
        {
            printf("Unable to i/o files...\n");
            exit(1);
        }
    }

/* Log */

fprintf(flog, "Processing: %s and %s\n", file1, file2);
fflush(flog);
fprintf(flog, "Processing log: %s\n", file1);
fflush(flog);

else
{
    printf("Invalid input DAT logscale entry\n");
    exit(1);
}
int CloseOutFiles()
{
    if (fout1) fclose(fout1);
    if (fout2) fclose(fout2);
    if (fcout1) fclose(fcout1);
    if (fcout2) fclose(fcout2);
}

int CloseInFiles()
{
    if (fin1) fclose(fin1);
    if (fin2) fclose(fin2);
    if (fmlin1) fclose(fmlin1);
    if (fmlin2) fclose(fmlin2);
}

int ProcessEstimation(dat, geostruct)
struct cdat *dat;
Geostruct *geostruct;
{
    int i, j, k;                /* loop control variables */
    int tval, tvar;            /* temp values for histogram distribution */
    char *outfile;             /* Output filename */
    int x, y;
    int it, ix, iy;            /* loop control variables for the site */
    int itmax, imax, xmax, ymax; /* loop control site maximums */
    iPoint ipt;                /* integer point value for be kriged */

    RealNumber *pElement;

    float tfl;

    /* Create Covariance Matrix based on Input geostruct description */
    CreateCovarianceMatrix(geostruct);

    /* Factor Covariance matrix for increased solve speed */
    spFactor(CMatrix);

    /* Allocate Weights */
    Weights = ALLOC(RealNumber, gdts + 2);

    /* Get output file */
    cdat_gstring(dat, &outfile, "OutputFilename");

    /* open log file */
    OpenLog(outfile);
/* Loop through grid */

givalmax = givalmax = givalmax = givalmax = -1;
givalmin = givalmin = givalmin = givalmin = 10000 * scale;

/* map (x,y,z,t) coordinate space to loop space */

for (it = 0, tmax = abs(geostruct->site.window); (it < tmax); it += geostruct->site.wincr)
{
    /* map from increment space to site space */
    ipt.t = geostruct->site.window > 0) ? it+geostruct->site.ipt.t :
            -it+geostruct->site.ipt.t;

    /* Open/Close new file for each time stamp */
    /* Open output files, normal scale */
    IDFile(BUT, NORMAL, outfile, &ipt);

    for (ix = 0, xmax = abs(geostruct->site.depth); (ix < xmax); ix++)
    {
        /* map from increment space to site space */
        ipt.x = (geostruct->site.depth > 0) ? ix+geostruct->site.ipt.x :
                -ix+geostruct->site.ipt.x;

        for (iy = 0, ymax = abs(geostruct->site.rows); (iy < ymax); iy++)
        {
            /* map from increment space to site space */
            ipt.y = (geostruct->site.rows > 0) ? iy+geostruct->site.ipt.y :
                    -iy+geostruct->site.ipt.y;

            for (ix = 0, xmax = abs(geostruct->site.cols); (ix < xmax); ix++)
            {
                /* map from increment space to site space */

                ipt.x = (geostruct->site.cols > 0) ? ix+geostruct->site.ipt.x :
                        -ix+geostruct->site.ipt.x;

                Krigge(geostruct, &ipt);
            }
        }
    }

    /* Log */

    fprintf(flog, "Kriged %d points\n", kpts);
    fflush(flog);

    /* Close output file */
    CloseOutFiles();
}

/* Renumber and scale output binary */
/* Load every time slice (i.e. file) and renumber */
for (it = 0, tmax = abs(geostruct->site.window); it < tmax; it += geostruct->site.wincr)
{
    /* map from increment space to site space */
    ipt.t = (geostruct->site.window > 0) ? it + geostruct->site.ipt.t : -it + geostruct->site.ipt.t;
    /* Reload input file, normal scale */
    IOFile(IN, NORMAL, outfile, &ipt);
    /* Output new binary rescaled information */
    IOFile(OUT, RESCALE, outfile, &ipt);
    for (i = 0; i < abs(geostruct->site.depth); i++)
    {
        for (j = 0; j < abs(geostruct->site.rows); j++)
        {
            for (k = 0; k < abs(geostruct->site.cols); k++)
            {
                if (logscale == NOLOG)
                {
                    fscanf(fin1, "%d", &tval);
                    fscanf(fin2, "%d", &tvar);
                    tval = (tval - gvalmin)*(double)(255.0/(double)(gvalmax-gvalmin));
                    tvar = (tvar - gvarmin)*(double)(255.0/(double)(gvarmax-gvarmin));
                    /* Output rescaled binary information */
                    fprintf((char) tval, fout1);
                    fflush(fout1);
                    fprintf((char) tvar, fout2);
                    fflush(fout2);
                }
                else if (logscale == LOG)
                {
                    fscanf(fin1, "%d", &tval);
                    fscanf(fin2, "%d", &tvar);
                    tval = (tval - gvalmin)*(double)(255.0/(double)(gvalmax-gvalmin));
                    tvar = (tvar - gvarmin)*(double)(255.0/(double)(gvarmax-gvarmin));
                    /* Output rescaled binary information */
                    fprintf((char) tval, fout1);
                    fflush(fout1);
                    fprintf((char) tvar, fout2);
                    fflush(fout2);
                }
                else if (logscale == BOTHLOG)
                {
                    fscanf(fin1, "%d", &tval);
                    fscanf(fin2, "%d", &tvar);
                    tval = (tval - gvalmin)*(double)(255.0/(double)(gvalmax-gvalmin));
                    tvar = (tvar - gvarmin)*(double)(255.0/(double)(gvarmax-gvarmin));
                    /* Output rescaled binary information */
                }
            }
        }
    }
}
tvar = (tvar - gvarmin)*(double)(255.0/(double)(gvarmax-gvarmin));

/* Output rescaled binary information */

fprintf(foutl, "%d", &tval);
fprintf(fout2, "%d", &tval);

fclose(flin1);
fclose(flin2);

CloseInFiles();
CloseOutFiles();

/* Log */

LogSession(geostruct);

int LogSession(geostruct)
{ Geostruct *geostruct;

fprintf(flog, "Kriged Xd points total\n", kpts);
fprintf(flog, "nScale Factor: Xd\n", scale);
fprintf(flog, "\nLogtype: (0 == NOLOG || 1 == LOG || 2 == BOTHLOG): Xd\n", logscale);
fprintf(flog, "\nRescale\n");

if (logscale == NOLOG)
{ fprintf(flog, "Value: (Maximum Xd, Minimum Xd)\n", gvalmax, gvalmin);
  fprintf(flog, "Variance: (Maximum Xd, Minimum Xd)\n", gvarmax, gvarmin);
}
else if (logscale == LOG)
{ fprintf(flog, "Log Value: (Maximum Xd, Minimum Xd)\n", gvalmax, gvalmin);
  fprintf(flog, "Variance: (Maximum Xd, Minimum Xd)\n", gvarmax, gvarmin);
}
else
{
printf(flog, "Value: (Maximum %d, Minimum %d)\n", gvalmax, gvalmin);
printf(flog, "Log Value: (Maximum %d, Minimum %d)\n", glogmax, glogmin);
printf(flog, "Variance: (Maximum %d, Minimum %d)\n", gvarmax, gvarmin);
}

printf(flog, "Site Characteristics\n");
printf(flog, "Origin: (x: %d, ty: %d, tx: %d, tt: %d)\n",
gestruct->site.ipt.x,
gestruct->site.ipt.y,
gestruct->site.ipt.x,
gestruct->site.ipt.t);
printf(flog, "Cols: %d, CRows: %d, tDepth: %d, tTimespan: %d, tTimesInc: %d\n",
gestruct->site.cols,
gestruct->site.cols,
gestruct->site.depth,
gestruct->site.window,
gestruct->site.window);
printf(flog, "\n\nVariogram (Type: (1 == EXPONENTIAL || 2 == SPHERICAL || 3 == GAUSSIAN))\n");
printf(flog, "VarioX: Type: %d, tcc0: %f, tci: %f, ta: %f\n",
gestruct->varigram[X].type,
gestruct->varigram[X].c0,
gestruct->varigram[X].c1,
gestruct->varigram[X].a);
printf(flog, "VarioY: Type: %d, tcc0: %f, tci: %f, ta: %f\n",
gestruct->varigram[Y].type,
gestruct->varigram[Y].c0,
gestruct->varigram[Y].c1,
gestruct->varigram[Y].a);
printf(flog, "VarioZ: Type: %d, tcc0: %f, tci: %f, ta: %f\n",
gestruct->varigram[Z].type,
gestruct->varigram[Z].c0,
gestruct->varigram[Z].c1,
gestruct->varigram[Z].a);
printf(flog, "\nAnisotropic Rotation: phi(\rad): %f, \theta(\rad): %f\n",
gestruct->aniso.phi,
gestruct->aniso.theta);
fflush(flog);
close(flog);
}

int Krig(estruct, ipt)
Geostruct *geestruct;
iPoint *ipt;
{
double value, variance;
int tval, tvar;
int tval, tivar;
int newmatrix;
int newcovariance;
int i;
kpts++;
/* Load distance matrix based on single point */
LoadDistanceVector(geostruct, ipt);

if (debug)
{
    printf("mb:\n");
    for (i = 0; i < gdts+1; i++)
        printf("\n", DMatrix[i+1]);
}

/* (re)initialize weights */
InitWeights();

/* Solve for weights */
spSolve(CMatrix, DMatrix, Weights);

if (debug)
{
    printf("mx:\n");
    for (i = 0; i < gdts+2; i++)
        printf("\n", Weights[i+1]);
}

/* Calculate value and variance for point of interest */
if (.geostruct->single)
{
    value = (double) ProcessValue(logscale);
    variance = (double) ProcessVariance(geostruct);
    printf("Point: %d, %d, %d: Value: %f\n\n",
            ipt->x, ipt->y, ipt->z, value);  
    return;
}

if (logscale == NOLOG)
{
    value = (double) ProcessValue(logscale);
    variance = (double) ProcessVariance(geostruct);
    tval = (int) (value * scale);
    tvar = (int) (variance * scale);
    /* Get minimum and maximum values for display */
    gvalmax = MAX(gvalmax, tval);
    gvarmax = MAX(gvarmax, tvar);
    gvalmin = MIN(gvalmin, tval);
    gvarmin = MIN(gvarmin, tvar);
    fprintf(fout1, "%d \n", tval);
    fprintf(fout2, "%d \n", tvar);
    return;
}
else if (logscale == LOG)
{
  value = (double) ProcessValue(logscale);
  variance = (double) ProcessVariance(geostruct);
  /* Logscale */
  tval = (int) (value * scale);
  tvar = (int) (variance * scale);
  /* Get minimum and maximum values for display */
  gvalmax = MAX(gvalmax, tval);
  gvarmax = MAX(gvarmax, tvar);
  gvalmin = MIN(gvalmin, tval);
  gvarmin = MIN(gvarmin, tvar);
  fprintf(fout1, "\d ", tval);
  fprintf(fout2, "\d ", tvar);
}
else if (logscale == BOTHLOG)
{
  value = (double) ProcessValue(NULLLOG);
  variance = (double) ProcessVariance(geostruct);
  tval = (int) (value * scale);
  tvar = (int) (variance * scale);
  /* Get minimum and maximum values for display */
  gvalmax = MAX(gvalmax, tval);
  gvarmax = MAX(gvarmax, tvar);
  gvalmin = MIN(gvalmin, tval);
  gvarmin = MIN(gvarmin, tvar);
  fprintf(fout1, "\d ", tval);
  fprintf(fout2, "\d ", tvar);
  /* Logscale */
  value = (double) ProcessValue(LOG);
  tval = (int) (value * scale);
  /* Get minimum and maximum values for display */
  gvalmax = MAX(gvalmax, tval);
  gvarmax = MAX(gvarmax, tvar);
  fprintf(fout1, "\d ", tval);
}
return;
}

int ProcessSingle(dat, geostruct)
struct cdat *dat;
Geostruct *geostruct;
{
    char *sx, *sy, *sz, *st;

    iPoint ipt;

    /* Only interested in single point calculations */

    cdat_gstring2(dat, &sx, "SinglePoint", "x");
    cdat_gstring2(dat, &sy, "SinglePoint", "y");
    cdat_gstring2(dat, &sz, "SinglePoint", "z");
    cdat_gstring2(dat, &st, "SinglePoint", "t");

    ipt.x = (int) atoi(sx);
    ipt.y = (int) atoi(sy);
    ipt.z = (int) atoi(sz);
    ipt.t = (int) atoi(st);

    fout1 = (FILE *) NULL;
    fout2 = (FILE *) NULL;

    /* Create Covariance Matrix based on Input geostruct description */

    CreateCovarianceMatrix(geostruct);

    /* Factor Covariance matrix for increased solve speed */

    spFactor(CMatrix);

    /* Allocate Weights */

    Weights = ALLOC(RealNumber, gdts + 2);

    Krige(geostruct, &ipt);
}

void InitWeights()
{
    register int i;

    for (i = 0; i < gdts + 2; i++)
        Weights[i] = 0.0;
}

double ProcessValue(logtype)
int logtype;
{
    double value;

    register int i;

    value = 0.0;

    if (logtype)
    {
        for (i = 0; i < gdts; i++)
        {
            value += (double) Weights[i+1]*log(gdt[i].v);
double ProcessVariance(geostruct)
Geostruct *geostruct;
{
    double variance;
    double wsum;
    register int i;
    variance = 0.0;
    variance = (double) (geostruct->c0 + geostruct->c1);
    for (i = 0, wsum = 0.0; i < gdts; i++)
    {
        variance -= (double) Weights[i+i]*DMatrix[i+i];
    }
    variance -= (double) Weights[i+i];
    variance *= (double) (geostruct->variogram[X].c0+geostruct->variogram[X].c1);
    return variance;
}

int CreateCovarianceMatrix(geostruct)
Geostruct *geostruct;
{
    int i, j;
    RealNumber *pElement;
    dPoint dpt;
    dPoint s;
    double nv;
    double dist;
    double cvalue;
    double vvalue;
    double vvalue;
    int sError;
    MatrixPtr Matrix;
    double xn, yn, zm, tn;          /* Normalized values */
/* Create Covariance matrix (n+1)x(n+1) */
CMatrix = spCovCreate(gdts+1, 0, &spError);

if (CMatrix == (char *) NULL)
{
    printf("Insufficient memory to allocate Covariance matrix\n");
    exit(1);
}

/* Process covariance matrix */
Matrix = (MatrixPtr) CMatrix;

for (i = 0; i < gdts; i++)
{
    for (j = i; j < gdts; j++)
    {
        dpt.x = gdtt[j].dpt.x - gdtt[i].dpt.x;
        dpt.y = gdtt[j].dpt.y - gdtt[i].dpt.y;
        dpt.z = gdtt[j].dpt.z - gdtt[i].dpt.z;
        dpt.t = gdtt[j].dpt.t - gdtt[i].dpt.t;

        dist = StatisticalDist(&dpt, geostruct);
        cvalue = (double) (geostruct->c0+geostruct->c1)*geostruct->action(geostruct, dist);

        /* Mirror covariance value */
        pElement = (RealNumber *) spGetElement(CMatrix, j+1, i+1);
        *pElement = cvalue;

        pElement = (RealNumber *) spGetElement(CMatrix, i+1, j+1);
        *pElement = cvalue;
    }
}

/* How do borders (n+1) rov and col */
for (i = 0, j = gdts+1; i < gdts; i++)
{
    /* Assign value in n+1 row */
    pElement = spGetElement(CMatrix, i+1, j);
    *pElement = (double) 1.0;
}

for (j = 0, i = gdts+1; j < gdts; j++)
{
    pElement = spGetElement(CMatrix, i, j+1);
    *pElement = (double) 1.0;
}

/* assign n+1, n+1 value */
pElement = spGetElement(CMatrix, gdts+1, gdts+1);
*pElement = (double) 0.0;
double StatisticalDist(dpt, geostruct)
DPoint *dpt;
Geostruct *geostruct;
{
    double Dist;
    double xn, yn, zn, tn;
    double xr, yr, xz;
    
    /* calculate distance based on isotropic or anisotropic values */
    if (geostruct->isotropic)
    {
        /* normalize distances */
        xn = (double) dpt->x/geostruct->variogram[X].a;
        yn = (double) dpt->y/geostruct->variogram[Y].a;
        zn = (double) dpt->z/geostruct->variogram[Z].a;
        tn = (double) dpt->t/geostruct->variogram[T].a;
        Dist = (double) sqrt(xn*xn+yn*yn+zn*zn+tn*tn);
    }
    else
    {
        /* Transform from iso-space to aniso-space */
        /* 3D rotation... No temporal rotation */
        /* rotation phi clockwise around cartesian Z axis */
        /* rotate theta clockwise around new Y' axis */
        xr = (double) ((dpt->x*geostruct->aniso.cosphi*geostruct->aniso.costheta) +
                       (dpt->y*geostruct->aniso.sinphi*geostruct->aniso.costheta) +
                       (dpt->z*geostruct->aniso.sintheta));
        yr = (double) (-dpt->x*geostruct->aniso.sinphi +
                       (dpt->y*geostruct->aniso.cosphi));
        xz = (double) (-dpt->x*geostruct->aniso.cosphi*geostruct->aniso.sintheta) +
                       (dpt->y*geostruct->aniso.sinphi*geostruct->aniso.sintheta) +
                       (dpt->z*geostruct->aniso.costheta));
        
        /* scale and normalize from aniso-space to isospace based on temporal magnitude */
        xn = (double) xr/geostruct->variogram[X].a;
        yn = (double) yr/geostruct->variogram[Y].a;
        zn = (double) xz/geostruct->variogram[Z].a;
        tn = (double) dpt->t/geostruct->variogram[T].a;
        Dist = (double) sqrt(xn*xn+yn*yn+zn*zn+tn*tn);
    }

    return Dist;
}

int LoadDistanceVector(geostruct, ipt)
Geostruct *geostruct;
IPoint *ipt;
{
    int i, j;
dPoint dpt;
dPoint s; /* Square */
double Dist; /* Normal distance value */
double nv;

if (DMatrix)
    FREE(DMatrix);
/* allocate new matrix */
DMatrix = ALLOC(RealNumber, gdts + 2);
/* Loop through distance table */
DMatrix[0] = 0.0;
for (i = 0; i < gdts; i++)
{
    dpt.x = (double) ipt->x - gdts[1].dpt.x;
    dpt.y = (double) ipt->y - gdts[1].dpt.y;
    dpt.z = (double) ipt->x - gdts[1].dpt.z;
    dpt.t = (double) ipt->t - gdts[1].dpt.t;
    Dist = StatisticalDist(&dpt, geosstruct);
    /* Assign value in distance table */
    DMatrix[i+1] = (double) (geosstruct->c0+geosstruct->c1)*geosstruct->action(geosstruct, Dist);
}
DMatrix[i+1] = (double) 1.0;
}

int ProcessInputFile(dat, geosstruct)
struct cdat *dat;
Geosstruct *geosstruct;
{
    char *infile; /* Input filename */
    FILE *fin; /* Input file matrix */
    char tbuff[1000];
    int i, j;
    int values;
    RealNumber *pElement;
    RealNumber *pInitInfo;
    RealNumber *pValue[1000];
    int *pError;
    int Count = 0;
    double Real;
    double dx, dy;
    float tx, ty, tz, tv;
    int tt;
char id[20];

data_gstring1(dat, &infile, "InputFilename");

if ((fin = (FILE *) fopen(infile, "r")) == (FILE *) NULL)
    {
        printf("Unable to process input file...\n");
        exit(1);
    }

/* read temp description line */

fgets(tbuf, 1000, fin);

printf("Processing: %s\n", tbuf);

fscanf(fin, "%ld", &values);

/* Allocate global distance table */
gdt = (DT *) malloc((size_t) (sizeof(DT) * values));

if (gdt == (DT *) NULL)
    {
        printf("Insufficient memory to allocate global data table \n");
        return TRUE;
    }

/* Assign global number of distances */
gdtype = 0;

/* read in sorted table */

for (i = 0; ((i < values) && (!feof(fin)))); i++)
    {
        fscanf(fin, "%s%f%f%f%f%f", id, &tx, &ty, &tz, &tt, &tv);

        if (tv != (double) 0.0)
            {
                gdt[gdtype].dpt.x = (double) tx;
                gdt[gdtype].dpt.y = (double) ty;
                gdt[gdtype].dpt.z = (double) tz;
                gdt[gdtype].dpt.t = (double) tt;
                gdt[gdtype].v = (double) tv;
                gdtype++;
            }
    }

double Spherical(geostruct, h)
Geostruct *geostruct;
double h;
{
    double hovers;
    if (h == 0.0)
        {
            return((double) geostruct->c0);
else if (h >= geostruct->a)
    {
        return((double) geostruct->c0 + geostruct->c1);
    }
else
    {
        hovera = (double) h/geostruct->a;
        return((double) 1.5 * hovera - 0.5 * (hovera * hovera * hovera));
    }
}

double Exponential(geostruct, h)
Geostruct *geostruct;
double h;
{
    if (h <= 0.0)
    {
        return((double) geostruct->c0 + geostruct->c1);
    }
else
    {
        return((double) (geostruct->c0 + geostruct->c1*exp((-3.0/geostruct->a)*h)));
    }
}

double Gaussian(geostruct, h)
Geostruct *geostruct;
double h;
{
    if (h <= 0.0)
    {
        return((double)(geostruct->c0 + geostruct->c1));
    }
else
    {
        return((double)(geostruct->c1*(exp((-3.0*h)/geostruct->a))/geostruct->a));
    }
}
Appendix E

Source Code for (VMAP) Variogram Modeling Analysis Package

/*
 * vmap.h
 *
 * Copyright (C) 1993 Eric J. Miller
 *
 * Permission to use, copy, and distribute this software and its
 * documentation for any purpose with or without fee is hereby granted,
 * provided that the above copyright notice appear in all copies and
 * that both that copyright notice and this permission notice appear
 * in supporting documentation.
 *
 * Permission to modify the software is granted, but not the right to
 * distribute the modified code. Modifications are to be distributed
 * as patches to released version.
 *
 * This software is provided "as is" without express or implied warranty.
 *
 * AUTHOR: Eric J. Miller
 * emiller@clc.org
 * emiller@cis.ohio-state.edu
 *
 * DATE: 08/01/93
 *
 */

/* Macro definitions */

#define ROUND(a) ((a)+0.5)
#define DTOR(f) (((f) * 2.0 * 3.14159265)/560.0)
#define SAME_SIZES(a, b) \
    (((long) ((unsigned long) a ~ (unsigned long) b)) == 0)

/* Global definitions */

#define TRUE 1
#define FALSE 0

#define BORDER 40
#define BACKGROUND_WIDTH 700
#define BACKGROUND_HEIGHT 360
#define CMDFRAME_HEIGHT 280
#define CMDFRAME_WIDTH 300

148
#define DISPLAY_WIDTH 620
#define DISPLAY_HEIGHT 270
#define INPUT_WIDTH 100
#define NONE 0
#define SPHERICAL 1
#define GAUSSIAN 2
#define EXPONENTIAL 3
#define LINEAR 4
#define XY_PLANE 0
#define YZ_PLANE 1
#define ZT_PLANE 2
#define DISPLAY_POINTS 0
#define DISPLAY_LINES 1
#define DISPLAY_DLINES 2
#define OMIDIRECTIONAL 0
#define DIRECTIONAL 1
#define CO_DEFAULT 0
#define C1_DEFAULT 10
#define A_DEFAULT 10
#define XANGE_DEFAULT 20
#define YANGE_DEFAULT 1

/* KEY deinitions */
#define BASE_FRAME_KEY 100
#define FRAME_KEY 101
#define CO_KEY 102
#define C1_KEY 103
#define A_KEY 104
#define MODEL_KEY 105
#define XANGE_KEY 106
#define YANGE_KEY 107
#define LAG_KEY 108
#define TOL_KEY 109
#define CANVAS_KEY 110
#define ANG_KEY 120
#define ANG_TOL_KEY 121
#define LAG_MAX_KEY 122
#define DVAR_TYPE_KEY 123
#define PLANE_TYPE_KEY 124
#define DISPLAY_TYPE_KEY 125
#define BACKGROUND_KEY 126
#define KEY_KEY 200
#define NEW_THEORETICAL_KEY 201
#define NEW_OBSERVED_KEY 202
#define HVALX_KEY 309
#define HVALY_KEY 310
#define HRAD_KEY 311
#define VVALX_KEY 312
#define VVXL_KEY 313
#define TVALX_KEY 314
#define TVALX_KEY 315
/* XView globals */

Frame VariogramFrame;
Panel VariogramPanel;
Canvas VariogramCanvas;
Canvas BackgroundCanvas;

Frame FilterFrame;
Panel FilterPanel;

GC tgc;             /* theoretical GC Structure */
XGCValues tgcvalues; /* theoretical GC values */
GC ogc;             /* observed GC Structure */
XGCValues ogcvalues; /* observed GC values */

typedef struct vector
{
    float mag, dir;     /* Magnitude, direction */
} Vector;

typedef struct lpoint
{
    long x, y, z;
    long t;
} lPoint;

typedef struct lpoint
{
    int x, y, z;
    int t;
} lPoint;

typedef struct dpoint
{
    double x, y, z;
    int t;
} dPoint;

typedef struct temp_dist    /* distance table struct */
{
    char wid[20];    /* Well ID */
    double x,y,z;
    int t;
    double v;
} GIT;

typedef struct polygon
{
    dPoint pt[3];    /* Assume triangle */
    int num;
} Polygon;

GIT gvt[10000];      /* global value table */
int gvtsize;         /* number of global values */

GIT gdt[5000];       /* global distance table (i.e. filter values) */
int gdts;             /* Number of distances (i.e. n) */
int Debug;
char *DMatrix = NULL;  /* Distance matrix */
/*
 * vmap.xview
 *
 * Copyright (C) 1993 eric j. miller
 *
 * Permission to use, copy, and distribute this software and its
 * documentation for any purpose with or without fee is hereby granted,
 * provided that the above copyright notice appear in all copies and
 * that both that copyright notice and this permission notice appear
 * in supporting documentation.
 *
 * Permission to modify the software is granted, but not the right to
 * distribute the modified code. Modifications are to be distributed
 * as patches to released version.
 *
 * This software is provided "as is" without express or implied warranty.
 *
 * AUTHOR: Eric J. Miller
 * emiller@osc.edu
 * emiller@cis.ohio-state.edu
 *
 * DATE: 08/01/93
 *
 */

VariogramFrame = xv_create(NULL, FRAME,
    XV_SHOW, TRUE,
    FRAME_LABEL, "VMAP: Variogram Modelling Analysis Package v1.1",
    FRAME_NO_CONFIRM, TRUE,
    XV_WIDTH, INPUT_WIDTH + BACKGROUND_WIDTH,
    XV_HEIGHT, BACKGROUND_HEIGHT,
    FRAME_SHOW_FOOTER, TRUE,
    FRAME_SHOW_RESIZE_CORNER, TRUE,
    NULL);

VariogramPanel = (Panel) xv_create(VariogramFrame, PANEL, NULL);

TheoreticalFrame = (Frame) xv_create(VariogramFrame, FRAME_CMD,
    XV_SHOW, FALSE,
    FRAME_CMD_PUSH_PIN, TRUE,
    XV_LABEL, "Theoretical Variogram Parameters",
    XV_WIDTH, CMDFRAME_WIDTH,
    XV_HEIGHT, CMDFRAME_HEIGHT,
    NULL);

TheoreticalPanel = (Panel) xv_get(TheoreticalFrame, FRAME_CMD_PANEL);

ObservedFrame = (Frame) xv_create(VariogramFrame, FRAME_CMD,
    XV_SHOW, FALSE,
    FRAME_CMD_PUSH_PIN, TRUE,
    XV_LABEL, "Observed Variogram Parameters",
    XV_WIDTH, CMDFRAME_WIDTH,
    XV_HEIGHT, CMDFRAME_HEIGHT,
    NULL);

ObservedPanel = (Panel) xv_get(ObservedFrame, FRAME_CMD_PANEL);

FilterFrame = (Frame) xv_create(VariogramFrame, FRAME_CMD,
    XV_SHOW, FALSE,
FRAME_CMD_PUSH_PNM, TRUE,
XV_LABEL, "Filter Window Parameters",
XV_WIDTH, CMDFRAME_WIDTH,
XV_HEIGHT, CMDFRAME_HEIGHT + 110,
NULL);

FilterPanel = (Panel) xv_get(FilterFrame, FRAME_CMD_PANEL);

BackgroundCanvas = (Canvas) xv_create(VariogramFrame, CANVAS,
XV_X, INPUT_WIDTH,
XV_Y, 0,
XV_WIDTH, BACKGROUND_WIDTH,
XV_HEIGHT, BACKGROUND_HEIGHT,
CANVAS_RETAINED, TRUE,
WIN_COLLAPSE_EXPOSURES, TRUE,
CANVAS_AUTO_SHRINK, TRUE,
CANVAS_AUTO_EXPAND, TRUE,
CANVAS_X_PAINT_WINDOW, TRUE,
CANVAS_REPAINT_PROC, BackgroundRepaintProc,
XV_KEY_DATA, FRAME_KEY, VariogramFrame,
NULL);

VariogramCanvas = (Canvas) xv_create(VariogramFrame, CANVAS,
XV_X, INPUT_WIDTH+80,
XV_Y, 20,
XV_WIDTH, DISPLAY_WIDTH,
XV_HEIGHT, DISPLAY_HEIGHT,
CANVAS_RETAINED, TRUE,
WIN_COLLAPSE_EXPOSURES, TRUE,
CANVAS_AUTO_SHRINK, TRUE,
CANVAS_AUTO_EXPAND, TRUE,
CANVAS_X_PAINT_WINDOW, TRUE,
CANVAS_REPAINT_PROC, VariogramRepaintProc,
XV_KEY_DATA, FRAME_KEY, VariogramFrame,
XV_KEY_DATA, CO_KEY, CO_DEFAULT,
XV_KEY_DATA, CI_KEY, CI_DEFAULT,
XV_KEY_DATA, A_KEY, A_DEFAULT,
XV_KEY_DATA, MODEL_KEY, SPHERICAL,
XV_KEY_DATA, XANGE_KEY, XANGE_DEFAULT,
XV_KEY_DATA, YANGE_KEY, YANGE_DEFAULT,
XV_KEY_DATA, LOG_KEY, 10,
XV_KEY_DATA, TOL_KEY, 5,
XV_KEY_DATA, ANG_KEY, 0,
XV_KEY_DATA, ANG_TOL_KEY, 0,
XV_KEY_DATA, LOG_MAX_KEY, 0,
XV_KEY_DATA, OVAR_TYPE_KEY, O,
XV_KEY_DATA, NEW_THEORETICAL_KEY, FALSE,
XV_KEY_DATA, NEW_OBSERVED_KEY, FALSE,
XV_KEY_DATA, PLANETYPE_KEY, XY_PLANE,
XV_KEY_DATA, DISPLAY_TYPE_KEY, DISPLAY_POINTS,
XV_KEY_DATA, BACKGROUND_KEY, BackgroundCanvas,
NULL);

(void) xv_create(VariogramPanel, PANEL_BUTTON,
PANEL_NOTIFY_PROC, openCmdFrame,
PANEL_LABEL_X, 10,
PANEL_LABEL_Y, 10,
PANEL_LABEL_STRING, "Filter File",
XV_KEY_DATA, FRAME_KEY, FilterFrame,
NULL):
(void) xv_create(VariogramPanel, PANEL_BUTTON,
PANEL_NOTIFY_PROC, OpenCmdFrame,
PANEL_LABEL_X, 10,
PANEL_LABEL_Y, 50,
PANEL_LABEL_STRING, "Theoretical",
XY_KEY_DATA, FRAME_KEY, TheoreticalFrame,
NULL);

(void) xv_create(VariogramPanel, PANEL_BUTTON,
PANEL_NOTIFY_PROC, OpenCmdFrame,
PANEL_LABEL_X, 10,
PANEL_LABEL_Y, 90,
PANEL_LABEL_STRING, "Observed",
XY_KEY_DATA, FRAME_KEY, ObservedFrame,
NULL);

SaveMenu = (Menu) xv_create(XV_NULL, MENU,
MENU_ITEM,
MENU_STRING, "Observed Data",
MENU_NOTIFY_PROC, SaveObserved,
NULL,
MENU_ITEM,
MENU_STRING, "Filter Data",
MENU_NOTIFY_PROC, SaveFilter,
NULL,
XY_KEY_DATA, CANVAS_KEY, VariogramCanvas,
XY_KEY_DATA, FRAME_KEY, FilterFrame,
NULL);

(void) xv_create(VariogramPanel, PANEL_BUTTON,
PANEL_LABEL_X, 10,
PANEL_LABEL_Y, 130,
PANEL_LABEL_STRING, "Save",
PANEL_ITEM_MENU, SaveMenu,
XY_KEY_DATA, CANVAS_KEY, VariogramCanvas,
NULL);

(void) xv_create(VariogramPanel, PANEL_BUTTON,
PANEL_NOTIFY_PROC, Quit,
PANEL_LABEL_X, 10,
PANEL_LABEL_Y, 170,
PANEL_LABEL_STRING, "Quit",
XY_KEY_DATA, FRAME_KEY, VariogramFrame,
NULL);

(void) xv_create(TheoreticalPanel, PANEL_CHOICE,
XY_X, 10,
XY_Y, 10,
PANEL_DISPLAY_LEVEL, PANEL_CURRENT,
PANEL_LABEL_STRING, "Model:"
PANEL_CHOICE_STRING, "None",
"Spherical", "Gaussian", "Exponential", "Linear", NULL,
PANEL_NOTIFY_PROC, VariogramSelected,
XY_KEY_DATA, FRAME_KEY, VariogramFrame,
XY_KEY_DATA, CANVAS_KEY, VariogramCanvas,
NULL);

(void) xv_create(TheoreticalPanel, PANEL_NUMERIC_TEXT,
XY_X, 10,
XY, 40,
 帧控件字符串，"CO:"
 帧控件值，CO_DEFAULT
 帧控件最大值，1000
 帧控件最小值，0
 帧控件显示长度，6
 帧控件通知函数，TheoreticalModification
 XY_KEY_DATA，CANVAS_KEY，VariogramCanvas,
 XY_KEY_DATA，KEY_KEY，CO_KEY,
 NULL);

(void) xv_create(TheoreticalPanel, PANEL_NUMERIC_TEXT,
 XY, 10,
 XY, 70,
 帧控件字符串，"C1:"
 帧控件值，C1_DEFAULT
 帧控件最大值，1000
 帧控件最小值，0
 帧控件显示长度，6
 帧控件通知函数，TheoreticalModification
 XY_KEY_DATA，CANVAS_KEY，VariogramCanvas,
 XY_KEY_DATA，KEY_KEY，C1_KEY,
 NULL);

(void) xv_create(TheoreticalPanel, PANEL_NUMERIC_TEXT,
 XY, 10,
 XY, 100,
 帧控件字符串，"a:"
 帧控件值，A_DEFAULT
 帧控件最大值，1000
 帧控件最小值，0
 帧控件显示长度，6
 帧控件通知函数，TheoreticalModification
 XY_KEY_DATA，CANVAS_KEY，VariogramCanvas,
 XY_KEY_DATA，KEY_KEY，A_KEY,
 NULL);

(void) xv_create(TheoreticalPanel, PANEL_NUMERIC_TEXT,
 XY, 10,
 XY, 130,
 帧控件字符串，"Xrange:"
 帧控件值，X_RANGE_DEFAULT
 帧控件最大值，DISPLAY_WIDTH
 帧控件最小值，1
 帧控件显示长度，6
 帧控件通知函数，TheoreticalModification
 XY_KEY_DATA，CANVAS_KEY，VariogramCanvas,
 XY_KEY_DATA，KEY_KEY，X_RANGE_KEY,
 NULL);

(void) xv_create(TheoreticalPanel, PANEL_NUMERIC_TEXT,
 XY, 10,
 XY, 160,
 帧控件字符串，"Yrange:"
 帧控件值，Y_RANGE_DEFAULT
 帧控件最大值，DISPLAY_HEIGHT
 帧控件最小值，1
 帧控件显示长度，6
 帧控件通知函数，TheoreticalModification
 XY_KEY_DATA，CANVAS_KEY，VariogramCanvas,
(void) xv_create(ObservedPanel, PANEL_CHOICE, 
    XV_X, 10, 
    XV_Y, 10, 
    PANEL_DISPLAY_LEVEL, PANEL_CURRENT, 
    PANEL_LABEL_STRING, "Model:", 
    PANEL_CHOICE_STRINGS, "Unidirectional", "Directional", NULL, 
    PANEL_NOTIFY_PROC, ObservedSelected, 
    XV_KEY_DATA, CANVAS_KEY, VariogramCanvas, 
    NULL);

(void) xv_create(ObservedPanel, PANEL_CHOICE, 
    XV_X, 10, 
    XV_Y, 40, 
    PANEL_DISPLAY_LEVEL, PANEL_CURRENT, 
    PANEL_LABEL_STRING, "Display:", 
    PANEL_CHOICE_STRINGS, "Points", "Points and Lines", "Points and Dashed Lines", NULL, 
    PANEL_NOTIFY_PROC, ObservedDisplaySelected, 
    XV_KEY_DATA, CANVAS_KEY, VariogramCanvas, 
    NULL);

(void) xv_create(ObservedPanel, PANEL_NUMERIC_TEXT, 
    XV_X, 10, 
    XV_Y, 70, 
    PANEL_LABEL_STRING, "Lag:", 
    PANEL_VALUE, 10, 
    PANEL_NOTIFY_PROC, ObservedModification, 
    PANEL_VALUE_DISPLAY_LENGTH, 6, 
    XV_KEY_DATA, CANVAS_KEY, VariogramCanvas, 
    PANEL_MAX_VALUE, 1000, 
    PANEL_MIN_VALUE, 0, 
    XV_KEY_DATA, KEY_KEY, LAG_KEY, 
    XV_KEY_DATA, LAG_KEY, 10, 
    NULL);

(void) xv_create(ObservedPanel, PANEL_NUMERIC_TEXT, 
    XV_X, 10, 
    XV_Y, 100, 
    PANEL_LABEL_STRING, "Tolerance:", 
    PANEL_VALUE, 5, 
    PANEL_VALUE_DISPLAY_LENGTH, 6, 
    PANEL_NOTIFY_PROC, ObservedModification, 
    PANEL_MAX_VALUE, 1000, 
    PANEL_MIN_VALUE, 0, 
    XV_KEY_DATA, CANVAS_KEY, VariogramCanvas, 
    XV_KEY_DATA, KEY_KEY, TOL_KEY, 
    XV_KEY_DATA, TOL_KEY, 5, 
    NULL);

(void) xv_create(ObservedPanel, PANEL_NUMERIC_TEXT, 
    XV_X, 10, 
    XV_Y, 130, 
    PANEL_LABEL_STRING, "Angular Direction:", 
    PANEL_VALUE, 0,
(void) xv_create(ObservedPanel, PANEL_NUMERIC_TEXT, 
   XV_X, 10, 
   XV_Y, 150, 
   PANEL_LABEL_STRING, "Angular Tolerance: ", 
   PANEL_VALU, 0, 
   PANEL_VALUE_DISPLAY_LENGTH, 6, 
   PANEL_NOTIFY_PROC, ObservedModification, 
   PANEL_MAX_VALUE, 360, 
   PANEL_MIN_VALUE, 0, 
   XV_KEY_DATA, CANVAS_KEY, VariogramCanvas, 
   XV_KEY_DATA, KEY_KEY, ANG_KEY, 
   XV_KEY_DATA, ANG_TOL_KEY, 0, 
   NULL);

(void) xv_create(ObservedPanel, PANEL_NUMERIC_TEXT, 
   XV_X, 10, 
   XV_Y, 190, 
   PANEL_LABEL_STRING, "Lag Max: ", 
   PANEL_VALUE, 0, 
   PANEL_VALUE_DISPLAY_LENGTH, 6, 
   PANEL_MAX_VALUE, 360, 
   PANEL_MIN_VALUE, 0, 
   PANEL_NOTIFY_PROC, ObservedModification, 
   XV_KEY_DATA, CANVAS_KEY, VariogramCanvas, 
   XV_KEY_DATA, KEY_KEY, ANG_LAG_MAX_KEY, 
   XV_KEY_DATA, LAG_MAX_KEY, 0, 
   NULL);

(void) xv_create(FilterPanel, PANEL_TEXT, 
   XV_X, 10, 
   XV_Y, 10, 
   PANEL_LABEL_STRING, "Input file ", 
   PANEL_NOTIFY_PROC, LoadObservedFile, 
   PANEL_VALUE, ", 
   PANEL_NOTIFY_LEVEL, PANEL_SPECIFIED, 
   PANEL_LARGE_STRING, "/a\t\t.33", 
   PANEL_VALUE DISPLAY LENGTH, 25, 
   PANEL_VALUE STORED LENGTH, 200, 
   XV_KEY_DATA, FRM_KEY, FilterFrame, 
   XV_KEY_DATA, BASE_FRM_KEY, FilterFrame, 
   NULL);

(void) xv_create(FilterPanel, PANEL_CHOICE, 
   XV_X, 10, 
   XV_Y, 130, 
   PANEL_DISPLAY_LEVEL, PANEL_CURRENT, 
   PANEL_LABEL_STRING, "Primary Plane: ", 
   PANEL_CHOICE_STRINGS, 
   "Horizontal/ Spatial (X,Y) ", 
   "Vertical (Y,Z) ") ;
"Temporal (Z,T)",
NULL,
PANEL_NOTIFY_PROC, ObservePlaneSelected,
XY_KEY_DATA, FRAME_KEY, FilterFrame,
NULL);

(void) xv_create(FilterPanel, PANEL_NUMERIC_TEXT,
    XY_X, 10,
    XY_Y, 160,
    PANEL_LABEL_STRING, "Horiz X:",
    PANEL_VALUE, 0,
    PANEL_VALUE_DISPLAY_LENGTH, 4,
    PANEL_MAX_VALUE, 1000,
    PANEL_MIN_VALUE, 0,
    XY_KEY_DATA, FRAME_KEY, FilterFrame,
    XY_KEY_DATA, KEY_KEY, HVALL_KEY,
    PANEL_NOTIFY_PROC, ModifyFilterWindow,
    NULL);

(void) xv_create(FilterPanel, PANEL_NUMERIC_TEXT,
    XY_X, 160,
    XY_Y, 160,
    PANEL_LABEL_STRING, "Horiz Y:",
    PANEL_VALUE, 0,
    PANEL_VALUE_DISPLAY_LENGTH, 4,
    PANEL_MAX_VALUE, 1000,
    PANEL_MIN_VALUE, 0,
    XY_KEY_DATA, FRAME_KEY, FilterFrame,
    XY_KEY_DATA, KEY_KEY, HVALL_KEY,
    PANEL_NOTIFY_PROC, ModifyFilterWindow,
    NULL);

(void) xv_create(FilterPanel, PANEL_NUMERIC_TEXT,
    XY_X, 10,
    XY_Y, 190,
    PANEL_LABEL_STRING, "Horizontal Radius:",
    PANEL_VALUE, 0,
    PANEL_VALUE_DISPLAY_LENGTH, 6,
    PANEL_MAX_VALUE, 1000,
    PANEL_MIN_VALUE, 0,
    XY_KEY_DATA, FRAME_KEY, FilterFrame,
    XY_KEY_DATA, KEY_KEY, HVAL_KEY,
    PANEL_NOTIFY_PROC, ModifyFilterWindow,
    NULL);

(void) xv_create(FilterPanel, PANEL_NUMERIC_TEXT,
    XY_X, 10,
    XY_Y, 220,
    PANEL_LABEL_STRING, "Vertical Value:",
    PANEL_VALUE, 0,
    PANEL_VALUE_DISPLAY_LENGTH, 6,
    PANEL_MAX_VALUE, 1000,
    PANEL_MIN_VALUE, 0,
    XY_KEY_DATA, FRAME_KEY, FilterFrame,
    XY_KEY_DATA, KEY_KEY, VVAL_KEY,
    PANEL_NOTIFY_PROC, ModifyFilterWindow,
    NULL);

(void) xv_create(FilterPanel, PANEL_NUMERIC_TEXT,
    XY_X, 10,
(void) xv_create(FilterPanel, PANEL_NUMERIC_TEXT,
    XV_X, 10,
    XV_Y, 280,
    PANEL_LABEL_STRING, "Vertical Tolerance:",
    PANEL_VALUE, 0,
    PANEL_MAX_VALUE, 1000,
    PANEL_MIN_VALUE, 0,
    XV_KEY_DATA, FRAME_KEY, FilterFrame,
    XV_KEY_DATA, KEY_KEY, VTOG_KEY,
    PANEL_NOTIFY_PROC, ModifyFilterWindow,
    NULL);

(void) xv_create(FilterPanel, PANEL_NUMERIC_TEXT,
    XV_X, 10,
    XV_Y, 260,
    PANEL_LABEL_STRING, "Temporal Value:",
    PANEL_VALUE, 0,
    PANEL_MAX_VALUE, 1000,
    PANEL_MIN_VALUE, 0,
    XV_KEY_DATA, FRAME_KEY, FilterFrame,
    XV_KEY_DATA, KEY_KEY, TVTOL_KEY,
    PANEL_NOTIFY_PROC, ModifyFilterWindow,
    NULL);

(void) xv_create(FilterPanel, PANEL_NUMERIC_TEXT,
    XV_X, 10,
    XV_Y, 310,
    PANEL_LABEL_STRING, "Temporal Tolerance:",
    PANEL_VALUE, 0,
    PANEL_MAX_VALUE, 1000,
    PANEL_MIN_VALUE, 0,
    XV_KEY_DATA, FRAME_KEY, FilterFrame,
    XV_KEY_DATA, KEY_KEY, TTOL_KEY,
    PANEL_NOTIFY_PROC, ModifyFilterWindow,
    NULL);

(void) xv_create(FilterPanel, PANEL_BUTTON,
    XV_X, 10,
    XV_Y, 340,
    PANEL_LABEL_STRING, "Process Selected Filters",
    PANEL_NOTIFY_PROC, ProcessFilterWindow,
    XV_KEY_DATA, FRAME_KEY, FilterFrame,
    NULL);
/*
 * vmap.c
 *
 * Copyright (C) 1993 eric j. miller
 *
 * Permission to use, copy, and distribute this software and its
 * documentation for any purpose with or without fee is hereby granted,
 * provided that the above copyright notice appear in all copies and
 * that both that copyright notice and this permission notice appear
 * in supporting documentation.
 *
 * Permission to modify the software is granted, but not the right to
 * distribute the modified code. Modifications are to be distributed
 * as patches to released version.
 *
 * This software is provided "as is" without express or implied warranty.
 *
 * AUTHOR: Eric J. Miller
 * emiller@oclc.org
 * emiller@cis.ohio-state.edu
 *
 * DATE:  06/01/93
 *
 */

#include <stdio.h>
#include <math.h>

#include "spMatrix.h"
#include "spDefs.h"

#include <xview/xview.h>
#include <xview/frame.h>
#include <xview/svimage.h>
#include <xview/panel.h>
#include <xview/scrollbar.h>
#include <xview/notice.h>
#include <xview/openmens.h>
#include <xview/rectlist.h>
#include <xview/xv_xrect.h>
#include <xview/font.h>
#include <xview/notify.h>
#include <xview/textw.h>

#include <X/x.h>
#include <X/Xlib.h> /* Using Xlib graphics well */

#include "variogram.h"

int values;
float valarr[1000];

int opta = 0;
XPoint opoints[1000];
double oymax = -1.0, oymin = -1.0;
double oxmax = -1.0, oxmin = -1.0;

int optb = 0;
XPoint tpoints[1000];
typedef struct outputdata
{
    int Nh, h;
    float avglag;
    float variovalue;
} OutputData;

OutputData odata[100];
int odatas;

main(argc, argv)
int argc;
char *argv[];
{
    extern int Quit();

    extern void SaveObserved();
    extern void SaveFilter();
    extern void VariogramRepaintProc();
    extern void BackgroundRepaintProc();
    extern void VariogramSelected();
    extern void TheoreticalModification();
    extern void ObservedModification();
    extern void ObservedDisplaySelected();
    extern void ObservedSelected();
    extern void ObservedPlaneSelected();
    extern void ModifyFilterWindow();
    extern void ProcessFilterWindow();

    extern int ProcessInputFile();
    extern int OpenCmdFrame();

    extern int LoadObservedFile();

    Frame TheoreticalFrame;
    Frame ObservedFrame;
    Panel TheoreticalPanel;
    Panel ObservedPanel;

    Menu SaveMenu;

    Display edpy;

    FILE *fin;

    unsigned int line_width = 2;
    int line_style = LineSolid;
    int cap_style = CapButt;
    int join_style = JoinMiter;

    int i;

    Debug = FALSE;

    xv_init(XV_INIT_ARGC_PTR_ARGV, &argc, argv, NULL);

    if (argc < 2)
    {
printf("No input file specified...\n");
}
else if ((fin = (FILE *)fopen(argv[1], "r")) == (FILE *)NULL)
{
    printf("Unable to open file %s\n", argv[1]);
}

if (fin != (FILE *)NULL)
{
    fscanf(fin, "%d", &values);
    for (i = 0; i < values; i++)
    {
        fscanf(fin, "%f", &valarr[i]);
    }
}

#include "variogram.xview"

/* Initialize global GC structure */
dpy = (Display *)X_DISPLAY_FROM_WINDOW(canvas_paint_window(VariogramCanvas));

tgcvalues.foreground = BlackPixel(dpy, DefaultScreen(dpy));
tgcvalues.background = WhitePixel(dpy, DefaultScreen(dpy));
tgcvalues.function = GXor;
tgcvalues.graphics_exposures = True;

tgc = XCreateGC(dpy, RootWindow(dpy, DefaultScreen(dpy)),
              (GCForeground | GCBACKGROUND | GCFUNCTION | GCGraphicsExposures),
              &tgcvalues);
XSetLineAttributes(dpy, tgc, line_width, line_style, cap_style, join_style);

ogcvalues.foreground = BlackPixel(dpy, DefaultScreen(dpy));
ogcvalues.background = WhitePixel(dpy, DefaultScreen(dpy));
ogcvalues.function = GXor;
ogcvalues.graphics_exposures = True;

ogc = XCreateGC(dpy, RootWindow(dpy, DefaultScreen(dpy)),
                (GCForeground | GCBACKGROUND | GCFUNCTION | GCGraphicsExposures),
              &ogcvalues);

line_width = 1;
XSetLineAttributes(dpy, ogc, line_width, line_style, cap_style, join_style);

xv_main_loop(VariogramFrame);
}

void BackgroundRepaintProc(canvas, pw, dpy, xwin, xrects)
Canvas canvas;
Xv_WWindow pw;
Display *dpy;
Window xwin;
Xv_rectangle *xrects;
{
    int yhash;               /* Number of y hash marks */
    int xhash;               /* Number of X */

float ydiff, xdiff;
float yincr, xincr;
float incr;

int x, y;

char cincr[40];

// trivial check */
if ((oymax < 0.0) || (oxmax < 0.0))
    return;

// Clear window */
XClearWindow(dpy, xwin);

// Display y axis information */
yhash = 10;
ydiff = (float)oymax - (float)oymin;
yincr = (float) (float)ydiff/(float)yhash;
for (y = 18 + DISPLAY_HEIGHT, x = 20, incr = oymin;
    y >= 20;
    y += (int) DISPLAY_HEIGHT/(float)yhash, incr += yincr)
    { /* Draw hash mark */
        XDrawString(dpy, xwin, ogc, 55, y, "\_", 1);
    /* Draw increment string */
        sprintf(cincr, "%4.2f", incr);
        XDrawString(dpy, xwin, ogc, x, y, cincr, strlen(cincr));
    } /* Display x axis information */
xhash = 10;
xdiff = (float)oxmax - (float)oxmin;
xincr = (float) (float)xdiff/(float)xhash;
for (y = DISPLAY_HEIGHT + 40, x = 55, incr = oxmin;
    x <= DISPLAY_WIDTH;
    x += (int) DISPLAY_WIDTH/(float)xhash, incr += xincr)
    { /* Draw hash mark */
        XDrawString(dpy, xwin, ogc, x, DISPLAY_HEIGHT + 25, "\|", 1);
    /* Draw increment string */
        sprintf(cincr, "%4.2f", incr);
        XDrawString(dpy, xwin, ogc, x, y, cincr, strlen(cincr));
    }

XFlush(dpy);
void VarigramRepaintProc(canvas, pw, dpy, xwin, x rects)
Canvas canva;
Xv_Window pw;
Display *dpy;
Window xwin;
Xvrectlist x rects;
{
    extern void BackgroundRepaintProc();
    extern void SphericalModel();
    extern void ExponentialModel();
    extern void GaussianModel();

    Canvas bg canvas;
    Xv_Window bg pw;
    Display *bg dpy;
    Window bg xwin;

    Frame frame = xv_get(canvas, XV_KEY_DATA, FRAME KEY);
    int c0 = (int) xv_get(canvas, XV_KEY_DATA, CO_KEY);
    int c1 = (int) xv_get(canvas, XV_KEY_DATA, C1_KEY);
    int a = (int) xv_get(canvas, XV_KEY_DATA, A KEY);
    int model = (int) xv_get(canvas, XV_KEY_DATA, MODEL KEY);
    int x range = (int) xv_get(canvas, XV_KEY_DATA, XRANGE KEY);
    int y range = (int) xv_get(canvas, XV_KEY_DATA, YRANGE KEY);
    int newt = (int) xv_get(canvas, XV_KEY_DATA, NEW THEORETICAL KEY);
    int newo = (int) xv_get(canvas, XV_KEY_DATA, NEW OBSERVED KEY);

    int o type = (int) xv_get(canvas, XV_KEY_DATA, OVAR TYPE KEY);
    int lag = (int) xv_get(canvas, XV KEY_DATA, LAG KEY);
    int tol = (int) xv_get(canvas, XV KEY_DATA, TOL KEY);
    int ang = (int) xv_get(canvas, XV KEY DATA, ANG KEY);
    int angtol = (int) xv_get(canvas, XV KEY_DATA, ANG TOL KEY);
    int hmax = (int) xv_get(canvas, XV KEY_DATA, LAG MAX KEY);
    int dtype = (int) xv_get(canvas, XV KEY_DATA, DISPLAY TYPE KEY);

    int i;

    // Draw theoretical */
    if ((newt) && (tpts > 0))
        XDrawLines(dpy, xwin, tgc, tpoints, tpts, CoordModeOrigin);
    // Draw observed */
    if ((newo) && (opts > 0))
    {
        for (i = 0; i < opts; i++)
        {
            XFILL Rectangle(dpy, xwin, gc,
            opoints[i].x-3, opoints[i].y-3,
            6, 6);
        }

        if (dtype == DISPLAY_LINES)
            XDrawLines(dpy, xwin, gc, opoints, opts, CoordModeOrigin);
        else if (dtype == DISPLAY_LINES)
        {
/* set gc struct for dashed lines and redisplay */

IDrawLines(dpy, xwin, ogc, opoints, opts, CoordNodeOrigin);
}

if (newo)
{
    ProcessVariogram(otype, lag, hmax, tol, ang, angtol);
    for (i = 0; i < opts; i++)
    {
        XPolygon(dpy, xwin, ogc, opoints[i].x-3, opoints[i].y-3,
                 6, 6);
    }
    if (dtype == DISPLAY_LINES)
        IDrawLines(dpy, xwin, ogc, opoints, opts, CoordNodeOrigin);
    else if (dtype == DISPLAY_DLINEs)
    {
        /* set gc struct for dashed lines and redisplay */
        IDrawLines(dpy, xwin, ogc, opoints, opts, CoordNodeOrigin);
    }
}

if (new)
{
    switch(model)
    {
    case RHOE:
        tpts = 0;
        break;
    case SPHERICAL:
        SphericalModel(c0, ci, a, xrange, yrange);
        IDrawLines(dpy, xwin, tgc, tpoints, tpts, CoordNodeOrigin);
        break;
    case GAUSSIAN:
        GaussianModel(c0, ci, a, xrange, yrange);
        IDrawLines(dpy, xwin, tgc, tpoints, tpts, CoordNodeOrigin);
        break;
    case EXPOENTIAL:
        ExponentialModel(c0, ci, a, xrange, yrange);
        IDrawLines(dpy, xwin, tgc, tpoints, tpts, CoordNodeOrigin);
        break;
    case LINEAR:
        break;
    default:
        break;
    }
}

/* reset new flag */

xv_set(canvas, XV_KEY_DATA, NEW_THEORETICAL_KEY, FALSE, NULL);
xv_set(canvas, XV_KEY_DATA, NEW_OBSERVED_KEY, FALSE, NULL);

/* assign background canvas information and repaint background */

bgcanvas = (Canvas) xv_get(canvas, XV_KEY_DATA, BACKGROUND_KEY);
bgpw = canvas_paint_window(bgcanvas);
bgdy = (Display *) xv_get(bgpw, XV_DISPLAY);
bgxwin = (Window) xv_get(bgpw, XV_XID);

BackgroundRepaintProc(bgcanvas, bgpw, bgdy, bgxwin, NULL);
XFlush(dpy);
}

void SphericalModel(c0, c1, a, xrange, yrange)
int c0, c1, a;
int xrange, yrange;
{
  double hovera;
  int h;
  double Th; /* Funtion of h value */
  int nx, ny;

  /* Initialize global model point counter */
  tpts = 0;

  /* Normalize points to display grid */
  nx = (int) (DISPLAY_WIDTH/(double)xrange);
  ny = (int) (DISPLAY_HEIGHT/(double)yrange);

  /* Loop through model of points based on user input resolution */
  for (h = 0; h < xrange; h++)
    {
      if (h <= a)
        {
          hovera = (double) (double)h/(double)a;

          fh = (double) (1.5 * hovera - 0.5 * (hovera + hovera + hovera));
        }
      else
        {
          fh = 1.0;
        }

      tpoints[tpts].x = (int) (nx*h);
      tpoints[tpts].y = DISPLAY_HEIGHT - (int) (ny*fh);

      tpts++;
    }
}

void GaussianModel(c0, c1, a, xrange, yrange)
int c0, c1, a;
int xrange, yrange;
{ double hov, h;
    int h;  /* h value */
    double fh;  /* Function of h value */

    int nx, ny;
    /* Initialize global model point counter */
    tpts = 0;
    /* Normalize points to display grid */
    nx = (int) (DISPLAY_WIDTH/(double)xrange);
    ny = (int) (DISPLAY_HEIGHT/(double)yrange);
    /* Loop through model of points based on user input resolution */
    for (h = 0; h < xrange; h++)
    {
        fh = (double) (1.0 - exp((double)(-3*(h*h))/(double)(aw*a)));
        tpoints[tpts].x = (int) (nx*h);
        tpoints[tpts].y = DISPLAY_HEIGHT - (int) (ny*fh);
        tpts++;
    }
}

void ExponentialModel(c0, c1, a, xrange, yrange)
int c0, c1, a;
int xrange, yrange;
{
    double hov, h;
    int h;  /* h value */
    double fh;  /* Function of h value */

    int nx, ny;
    /* Initialize global model point counter */
    tpts = 0;
    /* Normalize points to display grid */
    nx = (int) (DISPLAY_WIDTH/(double)xrange);
    ny = (int) (DISPLAY_HEIGHT/(double)yrange);
    /* Loop through model of points based on user input resolution */
    for (h = 0; h < xrange; h++)
    {
        fh = (double) (1.0 - exp((double)(-3*(h*h))/(double)(aw*a)));
        tpoints[tpts].x = (int) (nx*h);
        tpoints[tpts].y = DISPLAY_HEIGHT - (int) (ny*fh);
        tpts++;
    }
int OpenCmdFrame(item, event)
Panel_item item;
Event *event;
{
    Frame frame = (Frame) xv_get(item, XV_KEY_DATA, FRAME_KEY);
    xv_set(frame, FRAME_CMD_PUSH_PIN_IN, TRUE, NULL);
    xv_set(frame, XV_SHOW, TRUE, NULL);
}

void VariogramSelected(item, value, event)
Panel_item item;
int value;
Event *event;
{
    extern void VariogramRepaintProc(); /* Function declaration */
    Canvas canvas = (Canvas) xv_get(item, XV_KEY_DATA, CANVAS_KEY);
    Xv_Window pw;
    Display *dpy;
    Window xwin;
    xv_set(canvas, XV_KEY_DATA, MODEL_KEY, value, NULL);
    xv_set(canvas, XV_KEY_DATA, NEW_THEORETICAL_KEY, TRUE, NULL);
    pw = canvas_paint_window(canvas);
    dpy = (Display *) xv_get(pw, XV_DISPLAY);
    xwin = (Window) xv_get(pw, XV_XID);
    VariogramRepaintProc(canvas, pw, dpy, xwin, NULL);
}

void ObservedSelected(item, value, event)
Panel_item item;
int value;
Event *event;
{
    Canvas canvas = (Canvas) xv_get(item, XV_KEY_DATA, CANVAS_KEY);
    xv_set(canvas, XV_KEY_DATA, OVAR_TYPE_KEY, value, NULL);
}

void ObservedDisplaySelected(item, value, event)
Panel_item item;
int value;
Event *event;
{
    extern void VariogramRepaintProc(); /* Function declaration */
    Xv_Window pw;
    Display *dpy;
    Window xwin;
Canvas canvas = (Canvas) xv_get(item, XV_KEY_DATA, CANVAS_KEY);

xv_set(canvas, XV_KEY_DATA, DISPLAY_TYPE_KEY, value, NULL);
xv_set(canvas, XV_KEY_DATA, NEW_OBSERVED_KEY, TRUE, NULL);

/* set key to true but point count to false for single display */

opts = -1;

pw = canvas_paint_window(canvas);
dpy = (Display *) xv_get(pw, XV_DISPLAY);
xwin = (Window) xv_get(pw, XV_XID);

/* Clear window (at this point unknown previous selection of display) */

XClearWindow(dpy, xwin);
XFlush(dpy);

VariogramDepaintProc(canvas, pw, dpy, xwin, NULL);
}

void TheoreticalModification(item, event)
Panel_item item;
Event *event;
{
    extern void VariogramDepaintProc(); /* Function declaration */

    Canvas canvas = (Canvas) xv_get(item, XV_KEY_DATA, CANVAS_KEY);
    int key = (int) xv_get(item, XV_KEY_DATA, KEY_KEY);
    int value = (int) xv_get(item, PANEL_VALUE);

    Xv_Window pw;
    Display *dpy;
    window xwin;

    xv_set(canvas, XV_KEY_DATA, key, value, NULL);
    xv_set(canvas, XV_KEY_DATA, NEW_THEORETICAL_KEY, TRUE, NULL);

    pw = canvas_paint_window(canvas);
dpy = (Display *) xv_get(pw, XV_DISPLAY);
xwin = (Window) xv_get(pw, XV_XID);

    VariogramDepaintProc(canvas, pw, dpy, xwin, NULL);
}

void ObservedModification(item, event)
Panel_item item;
Event *event;
{
    extern void VariogramDepaintProc(); /* Function declaration */

    Canvas canvas = (Canvas) xv_get(item, XV_KEY_DATA, CANVAS_KEY);
    int key = (int) xv_get(item, XV_KEY_DATA, KEY_KEY);
    int value = (int) xv_get(item, PANEL_VALUE);

    Xv_Window pw;
    Display *dpy;
    Window xwin;
xv_set(canvas, XV_KEY_DATA, key, value, NULL);
xv_set(canvas, XV_KEY_DATA, XV_OBSERVED_KEY, TRUE, NULL);

pw = canvas_paint_window(canvas);
dpy = (Display *) xv_get(pw, XV_DISPLAY);
xwin = (Window) xv_get(pw, XV_XID);

VariogramRepaintProc(canvas, pw, dpy, xwin, NULL);
}

int ProcessInputFile(FILE *fin)
FILE *fin;
{
    char buf[1000];
    int i, j;
    int values;
    int spError;
    int Count = 0;
    float tfx, tfy, tfz, tv;
    int tdt;
    int dc;
    char key[10];
    char line[100];

double minx, miny, minz, maxx, maxy, maxz;
    int mint, maxt;
    printf("Variogram Processing\n");

    minx = miny = minz = 10000.0;
    maxx = maxy = maxz = -10000.0;
    mint = 100000;
    maxt = -100000;

    /* Assign global number of distances */
    /* First value is point of interest */
    for (i = 0; !(feof(fin)); i++)
    
    {
        fscanf(fin, "%d%f%f%f%f", key, &tx, &ty, &tz, &t);
        strcpy(gvt[i].wid, key);
        gvt[i].x = (double) tx;
        gvt[i].y = (double) ty;
        gvt[i].z = (double) tz;
    }


gvt[i].t = (int) tdt;
gvt[i].v = (double) tv;

minx = MIN(minx, tfx);
miny = MIN(miny, tfy);
minz = MIN(minz, tfz);
mint = MIN(mint, tdt);

maxx = MAX(maxx, tfx);
maxy = MAX(maxy, tfy);
maxz = MAX(maxz, tfz);
maxt = MAX(maxt, tdt);
}

gvts = 1;

/* Set-up filter frame for type of processing */

sprintf(line,
        "<X(tf)> min: %4.2f max: %4.2f",
        minx, maxx);

(void) xv_create(FilterPanel, PANEL_MESSAGE,
                   XV_X, 10,
                   XV_Y, 30,
                   PANEL_LABEL_STRING, (char *) line,
                   NULL);

sprintf(line,
        "<Y(tf)> min: %4.2f max: %4.2f",
        miny, maxy);

(void) xv_create(FilterPanel, PANEL_MESSAGE,
                   XV_X, 10,
                   XV_Y, 50,
                   PANEL_LABEL_STRING, (char *) line,
                   NULL);

sprintf(line,
        "<Z(ft)> min: %4.2f max: %4.2f",
        minz, maxz);

(void) xv_create(FilterPanel, PANEL_MESSAGE,
                   XV_X, 10,
                   XV_Y, 70,
                   PANEL_LABEL_STRING, (char *) line,
                   NULL);

sprintf(line,
        "<T(days)> min: %d max: %d",
        mint, maxt);

(void) xv_create(FilterPanel, PANEL_MESSAGE,
                   XV_X, 10,
                   XV_Y, 90,
                   PANEL_LABEL_STRING, (char *) line,
                   NULL);

return TRUE;
void ProcessDistance(plane)
{
    int mperror;
    RealNumber *pElement;
    RealNumber *pInitInfo;
    RealNumber *pValue[1000];

double Real;
dPoint pt;
double Distance;
double d;
int dc;
int i, j;

    /* Create distance matrix */
    if (DMatrix == (char *) NULL)
        DMatrix = spCreate(gdte, 0, &mperror);
    if (DMatrix == (char *) NULL)
    {
        printf("Insufficient memory to allocate distance matrix\n");
        return;
    }

    /* Calculate distance matrix */

    printf("Processing Distance Matrix...\n");
d = 0.0;
dc = 0;
for (i = 0; i < gdts; i++)
{
    for (j = 0; j < gdts; j++)
    {
        /* assign distance values to remaining cells */
        switch (plane)
        {
            case XY_PLANE:
            pt.x = gdtx[i].x - gdtx[j].x;
            pt.y = gdtx[i].y - gdtx[j].y;
            Distance = (double) sqrt(pt.x*pt.x+pt.y*pt.y);
            break;
            case YZ_PLANE:
                pt.y = gdtx[i].y - gdtx[j].y;
                pt.x = gdtx[i].x - gdtx[j].x;
                Distance = (double) sqrt(pt.x*pt.x+pt.y*pt.y);
                break;
            case XZ_PLANE:
                pt.x = gdtx[i].x - gdtx[j].x;
                pt.y = gdtx[i].y - gdtx[j].y;
                Distance = (double) sqrt(pt.x*pt.x+pt.y*pt.y);
                break;
            case Z_PLANE:
                pt.y = gdtx[i].y - gdtx[j].y;
                pt.x = gdtx[i].x - gdtx[j].x;
                Distance = (double) sqrt(pt.x*pt.x+pt.y*pt.y);
                break;
        }
    }
}
}
case ZT_PLANE:
    pt.x = gd[i].x - gd[j].x;
    pt.t = gd[i].t - gd[j].t;
    Distance = (double) sqrt(pt.x*pt.x+pt.t*pt.t);
    break;
default:
    Distance = (double) 0.0;
    break;
}

d += Distance;
dc++;

/* Assign value in distance table */
pElement = spGetElement(DMatrix, i+1, j+1);
*pElement = Distance;

printf("Average lag distance: %f\n", (float)((double)d/(double)dc));

int ProcessVariogram(omni, lag, hmax, tol, ang, angtol)
int omni;
int lag;
int hmax;
int tol;
int ang, angtol;
{
    extern int PtInPoly();    /* Function declaration */
    int spError;
    int i,j;                /* lcv */
    int h;                 /* Absolute distance for pairwise comparison */
    int Nh;                /* Number of separational pairs */
    double variosum;        /* Summation of variogram pair values */
    double variovalue;      /* Final value of summed pairs */
    int ainc;               /* Angular increment */
    double Distance;
    double maxDistance = 0.0;
    double avglag = 0.0;

dPoint pt1,pt2;        /* Points used for tolerance calculation */
dPoint dpoints[1000];
int dpts;
Polygon poly;
RealNumber spElement;
double rang;       /* angle in radians */
double sinrang, cosrang;    /* sin/cos radians angle */
char outfile[100];
int valid = FALSE;
double xmin, xmax;
double ymin, ymax;

if (Debug)
    spPrint(DMatrix, 0, 1, 0);

/* Loop through directional (or omnidirectional if angtol = 180) */
/* loop through each lag */
variosum = 0.0;
opts = 0;
dpts = 0;

/* Assign min and max values for display */
xmin = ymin = 1000000.0;
xmax = ymax = 0.0;

odatas = 0;

for (h = 0, Eh = 0; h <= hmax; h++=lag)
{
    printf("Processing lag: Xd += Xd\n", h, tol);
    /* Loop through and check for distance */
    for (i = 0; i < gdts; i++)
    {
        for (j = 0; j < gdts; j++)
        {
            pElement = spGetElement(DMatrix, i+j, j+j);
            Distance = (double)*pElement;
            /* Check for lag tolerance */
            if ((Distance >= (double)((double)h-(double)tol)) &&
                (Distance <= (double)((double)h+(double)tol)) && (i==j))
            {
                valid = TRUE;
                /* Check if omnidirectional calculation or directional */
                if (omni != OMNIDIRECTIONAL)
                {
                    /* Directional */
                    valid = FALSE;
/ Check for angular tolerance */
pt1.x = gdt[i].x;
pt1.y = gdt[i].y;
pt2.x = gdt[j].x;
pt2.y = gdt[j].y;

/* Check for point in arc */
poly.pt[0].x = pt1.x;
poly.pt[0].y = pt1.y;
rang = (double) DTOR(ang+angtol);
sinrang = sin(rang);
cosrang = cos(rang);
poly.pt[1].x = pt1.x + cosrang*(2*ht+tol);
poly.pt[1].y = pt1.y + sinrang*(2*ht+tol);
rang = (double) DTOR(ang-angtol);
sinrang = sin(rang);
cosrang = cos(rang);
poly.pt[2].x = pt1.x + cosrang*(2*ht+tol);
poly.pt[2].y = pt1.y + sinrang*(2*ht+tol);
poly.nums = 3;
if (PtInPoly(&poly, &pt2))
  valid = TRUE;
}

if (valid)
{
  /* Finally found a valid variogram thing */
  /* Increment number of variogram values found */
  Nh++;
  avglag += Distance;
  variosum += (double)
((gdt[i].v-gdt[j].v)*(gdt[i].v-gdt[j].v));
  /* reinit */
  valid = FALSE;
}

/* Check to see if there are any values worth adding */
if (Nh > 0)
{
  variovalue = (double) (1.0/(double)(2.0*Nh))*variosum;
  avglag /= Nh;
odata[odatas].H = Hh;
odata[odatas].h = h;
odata[odatas].avglag = (float) avglag;
odata[odatas].variovalue = (float) variovalue;

odatas++;

printf("%d %d %f %f\n", Hh, h, (float) avglag, (float) variovalue);

/* Load observed xpoints */
dpoints[dpts].x = (double) avglag;
dpoints[dpts].y = (double) variovalue;

xmax = MAX(xmax, dpoints[dpts].x);
ymax = MAX(ymax, dpoints[dpts].y);
xmin = MIN(xmin, dpoints[dpts].x);
ymin = MIN(ymin, dpoints[dpts].y);

dpts++;
}

/* reinitialize values */

Hh = 0;
variosum = (double) 0.0;
variovalue = (double) 0.0;
avglag = (double) 0.0;
}

/* Assign global points */

oxmin = xmin;
cxmax = xmax;
oymin = ymin;
cymax = ymax;

/* Map observed data points to display window */

for (i = 0, dpts = dpts; i < dpts; i++)
{
    opoints[i].x = (int)
        (dpoints[i].x-xmin)*(DISPLAY_WIDTH/((double)(xmax-xmin)));
    opoints[i].y = (int) (DISPLAY_HEIGHT -
        (dpoints[i].y-ymin)*(DISPLAY_HEIGHT/((double)(ymax-ymin))));
}

int lines_intersect(pt1, pt2, pt3, pt4)
dPoint *pt1, *pt2;  /* First line segment */
dPoint *pt3, *pt4;  /* Second line segment */
{
    double a1, a2, b1, b2, c1, c2;  /* Coefficients of line eqns. */
    double r1, r2, r3, r4;  /* 'Sign' values */

double denom, offset, num;  // Intermediate values */

/* Compute a1, b1, c1, where line joining points 1 and 2
 * is "a1 x + b1 y + c1 = 0".
 * */
a1 = pt2->y - pt1->y;
b1 = pt1->x - pt2->x;
c1 = pt2->x * pt1->y - pt1->x * pt2->y;

/* Compute r3 and r4. */
r3 = a1 * pt3->x + b1 * pt3->y + c1;
r4 = a1 * pt4->x + b1 * pt4->y + c1;

/* Check signs of r3 and r4. If both point 3 and point 4 lie on
 * same side of line 1, the line segments do not intersect.
 */
if ( r3 != 0 && r4 != 0 && SAME_SIGNS( r3, r4 ) )
    return (FALSE);

/* Compute a2, b2, c2 */
a2 = pt4->y - pt3->y;
b2 = pt3->x - pt4->x;
c2 = pt4->x * pt3->y - pt3->x * pt4->y;

/* Compute r1 and r2 */
r1 = a2 * pt1->x + b2 * pt1->y + c2;
r2 = a2 * pt2->x + b2 * pt2->y + c2;

/* Check signs of r1 and r2. If both point 1 and point 2 lie
 * on same side of second line segment, the line segments do
 * not intersect.
 */
if ( r1 != 0 && r2 != 0 && SAME_SIGNS( r1, r2 ) )
    return (FALSE);

/* Line segments intersect (assume collinearity to be polygon inclusive) */
return (TRUE);
}

int PtInPoly(poly, pt)
Polygons *poly;
dPoint *pt;
{
    extern int lines_intersect(); /* Function declaration */

    int i;
    int inpoly = FALSE;
    int intersectcnt = 0;
    int istatus;

    dPoint dummypoint;

    /* The rest of the code... */
dummpoint.x = pt->x + 1000.0;
dummpoint.y = pt->y + 100.0;
for (i = 0; i < (poly->nums); i++)
{
    lstatus = (lines_intersect(poly->pt[i],
                               poly->pt[(i < poly->nums-1) ? i+1 : 0],
                               pt, &dummpoint));

    if (lstatus == TRUE)
        intersectcnt++;
}

// check if intersecting line passed an odd number of times through poly */
if (((intersectcnt % 2) != 0)
    inpoly = TRUE;
return inpoly;
}

int LoadObservedFile(item, event)
Panel_item item;
Event *event;
{
    extern int ProcessInputFile();

    FILE *fin;
    Frame frame;
    Frame bfsname;
    Canvas canvas;
    char filename[200];
    int plane;
    int status;

    switch(event_action(event))
    {
    case '\033':
        xv_set(item, PANEL_VALUE, "", NULL);
        return PANEL_NONE;
    case 'r':
    case 'm':
        /* Copy over filename */
        strcpy(filename, (char *) xv_get(item, PANEL_VALUE));
        if (!filename)
            return PANEL_NONE;
    }
/* Load file */

fin = (FILE *) fopen(filename, "r");

if (fin == (FILE *) NULL)
{
    printf("Unable to open file: %s\n", filename);
    return PANEL_NONE;
}

frame = (Frame) xv_get(item, XV_KEY_DATA, FRAME_KEY);
frame = (Frame) xv_get(item, XV_KEY_DATA, BASE_FRAME_KEY);

/* Set frames busy */

xv_set(frame, FRAME_BUSY, TRUE, NULL);

xv_set(bframe, FRAME_LEFT_HEADER, "Loading observed data", NULL);

status = ProcessInputFile(fin);

if (status == FALSE)
{
    printf("error with processing input file\n");
    exit(1);
}

/* Clear busy */

xv_set(frame, FRAME_BUSY, FALSE, NULL);

xv_set(bframe, FRAME_LEFT_HEADER, "", NULL);

break;

default:
    return PANEL_INSERT;
}

int Quit(item, event)
Panel_item item;
Event *event;
{
    Frame frame = (Frame) xv_get(item, XV_KEY_DATA, FRAME_KEY);

    xv_destroy_safe(frame);
}

void SaveFilter(menu, mi)
Menu
Menu_item mi;
{
  FILE *fout;

  int i;

  Frame frame = (Frame) xv_get(menu, XV_KEY_DATA, FRAME_KEY);
  int pln = (int) xv_get(frame, XV_KEY_DATA, PLANE_TYPE_KEY);
  int hvalx = (int) xv_get(frame, XV_KEY_DATA, HVALX_KEY);
  int hvaly = (int) xv_get(frame, XV_KEY_DATA, HVALY_KEY);
  int hrad = (int) xv_get(frame, XV_KEY_DATA, HRAD_KEY);
  int vval = (int) xv_get(frame, XV_KEY_DATA, VVAL_KEY);
  int vtol = (int) xv_get(frame, XV_KEY_DATA, VTOLE_KEY);
  int val = (int) xv_get(frame, XV_KEY_DATA, VVAL_KEY);
  int ttol = (int) xv_get(frame, XV_KEY_DATA, TTOL_KEY);

  char filename[80];
  char planestr[20];

  switch(pln)
  {
    case XY_PLANES:
      sprintf(filename, "%s.%s.%s.%s.%s", "filter", "xy",
              val, vtol, val, ttol);
      break;
    case YZ_PLANES:
      sprintf(filename, "%s.%s.%s.%s.%s", "filter", "yz",
              hvalx, hvaly, hrad, val, ttol);
      break;
    case ZT_PLANES:
      sprintf(filename, "%s.%s.%s.%s.%s", "filter", "zt",
              hvalx, hvaly, hrad, val, ttol);
      break;
    default:
      printf("Unable to save output file...
")
      return;
  }

  if ((fout = (FILE *) fopen(filename, "w")) == (FILE *) NULL)
  {
    printf("Unable to save output file...
")
    return;
  }

  for (i = 0; i < gdts; i++)
  {
    fprintf(fout,
            "%s\t%f\t%f\t%f\n",
            gdt[i].wid,
            gdt[i].x,
            gdt[i].y,
            gdt[i].z,
            gdt[i].t,
            gdt[i].v);
fflush(fout);
}

tclose(fout);
}

void SaveObserved(menu, mi)
{
    FILE *fout;
    Canvas canvas = (Canvas) xv_get(menu, XV_KEY_DATA, CANVAS_KEY);
    Frame frame = (Frame) xv_get(menu, XV_KEY_DATA, FRAME_KEY);

    int lag = (int) xv_get(canvas, XV_KEY_DATA, LAG_KEY);
    int tol = (int) xv_get(canvas, XV_KEY_DATA, TOL_KEY);
    int otype = (int) xv_get(canvas, XV_KEY_DATA, OVAR_TYPE_KEY);
    int ang = (int) xv_get(canvas, XV_KEY_DATA, ANG_KEY);
    int angtol = (int) xv_get(canvas, XV_KEY_DATA, ANG_TOL_KEY);
    int hmax = (int) xv_get(canvas, XV_KEY_DATA, LAG_MAX_KEY);
    int dtype = (int) xv_get(canvas, XV_KEY_DATA, DISPLAY_TYPE_KEY);

    int plan = (int) xv_get(frame, XV_KEY_DATA, PLANE_TYPE_KEY);

    char filename[80];
    char plane[3];
    int i;

    switch(plan)
    {
    case XY_PLANE:
        strcpy(plane, "xy");
        break;
    case YZ_PLANE:
        strcpy(plane, "yz");
        break;
    case ZT_PLANE:
        strcpy(plane, "zt");
        break;
    default:
        printf("Unable to save output file...
");
        return;
    }

    /\* create filename */

    sprintf(filename, "vout.%s.%d.%d", plane, lag, tol);

    if (!(fout = (FILE *) fopen(filename, "w")))
    {
        printf("Unable to save output file...
");
        return;
    }


```c
for (i = 0; i < odatas; i++)
{
    fprintf(fout,
"\x\x\t\x\x\t\x\x\t\n",
odata[i].sh,
odata[i].h,
odata[i].avglag,
odata[i].variovalue);
    fflush(fout);
}
fclose(fout);
}

void ModifyFilterWindow(item, event)
Panel_item item;
Event *event;
{
    Frame frame = (Frame) xv_get(item, XV_KEY_DATA, FRAME_KEY);
    int key = (int) xv_get(item, XV_KEY_DATA, KEY_KEY);
    int value = (int) xv_get(item, PANEL_VALUE);
    xv_set(frame, XV_KEY_DATA, key, value, NULL);
}

void ObservedPlaneSelected(item, value, event)
Panel_item item;
int value;
Event *event;
{
    Frame frame = (Frame) xv_get(item, XV_KEY_DATA, FRAME_KEY);
    xv_set(frame, XV_KEY_DATA, PLANE_TYPE_KEY, value, NULL);
}

void ProcessFilterWindow(item, event)
Panel_item item;
Event *event;
{
    external void FilterWindow();
    Frame frame = (Frame) xv_get(item, XV_KEY_DATA, FRAME_KEY);
    int plane = (int) xv_get(frame, XV_KEY_DATA, PLANE_TYPE_KEY);
    int hvalx = (int) xv_get(frame, XV_KEY_DATA, HVALX_KEY);
    int hvaly = (int) xv_get(frame, XV_KEY_DATA, HVALY_KEY);
    int hrad = (int) xv_get(frame, XV_KEY_DATA, HRAD_KEY);
    int vval = (int) xv_get(frame, XV_KEY_DATA, VVAL_KEY);
    int vtol = (int) xv_get(frame, XV_KEY_DATA, VTOLE_KEY);
    int tval = (int) xv_get(frame, XV_KEY_DATA, TVAL_KEY);
    int ttol = (int) xv_get(frame, XV_KEY_DATA, TTOLE_KEY);
    FilterWindow(frame, plane,
```
hvalx, hvaly, hrad,
  vval, vtol,
tval, ttol);}

void FilterWindow(frame, plane, hvalx, hvaly, hrad, vval, vtol, tval, ttol){
  int plane;
  int hvalx, hvaly, hrad;
  int vval, vtol;
  int tval, ttol;
  
  extern int within();
  extern int range();
  extern void ProcessDistance();

  int i, j;

  xv_set(frame, FRAME_BUSY, TRUE, NULL);

  switch(plane){
    case XY_PLANE: /* Interested in V,T filter */
      for (i = 0, j = 0; i < nv; i++)
      {
        if ((range((double)gvt[i].x, (double)vval, (double)vtol)) &&
            (range((double)gvt[i].t, (double)tval, (double)ttol)))
        {
          strcpy(gdt[j].wid, gvt[i].wid);
          gdt[j].x = gvt[i].x;
          gdt[j].y = gvt[i].y;
          gdt[j].z = gvt[i].z;
          gdt[j].t = gvt[i].t;
          gdt[j].v = gvt[i].v;
          j++;
        }
      }
      gdts = j;
      break;
    case YZ_PLANE: /* Interested in H,T filter */
      for (i = 0, j = 0; i < nv; i++)
      {
        if ((within((double)gvt[i].x, (double)gvt[i].y,
                     (double)hvalx, (double)hvaly, (double)hrad)) &&
            (range((double)gvt[i].t, (double)tval, (double)ttol)))
        {
          strcpy(gdt[j].wid, gvt[i].wid);
          gdt[j].x = gvt[i].x;
          gdt[j].y = gvt[i].y;
          gdt[j].z = gvt[i].z;
          gdt[j].t = gvt[i].t;
          gdt[j].v = gvt[i].v;
          j++;
        }
      }
      break;
  }
}

}
}
gdts = j;
break;

case ZT_PLANE:    /* Interested in H,V filter */

for (i = 0, j = 0; i < gdts; i++)
{
    if ((within((double)gvt[i].x, (double)gvt[i].y,
                (double)hvalx, (double)hvaly, (double)hrad)) &&
        (range((double)gvt[i].x, (double)vval, (double)vtol)))
    {
        strcpy(gdt[j].wid, gvt[i].wid);
gt[j].x = gvt[i].x;
gt[j].y = gvt[i].y;
gt[j].z = gvt[i].z;
gt[j].t = gvt[i].t;
gt[j].v = gvt[i].v;
j++;
    }
}

gdts = j;
break;

default:

break;
}

/* Process distance matrix based on plane and filtered data */

ProcessDistance(plane);

xv_set(frame, FRAME_BUSY, FALSE, NULL);
}

int within(x, y, cx, cy, rad)
double x,y;
double cx,cy;
double rad;
{
    double dx, dy;
double distance;

    /* trivial check.. if val and tolerance are both zero */
    /* assume no filter.. i.e. return TRUE */

    if ((cx == (double)0.0) && (cy == (double)0.0) && (rad == (double)0.0))
        return TRUE;

    dx = x - cx;
dy = y - cy;
distance = (double) sqrt(dx*dx+dy*dy);
if (distance <= rad)
    return TRUE;

return FALSE;
}

int range(sample, val, tol)
{
    double sample;
    double val;
    double tol;
    /* trivial check.. if val and tolerance are both zero */
    /* assume no filter.. i.e. return TRUE */
    if ((val == (double) 0.0) && (tol == (double) 0.0))
        return TRUE;
    if ((sample >= (val - tol)) && (sample <= (val + tol)))
        return TRUE;

    return FALSE;
}
Appendix F

Input Four Dimensional Kriging Parameter File

% Debug flag
Debug 0

% Input/Output information
InputFilename "../input/no3.test.input"
OutputFilename "../outputora/no3.vol"

% Integer output scale factor
VScale 1000

% Indicate 0 || 1 || 2 for non-log log or both value calculation
Logscale 2

% Single point flag = 1 indicates krige only at particular
% Point, otherwise krige grid given by grid dimensions
SinglePoint Flag 0 x "2" y "12" z "260" t "10"

% Krige grid
% Given as upper left hand coordinate, right hand coordinate, rows, cols
% temporal: time (t+window; increment wincr)
Origin x "-20" y "20" z "280" t "0"
Size cols "40" rows "-30" depth "-40" window "20" wincr "2"

% Theoretical Variogram information (spherical | exponential | gaussian)
% CO = Nugget Effect
% a = range
% C1 = -> CO + C1 = sill
% EXPONENTIAL 1
% SPHERICAL 2
% GAUSSIAN 3

% Isotropic indicator: if true then anisotropic magnitude and directional
% indicators (Amax, Amin) are not needed

% Rotational directions indicate rotational modification to anisotropic
% structure (if any)

VariogramX type 1 c0 "0.0" a "20.0" c1 "10.0"
VariogramY type 1 c0 "0.0" a "10.0" c1 "10.0"
VariogramZ type 1 c0 "0.0" a "5.0" c1 "10.0"
VariogramT type 1 c0 "0.0" a "10.0" c1 "10.0"
AnisoRotDirection x 0 y 0 z 0
BIBLIOGRAPHY


